NAG Library Routine Document

E04CBF

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

E04CBF minimizes a general function $F(\mathbf{x})$ of n independent variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ by the Nelder and Mead simplex method (see Nelder and Mead (1965)). Derivatives of the function need not be supplied.

2 Specification

```
SUBROUTINE E04CBF(N, X, F, TOLF, TOLX, FUNCT, MONIT, MAXCAL, IUSER,
1                 RUSER, IFAIL)
INTEGER           N, MAXCAL, IUSER(*), IFAIL
double precision  X(N), F, TOLF, TOLX, RUSER(*)
EXTERNAL          FUNCT, MONIT
```

3 Description

E04CBF finds an approximation to a minimum of a function F of n variables. You must supply a subroutine to calculate the value of F for any set of values of the variables.

The method is iterative. A simplex of $n+1$ points is set up in the n-dimensional space of the variables (for example, in 2 dimensions the simplex is a triangle) under the assumption that the problem has been scaled so that the values of the independent variables at the minimum are of order unity. The starting point you have provided is the first vertex of the simplex, the remaining n vertices are generated by E04CBF. The vertex of the simplex with the largest function value is reflected in the centre of gravity of the remaining vertices and the function value at this new point is compared with the remaining function values. Depending on the outcome of this test the new point is accepted or rejected, a further expansion move may be made, or a contraction may be carried out. See Nelder and Mead (1965) and Parkinson and Hutchinson (1972) for more details. When no further progress can be made the sides of the simplex are reduced in length and the method is repeated.

The method can be slow, but computational bottlenecks have been reduced following Singer and Singer (2004). However, E04CBF is robust, and therefore very useful for functions that are subject to inaccuracies.

There are the following options for successful termination of the method: based only on the function values at the vertices of the current simplex (see (1)); based only on a volume ratio between the current simplex and the initial one (see (2)); or based on which one of the previous two tests passes first. The volume test may be useful if F is discontinuous, while the function-value test should be sufficient on its own if F is continuous.

4 References

Nelder J A and Mead R (1965) A simplex method for function minimization *Comput. J.* **7** 308–313

Parkinson J M and Hutchinson D (1972) An investigation into the efficiency of variants of the simplex method *Numerical Methods for Nonlinear Optimization* (ed F A Lootsma) Academic Press

Singer S and Singer S (2004) Efficient Implementation of the Nelder–Mead Search Algorithm *Appl. Num. Anal. Comp. Math.* **1(3)** 524–534

5 Parameters

1: N – INTEGER *Input*

On entry: n, the number of variables.

Constraint: N ≥ 1.

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

On entry: a guess at the position of the minimum. Note that the problem should be scaled so that the values of the X(i) are of order unity.

On exit: the value of **x** corresponding to the function value in F.

3: F – ***double precision*** *Output*

On exit: the lowest function value found.

4: TOLF – ***double precision*** *Input*

On entry: the error tolerable in the function values, in the following sense. If f_i, for $i = 1, 2, \ldots, n+1$, are the individual function values at the vertices of the current simplex, and if f_m is the mean of these values, then you can request that E04CBF should terminate if

$$\sqrt{\frac{1}{n+1}\sum_{i=1}^{n+1}(f_i - f_m)^2} < \text{TOLF}. \tag{1}$$

You may specify TOLF $= 0$ if you wish to use only the termination criterion (2) on the spatial values: see the description of TOLX.

Constraint: TOLF must be greater than or equal to the ***machine precision*** (see Chapter X02), or if TOLF equals zero then TOLX must be greater than or equal to the ***machine precision***.

5: TOLX – ***double precision*** *Input*

On entry: the error tolerable in the spatial values, in the following sense. If LV denotes the 'linearized' volume of the current simplex, and if LV_{init} denotes the 'linearized' volume of the initial simplex, then you can request that E04CBF should terminate if

$$\frac{LV}{LV_{\text{init}}} < \text{TOLX}. \tag{2}$$

You may specify TOLX $= 0$ if you wish to use only the termination criterion (1) on function values: see the description of TOLF.

Constraint: TOLX must be greater than or equal to the ***machine precision*** (see Chapter X02), or if TOLX equals zero then TOLF must be greater than or equal to the ***machine precision***.

