NAG Library Routine Document

E04YAF

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

E04YAF checks that a user-supplied subroutine for evaluating a vector of functions and the matrix of their first derivatives produces derivative values which are consistent with the function values calculated.

2 Specification

```
SUBROUTINE E04YAF (M, N, LSQFUN, X, FVEC, FJAC, LDFJAC, IW, LIW, W, LW,    &
                   IFAIL)

INTEGER             M, N, LDFJAC, IW(LIW), LIW, LW, IFAIL
REAL (KIND=nag_wp) X(N), FVEC(M), FJAC(LDFJAC,N), W(LW)
EXTERNAL            LSQFUN
```

3 Description

Routines for minimizing a sum of squares of m nonlinear functions (or 'residuals'), $f_i(x_1, x_2, \ldots, x_n)$, for $i = 1, 2, \ldots, m$ and $m \geq n$, may require you to supply a subroutine to evaluate the f_i and their first derivatives. E04YAF checks the derivatives calculated by such user-supplied subroutines, e.g., routines of the form required for E04GBF, E04GDF and E04HEF. As well as the routine to be checked (LSQFUN), you must supply a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the check will be made. E04YAF is essentially identical to CHKLSJ in the NPL Algorithms Library.

E04YAF first calls LSQFUN to evaluate the $f_i(x)$ and their first derivatives, and uses these to calculate the sum of squares $F(x) = \sum_{i=1}^{m} [f_i(x)]^2$, and its first derivatives $g_j = \left.\dfrac{\partial F}{\partial x_j}\right|_x$, for $j = 1, 2, \ldots, n$. The components of g 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: M – INTEGER *Input*

2: N – INTEGER *Input*

On entry: the number m of residuals, $f_i(x)$, and the number n of variables, x_j.

Constraint: $1 \leq \mathrm{N} \leq \mathrm{M}$.

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

LSQFUN must calculate the vector of values $f_i(x)$ and their first derivatives $\dfrac{\partial f_i}{\partial x_j}$ at any point x. (The minimization routines mentioned in Section 3 give you the option of resetting an argument to terminate immediately. E04YAF will also terminate immediately, without finishing the checking process, if the argument in question is reset.)

The specification of LSQFUN is:

```
SUBROUTINE LSQFUN (IFLAG, M, N, XC, FVEC, FJAC, LDFJAC, IW, LIW,     &
                   W, LW)

INTEGER            IFLAG, M, N, LDFJAC, IW(LIW), LIW, LW
REAL (KIND=nag_wp) XC(N), FVEC(M), FJAC(LDFJAC,N), W(LW)
```

1: IFLAG – INTEGER *Input/Output*

On entry: to LSQFUN, IFLAG will be set to 2.

On exit: if you reset IFLAG to some negative number in LSQFUN and return control to E04YAF, the routine will terminate immediately with IFAIL set to your setting of IFLAG.

2: M – INTEGER *Input*

On entry: the numbers m of residuals.

3: N – INTEGER *Input*

On entry: the numbers n of variables.

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

On entry: x, the point at which the values of the f_i and the $\dfrac{\partial f_i}{\partial x_j}$ are required.

5: FVEC(M) – REAL (KIND=nag_wp) array *Output*

On exit: unless IFLAG is reset to a negative number, FVEC(i) must contain the value of f_i at the point x, for $i = 1, 2, \ldots, m$.

6: FJAC(LDFJAC, N) – REAL (KIND=nag_wp) array *Output*

On exit: unless IFLAG is reset to a negative number, FJAC(i, j) must contain the value of $\dfrac{\partial f_i}{\partial x_j}$ at the point x, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

7: LDFJAC – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which E04YAF is called.

8: IW(LIW) – INTEGER array *Workspace*

9: LIW – INTEGER *Input*

10: W(LW) – REAL (KIND=nag_wp) array *Workspace*

11: LW – INTEGER *Input*

These arguments are present so that LSQFUN will be of the form required by the minimization routines mentioned in Section 3. LSQFUN is called with the same arguments IW, LIW, W, LW as in the call to E04YAF. If the recommendation in the minimization routine document is followed, you will have no reason to examine or change the elements of IW or W. In any case, LSQFUN **must not change** the first $3 \times \mathrm{N} + \mathrm{M} + \mathrm{M} \times \mathrm{N}$ elements of W.

