E04UNF – 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 11 and Section 12 for a description of the optional parameters and a description of the monitoring information produced by the routine.*

1 Purpose

E04UNF is designed to minimize an arbitrary smooth sum of squares function subject to constraints (which may include simple bounds on the variables, linear constraints and smooth nonlinear constraints) using a sequential quadratic programming (SQP) method. As many first derivatives as possible should be supplied by the user; any unspecified derivatives are approximated by finite differences. It is not intended for large sparse problems.

E04UNF may also be used for unconstrained, bound-constrained and linearly constrained optimization.

2 Specification

```
 SUBROUTINE E04UNF(M, N, NCLIN, NCNLN, LDA, LDCJ, LDFJ, LDR, A, BL,
1                  BU, Y, CONFUN, OBJFUN, ITER, ISTATE, C, CJAC, F,
2                  FJAC, CLAMDA, OBJF, R, X, IWORK, LIWORK, WORK,
3                  LWORK, IUSER, USER, IFAIL)
 INTEGER           M, N, NCLIN, NCNLN, LDA, LDCJ, LDFJ, LDR, ITER,
1                  ISTATE(N+NCLIN+NCNLN), IWORK(LIWORK), LIWORK,
2                  LWORK, IUSER(*), IFAIL
 real              A(LDA,*), BL(N+NCLIN+NCNLN), BU(N+NCLIN+NCNLN),
1                  Y(M), C(*), CJAC(LDCJ,*), F(M), FJAC(LDFJ,N),
2                  CLAMDA(N+NCLIN+NCNLN), OBJF, R(LDR,N), X(N),
3                  WORK(LWORK), USER(*)
 EXTERNAL          CONFUN, OBJFUN
```

3 Description

E04UNF is designed to solve the nonlinear least-squares programming problem – the minimization of a smooth nonlinear sum of squares function subject to a set of constraints on the variables. The problem is assumed to be stated in the following form:

$$\underset{x\in R^n}{\text{Minimize}}\ F(x) = \tfrac{1}{2}\sum_{i=1}^{m}(y_i - f_i(x))^2 \ \text{ subject to } \ l \leq \begin{Bmatrix} x \\ A_L x \\ c(x) \end{Bmatrix} \leq u, \tag{1}$$

where $F(x)$ (the *objective function)* is a nonlinear function which can be represented as the sum of squares of m subfunctions $(y_1 - f_1(x)), (y_2 - f_2(x)), \ldots, (y_m - f_m(x))$, the y_i are constant, A_L is an n_L by n constant matrix, and $c(x)$ is an n_N element vector of nonlinear constraint functions. (The matrix A_L and the vector $c(x)$ may be empty.) The objective function and the constraint functions are assumed to be smooth, i.e., at least twice-continuously differentiable. (The method of E04UNF will usually solve (1) if there are only isolated discontinuities away from the solution.)

Note that although the bounds on the variables could be included in the definition of the linear constraints, we prefer to distinguish between them for reasons of computational efficiency. For the same reason, the linear constraints should **not** be included in the definition of the nonlinear constraints. Upper and lower bounds are specified for all the variables and for all the constraints. An *equality* constraint can be specified by setting $l_i = u_i$. If certain bounds are not present, the associated elements of l or u can be set to special values that will be treated as $-\infty$ or $+\infty$. (See the description of the optional parameter **Infinite Bound Size** in Section 11.2.)

The user must supply an initial estimate of the solution to (1), together with subroutines that define $f(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T$, $c(x)$ and as many first partial derivatives as possible; unspecified derivatives are approximated by finite differences.

The subfunctions are defined by the array Y and subroutine OBJFUN, and the nonlinear constraints are defined by subroutine CONFUN. On every call, these subroutines must return appropriate values of $f(x)$ and $c(x)$. The user should also provide the available partial derivatives. Any unspecified derivatives are approximated by finite differences; see Section 11.2 for a discussion of the optional parameter **Derivative Level**. Just before either OBJFUN or CONFUN is called, each element of the current gradient array FJAC or CJAC is initialised to a special value. On exit, any element that retains the value is estimated by finite differences. Note that if there *are* any nonlinear constraints, then the *first* call to CONFUN will precede the *first* call to OBJFUN.

For maximum reliability, it is preferable for the user to provide all partial derivatives (see Chapter 8 of Gill *et al.* [1] for a detailed discussion). If all gradients cannot be provided, it is similarly advisable to provide as many as possible. While developing the subroutines OBJFUN and CONFUN, the optional parameter **Verify** (see Section 11.2) should be used to check the calculation of any known gradients.

4 References

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

[2] Hock W and Schittkowski K (1981) *Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems* **187** Springer-Verlag

5 Parameters

1: M — INTEGER *Input*

On entry: m, the number of subfunctions associated with $F(x)$.

Constraint: M > 0.

2: N — INTEGER *Input*

On entry: n, the number of variables.

Constraint: N > 0.

3: NCLIN — INTEGER *Input*

On entry: n_L, the number of general linear constraints.

Constraint: NCLIN ≥ 0.

4: NCNLN — INTEGER *Input*

On entry: n_N, the number of nonlinear constraints.

Constraint: NCNLN ≥ 0.

5: LDA — INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which E04UNF is called.

Constraint: LDA $\geq$ max(1,NCLIN).

6: LDCJ — INTEGER *Input*

On entry: the first dimension of the array CJAC as declared in the (sub)program from which E04UNF is called.

Constraint: LDCJ $\geq$ max(1,NCNLN).

7: LDFJ — INTEGER *Input*

On entry: the first dimension of the array FJAC as declared in the (sub)program from which E04UNF is called.

Constraint: LDFJ $\geq$ M.

8: LDR — INTEGER *Input*

On entry: the first dimension of the array R as declared in the (sub)program from which E04UNF is called.

Constraint: LDR $\geq$ N.

9: A(LDA,$*$) — ***real*** array *Input*

Note: the second dimension of the array A must be at least N when NCLIN > 0, and at least 1 otherwise.

On entry: the ith row of the array A must contain the ith row of the matrix A_L of general linear constraints in (1). That is, the ith row contains the coefficients of the ith general linear constraint, for $i = 1, 2, \ldots,$NCLIN.

If NCLIN $= 0$ then the array A is not referenced.

10: BL(N+NCLIN+NCNLN) — ***real*** array *Input*

11: BU(N+NCLIN+NCNLN) — ***real*** array *Input*

On entry: BL must contain the lower bounds and BU the upper bounds, for all the constraints in the following order. The first n elements of each array must contain the bounds on the variables, the next n_L elements the bounds for the general linear constraints (if any) and the next n_N elements the bounds for the general nonlinear constraints (if any). To specify a non-existent lower bound (i.e., $l_j = -\infty$), set BL(j) $\leq -$*bigbnd*, and to specify a non-existent upper bound (i.e., $u_j = +\infty$), set BU(j) $\geq$ *bigbnd*; the default value of *bigbnd* is 10^{20}, but this may be changed by the optional parameter **Infinite Bound Size** (see Section 11.2). To specify the jth constraint as an *equality*, set BL(j) = BU(j) = β, say, where $|\beta| <$ *bigbnd*.

Constraints:

BL(j) $\leq$ BU(j), for $j = 1, 2, \ldots,$N+NCLIN+NCNLN,
$|\beta| <$ *bigbnd* when BL(j) = BU(j) = β.

12: Y(M) — ***real*** array *Input*

On entry: the coefficients of the constant vector y of the objective function.

13: CONFUN — SUBROUTINE, supplied by the user. *External Procedure*

CONFUN must calculate the vector $c(x)$ of nonlinear constraint functions and (optionally) its Jacobian ($= \frac{\partial c}{\partial x}$) for a specified n element vector x. If there are no nonlinear constraints (i.e., NCNLN $= 0$), CONFUN will never be called by E04UNF and CONFUN may be the dummy routine E04UDM. (E04UDM is included in the NAG Fortran Library and so need not be supplied by the user. Its name may be implementation-dependent: see the Users' Note for your implementation for details.) If there are nonlinear constraints, the first call to CONFUN will occur before the first call to OBJFUN.

Its specification is:

```
      SUBROUTINE CONFUN(MODE, NCNLN, N, LDCJ, NEEDC, X, C, CJAC, NSTATE,
     1                  IUSER, USER)
      INTEGER           MODE, NCNLN, N, LDCJ, NEEDC(NCNLN), NSTATE,
     1                  IUSER(*)
      real              X(N), C(NCNLN), CJAC(LDCJ,N), USER(*)
```

1: MODE — INTEGER *Input/Output*

On entry: MODE indicates which values must be assigned during each call of CONFUN. Only the following values need be assigned, for each value of i such that NEEDC(i) > 0:

if MODE = 0, C(i);
if MODE = 1, all available elements in the ith row of CJAC;
if MODE = 2, C(i) and all available elements in the ith row of CJAC.

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 E04UNF will terminate with IFAIL set to MODE.

2: NCNLN — INTEGER *Input*

On entry: n_N, the number of nonlinear constraints.

3: N — INTEGER *Input*

On entry: n, the number of variables.

4: LDCJ — INTEGER *Input*

On entry: the first dimension of the array CJAC.

