NAG Library Routine Document

E04XAF/E04XAA

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

E04XAF/E04XAA computes an approximation to the gradient vector and/or the Hessian matrix for use in conjunction with, or following the use of an optimization routine (such as E04UFF/E04UFA).

E04XAA is a version of E04XAF that has additional arguments in order to make it safe for use in multithreaded applications (see Section 5).

2 Specification

2.1 Specification for E04XAF

```
SUBROUTINE E04XAF (MSGLVL, N, EPSRF, X, MODE, OBJFUN, LDH, HFORW, OBJF,     &
                   OBJGRD, HCNTRL, H, IWARN, WORK, IUSER, RUSER, INFO,      &
                   IFAIL)

INTEGER            MSGLVL, N, MODE, LDH, IWARN, IUSER(*), INFO(N),          &
                   IFAIL
REAL (KIND=nag_wp) EPSRF, X(N), HFORW(N), OBJF, OBJGRD(N), HCNTRL(N),       &
                   H(LDH,*), WORK(*), RUSER(*)
EXTERNAL           OBJFUN
```

2.2 Specification for E04XAA

```
SUBROUTINE E04XAA (MSGLVL, N, EPSRF, X, MODE, OBJFUN, LDH, HFORW, OBJF,     &
                   OBJGRD, HCNTRL, H, IWARN, WORK, IUSER, RUSER, INFO,      &
                   LWSAV, IWSAV, RWSAV, IFAIL)

INTEGER            MSGLVL, N, MODE, LDH, IWARN, IUSER(*), INFO(N),          &
                   IWSAV(1), IFAIL
REAL (KIND=nag_wp) EPSRF, X(N), HFORW(N), OBJF, OBJGRD(N), HCNTRL(N),       &
                   H(LDH,*), WORK(*), RUSER(*), RWSAV(1)
LOGICAL            LWSAV(1)
EXTERNAL           OBJFUN
```

3 Description

E04XAF/E04XAA is similar to routine FDCALC described in Gill *et al.* (1983a). It should be noted that this routine aims to compute sufficiently accurate estimates of the derivatives for use with an optimization algorithm. If you require more accurate estimates you should refer to Chapter D04.

E04XAF/E04XAA computes finite difference approximations to the gradient vector and the Hessian matrix for a given function. The simplest approximation involves the forward-difference formula, in which the derivative $f'(x)$ of a univariate function $f(x)$ is approximated by the quantity

$$\rho_F(f,h) = \frac{f(x+h) - f(x)}{h}$$

for some interval $h > 0$, where the subscript `F' denotes 'forward-difference' (see Gill *et al.* (1983b)).

To summarise the procedure used by E04XAF/E04XAA (for the case when the objective function is available and you require estimates of gradient values and Hessian matrix diagonal values, i.e., MODE $= 0$) consider a univariate function f at the point x. (In order to obtain the gradient of a multivariate function $F(x)$, where x is an n-vector, the procedure is applied to each component of x, keeping the other components fixed.) Roughly speaking, the method is based on the fact that the bound on the relative truncation error in the forward-difference approximation tends to be an increasing

function of h, while the relative condition error bound is generally a decreasing function of h, hence changes in h will tend to have opposite effects on these errors (see Gill *et al.* (1983b)).

The 'best' interval h is given by

$$h_F = 2\sqrt{\frac{(1+|f(x)|)e_R}{|\Phi|}} \tag{1}$$

where Φ is an estimate of $f''(x)$, and e_R is an estimate of the relative error associated with computing the function (see Chapter 8 of Gill *et al.* (1981)). Given an interval h, Φ is defined by the second-order approximation

$$\Phi = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

The decision as to whether a given value of Φ is acceptable involves $\hat{c}(\Phi)$, the following bound on the relative condition error in Φ:

$$\hat{c}(\Phi) = \frac{4e_R(1+|f|)}{h^2|\Phi|}$$

(When Φ is zero, $\hat{c}(\Phi)$ is taken as an arbitrary large number.)

The procedure selects the interval h_ϕ (to be used in computing Φ) from a sequence of trial intervals (h_k). The initial trial interval is taken as $10\bar{h}$, where

$$\bar{h} = 2(1+|x|)\sqrt{e_R}$$

unless you specify the initial value to be used.

