NAG Library Routine Document

E04HDF

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

E04HDF checks that a subroutine for calculating second derivatives of an objective function is consistent with a subroutine for calculating the corresponding first derivatives.

2 Specification

```
SUBROUTINE E04HDF (N, FUNCT, H, X, G, HESL, LH, HESD, IW, LIW, W, LW,      &
                   IFAIL)

INTEGER             N, LH, IW(LIW), LIW, LW, IFAIL
REAL (KIND=nag_wp) X(N), G(N), HESL(LH), HESD(N), W(LW)
EXTERNAL            FUNCT, H
```

3 Description

Routines for minimizing a function $F(x_1, x_2, \ldots, x_n)$ of the variables $x_1, x_2, \ldots, x_n$ may require you to provide a subroutine to evaluate the second derivatives of F. E04HDF is designed to check the second derivatives calculated by such user-supplied subroutines. As well as the routine to be checked (H), you must supply a subroutine (FUNCT) to evaluate the first derivatives, and a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the checks will be made. Note that E04HDF checks routines of the form required for E04LBF.

E04HDF first calls user-supplied subroutines FUNCT and H to evaluate the first and second derivatives of F at x. The user-supplied Hessian matrix (H, say) is projected onto two orthogonal vectors y and z to give the scalars $y^{\mathrm{T}}Hy$ and $z^{\mathrm{T}}Hz$ respectively. The same projections of the Hessian matrix are also estimated by finite differences, giving

$$p = (y^{\mathrm{T}}g(x + hy) - y^{\mathrm{T}}g(x))/h \quad \text{and} \quad q = (z^{\mathrm{T}}g(x + hz) - z^{\mathrm{T}}g(x))/h$$

respectively, where $g()$ denotes the vector of first derivatives at the point in brackets and h is a small positive scalar. If the relative difference between p and $y^{\mathrm{T}}Hy$ or between q and $z^{\mathrm{T}}Hz$ is judged too large, an error indicator is set.

4 References

None.

5 Parameters

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. (E04LBF gives you the option of resetting parameters of FUNCT to cause the minimization process to terminate immediately. E04HDF will also terminate immediately, without finishing the checking process, if the parameter 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: to FUNCT, IFLAG will be set to 2.

On exit: if you set IFLAG to some negative number in FUNCT and return control to E04HDF, E04HDF 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 the function and first 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 objective 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 parameters are present so that FUNCT will be of the form required by E04LBF. FUNCT is called with E04HDF's parameters IW, LIW, W, LW as these parameters. If the advice given in E04LBF 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 $5 \times$ N elements of W.

FUNCT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04HDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

E04HCF should be used to check the first derivatives calculated by FUNCT before E04HDF is used to check the second derivatives, since E04HDF assumes that the first derivatives are correct.

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

H must evaluate the second derivatives of the function at a given point. (As with FUNCT, a parameter can be set to cause immediate termination.)

The specification of H is:

```
SUBROUTINE H (IFLAG, N, XC, FHESL, LH, FHESD, IW, LIW, W, LW)
INTEGER             IFLAG, N, LH, IW(LIW), LIW, LW
REAL (KIND=nag_wp) XC(N), FHESL(LH), FHESD(N), W(LW)
```

1: IFLAG – INTEGER *Input/Output*

On entry: is set to a non-negative number.

On exit: if H resets IFLAG to a negative number, E04HDF 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 the second derivatives of $F(x)$ are required.

4: FHESL(LH) – REAL (KIND=nag_wp) array *Output*

On exit: unless IFLAG is reset, H must place the strict lower triangle of the second derivative matrix of F (evaluated at the point x) in FHESL, stored by rows, i.e., FHESL$((i-1)(i-2)/2+j)$ must be set to the value of $\dfrac{\partial^2 F}{\partial x_i \partial x_j}$ at the point x, for $i = 2, 3, \ldots, n$ and $j = 1, 2, \ldots, i-1$. (The upper triangle is not required because the matrix is symmetric.)

