NAG Library Routine Document

E04HEF

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

E04HEF is a comprehensive 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.

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

2 Specification

```
SUBROUTINE E04HEF (M, N, LSQFUN, LSQHES, LSQMON, IPRINT, MAXCAL, ETA,      &
                   XTOL, STEPMX, X, FSUMSQ, FVEC, FJAC, LDFJAC, S, V,      &
                   LDV, NITER, NF, IW, LIW, W, LW, IFAIL)
INTEGER            M, N, IPRINT, MAXCAL, LDFJAC, LDV, NITER, NF,           &
                   IW(LIW), LIW, LW, IFAIL
REAL (KIND=nag_wp) ETA, XTOL, STEPMX, X(N), FSUMSQ, FVEC(M),               &
                   FJAC(LDFJAC,N), S(N), V(LDV,N), W(LW)
EXTERNAL           LSQFUN, LSQHES, LSQMON
```

3 Description

E04HEF is essentially identical to the subroutine LSQSDN 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 subroutines to calculate the values of the $f_i(x)$ and their first derivatives and second derivatives at any point x.

From a starting point $x^{(1)}$ supplied by you, the routine generates a sequence of points $x^{(2)}, x^{(3)}, \ldots$, which is intended to converge to a local minimum of $F(x)$. The sequence of points is given by

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$$

where the vector $p^{(k)}$ is a direction of search, and $\alpha^{(k)}$ is chosen such that $F\left(x^{(k)} + \alpha^{(k)} p^{(k)}\right)$ is approximately a minimum with respect to $\alpha^{(k)}$.

The vector $p^{(k)}$ used depends upon the reduction in the sum of squares obtained during the last iteration. If the sum of squares was sufficiently reduced, then $p^{(k)}$ is the Gauss–Newton direction; otherwise the second derivatives of the $f_i(x)$ are taken into account.

The method is designed to ensure that steady progress is made whatever the starting point, and to have the rapid ultimate convergence of Newton's method.

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

LSQFUN must calculate the vector of values $f_i(x)$ and Jacobian matrix of first derivatives $\frac{\partial f_i}{\partial x_j}$ at any point x. (However, if you do not wish to calculate the residuals or first derivatives at a particular x, there is the option of setting a parameter to cause E04HEF to terminate immediately.)

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)
```

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

1: IFLAG – INTEGER *Input/Output*

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

On exit: if it is not possible to evaluate the $f_i(x)$ or their first derivatives at the point given in XC (or if it wished to stop the calculations for any other reason), you should reset IFLAG to some negative number and return control to E04HEF. E04HEF will then terminate immediately, with IFAIL set to your setting of IFLAG.

2: M – INTEGER *Input*

On entry: m, the numbers of residuals.

3: N – INTEGER *Input*

On entry: n, the numbers of variables.

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

On entry: the point x at which the values of the f_i and the $\frac{\partial f_i}{\partial x_j}$ 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 $\frac{\partial f_i}{\partial x_j}$ 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 E04HEF 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*

LSQFUN is called with E04HEF's parameters IW, LIW, W, LW as these parameters. They are present so that, when other library routines require the solution of a minimization subproblem, constants needed for the evaluation of residuals can be passed through IW and W. Similarly, you could pass quantities of LSQFUN from the segment which calls E04HEF by using partitions of IW and W beyond those used as workspace by E04HEF. However, because of the danger of mistakes in partitioning, it is recommended that you should pass information to LSQFUN via COMMON global variables and **not use IW or W** at all. In any case you **must not change** the elements of IW and W used as workspace by E04HEF.

LSQFUN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04HEF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

Note: LSQFUN should be tested separately before being used in conjunction with E04HEF.

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

LSQHES must 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 $f_i(x)$. (As with LSQFUN, there is the option of causing E04HEF to terminate immediately.)

The specification of LSQHES is:

```
SUBROUTINE LSQHES (IFLAG, M, N, FVEC, XC, B, LB, IW, LIW, W, LW)
INTEGER             IFLAG, M, N, LB, IW(LIW), LIW, LW
REAL (KIND=nag_wp) FVEC(M), XC(N), B(LB), W(LW)
```

1: IFLAG – INTEGER *Input/Output*

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

On exit: if LSQHES resets IFLAG to some negative number, E04HEF will terminate immediately, with IFAIL set to your setting of IFLAG.

2: M – INTEGER *Input*

On entry: m, the numbers of residuals.

3: N – INTEGER *Input*

On entry: n, the numbers of variables.

4: 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.

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

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

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

On exit: unless IFLAG is reset to a negative number, B must contain the lower triangle of the matrix $\mathrm{B}(x)$, evaluated at the point x, stored by rows. (The upper triangle is not required because the matrix is symmetric.) More precisely, $\mathrm{B}(j(j-1)/2+k)$ 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 $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, j$.

7: LB – INTEGER *Input*

On entry: the length of the array B.

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

As in LSQFUN, these parameters correspond to the parameters IW, LIW, W, LW of E04HEF. LSQHES **must not change** the sections of IW and W required as workspace by E04HEF. Again, it is recommended that you should pass quantities to LSQHES via COMMON global variables and not use IW or W at all.

LSQHES must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04HEF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

Note: LSQHES should be tested separately before being used in conjunction with E04HEF.

5: LSQMON – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

If IPRINT ≥ 0, you must supply LSQMON which is suitable for monitoring the minimization process. LSQMON must not change the values of any of its parameters.

If IPRINT < 0, the dummy routine E04FDZ can be used as LSQMON.

The specification of LSQMON is:

```
SUBROUTINE LSQMON (M, N, XC, FVEC, FJAC, LDFJAC, S, IGRADE,        &
                   NITER, NF, IW, LIW, W, LW)
INTEGER            M, N, LDFJAC, IGRADE, NITER, NF, IW(LIW),       &
                   LIW, LW
REAL (KIND=nag_wp) XC(N), FVEC(M), FJAC(LDFJAC,N), S(N), W(LW)
```

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 coordinates of the current point x.

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

On entry: the values of the residuals f_i at the current point x.

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

On entry: FJAC(i,j) contains the value of $\frac{\partial f_i}{\partial x_j}$ at the current 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 E04HEF is called.

7: S(N) – REAL (KIND=nag_wp) array *Input*

On entry: the singular values of the current Jacobian matrix. Thus S may be useful as information about the structure of your problem. (If IPRINT > 0, LSQMON is called at the initial point before the singular values have been calculated, so the elements of S are set to zero for the first call of LSQMON.)

8: IGRADE – INTEGER *Input*

On entry: E04HEF estimates the dimension of the subspace for which the Jacobian matrix can be used as a valid approximation to the curvature (see Gill and Murray (1978)). This estimate is called the grade of the Jacobian matrix, and IGRADE gives its current value.

9: NITER – INTEGER *Input*

On entry: the number of iterations which have been performed in E04HEF.

10: NF – INTEGER *Input*

On entry: the number of times that LSQFUN has been called so far. Thus NF gives the number of evaluations of the residuals and the Jacobian matrix.

11: IW(LIW) – INTEGER array *Workspace*
12: LIW – INTEGER *Input*
13: W(LW) – REAL (KIND=nag_wp) array *Workspace*
14: LW – INTEGER *Input*

As in LSQFUN and LSQHES, these parameters correspond to the parameters IW, LIW, W, LW of E04HEF. They are included in LSQMON's parameter list primarily for when E04HEF is called by other library routines.

LSQMON must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04HEF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

Note: you should normally print the sum of squares of residuals, so as to be able to examine the sequence of values of $F(x)$ mentioned in Section 7. It is usually helpful to also print XC, the gradient of the sum of squares, NITER and NF.

6: IPRINT – INTEGER *Input*

On entry: specifies the frequency with which LSQMON is to be called.

IPRINT > 0
: LSQMON is called once every IPRINT iterations and just before exit from E04HEF.

IPRINT $= 0$
: LSQMON is just called at the final point.

IPRINT < 0
: LSQMON is not called at all.

IPRINT should normally be set to a small positive number.

Suggested value: IPRINT $= 1$.

7: MAXCAL – INTEGER *Input*

On entry: this parameter is present so as to enable you to limit the number of times that LSQFUN is called by E04HEF. There will be an error exit (see Section 6) after MAXCAL calls of LSQFUN.

Suggested value: MAXCAL $= 50 \times n$.

Constraint: MAXCAL ≥ 1.

8: ETA – REAL (KIND=nag_wp) *Input*

On entry: every iteration of E04HEF involves a linear minimization (i.e., minimization of $F\left(x^{(k)} + \alpha^{(k)} p^{(k)}\right)$ with respect to $\alpha^{(k)}$). ETA must lie in the range $0.0 \leq$ ETA < 1.0, and specifies how accurately these linear minimizations are to be performed. The minimum with respect to $\alpha^{(k)}$ will be located more accurately for small values of ETA (say, 0.01) than for large values (say, 0.9).

Although accurate linear minimizations will generally reduce the number of iterations performed by E04HEF, they will increase the number of calls of LSQFUN made each iteration. On balance it is usually more efficient to perform a low accuracy minimization.

Suggested value: ETA $= 0.5$ (ETA $= 0.0$ if N $= 1$).

Constraint: $0.0 \leq$ ETA < 1.0.

9: XTOL – REAL (KIND=nag_wp) *Input*

On entry: the accuracy in x to which the solution is required.

If x_{true} is the true value of x at the minimum, then x_{sol}, the estimated position before a normal exit, is such that

$$\|x_{\text{sol}} - x_{\text{true}}\| < \text{XTOL} \times (1.0 + \|x_{\text{true}}\|),$$

where $\|y\| = \sqrt{\sum_{j=1}^{n} y_j^2}$. For example, if the elements of x_{sol} are not much larger than 1.0 in modulus and if XTOL $= 1.0\text{E}{-5}$, then x_{sol} is usually accurate to about five decimal places. (For further details see Section 7.)

If $F(x)$ and the variables are scaled roughly as described in Section 9 and ϵ is the ***machine precision***, then a setting of order XTOL $= \sqrt{\epsilon}$ will usually be appropriate. If XTOL is set to 0.0 or some positive value less than 10ϵ, E04HEF will use 10ϵ instead of XTOL, since 10ϵ is probably the smallest reasonable setting.

Constraint: XTOL ≥ 0.0.

10: STEPMX – REAL (KIND=nag_wp) *Input*

On entry: an estimate of the Euclidean distance between the solution and the starting point supplied by you. (For maximum efficiency, a slight overestimate is preferable.)

E04HEF will ensure that, for each iteration

$$\sum_{j=1}^{n} \left(x_j^{(k)} - x_j^{(k-1)}\right)^2 \leq (\text{STEPMX})^2,$$

where k is the iteration number. Thus, if the problem has more than one solution, E04HEF is most likely to find the one nearest to the starting point. On difficult problems, a realistic choice can prevent the sequence of $x^{(k)}$ entering a region where the problem is ill-behaved and can help avoid overflow in the evaluation of $F(x)$. However, an underestimate of STEPMX can lead to inefficiency.

Suggested value: STEPMX = 100000.0.

Constraint: STEPMX $\geq$ XTOL.

11: 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 $j = 1, 2, \ldots, n$.

On exit: the final point $x^{(k)}$. Thus, if IFAIL $= 0$ on exit, X(j) is the jth component of the estimated position of the minimum.

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

On exit: the value of $F(x)$, the sum of squares of the residuals $f_i(x)$, at the final point given in X.

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

On exit: the value of the residual $f_i(x)$ at the final point given in X, for $i = 1, 2, \ldots, m$.

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

On exit: the value of the first derivative $\frac{\partial f_i}{\partial x_j}$ evaluated at the final point given in X, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

15: LDFJAC – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which E04HEF is called.

Constraint: LDFJAC $\geq$ M.

16: S(N) – REAL (KIND=nag_wp) array *Output*

On exit: the singular values of the Jacobian matrix at the final point. Thus S may be useful as information about the structure of your problem.

17: V(LDV, N) – REAL (KIND=nag_wp) array *Output*

