

# NAG Library Routine Document

## E04GBF

**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

E04GBF is a comprehensive quasi-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 E04GBF (M, N, LSQLIN, LSQFUN, 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         LSQLIN, LSQFUN, LSQMON

```

### 3 Description

E04GBF is essentially identical to the subroutine LSQFDQ 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  $x = (x_1, x_2, \dots, x_n)^T$  and  $m \geq n$ . (The functions  $f_i(x)$  are often referred to as ‘residuals’.)

You must supply a subroutine 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  $x^{(2)}, x^{(3)}, \dots$ , 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 the Gauss–Newton direction; otherwise the second derivatives of the  $f_i(x)$  are taken into account using a quasi-Newton updating scheme.

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 Arguments

- 1: M – INTEGER *Input*  
 2: N – INTEGER *Input*

*On entry:* the number  $m$  of residuals,  $f_i(x)$ , and the number  $n$  of variables,  $x_j$ .

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

- 3: LSQLIN – SUBROUTINE, supplied by the NAG Library. *External Procedure*

LSQLIN enables you to specify whether the linear minimizations (i.e., minimizations of  $F(x^{(k)} + \alpha^{(k)}p^{(k)})$  with respect to  $\alpha^{(k)}$ ) are to be performed by a routine which just requires the evaluation of the  $f_i(x)$  (E04FCV), or by a routine which also requires the first derivatives of the  $f_i(x)$  (E04HEV).

It will often be possible to evaluate the first derivatives of the residuals in about the same amount of computer time that is required for the evaluation of the residuals themselves – if this is so then E04GBF should be called with routine E04HEV as the argument LSQLIN. However, if the evaluation of the derivatives takes more than about 4 times as long as the evaluation of the residuals, then E04FCV will usually be preferable. If in doubt, use E04HEV as it is slightly more robust.

Whichever subroutine is used, must be declared as EXTERNAL in the subroutine from which E04GBF is called.

- 4: 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 an argument to cause E04GBF 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 for FJAC must contain the variable LDFJAC, not an integer constant.

- 1: IFLAG – INTEGER *Input/Output*

*On entry:* will be set to 0, 1 or 2.

IFLAG = 0

Indicates that only the residuals need to be evaluated

IFLAG = 1

Indicates that only the Jacobian matrix needs to be evaluated

IFLAG = 2

Indicates that both the residuals and the Jacobian matrix must be calculated.

If E04HEV is used as E04GBF's LSQLIN, LSQFUN will always be called with IFLAG 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 is wished to stop the calculations for any other reason), you should reset IFLAG to some negative number and return control to E04GBF. E04GBF will then terminate immediately, with IFAIL set to your setting of IFLAG.

2:	M – INTEGER	<i>Input</i>
	<i>On entry:</i> $m$ , the number of residuals.	
3:	N – INTEGER	<i>Input</i>
	<i>On entry:</i> $n$ , the number of variables.	
4:	XC(N) – REAL (KIND=nag_wp) array	<i>Input</i>
	<i>On entry:</i> 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	<i>Output</i>
	<i>On exit:</i> unless IFLAG = 1 on entry, or IFLAG is reset to a negative number, then FVEC( $i$ ) must contain the value of $f_i$ at the point $x$ , for $i = 1, 2, \dots, m$ .	
6:	FJAC(LDFJAC, N) – REAL (KIND=nag_wp) array	<i>Output</i>
	<i>On exit:</i> unless IFLAG = 0 on entry, or IFLAG is reset to a negative number, then FJAC( $i, j$ ) must contain the value of $\frac{\partial f_i}{\partial x_j}$ at the point $x$ , for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$ .	
7:	LDFJAC – INTEGER	<i>Input</i>
	<i>On entry:</i> the first dimension of the array FJAC, set to $m$ by E04GBF.	
8:	IW(LIW) – INTEGER array	<i>Workspace</i>
9:	LIW – INTEGER	<i>Input</i>
10:	W(LW) – REAL (KIND=nag_wp) array	<i>Workspace</i>
11:	LW – INTEGER	<i>Input</i>
	LSQFUN is called with E04GBF's arguments IW, LIW, W, LW as these arguments. 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 E04GBF by using partitions of IW and W beyond those used as workspace by E04GBF. However, because of the danger of mistakes in partitioning, it is <b>recommended</b> that you should pass information to LSQFUN via COMMON global variables and <b>not use IW or W</b> at all. In any case you <b>must not change</b> the elements of IW and W used as workspace by E04GBF.	