5: LH – INTEGER *Input*

On entry: the length of the array FHESL.

6: FHESD(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: contains the value of $\dfrac{\partial F}{\partial x_j}$ at the point x, for $j = 1, 2, \ldots, n$. Routines written to take advantage of a similar feature of E04LBF can be tested as they stand by E04HDF.

On exit: unless IFLAG is reset, H must place the diagonal elements of the second derivative matrix of F (evaluated at the point x) in FHESD, i.e., FHESD(j) must be set to the value of $\dfrac{\partial^2 F}{\partial x_j^2}$ at the point x, for $j = 1, 2, \ldots, n$.

7: IW(LIW) – INTEGER array *Workspace*
8: LIW – INTEGER *Input*
9: W(LW) – REAL (KIND=nag_wp) array *Workspace*
10: LW – INTEGER *Input*

As in FUNCT, these parameters correspond to the parameters IW, LIW, W and LW of E04HDF. H **must not change** the first $5 \times$ N elements of W.

H must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04HDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

On entry: X(j), for $j = 1, 2, \ldots, n$ must contain 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 advisable that no two elements of X should be the same.

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 first derivative $\dfrac{\partial F}{\partial x_j}$ at the point given in X, as calculated by FUNCT, for $j = 1, 2, \ldots, \mathrm{N}$.

6: HESL(LH) – REAL (KIND=nag_wp) array *Output*

On exit: unless you set IFLAG negative in H, HESL contains the strict lower triangle of the second derivative matrix of F, as evaluated by H at the point given in X, stored by rows.

7: LH – INTEGER *Input*

On entry: the dimension of the array HESL as declared in the (sub)program from which E04HDF is called.

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

8: HESD(N) – REAL (KIND=nag_wp) array *Output*

On exit: unless you set IFLAG negative in H, HESD contains the diagonal elements of the second derivative matrix of F, as evaluated by H at the point given in X.

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

This array is in the parameter list so that it can be used by other library routines for passing integer quantities to FUNCT or H. It is not examined or changed by E04HDF. In general you must provide an array IW, but are advised not to use it.

10: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04HDF is called.

Constraint: $\mathrm{LIW} \geq 1$.

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

12: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04HDF is called.

Constraint: $\mathrm{LW} \geq 5 \times \mathrm{N}$.

13: 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, because for this routine the values of the output parameters may be useful even if $\mathrm{IFAIL} \neq 0$ 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: $\mathrm{IFAIL} = 0$ 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: E04HDF 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 E04HDF because you have set IFLAG negative in FUNCT or H. The value of IFAIL will be the same as the value you set for IFLAG. The check on H will not have been completed.

IFAIL $= 1$

On entry, N < 1,
or LH $< \max(1, \mathrm{N} \times (\mathrm{N} - 1)/2)$,
or LIW < 1,
or LW $< 5 \times \mathrm{N}$.

IFAIL $= 2$

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

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

IFAIL is set to 2 if

$$|y^{\mathrm{T}}Hy - p| \geq \sqrt{h} \times (|y^{\mathrm{T}}Hy| + 1.0) \quad \text{or}$$
$$|z^{\mathrm{T}}Hz - q| \geq \sqrt{h} \times (|z^{\mathrm{T}}Hz| + 1.0)$$

where h is set equal to $\sqrt{\epsilon}$ (ϵ being the ***machine precision*** as given by X02AJF) and other quantities are as defined in Section 3.

8 Parallelism and Performance

E04HDF is not threaded by NAG in any implementation.

E04HDF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

E04HDF calls H once and FUNCT three times.

10 Example

Suppose that it is intended to use E04LBF 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 second derivatives calculated by H required. (The call of E04HDF is preceded by a call of E04HCF to check FUNCT which calculates the first derivatives.)