5: NEEDC(NCNLN) — INTEGER array *Input*

On entry: the indices of the elements of C and/or CJAC that must be evaluated by CONFUN. If NEEDC(i) > 0, then the ith element of C and/or the available elements of the ith row of CJAC (see parameter MODE above) must be evaluated at x.

6: X(N) — ***real*** array *Input*

On entry: x, the vector of variables at which the constraint functions and/or all available elements of the constraint Jacobian are to be evaluated.

7: C(NCNLN) — ***real*** array *Output*

On exit: if NEEDC(i) > 0 and MODE = 0 or 2, C(i) must contain the value of the ith constraint at x. The remaining elements of C, corresponding to the non-positive elements of NEEDC, are ignored.

8: CJAC(LDCJ,N) — ***real*** array *Output*

On exit: if NEEDC(i) > 0 and MODE = 1 or 2, the ith row of CJAC must contain the available elements of the vector ∇c_i given by

$$\nabla c_i = \left(\frac{\partial c_i}{\partial x_1}, \frac{\partial c_i}{\partial x_2}, \ldots, \frac{\partial c_i}{\partial x_n} \right)^T,$$

where $\frac{\partial c_i}{\partial x_j}$ is the partial derivative of the ith constraint with respect to the jth variable, evaluated at the point x. See also the parameter NSTATE below. The remaining rows of CJAC, corresponding to non-positive elements of NEEDC, are ignored.

If all elements of the constraint Jacobian are known (i.e., **Derivative Level** = 2 or 3 (default value = 3; see Section 11.2)), any constant elements may be assigned to CJAC one time only at the start of the optimization. An element of CJAC that is not subsequently assigned in CONFUN will retain its initial value throughout. Constant elements may be loaded into CJAC either before the call to E04UNF or during the first call to CONFUN (signalled by the value NSTATE = 1). The ability to preload constants is useful when many Jacobian elements are identically zero, in which case CJAC may be initialised to zero and non-zero elements may be reset by CONFUN.

Note that constant non-zero elements do affect the values of the constraints. Thus, if CJAC(i, j) is set to a constant value, it need not be reset in subsequent calls to CONFUN, but the value CJAC(i, j)$*$X(j) must nonetheless be added to C(i). For example, if CJAC(1,1) = 2 and CJAC(1,2) = -5, then the term 2$*$X(1) $-5*$ X(2) must be included in the definition of C(1).

It must be emphasized that, if **Derivative Level** = 0 or 1, unassigned elements of CJAC are not treated as constant; they are estimated by finite differences, at non-trivial expense. If the user does not supply a value for **Difference Interval** (the default; see Section 11.2), an interval for each element of x is computed automatically at the start of the optimization. The automatic procedure can usually identify constant elements of CJAC, which are then computed once only by finite differences.

9: NSTATE — INTEGER *Input*

On entry: if NSTATE = 1 then E04UNF is calling CONFUN for the first time. This parameter setting allows the user to save computation time if certain data must be read or calculated only once.

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

CONFUN is called from E04UNF with the parameters IUSER and USER as supplied to E04UNF. The user is free to use the arrays IUSER and USER to supply information to CONFUN as an alternative to using COMMON.

CONFUN must be declared as EXTERNAL in the (sub)program from which E04UNF is called, and should be tested separately before being used in conjunction with E04UNF. See also the optional parameter **Verify** in Section 11.2. Parameters denoted as *Input* must **not** be changed by this procedure.

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

OBJFUN must calculate the vector $f(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T$ and (optionally) its Jacobian $(= \frac{\partial f}{\partial x})$ for a specified n element vector x.

