NAG Fortran Library Routine Document

E04GDF

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

E04GDF 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 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 E04GDF (M, N, LSQFUN, LSQMON, IPRINT, MAXCAL, ETA, XTOL,
1                   STEPMX, X, FSUMSQ, FVEC, FJAC, LJ, S, V, LV, NITER,
2                   NF, IW, LIW, W, LW, IFAIL)
 INTEGER            M, N, IPRINT, MAXCAL, LJ, LV, NITER, NF, IW(LIW),
1                   LIW, LW, IFAIL
 double precision   ETA, XTOL, STEPMX, X(N), FSUMSQ, FVEC(M), FJAC(LJ,N),
1                   S(N), V(LV,N), W(LW)
 EXTERNAL           LSQFUN, LSQMON
```

3 Description

E04GDF is essentially identical to the (sub)program LSQFDN 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 (sub)program to calculate the values of the $f_i(x)$ and their first derivatives $\frac{\partial f_i}{\partial x_j}$ 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 finite difference estimates of 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: $1 \le \mathrm{N} \le \mathrm{M}$.

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 E04GDF to terminate immediately.)

Its specification is:

```
      SUBROUTINE LSQFUN (IFLAG, M, N, XC, FVECC, FJACC, LJC, IW, LIW, W,
     1                   LW)
      INTEGER            IFLAG, M, N, LJC, IW(LIW), LIW, LW
      double precision   XC(N), FVECC(M), FJACC(LJC,N), W(LW)
```

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

1: IFLAG – INTEGER *Input/Output*

On entry: to LSQFUN, IFLAG will be set to 1 or 2. The value 1 indicates that only the Jacobian matrix needs to be evaluated, and the value 2 indicates that both the residuals and the Jacobian matrix must be calculated.

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 E04GDF. E04GDF will then terminate immediately, with IFAIL set to your 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) – ***double precision*** 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: FVECC(M) – ***double precision*** array *Output*

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

6: FJACC(LJC,N) – ***double precision*** array *Output*

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

7: LJC – INTEGER *Input*

On entry: the first dimension of the array FJACC.

8: IW(LIW) – INTEGER array *Workspace*
9: LIW – INTEGER *Input*
10: W(LW) – ***double precision*** array *Workspace*
11: LW – INTEGER *Input*

LSQFUN is called with E04GDF'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 to LSQFUN from the segment which calls E04GDF by using partitions of IW and W beyond those used as workspace by E04GDF. However, because of the danger of mistakes in partitioning, it is **recommended** that you should pass information to LSQFUN via COMMON 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 E04GDF.

LSQFUN must be declared as EXTERNAL in the (sub)program from which E04GDF 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 E04GDF.

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

If IPRINT ≥ 0, you must supply a (sub)program 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, please refer to 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
      double precision   XC(N), FVECC(M), FJACC(LJC,N), S(N), W(LW)
```

Note: 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) – ***double precision*** array *Input*

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

4: FVECC(M) – ***double precision*** array *Input*

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

5: FJACC(LJC,N) – ***double precision*** 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) – ***double precision*** 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: E04GDF 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 E04GDF.

10: NF – INTEGER *Input*

On entry: the number of times that user-supplied (sub)program LSQFUN has been called so far with IFLAG $= 2$. (In addition to these calls monitored by NF, LSQFUN is called not more than N times per iteration with IFLAG set to 1.)

11: IW(LIW) – INTEGER array *Workspace*
12: LIW – INTEGER *Input*
13: W(LW) – ***double precision*** array *Workspace*
14: LW – INTEGER *Input*

As in LSQFUN, these parameters correspond to the parameters IW, LIW, W, LW of E04GDF. They are included in LSQMON's parameter list primarily for when E04GDF is called by other library routines.

LSQMON must be declared as EXTERNAL in the (sub)program from which E04GDF 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 also helpful to print XC, the gradient of the sum of squares, NITER and NF.

5: IPRINT – INTEGER *Input*

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

IPRINT > 0

LSQMON is called once every IPRINT iterations and just before exit from E04GDF.

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 $= 1$.

6: MAXCAL – INTEGER *Input*

On entry: enables you to limit the number of times that user-supplied (sub)program LSQFUN is called by E04GDF. There will be an error exit (see Section 6) after MAXCAL evaluations of the residuals (i.e., calls of LSQFUN with IFLAG set to 2). It should be borne in mind that, in addition to the calls of LSQFUN which are limited directly by MAXCAL, there will be calls of LSQFUN (with IFLAG set to 1) to evaluate only first derivatives.

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

Constraint: MAXCAL ≥ 1.

