NAG Library Routine Document

E04KDF

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

E04KDF 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 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 E04KDF(N, FUNCT, MONIT, IPRINT, MAXCAL, ETA, XTOL, DELTA,
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
 double precision  ETA, XTOL, DELTA, STEPMX, BL(N), BU(N), X(N),
1                  HESL(LH), HESD(N), F, G(N), W(LW)
 EXTERNAL          FUNCT, MONIT
```

3 Description

E04KDF is applicable to problems of the form:

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

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. You must supply a starting point, and a FUNCT to calculate the value of $F(x)$ and its first derivatives $\dfrac{\partial F}{\partial x_j}$ at any point x.

A typical iteration starts at the current point x where n_z (say) variables are free from their bounds. The vector g_z, whose elements are the derivatives of $F(x)$ with respect to the free variables, is known. The matrix of second derivatives with respect to the free variables, H, is estimated by finite differences. (Note that g_z and H are both of 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 the 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

convergence criteria are satisfied. If at this point there are no negative or near-zero Lagrange multiplier estimates, the process is terminated.

If you specify that the problem is unconstrained, E04KDF 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 E04KDF 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 a specified point. (However, if you do not wish to calculate F or its first derivatives at a particular x, there is the option of setting a parameter to cause E04KDF to terminate immediately.)

The specification of FUNCT is:

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

1: IFLAG – INTEGER *Input/Output*

On entry: will have been set to 1 or 2. The value 1 indicates that only the first derivatives of F need be supplied, and the value 2 indicates that both F itself and its first derivatives must be calculated.

On exit: if it is not possible to evaluate F or its first derivatives at the point given in XC (or if it is wished to stop the calculations for any other reason) you should reset IFLAG to a negative number and return control to E04KDF. E04KDF will then terminate immediately, with IFAIL set to your setting of IFLAG.

2: N – INTEGER *Input*

On entry: the number n of variables.

3: XC(N) – ***double precision*** array *Input*

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

4: FC – ***double precision*** *Output*

On exit: unless IFLAG $= 1$ on entry or IFLAG is reset, FUNCT must set FC to the value of the objective function F at the current point x.

5: GC(N) – ***double precision*** array *Output*

On exit: unless FUNCT resets IFLAG, it 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) – ***double precision*** array *Workspace*
9: LW – INTEGER *Input*

FUNCT is called with the same parameters IW, LIW, W, LW as for E04KDF. 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, you **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 subroutine which calls E04KDF. However, because of the danger of mistakes in partitioning, it is recommended that you should pass information to FUNCT via COMMON global variables and not use IW or W at all. In any case you 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 E04KDF 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 E04KDF.

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

If IPRINT ≥ 0, you must supply MONIT which is suitable for monitoring the minimization process. MONIT must not change the values of any of its parameters.

If IPRINT < 0, a MONIT with the correct parameter list must still be supplied, although it will not be called.

The specification of MONIT 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
double precision 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) – ***double precision*** array *Input*

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

3: FC – ***double precision*** *Input*

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

4: GC(N) – ***double precision*** 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(j) is positive, its value gives the position of x_j in the sequence of free variables.

6: GPJNRM – ***double precision*** *Input*

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

7: COND – ***double precision*** *Input*

On entry: the ratio of the largest to the smallest elements of the diagonal factor D of the approximated projected Hessian matrix. 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: specifies .TRUE. or .FALSE. according to whether or not the approximation to the second derivative matrix for the current subspace, H, is positive-definite.

9: NITER – INTEGER *Input*

On entry: the number of iterations (as outlined in Section 3) which have been performed by E04KDF so far.

10: NF – INTEGER *Input*

On entry: the number of evaluations of $F(x)$ so far, i.e., the number of calls of FUNCT with IFLAG set to 2. Each such call of FUNCT also calculates the first derivatives of F. (In addition to these calls monitored by NF, FUNCT is called with IFLAG set to 1 not more than N times per iteration.)

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

As in FUNCT, these parameters correspond to the parameters IW, LIW, W, LW of E04KDF. They are included in MONIT's parameter list primarily for when E04KDF is called by other library routines.

MONIT must be declared as EXTERNAL in the (sub)program from which E04KDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

You should normally print FC, GPJNRM and COND to be able to compare the quantities mentioned in Section 7. It is usually helpful to examine XC, POSDEF and NF too.

4: IPRINT – INTEGER *Input*

On entry: the frequency with which MONIT is to be called.