LSQFUN must either be a module subprogram USED by, or declared as EXTERNAL in, the (sub)program from which E04GBF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

**Note:** LSQFUN should be tested separately before being used in conjunction with E04GBF.

- 5: LSQMON – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*
- If IPRINT  $\geq 0$ , you must supply LSQMON which is suitable for monitoring the minimization process. LSQMON must not change the values of any of its arguments.
- If IPRINT  $< 0$ , the NAG Library 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 <i>On entry:</i> $m$ , the numbers of residuals.	<i>Input</i>
2:	N – INTEGER <i>On entry:</i> $n$ , the numbers of variables.	<i>Input</i>
3:	XC(N) – REAL (KIND=nag_wp) array <i>On entry:</i> the coordinates of the current point $x$ .	<i>Input</i>
4:	FVEC(M) – REAL (KIND=nag_wp) array <i>On entry:</i> the values of the residuals $f_i$ at the current point $x$ .	<i>Input</i>
5:	FJAC(LDFJAC,N) – REAL (KIND=nag_wp) array <i>On entry:</i> FJAC( $i,j$ ) contains the value of $\frac{\partial f_i}{\partial x_j}$ at the current point $x$ , for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$ .	<i>Input</i>
6:	LDFJAC – INTEGER <i>On entry:</i> the first dimension of the array FJAC as declared in the (sub)program from which E04GBF is called.	<i>Input</i>
7:	S(N) – REAL (KIND=nag_wp) array <i>On entry:</i> the singular values of the current Jacobian matrix. Thus S may be useful as information about the structure of your problem.	<i>Input</i>
8:	IGRADE – INTEGER <i>On entry:</i> E04GBF 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.	<i>Input</i>
9:	NITER – INTEGER <i>On entry:</i> the number of iterations which have been performed in E04GBF.	<i>Input</i>
10:	NF – INTEGER <i>On entry:</i> the number of evaluations of the residuals. (If E04HEV is used as LSQLIN, NF is also the number of evaluations of the Jacobian matrix.)	<i>Input</i>
11:	IW(LIW) – INTEGER array	<i>Workspace</i>
12:	LIW – INTEGER	<i>Input</i>
13:	W(LW) – REAL (KIND=nag_wp) array	<i>Workspace</i>
14:	LW – INTEGER	<i>Input</i>
As in LSQFUN, these arguments correspond to the arguments IW, LIW, W, LW of E04GBF. They are included in LSQMON's argument list primarily for when E04GBF 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 E04GBF is called. Arguments 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:* the frequency with which LSQMON is to be called.

IPRINT > 0

LSQMON is called once every IPRINT iterations and just before exit from E04GBF.

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:* enables you to limit the number of times that LSQFUN is called by E04GBF. There will be an error exit (see Section 6) after MAXCAL calls of LSQFUN.

*Suggested value:*

MAXCAL =  $75 \times n$  if E04FCV is used as LSQLIN,

MAXCAL =  $50 \times n$  if E04HEV is used as LSQLIN.

*Constraint:* MAXCAL  $\geq 1$ .

8: ETA – REAL (KIND=nag\_wp) *Input*

*On entry:* every iteration of E04GBF involves a linear minimization (i.e., minimization of  $F(x^{(k)} + \alpha^{(k)}p^{(k)})$  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 performed by E04GBF, they will increase the number of calls of LSQFUN made every iteration. On balance it is usually more efficient to perform a low accuracy minimization.

*Suggested value:*

ETA = 0.9 if  $N > 1$  and E04HEV is used as LSQLIN,

ETA = 0.5 if  $N > 1$  and E04FCV is used as LSQLIN,

ETA = 0.0 if  $N = 1$ .

*Constraint:*  $0.0 \leq \text{ETA} < 1.0$ .

9: XTOL – REAL (KIND=nag\_wp) *Input*

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

If  $x_{\text{true}}$  is the true value of  $x$  at the minimum, then  $x_{\text{sol}}$ , 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  $x_{\text{sol}}$  are not much larger than 1.0 in modulus and if  $\text{XTOL} = 1.0\text{E}-5$ , then  $x_{\text{sol}}$  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  $\epsilon$  is the *machine precision*, then a setting of order  $\text{XTOL} = \sqrt{\epsilon}$  will usually be appropriate. If  $\text{XTOL}$  is set to 0.0 or some positive value less than  $10\epsilon$ , E04GBF will use  $10\epsilon$  instead of  $\text{XTOL}$ , since  $10\epsilon$  is probably the smallest reasonable setting.

*Constraint:*  $\text{XTOL} \geq 0.0$ .

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.)