Its specification is:

```
SUBROUTINE OBJFUN(MODE, M, N, LDFJ, X, F, FJAC, NSTATE, IUSER, USER)
INTEGER           MODE, M, N, LDFJ, NSTATE, IUSER(*)
real              X(N), F(M), FJAC(LDFJ,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, F;
if MODE = 1, all available elements of FJAC;
if MODE = 2, F and all available elements of FJAC.

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 E04UNF will terminate with IFAIL set to MODE.

2: M — INTEGER *Input*

On entry: m, the number of subfunctions.

3: N — INTEGER *Input*

On entry: n, the number of variables.

4: LDFJ — INTEGER *Input*

On entry: the first dimension of the array FJAC.

5: X(N) — ***real*** array *Input*

On entry: x, the vector of variables at which $f(x)$ and/or all available elements of its Jacobian are to be evaluated.

6: F(M) — ***real*** array *Output*

On exit: if MODE = 0 or 2, F(i) must contain the value of f_i at x, for $i = 1, 2, \ldots, m$.

7: FJAC(LDFJ,N) — ***real*** array *Output*

On exit: if MODE = 1 or 2, the ith row of FJAC must contain the available elements of the vector ∇f_i given by

$$\nabla f_i = \left(\frac{\partial f_i}{\partial x_1}, \frac{\partial f_i}{\partial x_2}, \ldots, \frac{\partial f_i}{\partial x_n} \right)^T,$$

evaluated at the point x. See also the parameter NSTATE below.

8: NSTATE — INTEGER *Input*

On entry: if NSTATE = 1 then E04UNF is calling OBJFUN for the first time. This parameter setting allows the user to save computation time if certain data must be read or calculated only once.

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

OBJFUN is called from E04UNF with the parameters IUSER and USER as supplied to E04UNF. The user is free to use the 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 E04UNF is called, and should be tested separately before being used in conjunction with E04UNF. See also the optional parameter **Verify** in Section 11.2. Parameters denoted as *Input* must **not** be changed by this procedure.

15: ITER — INTEGER *Output*

On exit: the number of major iterations performed.

16: ISTATE(N+NCLIN+NCNLN) — INTEGER array *Input/Output*

On entry: ISTATE need not be set if the (default) **Cold Start** option is used.

If the **Warm Start** option has been chosen (see Section 11.2), the elements of ISTATE corresponding to the bounds and linear constraints define the initial working set for the procedure that finds a feasible point for the linear constraints and bounds. The active set at the conclusion of this procedure and the elements of ISTATE corresponding to nonlinear constraints then define the initial working set for the first QP subproblem. More precisely, the first n elements of ISTATE refer to the upper and lower bounds on the variables, the next n_L elements refer to the upper and lower bounds on $A_L x$, and the next n_N elements refer to the upper and lower bounds on $c(x)$. Possible values for ISTATE(j) are as follows:

ISTATE(j)	**Meaning**
0	The corresponding constraint is *not* in the initial QP working set.
1	This inequality constraint should be in the working set at its lower bound.
2	This inequality constraint should be in the working set at its upper bound.
3	This equality constraint should be in the initial working set. This value must not be specified unless BL(j) = BU(j).

The values -2, -1 and 4 are also acceptable but will be modified by the routine. If E04UNF has been called previously with the same values of N, NCLIN and NCNLN, ISTATE already contains satisfactory information. (See also the description of the optional parameter **Warm Start** in Section 11.2). The routine also adjusts (if necessary) the values supplied in X to be consistent with ISTATE.

Constraint: $-2 \leq$ ISTATE(j) ≤ 4, for $j = 1, 2, \ldots,$N+NCLIN+NCNLN.

On exit:

ISTATE(j)	**Meaning**
-2	This constraint violates its lower bound by more than the appropriate feasibility tolerance (see the optional parameters **Linear Feasibility Tolerance** and **Nonlinear Feasibility Tolerance** in Section 11.2). This value can occur only when no feasible point can be found for a QP subproblem.
-1	This constraint violates its upper bound by more than the appropriate feasibility tolerance (see the optional parameters **Linear Feasibility Tolerance** and **Nonlinear Feasibility Tolerance** in Section 11.2). This value can occur only when no feasible point can be found for a QP subproblem.
0	The constraint is satisfied to within the feasibility tolerance, but is not in the QP working set.
1	This inequality constraint is included in the QP working set at its lower bound.
2	This inequality constraint is included in the QP working set at its upper bound.
3	This constraint is included in the QP working set as an equality. This value of ISTATE can occur only when BL(j) = BU(j).

17: C(∗) — ***real*** array *Output*

Note: the dimension of the array C must be at least max(1,NCNLN).

On exit: if NCNLN > 0, C(i) contains the value of the ith nonlinear constraint function c_i at the final iterate, for $i = 1, 2, \ldots$,NCNLN.

If NCNLN $= 0$ then the array C is not referenced.

18: CJAC(LDCJ,∗) — ***real*** array *Input/Output*

Note: the second dimension of the array CJAC must be at least N when NCNLN > 0, and at least 1 otherwise.

On entry: in general, CJAC need not be initialised before the call to E04UNF. However, if **Derivative Level** $= 3$ (the default; see Section 11.2), the user may optionally set the constant elements of CJAC (see parameter NSTATE in the description of CONFUN). Such constant elements need not be re-assigned on subsequent calls to CONFUN.

On exit: if NCNLN > 0, CJAC contains the Jacobian matrix of the nonlinear constraint functions at the final iterate, i.e., CJAC(i, j) contains the partial derivative of the ith constraint function with respect to the jth variable, for $i = 1, 2, \ldots$,NCNLN; $j = 1, 2, \ldots$,N. (See the discussion of parameter CJAC under CONFUN.)

If NCNLN $= 0$ then the array CJAC is not referenced.

19: F(M) — ***real*** array *Output*

On exit: F(i) contains the value of the ith function f_i at the final iterate, for $i = 1, 2, \ldots$,M.

20: FJAC(LDFJ,N) — ***real*** array *Input/Output*

On entry: in general, FJAC need not be initialised before the call to E04UNF. However, if **Derivative Level** $= 3$ (the default; see Section 11.2), the user may optionally set the constant elements of FJAC (see parameter NSTATE in the description of OBJFUN). Such constant elements need not be re-assigned on subsequent calls to OBJFUN.

On exit: the Jacobian matrix of the functions $f_1, f_2, \ldots, f_m$ at the final iterate, i.e., FJAC(i, j) contains the partial derivative of the ith function with respect to the jth variable, for $i = 1, 2 \ldots$,M; $j = 1, 2, \ldots$,N. (See also the discussion of parameter FJAC under OBJFUN.)

21: CLAMDA(N+NCLIN+NCNLN) — ***real*** array *Input/Output*

On entry: CLAMDA need not be set if the (default) **Cold Start** option is used.

If the **Warm Start** option has been chosen (see Section 11.2), CLAMDA(j) must contain a multiplier estimate for each nonlinear constraint with a sign that matches the status of the constraint specified by the ISTATE array (as above), for j = N+NCLIN+1,N+NCLIN+2,. . .,N+NCLIN+NCNLN. The remaining elements need not be set. Note that if the jth constraint is defined as 'inactive' by the

initial value of the ISTATE array (i.e., ISTATE(j) = 0), CLAMDA(j) should be zero; if the jth constraint is an inequality active at its lower bound (i.e., ISTATE(j) = 1), CLAMDA(j) should be non-negative; if the jth constraint is an inequality active at its upper bound (i.e., ISTATE(j) = 2, CLAMDA(j) should be non-positive. If necessary, the routine will modify CLAMDA to match these rules.

On exit: the values of the QP multipliers from the last QP subproblem. CLAMDA(j) should be non-negative if ISTATE(j) =1 and non-positive if ISTATE(j) = 2.

22: OBJF — ***real*** *Output*

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

23: R(LDR,N) — ***real*** array *Input/Output*

On entry: R need not be initialised if the (default) **Cold Start** option is used.

If the **Warm Start** option has been chosen (see Section 11.2), R must contain the upper triangular Cholesky factor R of the initial approximation of the Hessian of the Lagrangian function, with the variables in the natural order. Elements not in the upper triangular part of R are assumed to be zero and need not be assigned.

On exit: if **Hessian** = **No** (the default; see Section 11.2), R contains the upper triangular Cholesky factor R of $Q^T\tilde{H}Q$, an estimate of the transformed and re-ordered Hessian of the Lagrangian at x (see (6) in Section 10.1 of the document for E04UCF). If **Hessian** = **Yes**, R contains the upper triangular Cholesky factor R of H, the approximate (untransformed) Hessian of the Lagrangian, with the variables in the natural order.

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

On entry: an initial estimate of the solution.

On exit: the final estimate of the solution.

25: IWORK(LIWORK) — INTEGER array *Workspace*

26: LIWORK — INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which E04UNF is called.

Constraint: LIWORK $\geq$ 3 $\times$ N +NCLIN + 2 $\times$ NCNLN.

27: WORK(LWORK) — ***real*** array *Workspace*

28: LWORK — INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which E04UNF is called.

Constraints:

if NCNLN = 0 and NCLIN = 0 then
    LWORK $\geq$ 20 $\times$ N + M $\times$ (N+3);
if NCNLN = 0 and NCLIN > 0 then
    LWORK $\geq$ 2 $\times$ N^2 + 20 $\times$ N + 11 $\times$ NCLIN + M $\times$ (N+3);
if NCNLN > 0 and NCLIN $\geq$ 0 then
    LWORK $\geq$ 2 $\times$ N^2 + N $\times$ NCLIN + 2 $\times$ N $\times$ NCNLN + 20 $\times$ N + 11 $\times$ NCLIN + 21 $\times$ NCNLN + M $\times$ (N+3).

The amounts of workspace provided and required are (by default) output on the current advisory message unit (as defined by X04ABF). As an alternative to computing LIWORK and LWORK from the formulas given above, the user may prefer to obtain appropriate values from the output of a preliminary run with LIWORK and LWORK set to 1. (E04UNF will then terminate with IFAIL = 9.)

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

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

IUSER is not used by E04UNF, but is passed directly to routines CONFUN and OBJFUN and may be used to pass information to those routines.

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

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

USER is not used by E04UNF, but is passed directly to routines CONFUN and OBJFUN and may be used to pass information to those routines.

31: 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 ≠ 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**.

E04UNF returns with IFAIL = 0 if the iterates have converged to a point x that satisfies the first-order Kuhn–Tucker conditions (see Section 10.1 of the document for E04UCF) to the accuracy requested by the optional parameter **Optimality Tolerance** (default value = $\epsilon_R^{0.8}$, where ϵ_R is the value of the optional parameter **Function Precision** (default value = $\epsilon^{0.9}$, where ϵ is the ***machine precision***; see Section 11.2)), i.e., the projected gradient and active constraint residuals are negligible at x.

The user should check whether the following four conditions are satisfied:

(i) the final value of `Norm Gz` (see Section 8.1) is significantly less than that at the starting point;
(ii) during the final major iterations, the values of `Step` and `Mnr` (see Section 8.1) are both one;
(iii) the last few values of both `Norm Gz` and `Violtn` (see Section 8.1) become small at a fast linear rate; and
(iv) `Cond Hz` (see Section 8.1) is small.

If all these conditions hold, x is almost certainly a local minimum of (1).

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 E04UNF because the user set MODE < 0 in routine OBJFUN or CONFUN. The value of IFAIL will be the same as the user's setting of MODE.

IFAIL = 1

The final iterate x satisfies the first-order Kuhn–Tucker conditions (see Section 10.1 of the document for E04UCF) to the accuracy requested, but the sequence of iterates has not yet converged. E04UNF was terminated because no further improvement could be made in the merit function (see Section 8.1).

This value of IFAIL may occur in several circumstances. The most common situation is that the user asks for a solution with accuracy that is not attainable with the given precision of the problem (as specified by the optional parameter **Function Precision** (default value = $\epsilon^{0.9}$, where ϵ is the ***machine precision***; see Section 11.2)). This condition will also occur if, by chance,

an iterate is an 'exact' Kuhn–Tucker point, but the change in the variables was significant at the previous iteration. (This situation often happens when minimizing very simple functions, such as quadratics.)

If the four conditions listed in Section 5 for IFAIL = 0 are satisfied, x is likely to be a solution of (1) even if IFAIL = 1.

IFAIL = 2