The value of $\hat{c}(\Phi)$ for a trial value h_k is defined as 'acceptable' if it lies in the interval $[0.001, 0.1]$. In this case h_ϕ is taken as h_k, and the current value of Φ is used to compute h_F from (1). If $\hat{c}(\Phi)$ is unacceptable, the next trial interval is chosen so that the relative condition error bound will either decrease or increase, as required. If the bound on the relative condition error is too large, a larger interval is used as the next trial value in an attempt to reduce the condition error bound. On the other hand, if the relative condition error bound is too small, h_k is reduced.

The procedure will fail to produce an acceptable value of $\hat{c}(\Phi)$ in two situations. Firstly, if $f''(x)$ is extremely small, then $\hat{c}(\Phi)$ may never become small, even for a very large value of the interval. Alternatively, $\hat{c}(\Phi)$ may never exceed 0.001, even for a very small value of the interval. This usually implies that $f''(x)$ is extremely large, and occurs most often near a singularity.

As a check on the validity of the estimated first derivative, the procedure provides a comparison of the forward-difference approximation computed with h_F (as above) and the central-difference approximation computed with h_ϕ. Using the central-difference formula the first derivative can be approximated by

$$\rho_c(f,h) = \frac{f(x+h) - f(x-h)}{2h}$$

where $h > 0$. If the values h_F and h_ϕ do not display some agreement, neither can be considered reliable.

When both function and gradients are available and you require the Hessian matrix (i.e., MODE $= 1$) E04XAF/E04XAA follows a similar procedure to the case above with the exception that the gradient function $g(x)$ is substituted for the objective function and so the forward-difference interval for the first derivative of $g(x)$ with respect to variable x_j is computed. The jth column of the approximate Hessian matrix is then defined as in Chapter 2 of Gill *et al.* (1981), by

$$\frac{g\big(x + h_j e_j\big) - g(x)}{h_j}$$

where h_j is the best forward-difference interval associated with the jth component of g and e_j is the vector with unity in the jth position and zeros elsewhere.

When only the objective function is available and you require the gradients and Hessian matrix (i.e., MODE = 2) E04XAF/E04XAA again follows the same procedure as the case for MODE = 0 except that this time the value of $\hat{c}(\Phi)$ for a trial value h_k is defined as acceptable if it lies in the interval $[0.0001, 0.01]$ and the initial trial interval is taken as

$$\bar{h} = 2(1 + |x|)\sqrt[4]{e_R}.$$

The approximate Hessian matrix G is then defined as in Chapter 2 of Gill *et al.* (1981), by

$$G_{ij}(x) = \frac{1}{h_i h_j}\big(f\big(x + h_i e_i + h_j e_j\big) - f(x + h_i e_i) - f\big(x + h_j e_j\big) + f(x)\big).$$

4 References

Gill P E, Murray W, Saunders M A and Wright M H (1983a) Documentation for FDCALC and FDCORE *Technical Report SOL* 83–6 Stanford University

Gill P E, Murray W, Saunders M A and Wright M H (1983b) Computing forward-difference intervals for numerical optimization *SIAM J. Sci. Statist. Comput.* **4** 310–321

Gill P E, Murray W and Wright M H (1981) *Practical Optimization* Academic Press

5 Arguments

1: MSGLVL – INTEGER *Input*

On entry: must indicate the amount of intermediate output desired (see Section 9.1 for a description of the printed output). All output is written on the current advisory message unit (see X04ABF).

Value	Definition
0	No printout
1	A summary is printed out for each variable plus any warning messages.
Other	Values other than 0 and 1 should normally be used **only** at the direction of NAG.

2: N – INTEGER *Input*

On entry: the number n of independent variables.

Constraint: N $\geq$ 1.

3: EPSRF – REAL (KIND=nag_wp) *Input*

On entry: must define e_R, which is intended to be a measure of the accuracy with which the problem function F can be computed. The value of e_R should reflect the relative precision of $1 + |F(x)|$, i.e., acts as a relative precision when $|F|$ is large, and as an absolute precision when $|F|$ is small. For example, if $F(x)$ is typically of order 1000 and the first six significant digits are known to be correct, an appropriate value for e_R would be 1.0E−6.

A discussion of EPSRF is given in Chapter 8 of Gill *et al.* (1981). If EPSRF is either too small or too large on entry a warning will be printed if MSGLVL = 1, the argument IWARN set to the appropriate value on exit and E04XAF/E04XAA will use a default value of $e_M^{0.9}$, where e_M is the ***machine precision***.

If EPSRF $\leq$ 0.0 on entry then E04XAF/E04XAA will use the default value internally. The default value will be appropriate for most simple functions that are computed with full accuracy.

