E04DGF – 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.

Note. *This routine uses* optional parameters *to define choices in the problem specification and in the details of the algorithm. If you wish to use* default *settings for all of the optional parameters, you need only read Section 1 to Section 9 of this document. Refer to the additional Section 10 and Section 11 for a description of the algorithm and the specification of the optional parameters.*

Warning: the specification of the optional parameter **Maximum Step Length** changed at Mark 16.

1 Purpose

E04DGF minimizes an unconstrained nonlinear function of several variables using a pre-conditioned, limited memory quasi-Newton conjugate gradient method. First derivatives (or an 'acceptable' finite difference approximation to them) are required. It is intended for use on large scale problems.

2 Specification

```
 SUBROUTINE E04DGF(OBJFUN, ITER, OBJF, OBJGRD, X, IWORK, WORK,
1                  IUSER, USER, IFAIL)
 INTEGER           ITER, IWORK(N+1), IUSER(*), IFAIL
 real              OBJF, OBJGRD(N), X(N), WORK(13*N), USER(*)
 EXTERNAL          OBJFUN
```

3 Description

E04DGF is designed to solve unconstrained minimization problems of the form

$$\underset{x\in R^n}{\text{minimize}}\, F(x), \quad \text{subject to} \quad -\infty \leq x \leq \infty,$$

where x is an n element vector.

The user must supply an initial estimate of the solution.

For maximum reliability, it is preferable to provide all first partial derivatives. If all of the derivatives cannot be provided, users are recommended to obtain approximate values (using finite differences) by calling E04XAF from within OBJFUN. This is illustrated in Section 9 of the document for E04DJF.

The method used by E04DGF is described in Section 10.

4 References

[1] Gill P E and Murray W (1979) Conjugate-gradient methods for large-scale nonlinear optimization *Technical Report SOL 79–15* Department of Operations Research, Stanford University

[2] Gill P E, Murray W and Wright M H (1981) *Practical Optimization* Academic Press

5 Parameters

1: N — INTEGER *Input*

On entry: n, the number of variables.

Constraint: N > 0.

2: OBJFUN — SUBROUTINE, supplied by the user. *External Procedure*

On exit: the number of iterations performed.

Its specification is:

```
SUBROUTINE OBJFUN(MODE, N, X, OBJF, OBJGRD, NSTATE, IUSER, USER)
INTEGER          MODE, N, NSTATE, IUSER(*)
real             X(N), OBJF, OBJGRD(N), USER(*)
```

1: MODE — INTEGER *Input/Output*

On entry: MODE indicates which values must be assigned during each call of OBJFUN. Only the following values need be assigned:

if MODE = 0, OBJF;

if MODE = 2, OBJF and OBJGRD.

On exit: MODE may be set to a negative value if the user wishes to terminate the solution to the current problem, and in this case E04DGF will terminate with IFAIL set to MODE.

2: N — INTEGER *Input*

On entry: n, the number of variables.

3: X(N) — ***real*** array *Input*

On entry: x, the vector of variables at which the objective function and its gradient are to be evaluated.

4: OBJF — ***real*** *Output*

On exit: the value of the objective function at x.

5: OBJGRD(N) — ***real*** array *Output*

On exit: if MODE = 2, OBJGRD(i) must contain the value of $\frac{\partial F}{\partial x_i}$ evaluated at x, for $i = 1, 2, \ldots, n$.

6: NSTATE — INTEGER *Input*

On entry: NSTATE will be 1 on the first call of OBJFUN by E04DGF, and 0 for all subsequent calls. Thus, if the user wishes, NSTATE may be tested within OBJFUN in order to perform certain calculations once only. For example, the user may read data or initialise COMMON blocks when NSTATE = 1.

7: IUSER(∗) — INTEGER array *User Workspace*
8: USER(∗) — ***real*** array *User Workspace*

OBJFUN is called from E04DGF with the parameters IUSER and USER as supplied to E04DGF. The user is free to use arrays IUSER and USER to supply information to OBJFUN as an alternative to using COMMON.