E04GBF will ensure that, for each iteration,

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

where  $k$  is the iteration number. Thus, if the problem has more than one solution, E04GBF 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:*  $\text{STPEMX} = 100000.0$ .

*Constraint:*  $\text{STPEMX} \geq \text{XTOL}$ .

11: X(N) – REAL (KIND=nag\_wp) 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  $j = 1, 2, \dots, n$ .

*On exit:* the final point  $x^{(k)}$ . Thus, if  $\text{IFAIL} = 0$  on exit,  $X(j)$  is the  $j$ th 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  $i = 1, 2, \dots, m$ .

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  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, n$ .

15: LDFJAC – INTEGER *Input*

*On entry:* the first dimension of the array FJAC as declared in the (sub)program from which E04GBF is called.

*Constraint:*  $\text{LDFJAC} \geq M$ .

- 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
- $$J = USV^T$$
- 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  $J^T J$ .
- 18: LDV – INTEGER Input  
*On entry:* the first dimension of the array  $V$  as declared in the (sub)program from which E04GBF is called.  
*Constraint:*  $LDV \geq N$ .
- 19: NITER – INTEGER Output  
*On exit:* the number of iterations which have been performed in E04GBF.
- 20: NF – INTEGER Output  
*On exit:* the number of times that the residuals have been evaluated (i.e., the number of calls of LSQFUN). If E04HEV is used as LSQLIN, NF is also the number of times that the Jacobian matrix has been evaluated.
- 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 E04GBF is called.  
*Constraint:*  $LIW \geq 1$ .
- 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 E04GBF is called.  
*Constraints:*
- $$\begin{aligned} &\text{if } N > 1, LW \geq 7 \times N + M \times N + 2 \times M + N \times N; \\ &\text{if } N = 1, LW \geq 9 + 3 \times M. \end{aligned}$$
- 25: IFAIL – INTEGER Input/Output  
*On entry:* IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.  
 For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if  $IFAIL \neq 0$  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:*  $IFAIL = 0$  unless the routine detects an error or a warning has been flagged (see Section 6).

## 6 Error Indicators and Warnings

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

**Note:** E04GBF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

$IFAIL < 0$

A negative value of  $IFAIL$  indicates an exit from E04GBF because you have set  $IFLAG$  negative in  $LSQFUN$ . 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  $ETA \geq 1.0$ ,  
 or  $XTOL < 0.0$ ,  
 or  $STEPMX < XTOL$ ,  
 or  $LDFJAC < M$ ,  
 or  $LDV < N$ ,  
 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  $MAXCAL$  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. See Section 7 for further information.

$IFAIL = 4$

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 E04GBF 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 3.9 in How to Use the NAG Library and its Documentation for further information.

$IFAIL = -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.



IFAIL = -999

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

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 E04GBF when (B1, B2 and B3) or B4 or B5 hold, where

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

and where  $\|\cdot\|$  and  $\epsilon$  are as defined in XTOL, 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,  $x_{\text{sol}}$ , is almost certainly an estimate of  $x_{\text{true}}$ , the position of the minimum to the accuracy specified by XTOL.

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

- the sequence  $\{F(x^{(k)})\}$  converges to  $F(x_{\text{sol}})$  at a superlinear or a fast linear rate, and
- $g(x_{\text{sol}})^T g(x_{\text{sol}}) < 10\epsilon$  where T denotes transpose, then it is almost certain that  $x_{\text{sol}}$  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 E04GBF.

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

## 8 Parallelism and Performance

E04GBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

E04GBF 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 E04GBF 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 E04GBF 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 E04GBF, 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.

### 10.1 Program Text

