NAG Library Routine Document

E04NCF/E04NCA

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 Sections 1 to 10 of this document. If, however, you wish to reset some or all of the settings please refer to Section 11 for a detailed description of the algorithm, to Section 12 for a detailed description of the specification of the optional parameters and to Section 13 for a detailed description of the monitoring information produced by the routine.*

1 Purpose

E04NCF/E04NCA solves linearly constrained linear least squares problems and convex quadratic programming problems. It is not intended for large sparse problems.

E04NCA is a version of E04NCF that has additional arguments in order to make it safe for use in multithreaded applications (see Section 5). The initialization routine E04WBF **must** have been called before calling E04NCA.

2 Specification

2.1 Specification for E04NCF

```
SUBROUTINE E04NCF (M, N, NCLIN, LDC, LDA, C, BL, BU, CVEC, ISTATE, KX,      &
                   X, A, B, ITER, OBJ, CLAMDA, IWORK, LIWORK, WORK,         &
                   LWORK, IFAIL)

INTEGER            M, N, NCLIN, LDC, LDA, ISTATE(N+NCLIN), KX(N), ITER,     &
                   IWORK(LIWORK), LIWORK, LWORK, IFAIL
REAL (KIND=nag_wp) C(LDC,*), BL(N+NCLIN), BU(N+NCLIN), CVEC(*), X(N),      &
                   A(LDA,*), B(*), OBJ, CLAMDA(N+NCLIN), WORK(LWORK)
```

2.2 Specification for E04NCA

```
SUBROUTINE E04NCA (M, N, NCLIN, LDC, LDA, C, BL, BU, CVEC, ISTATE, KX,      &
                   X, A, B, ITER, OBJ, CLAMDA, IWORK, LIWORK, WORK,         &
                   LWORK, LWSAV, IWSAV, RWSAV, IFAIL)

INTEGER            M, N, NCLIN, LDC, LDA, ISTATE(N+NCLIN), KX(N), ITER,     &
                   IWORK(LIWORK), LIWORK, LWORK, IWSAV(610), IFAIL
REAL (KIND=nag_wp) C(LDC,*), BL(N+NCLIN), BU(N+NCLIN), CVEC(*), X(N),      &
                   A(LDA,*), B(*), OBJ, CLAMDA(N+NCLIN), WORK(LWORK),       &
                   RWSAV(475)
LOGICAL            LWSAV(120)
```

Before calling E04NCA, or either of the option setting routines E04NDA or E04NEA, E04WBF **must** be called. The specification for E04WBF is:

```
SUBROUTINE E04WBF (RNAME, CWSAV, LCWSAV, LWSAV, LLWSAV, IWSAV, LIWSAV,      &
                   RWSAV, LRWSAV, IFAIL)

INTEGER            LCWSAV, LLWSAV, IWSAV(LIWSAV), LIWSAV, LRWSAV,           &
                   IFAIL
REAL (KIND=nag_wp) RWSAV(LRWSAV)
LOGICAL            LWSAV(LLWSAV)
CHARACTER(*)       RNAME
CHARACTER(80)      CWSAV(LCWSAV)
```

E04WBF should be called with RNAME = 'E04NCA'. LCWSAV, LLWSAV, LIWSAV and LRWSAV, the declared lengths of CWSAV, LWSAV, IWSAV and RWSAV respectively, must satisfy:

LCWSAV ≥ 1

LLWSAV ≥ 120

LIWSAV ≥ 610

LRWSAV ≥ 475

The contents of the arrays CWSAV, LWSAV, IWSAV and RWSAV **must not** be altered between calling routines E04NCA, E04NDA, E04NEA and E04WBF.

3 Description

E04NCF/E04NCA is designed to solve a class of quadratic programming problems of the following general form:

$$\underset{x \in R^n}{\text{minimize}} F(x) \quad \text{subject to} \quad l \leq \left\{ \begin{matrix} x \\ Cx \end{matrix} \right\} \leq u \tag{1}$$

where C is an n_L by n matrix and the objective function $F(x)$ may be specified in a variety of ways depending upon the particular problem to be solved. The available forms for $F(x)$ are listed in Table 1, in which the prefixes FP, LP, QP and LS stand for 'feasible point', 'linear programming', 'quadratic programming' and 'least squares' respectively, c is an n-element vector, b is an m element vector and $\|z\|$ denotes the Euclidean length of z.

Problem type	$F(x)$	Matrix A
FP	None	Not applicable
LP	$c^{\mathrm{T}}x$	Not applicable
QP1	$\frac{1}{2}x^{\mathrm{T}}Ax$	n by n symmetric positive semidefinite
QP2	$c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Ax$	n by n symmetric positive semidefinite
QP3	$\frac{1}{2}x^{\mathrm{T}}A^{\mathrm{T}}Ax$	m by n upper trapezoidal
QP4	$c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}A^{\mathrm{T}}Ax$	m by n upper trapezoidal
LS1	$\frac{1}{2}\|b - Ax\|^2$	m by n
LS2	$c^{\mathrm{T}}x + \frac{1}{2}\|b - Ax\|^2$	m by n
LS3	$\frac{1}{2}\|b - Ax\|^2$	m by n upper trapezoidal
LS4	$c^{\mathrm{T}}x + \frac{1}{2}\|b - Ax\|^2$	m by n upper trapezoidal

In the standard LS problem $F(x)$ will usually have the form LS1, and in the standard convex QP problem $F(x)$ will usually have the form QP2. The default problem type is LS1 and other objective functions are selected by using the optional parameter **Problem Type**.