IPRINT > 0
MONIT is called once every IPRINT iterations and just before exit from E04KDF.

IPRINT $= 0$
MONIT is just called at the final point.

IPRINT < 0
MONIT is not called at all.

IPRINT should normally be set to a small positive number.

Suggested value: IPRINT $= 1$.

5: MAXCAL – INTEGER *Input*

On entry: the maximum permitted number of evaluations of $F(x)$, i.e., the maximum permitted number of calls of FUNCT with IFLAG set to 2. It should be borne in mind that, in addition to the calls of FUNCT which are limited directly by MAXCAL, there will be calls of FUNCT (with IFLAG set to 1) to evaluate only first derivatives.

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

Constraint: MAXCAL ≥ 1.

6: ETA – ***double precision*** *Input*

On entry: every iteration of E04KDF 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 large values (say, 0.9).

Although accurate linear minimizations will generally reduce the number of iterations (and hence the number of calls of FUNCT to estimate the second derivatives), they will tend to increase the number of calls of FUNCT needed for each linear minimization. On balance, it is usually more efficient to perform a low accuracy linear minimization when n is small and a high accuracy minimization when n is large.

Suggested value:

ETA $= 0.5$ if $1 < n < 10$;
ETA $= 0.1$ if $10 \leq n \leq 20$;
ETA $= 0.01$ if $n > 20$.

If N $= 1$, ETA should be set to 0.0 (also when the problem is effectively one-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.

7: XTOL – ***double precision*** *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 before a normal exit, is such that $\|x_{\text{sol}} - x_{\text{true}}\| <$ XTOL $\times (1.0 + \|x_{\text{true}}\|)$ 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 five decimal places. (For further details see Section 7.)

If the problem is scaled as described in Section 8.2 and ϵ is the ***machine precision***, then $\sqrt{\epsilon}$ is probably the smallest reasonable choice for XTOL. This is because, normally, to machine accuracy, $F\left(x + \sqrt{\epsilon} e_j\right) = F(x)$, for any j where e_j is the jth column of the identity matrix. If you set XTOL to 0.0 (or any positive value less than ϵ), E04KDF will use $10.0 \times \sqrt{\epsilon}$ instead of XTOL.

Suggested value: XTOL $= 0.0$.

Constraint: XTOL ≥ 0.0.

8: DELTA – ***double precision*** *Input*

On entry: the differencing interval to be used for approximating the second derivatives of $F(x)$. Thus, for the finite difference approximations, the first derivatives of $F(x)$ are evaluated at points which are DELTA apart. If ϵ is the ***machine precision***, then $\sqrt{\epsilon}$ will usually be a suitable setting for DELTA. If you set DELTA to 0.0 (or to any positive value less than ϵ), E04KDF will automatically use $\sqrt{\epsilon}$ as the differencing interval.

Suggested value: DELTA $= 0.0$.

Constraint: DELTA ≥ 0.0.

9: STEPMX – ***double precision*** *Input*

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

E04KDF 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, E04KDF 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: indicates 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 you are 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 for consistency with other routines. In E04KDF it produces the same effect as IBOUND $= 1$.)

Constraint: $0 \leq$ IBOUND ≤ 4.

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

On entry: the fixed lower bounds l_j.

If IBOUND is set to 0, you 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, you must set BL(1) to l_1; E04KDF will then set the remaining elements of BL equal to BL(1).

If IBOUND is set to 1, 2 or 4, BL will be initialized by E04KDF.

On exit: the lower bounds actually used by E04KDF, e.g., If IBOUND $= 2$, BL(1) $=$ BL(2) $= \cdots =$ BL(n) $= 0.0$.

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

On entry: the fixed upper bounds u_j.

If IBOUND is set to 0, you 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, you must set BU(1) to u_1; E04KDF will then set the remaining elements of BU equal to BU(1).

If IBOUND is set to 1, 2 or 4, BU will be initialized by E04KDF.

On exit: the upper bounds actually used by E04KDF, e.g., if IBOUND $= 2$, BU(1) $=$ BU(2) $= \cdots =$ BU(n) $= 10^6$.

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

On entry: X(j) must be set to a guess at the jth component of the position of the 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) – ***double precision*** 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 estimated 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 estimated second derivative matrix H. The elements of HESD are useful for deciding whether to accept the results produced by E04KDF (see Section 7).

15: LH – INTEGER *Input*

On entry: the dimension of the array HESL as declared in the (sub)program from which E04KDF is called.

