NAG Library Routine Document

E04HYF

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

E04HYF is an easy-to-use modified Gauss–Newton algorithm for finding an unconstrained minimum of a sum of squares of m nonlinear functions in n variables $(m \geq n)$. First and second derivatives are required.

It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

2 Specification

```
SUBROUTINE E04HYF (M, N, LSFUN2, LSHES2, X, FSUMSQ, W, LW, IUSER, RUSER,    &
                   IFAIL)

INTEGER             M, N, LW, IUSER(*), IFAIL
REAL (KIND=nag_wp) X(N), FSUMSQ, W(LW), RUSER(*)
EXTERNAL            LSFUN2, LSHES2
```

3 Description

E04HYF is similar to the subroutine LSSDN2 in the NPL Algorithms Library. It is applicable to problems of the form:

$$\text{Minimize } F(x) = \sum_{i=1}^{m} [f_i(x)]^2$$

where $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ and $m \geq n$. (The functions $f_i(x)$ are often referred to as 'residuals'.)

You must supply a subroutine to evaluate the residuals and their first derivatives at any point x, and a subroutine to evaluate the elements of the second derivative term of the Hessian matrix of $F(x)$.

Before attempting to minimize the sum of squares, the algorithm checks the user-supplied subroutines for consistency. Then, from a starting point supplied by you, a sequence of points is generated which is intended to converge to a local minimum of the sum of squares. These points are generated using estimates of the curvature of $F(x)$.

4 References

Gill P E and Murray W (1978) Algorithms for the solution of the nonlinear least squares problem *SIAM J. Numer. Anal.* **15** 977–992

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: LSFUN2 – SUBROUTINE, supplied by the user. *External Procedure*

You must supply this routine to calculate the vector of values $f_i(x)$ and the Jacobian matrix of first derivatives $\dfrac{\partial f_i}{\partial x_j}$ at any point x. It should be tested separately before being used in conjunction with E04HYF (see the E04 Chapter Introduction).

The specification of LSFUN2 is:

```
SUBROUTINE LSFUN2 (M, N, XC, FVEC, FJAC, LDFJAC, IUSER, RUSER)
INTEGER            M, N, LDFJAC, IUSER(*)
REAL (KIND=nag_wp) XC(N), FVEC(M), FJAC(LDFJAC,N), RUSER(*)
```

Important: the dimension declaration for FJAC must contain the variable LDFJAC, not an integer constant.

1: M – INTEGER *Input*

On entry: m, the numbers of residuals.

2: N – INTEGER *Input*

On entry: n, the numbers of variables.

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

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

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

On exit: FVEC(i) must be set to the value of f_i at the point x, for $i = 1, 2, \ldots, m$.

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

On exit: FJAC(i, j) must be set to the value of $\dfrac{\partial f_i}{\partial x_j}$ at the point x, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

6: LDFJAC – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which E04HYF is called.

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

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

LSFUN2 is called with the arguments IUSER and RUSER as supplied to E04HYF. You should use the arrays IUSER and RUSER to supply information to LSFUN2.

LSFUN2 must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which E04HYF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

4: LSHES2 – SUBROUTINE, supplied by the user. *External Procedure*

You must supply this routine to calculate the elements of the symmetric matrix

$$B(x) = \sum_{i=1}^{m} f_i(x) G_i(x),$$

at any point x, where $G_i(x)$ is the Hessian matrix of $f_i(x)$. It should be tested separately before being used in conjunction with E04HYF (see the E04 Chapter Introduction).

The specification of LSHES2 is:

```
SUBROUTINE LSHES2 (M, N, FVEC, XC, B, LB, IUSER, RUSER)
INTEGER             M, N, LB, IUSER(*)
REAL (KIND=nag_wp) FVEC(M), XC(N), B(LB), RUSER(*)
```

1: M – INTEGER *Input*

On entry: m, the number of residuals.

2: N – INTEGER *Input*

On entry: n, the number of residuals.

3: FVEC(M) – REAL (KIND=nag_wp) array *Input*

On entry: the value of the residual f_i at the point x, for $i = 1, 2, \ldots, m$, so that the values of the f_i can be used in the calculation of the elements of B.

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

On entry: the point x at which the elements of B are to be evaluated.

5: B(LB) – REAL (KIND=nag_wp) array *Output*

On exit: must contain the lower triangle of the matrix $B(x)$, evaluated at the point x, stored by rows. (The upper triangle is not required because the matrix is symmetric.) More precisely, B$(j(j-1)/2+k)$ must contain $\displaystyle\sum_{i=1}^{m} f_i \frac{\partial^2 f_i}{\partial x_j \partial x_k}$ evaluated at the point x, for $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, j$.

