E04KZF – NAG Fortran Library Routine Document

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

E04KZF is an easy-to-use modified-Newton algorithm for finding a minimum of a function $F(x_1, x_2, \ldots, x_n)$, subject to fixed upper and lower bounds on the independent variables $x_1, x_2, \ldots, x_n$, when first derivatives of F are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

2 Specification

```
      SUBROUTINE E04KZF(N, IBOUND, FUNCT2, BL, BU, X, F, G, IW, LIW, W,
     1                  LW, IUSER, USER, IFAIL)
      INTEGER           N, IBOUND, IW(LIW), LIW, LW, IUSER(*), IFAIL
      real              BL(N), BU(N), X(N), F, G(N), W(LW), USER(*)
      EXTERNAL          FUNCT2
```

3 Description

This routine is applicable to problems of the form:

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

when first derivatives are known.

Special provision is made for problems which actually have no bounds on the x_j, problems which have only non-negativity bounds, and problems in which $l_1 = l_2 = \ldots = l_n$ and $u_1 = u_2 = \ldots = u_n$. The user must supply a subroutine to calculate the values of $\mathrm{F}(x)$ and its first derivatives at any point x.

From a starting point supplied by the user there is generated, on the basis of estimates of the gradient of the curvature of $\mathrm{F}(x)$, a sequence of feasible points which is intended to converge to a local minimum of the constrained function.

4 References

[1] 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: $\mathrm{N} \ge 1$.

2: IBOUND — INTEGER *Input*

On entry: indicates 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 user will be supplying all the l_j and u_j individually.

IBOUND = 1

if there are no bounds on any x_j.

IBOUND = 2

if all the bounds are of the form $0 \leq x_j$.

IBOUND = 3

if $l_1 = l_2 = \ldots = l_n$ and $u_1 = u_2 = \ldots = u_n$.

Constraint: $0 \leq$ IBOUND ≤ 3.

3: FUNCT2 — SUBROUTINE, supplied by the user. *External Procedure*

This routine must be supplied by the user to calculate the values of the function F(x) and its first derivatives $\frac{\partial F}{\partial x_j}$ at any point x. It should be tested separately before being used in conjunction with E04KZF (see the the Chapter Introduction).

Its specification is:

```
SUBROUTINE FUNCT2(N, XC, FC, GC, IUSER, USER)
INTEGER          N, IUSER(*)
real             XC(N), FC, GC(N), USER(*)
```

1: N — INTEGER *Input*

On entry: the number n of variables.

2: XC(N) — ***real*** array *Input*

On entry: the point x at which the function and derivatives are required.

3: FC — ***real*** *Output*

On exit: the value of the function F at the current point x,

4: GC(N) — ***real*** array *Output*

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

5: IUSER(*) — INTEGER array *User Workspace*

6: USER(*) — ***real*** array *User Workspace*

FUNCT2 is called from E04KZF with the parameters IUSER and USER as supplied to E04KZF. The user is free to use the arrays IUSER and USER to supply information to FUNCT2 as an alternative to using COMMON.

FUNCT2 must be declared as EXTERNAL in the (sub)program from which E04KZF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

4: BL(N) — ***real*** array *Input/Output*

On entry: the 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 a particular x_j, the corresponding BL(j) should be set to -10^6.)

If IBOUND is set to 3, the user must set BL(1) to l_1; E04KZF will then set the remaining elements of BL equal to BL(1).

On exit: the lower bounds actually used by E04KZF.

5: BU(N) — ***real*** array *Input/Output*

On entry: the 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 a particular x_j, the corresponding BU(j) should be set to 10^6.)

If IBOUND is set to 3, the user must set BU(1) to u_1; E04KZF will then set the remaining elements of BU equal to BU(1).

On exit: the upper bounds actually used by E04KZF.

6: X(N) — ***real*** array *Input/Output*

On entry: X(j) must be set to a guess at the jth component of the position of the minimum, for $j = 1, 2, \ldots, n$. The routine checks the gradient at the starting point, and is more likely to detect any error in the user's programming if the initial X(j) are non-zero and mutually distinct.

On exit: the lowest point found during the calculations of the position of the minimum.

7: F — ***real*** *Output*

On exit: the value of $F(x)$ corresponding to the final point stored in X.

8: G(N) — ***real*** array *Output*

On exit: the value of $\frac{\partial F}{\partial x_j}$ corresponding to the final point stored in X, for $j = 1, 2, \ldots, n$; the value of G(j) for variables not on a bound should normally be close to zero.

9: IW(LIW) — INTEGER array *Workspace*

10: LIW — INTEGER *Input*

On entry: the length of IW, as declared in the (sub)program from which E04KZF is called.

Constraint: LIW $\geq$ N + 2.

11: W(LW) — ***real*** array *Workspace*

12: LW — INTEGER *Input*

On entry: the length of W, as declared in the (sub)program from which E04KZF is called.

Constraint: LW $\geq$ max(N$\times$(N+7),10).

13: IUSER($*$) — INTEGER array *User Workspace*

Note: the dimension of the array IUSER must be at least 1.

IUSER is not used by E04KZF, but is passed directly to FUNCT2 and may be used to pass information to those routines.

14: USER($*$) — ***real*** array *User Workspace*

Note: the dimension of the array USER must be at least 1.

USER is not used by E04KZF, but is passed directly to FUNCT2 and may be used to pass information to those routines.

15: 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 or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL $\neq$ 0 on exit, users are recommended to set IFAIL to -1 before entry. **It is then essential to test the value of IFAIL on exit**. To suppress the output of an error message when soft failure occurs, set IFAIL to 1.