When A is upper trapezoidal it will usually be the case that $m = n$, so that A is upper triangular, but full generality has been allowed for in the specification of the problem. The upper trapezoidal form is intended for cases where a previous factorization, such as a QR factorization, has been performed.

The constraints involving C are called the *general* constraints. Note that upper and lower bounds are specified for all the variables and for all the general 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**.)

The defining feature of a quadratic function $F(x)$ is that the second-derivative matrix H (the *Hessian matrix*) is constant. For the LP case $H = 0$; for QP1 and QP2, $H = A$; for QP3 and QP4, $H = A^{\mathrm{T}}A$ and for LS1 (the default), LS2, LS3 and LS4, $H = A^{\mathrm{T}}A$.

Problems of type QP3 and QP4 for which A is not in upper trapezoidal form should be solved as types LS1 and LS2 respectively, with $b = 0$.

For problems of type LS, we refer to A as the *least squares* matrix, or the *matrix of observations* and to b as the *vector of observations.*

You must supply an initial estimate of the solution.

If H is nonsingular then E04NCF/E04NCA will obtain the unique (global) minimum. If H is singular then the solution may still be a global minimum if all active constraints have nonzero Lagrange multipliers. Otherwise the solution obtained will be either a weak minimum (i.e., with a unique optimal objective value, but an infinite set of optimal x), or else the objective function is unbounded below in the feasible region. The last case can only occur when $F(x)$ contains an explicit linear term (as in problems LP, QP2, QP4, LS2 and LS4).

The method used by E04NCF/E04NCA is described in detail in Section 11.

4 References

Gill P E, Hammarling S, Murray W, Saunders M A and Wright M H (1986) Users' guide for LSSOL (Version 1.0) *Report SOL 86-1* Department of Operations Research, Stanford University

Gill P E, Murray W, Saunders M A and Wright M H (1984) Procedures for optimization problems with a mixture of bounds and general linear constraints *ACM Trans. Math. Software* **10** 282–298

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

Stoer J (1971) On the numerical solution of constrained least squares problems *SIAM J. Numer. Anal.* **8** 382–411

5 Arguments

1: M – INTEGER *Input*

On entry: m, the number of rows in the matrix A. If the problem is specified as type FP or LP, M is not referenced and is assumed to be zero.

If the problem is of type QP, M will usually be n, the number of variables. However, a value of M less than n is appropriate for QP3 or QP4 if A is an upper trapezoidal matrix with m rows. Similarly, M may be used to define the dimension of a leading block of nonzeros in the Hessian matrices of QP1 or QP2, in which case the last $(n - m)$ rows and columns of A are assumed to be zero. In the QP case, m should not be greater than n; if it is, the last $(m - n)$ rows of A are ignored.

If the problem is of type LS1 (the default) or specified as type LS2, LS3 or LS4, M is also the dimension of the array B. Note that all possibilities ($m < n$, $m = n$ and $m > n$) are allowed in this case.

Constraint: M > 0 if the problem is not of type FP or LP.

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: LDC – INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which E04NCF/E04NCA is called.

Constraint: LDC $\geq \max(1, \text{NCLIN})$.

5: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which E04NCF/E04NCA is called.

Constraint: $\mathrm{LDA} \geq \max(1, \mathrm{M})$.

6: $\mathrm{C}(\mathrm{LDC}, *)$ – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array C must be at least N if $\mathrm{NCLIN} > 0$, and at least 1 otherwise.

On entry: the ith row of C must contain the coefficients of the ith general constraint, for $i = 1, 2, \ldots, \mathrm{NCLIN}$.

If $\mathrm{NCLIN} = 0$, C is not referenced.

7: $\mathrm{BL}(\mathrm{N} + \mathrm{NCLIN})$ – REAL (KIND=nag_wp) array *Input*

8: $\mathrm{BU}(\mathrm{N} + \mathrm{NCLIN})$ – REAL (KIND=nag_wp) 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, and the next n_L elements must contain the bounds for the general linear constraints (if any). To specify a nonexistent lower bound (i.e., $l_j = -\infty$), set $\mathrm{BL}(j) \leq -bigbnd$, and to specify a nonexistent upper bound (i.e., $u_j = +\infty$), set $\mathrm{BU}(j) \geq bigbnd$; the default value of $bigbnd$ is 10^{20}, but this may be changed by the optional parameter **Infinite Bound Size**. To specify the jth constraint as an equality, set $\mathrm{BU}(j) = \mathrm{BL}(j) = \beta$, say, where $|\beta| < bigbnd$.

Constraints:

$\mathrm{BL}(j) \leq \mathrm{BU}(j)$, for $j = 1, 2, \ldots, \mathrm{N} + \mathrm{NCLIN}$;
if $\mathrm{BL}(j) = \mathrm{BU}(j) = \beta$, $|\beta| < bigbnd$.

9: $\mathrm{CVEC}(*)$ – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array CVEC must be at least N if the problem is of type LP, QP2, QP4, LS2 or LS4, and at least 1 otherwise.

On entry: the coefficients of the explicit linear term of the objective function.

If the problem is of type FP, QP1, QP3, LS1 (the default) or LS3, CVEC is not referenced.

10: $\mathrm{ISTATE}(\mathrm{N} + \mathrm{NCLIN})$ – INTEGER array *Input/Output*

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

If the optional parameter **Warm Start** has been chosen, ISTATE specifies the desired status of the constraints at the start of the feasibility phase. More precisely, the first n elements of ISTATE refer to the upper and lower bounds on the variables, and the next n_L elements refer to the general linear constraints (if any). Possible values for $\mathrm{ISTATE}(j)$ are as follows:

ISTATE(*j*)	**Meaning**
0	The constraint should *not* be in the initial working set.
1	The constraint should be in the initial working set at its lower bound.
2	The constraint should be in the initial working set at its upper bound.
3	The constraint should be in the initial working set as an equality. This value must not be specified unless $\mathrm{BL}(j) = \mathrm{BU}(j)$.