10.1 Program Text

```
!   E04HDF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.
    Module e04hdfe_mod

!     E04HDF 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, h
!     .. Parameters ..
      Integer, Parameter, Public            :: liw = 1, n = 4, nout = 6
      Integer, Parameter, Public            :: lh = n*(n-1)/2
      Integer, Parameter, Public            :: lw = 5*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
      Subroutine h(iflag,n,xc,fhesl,lh,fhesd,iw,liw,w,lw)
!       Routine to evaluate 2nd derivatives
```

```
!       .. Scalar Arguments ..
        Integer, Intent (Inout)              :: iflag
        Integer, Intent (In)                 :: lh, liw, lw, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: fhesd(n), w(lw)
        Real (Kind=nag_wp), Intent (Out)     :: fhesl(lh)
        Real (Kind=nag_wp), Intent (In)      :: xc(n)
        Integer, Intent (Inout)              :: iw(liw)
!       .. Executable Statements ..
        fhesd(1) = 2.0_nag_wp + 120.0_nag_wp*(xc(1)-xc(4))**2
        fhesd(2) = 200.0_nag_wp + 12.0_nag_wp*(xc(2)-2.0_nag_wp*xc(3))**2
        fhesd(3) = 10.0_nag_wp + 48.0_nag_wp*(xc(2)-2.0_nag_wp*xc(3))**2
        fhesd(4) = 10.0_nag_wp + 120.0_nag_wp*(xc(1)-xc(4))**2

        fhesl(1) = 20.0_nag_wp
        fhesl(2) = 0.0_nag_wp
        fhesl(3) = -24.0_nag_wp*(xc(2)-2.0_nag_wp*xc(3))**2
        fhesl(4) = -120.0_nag_wp*(xc(1)-xc(4))**2
        fhesl(5) = 0.0_nag_wp
        fhesl(6) = -10.0_nag_wp

        Return

      End Subroutine h
    End Module e04hdfe_mod
    Program e04hdfe

!     E04HDF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04hcf, e04hdf, nag_wp
      Use e04hdfe_mod, Only: funct, h, lh, liw, lw, n, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: f
      Integer                              :: i, ifail, k
!     .. Local Arrays ..
      Real (Kind=nag_wp)                   :: g(n), hesd(n), hesl(lh), w(lw),  &
                                              x(n)
      Integer                              :: iw(liw)
!     .. Executable Statements ..
      Write (nout,*) 'E04HDF Example Program Results'

!     Set up an arbitrary point at which to check the 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)

!     Check the 1st derivatives

      ifail = 0
      Call e04hcf(n,funct,x,f,g,iw,liw,w,lw,ifail)

!     Check the 2nd derivatives

      ifail = -1
      Call e04hdf(n,funct,h,x,g,hesl,lh,hesd,iw,liw,w,lw,ifail)

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

        If (ifail==0) Then
          Write (nout,*) '2nd derivatives are consistent with 1st derivatives'
        Else If (ifail==2) Then
          Write (nout,*) 'Probable error in calculation of 2nd 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)
        Write (nout,*)
        Write (nout,*) 'H gives the lower triangle of the Hessian matrix'
        Write (nout,99996) hesd(1)

        k = 1

        Do i = 2, n
          Write (nout,99996) hesl(k:(k+i-2)), hesd(i)
          k = k + i - 1
        End Do

      End If

99999 Format (1X,4F9.4)
99998 Format (1X,A,1P,E12.4)
99997 Format (1X,1P,4E12.3)
99996 Format (1X,1P,4E12.3)
    End Program e04hdfe
```

10.2 Program Data

None.

10.3 Program Results

```
 E04HDF Example Program Results

 The test point is
    1.4600  -0.8200   0.5700   1.2100

 2nd derivatives are consistent with 1st derivatives

 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

 H gives the lower triangle of the Hessian matrix
    9.500E+00
    2.000E+01   2.461E+02
    0.000E+00  -9.220E+01   1.944E+02
   -7.500E+00   0.000E+00  -1.000E+01   1.750E+01
```