LSQFUN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04YAF is called. Arguments 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 be set to the coordinates of a suitable point at which to check the derivatives calculated by LSQFUN. 'Obvious' settings, such as 0 or 1, should not be used since, at such particular points, incorrect terms may take correct values (particularly zero), so that errors can go undetected. For a similar reason, it is preferable that no two elements of X should have the same value.

5: FVEC(M) – REAL (KIND=nag_wp) array *Output*

On exit: unless you set IFLAG negative in the first call of LSQFUN, FVEC(i) contains the value of f_i at the point supplied by you in X, for $i = 1, 2, \ldots, m$.

6: FJAC(LDFJAC, N) – REAL (KIND=nag_wp) array *Output*

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

7: LDFJAC – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which E04YAF is called.

Constraint: LDFJAC $\geq$ M.

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

This array appears in the argument list purely so that, if E04YAF is called by another library routine, the library routine can pass quantities to LSQFUN via IW. IW is not examined or changed by E04YAF. In general you must provide an array IW, but are advised not to use it.

9: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04YAF is called.

Constraint: LIW ≥ 1.

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

11: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04YAF is called.

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

12: 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: E04YAF 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 E04YAF because you have set IFLAG negative in LSQFUN. The setting of IFAIL will be the same as your setting of IFLAG. The check on LSQFUN will not have been completed.

IFAIL $= 1$

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

IFAIL $= 2$

You should check carefully the derivation and programming of expressions for the $\dfrac{\partial f_i}{\partial x_j}$, because it is very unlikely that LSQFUN 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

E04YAF is not threaded in any implementation.

9 Further Comments

E04YAF calls LSQFUN three times.

Before using E04YAF to check the calculation of the first derivatives, you should be confident that LSQFUN is calculating the residuals correctly.

E04YAF only checks the derivatives calculated by a user-supplied routine when IFLAG $= 2$. So, if LSQFUN 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), LSQFUN produces the same values for the $\dfrac{\partial f_i}{\partial x_j}$ when IFLAG is set to 1 as when IFLAG is set to 2.

10 Example

Suppose that it is intended to use E04GBF or E04GDF to find least squares estimates of x_1, x_2 and x_3 in the model

$$y = x_1 + \frac{t_1}{x_2 t_2 + x_3 t_3}$$

using the 15 sets of data given in the following table.

y	t_1	t_2	t_3
0.14	1.0	15.0	1.0
0.18	2.0	14.0	2.0
0.22	3.0	13.0	3.0
0.25	4.0	12.0	4.0
0.29	5.0	11.0	5.0
0.32	6.0	10.0	6.0
0.35	7.0	9.0	7.0
0.39	8.0	8.0	8.0
0.37	9.0	7.0	7.0
0.58	10.0	6.0	6.0
0.73	11.0	5.0	5.0
0.96	12.0	4.0	4.0
1.34	13.0	3.0	3.0
2.10	14.0	2.0	2.0
4.39	15.0	1.0	1.0

The following program could be used to check the first derivatives calculated by LSQFUN. (The tests of whether IFLAG $= 0$ or 1 in LSQFUN are present ready for when LSQFUN is called by E04GBF or E04GDF. E04YAF will always call LSQFUN with IFLAG set to 2.)

10.1 Program Text

```
!   E04YAF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module e04yafe_mod

!     E04YAF 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                             :: lsqfun
!     .. Parameters ..
      Integer, Parameter, Public         :: liw = 1, mdec = 15, ndec = 3,        &
                                            nin = 5, nout = 6
      Integer, Parameter, Public         :: ldfjac = mdec
```

```
      Integer, Parameter, Public       :: lw = 3*ndec + mdec + mdec*ndec
!     .. Local Arrays ..
      Real (Kind=nag_wp), Public, Save :: t(mdec,ndec), y(mdec)
    Contains
      Subroutine lsqfun(iflag,m,n,xc,fvec,fjac,ldfjac,iw,liw,w,lw)

!       Routine to evaluate the residuals and their 1st derivatives

!       .. Scalar Arguments ..
        Integer, Intent (Inout)        :: iflag
        Integer, Intent (In)           :: ldfjac, liw, lw, m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: fjac(ldfjac,n), w(lw)
        Real (Kind=nag_wp), Intent (Out) :: fvec(m)
        Real (Kind=nag_wp), Intent (In) :: xc(n)
        Integer, Intent (Inout)        :: iw(liw)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: denom, dummy
        Integer                        :: i
!       .. Executable Statements ..
        Do i = 1, m
          denom = xc(2)*t(i,2) + xc(3)*t(i,3)

          If (iflag/=1) Then
            fvec(i) = xc(1) + t(i,1)/denom - y(i)
          End If

          If (iflag/=0) Then
            fjac(i,1) = 1.0E0_nag_wp
            dummy = -1.0E0_nag_wp/(denom*denom)
            fjac(i,2) = t(i,1)*t(i,2)*dummy
            fjac(i,3) = t(i,1)*t(i,3)*dummy
          End If

        End Do

        Return

      End Subroutine lsqfun
    End Module e04yafe_mod
    Program e04yafe

!     E04YAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04yaf, nag_wp
      Use e04yafe_mod, Only: ldfjac, liw, lsqfun, lw, mdec, ndec, nin, nout,   &
                             t, y
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Integer                          :: i, ifail, m, n
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: fjac(ldfjac,ndec), fvec(mdec),       &
                                          w(lw), x(ndec)
      Integer                          :: iw(liw)
!     .. Executable Statements ..
      Write (nout,*) 'E04YAF Example Program Results'

!     Skip heading in data file
      Read (nin,*)

      n = ndec
      m = mdec

!     Observations of TJ (J = 1, 2, ..., n) are held in T(I, J)
!     (I = 1, 2, ..., m)

      Do i = 1, m
        Read (nin,*) y(i), t(i,1:n)
      End Do
```