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

On entry: the point x at which the derivatives are to be computed.

5: MODE – INTEGER *Input/Output*

On entry: indicates which derivatives are required.

MODE = 0
: The gradient and Hessian diagonal values having supplied the objective function via OBJFUN.

MODE = 1
: The Hessian matrix having supplied both the objective function and gradients via OBJFUN.

MODE = 2
: The gradient values and Hessian matrix having supplied the objective function via OBJFUN.

On exit: is changed **only** if you set MODE negative in OBJFUN, i.e., you have requested termination of E04XAF/E04XAA.

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

If MODE = 0 or 2, OBJFUN must calculate the objective function; otherwise if MODE = 1, OBJFUN must calculate the objective function and the gradients.

The specification of OBJFUN is:

```
SUBROUTINE OBJFUN (MODE, N, X, OBJF, OBJGRD, NSTATE, IUSER,          &
                   RUSER)
INTEGER            MODE, N, NSTATE, IUSER(*)
REAL (KIND=nag_wp) X(N), OBJF, OBJGRD(N), RUSER(*)
```

1: MODE – INTEGER *Input/Output*

MODE indicates which argument values within OBJFUN need to be set.

On entry: to OBJFUN, MODE is always set to the value that you set it to before the call to E04XAF/E04XAA.

On exit: its value must not be altered unless you wish to indicate a failure within OBJFUN, in which case it should be set to a negative value. If MODE is negative on exit from OBJFUN, the execution of E04XAF/E04XAA is terminated with IFAIL set to MODE.

2: N – INTEGER *Input*

On entry: the number n of variables as input to E04XAF/E04XAA.

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

On entry: the point x at which the objective function (and gradients if MODE = 1) is to be evaluated.

4: OBJF – REAL (KIND=nag_wp) *Output*

On exit: must be set to the value of the objective function.

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

On exit: if MODE = 1, OBJGRD(j) must contain the value of the first derivative with respect to x.

If MODE $\neq$ 1, OBJGRD need not be set.

6: NSTATE – INTEGER *Input*

On entry: will be set to 1 on the first call of OBJFUN by E04XAF/E04XAA, and is 0 for all subsequent calls. Thus, if you wish, NSTATE may be tested within OBJFUN in order to perform certain calculations once only. For example you may read data.

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

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

OBJFUN is called with the arguments IUSER and RUSER as supplied to E04XAF/E04XAA. You should use the arrays IUSER and RUSER to supply information to OBJFUN.

OBJFUN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which E04XAF/E04XAA is called. Arguments denoted as *Input* must **not** be changed by this procedure.

7: LDH – INTEGER *Input*

On entry: the first dimension of the array H as declared in the (sub)program from which E04XAF/E04XAA is called.

Constraint: LDH $\geq$ N.

8: HFORW(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the initial trial interval for computing the appropriate partial derivative to the jth variable.

If HFORW$(j) \leq 0.0$, then the initial trial interval is computed by E04XAF/E04XAA (see Section 3).

On exit: HFORW(j) is the best interval found for computing a forward-difference approximation to the appropriate partial derivative for the jth variable.

9: OBJF – REAL (KIND=nag_wp) *Output*

On exit: the value of the objective function evaluated at the input vector in X.

10: OBJGRD(N) – REAL (KIND=nag_wp) array *Output*

On exit: if MODE $= 0$ or 2, OBJGRD(j) contains the best estimate of the first partial derivative for the jth variable.

If MODE $= 1$, OBJGRD(j) contains the first partial derivative for the jth variable evaluated at the input vector in X.

11: HCNTRL(N) – REAL (KIND=nag_wp) array *Output*

On exit: HCNTRL(j) is the best interval found for computing a central-difference approximation to the appropriate partial derivative for the jth variable.

12: H(LDH, ∗) – REAL (KIND=nag_wp) array *Output*

Note: the second dimension of the array H must be at least 1 if MODE $= 0$ and at least N if MODE $= 1$ or 2.

On exit: if MODE $= 0$, the estimated Hessian diagonal elements are contained in the first column of this array.

If MODE $= 1$ or 2, the estimated Hessian matrix is contained in the leading n by n part of this array.

13: IWARN – INTEGER *Output*

On exit: IWARN $= 0$ on successful exit.

If the value of EPSRF on entry is too small or too large then IWARN is set to 1 or 2 respectively on exit and the default value for EPSRF is used within E04XAF/E04XAA.