The values -2, -1 and 4 are also acceptable but will be reset to zero by the routine. If E04NCF/E04NCA has been called previously with the same values of N and NCLIN, ISTATE already contains satisfactory information. (See also the description of the optional parameter **Warm**

Start.) The routine also adjusts (if necessary) the values supplied in X to be consistent with ISTATE.

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

On exit: the status of the constraints in the working set at the point returned in X. The significance of each possible value of ISTATE(j) is as follows:

ISTATE(j)	**Meaning**
-2	The constraint violates its lower bound by more than the feasibility tolerance.
-1	The constraint violates its upper bound by more than the feasibility tolerance.
0	The constraint is satisfied to within the feasibility tolerance, but is not in the working set.
1	This inequality constraint is included in the working set at its lower bound.
2	This inequality constraint is included in the working set at its upper bound.
3	The constraint is included in the working set as an equality. This value of ISTATE can occur only when $\text{BL}(j) = \text{BU}(j)$.
4	This corresponds to optimality being declared with X(j) being temporarily fixed at its current value.

11: KX(N) – INTEGER array *Input/Output*

On entry: need not be initialized for problems of type FP, LP, QP1, QP2, LS1 (the default) or LS2.

For problems QP3, QP4, LS3 or LS4, KX must specify the order of the columns of the matrix A with respect to the ordering of X. Thus if column j of A is the column associated with the variable x_i then $\text{KX}(j) = i$.

Constraints:

$1 \leq \text{KX}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{N}$;
if $i \neq j$, $\text{KX}(i) \neq \text{KX}(j)$.

On exit: defines the order of the columns of A with respect to the ordering of X, as described above.

12: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: an initial estimate of the solution.

Note: that it may be best to avoid the choice $\text{X} = 0.0$.

On exit: the point at which E04NCF/E04NCA terminated. If IFAIL $= 0$, 1 or 4, X contains an estimate of the solution.

13: A(LDA, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array A must be at least N if the problem is of type QP1, QP2, QP3, QP4, LS1 (the default), LS2, LS3 or LS4, and at least 1 otherwise.

On entry: the array A must contain the matrix A as specified in Table 1 (see Section 3).

If the problem is of type QP1 or QP2, the first m rows and columns of A must contain the leading m by m rows and columns of the symmetric Hessian matrix. Only the diagonal and upper triangular elements of the leading m rows and columns of A are referenced. The remaining elements are assumed to be zero and need not be assigned.

For problems QP3, QP4, LS3 or LS4, the first m rows of A must contain an m by n upper trapezoidal factor of either the Hessian matrix or the least squares matrix, ordered according to the KX array. The factor need not be of full rank, i.e., some of the diagonals may be zero. However, as a general rule, the larger the dimension of the leading nonsingular sub-matrix of A, the fewer iterations will be required. Elements outside the upper triangular part of the first m rows of A are assumed to be zero and need not be assigned.

If a constrained least squares problem contains a very large number of observations, storage limitations may prevent storage of the entire least squares matrix. In such cases, you should transform the original A into a triangular matrix before the call to E04NCF/E04NCA and solve the problem as type LS3 or LS4.

On exit: if **Hessian** $=$ NO and the problem is of type LS or QP, A contains the upper triangular Cholesky factor R of (8) (see Section 11.3), with columns ordered as indicated by KX.

If **Hessian** $=$ YES and the problem is of type LS or QP, A contains the upper triangular Cholesky factor R of the Hessian matrix H, with columns ordered as indicated by KX. In either case R may be used to obtain the variance-covariance matrix or to recover the upper triangular factor of the original least squares matrix.

If the problem is of type FP or LP, A is not referenced.

14: B($*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array B must be at least M if the problem is of type LS1 (the default), LS2, LS3 or LS4, and at least 1 otherwise.

On entry: the m elements of the vector of observations.

On exit: the transformed residual vector of equation (10) (see Section 11.3).

If the problem is of type FP, LP, QP1, QP2, QP3 or QP4, B is not referenced.

15: ITER – INTEGER *Output*

On exit: the total number of iterations performed.

16: OBJ – REAL (KIND=nag_wp) *Output*

On exit: the value of the objective function at x if x is feasible, or the sum of infeasibiliites at x otherwise. If the problem is of type FP and x is feasible, OBJ is set to zero.

17: CLAMDA(N $+$ NCLIN) – REAL (KIND=nag_wp) array *Output*

On exit: the values of the Lagrange multipliers for each constraint with respect to the current working set. The first n elements contain the multipliers for the bound constraints on the variables, and the next n_L elements contain the multipliers for the general linear constraints (if any). If ISTATE$(j) = 0$ (i.e., constraint j is not in the working set), CLAMDA(j) is zero. If x is optimal, CLAMDA(j) should be non-negative if ISTATE$(j) = 1$, non-positive if ISTATE$(j) = 2$ and zero if ISTATE$(j) = 4$.

18: IWORK(LIWORK) – INTEGER array *Workspace*
19: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which E04NCF/E04NCA is called.

Constraint: LIWORK $\geq$ N.

20: WORK(LWORK) – REAL (KIND=nag_wp) array *Workspace*
21: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which E04NCF/E04NCA is called.