```
! E04GBF Example Program Text
! Mark 26 Release. NAG Copyright 2016.
! Module e04gbfe_mod

! E04GBF 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, lsqmon
! .. Parameters ..
! Real (Kind=nag_wp), Parameter :: one = 1.0_nag_wp
! Real (Kind=nag_wp), Parameter :: two = 2.0_nag_wp
! Real (Kind=nag_wp), Parameter :: zero = 0.0_nag_wp
! Integer, Parameter :: incl = 1
```

```

Integer, Parameter, Public      :: liw = 1, m = 15, n = 3, nin = 5,      &
                                nout = 6, nt = 3
Integer, Parameter, Public      :: ldfjac = m
Integer, Parameter, Public      :: ldv = n
Integer, Parameter, Public      :: lw = 7*n + m*n + 2*m + n*n
Character (1), Parameter        :: trans = 'T'
! .. 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(trans,m,n,one,fjac,ldfjac,fvec,incl,zero,g,incl)

g(1:n) = two*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.
!   This routine is also suitable for use when E04FCV is used as
!   LSQLIN, since it can deal with IFLAG = 0 as well as IFLAG = 2.
!
!   .. 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(liw)
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)

    If (iflag/=0) Then
        fjac(i,1) = one
        dummy = -one/(denom*denom)
        fjac(i,2) = t(i,1)*t(i,2)*dummy
        fjac(i,3) = t(i,1)*t(i,3)*dummy
    End If

End Do

Return

End Subroutine lsqfun
Subroutine lsqmon(m,n,xc,fvec,fjac,ldfjac,s,igrade,niter,nf,iw,liw,w,lw)
!   Monitoring routine
!
!   .. Use Statements ..
Use nag_library, Only: ddot
!   .. 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 ..
!   The NAG name equivalent of ddot is f06eaf
fsumsq = ddot(m,fvec,incl,fvec,incl)

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

gtg = ddot(n,g,incl,g,incl)

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'      &

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

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 e04gbfe_mod
Program e04gbfe

!   E04GBF Example Main Program

!   .. Use Statements ..
Use nag_library, Only: e04gbf, e04hev, e04yaf, nag_wp, x02ajf
Use e04gbfe_mod, Only: ldfjac, ldv, liw, lsqfun, lsqgrd, 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)                :: 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,*) 'E04GBF 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

!   Check LSQFUN by calling E04YAF at an arbitrary point
x(1:nt) = (/0.19_nag_wp,-1.34_nag_wp,0.88_nag_wp/)

```

```

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

!   Continue setting parameters for E04GBF

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

    maxcal = 50*n

!   Since E04HEV is being used as LSQLIN, we set ETA to 0.9
    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 e04gbf(m,n,e04hev,lsqfun,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 e04gbfe

```

## 10.2 Program Data

```

E04GBF 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

E04GBF 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 1.199E-09 -1.865E-11 1.807E-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

---