If MSGLVL > 0 then warnings will be printed if EPSRF is too small or too large.

14: WORK($*$) – REAL (KIND=nag_wp) array *Workspace*

Note: the dimension of the array WORK must be at least N if MODE $= 0$ and at least N $\times$ (N $+ 1$) if MODE $= 1$ or 2.

15: IUSER($*$) – INTEGER array *User Workspace*

16: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by E04XAF/E04XAA, but are passed directly to OBJFUN and should be used to pass information to this routine.

17: INFO(N) – INTEGER array *Output*

On exit: INFO(j) represents diagnostic information on variable j. (See Section 6 for more details.)

18: IFAIL – INTEGER *Input/Output*

Note: *for E04XAA, IFAIL does not occur in this position in the argument list. See the additional arguments described below.*

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, if you are not familiar with this argument, 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).

Note: *the following are additional arguments for specific use with E04XAA. Users of E04XAF therefore need not read the remainder of this description*.

19: LWSAV(1) – LOGICAL array *Communication Array*

20: IWSAV(1) – INTEGER array *Communication Array*

21: RWSAV(1) – REAL (KIND=nag_wp) array *Communication Array*

These arguments are no longer required by E04XAF/E04XAA.

22: IFAIL – INTEGER *Input/Output*

Note: see the argument description for IFAIL above.

6 Error Indicators and Warnings

On exit from E04XAF/E04XAA both diagnostic arguments INFO and IFAIL should be tested. IFAIL represents an overall diagnostic indicator, whereas the integer array INFO represents diagnostic information on each variable.

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 < 0

A negative value of IFAIL indicates an exit from E04XAF/E04XAA because you set MODE negative in OBJFUN. The value of IFAIL will be the same as your setting of MODE.

IFAIL $= 1$

On entry, one or more of the following conditions are satisfied: N < 1, LDH $<$ N or MODE is invalid.

IFAIL $= 2$

One or more variables have a nonzero INFO value. This may not necessarily represent an unsuccessful exit – see diagnostic information on INFO.

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.

Diagnostic information returned via INFO is as follows:

INFO $= 1$

The appropriate function appears to be constant. HFORW(i) is set to the initial trial interval value (see Section 3) corresponding to a well-scaled problem and `Error est.` in the printed output is set to zero. This value occurs when the estimated relative condition error in the first derivative approximation is unacceptably large for every value of the finite difference interval. If this happens when the function is not constant the initial interval may be too small; in this case, it may be worthwhile to rerun E04XAF/E04XAA with larger initial trial interval values supplied in HFORW (see Section 3). This error may also occur if the function evaluation includes an inordinately large constant term or if EPSRF is too large.

INFO $= 2$

The appropriate function appears to be linear or odd. HFORW(i) is set to the smallest interval with acceptable bounds on the relative condition error in the forward- and backward-difference estimates. In this case, the estimated relative condition error in the second derivative approximation remained large for every trial interval, but the estimated error in the first derivative approximation was acceptable for at least one interval. If the function is not linear or odd the relative condition error in the second derivative may be decreasing very slowly, it may be worthwhile to rerun E04XAF/E04XAA with larger initial trial interval values supplied in HFORW (see Section 3).

INFO $= 3$

The second derivative of the appropriate function appears to be so large that it cannot be reliably estimated (i.e., near a singularity). HFORW(i) is set to the smallest trial interval.

This value occurs when the relative condition error estimate in the second derivative remained very small for every trial interval.

If the second derivative is not large the relative condition error in the second derivative may be increasing very slowly. It may be worthwhile to rerun E04XAF/E04XAA with smaller initial trial interval values supplied in HFORW (see Section 3). This error may also occur when the given value of EPSRF is not a good estimate of a bound on the absolute error in the appropriate function (i.e., EPSRF is too small).

INFO $= 4$

The algorithm terminated with an apparently acceptable estimate of the second derivative. However the forward-difference estimates of the appropriate first derivatives (computed with the final estimate of the 'optimal' forward-difference interval) and the central difference estimates (computed with the interval used to compute the final estimate of the second derivative) do not agree to half a decimal place. The usual reason that the forward- and central-difference estimates fail to agree is that the first derivative is small.

If the first derivative is not small, it may be helpful to execute the procedure at a different point.

7 Accuracy

If IFAIL $= 0$ on exit the algorithm terminated successfully, i.e., the forward-difference estimates of the appropriate first derivatives (computed with the final estimate of the 'optimal' forward-difference interval h_F) and the central-difference estimates (computed with the interval h_ϕ used to compute the final estimate of the second derivative) agree to at least half a decimal place.