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

FUNCT must evaluate the function F at a specified point. It should be tested separately before being used in conjunction with E04CBF.

The specification of FUNCT is:

```
SUBROUTINE FUNCT(N, XC, FC, IUSER, RUSER)
INTEGER           N, IUSER(*)
double precision  XC(N), FC, RUSER(*)
```

1: N – INTEGER *Input*

On entry: n, the number of variables.

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

On entry: the point at which the function value is required.

3: FC – ***double precision*** *Output*

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

4: IUSER(∗) – INTEGER array *User Workspace*
5: RUSER(∗) – ***double precision*** array *User Workspace*

FUNCT is called from E04CBF with the parameters IUSER and RUSER as supplied to E04CBF. You are free to use the arrays IUSER and RUSER to supply information to FUNCT.

FUNCT must be declared as EXTERNAL in the (sub)program from which E04CBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

MONIT is called once every iteration in E04CBF. It can be used to print out the current values of any selection of its parameters but must not be used to change the values of the parameters.

The specification of MONIT is:

```
      SUBROUTINE MONIT(FMIN, FMAX, SIM, N, NCALL, SERROR, VRATIO, IUSER,
     1                 RUSER)
      INTEGER           N, NCALL, IUSER(*)
      double precision  FMIN, FMAX, SIM(N+1,N), SERROR, VRATIO, RUSER(*)
```

1: FMIN – ***double precision*** *Input*

On entry: the smallest function value in the current simplex.

2: FMAX – ***double precision*** *Input*

On entry: the largest function value in the current simplex.

3: SIM(N + 1,N) – ***double precision*** array *Input*

On entry: the $n + 1$ position vectors of the current simplex..

4: N – INTEGER *Input*

On entry: n, the number of variables.

5: NCALL – INTEGER *Input*

On entry: the number of times that FUNCT has been called so far.

6: SERROR – ***double precision*** *Input*

On entry: the current value of the standard deviation in function values used in termination test (1).

7: VRATIO – ***double precision*** *Input*

On entry: the current value of the linearized volume ratio used in termination test (2).

8: IUSER(∗) – INTEGER array *User Workspace*
9: RUSER(∗) – ***double precision*** array *User Workspace*

MONIT is called from E04CBF with the parameters IUSER and RUSER as supplied to E04CBF. You are free to use the arrays IUSER and RUSER to supply information to MONIT.

MONIT must be declared as EXTERNAL in the (sub)program from which E04CBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

8: MAXCAL – INTEGER *Input*

On entry: the maximum number of function evaluations to be allowed.

Constraint: MAXCAL ≥ 1.

9: IUSER(∗) – INTEGER array *User Workspace*

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

IUSER is not used by E04CBF, but is passed directly to FUNCT and MONIT. It may be used to pass information to and from those routines.

10: RUSER(∗) – ***double precision*** array *User Workspace*

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

RUSER is not used by E04CBF, but is passed directly to FUNCT and MONIT. It may be used to pass information to and from those routines.

11: 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, if you are not familiar with this parameter, the recommended value is 0. **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 $= 1$

An input parameter is invalid. The output message provides more details of the invalid argument.

IFAIL $= 2$

MAXCAL function evaluations have been completed without termination test (1) or (2) passing. Check the coding of FUNCT before increasing the value of MAXCAL.

IFAIL $= -999$

Internal memory allocation failed.

7 Accuracy

On a successful exit the accuracy will be as defined by TOLF or TOLX, depending on which criterion was satisfied first.

8 Further Comments

Local workspace arrays of fixed lengths (depending on N) are allocated internally by E04CBF. The total size of these arrays amounts to $\mathrm{N}^2 + 6\mathrm{N} + 1$ ***double precision*** elements.

The time taken by E04CBF depends on the number of variables, the behaviour of the function and the distance of the starting point from the minimum. Each iteration consists of 1 or 2 function evaluations unless the size of the simplex is reduced, in which case $n + 1$ function evaluations are required.

9 Example

This example finds a minimum of the function

$$F(x_1, x_2) = e^{x_1}\left(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1\right).$$

This example uses the initial point $(-1, 1)$ (see Section 9.3), and we expect to reach the minimum at $(0.5, -1)$.