6: LB – INTEGER *Input*

On entry: the length of the array B.

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

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

LSHES2 is called with the arguments IUSER and RUSER as supplied to E04HYF. You should use the arrays IUSER and RUSER to supply information to LSHES2.

LSHES2 must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which E04HYF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

5: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: X(j) must be set to a guess at the jth component of the position of the minimum, for $j = 1, 2, \ldots, n$. The routine checks LSFUN2 and LSHES2 at the starting point and so is more likely to detect any error in your routines if the initial X(j) are nonzero and mutually distinct.

On exit: the lowest point found during the calculations. Thus, if IFAIL $= 0$ on exit, X(j) is the jth component of the position of the minimum.

6: FSUMSQ – REAL (KIND=nag_wp) *Output*

On exit: the value of the sum of squares, $F(x)$, corresponding to the final point stored in X.

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

8: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04HYF is called.

Constraints:

if $\mathrm{N} > 1$, $\mathrm{LW} \geq 8 \times \mathrm{N} + 2 \times \mathrm{N} \times \mathrm{N} + 2 \times \mathrm{M} \times \mathrm{N} + 3 \times \mathrm{M}$;
if $\mathrm{N} = 1$, $\mathrm{LW} \geq 11 + 5 \times \mathrm{M}$.

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

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

IUSER and RUSER are not used by E04HYF, but are passed directly to LSFUN2 and LSHES2 and should be used to pass information to these routines.

11: 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 $\mathrm{IFAIL} \neq 0$ 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 $\mathrm{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Note: E04HYF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

On entry, $\mathrm{N} < 1$,
or $\mathrm{M} < \mathrm{N}$,
or $\mathrm{LW} < 8 \times \mathrm{N} + 2 \times \mathrm{N} \times \mathrm{N} + 2 \times \mathrm{M} \times \mathrm{N} + 3 \times \mathrm{M}$, when $\mathrm{N} > 1$,
or $\mathrm{LW} < 11 + 5 \times \mathrm{M}$, when $\mathrm{N} = 1$.

$\mathrm{IFAIL} = 2$

There have been $50 \times n$ calls of LSFUN2, yet the algorithm does not seem to have converged. This may be due to an awkward function or to a poor starting point, so it is worth restarting E04HYF from the final point held in X.

$\mathrm{IFAIL} = 3$

The final point does not satisfy the conditions for acceptance as a minimum, but no lower point could be found.

$\mathrm{IFAIL} = 4$

An auxiliary routine has been unable to complete a singular value decomposition in a reasonable number of sub-iterations.

IFAIL $= 5$
IFAIL $= 6$
IFAIL $= 7$
IFAIL $= 8$

There is some doubt about whether the point Xx found by E04HYF is a minimum of $F(x)$. The degree of confidence in the result decreases as IFAIL increases. Thus, when IFAIL $= 5$, it is probable that the final x gives a good estimate of the position of a minimum, but when IFAIL $= 8$ it is very unlikely that the routine has found a minimum.

IFAIL $= 9$

It is very likely that you have made an error in forming the derivatives $\dfrac{\partial f_i}{\partial x_j}$ in LSFUN2.

IFAIL $= 10$

It is very likely that you have made an error in forming the quantities B_{jk} in LSHES2.

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.

If you are not satisfied with the result (e.g., because IFAIL lies between 3 and 8), it is worth restarting the calculations from a different starting point (not the point at which the failure occurred) in order to avoid the region which caused the failure. Repeated failure may indicate some defect in the formulation of the problem.

7 Accuracy

If the problem is reasonably well scaled and a successful exit is made, then, for a computer with a mantissa of t decimals, one would expect to get about $t/2 - 1$ decimals accuracy in the components of x and between $t - 1$ (if $F(x)$ is of order 1 at the minimum) and $2t - 2$ (if $F(x)$ is close to zero at the minimum) decimals accuracy in $F(x)$.

8 Parallelism and Performance

E04HYF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

E04HYF 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

The number of iterations required depends on the number of variables, the number of residuals and their behaviour, and the distance of the starting point from the solution. The number of multiplications performed per iteration of E04HYF varies, but for $m \gg n$ is approximately $n \times m^2 + O(n^3)$. In addition, each iteration makes at least one call of LSFUN2 and some iterations may call LSHES2. So, unless the residuals and their derivatives can be evaluated very quickly, the run time will be dominated by the time spent in LSFUN2 (and, to a lesser extent, in LSHES2).