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

      x(1:n) = (/0.19E0_nag_wp,-1.34E0_nag_wp,0.88E0_nag_wp/)

      Write (nout,*)
      Write (nout,*) 'The test point is'
      Write (nout,99999) x(1:n)

      ifail = -1
      Call e04yaf(m,n,lsqfun,x,fvec,fjac,ldfjac,iw,liw,w,lw,ifail)

      If (ifail>=0 .And. ifail/=1) Then

        Select Case (ifail)
        Case (0)
          Write (nout,*)
          Write (nout,*) '1st derivatives are consistent with residual values'
        Case (2)
          Write (nout,*)
          Write (nout,*) 'Probable error in calculation of 1st derivatives'
        End Select

        Write (nout,*)
        Write (nout,*) 'At the test point, LSQFUN gives'
        Write (nout,*)
        Write (nout,*) '      Residuals                    1st derivatives'
        Write (nout,99998)(fvec(i),fjac(i,1:n),i=1,m)
      End If

99999 Format (1X,4F10.5)
99998 Format (1X,1P,4E15.3)
    End Program e04yafe
```

10.2 Program Data

```
E04YAF Example Program Data
 0.14  1.0 15.0  1.0
 0.18  2.0 14.0  2.0
 0.22  3.0 13.0  3.0
 0.25  4.0 12.0  4.0
 0.29  5.0 11.0  5.0
 0.32  6.0 10.0  6.0
 0.35  7.0  9.0  7.0
 0.39  8.0  8.0  8.0
 0.37  9.0  7.0  7.0
 0.58 10.0  6.0  6.0
 0.73 11.0  5.0  5.0
 0.96 12.0  4.0  4.0
 1.34 13.0  3.0  3.0
 2.10 14.0  2.0  2.0
 4.39 15.0  1.0  1.0
```

10.3 Program Results

```
 E04YAF Example Program Results

 The test point is
    0.19000  -1.34000   0.88000

 1st derivatives are consistent with residual values

 At the test point, LSQFUN gives

       Residuals                  1st derivatives
      -2.029E-03      1.000E+00     -4.061E-02     -2.707E-03
      -1.076E-01      1.000E+00     -9.689E-02     -1.384E-02
      -2.330E-01      1.000E+00     -1.785E-01     -4.120E-02
      -3.785E-01      1.000E+00     -3.043E-01     -1.014E-01
```

```
-5.836E-01      1.000E+00     -5.144E-01     -2.338E-01
-8.689E-01      1.000E+00     -9.100E-01     -5.460E-01
-1.346E+00      1.000E+00     -1.810E+00     -1.408E+00
-2.374E+00      1.000E+00     -4.726E+00     -4.726E+00
-2.975E+00      1.000E+00     -6.076E+00     -6.076E+00
-4.013E+00      1.000E+00     -7.876E+00     -7.876E+00
-5.323E+00      1.000E+00     -1.040E+01     -1.040E+01
-7.292E+00      1.000E+00     -1.418E+01     -1.418E+01
-1.057E+01      1.000E+00     -2.048E+01     -2.048E+01
-1.713E+01      1.000E+00     -3.308E+01     -3.308E+01
-3.681E+01      1.000E+00     -7.089E+01     -7.089E+01
```