# E04FCF – NAG Fortran Library Routine Document

**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

E04FCF is a comprehensive algorithm for finding an unconstrained minimum of a sum of squares of $m$ nonlinear functions in $n$ variables ($m \geq n$). No 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 E04FCF(M, N, LSQFUN, LSQMON, IPRINT, MAXCAL, ETA, XTOL,
1                  STEPMX, X, FSUMSQ, FVEC, FJAC, LJ, S, V, LV,
2                  NITER, NF, IW, LIW, W, LW, IFAIL)
 INTEGER           M, N, IPRINT, MAXCAL, LJ, LV, NITER, NF,
1                  IW(LIW), LIW, LW, IFAIL
 real              ETA, XTOL, STEPMX, X(N), FSUMSQ, FVEC(M),
1                  FJAC(LJ,N), S(N), V(LV,N), W(LW)
 EXTERNAL          LSQFUN, LSQMON
```

## 3 Description

This routine is essentially identical to the subroutine LSQNDN 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 $x = (x_1, x_2, \ldots, x_n)^T$ and $m \geq n$. (The functions $f_i(x)$ are often referred to as 'residuals'.) The user must supply a subroutine LSQFUN to calculate the values of the $f_i(x)$ at any point $x$.

From a starting point $x^{(1)}$ supplied by the user, 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 $F(x^{(k)} + \alpha^{(k)} p^{(k)})$ 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 an approximation to the Gauss–Newton direction; otherwise additional function evaluations are made so as to enable $p^{(k)}$ to be a more accurate approximation to the Newton direction.

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

[1] 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:* $1 \leq \mathrm{N} \leq \mathrm{M}$.

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

LSQFUN must calculate the vector of values $f_i(x)$ at any point $x$. (However, if the user does not wish to calculate the residuals at a particular $x$, there is the option of setting a parameter to cause E04FCF to terminate immediately.)

Its specification is:

```
SUBROUTINE LSQFUN(IFLAG, M, N, XC, FVECC, IW, LIW, W, LW)
INTEGER          IFLAG, M, N, IW(LIW), LIW, LW
real             XC(N), FVECC(M), W(LW)
```

**1:** IFLAG — INTEGER *Input/Output*

*On entry:* IFLAG has a non-negative value.

*On exit:* if LSQFUN resets IFLAG to some negative number, E04FCF will terminate immediately, with IFAIL set to the user's setting of IFLAG.

**2:** M — INTEGER *Input*
**3:** N — INTEGER *Input*

*On entry:* the numbers $m$ and $n$ of residuals and variables, respectively.

**4:** XC(N) — ***real*** array *Input*

*On entry:* the point $x$ at which the values of the $f_i$ are required.

**5:** FVECC(M) — ***real*** array *Output*

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

**6:** IW(LIW) — INTEGER array *Workspace*
**7:** LIW — INTEGER *Input*
**8:** W(LW) — ***real*** array *Workspace*
**9:** LW — INTEGER *Input*

LSQFUN is called with these parameters as in the call to E04FCF, so users can pass quantities to LSQFUN from the (sub)program which calls E04FCF by using partitions of IW and W beyond those used as workspace by E04FCF. However, because of the danger of mistakes in partitioning, it is recommended that this facility be used very selectively, e.g., for stable applications packages which need to pass their own variable dimension workspace to LSQFUN. It is **recommended** that the normal method for passing information from the user's (sub)program to LSQFUN should be via COMMON. In any case, users **must not change** LIW, LW or the elements of IW and W used as workspace by E04FCF.

**Note.** LSQFUN should be tested separately before being used in conjunction with E04FCF.

LSQFUN must be declared as EXTERNAL in the (sub)program from which E04FCF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

**4:** LSQMON — SUBROUTINE, supplied by the user. *External Procedure*

If IPRINT $\geq 0$, the user must supply a subroutine 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. (In some implementations the name of this routine is FDZE04; refer to the Users' Note for your implementation.)

Its specification is:

```
      SUBROUTINE LSQMON(M, N, XC, FVECC, FJACC, LJC, S, IGRADE, NITER,
     1                  NF, IW, LIW, W, LW)
      INTEGER           M, N, LJC, IGRADE, NITER, NF, IW(LIW), LIW, LW
      real              XC(N), FVECC(M), FJACC(LJC,N), S(N), W(LW)
```

Important: the dimension declaration for FJACC must contain the variable LJC, not an integer constant.

**1:** M — INTEGER *Input*
**2:** N — INTEGER *Input*

*On entry:* the numbers $m$ and $n$ of residuals and variables, respectively.

**3:** XC(N) — ***real*** array *Input*

*On entry:* the co-ordinates of the current point $x$.

**4:** FVECC(M) — ***real*** array *Input*

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

**5:** FJACC(LJC,N) — ***real*** array *Input*

*On entry:* FJACC$(i,j)$ contains the value of $\frac{\partial f_i}{\partial x_j}$, at the current point $x$, for $i = 1,2,\ldots,m$; $j = 1,2,\ldots,n$.

**6:** LJC — INTEGER *Input*

*On entry:* the first dimension of the array FJACC.

**7:** S(N) — ***real*** array *Input*

*On entry:* the singular values of the current approximation to the Jacobian matrix. Thus S may be useful as information about the structure of the user's problem.

**8:** IGRADE — INTEGER *Input*

*On entry:* E04FCF 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 [1]). 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 E04FCF.

**10:** NF — INTEGER *Input*

*On entry:* the number of times that LSQFUN has been called so far. (However, for intermediate calls of LSQMON, NF is calculated on the assumption that the latest linear search has been successful. If this is not the case, then the $n$ evaluations allowed for approximating the Jacobian at the new point will not in fact have been made. NF will be accurate at the final call of LSQMON.)

**11:** IW(LIW) — INTEGER array *Workspace*
**12:** LIW — INTEGER *Input*
**13:** W(LW) — ***real*** array *Workspace*
**14:** LW — INTEGER *Input*

These parameters correspond to the parameters IW, LIW, W, LW of E04FCF. They are included in LSQMON's parameter list primarily for when E04FCF is called by other Library routines.

**Note.** The user 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 print XC, the estimated gradient of the sum of squares, NITER and NF.
LSQMON must be declared as EXTERNAL in the (sub)program from which E04FCF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

**5:** IPRINT — INTEGER *Input*

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

If IPRINT > 0, LSQMON is called once every IPRINT iterations and just before exit from E04FCF.

If IPRINT = 0, LSQMON is just called at the final point.

If IPRINT < 0, LSQMON is not called at all.

IPRINT should normally be set to a small positive number.

*Suggested value:* IPRINT = 1.

**6:** MAXCAL — INTEGER *Input*

*On entry:* the limit set by the user on the number of times that LSQFUN may be called by E04FCF. There will be an error exit (see Section 6) after MAXCAL calls of LSQFUN.

*Suggested value:* MAXCAL = $400 \times n$.

*Constraint:* MAXCAL $\geq 1$.

**7:** ETA — ***real*** *Input*

Every iteration of E04FCF involves a linear minimization i.e., minimization of $F(x^{(k)} + \alpha^{(k)} p^{(k)})$ with respect to $\alpha^{(k)}$.

*On entry:* specifies how accurately the 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 E04FCF, 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:* $0.0 \leq \text{ETA} < 1.0$.

**8:** XTOL — ***real*** *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 prior to a normal exit, is such that

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

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 = 1.0E−5, then $x_{sol}$ is usually accurate to about 5 decimal places. (For further details see Section 7.)

If $F(x)$ and the variables are scaled roughly as described in Section 8 and $\epsilon$ 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 $10\epsilon$, E04FCF will use $10\epsilon$ instead of XTOL, since $10\epsilon$ is probably the smallest reasonable setting.

*Constraint:* XTOL $\geq 0.0$.

**9:** STEPMX — ***real*** *Input*

*On entry:* an estimate of the Euclidean distance between the solution and the starting point supplied by the user. (For maximum efficiency, a slight overestimate is preferable.) E04FCF will ensure that, for each iteration,

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

where $k$ is the iteration number. Thus, if the problem has more than one solution, E04FCF is most likely to find the one nearest to the starting point. On difficult problems, a realistic choice can prevent the sequence $x^{(k)}$ 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.

**10:** 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 $j = 1, 2, \ldots, n$.

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