Constraints:

if the problem is of type FP,

if $\mathrm{NCLIN} = 0$, $\mathrm{LWORK} \geq 6 \times \mathrm{N}$;
if $\mathrm{NCLIN} \geq \mathrm{N}$, $\mathrm{LWORK} \geq 2 \times \mathrm{N}^2 + 6 \times \mathrm{N} + 6 \times \mathrm{NCLIN}$;
otherwise $\mathrm{LWORK} \geq 2 \times (\mathrm{NCLIN} + 1)^2 + 6 \times \mathrm{N} + 6 \times \mathrm{NCLIN}$.;

if the problem is of type LP,

if $\mathrm{NCLIN} = 0$, $\mathrm{LWORK} \geq 7 \times \mathrm{N}$;
if $\mathrm{NCLIN} \geq \mathrm{N}$, $\mathrm{LWORK} \geq 2 \times \mathrm{N}^2 + 7 \times \mathrm{N} + 6 \times \mathrm{NCLIN}$;
otherwise $\mathrm{LWORK} \geq 2 \times (\mathrm{NCLIN} + 1)^2 + 7 \times \mathrm{N} + 6 \times \mathrm{NCLIN}$.;

if problems QP1, QP3, LS1 (the default) and LS3,

if $\mathrm{NCLIN} > 0$, $\mathrm{LWORK} \geq 2 \times \mathrm{N}^2 + 9 \times \mathrm{N} + 6 \times \mathrm{NCLIN}$;
if $\mathrm{NCLIN} = 0$, $\mathrm{LWORK} \geq 9 \times \mathrm{N}$.;

if problems QP2, QP4, LS2 and LS4,

if $\mathrm{NCLIN} > 0$, $\mathrm{LWORK} \geq 2 \times \mathrm{N}^2 + 10 \times \mathrm{N} + 6 \times \mathrm{NCLIN}$;
if $\mathrm{NCLIN} = 0$, $\mathrm{LWORK} \geq 10 \times \mathrm{N}$..

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, you may prefer to obtain appropriate values from the output of a preliminary run with LIWORK and LWORK set to 1. (E04NCF/E04NCA will then terminate with $\mathrm{IFAIL} = 6$.)

22: IFAIL – INTEGER *Input/Output*

Note: *for E04NCA, IFAIL does not occur in this position in the argument list. See the additional arguments described below.*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if $\mathrm{IFAIL} \neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: $\mathrm{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

E04NCF/E04NCA returns with $\mathrm{IFAIL} = 0$ if x is a strong local minimizer, i.e., the projected gradient (`Norm Gz`; see Section 9.2) is negligible, the Lagrange multipliers (`Lagr Mult`; see Section 11.2) are optimal and R_Z (see Section 11.3) is nonsingular.

Note: *the following are additional arguments for specific use with E04NCA. Users of E04NCF therefore need not read the remainder of this description.*

23: LWSAV(120) – LOGICAL array *Communication Array*
24: IWSAV(610) – INTEGER array *Communication Array*
25: RWSAV(475) – REAL (KIND=nag_wp) array *Communication Array*

The arrays LWSAV, IWSAV and RWSAV **must not** be altered between calls to any of the routines E04NCA, E04NDA or E04NEA.

26: IFAIL – INTEGER *Input/Output*

Note: see the argument description for IFAIL above.

6 Error Indicators and Warnings

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

Note: E04NCF/E04NCA may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= 1$

X is a weak local minimum, (i.e., the projected gradient is negligible, the Lagrange multipliers are optimal, but either R_Z (see Section 11.3) is singular, or there is a small multiplier). This means that x is not unique.

IFAIL $= 2$

The solution appears to be unbounded. This value of IFAIL implies that a step as large as **Infinite Bound Size** (default value $= 10^{20}$) would have to be taken in order to continue the algorithm. This situation can occur only when A is singular, there is an explicit linear term, and at least one variable has no upper or lower bound.

IFAIL $= 3$

No feasible point was found, i.e., it was not possible to satisfy all the constraints to within the feasibility tolerance. In this case, the constraint violations at the final x will reveal a value of the tolerance for which a feasible point will exist – for example, when the feasibility tolerance for each violated constraint exceeds its `Slack` (see Section 9.2) at the final point. The modified problem (with an altered feasibility tolerance) may then be solved using a **Warm Start**. You should check that there are no constraint redundancies. If the data for the constraints are accurate only to the absolute precision σ, you should ensure that the value of the optional parameter **Feasibility Tolerance** (default value $= \sqrt{\epsilon}$, where ϵ is the ***machine precision***) is *greater* than σ. For example, if all elements of C are of order unity and are accurate only to three decimal places, the **Feasibility Tolerance** should be at least 10^{-3}.

IFAIL $= 4$

The limiting number of iterations (determined by the optional parameters **Feasibility Phase Iteration Limit** (default value $= \max(50, 5(n + n_L))$) and **Optimality Phase Iteration Limit** (default value $= \max(50, 5(n + n_L))$)) was reached before normal termination occurred. If the method appears to be making progress (e.g., the objective function is being satisfactorily reduced), either increase the iterations limit and rerun E04NCF/E04NCA or, alternatively, rerun E04NCF/E04NCA using the **Warm Start** facility to specify the initial working set. If the iteration limit is already large, but some of the constraints could be nearly linearly dependent, check the monitoring information (see Section 13) for a repeated pattern of constraints entering and leaving the working set. (Near-dependencies are often indicated by wide variations in size in the diagonal elements of the matrix T (see Section 11.2), which will be printed if **Print Level** ≥ 30 (default value $= 10$). In this case, the algorithm could be cycling (see the comments for IFAIL $= 5$).