On exit: the matrix V associated with the singular value decomposition

$$J = USV^{\mathrm{T}}$$

of the Jacobian matrix at the final point, stored by columns. This matrix may be useful for statistical purposes, since it is the matrix of orthonormalized eigenvectors of $J^{\mathrm{T}}J$.

18: LDV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which E04HEF is called.

Constraint: LDV $\geq$ N.

19: NITER – INTEGER *Output*

On exit: the number of iterations which have been performed in E04HEF.

20: NF – INTEGER *Output*

On exit: the number of times that the residuals and Jacobian matrix have been evaluated (i.e., number of calls of LSQFUN).

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

22: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04HEF is called.

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

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

24: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04HEF is called.

Constraints:

if $\text{N} > 1$, $\text{LW} \geq 7 \times \text{N} + 2 \times \text{M} \times \text{N} + \text{M} + \text{N} \times \text{N}$;
if $\text{N} = 1$, $\text{LW} \geq 9 + 3 \times \text{M}$.

25: 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 $\text{IFAIL} \neq 0$ 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: $\text{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

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

Note: E04HEF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

$\text{IFAIL} < 0$

A negative value of IFAIL indicates an exit from E04HEF because you have set IFLAG negative in LSQFUN or LSQHES. The value of IFAIL will be the same as your setting of IFLAG.

$\text{IFAIL} = 1$

On entry, $\text{N} < 1$,
or $\text{M} < \text{N}$,
or $\text{MAXCAL} < 1$,
or $\text{ETA} < 0.0$,
or $\text{ETA} \geq 1.0$,
or $\text{XTOL} < 0.0$,
or $\text{STEPMX} < \text{XTOL}$,
or $\text{LDFJAC} < \text{M}$,
or $\text{LDV} < \text{N}$,
or $\text{LIW} < 1$,
or $\text{LW} < 7 \times \text{N} + \text{M} \times \text{N} + 2 \times \text{M} + \text{N} \times \text{N}$ when $\text{N} > 1$,
or $\text{LW} < 9 + 3 \times \text{M}$ when $\text{N} = 1$.

When this exit occurs, no values will have been assigned to FSUMSQ, or to the elements of FVEC, FJAC, S or V.

IFAIL $= 2$

There have been MAXCAL calls of LSQFUN. If steady reductions in the sum of squares, $F(x)$, were monitored up to the point where this exit occurred, then the exit probably occurred simply because MAXCAL was set too small, so the calculations should be restarted from the final point held in X. This exit may also indicate that $F(x)$ has no minimum.

IFAIL $= 3$

The conditions for a minimum have not all been satisfied, but a lower point could not be found. This could be because XTOL has been set so small that rounding errors in the evaluation of the residuals and derivatives make attainment of the convergence conditions impossible.

IFAIL $= 4$

The method for computing the singular value decomposition of the Jacobian matrix has failed to converge in a reasonable number of sub-iterations. It may be worth applying E04HEF again starting with an initial approximation which is not too close to the point at which the failure occurred.

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.

The values IFAIL $= 2$, 3 and 4 may also be caused by mistakes in LSQFUN or LSQHES, by the formulation of the problem or by an awkward function. If there are no such mistakes 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.

7 Accuracy

A successful exit (IFAIL $= 0$) is made from E04HEF when the matrix of second derivatives of $F(x)$ is positive definite, and when (B1, B2 and B3) or B4 or B5 hold, where

$$\begin{array}{lll} \mathrm{B1} & \equiv & \alpha^{(k)} \times \left\|p^{(k)}\right\| < (\mathrm{XTOL} + \epsilon) \times \left(1.0 + \left\|x^{(k)}\right\|\right) \\ \mathrm{B2} & \equiv & \left|F^{(k)} - F^{(k-1)}\right| < (\mathrm{XTOL} + \epsilon)^2 \times \left(1.0 + F^{(k)}\right) \\ \mathrm{B3} & \equiv & \left\|g^{(k)}\right\| < \epsilon^{1/3} \times \left(1.0 + F^{(k)}\right) \\ \mathrm{B4} & \equiv & F^{(k)} < \epsilon^2 \\ \mathrm{B5} & \equiv & \left\|g^{(k)}\right\| < \left(\epsilon \times \sqrt{F^{(k)}}\right)^{1/2} \end{array}$$

and where $\|.\|$ and ϵ are as defined in Section 5, and $F^{(k)}$ and $g^{(k)}$ are the values of $F(x)$ and its vector of first derivatives at $x^{(k)}$.

If IFAIL $= 0$, then the vector in X on exit, x_{sol}, is almost certainly an estimate of x_{true}, the position of the minimum to the accuracy specified by XTOL.

If IFAIL $= 3$, then x_{sol} may still be a good estimate of x_{true}, but to verify this you should make the following checks. If

(a) the sequence $\{F(x^{(k)})\}$ converges to $F(x_{\text{sol}})$ at a superlinear or a fast linear rate, and

(b) $g(x_{\text{sol}})^{\text{T}}g(x_{\text{sol}}) < 10\epsilon$, where T denotes transpose, then it is almost certain that x_{sol} is a close approximation to the minimum.

When (b) is true, then usually $F(x_{\text{sol}})$ is a close approximation to $F(x_{\text{true}})$. The values of $F(x^{(k)})$ can be calculated in LSQMON, and the vector $g(x_{\text{sol}})$ can be calculated from the contents of FVEC and FJAC on exit from E04HEF.

Further suggestions about confirmation of a computed solution are given in the E04 Chapter Introduction.

8 Parallelism and Performance

E04HEF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

E04HEF 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, the behaviour of $F(x)$, the accuracy demanded and the distance of the starting point from the solution. The number of multiplications performed per iteration of E04HEF 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 LSQFUN and some iterations may call LSQHES. So, unless the residuals and their derivatives can be evaluated very quickly, the run time will be dominated by the time spent in LSQFUN (and, to a lesser extent, in LSQHES).

Ideally, the problem should be scaled so that, at the solution, $F(x)$ and the corresponding values of the x_j are each in the range $(-1, +1)$, and so that at points one unit away from the solution, $F(x)$ differs from its value at the solution by approximately one unit. This will usually imply that the Hessian matrix of $F(x)$ at the solution is well-conditioned. 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 E04HEF 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 the arrays S and V. 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

Before calling E04HEF, the program calls E04YAF and E04YBF to check LSQFUN and LSQHES. It uses $(0.5, 1.0, 1.5)$ as the initial guess at the position of the minimum.

10.1 Program Text