E04UNF has terminated without finding a feasible point for the linear constraints and bounds, which means that either no feasible point exists for the given value of the optional parameter **Linear Feasibility Tolerance** (default value = $\sqrt{\epsilon}$, where ϵ is the ***machine precision***; see Section 11.2), or no feasible point could be found in the number of iterations specified by the optional parameter **Minor Iteration Limit** (default value = $\max(50, 3(n+n_L+n_N))$; see Section 11.2). The user should check that there are no constraint redundancies. If the data for the constraints are accurate only to an absolute precision σ, the user should ensure that the value of the optional parameter **Linear Feasibility Tolerance** is greater than σ. For example, if all elements of A_L are of order unity and are accurate to only three decimal places, **Linear Feasibility Tolerance** should be at least 10^{-3}.

IFAIL = 3

No feasible point could be found for the nonlinear constraints. The problem may have no feasible solution. This means that there has been a sequence of QP subproblems for which no feasible point could be found (indicated by `I` at the end of each line of intermediate printout produced by the major iterations; see Section 8.1). This behaviour will occur if there is no feasible point for the nonlinear constraints. (However, there is no general test that can determine whether a feasible point exists for a set of nonlinear constraints.) If the infeasible subproblems occur from the very first major iteration, it is highly likely that no feasible point exists. If infeasibilities occur when earlier subproblems have been feasible, small constraint inconsistencies may be present. The user should check the validity of constraints with negative values of ISTATE. If the user is convinced that a feasible point does exist, E04UNF should be restarted at a different starting point.

IFAIL = 4

The limiting number of iterations (as determined by the optional parameter **Major Iteration Limit** (default value = $\max(50, 3(n+n_L)+10n_N)$; see Section 11.2) has been reached.

If the algorithm appears to be making satisfactory progress, then **Major Iteration Limit** may be too small. If so, either increase its value and rerun E04UNF or, alternatively, rerun E04UNF using the **Warm Start** option (see Section 11.2). If the algorithm seems to be making little or no progress however, then the user should check for incorrect gradients or ill-conditioning as described below under IFAIL = 6.

Note that ill-conditioning in the working set is sometimes resolved automatically by the algorithm, in which case performing additional iterations may be helpful. However, ill-conditioning in the Hessian approximation tends to persist once it has begun, so that allowing additional iterations without altering R is usually inadvisable. If the quasi-Newton update of the Hessian approximation was reset during the latter major iterations (i.e., an `R` occurs at the end of each line of intermediate printout; see Section 8.1), it may be worthwhile to try a **Warm Start** at the final point as suggested above.

IFAIL = 5

Not used by this routine.

IFAIL = 6

x does not satisfy the first-order Kuhn–Tucker conditions (see Section 10.1 of the document for E04UCF), and no improved point for the merit function (see Section 8.1) could be found during the final line search.

This sometimes occurs because an overly stringent accuracy has been requested, i.e., the value of the optional parameter **Optimality Tolerance** (default value = $\epsilon_R^{0.8}$, where ϵ_R is the value of the optional parameter **Function Precision** (default value = $\epsilon^{0.9}$, where ϵ is the ***machine precision***;

see Section 11.2)) is too small. In this case the user should apply the four tests described above under IFAIL = 0 to determine whether or not the final solution is acceptable (see Gill *et al.* [1], for a discussion of the attainable accuracy).

If many iterations have occurred in which essentially no progress has been made and E04UNF has failed completely to move from the initial point then subroutines OBJFUN and/or CONFUN may be incorrect. The user should refer to 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 and constraint gradients that cannot be detected by the verification process. Finite difference approximations to first derivatives are catastrophically affected by even small inaccuracies. An indication of this situation is a dramatic alteration in the iterates if the finite difference interval is altered. One might also suspect this type of error if a switch is made to central differences even when `Norm Gz` and `Violtn` (see Section 8.1) are large.

Another possibility is that the search direction has become inaccurate because of ill-conditioning in the Hessian approximation or the matrix of constraints in the working set; either form of ill-conditioning tends to be reflected in large values of `Mnr` (the number of iterations required to solve each QP subproblem; see Section 8.1).

If the condition estimate of the projected Hessian (`Cond Hz`; see Section 12) is extremely large, it may be worthwhile rerunning E04UNF from the final point with the **Warm Start** option. In this situation, ISTATE and CLAMDA should be left unaltered and R should be reset to the identity matrix.

If the matrix of constraints in the working set is ill-conditioned (i.e., `Cond T` is extremely large; see Section 12), it may be helpful to run E04UNF with a relaxed value of the **Feasibility Tolerance** (default value = $\sqrt{\epsilon}$, where ϵ is the ***machine precision***; see Section 11.2). (Constraint dependencies are often indicated by wide variations in size in the diagonal elements of the matrix T, whose diagonals will be printed if **Major Print Level** ≥ 30 (default value = 10; see Section 11.2)).

IFAIL = 7

The user-provided derivatives of the subfunctions and/or nonlinear constraints appear to be incorrect.

Large errors were found in the derivatives of the subfunctions and/or nonlinear constraints. This value of IFAIL will occur if the verification process indicated that at least one Jacobian 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 subfunction and constraint values is correct – for example, by computing the subfunctions at a point where the correct value of $F(x)$ is known. However, care should be taken that the chosen point fully tests the evaluation of the subfunctions. 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 subfunctions involves subsidiary data communicated in COMMON storage. Although the first evaluation of the subfunctions may be correct, subsequent calculations may be in error because some of the subsidiary data has accidently been overwritten.

Gradient checking will be ineffective if the objective function uses information computed by the constraints, since they are not necessarily computed prior to each function evaluation.

Errors in programming the subfunctions may be quite subtle in that the subfunction values are 'almost' correct. For example, a subfunction 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 subfunction 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 subfunction; 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

Not used by this routine.

IFAIL = 9

An input parameter is invalid.

Overflow

If the printed output before the overflow error contains a warning about serious ill-conditioning in the working set when adding the jth constraint, it may be possible to avoid the difficulty by increasing the magnitude of the optional parameter **Linear Feasibility Tolerance** (default value $= \sqrt{\epsilon}$, where ϵ is the ***machine precision***; see Section 11.2) and/or the optional parameter **Nonlinear Feasibility Tolerance** (default value $= \epsilon^{0.33}$ or $\sqrt{\epsilon}$; see Section 11.2), and rerunning the program. If the message recurs even after this change, the offending linearly dependent constraint (with index 'j') must be removed from the problem. If overflow occurs in one of the user-supplied routines (e.g., if the nonlinear functions involve exponentials or singularities), it may help to specify tighter bounds for some of the variables (i.e., reduce the gap between the appropriate l_j and u_j).

7 Accuracy

If IFAIL = 0 on exit, then the vector returned in the array X is an estimate of the solution to an accuracy of approximately **Optimality Tolerance** (default value $= \epsilon^{0.8}$, where ϵ is the ***machine precision***; see Section 11.2).

8 Further Comments

8.1 Description of Printed Output

This section describes the (default) intermediate printout and final printout produced by E04UNF. The intermediate printout is a subset of the monitoring information produced by the routine at every iteration (see Section 12). The level of printed output can be controlled by the user (see the description of the optional parameter **Major Print Level** in Section 11.2). Note that the intermediate printout and final printout are produced only if **Major Print Level** ≥ 10 (the default).

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

`Maj` is the major iteration count.

`Mnr` is the number of minor iterations required by the feasibility and optimality phases of the QP subproblem. Generally, `Mnr` will be 1 in the later iterations, since theoretical analysis predicts that the correct active set will be identified near the solution (see Section 10 of the document for E04UCF).

Note that `Mnr` may be greater than the **Minor Iteration Limit** (default value $= \max(50, 3(n + n_L + n_N))$; see Section 11.2) if some iterations are required for the feasibility phase.

`Step` is the step taken along the computed search direction. On reasonably well-behaved problems, the unit step will be taken as the solution is approached.

`A` *Alternative optimum possible.* The variable is active at one of its bounds, but its Lagrange multiplier is essentially zero. This means that if the variable were allowed to start moving away from its bound, there would be no change to the objective function. The values of the other free variables *might* change, giving a genuine alternative solution. However, if there are any degenerate variables (labelled `D`), the actual change might prove to be zero, since one of them could encounter a bound immediately. In either case the values of the Lagrange multipliers might also change.

`D` *Degenerate.* The variable is free, but it is equal to (or very close to) one of its bounds.

`I` *Infeasible.* The variable is currently violating one of its bounds by more than the **Feasibility Tolerance**.

`Value` is the value of the variable at the final iterate.

`Lower Bound` is the lower bound specified for the variable. `None` indicates that $\mathrm{BL}(j) \leq -bigbnd$.

`Upper Bound` is the upper bound specified for the variable. `None` indicates that $\mathrm{BU}(j) \geq bigbnd$.

`Lagr Mult` is the Lagrange multiplier for the associated bound. This will be zero if `State` is `FR` unless $\mathrm{BL}(j) \leq -bigbnd$ and $\mathrm{BU}(j) \geq bigbnd$, in which case the entry will be blank. If x is optimal, the multiplier should be non-negative if `State` is `LL`, and non-positive if `State` is `UL`.

`Slack` is the difference between the variable `Value` and the nearer of its (finite) bounds $\mathrm{BL}(j)$ and $\mathrm{BU}(j)$. A blank entry indicates that the associated variable is not bounded (i.e., $\mathrm{BL}(j) \leq -bigbnd$ and $\mathrm{BU}(j) \geq bigbnd$).

The meaning of the printout for linear and nonlinear constraints is the same as that given above for variables, with 'variable' replaced by 'constraint', $\mathrm{BL}(j)$ and $\mathrm{BU}(j)$ are replaced by $\mathrm{BL}(n+j)$ and $\mathrm{BU}(n+j)$ respectively, and with the following changes in the heading:

`L Con` gives the name (`L`) and index j, for $j = 1, 2, \ldots, n_L$ of the linear constraint.

`N Con` gives the name (`N`) and index $(j - n_L)$, for $j = n_L + 1, n_L + 2, \ldots, n_L + n_N$ of the nonlinear constraint.

Note that movement off a constraint (as opposed to a variable moving away from its bound) can be interpreted as allowing the entry in the `Slack` column to become positive.

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

9 Example

This is based on Problem 57 in Hock and Schittkowski [2] and involves the minimization of the sum of squares function

$$F(x) = \frac{1}{2}\sum_{i=1}^{44}(y_i - f_i(x))^2,$$

where

$$f_i(x) = x_1 + (0.49 - x_1)e^{-x_2(a_i - 8)}$$

and

i	y_i	a_i	i	y_i	a_i
1	0.49	8	23	0.41	22
2	0.49	8	24	0.40	22
3	0.48	10	25	0.42	24
4	0.47	10	26	0.40	24
5	0.48	10	27	0.40	24
6	0.47	10	28	0.41	26
7	0.46	12	29	0.40	26
8	0.46	12	30	0.41	26
9	0.45	12	31	0.41	28
10	0.43	12	32	0.40	28
11	0.45	14	33	0.40	30
12	0.43	14	34	0.40	30
13	0.43	14	35	0.38	30
14	0.44	16	36	0.41	32
15	0.43	16	37	0.40	32
16	0.43	16	38	0.40	34
17	0.46	18	39	0.41	36
18	0.45	18	40	0.38	36
19	0.42	20	41	0.40	38
20	0.42	20	42	0.40	38
21	0.43	20	43	0.39	40
22	0.41	22	44	0.39	42

subject to the bounds

$$\begin{aligned} x_1 &\geq 0.4 \\ x_2 &\geq -4.0 \end{aligned}$$

to the general linear constraint

$$x_1 + x_2 \geq 1.0$$

and to the nonlinear constraint

$$0.49x_2 - x_1x_2 \geq 0.09.$$

The initial point, which is infeasible, is

$$x_0 = (0.4, 0.0)^T$$

and $\mathrm{F}(x_0) = 0.002241$.

The optimal solution (to five figures) is

$$x^* = (0.41995, 1.28484)^T,$$

and $\mathrm{F}(x^*) = 0.01423$. The nonlinear constraint is active at the solution.

The document for E04UQF includes an example program to solve the same problem using some of the optional parameters described in Section 11.

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.