IFAIL $= 5$

The algorithm could be cycling, since a total of 50 changes were made to the working set without altering x. You should check the monitoring information (see Section 13) for a repeated pattern of constraint deletions and additions.

If a sequence of constraint changes is being repeated, the iterates are probably cycling. (E04NCF/E04NCA does not contain a method that is guaranteed to avoid cycling; such a method would be combinatorial in nature.) Cycling may occur in two circumstances: at a constrained stationary point where there are some small or zero Lagrange multipliers; or at a point (usually a vertex) where the constraints that are satisfied exactly are nearly linearly dependent. In the latter case, you have the option of identifying the offending dependent constraints and removing them from the problem, or restarting the run with a larger value of the optional parameter **Feasibility Tolerance** (default value $= \sqrt{\epsilon}$, where ϵ is the ***machine precision***). If E04NCF/E04NCA terminates with IFAIL $= 5$, but no suspicious pattern of constraint changes can be observed, it may be worthwhile to restart with the final x (with or without the **Warm Start** option).

Note: that this error exit may also occur if a poor starting point X is supplied (for example, X $= 0.0$). You are advised to try a nonzero starting point.

IFAIL $= 6$

An input argument is invalid.

IFAIL $= 7$

The problem to be solved is of type QP1 or QP2, but the Hessian matrix supplied in A is not positive semidefinite.

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 **Feasibility Tolerance** (default value $= \sqrt{\epsilon}$, where ϵ is the ***machine precision***) 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.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

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

IFAIL $= -399$

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

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

IFAIL $= -999$

Dynamic memory allocation failed.

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

7 Accuracy

E04NCF/E04NCA implements a numerically stable active set strategy and returns solutions that are as accurate as the condition of the problem warrants on the machine.

8 Parallelism and Performance

E04NCF/E04NCA is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

E04NCF/E04NCA makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

This section contains some comments on scaling and a description of the printed output.

9.1 Scaling

Sensible scaling of the problem is likely to reduce the number of iterations required and make the problem less sensitive to perturbations in the data, thus improving the condition of the problem. In the absence of better information it is usually sensible to make the Euclidean lengths of each constraint of comparable magnitude. See the E04 Chapter Introduction and Gill *et al.* (1981) for further information and advice.

9.2 Description of the Printed Output

This section describes the intermediate printout and final printout produced by E04NCF/E04NCA. The intermediate printout is a subset of the monitoring information produced by the routine at every iteration (see Section 13). You can control the level of printed output (see the description of the optional parameter **Print Level**). Note that the intermediate printout and final printout are produced only if **Print Level** ≥ 10 (the default for E04NCF, by default no output is produced by E04NCA).

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

`Itn` is the iteration count.

`Step` is the step taken along the computed search direction. If a constraint is added during the current iteration (i.e., `Jadd` is positive), `Step` will be the step to the nearest constraint. During the optimality phase, the step can be greater than one only if the factor R_Z is singular. (See Section 11.3.)

`Ninf` is the number of violated constraints (infeasibilities). This will be zero during the optimality phase.

`Sinf/Objective` is the value of the current objective function. If x is not feasible, `Sinf` gives a weighted sum of the magnitudes of constraint violations. If x is feasible, `Objective` is the value of the objective function of (1). The output line for the final iteration of the feasibility phase (i.e., the first iteration for which `Ninf` is zero) will give the value of the true objective at the first feasible point.

During the optimality phase the value of the objective function will be nonincreasing. During the feasibility phase the number of constraint infeasibilities will not increase until either a feasible point is found or the optimality of the multipliers implies that no feasible point exists. Once optimal multipliers are obtained the number of infeasibilities can increase, but the sum of infeasibilities will either remain constant or be reduced until the minimum sum of infeasibilities is found.

`Norm Gz` is $\left\|Z_1^{\mathrm{T}} g_{\mathrm{FR}}\right\|$, the Euclidean norm of the reduced gradient with respect to Z_1. During the optimality phase, this norm will be approximately zero after a unit step. (See Sections 11.2 and 11.3.)

The final printout includes a listing of the status of every variable and constraint.

The following describes the printout for each variable. A full stop (.) is printed for any numerical value that is zero.

`Varbl` gives the name (`V`) and index j, for $j = 1, 2, \ldots, n$, of the variable.

`State` gives the state of the variable (`FR` if neither bound is in the working set, `EQ` if a fixed variable, `LL` if on its lower bound, `UL` if on its upper bound, `TF` if temporarily fixed at its current value). If `Value` lies outside the upper or lower bounds by more than the **Feasibility Tolerance**, `State` will be `++` or `--` respectively.

A key is sometimes printed before `State`.

`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 then 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 iteration.

