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, RUSER,     &
                   IFAIL)
INTEGER            N, MAXCAL, IUSER(*), IFAIL
REAL (KIND=nag_wp) 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) – REAL (KIND=nag_wp) 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 – REAL (KIND=nag_wp) *Output*

On exit: the lowest function value found.

4: TOLF – REAL (KIND=nag_wp) *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 – REAL (KIND=nag_wp) *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(*)
REAL (KIND=nag_wp) XC(N), FC, RUSER(*)
```

1: N – INTEGER *Input*

On entry: n, the number of variables.

2: XC(N) – REAL (KIND=nag_wp) array *Input*

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

3: FC – REAL (KIND=nag_wp) *Output*

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

4: IUSER(∗) – INTEGER array *User Workspace*
5: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

FUNCT is called 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 as an alternative to using COMMON global variables.

FUNCT must either be a module subprogram USEd by, or 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 NAG Library or the user. *External Procedure*

MONIT may be used to monitor the optimization process. It is invoked once every iteration.

If no monitoring is required, MONIT may be the dummy monitoring routine E04CBK supplied by the NAG Library.

The specification of MONIT is:

```
SUBROUTINE MONIT (FMIN, FMAX, SIM, N, NCALL, SERROR, VRATIO, IUSER,      &
                  RUSER)
INTEGER             N, NCALL, IUSER(*)
REAL (KIND=nag_wp) FMIN, FMAX, SIM(N+1,N), SERROR, VRATIO, RUSER(*)
```

1: FMIN – REAL (KIND=nag_wp) *Input*

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

2: FMAX – REAL (KIND=nag_wp) *Input*

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

3: SIM(N + 1,N) – REAL (KIND=nag_wp) 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 – REAL (KIND=nag_wp) *Input*

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

7: VRATIO – REAL (KIND=nag_wp) *Input*

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

8: IUSER(∗) – INTEGER array *User Workspace*
9: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

MONIT is called 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 as an alternative to using COMMON global variables.

MONIT must either be a module subprogram USEd by, or 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*
10: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by E04CBF, but are passed directly to FUNCT and MONIT and may be used to pass information to these routines as an alternative to using COMMON global variables.

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.

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.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

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 $\mathrm{N}^2 + 6\mathrm{N} + 2$ real 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 24 Release. NAG Copyright 2012.
    Module e04cbfe_mod

!     E04CBF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: nout = 6
    Contains
      Subroutine funct(n,xc,fc,iuser,ruser)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: fc
        Integer, Intent (In)                 :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Real (Kind=nag_wp), Intent (In)      :: xc(n)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Intrinsic Procedures ..
        Intrinsic                            :: exp
!       .. Executable Statements ..
        fc = exp(xc(1))*(4.0_nag_wp*xc(1)*(xc(1)+xc(2))+2.0_nag_wp*xc(2)*(xc(2 &
          )+1.0_nag_wp)+1.0_nag_wp)

        Return

      End Subroutine funct
      Subroutine monit(fmin,fmax,sim,n,ncall,serror,vratio,iuser,ruser)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: fmax, fmin, serror, vratio
        Integer, Intent (In)                 :: n, ncall
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Real (Kind=nag_wp), Intent (In)      :: sim(n+1,n)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Executable Statements ..
        Write (nout,*)
        Write (nout,99999) ncall
        Write (nout,99998) fmin
        Write (nout,99997)
        Write (nout,99996) sim(1:(n+1),1:n)
        Write (nout,99995) serror
```

```
        Write (nout,99994) vratio

        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 Subroutine monit
    End Module e04cbfe_mod
    Program e04cbfe

!     E04CBF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04cbf, e04cbk, nag_wp, x02ajf
      Use e04cbfe_mod, Only: funct, monit, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                  :: n = 2
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: f, tolf, tolx
      Integer                             :: ifail, maxcal
      Logical                             :: monitoring
!     .. Local Arrays ..
      Real (Kind=nag_wp)                  :: ruser(1), x(n)
      Integer                             :: iuser(1)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: sqrt
!     .. Executable Statements ..
      Write (nout,*) 'E04CBF Example Program Results'

!     Set MONITORING to .TRUE. to obtain monitoring information

      monitoring = .False.

      x(1:n) = (/-1.0_nag_wp,1.0_nag_wp/)
      tolf = sqrt(x02ajf())
      tolx = sqrt(tolf)
      maxcal = 100

      ifail = 0

      If (.Not. monitoring) Then

        Call e04cbf(n,x,f,tolf,tolx,funct,e04cbk,maxcal,iuser,ruser,ifail)

      Else

        Call e04cbf(n,x,f,tolf,tolx,funct,monit,maxcal,iuser,ruser,ifail)

      End If

      Write (nout,*)
      Write (nout,99999) f
      Write (nout,99998) x(1:n)

99999 Format (1X,'The final function value is',F12.4)
99998 Format (1X,'at the point',2F12.4)
    End Program e04cbfe
```

9.2 Program Data

None.

9.3 Program Results

```
E04CBF Example Program Results

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 (*)