```
*     E04UNF Example Program Text
*     Mark 17 Release. NAG Copyright 1995.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          MMAX, NMAX, NCLMAX, NCNMAX
      PARAMETER        (MMAX=50,NMAX=10,NCLMAX=10,NCNMAX=10)
      INTEGER          LDA, LDCJ, LDFJ, LDR
      PARAMETER        (LDA=NCLMAX,LDCJ=NCNMAX,LDFJ=MMAX,LDR=NMAX)
```

```
      INTEGER          LIWORK, LWORK
      PARAMETER        (LIWORK=100,LWORK=1000)
*     .. Local Scalars ..
      real             OBJF
      INTEGER          I, IFAIL, ITER, J, M, N, NCLIN, NCNLN
*     .. Local Arrays ..
      real             A(LDA,NMAX), BL(NMAX+NCLMAX+NCNMAX),
     +                 BU(NMAX+NCLMAX+NCNMAX), C(NCNMAX),
     +                 CJAC(LDCJ,NMAX), CLAMDA(NMAX+NCLMAX+NCNMAX),
     +                 F(MMAX), FJAC(LDFJ,NMAX), R(LDR,NMAX), USER(1),
     +                 WORK(LWORK), X(NMAX), Y(MMAX)
      INTEGER          ISTATE(NMAX+NCLMAX+NCNMAX), IUSER(1),
     +                 IWORK(LIWORK)
*     .. External Subroutines ..
      EXTERNAL         CONFUN, E04UNF, OBJFUN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04UNF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) M, N
      READ (NIN,*) NCLIN, NCNLN
      IF (M.LE.MMAX .AND. N.LE.NMAX .AND. NCLIN.LE.NCLMAX .AND.
     +    NCNLN.LE.NCNMAX) THEN
*
*        Read A, Y, BL, BU and X from data file
*
         IF (NCLIN.GT.0) READ (NIN,*) ((A(I,J),J=1,N),I=1,NCLIN)
         READ (NIN,*) (Y(I),I=1,M)
         READ (NIN,*) (BL(I),I=1,N+NCLIN+NCNLN)
         READ (NIN,*) (BU(I),I=1,N+NCLIN+NCNLN)
         READ (NIN,*) (X(I),I=1,N)
*
*        Solve the problem
*
         IFAIL = -1
*
         CALL E04UNF(M,N,NCLIN,NCNLN,LDA,LDCJ,LDFJ,LDR,A,BL,BU,Y,CONFUN,
     +               OBJFUN,ITER,ISTATE,C,CJAC,F,FJAC,CLAMDA,OBJF,R,X,
     +               IWORK,LIWORK,WORK,LWORK,IUSER,USER,IFAIL)
*
      END IF
      STOP
      END
      SUBROUTINE OBJFUN(MODE,M,N,LDFJ,X,F,FJAC,NSTATE,IUSER,USER)
*     Routine to evaluate the subfunctions and their 1st derivatives.
*     .. Parameters ..
      real             PT49, ONE, EIGHT
      PARAMETER        (PT49=0.49e0,ONE=1.0e0,EIGHT=8.0e0)
*     .. Scalar Arguments ..
      INTEGER          LDFJ, M, MODE, N, NSTATE
*     .. Array Arguments ..
      real             F(*), FJAC(LDFJ,*), USER(*), X(N)
      INTEGER          IUSER(*)
*     .. Local Scalars ..
      real             AI, TEMP, X1, X2
      INTEGER          I
      LOGICAL          MODE02, MODE12
*     .. Local Arrays ..
```

```
      real              A(44)
*     .. Intrinsic Functions ..
      INTRINSIC         EXP
*     .. Data statements ..
      DATA              A/8.0e0, 8.0e0, 10.0e0, 10.0e0, 10.0e0, 10.0e0,
     +                  12.0e0, 12.0e0, 12.0e0, 12.0e0, 14.0e0, 14.0e0,
     +                  14.0e0, 16.0e0, 16.0e0, 16.0e0, 18.0e0, 18.0e0,
     +                  20.0e0, 20.0e0, 20.0e0, 22.0e0, 22.0e0, 22.0e0,
     +                  24.0e0, 24.0e0, 24.0e0, 26.0e0, 26.0e0, 26.0e0,
     +                  28.0e0, 28.0e0, 30.0e0, 30.0e0, 30.0e0, 32.0e0,
     +                  32.0e0, 34.0e0, 36.0e0, 36.0e0, 38.0e0, 38.0e0,
     +                  40.0e0, 42.0e0/
*     .. Executable Statements ..
      X1 = X(1)
      X2 = X(2)
      MODE02 = MODE .EQ. 0 .OR. MODE .EQ. 2
      MODE12 = MODE .EQ. 1 .OR. MODE .EQ. 2
      DO 20 I = 1, M
         AI = A(I)
         TEMP = EXP(-X2*(AI-EIGHT))
         IF (MODE02) F(I) = X1 + (PT49-X1)*TEMP
         IF (MODE12) THEN
            FJAC(I,1) = ONE - TEMP
            FJAC(I,2) = -(PT49-X1)*(AI-EIGHT)*TEMP
         END IF
   20 CONTINUE
*
      RETURN
      END
*
      SUBROUTINE CONFUN(MODE,NCNLN,N,LDCJ,NEEDC,X,C,CJAC,NSTATE,IUSER,
     +                  USER)
*     Routine to evaluate the nonlinear constraint and its 1st
*     derivatives.
*     .. Parameters ..
      real              ZERO, PT09, PT49
      PARAMETER         (ZERO=0.0e0,PT09=0.09e0,PT49=0.49e0)
*     .. Scalar Arguments ..
      INTEGER           LDCJ, MODE, N, NCNLN, NSTATE
*     .. Array Arguments ..
      real              C(*), CJAC(LDCJ,*), USER(*), X(N)
      INTEGER           IUSER(*), NEEDC(*)
*     .. Local Scalars ..
      INTEGER           I, J
*     .. Executable Statements ..
      IF (NSTATE.EQ.1) THEN
*        First call to CONFUN.  Set all Jacobian elements to zero.
*        Note that this will only work when 'Derivative Level = 3'
*        (the default; see Section 11.2).
         DO 40 J = 1, N
            DO 20 I = 1, NCNLN
               CJAC(I,J) = ZERO
   20       CONTINUE
   40    CONTINUE
      END IF
*
      IF (NEEDC(1).GT.0) THEN
         IF (MODE.EQ.0 .OR. MODE.EQ.2) C(1) = -PT09 - X(1)*X(2) +
```

```
     +       PT49*X(2)
             IF (MODE.EQ.1 .OR. MODE.EQ.2) THEN
                CJAC(1,1) = -X(2)
                CJAC(1,2) = -X(1) + PT49
             END IF
          END IF
*
          RETURN
          END
```

9.2 Program Data

```
E04UNF Example Program Data
  44  2                                :Values of M and N
  1   1                                :Values of NCLIN and NCNLN
  1.0   1.0                            :End of matrix A
  0.49  0.49  0.48  0.47  0.48  0.47  0.46  0.46  0.45  0.43  0.45
  0.43  0.43  0.44  0.43  0.43  0.46  0.45  0.42  0.42  0.43  0.41
  0.41  0.40  0.42  0.40  0.40  0.41  0.40  0.41  0.41  0.40  0.40
  0.40  0.38  0.41  0.40  0.40  0.41  0.38  0.40  0.40  0.39  0.39 :End of Y
  0.4      -4.0      1.0      0.0      :End of BL
  1.0E+25   1.0E+25  1.0E+25  1.0E+25  :End of BU
  0.4   0.0                            :End of X
```

9.3 Program Results

```
E04UNF Example Program Results

*** E04UNF
*** 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
----------

Linear constraints.....         1       Variables..............         2
Nonlinear constraints..         1       Subfunctions...........        44

Infinite bound size....  1.00E+20       COLD start.............
Infinite step size.....  1.00E+20       EPS (machine precision)  1.11E-16
Step limit.............  2.00E+00       Hessian................        NO

Linear feasibility.....  1.05E-08       Crash tolerance........  1.00E-02
Nonlinear feasibility..  1.05E-08       Optimality tolerance...  3.26E-12
Line search tolerance..  9.00E-01       Function precision.....  4.37E-15

Derivative level.......         3       Monitoring file........        -1
Verify level...........         0

Major iterations limit.        50       Major print level......        10
```

```
Minor iterations limit.        50       Minor print level......         0

J'J initial Hessian....                 Reset frequency........         2

Workspace provided is     IWORK(     100),  WORK(    1000).
To solve problem we need  IWORK(       9),  WORK(     306).


Verification of the constraint gradients.
-----------------------------------------

The constraint Jacobian seems to be ok.

The largest relative error was    1.89E-08  in constraint    1



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

The objective Jacobian seems to be ok.

