

# NAG Fortran 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, USER,
1          IFAIL)
  INTEGER          M, N, LW, IUSER(*), IFAIL
  real           X(N), FSUMSQ, W(LW), USER(*)
  EXTERNAL        LSFUN2, LSHES2

```

### 3 Description

This routine is similar to the subroutine LSSDN2 in the National Physical Laboratory Algorithms Library. It is applicable to problems of the form:

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

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

The user 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 routines for consistency. Then, from a starting point supplied by the user, 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 Parameters

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:*  $1 \leq N \leq M$ .

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

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

Its specification is:

<pre style="margin: 0;">SUBROUTINE LSFUN2(M, N, XC, FVECC, FJACC, LJC, IUSER, USER)   INTEGER          M, N, LJC, IUSER(*)   <b>real</b>           XC(N), FVECC(M), FJACC(LJC,N), USER(*)</pre>
<p>Important: the dimension declaration for FJACC must contain the variable LJC, not an integer constant.</p>
<p>1: M – INTEGER <span style="float: right;"><i>Input</i></span></p>
<p>2: N – INTEGER <span style="float: right;"><i>Input</i></span></p> <p style="padding-left: 20px;"><i>On entry:</i> the numbers <math>m</math> and <math>n</math> of residuals and variables, respectively.</p>
<p>3: XC(N) – <b>real</b> array <span style="float: right;"><i>Input</i></span></p> <p style="padding-left: 20px;"><i>On entry:</i> the point <math>x</math> at which the values of the <math>f_i</math> and the <math>\frac{\partial f_i}{\partial x_j}</math> are required.</p>
<p>4: FVECC(M) – <b>real</b> array <span style="float: right;"><i>Output</i></span></p> <p style="padding-left: 20px;"><i>On exit:</i> FVECC(<math>i</math>) must be set to the value of <math>f_i</math> at the point <math>x</math>, for <math>i = 1, 2, \dots, m</math>.</p>
<p>5: FJACC(LJC,N) – <b>real</b> array <span style="float: right;"><i>Output</i></span></p> <p style="padding-left: 20px;"><i>On exit:</i> FJACC(<math>i, j</math>) must be set to the value of <math>\frac{\partial f_i}{\partial x_j}</math> at the point <math>x</math>, for <math>i = 1, 2, \dots, m</math>; <math>j = 1, 2, \dots, n</math>.</p>
<p>6: LJC – INTEGER <span style="float: right;"><i>Input</i></span></p> <p style="padding-left: 20px;"><i>On entry:</i> the first dimension of the array FJACC.</p>
<p>7: IUSER(*) – INTEGER array <span style="float: right;"><i>User Workspace</i></span></p>
<p>8: USER(*) – <b>real</b> array <span style="float: right;"><i>User Workspace</i></span></p> <p style="padding-left: 20px;">LSFUN2 is called from E04HYF with the parameters IUSER and USER as supplied to E04HYF. The user is free to use the arrays IUSER and USER to supply information to LSFUN2 as an alternative to using COMMON.</p>

LSFUN2 must be declared as EXTERNAL in the (sub)program from which E04HYF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

This routine must be supplied by the user 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).

Its specification is:

	SUBROUTINE LSHES2(M, N, FVECC, XC, B, LB, IUSER, USER)	
	INTEGER	M, N, LB, IUSER(*)
	<b>real</b>	FVECC(M), XC, B(LB), USER(*)
1:	M – INTEGER	<i>Input</i>
2:	N – INTEGER	<i>Input</i>
	<i>On entry:</i> the numbers $m$ and $n$ of residuals and variables, respectively.	
3:	FVECC(M) – <b>real</b> array	<i>Input</i>
	<i>On entry:</i> the value of the residual $f_i$ at the point in XC, for $i = 1, 2, \dots, m$ , so that the values of the $f_i$ can be used in the calculation of the elements of $B$ .	