```
!   E04HEF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.
    Module e04hefe_mod

!     E04HEF 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, lsqgrd, lsqhes, lsqmon
!     .. Parameters ..
      Integer, Parameter, Public          :: liw = 1, m = 15, n = 3, nin = 5, &
                                             nout = 6, nt = 3
      Integer, Parameter, Public          :: lb = n*(n+1)/2
      Integer, Parameter, Public          :: ldfjac = m
      Integer, Parameter, Public          :: ldv = n
      Integer, Parameter, Public          :: lw = 7*n + m*n + 2*m + n*n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Public, Save    :: t(m,nt), y(m)
    Contains
      Subroutine lsqgrd(m,n,fvec,fjac,ldfjac,g)
!       Routine to evaluate gradient of the sum of squares

!       .. Use Statements ..
        Use nag_library, Only: dgemv
!       .. Scalar Arguments ..
        Integer, Intent (In)                :: ldfjac, m, n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)     :: fjac(ldfjac,n), fvec(m)
        Real (Kind=nag_wp), Intent (Out)    :: g(n)
!       .. Executable Statements ..
!       The NAG name equivalent of dgemv is f06paf
        Call dgemv('T',m,n,1.0_nag_wp,fjac,ldfjac,fvec,1,0.0_nag_wp,g,1)

        g(1:n) = 2.0_nag_wp*g(1:n)

        Return

      End Subroutine lsqgrd
      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)
          fvec(i) = xc(1) + t(i,1)/denom - y(i)
          fjac(i,1) = 1.0_nag_wp
          dummy = -1.0_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 lsqfun
      Subroutine lsqhes(iflag,m,n,fvec,xc,b,lb,iw,liw,w,lw)

!       Routine to compute the lower triangle of the matrix B
!       (stored by rows in the array B)

!       .. Scalar Arguments ..
        Integer, Intent (Inout)             :: iflag
        Integer, Intent (In)                :: lb, liw, lw, 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)  :: w(lw)
        Integer, Intent (Inout)             :: iw(liw)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                  :: dummy, sum22, sum32, sum33
        Integer                             :: i
!       .. Executable Statements ..
        b(1) = 0.0_nag_wp
        b(2) = 0.0_nag_wp
        sum22 = 0.0_nag_wp
        sum32 = 0.0_nag_wp
        sum33 = 0.0_nag_wp

        Do i = 1, m
          dummy = 2.0_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) = 0.0_nag_wp
        b(5) = sum32
        b(6) = sum33

        Return

      End Subroutine lsqhes
      Subroutine lsqmon(m,n,xc,fvec,fjac,ldfjac,s,igrade,niter,nf,iw,liw,w,lw)
!       Monitoring routine

!       .. Use Statements ..
        Use nag_library, Only: f06eaf
!       .. Parameters ..
```