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

16: HESD(N) – ***double precision*** 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 estimated 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 estimated second derivative matrix H. The elements of HESD are useful for deciding whether to accept the results produced by E04KDF (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 – ***double precision*** *Output*

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

19: G(N) – ***double precision*** 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 *Communication Array*

21: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04KDF is called.

Constraint: LIW ≥ 2.

22: W(LW) – ***double precision*** array *Communication Array*

23: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04KDF 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. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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).

Note: E04KDF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL < 0

A negative value of IFAIL indicates an exit from E04KDF because you have set IFLAG negative in FUNCT. The value of IFAIL will be the same as your 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 DELTA < 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 you have set XTOL or DELTA to 0.0, E04KDF uses the default values 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 recognize 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. You should investigate as for IFAIL $=$ 3.

The values IFAIL $=$ 2, 3 or 5 may also be caused by mistakes in FUNCT, 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 E04KDF when $H^{(k)}$ is positive-definite and when (B1, B2 and B3) or B4 hold, where

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

(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 $\left\{F\left(x^{(k)}\right)\right\}$ converges to $F(x_{\text{sol}})$ at a superlinear or fast linear rate,

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

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

then it is almost certain that x_{sol} is a close approximation to the position of a minimum. When (ii) is true, then usually $F(x_{\text{sol}})$ is a close approximation to $F(x_{\text{true}})$. 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 E04KDF is $\frac{n_z^3}{6} + O\left(n_z^2\right)$. In addition, each iteration makes n_z calls of FUNCT (with IFLAG set to 1) in approximating the projected Hessian matrix, and at least one other call of FUNCT (with IFLAG set to 2). So, unless $F(x)$ and its first derivatives can be evaluated very quickly, the run time will be dominated by the time spent in FUNCT.

8.2 Scaling

Ideally, the problem should be scaled so that, at the solution, $F(x)$ and the corresponding values of 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 you 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 E04KDF 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 estimated 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 E04KDF 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 UNCKDF(N,FUNCT,MONIT,IPRINT,MAXCAL,ETA,XTOL,DELTA,
     *                  STEPMX,X,HESL,LH,HESD,F,G,IWORK,LIWORK,WORK,
     *                  LWORK,IFAIL)
C
C     A ROUTINE TO APPLY E04KDF 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 E04KDF 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 E04KDF 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 E04KDF.
C

C     .. Parameters ..
      INTEGER NOUT
      PARAMETER (NOUT=6)
C     .. Scalar Arguments ..
      real DELTA, 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, MONIT
C     .. Local Scalars ..
      INTEGER IBOUND, J, JBL, JBU, NH
      LOGICAL TOOBIG
C     .. External Subroutines ..
      EXTERNAL E04KDF
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 E04KDF(N,FUNCT,MONIT,IPRINT,MAXCAL,ETA,XTOL,DELTA,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

$$\begin{array}{rcrcl} 1 & \leq & x_1 & \leq & 3 \\ -2 & \leq & x_2 & \leq & 0 \\ 1 & \leq & x_4 & \leq & 3, \end{array}$$

starting from the initial guess $(3, -1, 0, 1)$. Before calling E04KDF, the program calls E04HCF to check the first derivatives calculated by FUNCT.

9.1 Program Text

```
*     E04KDF 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 ..
      DOUBLE PRECISION DELTA, ETA, F, STEPMX, XTOL
      INTEGER          IBOUND, IFAIL, IPRINT, J, MAXCAL
*     .. Local Arrays ..
      DOUBLE PRECISION BL(N), BU(N), G(N), HESD(N), HESL(LH), W(LW),
     +                 X(N)
      INTEGER          ISTATE(N), IW(LIW)
*     .. External Subroutines ..
      EXTERNAL         E04HCF, E04KDF, FUNCT, MONIT
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04KDF Example Program Results'
*     Check FUNCT by calling E04HCF at an arbitrary point. Since E04HCF
*     only checks the derivatives calculated when IFLAG = 2, a separate
*     program should be run before using E04HCF or E04KDF to check that
*     FUNCT gives the same values for the GC(J) when IFLAG is set to 1
*     as when IFLAG is set to 2.