OBJFUN must be declared as EXTERNAL in the (sub)program from which E04DGF is called, and should be tested separately before being used in conjunction with E04DGF. See also the optional parameter **Verify** in Section 11.2. Parameters denoted as *Input* must **not** be changed by this procedure.

3: ITER — INTEGER *Output*

On exit: the total number of iterations performed.

4: OBJF — ***real*** *Output*

On exit: the value of the objective function at the final iterate.

5: OBJGRD(N) — ***real*** array *Output*

On exit: the gradient of the objective function at the final iterate (or its finite difference approximation).

6: X(N) — ***real*** array *Input/Output*

On entry: an initial estimate of the solution.

On exit: the final estimate of the solution.

7: IWORK(N+1) — INTEGER array *Workspace*

8: WORK(13∗N) — ***real*** array *Workspace*

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

This array is not used by E04DGF, but is passed directly to routine OBJFUN and may be used to supply information to OBJFUN.

9: IUSER(∗) — INTEGER array *User Workspace*

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

This array is not used by E04DGF, but is passed directly to routine OBJFUN and may be used to supply information to OBJFUN.

10: USER(∗) — ***real*** array *User Workspace*

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

This array is not used by E04DGF, but is passed directly to routine OBJFUN and may be used to supply information to OBJFUN.

11: IFAIL — INTEGER *Input/Output*

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

On exit: IFAIL = 0 unless the routine detects an error 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**.

E04DGF returns with IFAIL = 0 if the following three conditions are satisfied:

(i) $F_{k-1} - F_k < \tau_F(1 + |F_k|)$
(ii) $\|x_{k-1} - x_k\| < \sqrt{\tau_F}(1 + \|x_k\|)$
(iii) $\|g_k\| \leq \sqrt[3]{\tau_F}(1 + |F_k|)$ or $\|g_k\| < \epsilon_A$

where τ_F is the value of the optional parameter **Optimality Tolerance** (default value = $\epsilon^{0.8}$; see Section 11.2) and ϵ_A is the absolute error associated with computing the objective function.

For a full discussion on termination criteria see Gill *et al.* [2] Chapter 8.

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 E04DGF because the user set MODE < 0 in routine OBJFUN. The value of IFAIL will be the same as the user's setting of MODE.

IFAIL = 1

Not used by this routine.

IFAIL = 2

Not used by this routine.

IFAIL = 3

The limiting number of iterations (as determined by the optional parameter **Iteration Limit** (default value = $\max(50, 5n)$; see Section 11.2) has been reached.

If the algorithm appears to be making satisfactory progress, then **Iteration Limit** may be too small. If so, increase its value and rerun E04DGF. If the algorithm seems to be making little or no progress, then the user should check for incorrect gradients as described below under IFAIL = 7.

IFAIL = 4

The computed upper bound on the step length taken during the linesearch was too small. A rerun with an increased value of the optional parameter **Maximum Step Length** (ρ say) may be successful unless $\rho \geq 10^{20}$ (the default value; see Section 11.2), in which case the current point cannot be improved upon.

IFAIL = 5

Not used by this routine.

IFAIL = 6

The conditions for an acceptable solution (see parameter IFAIL in Section 5) have not all been met, but a lower point could not be found.

If routine OBJFUN computes the objective function and its gradient correctly, then this may occur because an overly stringent accuracy has been requested, i.e., the value of the optional parameter **Optimality Tolerance** (default value = $\epsilon^{0.8}$; see Section 11.2) is too small or if $\alpha_k \simeq 0$. In this case the user should apply the three tests described above under IFAIL = 0 to determine whether or not the final solution is acceptable. For a discussion of attainable accuracy see Gill and Murray [2].

If many iterations have occurred in which essentially no progress has been made or E04DGF has failed to move from the initial point, routine OBJFUN may be incorrect. The user should refer to the comments below under IFAIL = 7 and check the gradients using the optional parameter **Verify** (default value = 0; see Section 11.2). Unfortunately, there may be small errors in the objective gradients that cannot be detected by the verification process. Finite-difference approximations to first derivatives are catastrophically affected by even small inaccuracies.