`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 general 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 change in the heading:

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

10 Example

This example minimizes the function $\frac{1}{2}\|b - Ax\|^2$, where

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 3 & 1 & 1 & 1 & -1 & -1 & -3 \\ 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 3 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & -1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 2 & 2 & 3 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 2 & 2 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

subject to the bounds

$$\begin{aligned} 0 &\leq x_1 \leq 2 \\ 0 &\leq x_2 \leq 2 \\ -\infty &\leq x_3 \leq 2 \\ 0 &\leq x_4 \leq 2 \\ 0 &\leq x_5 \leq 2 \\ 0 &\leq x_6 \leq 2 \\ 0 &\leq x_7 \leq 2 \\ 0 &\leq x_8 \leq 2 \\ 0 &\leq x_9 \leq 2 \end{aligned}$$

and to the general constraints

$$\begin{array}{rcccccccccccccccccccccc}
2.0 & \le & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & + & x_8 & + & 4x_9 & \le & \infty \\
-\infty & \le & x_1 & + & 2x_2 & + & 3x_3 & + & 4x_4 & - & 2x_5 & + & x_6 & + & x_7 & + & x_8 & + & x_9 & \le & 2.0 \\
1.0 & \le & x_1 & - & x_2 & + & x_3 & - & x_4 & + & x_5 & + & x_6 & + & x_7 & + & x_8 & + & x_9 & \le & 4.0
\end{array}$$

The initial point, which is infeasible, is

$$x_0 = (1.0, 0.5, 0.3333, 0.25, 0.2, 0.1667, 0.1428, 0.125, 0.1111)^{\mathrm{T}},$$

and $F(x_0) = 9.4746$ (to five figures).

The optimal solution (to five figures) is

$$x_* = (0.0, 0.041526, 0.58718, 0.0, 0.099643, 0.0, 0.04906, 0.0, 0.30565)^{\mathrm{T}},$$

and $F(x^*) = 0.081341$. Four bound constraints and all three general constraints are active at the solution.

The document for E04NDF/E04NDA includes an example program to solve a convex quadratic programming problem, using some of the optional parameters described in Section 12.

10.1 Program Text

the following program illustrates the use of E04NCF. An equivalent program illustrating the use of E04NCA is available with the supplied Library and is also available from the NAG web site.

```
    Program e04ncfe

!     E04NCF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: dgemv, e04ncf, e04nef, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: one = 1.0_nag_wp
      Real (Kind=nag_wp), Parameter    :: zero = 0.0_nag_wp
      Integer, Parameter               :: inc1 = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: obj
      Integer                          :: i, ifail, iter, lda, ldc, liwork,    &
                                          lwork, m, n, nclin, sdc
      Logical                          :: verbose_output
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), b(:), bl(:), bu(:), c(:,:),  &
                                          clamda(:), cvec(:), work(:), x(:)
      Integer, Allocatable             :: istate(:), iwork(:), kx(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max
!     .. Executable Statements ..
      Write (nout,*) 'E04NCF Example Program Results'

!     Skip heading in data file
      Read (nin,*)

      Read (nin,*) m, n, nclin
      liwork = n
      ldc = max(1,nclin)
      lda = max(1,m)

      If (nclin>0) Then
        sdc = n
      Else
        sdc = 1
      End If

!     This particular example problem is of type LS1, so we allocate
!     A(LDA,N), CVEC(1), B(M) and define LWORK as below

      If (nclin>0) Then
        lwork = 2*n**2 + 9*n + 6*nclin
      Else
        lwork = 9*n
      End If

      Allocate (istate(n+nclin),kx(n),iwork(liwork),c(ldc,sdc),bl(n+nclin),    &
        bu(n+nclin),cvec(1),x(n),a(lda,n),b(m),clamda(n+nclin),work(lwork))

      Read (nin,*)(a(i,1:n),i=1,m)
      Read (nin,*) b(1:m)
      Read (nin,*)(c(i,1:sdc),i=1,nclin)
      Read (nin,*) bl(1:(n+nclin))
      Read (nin,*) bu(1:(n+nclin))
      Read (nin,*) x(1:n)

!     Set this to .True. to cause e04nqf to produce intermediate
!     progress output
      verbose_output = .False.
      If (.Not. verbose_output) Then
!       Switch off intermediate output from e04ncf
        Call e04nef('Nolist')
        Call e04nef('Print level = 0')
      End If

!     Solve the problem

      ifail = -1
      Call e04ncf(m,n,nclin,ldc,lda,c,bl,bu,cvec,istate,kx,x,a,b,iter,obj,     &
        clamda,iwork,liwork,work,lwork,ifail)

      Select Case (ifail)
      Case (0:5,7:)
!       Print variable headers
        Write (nout,99999)

        Do i = 1, n
          Write (nout,99998) i, istate(i), x(i), clamda(i)
        End Do

        If (nclin>0) Then

!         C*x --> work
!         The NAG name equivalent of dgemv is f06paf
          Call dgemv('N',nclin,n,one,c,ldc,x,inc1,zero,work,inc1)

!         Print constraint headers
          Write (nout,99997)

          Do i = 1, nclin
            Write (nout,99996) i, istate(i+n), work(i), clamda(i+n)
          End Do

        End If

        Write (nout,99995) obj
      End Select

99999 Format (/,1X,'Varbl',3X,'Istate',4X,'Value',9X,'Lagr Mult')
99998 Format (1X,'V',2(1X,I3),4X,1P,E14.3,2X,1P,E12.3)
99997 Format (/,1X,'L Con',3X,'Istate',4X,'Value',9X,'Lagr Mult')
99996 Format (1X,'L',2(1X,I3),4X,1P,E14.3,2X,1P,E12.3)
99995 Format (/,1X,'Final objective value = ',1P,E15.3)
    End Program e04ncfe
```