**11:** FSUMSQ — ***real*** *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.

**12:** FVEC(M) — ***real*** array *Output*

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

**13:** FJAC(LJ,N) — ***real*** array *Output*

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

**14:** LJ — INTEGER *Input*

*On entry:* the first dimension of the array FJAC as declared in the (sub)program from which E04FCF is called.

*Constraint:* LJ $\geq$ M.

**15:** S(N) — ***real*** array *Output*

*On exit:* the singular values of the estimated Jacobian matrix at the final point. Thus S may be useful as information about the structure of the user's problem.

**16:** V(LV,N) — ***real*** array *Output*

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

$$J = USV^T$$

of the estimated Jacobian matrix at the final point, stored by columns. This matrix may be useful for statistical purposes, since it is the matrix of orthonormalised eigenvectors of $J^TJ$.

**17:** LV — INTEGER *Input*

*On entry:* the first dimension of the array V as declared in the (sub)program from which E04FCF is called.

*Constraint:* LV $\geq$ N.

**18:** NITER — INTEGER *Output*

*On exit:* the number of iterations which have been performed in E04FCF.

**19:** NF — INTEGER *Output*

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

**20:** IW(LIW) — INTEGER array *Workspace*

**21:** LIW — INTEGER *Input*

*On entry:* the length of IW as declared in the (sub)program from which E04FCF is called.

*Constraint:* LIW $\geq 1$.

**22:** W(LW) — ***real*** array *Workspace*

**23:** LW — INTEGER *Input*

*On entry:* the length of W as declared in the (sub)program from which E04FCF is called.

*Constraints:*

$\text{LW} \geq 6 \times \text{N} + \text{M} \times \text{N} + 2 \times \text{M} + \text{N} \times (\text{N} - 1)/2$, if $\text{N} > 1$
$\text{LW} \geq 7 + 3 \times \text{M}$, if $\text{N} = 1$.

**24:** 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 or gives a warning (see Section 6).

**For this routine**, because the values of output parameters may be useful even if IFAIL $\neq 0$ on exit, users are recommended to set IFAIL to $-1$ before entry. **It is then 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 specified by the routine:

IFAIL < 0

A negative value of IFAIL indicates an exit from E04FCF because the user has set IFLAG negative in LSQFUN. The value of IFAIL will be the same as the user's setting of IFLAG.

IFAIL = 1

On entry, N < 1,
or M < N,
or MAXCAL < 1,
or ETA < 0.0,
or ETA $\geq$ 1.0,
or XTOL < 0.0,
or STEPMX < XTOL,
or LJ < M,
or LV < N,
or LIW < 1,
or $\text{LW} < 6 \times \text{N} + \text{M} \times \text{N} + 2 \times \text{M} + \text{N} \times (\text{N} - 1)/2$, when $\text{N} > 1$,
or $\text{LW} < 7 + 3 \times \text{M}$, 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 make attainment of the convergence conditions impossible.

IFAIL = 4

The method for computing the singular value decomposition of the estimated Jacobian matrix has failed to converge in a reasonable number of sub-iterations. It may be worth applying E04FCF again starting with an initial approximation which is not too close to the point at which the failure occurred.

The values IFAIL = 2, 3 and 4 may also be caused by mistakes in LSQFUN, 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)

# 7 Accuracy

A successful exit (IFAIL = 0) is made from E04FCF when (B1, B2 and B3) or B4 or B5 hold, where

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

and where $\|.\|$ and $\epsilon$ are as defined in Section 5, and $F^{(k)}$ and $g^{(k)}$ are the values of $F(x)$ and its vector of estimated 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 the user should make the following checks. If

(a) the sequence $\{F(x^{(k)})\}$ converges to $F(x_{sol})$ at a superlinear or a fast linear rate, and
(b) $g(x_{sol})^T g(x_{sol}) < 10\epsilon$,

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 $F(x_{sol})$ is a close approximation to $F(x_{true})$. The values of $F(x^{(k)})$ can be calculated in LSQMON, and the vector $g(x_{sol})$ can be calculated from the contents of FVEC and FJAC on exit from E04FCF.

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

# 8 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 E04FCF varies, but for $m \gg n$ is approximately $n \times m^2 + \mathrm{O}(n^3)$. In addition, each iteration makes at least $n+1$ calls of LSQFUN. So, unless the residuals can be evaluated very quickly, the run time will be dominated by the time spent in LSQFUN.

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 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 E04FCF 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.

# 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.

| $y$ | $t_1$ | $t_2$ | $t_3$ |
|---|---|---|---|
| 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.

## 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.