IFAIL = 7

The user-provided derivatives of the objective function appear to be incorrect.

Large errors were found in the derivatives of the objective function. This value of IFAIL will occur if the verification process indicated that at least one gradient element had no correct figures. The user should refer to the printed output to determine which elements are suspected to be in error.

As a first step, the user should check that the code for the objective values is correct – for example, by computing the function at a point where the correct value is known. However, care should be taken that the chosen point fully tests the evaluation of the function. It is remarkable how often the values $x = 0$ or $x = 1$ are used to test function evaluation procedures, and how often the special properties of these numbers make the test meaningless.

Special care should be used in this test if computation of the objective function involves subsidiary data communicated in COMMON storage. Although the first evaluation of the function may be correct, subsequent calculations may be in error because some of the subsidiary data has accidentally been overwritten.

Errors in programming the function may be quite subtle in that the function value is 'almost' correct. For example, the function may not be accurate to full precision because of the inaccurate calculation of a subsidiary quantity, or the limited accuracy of data upon which the function depends. A common error on machines where numerical calculations are usually performed in double precision is to include even one single precision constant in the calculation of the function; since some compilers do not convert such constants to double precision, half the correct figures may be lost by such a seemingly trivial error.

IFAIL = 8

The gradient ($g = \frac{\partial F}{\partial x}$) at the starting point x_0 is 'too small'. More precisely, the value of $g(x_0)^T g(x_0)$ is less than $\epsilon_m |F(x_0)|$, where ϵ_m is the ***machine precision***.

The problem should be rerun from a different starting point.

IFAIL = 9

An input parameter is invalid.

7 Accuracy

On successful exit (IFAIL = 0) the accuracy of the solution will be as defined by the optional parameter **Optimality Tolerance** (default value = $\epsilon^{0.8}$; see Section 11.2).

8 Further Comments

To evaluate an 'acceptable' set of finite difference intervals using E04XAF requires 2 function evaluations per variable for a well-scaled problem and up to 6 function evaluations per variable for a badly scaled problem.

8.1 Description of Printed Output

This section describes the (default) intermediate printout and final printout produced by E04DGF. The level of printed output can be controlled by the user (see the description of the optional parameter **Print Level** in Section 11.2). Note that the intermediate printout and final printout are produced only if Print Level ≥ 10 (the default).

The following line of summary output ($<$ 80 characters) is produced at every iteration. In all cases, the values of the quantities are those in effect *on completion* of the given iteration.

`Itn`	is the iteration count.
`Step`	is the step α_k taken along the computed search direction. On reasonably well-behaved problems, the unit step (i.e., $\alpha_k = 1$) will be taken as the solution is approached.
`Nfun`	is the cumulated number of evaluations of the objective function needed forthe linesearch. Evaluations needed for the verification of the gradients by finite differences are not included. `Nfun` is printed as a guide to the amount of work required for the linesearch. E04DGF will perform at most 11 function evaluations per iteration.
`Objective`	is the value of the objective function at x_k.
`Norm G`	is the Euclidean norm of the gradient of the objective function at x_k.
`Norm X`	is the Euclidean norm of x_k.
`Norm (X(k-1)-X(k))`	is the Euclidean norm of $x_{k-1} - x_k$.

The following describes the printout for each variable.

`Variable`	gives the name (`Varbl`) and index j, for $j = 1, 2, \ldots, n$ of the variable.
`Value`	is the value of the variable at the final iteration.
`Gradient Value`	is the value of the gradient of the objective function with respect to the jth variable at the final iteration.

Numerical values are output with a fixed number of digits; they are not guaranteed to be accurate to this precision.

9 Example

To find a minimum of the function

$$F = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1).$$

The initial point is

$$x_0 = (-1.0, 1.0)^T,$$

and $F(x_0) = 1.8394$ (to five figures).

The optimal solution is

$$x^* = (0.5, -1.0)^T,$$

and $F(x^*) = 0$.