The largest relative error was    1.04E-08  in subfunction    3



 Maj  Mnr    Step Merit Function Norm Gz  Violtn Cond Hz
   0    2 0.0E+00   2.224070E-02 8.5E-02 3.6E-02 1.0E+00
   1    1 1.0E+00   1.455402E-02 1.5E-03 9.8E-03 1.0E+00
   2    1 1.0E+00   1.436491E-02 4.9E-03 7.2E-04 1.0E+00
   3    1 1.0E+00   1.427013E-02 2.9E-03 9.2E-06 1.0E+00
   4    1 1.0E+00   1.422989E-02 1.6E-04 3.6E-05 1.0E+00
   5    1 1.0E+00   1.422983E-02 5.4E-07 6.4E-08 1.0E+00
   6    1 1.0E+00   1.422983E-02 3.4E-09 9.8E-13 1.0E+00

Exit from NP problem after     6 major iterations,
                               8 minor iterations.


Varbl State     Value       Lower Bound   Upper Bound    Lagr Mult   Slack

V   1    FR   0.419953      0.400000          None           .      1.9953E-02
V   2    FR    1.28485      -4.00000          None           .       5.285


L Con State     Value       Lower Bound   Upper Bound    Lagr Mult   Slack

L   1    FR    1.70480       1.00000          None           .      0.7048


N Con State     Value       Lower Bound   Upper Bound    Lagr Mult   Slack

N   1    LL  -9.767742E-13       .            None      3.3358E-02 -9.7677E-13

Exit E04UNF - Optimal solution found.

Final objective value =   0.1422983E-01
```

The remainder of this document is intended for more advanced users. Section 11 describes the optional parameters which may be set by calls to E04UQF and/or E04URF. Section 12 describes the quantites which can be requested to monitor the course of the computation.

10 Algorithmic Details

E04UNF implements a sequential quadratic programming (SQP) method incorporating an augmented Lagrangian merit function and a BFGS (Broyden–Fletcher–Goldfarb–Shanno) quasi-Newton approximation to the Hessian of the Lagrangian, and is based on E04UCF. The documents for E04UCF and E04NCF should be consulted for details of the method.

11 Optional Parameters

Several optional parameters in E04UNF define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of E04UNF 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 E04UQF and E04URF prior to a call to E04UNF.

E04UQF 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 E04UQF (IOPTNS, INFORM)
```

can then be used to read the file on unit IOPTNS. INFORM will be zero on successful exit. E04UQF should be consulted for a full description of this method of supplying optional parameters.

E04URF can be called to supply options directly, one call being necessary for each optional parameter. For example,

```
CALL E04URF ('Print level = 1')
```

E04URF 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 E04UNF (unless they define invalid values) and so remain in effect for subsequent calls to E04UNF, 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**
Central difference interval	Computed automatically
Cold/Warm start	**Cold start**
Crash tolerance	0.01

Defaults	
Derivative level	3
Difference interval	Computed automatically
Feasibility tolerance	$\sqrt{\epsilon}$
Function precision	$\epsilon^{0.9}$
Hessian	**No**
Infinite bound size	10^{20}
Infinite step size	10^{20}
JTJ/Unit initial hessian	**JTJ initial hessian**
Line search tolerance	0.9
Linear feasibility tolerance	$\sqrt{\epsilon}$
List/Nolist	**List**
Major iteration limit	$\max(50, 3(n + n_L) + 10n_N)$
Major print level	10
Minor iteration limit	$\max(50, 3(n + n_L + n_N))$
Minor print level	0
Monitoring file	-1
Nonlinear feasibility tolerance	$\epsilon^{0.33}$ or $\sqrt{\epsilon}$
Optimality tolerance	$\epsilon_R^{0.8}$
Reset frequency	2
Step limit	2.0
Start objective check	1
Start constraint check	1
Stop objective check	n
Stop constraint check	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 (the optional parameter **Function Precision**; see below). Further details of other quantities not explicitly defined in this section may be found by consulting the document for E04UCF.

Central Difference Interval r Default values are computed

If the algorithm switches to central differences because the forward-difference approximation is not sufficiently accurate, the value of r is used as the difference interval for every element of x. The switch to central differences is indicated by `C` at the end of each line of intermediate printout produced by the major iterations (see Section 8.1). The use of finite differences is discussed further below under the optional parameter **Difference Interval**.

Cold Start Default = **Cold Start**

Warm Start

This option controls the specification of the initial working set in both the procedure for finding a feasible point for the linear constraints and bounds, and in the first QP subproblem thereafter. With a **Cold Start**, the first working set is chosen by E04UNF based on the values of the variables and constraints at the initial point. Broadly speaking, the initial working set will include equality constraints and bounds or inequality constraints that violate or 'nearly' satisfy their bounds (to within **Crash Tolerance**; see below).

With a **Warm Start**, the user must set the ISTATE array and define CLAMDA and R as discussed in Section 5. ISTATE values associated with bounds and linear constraints determine the initial working set of the procedure to find a feasible point with respect to the bounds and linear constraints. ISTATE values associated with nonlinear constraints determine the initial working set of the first QP subproblem after such a feasible point has been found. E04UNF will override the user's specification of ISTATE if necessary, so that a poor choice of the working set will not cause a fatal error. For instance, any elements

of ISTATE which are set to -2, -1 or 4 will be reset to zero, as will any elements which are set to 3 when the corresponding elements of BL and BU are not equal. A warm start will be advantageous if a good estimate of the initial working set is available – for example, when E04UNF is called repeatedly to solve related problems.

Crash Tolerance r Default = 0.01

This value is used in conjunction with the optional parameter **Cold Start** (the default value) when E04UNF selects an initial working set. If $0 \leq r \leq 1$, the initial working set will include (if possible) bounds or general inequality constraints that lie within r of their bounds. In particular, a constraint of the form $a_j^T x \geq l$ will be included in the initial working set if $|a_j^T x - l| \leq r(1 + |l|)$. If $r < 0$ or $r > 1$, the default value is used.

Defaults

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

Derivative Level i Default = 3

This parameter indicates which derivatives are provided by the user in subroutines OBJFUN and CONFUN. The possible choices for i are the following.

i	**Meaning**
3	All elements of the objective Jacobian and the constraint Jacobian are provided by the user.
2	All elements of the constraint Jacobian are provided, but some elements of the objective Jacobian are not specified by the user.
1	All elements of the objective Jacobian are provided, but some elements of the constraint Jacobian are not specified by the user.
0	Some elements of both the objective Jacobian and the constraint Jacobian are not specified by the user.

The value $i = 3$ should be used whenever possible, since E04UNF is more reliable (and will usually be more efficient) when all derivatives are exact.

If $i = 0$ or 2, E04UNF will approximate unspecified elements of the objective Jacobian, using finite differences. The computation of finite difference approximations usually increases the total run-time, since a call to OBJFUN is required for each unspecified element. Furthermore, less accuracy can be attained in the solution (see Chapter 8 of Gill *et al.* [1], for a discussion of limiting accuracy).

If $i = 0$ or 1, E04UNF will approximate unspecified elements of the constraint Jacobian. One call to CONFUN is needed for each variable for which partial derivatives are not available. For example, if the constraint Jacobian has the form

$$\begin{pmatrix} * & * & * & * \\ * & ? & ? & * \\ * & * & ? & * \\ * & * & * & * \end{pmatrix}$$

where '$*$' indicates an element provided by the user and '?' indicates an unspecified element, E04UNF will call CONFUN twice: once to estimate the missing element in column 2, and again to estimate the two missing elements in column 3. (Since columns 1 and 4 are known, they require no calls to CONFUN.)

At times, central differences are used rather than forward differences, in which case twice as many calls to OBJFUN and CONFUN are needed. (The switch to central differences is not under the user's control.)

If $i < 0$ or $i > 3$, the default value is used.

Difference Interval r Default values are computed

This option defines an interval used to estimate derivatives by finite differences in the following circumstances:

(a) For verifying the objective and/or constraint gradients (see the description of **Verify**, below).
(b) For estimating unspecified elements of the objective and/or constraint Jacobian matrix.

In general, a derivative with respect to the jth variable is approximated using the interval δ_j, where $\delta_j = r(1 + |\hat{x}_j|)$, with $\hat{x}$ the first point feasible with respect to the bounds and linear constraints. If the

functions are well scaled, the resulting derivative approximation should be accurate to O(r). See Gill *et al.* [1] for a discussion of the accuracy in finite difference approximations.

If a difference interval is not specified by the user, a finite difference interval will be computed automatically for each variable by a procedure that requires up to six calls of CONFUN and OBJFUN for each element. This option is recommended if the function is badly scaled or the user wishes to have E04UNF determine constant elements in the objective and constraint gradients (see the descriptions of CONFUN and OBJFUN in Section 5).

Feasibility Tolerance r Default = $\sqrt{\epsilon}$

The scalar r defines the maximum acceptable *absolute* violations in linear and nonlinear constraints at a 'feasible' point; i.e., a constraint is considered satisfied if its violation does not exceed r. If $r < \epsilon$ or $r \geq 1$, the default value is used. Using this keyword sets both optional parameters **Linear Feasibility Tolerance** and **Nonlinear Feasibility Tolerance** to r, if $\epsilon \leq r < 1$. (Additional details are given below under the descriptions of these parameters.)

