NAG Library Routine Document

E04HCF

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

E04HCF checks that a subroutine for evaluating an objective function and its first derivatives produces derivative values which are consistent with the function values calculated.

2 Specification

```
SUBROUTINE E04HCF (N, FUNCT, X, F, G, IW, LIW, W, LW, IFAIL)
INTEGER            N, IW(LIW), LIW, LW, IFAIL
REAL (KIND=nag_wp) X(N), F, G(N), W(LW)
EXTERNAL           FUNCT
```

3 Description

Routines for minimizing a function of several variables may require you to supply a subroutine to evaluate the objective function $F(x_1, x_2, \ldots, x_n)$ and its first derivatives. E04HCF is designed to check the derivatives calculated by such user-supplied subroutines . As well as the routine to be checked (FUNCT), you must supply a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the check will be made. Note that E04HCF checks routines of the form required for E04KDF and E04LBF.

E04HCF first calls FUNCT to evaluate F and its first derivatives $g_j = \dfrac{\partial F}{\partial x_j}$, for $j = 1, 2, \ldots, n$ at x. The components of the user-supplied derivatives along two orthogonal directions (defined by unit vectors p_1 and p_2, say) are then calculated; these will be $g^{\mathrm{T}}p_1$ and $g^{\mathrm{T}}p_2$ respectively. The same components are also estimated by finite differences, giving quantities

$$v_k = \frac{F(x + hp_k) - F(x)}{h}, \quad k = 1, 2$$

where h is a small positive scalar. If the relative difference between v_1 and $g^{\mathrm{T}}p_1$ or between v_2 and $g^{\mathrm{T}}p_2$ is judged too large, an error indicator is set.

4 References

None.

5 Arguments

1: N – INTEGER *Input*

On entry: the number n of independent variables in the objective function.

Constraint: N ≥ 1.

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

FUNCT must evaluate the function and its first derivatives at a given point. (The minimization routines mentioned in Section 3 gives you the option of resetting arguments of FUNCT to cause the minimization process to terminate immediately. E04HCF will also terminate immediately, without finishing the checking process, if the argument in question is reset.)

The specification of FUNCT is:

```
SUBROUTINE FUNCT (IFLAG, N, XC, FC, GC, IW, LIW, W, LW)
INTEGER             IFLAG, N, IW(LIW), LIW, LW
REAL (KIND=nag_wp) XC(N), FC, GC(N), W(LW)
```

1: IFLAG – INTEGER *Input/Output*

On entry: will be set to 2.

On exit: if you reset IFLAG to a negative number in FUNCT and return control to E04HCF, E04HCF will terminate immediately with IFAIL set to your setting of IFLAG.

2: N – INTEGER *Input*

On entry: the number n of variables.

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

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

4: FC – REAL (KIND=nag_wp) *Output*

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

5: GC(N) – REAL (KIND=nag_wp) array *Output*

On exit: unless FUNCT resets IFLAG, GC(j) must be set 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 (KIND=nag_wp) array *Workspace*

9: LW – INTEGER *Input*

These arguments are present so that FUNCT will be of the form required by the minimization routines mentioned in Section 3. FUNCT is called with E04HCF's arguments IW, LIW, W, LW as these arguments. If the advice given in the minimization routine documents is being followed, you will have no reason to examine or change any elements of IW or W. In any case, FUNCT **must not change** the first 3 × N elements of W.

FUNCT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which E04HCF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

3: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: X(j), for $j = 1, 2, \ldots, n$, must be set to the coordinates of a suitable point at which to check the derivatives calculated by FUNCT. 'Obvious' settings, such as 0.0 or 1.0, should not be used since, at such particular points, incorrect terms may take correct values (particularly zero), so that errors could go undetected. Similarly, it is preferable that no two elements of X should be the same.

4: F – REAL (KIND=nag_wp) *Output*

On exit: unless you set IFLAG negative in the first call of FUNCT, F contains the value of the objective function $F(x)$ at the point given by you in X.

5: G(N) – REAL (KIND=nag_wp) array *Output*

On exit: unless you set IFLAG negative in the first call of FUNCT, G(j) contains the value of the derivative $\dfrac{\partial F}{\partial x_j}$ at the point given in X, as calculated by FUNCT, for $j = 1, 2, \ldots, n$.

6: IW(LIW) – INTEGER array *Communication Array*

This array is in the argument list so that it can be used by other library routines for passing integer quantities to FUNCT. It is not examined or changed by E04HCF. Generally, you must provide an array IW but are advised not to use it.

7: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04HCF is called.

Constraint: LIW ≥ 1.

8: W(LW) – REAL (KIND=nag_wp) array *Communication Array*

9: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04HCF is called.

Constraint: LW $\geq 3 \times$ N.

10: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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, because for this routine the values of the output arguments 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.**

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).

Note: E04HCF 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 E04HCF because you have set IFLAG negative in FUNCT. The setting of IFAIL will be the same as your setting of IFLAG. The check on FUNCT will not have been completed.

IFAIL $= 1$

On entry, N < 1,
or LIW < 1,
or LW $< 3 \times$ N.