The document for E04DJF includes an example program to solve the same problem using some of the optional parameters described in Section 11. The remainder of this document is intended for more advanced users. Section 10 contains a description of the algorithm which may be needed in order to understand Section 11. Section 11 describes the optional parameters which may be set by calls to E04DJF and/or E04DKF.

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.

```
*     E04DGF Example Program Text
*     Mark 16 Revised. NAG Copyright 1993.
*     .. Parameters ..
      INTEGER          NMAX
      PARAMETER        (NMAX=10)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             OBJF
      INTEGER          I, IFAIL, ITER, N
*     .. Local Arrays ..
      real             OBJGRD(NMAX), USER(1), WORK(13*NMAX), X(NMAX)
      INTEGER          IUSER(1), IWORK(NMAX+1)
*     .. External Subroutines ..
      EXTERNAL         E04DGF, OBJFUN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04DGF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.LE.NMAX) THEN
*
*        Read X from data file
*
         READ (NIN,*) (X(I),I=1,N)
*
*        Solve the problem
*
         IFAIL = -1
*
```

```
         CALL E04DGF(N,OBJFUN,ITER,OBJF,OBJGRD,X,IWORK,WORK,IUSER,USER,
     +               IFAIL)
*
      END IF
      STOP
      END
*
      SUBROUTINE OBJFUN(MODE,N,X,OBJF,OBJGRD,NSTATE,IUSER,USER)
*     Routine to evaluate F(x) and its 1st derivatives.
*     .. Parameters ..
      real              ONE, TWO, FOUR
      PARAMETER         (ONE=1.0e0,TWO=2.0e0,FOUR=4.0e0)
*     .. Scalar Arguments ..
      real              OBJF
      INTEGER           MODE, N, NSTATE
*     .. Array Arguments ..
      real              OBJGRD(N), USER(*), X(N)
      INTEGER           IUSER(*)
*     .. Local Scalars ..
      real              EXPX1, X1, X2
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Executable Statements ..
      X1 = X(1)
      X2 = X(2)
*
      EXPX1 = EXP(X1)
      OBJF = EXPX1*(FOUR*X1**2+TWO*X2**2+FOUR*X1*X2+TWO*X2+ONE)
*
      IF (MODE.EQ.2) THEN
         OBJGRD(1) = FOUR*EXPX1*(TWO*X1+X2) + OBJF
         OBJGRD(2) = TWO*EXPX1*(TWO*X2+TWO*X1+ONE)
      END IF
*
      RETURN
      END
```

9.2 Program Data

```
E04DGF Example Program Data
  2                         :Value of N
 -1.0  1.0                  :End of X
```

9.3 Program Results

```
 E04DGF Example Program Results

 *** E04DGF
 *** Start of NAG Library implementation details ***

 Implementation title: Generalised Base Version
            Precision: FORTRAN double precision
         Product Code: FLBAS19D
                 Mark: 19A

 *** End of NAG Library implementation details ***
```

```
Parameters
----------

Variables.............          2

Maximum step length....  1.00E+20      EPS (machine precision)  1.11E-16
Optimality tolerance...  3.26E-12      Linesearch tolerance...  9.00E-01

Est. opt. function val.      None      Function precision.....  4.37E-15
Verify level...........         0

Iteration limit........        50      Print level...........        10


Verification of the objective gradients.
----------------------------------------

The objective gradients seem to be ok.

Directional derivative of the objective   -1.47151776E-01
Difference approximation                  -1.47151796E-01


 Itn      Step  Nfun      Objective    Norm G    Norm X   Norm (X(k-1)-X(k))
   0               1   1.839397E+00   8.2E-01   1.4E+00
   1   3.7E-01     3   1.724275E+00   2.8E-01   1.3E+00         3.0E-01
   2   1.6E+01     8   6.083488E-02   9.2E-01   9.3E-01         2.2E+00
   3   1.6E-03    14   5.367978E-02   1.0E+00   9.6E-01         3.7E-02
   4   4.8E-01    16   1.783392E-04   5.8E-02   1.1E+00         1.6E-01
   5   1.0E+00    17   1.671122E-05   2.0E-02   1.1E+00         6.7E-03
   6   1.0E+00    18   1.101991E-07   1.7E-03   1.1E+00         2.4E-03
   7   1.0E+00    19   2.332133E-09   1.8E-04   1.1E+00         1.5E-04
   8   1.0E+00    20   9.130924E-11   3.3E-05   1.1E+00         3.0E-05
   9   1.0E+00    21   1.085455E-12   4.7E-06   1.1E+00         7.0E-06
  10   1.0E+00    22   5.308300E-14   1.2E-06   1.1E+00         6.4E-07

Exit from E04DGF after    10 iterations.


Variable          Value      Gradient value
Varbl    1     0.500000         9.1E-07
Varbl    2    -1.000000         8.3E-07

Exit E04DGF - Optimal solution found.

Final objective value =   0.5308300E-13
```