7: ETA – ***double precision*** *Input*

On entry: every iteration of E04GDF involves a linear minimization, i.e., minimization of $F\left(x^{(k)} + \alpha^{(k)}p^{(k)}\right)$ with respect to $\alpha^{(k)}$. ETA 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, they will tend to increase the number of calls of the user-supplied (sub)program LSQFUN (with IFLAG set to 2) needed for each linear minimization. On balance it is usually efficient to perform a low accuracy linear minimization.

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

Constraint: $0.0 \leq$ ETA < 1.0.

8: XTOL – ***double precision*** *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

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

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 $= 1.0\mathrm{D} - 5$, 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 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, E04GDF will use 10ϵ instead of XTOL, since 10ϵ is probably the smallest reasonable setting.

Constraint: XTOL ≥ 0.0.

9: STEPMX – ***double precision*** *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.) E04GDF will ensure that, for each iteration,

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

where k is the iteration number. Thus, if the problem has more than one solution, E04GDF 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.

10: X(N) – ***double precision*** array *Input/Output*

On entry: X(j) must be set to a guess at the jth component of the position of the minimum ($j = 1, 2, \ldots, n$).

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.

11: FSUMSQ – ***double precision*** *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) – ***double precision*** 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) – ***double precision*** 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 $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$.

14: LJ – INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which E04GDF is called.

Constraint: LJ $\geq$ M.

15: S(N) – ***double precision*** 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.

16: V(LV,N) – ***double precision*** 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 $J^{\mathrm{T}}J$.

17: LV – INTEGER *Input*

On entry: the first dimension of the array V as declared in the (sub)program from which E04GDF is called.

Constraint: LV $\geq$ N.

18: NITER – INTEGER *Output*

On exit: the number of iterations which have been performed in E04GDF.

19: NF – INTEGER *Output*

On exit: the number of times that the residuals have been evaluated (i.e., number of calls of LSQFUN with IFLAG set to 2).

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

On entry: the dimension of the array IW as declared in the (sub)program from which E04GDF is called.

Constraint: LIW $\geq$ 1.

22: W(LW) – ***double precision*** array *Communication Array*
23: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04GDF is called.

Constraints:

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

24: IFAIL – INTEGER *Input/Output*

On initial entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Chapter P01 for details.

On final 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 < 0

A negative value of IFAIL indicates an exit from E04GDF because you have set IFLAG negative in the user-supplied (sub)program LSQFUN. The value of IFAIL will be the same as your setting of IFLAG.

IFAIL $= 1$

On entry, $N < 1$,
or $M < N$,
or MAXCAL < 1,
or ETA < 0.0,
or ETA ≥ 1.0,
or XTOL < 0.0,
or STEPMX $<$ XTOL,
or $LJ < M$,
or $LV < N$,
or LIW < 1,
or $LW < 7 \times N + M \times N + 2 \times M + N \times N$ when $N > 1$,
or $LW < 9 + 3 \times M$ when $N = 1$.

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 evaluations of the residuals. 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 E04GDF again starting with an initial approximation which is not too close to the point at which the failure occurred.

The values IFAIL = 2, 3 or 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) in order to avoid the region which caused the failure.

7 Accuracy

A successful exit (IFAIL = 0) is made from E04GDF when the matrix of approximate second derivatives of $F(x)$ is positive-definite, and when (B1, B2 and B3) or B4 or B5 hold, where

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

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 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 you should make the following checks. If

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

(b) $g(x_{\mathrm{sol}})^{\mathrm{T}} g(x_{\mathrm{sol}}) < 10\epsilon$, 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 $F(x_{\mathrm{sol}})$ is a close approximation to $F(x_{\mathrm{true}})$. The values of $F\left(x^{(k)}\right)$ can be calculated in LSQMON, and the vector $g(x_{\mathrm{sol}})$ can be calculated from the contents of FVEC and FJAC on exit from E04GDF.

Further suggestions about confirmation of a computed solution are given in the E04 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 E04GDF 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. So, unless the residuals and their derivatives 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 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 E04GDF 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

Before calling E04GDF, the program calls E04YAF to check LSQFUN. It uses $(0.5, 1.0, 1.5)$ as the initial guess at the position of the minimum.