9.1 Program Text

```
*     E04CBF Example Program Text
*     Mark 22 Release. NAG Copyright 2007.
*     .. Parameters ..
      INTEGER          N, NOUT
      PARAMETER        (N=2,NOUT=6)
*     .. Local Scalars ..
      DOUBLE PRECISION F, TOLF, TOLX
      INTEGER          I, IFAIL, MAXCAL
*     .. Local Arrays ..
      DOUBLE PRECISION RUSER(1), X(N)
      INTEGER          IUSER(1)
*     .. External Functions ..
      DOUBLE PRECISION X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         E04CBF, FUNCT, MONIT
*     .. Intrinsic Functions ..
      INTRINSIC        SQRT
*     .. Executable Statements ..
      CONTINUE
*
      WRITE (NOUT,*) 'E04CBF Example Program Results'
*
*     Set IUSER(1) to 1 to obtain monitoring information
*
      IUSER(1) = 0
*
      X(1) = -1.0D0
      X(2) = 1.0D0
      TOLF = SQRT(X02AJF())
      TOLX = SQRT(TOLF)
      MAXCAL = 100
      IFAIL = 1
*
      CALL E04CBF(N,X,F,TOLF,TOLX,FUNCT,MONIT,MAXCAL,IUSER,RUSER,IFAIL)
*
      WRITE (NOUT,*)
      IF (IFAIL.GE.0) THEN
         WRITE (NOUT,99999) IFAIL
*
         IF (IFAIL.EQ.0) THEN
            WRITE (NOUT,99998) F
```

```
            WRITE (NOUT,99997) (X(I),I=1,N)
         END IF
      ELSE
         WRITE (NOUT,99996) ' ** E04CBF returned with IFAIL = ', IFAIL
      END IF
*
99999 FORMAT (1X,'On exit from E04CBF, IFAIL = ',I4)
99998 FORMAT (1X,'The final function value is',F12.4)
99997 FORMAT (1X,'at the point',2F12.4)
99996 FORMAT (1X,A,I5)
      END
*
      SUBROUTINE FUNCT(N,XC,FC,IUSER,RUSER)
*     .. Scalar Arguments ..
      DOUBLE PRECISION FC
      INTEGER          N
*     .. Array Arguments ..
      DOUBLE PRECISION RUSER(*), XC(N)
      INTEGER          IUSER(*)
*     .. Intrinsic Functions ..
      INTRINSIC        EXP
*     .. Executable Statements ..
      CONTINUE
*
      FC = EXP(XC(1))*(4.0D0*XC(1)*(XC(1)+XC(2))+2.0D0*XC(2)*(XC(2)
     +     +1.0D0)+1.0D0)
*
      RETURN
      END
*
      SUBROUTINE MONIT(FMIN,FMAX,SIM,N,NCALL,SERROR,VRATIO,IUSER,RUSER)
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalar Arguments ..
      DOUBLE PRECISION FMAX, FMIN, SERROR, VRATIO
      INTEGER          N, NCALL
*     .. Array Arguments ..
      DOUBLE PRECISION RUSER(*), SIM(N+1,N)
      INTEGER          IUSER(*)
*     .. Local Scalars ..
      INTEGER          I, J
*     .. Executable Statements ..
      CONTINUE
*
      IF (IUSER(1).NE.0) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99999) NCALL
         WRITE (NOUT,99998) FMIN
         WRITE (NOUT,99997)
         WRITE (NOUT,99996) ((SIM(I,J),J=1,N),I=1,N+1)
         WRITE (NOUT,99995) SERROR
         WRITE (NOUT,99994) VRATIO
      END IF
*
      RETURN
*
99999 FORMAT (1X,'There have been',I5,' function calls')
99998 FORMAT (1X,'The smallest function value is',F10.4)
99997 FORMAT (1X,'The simplex is')
99996 FORMAT (1X,2F10.4)
99995 FORMAT (1X,'The standard deviation in function values at the ',
     +       'vertices of the simplex is',F10.4)
99994 FORMAT (1X,'The linearized volume ratio of the current simplex',
     +       ' to the starting one is',F10.4)
      END
```

9.2 Program Data

None.

9.3 Program Results

```
E04CBF Example Program Results

On exit from E04CBF, IFAIL =    0
The final function value is      0.0000
at the point      0.5000     -0.9999
```

Example Program
Contours of *F*Showing the Initial Point (X) and Local Minimum (*)