10 Algorithmic Details

This section contains a description of the method used by E04DGF.

E04DGF uses a pre-conditioned conjugate gradient method and is based upon algorithm PLMA as described in Gill and Murray [1] and Gill *et al.* [2] Section 4.8.3.

The algorithm proceeds as follows:

Let x_0 be a given starting point and let k denote the current iteration, starting with $k = 0$. The iteration requires g_k, the gradient vector evaluated at x_k, the kth estimate of the minimum. At each iteration a vector p_k (known as the direction of search) is computed and the new estimate x_{k+1} is given by $x_k + \alpha_k p_k$

where α_k (the step length) minimizes the function $F(x_k + \alpha_k p_k)$ with respect to the scalar α_k. A choice of initial step α_0 is taken as

$$\alpha_0 = \min\{1, 2 \times |F_k - F_{est}|/g_k^T g_k\}$$

where F_{est} is a user-supplied estimate of the function value at the solution. If F_{est} is not specified, the software always chooses the unit step length for α_0. Subsequent step length estimates are computed using cubic interpolation with safeguards.

A quasi-Newton method can be used to compute the search direction p_k by updating the inverse of the approximate Hessian (H_k) and computing

$$p_{k+1} = -H_{k+1} g_{k+1} \tag{1}$$

The updating formula for the approximate inverse is given by

$$H_{k+1} = H_k - \frac{1}{y_k^T s_k}\left(H_k y_k s_k^T + s_k y_k^T H_k\right) + \frac{1}{y_k^T s_k}\left(1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right) s_k s_k^T \tag{2}$$

where $y_k = g_{k-1} - g_k$ and $s_k = x_{k+1} - x_k = \alpha_k p_k$.

The method used by E04DGF to obtain the search direction is based upon computing p_{k+1} as $-H_{k+1} g_{k+1}$ where H_{k+1} is a matrix obtained by updating the identity matrix with a limited number of quasi-Newton corrections. The storage of an n by n matrix is avoided by storing only the vectors that define the rank two corrections – hence the term 'limited-memory' quasi-Newton method. The precise method depends upon the number of updating vectors stored. For example, the direction obtained with the 'one-step' limited memory update is given by (1) using (2) with H_k equal to the identity matrix, viz.

$$p_{k+1} = -g_{k+1} + \frac{1}{y_k^T s_k}\left(s_k^T g_{k+1} y_k + y_k^T g_{k+1} s_k\right) - \frac{s_k^T g_{k+1}}{y_k^T s_k}\left(1 + \frac{y_k^T y_k}{y_k^T s_k}\right) s_k.$$

E04DGF uses a two-step method described in detail in Gill and Murray [1] in which restarts and pre-conditioning are incorporated. Using a limited-memory quasi-Newton formula, such as the one above, guarantees p_{k+1} to be a descent direction if all the inner products $y_k^T s_k$ are positive for all vectors y_k and s_k used in the updating formula.