Ideally, the problem should be scaled so that the minimum value of the sum of squares is in the range $(0, +1)$ and so that at points a unit distance away from the solution the sum of squares is approximately a unit value greater than at the minimum. It is unlikely that you will be able to follow these recommendations very closely, but it is worth trying (by guesswork), as sensible scaling will reduce the difficulty of the minimization problem, so that E04HYF will take less computer time.

When the sum of squares represents the goodness-of-fit of a nonlinear model to observed data, elements of the variance-covariance matrix of the estimated regression coefficients can be computed by a subsequent call to E04YCF, using information returned in segments of the workspace array W. See E04YCF for further details.

10 Example

This example finds 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 program uses $(0.5, 1.0, 1.5)$ as the initial guess at the position of the minimum.

10.1 Program Text

```
!   E04HYF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module e04hyfe_mod

!     E04HYF 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                           :: lsfun2, lshes2
```

```
!     .. Parameters ..
      Integer, Parameter, Public       :: m = 15, n = 3, nin = 5, nout = 6,    &
                                          nt = 3
      Integer, Parameter, Public       :: lw = 8*n + 2*n*n + 2*m*n + 3*m
!     .. Local Arrays ..
      Real (Kind=nag_wp), Public, Save :: t(m,nt), y(m)
    Contains
      Subroutine lsfun2(m,n,xc,fvec,fjac,ldfjac,iuser,ruser)
!       Routine to evaluate the residuals and their 1st derivatives.

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: ldfjac, m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: fjac(ldfjac,n), ruser(*)
        Real (Kind=nag_wp), Intent (Out) :: fvec(m)
        Real (Kind=nag_wp), Intent (In) :: xc(n)
        Integer, Intent (Inout)        :: iuser(*)
!       .. 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)
          fvec(i) = xc(1) + t(i,1)/denom - y(i)
          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 Do

        Return

      End Subroutine lsfun2
      Subroutine lshes2(m,n,fvec,xc,b,lb,iuser,ruser)
!       Routine to compute the lower triangle of the matrix B
!       (stored by rows in the array B).

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: lb, m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: b(lb)
        Real (Kind=nag_wp), Intent (In) :: fvec(m), xc(n)
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Integer, Intent (Inout)        :: iuser(*)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: dummy, sum22, sum32, sum33
        Integer                        :: i
!       .. Executable Statements ..
        b(1) = 0.0E0_nag_wp
        b(2) = 0.0E0_nag_wp
        sum22 = 0.0E0_nag_wp
        sum32 = 0.0E0_nag_wp
        sum33 = 0.0E0_nag_wp

        Do i = 1, m
          dummy = 2.0E0_nag_wp*t(i,1)/(xc(2)*t(i,2)+xc(3)*t(i,3))**3
          sum22 = sum22 + fvec(i)*dummy*t(i,2)**2
          sum32 = sum32 + fvec(i)*dummy*t(i,2)*t(i,3)
          sum33 = sum33 + fvec(i)*dummy*t(i,3)**2
        End Do

        b(3) = sum22
        b(4) = 0.0E0_nag_wp
        b(5) = sum32
        b(6) = sum33

        Return

      End Subroutine lshes2
    End Module e04hyfe_mod
    Program e04hyfe
```

```
!     E04HYF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04hyf, nag_wp
      Use e04hyfe_mod, Only: lsfun2, lshes2, lw, m, n, nin, nout, nt, t, y
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: fsumsq
      Integer                          :: i, ifail
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: ruser(1), w(lw), x(n)
      Integer                          :: iuser(1)
!     .. Executable Statements ..
      Write (nout,*) 'E04HYF Example Program Results'

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

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

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

      x(1:nt) = (/0.5E0_nag_wp,1.0E0_nag_wp,1.5E0_nag_wp/)

      ifail = -1
      Call e04hyf(m,n,lsfun2,lshes2,x,fsumsq,w,lw,iuser,ruser,ifail)

      Select Case (ifail)
      Case (0,2:8,10:)
        Write (nout,*)
        Write (nout,99999) 'On exit, the sum of squares is', fsumsq
        Write (nout,99999) 'at the point', x(1:n)
      End Select

99999 Format (1X,A,3F12.4)
    End Program e04hyfe
```

10.2 Program Data

```
E04HYF 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

```
 E04HYF Example Program Results

 On exit, the sum of squares is      0.0082
 at the point      0.0824      1.1330      2.3437
```