4:	XC – <b>real</b>	<i>Input</i>
	<i>On entry:</i> the point $x$ at which the elements of $B$ are to be evaluated.	
5:	B(LB) – <b>real</b> array	<i>Input</i>
	<i>On exit:</i> $B$ 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 $\sum_{i=1}^m f_i \frac{\partial^2 f_i}{\partial x_j \partial x_k}$ evaluated at the point $x$ , for $j = 1, 2, \dots, n$ ; $k = 1, 2, \dots, j$ .	
6:	LB – INTEGER	<i>Input</i>
	<i>On entry:</i> the length of the array $B$ .	
7:	IUSER(*) – INTEGER array	<i>User Workspace</i>
8:	USER(*) – <b>real</b> array	<i>User Workspace</i>
	LSHES2 is called from E04HYF with the parameters IUSER and USER as supplied to E04HYF. The user is free to use the arrays IUSER and USER to supply information to LSHES2 as an alternative to using COMMON.	

LSHES2 must be declared as EXTERNAL in the (sub)program from which E04HYF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

- 5: X(N) – **real** array *Input/Output*  
*On entry:*  $X(j)$  must be set to a guess at the  $j$ th component of the position of the minimum, for  $j = 1, 2, \dots, n$ . The routine checks LSFUN2 and LSHES2 at the starting point and so is more likely to detect any error in the user's routines if the initial  $X(j)$  are non-zero and mutually distinct.  
*On exit:* the lowest point found during the calculations. Thus, if IFAIL = 0 on exit,  $X(j)$  is the  $j$ th component of the position of the minimum.
- 6: FSUMSQ – **real** *Output*  
*On exit:* the value of the sum of squares,  $F(x)$ , corresponding to the final point stored in X.
- 7: W(LW) – **real** 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:*

$$LW \geq 8 \times N + 2 \times N \times N + 2 \times M \times N + 3 \times M, \text{ if } N > 1,$$

$$LW \geq 11 + 5 \times M, \text{ if } N = 1.$$

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

**Note:** the dimension of the array IUSER must be at least 1.

IUSER is not used by E04HYF, but is passed directly to LSFUN2 and LSHES2 and may be used to pass information to those routines.

10: USER(\*) – *real* array *User Workspace*

**Note:** the dimension of the array USER must be at least 1.

USER is not used by E04HYF, but is passed directly to LSFUN2 and LSHES2 and may be used to pass information to those routines.

11: IFAIL – INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, –1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

*On exit:* IFAIL = 0 unless the routine detects an error (see Section 6).

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 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.**

## 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).

Errors or warnings detected by the routine:

IFAIL = 1

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

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.

IFAIL = 3

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

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  $x$  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 the user has made an error in forming the derivatives  $\frac{\partial f_i}{\partial x_j}$  in LSFUN2.

IFAIL = 10

It is very likely that the user has made an error in forming the quantities  $B_{jk}$  in LSHES2.

If the user is 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 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 the user 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.

## 9 Example

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.

<i>y</i>	<i>t</i> <sub>1</sub>	<i>t</i> <sub>2</sub>	<i>t</i> <sub>3</sub>
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.

## 9.1 Program Text

**Note:** the listing of the example program presented below uses *bold italicised* terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```

*      E04HYF Example Program Text.
*      Mark 19 Revised. NAG Copyright 1999.
*      .. Parameters ..
      INTEGER          M, N, NT, LW
      PARAMETER        (M=15,N=3,NT=3,LW=8*N+2*N*N+2*M*N+3*M)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*      .. Local Scalars ..
      real            FSUMSQ
      INTEGER          I, IFAIL, J, K
*      .. Local Arrays ..
      real            T(M,NT), USER(M+M*NT), W(LW), X(N), Y(M)
      INTEGER          IUSER(1)
*      .. External Subroutines ..
      EXTERNAL         E04HYF, LSFUN3, LSHES3
*      .. Executable Statements ..