In short word length implementations when computing the full Hessian matrix given function values only (i.e., MODE $= 2$) the elements of the computed Hessian will have at best 1 to 2 figures of accuracy.

8 Parallelism and Performance

E04XAF/E04XAA is not threaded in any implementation.

9 Further Comments

To evaluate an acceptable set of finite difference intervals for a well-scaled problem, the routine will require around two function evaluations per variable; in a badly scaled problem however, as many as six function evaluations per variable may be needed.

If you request the full Hessian matrix supplying both function and gradients (i.e., MODE $= 1$) or function only (i.e., MODE $= 2$) then a further N or $3 \times \mathrm{N} \times (\mathrm{N}+1)/2$ function evaluations respectively are required.

9.1 Description of the Printed Output

The following is a description of the printed output from E04XAF/E04XAA as controlled by the argument MSGLVL.

Output when MSGLVL $= 1$ is as follows:

`J`	number of variable for which the difference interval has been computed.
`X`(j)	jth variable of x as set by you.
`F. dif. int.`	the best interval found for computing a forward-difference approximation to the appropriate partial derivative with respect to the jth variable.
`C. dif. int.`	the best interval found for computing a central-difference approximation to the appropriate partial derivative with respect to the jth variable.
`Error est.`	a bound on the estimated error in the final forward-difference approximation. When INFO$(j) = 1$, `Error est.` is set to zero.

`Grad. est.`	best estimate of the first partial derivative with respect to the jth variable.
`Hess diag est.`	best estimate of the second partial derivative with respect to the jth variable.
`fun evals.`	the number of function evaluations used to compute the final difference intervals for the jth variable.
`info`(j)	the value of INFO for the jth variable.

10 Example

This example computes the gradient vector and the Hessian matrix of the following function:

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

at the point $(2, -1, 1, 1)$.