11 Optional Parameters

Several optional parameters in E04DGF define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of E04DGF these optional parameters have associated *default values* that are appropriate for most problems. Therefore, the user need only specify those optional parameters whose values are to be different from their default values. The remainder of this section can be skipped by users who wish to use the default values for *all* optional parameters. A complete list of optional parameters and their default values is given in Section 11.1.
Optional parameters may be specified by calling one, or both, of the routines E04DJF and E04DKF prior to a call to E04DGF. E04DJF reads options from an external options file, with `Begin` and `End` as the first and last lines respectively and each intermediate line defining a single optional parameter. For example,

```
Begin
  Print Level = 1
End
```

The call

```
CALL E04DJF (IOPTNS, INFORM)
```

can then be used to read the file on unit IOPTNS. INFORM will be zero on successful exit. E04DJF should be consulted for a full description of this method of supplying optional parameters.

E04DKF can be called to supply options directly, one call being necessary for each optional parameter. For example,

```
        CALL E04DKF ('Print Level = 1')
```

E04DKF should be consulted for a full description of this method of supplying optional parameters.

All optional parameters not specified by the user are set to their default values. Optional parameters specified by the user are unaltered by E04DGF (unless they define invalid values) and so remain in effect for subsequent calls unless altered by the user.

11.1 Optional Parameter Checklist and Default Values

For easy reference, the following list shows all the valid keywords and their default values. The symbol ϵ represents the ***machine precision*** (see X02AJF).

Optional Parameters	**Default Values**
Defaults	
Estimated optimal function value	
Function precision	$\epsilon^{0.9}$
Iteration limit	$\max(50, 5n)$
Linesearch tolerance	0.9
List/Nolist	**List**
Maximum step length	10^{20}
Optimality tolerance	$\epsilon^{0.8}$
Print level	10
Start objective check at variable	1
Stop objective check at variable	n
Verify level	0

11.2 Description of the Optional Parameters

The following list (in alphabetical order) gives the valid options. For each option, we give the keyword, any essential optional qualifiers, the default value, and the definition. The minimum abbreviation of each keyword is underlined. If no characters of an optional qualifier are underlined, the qualifier may be omitted. The letter a denotes a phrase (character string) that qualifies an option. The letters i and r denote INTEGER and ***real*** values required with certain options. The number ϵ is a generic notation for ***machine precision*** (see X02AJF), and ϵ_R denotes the relative precision of the objective function.

Defaults

This special keyword may be used to reset all optional parameters to their default values.

Estimated Optimal Function Value r

This value of r specifies the user-supplied guess of the optimum objective function value F_{est}. This value is used by E04DGF to calculate an initial step length α_0 (see Section 10). If the value of r is not specified by the user (the default), then this has the effect of setting α_0 to unity. It should be noted that for badly scaled functions a unit step along the steepest descent direction will often compute the objective function at very large values of x.

Function Precision r Default $= \epsilon^{0.9}$

The parameter defines ϵ_R, which is intended to be a measure of the accuracy with which the problem function $F(x)$ can be computed. If $r < \epsilon$ or $r \geq 1$, the default value is used.

The value of ϵ_R should reflect the relative precision of $1 + |F(x)|$; i.e., ϵ_R acts as a relative precision when $|F|$ is large, and as an absolute precision when $|F|$ is small. For example, if $F(x)$ is typically of order 1000 and the first six significant digits are known to be correct, an appropriate value for ϵ_R would be 10^{-6}. In contrast, if $F(x)$ is typically of order 10^{-4} and the first six significant digits are known to be correct, an appropriate value for ϵ_R would be 10^{-10}. The choice of ϵ_R can be quite complicated for badly scaled problems; see Chapter 8 of Gill and Murray [2], for a discussion of scaling techniques. The default value is appropriate for most simple functions that are computed with full accuracy. However when the accuracy of the computed function values is known to be significantly worse than full precision, the value of ϵ_R should be large enough so that E04DGF will not attempt to distinguish between function values that differ by less than the error inherent in the calculation.

Iteration Limit i Default $= \max(50, 5n)$
Iters
Itns

The value of i specifies the maximum number of iterations allowed before termination. If $i < 0$, the default value is used.