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

This parameter defines ϵ_R, which is intended to be a measure of the accuracy with which the problem functions $F(x)$ and $c(x)$ 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 *et al.* [1] 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 E04UNF will not attempt to distinguish between function values that differ by less than the error inherent in the calculation.

Hessian **No** Default = **No**

Hessian **Yes**

This option controls the contents of the upper triangular matrix R (see Section 5). E04UNF works exclusively with the *transformed and re-ordered* Hessian H_Q, and hence extra computation is required to form the Hessian itself. If **Hessian** = **No**, R contains the Cholesky factor of the transformed and re-ordered Hessian. If **Hessian** = **Yes**, the Cholesky factor of the approximate Hessian itself is formed and stored in R. The user should select **Hessian** = **Yes** if a **Warm Start** will be used for the next call to E04UNF.

Infinite Bound Size r Default = 10^{20}

If $r > 0$, r defines the 'infinite' bound *bigbnd* in the definition of the problem constraints. Any upper bound greater than or equal to *bigbnd* will be regarded as plus infinity (and similarly any lower bound less than or equal to $-$*bigbnd* will be regarded as minus infinity). If $r \leq 0$, the default value is used.

Infinite Step Size r Default = $\max(bigbnd, 10^{20})$

If $r > 0$, r specifies the magnitude of the change in variables that is treated as a step to an unbounded solution. If the change in x during an iteration would exceed the value of r, the objective function is considered to be unbounded below in the feasible region. If $r \leq 0$, the default value is used.

Iteration Limit i Default = $\max(50, 3(n + n_L) + 10n_N)$

See **Major Iteration Limit** below.

JTJ Initial Hessian Default = **JTJ Initial Hessian**

Unit Initial Hessian

This option controls the initial value of the upper triangular matrix R. If J denotes the objective Jacobian matrix $\nabla f(x)$, then $J^T J$ is often a good approximation to the objective Hessian matrix $\nabla^2 F(x)$ (see also **Reset Frequency**, below).

Line Search 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 merit function along the search direction (the smaller the value of r, the more accurate the line search). The default value $r = 0.9$ requests an inaccurate search, and is appropriate for most problems, particularly those with any nonlinear constraints.

If there are no nonlinear constraints, a more accurate search may be appropriate when it is desirable to reduce the number of major iterations – for example, if the objective function is cheap to evaluate, or if a substantial number of derivatives are unspecified. If $r < 0$ or $r \geq 1$, the default value is used.

Linear Feasibility Tolerance r_1 Default = $\sqrt{\epsilon}$

Nonlinear Feasibility Tolerance r_2 Default = $\epsilon^{0.33}$ or $\sqrt{\epsilon}$ (see below)

The default value of r_2 is $\epsilon^{0.33}$ if **Derivative Level** = 0 or 1, and $\sqrt{\epsilon}$ otherwise.

The scalars r_1 and r_2 define the maximum acceptable *absolute* violations in linear and nonlinear constraints at a 'feasible' point; i.e., a linear constraint is considered satisfied if its violation does not exceed r_1, and similarly for a nonlinear constraint and r_2. If $r_m < \epsilon$ or $r_m \geq 1$, the default value is used, for $m = 1, 2$.

On entry to E04UNF, an iterative procedure is executed in order to find a point that satisfies the linear constraints and bounds on the variables to within the tolerance r_1. All subsequent iterates will satisfy the linear constraints to within the same tolerance (unless r_1 is comparable to the finite difference interval).

For nonlinear constraints, the feasibility tolerance r_2 defines the largest constraint violation that is acceptable at an optimal point. Since nonlinear constraints are generally not satisfied until the final iterate, the value of **Nonlinear Feasibility Tolerance** acts as a partial termination criterion for the iterative sequence generated by E04UNF (see also **Optimality Tolerance**, below).

These tolerances should reflect the precision of the corresponding constraints. For example, if the variables and the coefficients in the linear constraints are of order unity, and the latter are correct to about 6 decimal digits, it would be appropriate to specify r_1 as 10^{-6}.

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.

Major Iteration Limit i Default = $\max(50, 3(n + n_L) + 10n_N)$

Iteration Limit

Iters

Itns

The value of i specifies the maximum number of major iterations allowed before termination. Setting i = 0 and **Major Print Level** > 0 means that the workspace needed will be computed and printed, but no iterations will be performed. If $i < 0$, the default value is used.

Major Print Level i Default = 10

Print Level

The value of i controls the amount of printout produced by the major iterations of E04UNF, as indicated below. A detailed description of the printed output is given in Section 8.1 (summary output at each major iteration and the final solution) and Section 12 (monitoring information at each major iteration). (See also **Minor Print Level**, below.)

The following printout is sent to the current advisory message unit (as defined by X04ABF):

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

The following printout is sent to the logical unit number defined by the optional parameter **Monitoring File** (see below):

i	**Output**
< 5	No output.
≥ 5	One long line of output (> 80 characters; see Section 12) for each major iteration (no printout of the final solution).
≥ 20	At each major iteration, the objective function, the Euclidean norm of the nonlinear constraint violations, the values of the nonlinear constraints (the vector c), the values of the linear constraints (the vector $A_L x$), and the current values of the variables (the vector x).
≥ 30	At each major iteration, the diagonal elements of the matrix T associated with the TQ factorization (see (5) Section 10.1 of the document for E04UCF) of the QP working set, and the diagonal elements of R, the triangular factor of the transformed and re-ordered Hessian (see (6) Section 10.1 of the document for E04UCF).

If **Major Print Level** ≥ 5 and the unit number defined by **Monitoring File** is the same as that defined by X04ABF, then the summary output for each major iteration is suppressed.

Minor Iteration Limit i Default $= \max(50, 3(n + n_L + n_N))$

The value of i specifies the maximum number of iterations for finding a feasible point with respect to the bounds and linear constraints (if any). The value of i also specifies the maximum number of minor iterations for the optimality phase of each QP subproblem. If $i \leq 0$, the default value is used.

Minor Print Level i Default $= 0$

The value of i controls the amount of printout produced by the minor iterations of E04UNF (i.e., the iterations of the quadratic programming algorithm), as indicated below. A detailed description of the printed output is given in ? (summary output at each minor iteration and the final QP solution) and Section 12 (monitoring information at each minor iteration) of the document for E04NCF. (See also **Major Print Level**, above.)

The following printout is sent to the current advisory message unit (as defined by X04ABF):

i	**Output**
0	No output.
1	The final QP solution only.
5	One line of summary output (< 80 characters; see Section 8.1 of the document for E04NCF) for each minor iteration (no printout of the final QP solution).
≥ 10	The final QP solution and one line of summary output for each minor iteration.

The following printout is sent to the logical unit number defined by the optional parameter **Monitoring File** (see below):

i	**Output**
< 5	No output.
≥ 5	One long line of output (> 80 characters; see Section 12 of the document for E04NCF) for each minor iteration (no printout of the final QP solution).
≥ 20	At each minor iteration, the current estimates of the QP multipliers, the current estimate of the QP search direction, the QP constraint values, and the status of each QP constraint.
≥ 30	At each minor iteration, the diagonal elements of the matrix T associated with the TQ factorization (see (5) Section 10.1 of the document for E04UCF) of the QP working set, and the diagonal elements of the Cholesky factor R of the transformed Hessian (see (6) Section 10.1 of the document for E04UCF).

If **Minor Print Level** ≥ 5 and the unit number defined by **Monitoring File** is the same as that defined by X04ABF, then the summary output for each minor iteration is suppressed.

Monitoring File i Default $= -1$

If $i \geq 0$ and **Major Print Level** ≥ 5 (see above) or $i \geq 0$ and **Minor Print Level** ≥ 5 (see above), monitoring information produced by E04UNF at every iteration is sent to a file with logical unit number i. If $i < 0$ and/or **Major Print Level** < 5 and **Minor Print Level** < 5, no monitoring information is produced.

Nonlinear Feasibility Tolerance r Default $= \epsilon^{0.33}$ or $\sqrt{\epsilon}$

See **Linear Feasibility Tolerance** above.

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 E04UNF terminates successfully, the final value of F should have approximately six correct figures. If $r < \epsilon_R$ or $r \geq 1$, the default value is used.

E04UNF will terminate successfully if the iterative sequence of x-values is judged to have converged and the final point satisfies the first-order Kuhn–Tucker conditions (see Section 10.1 of the document for E04UCF). The sequence of iterates is considered to have converged at x if

$$\alpha\|p\| \leq \sqrt{r}(1 + \|x\|), \tag{2}$$

where p is the search direction and α the step length. An iterate is considered to satisfy the first-order conditions for a minimum if

$$\|Z^T g_{\mathrm{FR}}\| \leq \sqrt{r}(1 + \max(1 + |F(x)|, \|g_{\mathrm{FR}}\|)) \tag{3}$$

and

$$|res_j| \leq ftol \text{ for all } j, \tag{4}$$

where $Z^T g_{\mathrm{FR}}$ is the projected gradient, g_{FR} is the gradient of $F(x)$ with respect to the free variables, res_j is the violation of the jth active nonlinear constraint, and $ftol$ is the **Nonlinear Feasibility Tolerance**.

Print Level

See **Major Print Level** above.

Reset Frequency i Default $= 2$

If $i > 0$, this parameter allows the user to reset the approximate Hessian matrix to $J^T J$ every i iterations, where J is the objective Jacobian matrix $\nabla f(x)$ (see also **JTJ Initial Hessian**, above).