10.1 Program Text

the following program illustrates the use of E04XAF. An equivalent program illustrating the use of E04XAA is available with the supplied Library and is also available from the NAG web site.

```
!   E04XAF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module e04xafe_mod

!     E04XAF 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                           :: objfun
!     .. Parameters ..
      Integer, Parameter, Public       :: n = 4, nout = 6
      Integer, Parameter, Public       :: lhes = n
      Integer, Parameter, Public       :: lwork = n*n + n
    Contains
      Subroutine objfun(mode,n,x,objf,objgrd,nstate,iuser,ruser)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: objf
        Integer, Intent (Inout)        :: mode
        Integer, Intent (In)           :: n, nstate
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: objgrd(n)
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (In) :: x(n)
        Integer, Intent (Inout)        :: iuser(*)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: a, b, c, d
!       .. Executable Statements ..
        a = x(1) + 10.0E0_nag_wp*x(2)
        b = x(3) - x(4)
        c = x(2) - 2.0E0_nag_wp*x(3)
        d = x(1) - x(4)
        objf = a**2 + 5.0E0_nag_wp*b**2 + c**4 + 10.0E0_nag_wp*d**4

        If (mode==1) Then
          objgrd(1) = 4.0E1_nag_wp*x(1)**3 + 2.0E0_nag_wp*x(1) -              &
            1.2E2_nag_wp*x(4)*x(1)**2 + 1.2E2_nag_wp*x(1)*x(4)**2 +           &
            2.0E1_nag_wp*x(2) - 4.0E1_nag_wp*x(4)**3
          objgrd(2) = 2.0E2_nag_wp*x(2) + 2.0E1_nag_wp*x(1) +                 &
            4.0E0_nag_wp*x(2)**3 + 4.8E1_nag_wp*x(2)*x(3)**2 -                &
            2.4E1_nag_wp*x(3)*x(2)**2 - 32.0E0_nag_wp*x(3)**3
```

```
          objgrd(3) = 1.0E1_nag_wp*x(3) - 1.0E1_nag_wp*x(4) -                  &
            8.0E0_nag_wp*x(2)**3 + 4.8E1_nag_wp*x(3)*x(2)**2 -                 &
            9.6E1_nag_wp*x(2)*x(3)**2 + 6.4E1_nag_wp*x(3)**3
          objgrd(4) = 1.0E1_nag_wp*x(4) - 1.0E1_nag_wp*x(3) -                  &
            4.0E1_nag_wp*x(1)**3 + 1.2E2_nag_wp*x(4)*x(1)**2 -                 &
            1.2E2_nag_wp*x(1)*x(4)**2 + 4.0E1_nag_wp*x(4)**3
        End If

        Return

      End Subroutine objfun
    End Module e04xafe_mod
    Program e04xafe

!     E04XAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04xaf, nag_wp
      Use e04xafe_mod, Only: lhes, lwork, n, nout, objfun
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: epsrf, objf
      Integer                          :: i, ifail, imode, iwarn, mode, msglvl
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: hcntrl(n), hesian(lhes,n), hforw(n), &
                                          objgrd(n), user(1), work(lwork),     &
                                          x(n)
      Integer                          :: info(n), iuser(1)
      Character (80)                   :: rc(3)
!     .. Executable Statements ..
      Write (nout,*) 'E04XAF Example Program Results'

      msglvl = 0

!     Set the point at which the derivatives are to be estimated.

      x(1:n) = (/2.0E0_nag_wp,-1.0E0_nag_wp,1.0E0_nag_wp,1.0E0_nag_wp/)

      rc(1) = 'Find gradients and Hessian diagonals given function only'
      rc(2) = 'Find Hessian matrix given function and gradients'
      rc(3) = 'Find gradients and Hessian matrix given function only'

!     Take default value of EPSRF.

      epsrf = -1.0E0_nag_wp

!     Illustrate the different values of MODE.

loop: Do imode = 0, 2
        mode = imode

!       Set HFORW(I) = -1.0 so that E04XAF computes the initial trial
!       interval.

        hforw(1:n) = -1.0E0_nag_wp

        ifail = -1
        Call e04xaf(msglvl,n,epsrf,x,mode,objfun,lhes,hforw,objf,objgrd,       &
          hcntrl,hesian,iwarn,work,iuser,user,info,ifail)

        Select Case (ifail)
        Case (0,2)
          Write (nout,99999) rc(mode+1), mode
          Write (nout,99998) 'Function value is ', objf

          If (mode==1) Then
            Write (nout,*) 'Gradient vector is'
            Write (nout,99997) objgrd(1:n)
          Else
            Write (nout,*) 'Estimated gradient vector is'
```

```
            Write (nout,99997) objgrd(1:n)
          End If

          If (mode==0) Then
            Write (nout,*) 'Estimated Hessian matrix diagonal is'
            Write (nout,99997) hesian(1:n,1)
          Else
            Write (nout,*) 'Estimated Hessian matrix (machine dependent) is'
            Write (nout,99997)(hesian(i,1:n),i=1,n)
          End If

        Case Default
          Exit loop
        End Select

      End Do loop

99999 Format (1X,/,1X,A,/,1X,'( i.e. MODE =',I2,' ).')
99998 Format (1X,A,1P,E12.3)
99997 Format (4(1X,1P,E12.3))
    End Program e04xafe
```

10.2 Program Data

None.

10.3 Program Results

```
 E04XAF Example Program Results

 Find gradients and Hessian diagonals given function only
 ( i.e. MODE = 0 ).
 Function value is    1.550E+02
 Estimated gradient vector is
    2.400E+01   -2.680E+02    2.160E+02   -4.000E+01
 Estimated Hessian matrix diagonal is
    1.220E+02    3.080E+02    4.420E+02    1.300E+02

 Find Hessian matrix given function and gradients
 ( i.e. MODE = 1 ).
 Function value is    1.550E+02
 Gradient vector is
    2.400E+01   -2.680E+02    2.160E+02   -4.000E+01
 Estimated Hessian matrix (machine dependent) is
    1.220E+02    2.000E+01    0.000E+00   -1.200E+02
    2.000E+01    3.080E+02   -2.160E+02    0.000E+00
    0.000E+00   -2.160E+02    4.420E+02   -1.000E+01
   -1.200E+02    0.000E+00   -1.000E+01    1.300E+02

 Find gradients and Hessian matrix given function only
 ( i.e. MODE = 2 ).
 Function value is    1.550E+02
 Estimated gradient vector is
    2.400E+01   -2.680E+02    2.160E+02   -4.000E+01
 Estimated Hessian matrix (machine dependent) is
    1.220E+02    2.000E+01   -4.404E-03   -1.200E+02
    2.000E+01    3.080E+02   -2.160E+02    6.605E-03
   -4.404E-03   -2.160E+02    4.420E+02   -1.000E+01
   -1.200E+02    6.605E-03   -1.000E+01    1.300E+02
```