9.1 Program Text

```
*     E04GDF Example Program Text
*     Mark 15 Revised. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          N, M, NT, LV, LJ, LIW, LW
      PARAMETER        (N=3,M=15,NT=3,LV=N,LJ=M,LIW=1,
     +                 LW=7*N+M*N+2*M+N*N)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Arrays in Common ..
      DOUBLE PRECISION T(M,NT), Y(M)
*     .. Local Scalars ..
      DOUBLE PRECISION ETA, FSUMSQ, STEPMX, XTOL
      INTEGER          I, IFAIL, IPRINT, J, MAXCAL, NF, NITER
*     .. Local Arrays ..
      DOUBLE PRECISION FJAC(LJ,N), FVEC(M), G(N), S(N), V(LV,N), W(LW),
     +                 X(N)
      INTEGER          IW(LIW)
*     .. External Functions ..
      DOUBLE PRECISION X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         E04GDF, E04YAF, LSQFUN, LSQGRD, LSQMON
*     .. Intrinsic Functions ..
      INTRINSIC        SQRT
*     .. Common blocks ..
      COMMON           Y, T
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04GDF 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
*     Check LSQFUN by calling E04YAF at an arbitrary point. Since
*     E04YAF only checks the derivatives calculated when IFLAG = 2,
*     a separate program should be run before using E04YAF or
*     E04GDF to check that LSQFUN gives the same values for the
*     elements of FJACC when IFLAG is set to 1 as when IFLAG is
*     set to 2.
      X(1) = 0.19D0
      X(2) = -1.34D0
      X(3) = 0.88D0
      IFAIL = 0
*
      CALL E04YAF(M,N,LSQFUN,X,FVEC,FJAC,LJ,IW,LIW,W,LW,IFAIL)
*
*     Continue setting parameters for E04GDF
*     * Set IPRINT to 1 to obtain output from LSQMON at each iteration *
      IPRINT = -1
      MAXCAL = 50*N
      ETA = 0.9D0
      XTOL = 10.0D0*SQRT(X02AJF())
*     We estimate that the minimum will be within 10 units of the
*     starting point
      STEPMX = 10.0D0
*     Set up the starting point
      X(1) = 0.5D0
      X(2) = 1.0D0
      X(3) = 1.5D0
      IFAIL = 1
*
      CALL E04GDF(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 corresponding gradient is',
     +     (G(J),J=1,N)
         WRITE (NOUT,*) '                           (machine dependent)'
         WRITE (NOUT,*) 'and the residuals are'
         DO 40 I = 1, M
            WRITE (NOUT,99996) FVEC(I)
   40    CONTINUE
      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,FJACC,LJC,IW,LIW,W,LW)
*     Routine to evaluate the residuals and their 1st derivatives.
*     A COMMON variable could be updated here to count the
*     number of calls of LSQFUN with IFLAG set to 1 (since NF
*     in LSQMON only counts calls with IFLAG set to 2)
*     .. Parameters ..
      INTEGER          MDEC, NT
      PARAMETER        (MDEC=15,NT=3)
*     .. Scalar Arguments ..
```

```
      INTEGER           IFLAG, LIW, LJC, LW, M, N
*     .. Array Arguments ..
      DOUBLE PRECISION  FJACC(LJC,N), FVECC(M), W(LW), XC(N)
      INTEGER           IW(LIW)
*     .. Arrays in Common ..
      DOUBLE PRECISION  T(MDEC,NT), Y(MDEC)
*     .. Local Scalars ..
      DOUBLE PRECISION  DENOM, DUMMY
      INTEGER           I
*     .. Common blocks ..
      COMMON            Y, T
*     .. Executable Statements ..
      DO 20 I = 1, M
         DENOM = XC(2)*T(I,2) + XC(3)*T(I,3)
         IF (IFLAG.EQ.2) FVECC(I) = XC(1) + T(I,1)/DENOM - Y(I)
         FJACC(I,1) = 1.0D0
         DUMMY = -1.0D0/(DENOM*DENOM)
         FJACC(I,2) = T(I,1)*T(I,2)*DUMMY
         FJACC(I,3) = T(I,1)*T(I,3)*DUMMY
   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 ..
      DOUBLE PRECISION  FJACC(LJC,N), FVECC(M), S(N), W(LW), XC(N)
      INTEGER           IW(LIW)
*     .. Local Scalars ..
      DOUBLE PRECISION  FSUMSQ, GTG
      INTEGER           J
*     .. Local Arrays ..
      DOUBLE PRECISION  G(NDEC)
*     .. External Functions ..
      DOUBLE PRECISION  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)
*     A COMMON variable giving the number of calls of
*     LSQFUN with IFLAG set to 1 could be printed here
      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 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 ..
      DOUBLE PRECISION  FJACC(LJC,N), FVECC(M), G(N)
*     .. Local Scalars ..
      DOUBLE PRECISION  SUM
      INTEGER           I, J
*     .. Executable Statements ..
      DO 40 J = 1, N
         SUM = 0.0D0
         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

```
E04GDF 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

```
 E04GDF 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  -6.057E-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
```