IFAIL $= 2$

You should check carefully the derivation and programming of expressions for the derivatives of $F(x)$, because it is very unlikely that FUNCT is calculating them correctly.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

IFAIL is set to 2 if

$$\left(v_k - g^{\mathrm{T}} p_k\right)^2 \geq h \times \left(\left(g^{\mathrm{T}} p_k\right)^2 + 1\right)$$

for $k = 1$ or 2. (See Section 3 for definitions of the quantities involved.) The scalar h is set equal to $\sqrt{\epsilon}$, where ϵ is the ***machine precision*** as given by X02AJF.

8 Parallelism and Performance

E04HCF is not threaded in any implementation.

9 Further Comments

FUNCT is called 3 times.

Before using E04HCF to check the calculation of first derivatives, you should be confident that FUNCT is calculating F correctly. The usual way of checking the calculation of the function is to compare values of $F(x)$ calculated by FUNCT at nontrivial points x with values calculated independently. ('Non-trivial' means that, as when setting x before calling E04HCF, coordinates such as 0.0 or 1.0 should be avoided.)

E04HCF only checks the derivatives calculated when IFLAG $= 2$. So, if FUNCT is intended for use in conjunction with a minimization routine which may set IFLAG to 1, you must check that, for given settings of the XC(j), FUNCT produces the same values for the GC(j) when IFLAG is set to 1 as when IFLAG is set to 2.

10 Example

Suppose that it is intended to use E04KDF to minimize

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

The following program could be used to check the first derivatives calculated by FUNCT. (The tests of whether IFLAG $= 0$ or 1 in FUNCT are present ready for when FUNCT is called by E04KDF. E04HCF will always call FUNCT with IFLAG set to 2.)

10.1 Program Text

```
!   E04HCF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module e04hcfe_mod

!     E04HCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: funct
!     .. Parameters ..
      Integer, Parameter, Public       :: liw = 1, n = 4, nout = 6
      Integer, Parameter, Public       :: lw = 3*n
    Contains
      Subroutine funct(iflag,n,xc,fc,gc,iw,liw,w,lw)
!       Routine to evaluate objective function and its 1st derivatives.

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: fc
        Integer, Intent (Inout)        :: iflag
        Integer, Intent (In)           :: liw, lw, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: gc(n)
        Real (Kind=nag_wp), Intent (Inout) :: w(lw)
        Real (Kind=nag_wp), Intent (In) :: xc(n)
        Integer, Intent (Inout)        :: iw(liw)
!       .. Executable Statements ..
        fc = (xc(1)+10.0_nag_wp*xc(2))**2 + 5.0_nag_wp*(xc(3)-xc(4))**2 +      &
          (xc(2)-2.0_nag_wp*xc(3))**4 + 10.0_nag_wp*(xc(1)-xc(4))**4
        gc(1) = 2.0_nag_wp*(xc(1)+10.0_nag_wp*xc(2)) +                         &
          40.0_nag_wp*(xc(1)-xc(4))**3
        gc(2) = 20.0_nag_wp*(xc(1)+10.0_nag_wp*xc(2)) +                        &
          4.0_nag_wp*(xc(2)-2.0_nag_wp*xc(3))**3
        gc(3) = 10.0_nag_wp*(xc(3)-xc(4)) - 8.0_nag_wp*(xc(2)-2.0_nag_wp*xc(3) &
          )**3
        gc(4) = 10.0_nag_wp*(xc(4)-xc(3)) - 40.0_nag_wp*(xc(1)-xc(4))**3

        Return

      End Subroutine funct
    End Module e04hcfe_mod
    Program e04hcfe

!     E04HCF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04hcf, nag_wp
      Use e04hcfe_mod, Only: funct, liw, lw, n, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: f
      Integer                          :: ifail
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: g(n), w(lw), x(n)
      Integer                          :: iw(liw)
!     .. Executable Statements ..
      Write (nout,*) 'E04HCF Example Program Results'

!     Set up an arbitrary point at which to check the 1st derivatives

      x(1:n) = (/1.46_nag_wp,-0.82_nag_wp,0.57_nag_wp,1.21_nag_wp/)

      Write (nout,*)
      Write (nout,*) 'The test point is'
```

```
      Write (nout,99999) x(1:n)

      ifail = -1
      Call e04hcf(n,funct,x,f,g,iw,liw,w,lw,ifail)

      If (ifail>=0) Then
        Write (nout,*)

        If (ifail==0) Then
          Write (nout,*) '1st derivatives are consistent with function values'
        Else
          Write (nout,*) 'Probable error in calculation of 1st derivatives'
        End If

        Write (nout,*)
        Write (nout,99998) 'At the test point, FUNCT gives the function value' &
          , f
        Write (nout,*) 'and the 1st derivatives'
        Write (nout,99997) g(1:n)
      End If

99999 Format (1X,4F10.4)
99998 Format (1X,A,1P,E12.4)
99997 Format (1X,1P,4E12.3)
    End Program e04hcfe
```

10.2 Program Data

None.

10.3 Program Results

```
 E04HCF Example Program Results

 The test point is
     1.4600   -0.8200    0.5700    1.2100

 1st derivatives are consistent with function values

 At the test point, FUNCT gives the function value  6.2273E+01
 and the 1st derivatives
   -1.285E+01  -1.649E+02   5.384E+01   5.775E+00
```