10.2 Program Data

```
E04NCF Example Program Data
  10   9   3                                                    :Values of M, N and NCLIN
  1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0
  1.0   2.0   1.0   1.0   1.0   1.0   2.0   0.0   0.0
  1.0   1.0   3.0   1.0   1.0   1.0  -1.0  -1.0  -3.0
  1.0   1.0   1.0   4.0   1.0   1.0   1.0   1.0   1.0
  1.0   1.0   1.0   3.0   1.0   1.0   1.0   1.0   1.0
  1.0   1.0   2.0   1.0   1.0   0.0   0.0   0.0  -1.0
  1.0   1.0   1.0   1.0   0.0   1.0   1.0   1.0   1.0
  1.0   1.0   1.0   0.0   1.0   1.0   1.0   1.0   1.0
  1.0   1.0   0.0   1.0   1.0   1.0   2.0   2.0   3.0
  1.0   0.0   1.0   1.0   1.0   1.0   0.0   2.0   2.0        :End of matrix A
  1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0  :End of B
  1.0   1.0   1.0   1.0   1.0   1.0   1.0   1.0   4.0
  1.0   2.0   3.0   4.0  -2.0   1.0   1.0   1.0   1.0
  1.0  -1.0   1.0  -1.0   1.0   1.0   1.0   1.0   1.0        :End of matrix C
  0.0       0.0      -1.0E+25   0.0   0.0   0.0   0.0   0.0   0.0
  2.0      -1.0E+25   1.0                                           :End of BL
  2.0       2.0       2.0       2.0   2.0   2.0   2.0   2.0   2.0
  1.0E+25   2.0       4.0                                           :End of BU
  1.0   0.5   0.3333   0.25   0.2   0.1667   0.1428   0.125   0.1111  :End of X
```

10.3 Program Results

```
 E04NCF Example Program Results

 Varbl   Istate    Value         Lagr Mult
 V   1   1         0.000E+00     1.572E-01
 V   2   0         4.153E-02     0.000E+00
 V   3   0         5.872E-01     0.000E+00
 V   4   1         0.000E+00     8.782E-01
 V   5   0         9.964E-02     0.000E+00
 V   6   1         0.000E+00     1.473E-01
 V   7   0         4.906E-02     0.000E+00
 V   8   1         0.000E+00     8.603E-01
 V   9   0         3.056E-01     0.000E+00

 L Con   Istate    Value         Lagr Mult
 L   1   1         2.000E+00     3.777E-01
 L   2   2         2.000E+00    -5.791E-02
 L   3   1         1.000E+00     1.075E-01

 Final objective value =       8.134E-02
```

Note: *the remainder of this document is intended for more advanced users. Section 11 contains a detailed description of the algorithm which may be needed in order to understand Sections 12 and 13. Section 12 describes the optional parameters which may be set by calls to E04NDF/E04NDA and/or E04NEF/E04NEA. Section 13 describes the quantities which can be requested to monitor the course of the computation.*

11 Algorithmic Details

This section contains a detailed description of the method used by E04NCF/E04NCA.

11.1 Overview

E04NCF/E04NCA is essentially identical to the subroutine LSSOL described in Gill *et al.* (1986). It is based on a two-phase (primal) quadratic programming method with features to exploit the convexity of the objective function due to Gill *et al.* (1984). (In the full-rank case, the method is related to that of Stoer (1971).) E04NCF/E04NCA has two phases: finding an initial feasible point by minimizing the sum of infeasibilities (the *feasibility phase*), and minimizing the quadratic objective function within the feasible region (the *optimality phase*). The two-phase nature of the algorithm is reflected by changing the function being minimized from the sum of infeasibilities to the quadratic objective function. The feasibility phase does *not* perform the standard simplex method (i.e., it does not necessarily find a vertex), except in the LP case when $n_L \leq n$. Once any iterate is feasible, all subsequent iterates remain feasible.

E04NCF/E04NCA has been designed to be efficient when used to solve a *sequence* of related problems – for example, within a sequential quadratic programming method for nonlinearly constrained optimization (e.g., E04UFF/E04UFA or E04WDF). In particular, you may specify an initial working set (the indices of the constraints believed to be satisfied exactly at the solution); see the discussion of the optional parameter **Warm Start**.

In general, an iterative process is required to solve a quadratic program. (For simplicity, we shall always consider a typical iteration and avoid reference to the index of the iteration.) Each new iterate $\bar{x}$ is defined by

$$\bar{x} = x + \alpha p, \tag{2}$$

where the *step length* α is a non-negative scalar, and p is called the *search direction.*

At each point x, a *working set* of constraints is defined to be a linearly independent subset of the constraints that are satisfied 'exactly' (to within the tolerance defined by the optional parameter **Feasibility Tolerance**). The working set is the current prediction of the constraints that hold with equality at a solution of (1). The search direction is constructed so that the constraints in the working set remain *unaltered* for any value of the step length. For a bound constraint in the working set, this property is achieved by setting the corresponding element of the search direction to zero. Thus, the associated variable is *fixed*, and specification of the working set induces a partition of x into *fixed* and *free* variables. During a given iteration, the fixed variables are effectively removed from the problem; since the relevant elements of the search direction are zero, the columns of C corresponding to fixed variables may be ignored.

Let n_{W} denote the number of general constraints in the working set and let n_{FX} denote the number of variables fixed at one of their bounds (n_{W} and n_{FX} are the quantities `Lin` and `Bnd` in the monitoring file output from E04NCF/E04NCA; see Section 13). Similarly, let $n_{\mathrm{FR}} (n_{\mathrm{FR}} = n - n_{\mathrm{FX}})$ denote the number of free variables. At every iteration, *the variables are reordered so that the last* n_{FX} *variables are fixed,* with all other relevant vectors and matrices ordered accordingly. The order of the variables is indicated by the contents of the array KX on exit (see Section 5).