6 Error Indicators and Warnings

Errors or warnings specified by the routine:

IFAIL = 1

On entry, N < 1,
or IBOUND < 0,
or IBOUND > 3,
or IBOUND = 0 and BL(j) > BU(j) for some j,
or IBOUND = 3 and BL(1) > BU(1),
or LIW < N + 2,
or LW < max(10,N× (N+7)).

IFAIL = 2

There has been a large number of function evaluations, yet the algorithm does not seem to be converging. The calculations can be restarted from the final point held in X. The error 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 and the algorithm has failed.

IFAIL = 4

Not used. (This value of the parameter is included to make the significance of IFAIL = 5 etc. consistent in the easy-to-use routines.)

IFAIL = 5, 6, 7 and 8

There is some doubt about whether the point x found by E04KZF is a minimum. The degree of confidence in the result decreases as IFAIL increases. Thus, when IFAIL = 5 it is probable that the final x gives a good estimate of the position of a minimum, but when IFAIL = 8 it is very unlikely that the routine has found a minimum.

IFAIL = 9

In the search for a minimum, the modulus of one of the variables has become very large ($\sim 10^6$). This indicates that there is a mistake in FUNCT2, that the user's problem has no finite solution, or that the problem needs rescaling (see Section 8).

IFAIL = 10

It is very likely that the user has made an error in forming the gradient.

If the user is dissatisfied with the result (e.g., because IFAIL = 5, 6, 7 and 8), 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. If persistent trouble occurs and it is possible to calculate second derivatives it may be advisable to change to a routine which uses second derivatives (see the Chapter Introduction).

7 Accuracy

When a successful exit is made then, for a computer with a mantissa of t decimals, one would expect to get about $t/2-1$ decimals accuracy in x and about $t-1$ decimals accuracy in F, provided the problem is reasonably well scaled.

8 Further Comments

The number of iterations required depends on the number of variables, the behaviour of $F(x)$ and the distance of the starting point from the solution. The number of operations performed in an iteration of E04KZF is roughly proportional to $n^3 + O(n^2)$. In addition, each iteration makes at least $m+1$ calls of FUNCT2 where m is the number of variables not fixed on bounds. So unless $F(x)$ and the gradient vector can be evaluated very quickly, the run time will be dominated by the time spent in FUNCT2.

Ideally the problem should be scaled so that at the solution the value of $F(x)$ and the corresponding values of $x_1, x_2, \ldots, x_n$ are in the range $(-1,+1)$, and so that at points a unit distance away from the solution, F is approximately a unit value greater than at the minimum. 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 E04KZF will take less computer time.

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

$$\begin{array}{rcccl} 1 & \le & x_1 & \le & 3 \\ -2 & \le & x_2 & \le & 0 \\ 1 & \le & x_4 & \le & 3 \end{array}$$

starting from the initial guess (3, −1, 0, 1). (In practice, it is worth trying to make FUNCT2 as efficient as possible. This has not been done in the example program for reasons of clarity.)

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.

```
*     E04KZF Example Program Text.
*     Mark 18 Release. NAG Copyright 1997.
*     .. Parameters ..
      INTEGER          N, LIW, LW
      PARAMETER        (N=4,LIW=N+2,LW=N*(N+7))
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Local Scalars ..
      real             F
      INTEGER          IBOUND, IFAIL, J
*     .. Local Arrays ..
      real             BL(N), BU(N), G(N), USER(1), W(LW), X(N)
      INTEGER          IUSER(1), IW(LIW)
*     .. External Subroutines ..
      EXTERNAL         E04KZF, FUNCT2
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04KZF Example Program Results'
      X(1) = 3.0e0
      X(2) = -1.0e0
      X(3) = 0.0e0
      X(4) = 1.0e0
      IBOUND = 0
      BL(1) = 1.0e0
      BU(1) = 3.0e0
      BL(2) = -2.0e0
```

```
      BU(2) = 0.0e0
*
*     X(3) is unconstrained, 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
      IFAIL = 1
*
      CALL E04KZF(N,IBOUND,FUNCT2,BL,BU,X,F,G,IW,LIW,W,LW,IUSER,USER,
     +            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,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)
      END IF
      STOP
*
99999 FORMAT (1X,A,I3,A)
99998 FORMAT (1X,A,4F12.4)
99997 FORMAT (13X,4e12.4)
      END
*
      SUBROUTINE FUNCT2(N,XC,FC,GC,IUSER,USER)
*     Routine to evaluate objective function and its 1st derivatives.
*     .. Scalar Arguments ..
      real              FC
      INTEGER           N
*     .. Array Arguments ..
      real              GC(N), USER(*), XC(N)
      INTEGER           IUSER(*)
*     .. Local Scalars ..
      real              X1, X2, X3, X4
*     .. Executable Statements ..
      X1 = XC(1)
      X2 = XC(2)
      X3 = XC(3)
      X4 = XC(4)
      FC = (X1+10.0e0*X2)**2 + 5.0e0*(X3-X4)**2 + (X2-2.0e0*X3)**4 +
     +     10.0e0*(X1-X4)**4
      GC(1) = 2.0e0*(X1+10.0e0*X2) + 40.0e0*(X1-X4)**3
      GC(2) = 20.0e0*(X1+10.0e0*X2) + 4.0e0*(X2-2.0e0*X3)**3
      GC(3) = 10.0e0*(X3-X4) - 8.0e0*(X2-2.0e0*X3)**3
      GC(4) = -10.0e0*(X3-X4) - 40.0e0*(X1-X4)**3
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
E04KZF Example Program Results

Error exit type  5 - 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
              0.2953E+00 -0.5872E-09  0.1177E-08  0.5907E+01
```