```
*     E04FCF Example Program Text.
*     Mark 15 Revised.  NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          N, M, NT, LIW, LV, LJ, LW
      PARAMETER        (N=3,M=15,NT=3,LIW=1,LV=N,LJ=M,
     +                 LW=6*N+M*N+2*M+N*(N-1)/2)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Arrays in Common ..
      real             T(M,NT), Y(M)
*     .. Local Scalars ..
      real             ETA, FSUMSQ, STEPMX, XTOL
      INTEGER          I, IFAIL, IPRINT, J, MAXCAL, NF, NITER
*     .. Local Arrays ..
      real             FJAC(M,N), FVEC(M), G(N), S(N), V(LV,N), W(LW),
     +                 X(N)
      INTEGER          IW(LIW)
```

```
*     .. External Functions ..
      real              X02AJF
      EXTERNAL          X02AJF
*     .. External Subroutines ..
      EXTERNAL          E04FCF, LSQFUN, LSQGRD, LSQMON
*     .. Intrinsic Functions ..
      INTRINSIC         SQRT
*     .. Common blocks ..
      COMMON            Y, T
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04FCF 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)
      DO 20 I = 1, M
         READ (NIN,*) Y(I), (T(I,J),J=1,NT)
   20 CONTINUE
*     * Set IPRINT to 1 to obtain output from LSQMON at each iteration *
      IPRINT = -1
      MAXCAL = 400*N
      ETA = 0.5e0
      XTOL = 10.0e0*SQRT(X02AJF())
*     We estimate that the minimum will be within 10 units of the
*     starting point
      STEPMX = 10.0e0
*     Set up the starting point
      X(1) = 0.5e0
      X(2) = 1.0e0
      X(3) = 1.5e0
      IFAIL = 1
*
      CALL E04FCF(M,N,LSQFUN,LSQMON,IPRINT,MAXCAL,ETA,XTOL,STEPMX,X,
     +            FSUMSQ,FVEC,FJAC,LJ,S,V,LV,NITER,NF,IW,LIW,W,LW,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)
         CALL LSQGRD(M,N,FVEC,FJAC,LJ,G)
         WRITE (NOUT,99997) 'The estimated gradient is', (G(J),J=1,N)
         WRITE (NOUT,*) '                             (machine dependent)'
         WRITE (NOUT,*) 'and the residuals are'
         WRITE (NOUT,99996) (FVEC(I),I=1,M)
      END IF
      STOP
*
99999 FORMAT (1X,A,I3,A)
99998 FORMAT (1X,A,3F12.4)
99997 FORMAT (1X,A,1P,3e12.3)
99996 FORMAT (1X,1P,e9.1)
      END
```

```
*
      SUBROUTINE LSQFUN(IFLAG,M,N,XC,FVECC,IW,LIW,W,LW)
*     Routine to evaluate the residuals
*     .. Parameters ..
      INTEGER          MDEC, NT
      PARAMETER        (MDEC=15,NT=3)
*     .. Scalar Arguments ..
      INTEGER          IFLAG, LIW, LW, M, N
*     .. Array Arguments ..
      real             FVECC(M), W(LW), XC(N)
      INTEGER          IW(LIW)
*     .. Arrays in Common ..
      real             T(MDEC,NT), Y(MDEC)
*     .. Local Scalars ..
      INTEGER          I
*     .. Common blocks ..
      COMMON           Y, T
*     .. Executable Statements ..
      DO 20 I = 1, M
         FVECC(I) = XC(1) + T(I,1)/(XC(2)*T(I,2)+XC(3)*T(I,3)) - Y(I)
   20 CONTINUE
      RETURN
      END
*
      SUBROUTINE LSQMON(M,N,XC,FVECC,FJACC,LJC,S,IGRADE,NITER,NF,IW,LIW,
     +                  W,LW)
*     Monitoring routine
*     .. Parameters ..
      INTEGER          NDEC
      PARAMETER        (NDEC=3)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalar Arguments ..
      INTEGER          IGRADE, LIW, LJC, LW, M, N, NF, NITER
*     .. Array Arguments ..
      real             FJACC(LJC,N), FVECC(M), S(N), W(LW), XC(N)
      INTEGER          IW(LIW)
*     .. Local Scalars ..
      real             FSUMSQ, GTG
      INTEGER          J
*     .. Local Arrays ..
      real             G(NDEC)
*     .. External Functions ..
      real             F06EAF
      EXTERNAL         F06EAF
*     .. External Subroutines ..
      EXTERNAL         LSQGRD
*     .. Executable Statements ..
      FSUMSQ = F06EAF(M,FVECC,1,FVECC,1)
      CALL LSQGRD(M,N,FVECC,FJACC,LJC,G)
      GTG = F06EAF(N,G,1,G,1)
      WRITE (NOUT,*)
      WRITE (NOUT,*)
     +  '  Itn      F evals        SUMSQ             GTG        Grade'
      WRITE (NOUT,99999) NITER, NF, FSUMSQ, GTG, IGRADE
      WRITE (NOUT,*)
      WRITE (NOUT,*)
     +  '       X                     G           Singular values'
```

```
      DO 20 J = 1, N
         WRITE (NOUT,99998) XC(J), G(J), S(J)
   20 CONTINUE
      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 LSQGRD(M,N,FVECC,FJACC,LJC,G)
*     Routine to evaluate gradient of the sum of squares
*     .. Scalar Arguments ..
      INTEGER           LJC, M, N
*     .. Array Arguments ..
      real              FJACC(LJC,N), FVECC(M), G(N)
*     .. Local Scalars ..
      real              SUM
      INTEGER           I, J
*     .. Executable Statements ..
      DO 40 J = 1, N
         SUM = 0.0e0
         DO 20 I = 1, M
            SUM = SUM + FJACC(I,J)*FVECC(I)
   20    CONTINUE
         G(J) = SUM + SUM
   40 CONTINUE
      RETURN
      END
```

## 9.2 Program Data

```
E04FCF 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

```
 E04FCF Example Program Results

 On exit, the sum of squares is      0.0082
 at the point      0.0824      1.1330      2.3437
 The estimated gradient is   1.225E-09  -2.209E-10   2.723E-10
                            (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
```