      X(1) = 1.46D0
      X(2) = -0.82D0
      X(3) = 0.57D0
      X(4) = 1.21D0
      IFAIL = 1
*
      CALL E04HCF(N,FUNCT,X,F,G,IW,LIW,W,LW,IFAIL)
*
      IF (IFAIL.NE.0) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99994) ' ** E04HCF returned with IFAIL = ', IFAIL
         GO TO 20
      END IF
*
*     Continue setting parameters for E04KDF
*     * Set IPRINT to 1 to obtain output from MONIT at each iteration *
      IPRINT = -1
      MAXCAL = 50*N
      ETA = 0.5D0
*     Set XTOL and DELTA to zero so that E04KDF will use the default
*     values
      XTOL = 0.0D0
      DELTA = 0.0D0
*     We estimate that the minimum will be within 4 units of the
*     starting point
      STEPMX = 4.0D0
      IBOUND = 0
      BL(1) = 1.0D0
      BU(1) = 3.0D0
```

```
      BL(2) = -2.0D0
      BU(2) = 0.0D0
*     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) = -1.0D6
      BU(3) = 1.0D6
      BL(4) = 1.0D0
      BU(4) = 3.0D0
*     Set up starting point
      X(1) = 3.0D0
      X(2) = -1.0D0
      X(3) = 0.0D0
      X(4) = 1.0D0
      IFAIL = 1
*
      CALL E04KDF(N,FUNCT,MONIT,IPRINT,MAXCAL,ETA,XTOL,DELTA,STEPMX,
     +            IBOUND,BL,BU,X,HESL,LH,HESD,ISTATE,F,G,IW,LIW,W,LW,
     +            IFAIL)
*
      IF (IFAIL.GE.0) THEN
         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,99998) 'at the point', (X(J),J=1,N)
            WRITE (NOUT,*)
     +        'The corresponding (machine dependent) gradient is'
            WRITE (NOUT,99997) (G(J),J=1,N)
            WRITE (NOUT,99996) 'ISTATE contains', (ISTATE(J),J=1,N)
            WRITE (NOUT,99995) 'and HESD contains', (HESD(J),J=1,N)
         END IF
      ELSE
         WRITE (NOUT,*)
         WRITE (NOUT,99994) ' ** E04KDF returned with IFAIL = ', IFAIL
      END IF
   20 CONTINUE
*
99999 FORMAT (1X,A,I3,A)
99998 FORMAT (1X,A,4F12.4)
99997 FORMAT (24X,1P,4E12.3)
99996 FORMAT (1X,A,4I5)
99995 FORMAT (1X,A,4E12.4)
99994 FORMAT (1X,A,I5)
      END
*
      SUBROUTINE FUNCT(IFLAG,N,XC,FC,GC,IW,LIW,W,LW)
*     Routine to evaluate objective function and its 1st derivatives.
*     A COMMON variable could be updated here to count the number of
*     calls of FUNCT with IFLAG = 1 (since NF in MONIT only counts
*     calls with IFLAG = 2)
*     .. Scalar Arguments ..
      DOUBLE PRECISION FC
      INTEGER          IFLAG, LIW, LW, N
*     .. Array Arguments ..
      DOUBLE PRECISION GC(N), W(LW), XC(N)
      INTEGER          IW(LIW)
*     .. Executable Statements ..
      IF (IFLAG.NE.1) FC = (XC(1)+10.0D0*XC(2))**2 + 5.0D0*(XC(3)-XC(4))
     +                     **2 + (XC(2)-2.0D0*XC(3))**4 + 10.0D0*(XC(1)
     +                     -XC(4))**4
      GC(1) = 2.0D0*(XC(1)+10.0D0*XC(2)) + 40.0D0*(XC(1)-XC(4))**3
      GC(2) = 20.0D0*(XC(1)+10.0D0*XC(2)) + 4.0D0*(XC(2)-2.0D0*XC(3))**3
      GC(3) = 10.0D0*(XC(3)-XC(4)) - 8.0D0*(XC(2)-2.0D0*XC(3))**3
      GC(4) = 10.0D0*(XC(4)-XC(3)) - 40.0D0*(XC(1)-XC(4))**3
      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 ..
      DOUBLE PRECISION COND, FC, GPJNRM
      INTEGER          LIW, LW, N, NF, NITER
      LOGICAL          POSDEF
*     .. Array Arguments ..
      DOUBLE PRECISION 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.0.0D0) THEN
         IF (COND.GT.1.0D6) 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 be used in conjunction with either of the routines
*           E04KDF or  E04LBF
            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

```
E04KDF 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   1.177E-09   5.907E+00
ISTATE contains   -2    1    2   -2
and HESD contains  0.2098E+03  0.4738E+02  0.4552E+02  0.0000E+00
```