```
        Integer, Parameter                   :: ndec = 3
!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: igrade, ldfjac, liw, lw, m, n, &
                                                nf, niter
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: fjac(ldfjac,n), fvec(m), s(n), &
                                                xc(n)
        Real (Kind=nag_wp), Intent (Inout)   :: w(lw)
        Integer, Intent (Inout)              :: iw(liw)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                   :: fsumsq, gtg
        Integer                              :: j
!       .. Local Arrays ..
        Real (Kind=nag_wp)                   :: g(ndec)
!       .. Executable Statements ..
        fsumsq = f06eaf(m,fvec,1,fvec,1)

        Call lsqgrd(m,n,fvec,fjac,ldfjac,g)

        gtg = f06eaf(n,g,1,g,1)

        Write (nout,*)
        Write (nout,*) &
          ' Itns    F evals          SUMSQ             GTG        grade'
        Write (nout,99999) niter, nf, fsumsq, gtg, igrade
        Write (nout,*)
        Write (nout,*) &
          '       X                    G           Singular values'

        Do j = 1, n
          Write (nout,99998) xc(j), g(j), s(j)
        End Do

        Return

99999   Format (1X,I4,6X,I5,6X,1P,E13.5,6X,1P,E9.1,6X,I3)
99998   Format (1X,1P,E13.5,10X,1P,E9.1,10X,1P,E9.1)
      End Subroutine lsqmon
    End Module e04hefe_mod
    Program e04hefe

!     E04HEF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04hef, e04yaf, e04ybf, nag_wp, x02ajf
      Use e04hefe_mod, Only: lb, ldfjac, ldv, liw, lsqfun, lsqgrd, lsqhes,     &
                             lsqmon, lw, m, n, nin, nout, nt, t, y
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: eta, fsumsq, stepmx, xtol
      Integer                              :: i, ifail, iprint, maxcal, nf,    &
                                              niter
!     .. Local Arrays ..
      Real (Kind=nag_wp)                   :: b(lb), fjac(ldfjac,n), fvec(m),  &
                                              g(n), s(n), v(ldv,n), w(lw), x(n)
      Integer                              :: iw(liw)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: sqrt
!     .. Executable Statements ..
      Write (nout,*) 'E04HEF 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
```

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

      x(1:nt) = (/0.19_nag_wp,-1.34_nag_wp,0.88_nag_wp/)

!     Check the 1st derivatives

      ifail = 0
      Call e04yaf(m,n,lsqfun,x,fvec,fjac,ldfjac,iw,liw,w,lw,ifail)

!     Check the evaluation of B

      ifail = 0
      Call e04ybf(m,n,lsqfun,lsqhes,x,fvec,fjac,ldfjac,b,lb,iw,liw,w,lw,ifail)

!     Continue setting parameters for E04HEF

!     Set IPRINT to 1 to obtain output from LSQMON at each iteration

      iprint = -1

      maxcal = 50*n
      eta = 0.9_nag_wp
      xtol = 10.0_nag_wp*sqrt(x02ajf())

!     We estimate that the minimum will be within 10 units of the
!     starting point

      stepmx = 10.0_nag_wp

!     Set up the starting point

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

      ifail = -1
      Call e04hef(m,n,lsqfun,lsqhes,lsqmon,iprint,maxcal,eta,xtol,stepmx,x, &
        fsumsq,fvec,fjac,ldfjac,s,v,ldv,niter,nf,iw,liw,w,lw,ifail)

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

        Call lsqgrd(m,n,fvec,fjac,ldfjac,g)

        Write (nout,99998) 'The corresponding gradient is', g(1:n)
        Write (nout,*) '                               (machine dependent)'
        Write (nout,*) 'and the residuals are'
        Write (nout,99997) fvec(1:m)
      End Select

99999 Format (1X,A,3F12.4)
99998 Format (1X,A,1P,3E12.3)
99997 Format (1X,1P,E9.1)
    End Program e04hefe
```

10.2 Program Data

```
E04HEF 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

```
E04HEF Example Program Results

On exit, the sum of squares is      0.0082
at the point      0.0824      1.1330      2.3437
The corresponding gradient is  -6.060E-12   9.030E-11   9.385E-11
                           (machine dependent)
and the residuals are
 -5.9E-03
 -2.7E-04
  2.7E-04
  6.5E-03
 -8.2E-04
 -1.3E-03
 -4.5E-03
 -2.0E-02
  8.2E-02
 -1.8E-02
 -1.5E-02
 -1.5E-02
 -1.1E-02
 -4.2E-03
  6.8E-03
```