      WRITE (NOUT,*) 'E04HYF Example Program Results'
*      Skip heading in data file
      READ (NIN,*)

*
*      Observations of TJ (J = 1, 2, 3) are held in T(I, J)
*      (I = 1, 2, . . . , 15)
*
      IUSER(1) = NT
      K = M
      DO 40 I = 1, M
         READ (NIN,*) Y(I), (T(I,J),J=1,NT)
         USER(I) = Y(I)
         DO 20 J = 1, NT
            USER(K+J) = T(I,J)
20      CONTINUE
         K = K + NT
40 CONTINUE
*
      X(1) = 0.5e0
      X(2) = 1.0e0
      X(3) = 1.5e0
*
      IFAIL = 1
*

```

```

CALL E04HYF(M,N,LSFUN3,LSHES3,X,FSUMSQ,W,LW,IUSER,USER,IFAIL)
*
IF (IFAIL.NE.0) THEN
  WRITE (NOUT,*)
  WRITE (NOUT,99999) 'Error exit type', IFAIL,
+   ' - see routine document'
END IF
IF (IFAIL.NE.1) THEN
  WRITE (NOUT,*)
  WRITE (NOUT,99998) 'On exit, the sum of squares is', FSUMSQ
  WRITE (NOUT,99998) 'at the point', (X(J),J=1,N)
END IF
STOP
*
99999 FORMAT (1X,A,I3,A)
99998 FORMAT (1X,A,3F12.4)
END

SUBROUTINE LSFUN3(M,N,XC,FVECC,FJACC,LJC,IUSER,USER)
*
Routine to evaluate the residuals and their 1st derivatives.
*
.. Scalar Arguments ..
INTEGER          LJC, M, N
*
.. Array Arguments ..
real            FJACC(LJC,N), FVECC(M), USER(*), XC(N)
INTEGER          IUSER(*)
*
.. Local Scalars ..
real            DENOM, DUMMY
INTEGER          I, K
*
.. Executable Statements ..
K = M
DO 20 I = 1, M
  DENOM = XC(2)*USER(K+2) + XC(3)*USER(K+3)
  FVECC(I) = XC(1) + USER(K+1)/DENOM - USER(I)
  FJACC(I,1) = 1.0e0
  DUMMY = -1.0e0/(DENOM*DENOM)
  FJACC(I,2) = USER(K+1)*USER(K+2)*DUMMY
  FJACC(I,3) = USER(K+1)*USER(K+3)*DUMMY
  K = K + IUSER(1)
20 CONTINUE
RETURN
END
*
SUBROUTINE LSHES3(M,N,FVECC,XC,B,LB,IUSER,USER)
*
Routine to compute the lower triangle of the matrix B
*
(stored by rows in the array B).
*
.. Scalar Arguments ..
INTEGER          LB, M, N
*
.. Array Arguments ..
real            B(LB), FVECC(M), USER(*), XC(N)
INTEGER          IUSER(*)
*
.. Local Scalars ..
real            DUMMY, SUM22, SUM32, SUM33
INTEGER          I, K
*
.. Executable Statements ..
B(1) = 0.0e0
B(2) = 0.0e0
SUM22 = 0.0e0
SUM32 = 0.0e0
SUM33 = 0.0e0
K = M
DO 20 I = 1, M
  DUMMY = 2.0e0*USER(K+1)/(XC(2)*USER(K+2)+XC(3)*USER(K+3))**3
  SUM22 = SUM22 + FVECC(I)*DUMMY*USER(K+2)**2
  SUM32 = SUM32 + FVECC(I)*DUMMY*USER(K+2)*USER(K+3)
  SUM33 = SUM33 + FVECC(I)*DUMMY*USER(K+3)**2
  K = K + IUSER(1)
20 CONTINUE
B(3) = SUM22
B(4) = 0.0e0
B(5) = SUM32
B(6) = SUM33

```

```
RETURN
END
```

## 9.2 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
```

## 9.3 Program Results

E04HYF Example Program Results

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

---