At any point where there are no nonlinear constraints active and the values of f are small in magnitude compared to the norm of J, $J^T J$ will be a good approximation to the objective Hessian $\nabla^2 F(x)$. Under these circumstances, frequent resetting can significantly improve the convergence rate of E04UNF.

Resetting is suppressed at any iteration during which there are nonlinear constraints active.

If $i \leq 0$, the default value is used.

Start Objective Check At Variable i_1 Default $= 1$

Stop Objective Check At Variable i_2 Default $= n$

Start Constraint Check At Variable i_3 Default $= 1$

Stop Constraint Check At Variable i_4 Default $= n$

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

If $i_{2m-1} \leq 0$ or $i_{2m-1} > \min(n, i_{2m})$, the default value is used, for $m = 1, 2$. If $i_{2m} \leq 0$ or $i_{2m} > n$, the default value is used, for $m = 1, 2$.

Step Limit r Default $= 2.0$

If $r > 0, r$ specifies the maximum change in variables at the first step of the line search. In some cases, such as $F(x) = ae^{bx}$ or $F(x) = ax^b$, even a moderate change in the elements of x can lead to floating-point overflow. The parameter r is therefore used to encourage evaluation of the problem functions at

meaningful points. Given any major iterate x, the first point $\tilde{x}$ at which F and c are evaluated during the line search is restricted so that

$$\|\tilde{x} - x\|_2 \leq r(1 + \|x\|_2).$$

The line search may go on and evaluate F and c at points further from x if this will result in a lower value of the merit function (indicated by `L` at the end of each line of output produced by the major iterations; see Section 8.1). If `L` is printed for most of the iterations, r should be set to a larger value.

Wherever possible, upper and lower bounds on x should be used to prevent evaluation of nonlinear functions at wild values. The default value **Step Limit** $= 2.0$ should not affect progress on well-behaved functions, but values such as 0.1 or 0.01 may be helpful when rapidly varying functions are present. If a small value of **Step Limit** is selected, a good starting point may be required. An important application is to the class of nonlinear least-squares problems. If $r \leq 0$, the default value is used.

Stop Constraint Check At Variable

See **Start Constraint Check At Variable** above.

Stop Objective Check At Variable

See **Start Objective Check At Variable** above.

Unit Initial Hessian

See **JTJ Initial Hessian** above.

Verify Level	i	Default $= 0$
Verify No		
Verify Level	-1	
Verify Level	0	
Verify Objective Gradients		
Verify Level	1	
Verify Constraint Gradients		
Verify Level	2	
Verify		
Verify Yes		
Verify Gradients		
Verify Level	3	

These keywords refer to finite difference checks on the gradient elements computed by the user-provided subroutines OBJFUN and CONFUN. (Unspecified gradient elements are not checked.) It is possible to specify **Verify Level**s $0 - 3$ in several ways, as indicated above. For example, the nonlinear objective gradient (if any) will be verified if either **Verify Objective Gradients** or **Verify Level** 1 is specified. Similarly, the objective and the constraint gradients will be verified if **Verify Yes** or **Verify Level** 3 or **Verify** is specified.

If $i = -1$, then no checking will be performed.

If $0 \leq i \leq 3$, gradients will be verified at the first point that satisfies the linear constraints and bounds. If $i = 0$, only a 'cheap' test will be performed, requiring one call to OBJFUN and (if appropriate) one call to CONFUN. If $1 \leq i \leq 3$, a more reliable (but more expensive) check will be made on individual gradient elements, within the ranges specified by the **Start** and **Stop** keywords described above. A result of the form `OK` or `BAD?` is printed by E04UNF to indicate whether or not each element appears to be correct.

If $10 \leq i \leq 13$, the action is the same as for $i - 10$, except that it will take place at the user-specified initial value of x.

If $i < -1$ or $4 \leq i \leq 9$ or $i > 13$, the default value is used.

We suggest that **Verify Level** $= 3$ be used whenever a new function routine is being developed.

Warm Start

See **Cold Start** above.

12 Description of Monitoring Information

This section describes the long line of output (> 80 characters) which forms part of the monitoring information produced by E04UNF. (See also the description of the optional parameters **Major Print Level**, **Minor Print Level** and **Monitoring File** in Section 11.2). The level of printed output can be controlled by the user.

When **Major Print Level** ≥ 5 and **Monitoring File** ≥ 0, the following line of output is produced at every major iteration of E04UNF on the unit number specified by **Monitoring File**. In all cases, the values of the quantities printed are those in effect *on completion* of the given iteration.

`Maj` is the major iteration count.

`Mnr` is the number of minor iterations required by the feasibility and optimality phases of the QP subproblem. Generally, `Mnr` will be 1 in the later iterations, since theoretical analysis predicts that the correct active set will be identified near the solution (see Section 10 of the document for E04UCF).

Note that `Mnr` may be greater than the **Minor Iteration Limit** if some iterations are required for the feasibility phase.

`Step` is the step taken along the computed search direction. On reasonably well-behaved problems, the unit step will be taken as the solution is approached.

`Nfun` is the cumulative number of evaluations of the objective function needed for the line search. Evaluations needed for the estimation of the gradients by finite differences are not included. `Nfun` is printed as a guide to the amount of work required for the line search.

`Merit Function` is the value of the augmented Lagrangian merit function (12) at the current iterate. This function will decrease at each iteration unless it was necessary to increase the penalty parameters (see Section 10.3 of the document for E04UCF). As the solution is approached, `Merit Function` will converge to the value of the objective function at the solution.

If the QP subproblem does not have a feasible point (signified by `I` at the end of the current output line), the merit function is a large multiple of the constraint violations, weighted by the penalty parameters. During a sequence of major iterations with infeasible subproblems, the sequence of `Merit Function` values will decrease monotonically until either a feasible subproblem is obtained or E04UNF terminates with IFAIL = 3 (no feasible point could be found for the nonlinear constraints).

If no nonlinear constraints are present (i.e., NCNLN = 0), this entry contains `Objective`, the value of the objective function $F(x)$. The objective function will decrease monotonically to its optimal value when there are no nonlinear constraints.

`Norm Gz` is $\|Z^T g_{\mathrm{FR}}\|$, the Euclidean norm of the projected gradient (see Section 10.2 of the document for E04UCF). `Norm Gz` will be approximately zero in the neighbourhood of a solution.

`Violtn` is the Euclidean norm of the residuals of constraints that are violated or in the predicted active set (not printed if NCNLN is zero). `Violtn` will be approximately zero in the neighbourhood of a solution.

`Nz` is the number of columns of Z (see Section 10.2 of the document for E04UCF). The value of `Nz` is the number of variables minus the number of constraints in the predicted active set; i.e., `Nz` $= n -$ (`Bnd` + `Lin` + `Nln`).

`Bnd` is the number of simple bound constraints in the predicted active set.

`Lin` is the number of general linear constraints in the predicted active set.

`Nln` is the number of nonlinear constraints in the predicted active set (not printed if NCNLN is zero).

`Penalty` is the Euclidean norm of the vector of penalty parameters used in the augmented Lagrangian merit function (not printed if NCNLN is zero).

`Cond H` is a lower bound on the condition number of the Hessian approximation H.

`Cond Hz` is a lower bound on the condition number of the projected Hessian approximation H_Z ($H_Z = Z^T H_{\mathrm{FR}} Z = R_Z^T R_Z$; see (6) in Section 10.1 of the document for E04UCF and (11) in Section 10.2 of the document for E04UCF). The larger this number, the more difficult the problem.

`Cond T` is a lower bound on the condition number of the matrix of predicted active constraints.

`Conv` is a three-letter indication of the status of the three convergence tests (2)–(4) defined in the description of the optional parameter **Optimality Tolerance** in Section 11.2. Each letter is `T` if the test is satisfied, and `F` otherwise. The three tests indicate whether:

(a) the sequence of iterates has converged;
(b) the projected gradient (`Norm Gz`) is sufficiently small; and
(c) the norm of the residuals of constraints in the predicted active set (`Violtn`) is small enough.

If any of these indicators is `F` when E04UNF terminates with IFAIL = 0, the user should check the solution carefully.

`M` is printed if the quasi-Newton update has been modified to ensure that the Hessian approximation is positive-definite (see Section 10.4 of the document for E04UCF).

`I` is printed if the QP subproblem has no feasible point.

`C` is printed if central differences have been used to compute the unspecified objective and constraint gradients. If the value of `Step` is zero, the switch to central differences was made because no lower point could be found in the line search. (In this case, the QP subproblem is re-solved with the central difference gradient and Jacobian.) If the value of `Step` is non-zero, central differences were computed because `Norm Gz` and `Violtn` imply that x is close to a Kuhn–Tucker point (see Section 10.1 of the document for E04UCF).

`L` is printed if the line search has produced a relative change in x greater than the value defined by the optional parameter **Step Limit**. If this output occurs frequently during later iterations of the run, **Step Limit** should be set to a larger value.

`R` is printed if the approximate Hessian has been refactorized. If the diagonal condition estimator of R indicates that the approximate Hessian is badly conditioned, the approximate Hessian is refactorized using column interchanges. If necessary, R is modified so that its diagonal condition estimator is bounded.