Problems whose Hessian matrices at the solution contain sets of clustered eigenvalues are likely to be minimized in significantly fewer than n iterations. Problems without this property may require anything between n and $5n$ iterations, with approximately $2n$ iterations being a common figure for moderately difficult problems.

Linesearch Tolerance r Default $= 0.9$

The value r $(0 \leq r < 1)$ controls the accuracy with which the step α taken during each iteration approximates a minimum of the function along the search direction (the smaller the value of r, the more accurate the linesearch). The default value $r = 0.9$ requests an inaccurate search, and is appropriate for most problems. A more accurate search may be appropriate when it is desirable to reduce the number of iterations – for example, if the objective function is cheap to evaluate. If $r < 0$ or $r \geq 1$, the default value is used.

List Default = List
Nolist

Normally each optional parameter specification is printed as it is supplied. **Nolist** may be used to suppress the printing and **List** may be used to restore printing.

Maximum Step Length r Default $= 10^{20}$

If $r > 0$, the maximum allowable step length for the linesearch is taken as $\min\left(\frac{1}{\text{X02AMF()}}, \frac{r}{\|p_k\|}\right)$. If $r \leq 0$, the default value is used.

Optimality Tolerance r Default $= \epsilon_R^{0.8}$

The parameter r $(\epsilon_R \leq r < 1)$ specifies the accuracy to which the user wishes the final iterate to approximate a solution of the problem. Broadly speaking, r indicates the number of correct figures desired in the objective function at the solution. For example, if r is 10^{-6} and E04DGF terminates successfully, the final value of F should have approximately six correct figures. E04DGF will terminate successfully if the iterative sequence of x-values is judged to have converged and the final point satisfies the termination criteria (as described under the parameter IFAIL in Section 5, where τ_F represents **Optimality Tolerance**). If $r < \epsilon_R$ or $r \geq 1$, the default value is used.

Print Level i Default $= 10$

The value i controls the amount of printout produced by E04DGF, as indicated below. A detailed description of the printout is given in Section 8.1 (summary output at each iteration and the final solution).

i	**Output**
0	No output.
1	The final solution only.
5	One line of summary output (< 80 characters; see Section 8.1) for each iteration (no printout of the final solution).
10	The final solution and one line of summary output for each iteration.

Start Objective Check at Variable i_1 Default $= 1$
Stop Objective Check at Variable i_2 Default $= n$

These keywords take effect only if **Verify Level** > 0 (see below). They may be used to control the verification of gradient elements computed by routine OBJFUN. For example, if the first 30 elements of the objective gradient appeared to be correct in an earlier run, so that only element 31 remains questionable, it is reasonable to specify **Start Objective Check at Variable** 31. If the first 30 variables appear linearly in the objective, so that the corresponding gradient elements are constant, the above choice would also be appropriate.

If $i_1 \leq 0$ or $i_1 > \max(1, \min(n, i_2))$, the default value is used. If $i_2 \leq 0$ or $i_2 > n$, the default value is used.

Verify Level	i	Default $= 0$
Verify	No	
Verify Level	-1	
Verify Level	0	
Verify		
Verify	Yes	
Verify Objective Gradients		
Verify Gradients		
Verify Level	1	

These keywords refer to finite-difference checks on the gradient elements computed by the user-provided routine OBJFUN. It is possible to specify **Verify Level** in several ways, as indicated above. For example, the objective gradient will be verified if **Verify**, **Verify Yes**, **Verify Gradients**, **Verify Objective Gradients** or **Verify Level** $= 1$ is specified. If $i = -1$, then no checking will be performed. If $i = 0$ or 1, then the objective gradient will be verified at the user-supplied initial estimate of the solution. If $i = 0$ only a 'cheap' test will be performed, requiring one call to OBJFUN. If $i = 1$, a more reliable (but more expensive) check will be made on individual gradient elements, within the ranges specified by the **Start** and **Stop** keywords as described above. A result of the form `OK` or `BAD`? is printed by E04DGF to indicate whether or not each element appears to be correct. If $i < -1$ or $i > 1$, the default value is used.