11.2 Definition of Search Direction

Let C_{FR} denote the n_{W} by n_{FR} sub-matrix of general constraints in the working set corresponding to the free variables, and let p_{FR} denote the search direction with respect to the free variables only. The general constraints in the working set will be unaltered by any move along p if

$$C_{\mathrm{FR}} p_{\mathrm{FR}} = 0. \tag{3}$$

In order to compute p_{FR}, the *TQ factorization* of C_{FR} is used:

$$C_{\mathrm{FR}} Q_{\mathrm{FR}} = (0 \quad T) \tag{4}$$

where T is a nonsingular n_{W} by n_{W} reverse-triangular matrix (i.e., $t_{ij} = 0$ if $i + j < n_{\mathrm{W}}$), and the nonsingular n_{FR} by n_{FR} matrix Q_{FR} is the product of orthogonal transformations (see Gill *et al.* (1984)). If the columns of Q_{FR} are partitioned so that

$$Q_{\mathrm{FR}} = (Z \quad Y), \tag{5}$$

where Y is n_{FR} by n_{W}, then the $n_Z (n_Z = n_{\mathrm{FR}} - n_{\mathrm{W}})$ columns of Z form a basis for the null space of C_{FR}. Let n_R be an integer such that $0 \leq n_R \leq n_Z$, and let Z_1 denote a matrix whose n_R columns are a subset of the columns of Z. (The integer n_R is the quantity `Zr` in the monitoring file output from E04NCF/E04NCA. In many cases, Z_1 will include *all* the columns of Z.) The direction p_{FR} will satisfy (3) if

$$p_{\mathrm{FR}} = Z_1 p_Z \tag{6}$$

where p_Z is any n_R-vector.

11.3 Main Iteration

Let Q denote the n by n matrix

$$Q = \begin{pmatrix} Q_{\mathrm{FR}} & \\ & I_{\mathrm{FX}} \end{pmatrix}, \tag{7}$$

where I_{FX} is the identity matrix of order n_{FX}. Let R denote an n by n upper triangular matrix (the *Cholesky factor*) such that

$$R^{\mathrm{T}}R = H_Q \equiv Q^{\mathrm{T}}\tilde{H}Q, \tag{8}$$

where $\tilde{H}$ is the Hessian H with rows and columns permuted so that the free variables are first.

Let the matrix of the first n_Z rows and columns of R be denoted by R_Z. The definition of p_Z in (6) depends on whether or not the matrix R_Z is singular at x. In the nonsingular case, p_Z satisfies the equations

$$R_Z^{\mathrm{T}}R_Zp_Z = -g_Z \tag{9}$$

where g_Z denotes the vector $Z^{\mathrm{T}}g_{\mathrm{FR}}$ and g denotes the objective gradient. (The norm of g_{FR} is the printed quantity `Norm Gf`; see Section 13.) When p_Z is defined by (9), $x+p$ is the minimizer of the objective function subject to the constraints (bounds and general) in the working set treated as equalities. In general, a vector f_Z is available such that $R_Z^{\mathrm{T}}f_Z = -g_Z$, which allows p_Z to be computed from a single back-substitution $R_Zp_Z = f_Z$. For example, when solving problem LS1, f_Z comprises the first n_Z elements of the *transformed residual vector*

$$f = P(b - Ax), \tag{10}$$

which is recurred from one iteration to the next, where P is an orthogonal matrix.

In the singular case, p_Z is defined such that

$$R_Zp_Z = 0 \quad \text{and} \quad g_Z^{\mathrm{T}}p_Z < 0. \tag{11}$$

This vector has the property that the objective function is linear along p and may be reduced by any step of the form $x + \alpha p$, where $\alpha > 0$.

The vector $Z^{\mathrm{T}}g_{\mathrm{FR}}$ is known as the *projected gradient* at x. If the projected gradient is zero, x is a constrained stationary point in the subspace defined by Z. During the feasibility phase, the projected gradient will usually be zero only at a vertex (although it may be zero at non-vertices in the presence of constraint dependencies). During the optimality phase, a zero projected gradient implies that x minimizes the quadratic objective when the constraints in the working set are treated as equalities. At a constrained stationary point, Lagrange multipliers λ_{C} and λ_{B} for the general and bound constraints are defined from the equations

$$C_{\mathrm{FR}}^{\mathrm{T}}\lambda_C = g_{\mathrm{FR}} \text{and } \lambda_B = g_{\mathrm{FX}} - C_{\mathrm{FX}}^{\mathrm{T}}\lambda_C. \tag{12}$$

Given a positive constant δ of the order of the ***machine precision***, the Lagrange multiplier λ_j corresponding to an inequality constraint in the working set is said to be *optimal* if $\lambda_j \leq \delta$ when the associated constraint is at its *upper bound*, or if $\lambda_j \geq -\delta$ when the associated constraint is at its *lower bound*. If a multiplier is nonoptimal, the objective function (either the true objective or the sum of infeasibilities) can be reduced by deleting the corresponding constraint (with index `Jdel`; see Section 13) from the working set.

If optimal multipliers occur during the feasibility phase and the sum of infeasibilities is nonzero, there is no feasible point, and E04NCF/E04NCA will continue until the minimum value of the sum of infeasibilities has been found. At this point, the Lagrange multiplier λ_j corresponding to an inequality constraint in the working set will be such that $-(1+\delta) \leq \lambda_j \leq \delta$ when the associated constraint is at its *upper bound*, and $-\delta \leq \lambda_j \leq (1+\delta)$ when the associated constraint is at its *lower bound*. Lagrange multipliers for equality constraints will satisfy $|\lambda_j| \leq 1 + \delta$.

The choice of step length is based on remaining feasible with respect to the satisfied constraints. If R_Z is nonsingular and $x+p$ is feasible, α will be taken as unity. In this case, the projected gradient at $\bar{x}$

will be zero, and Lagrange multipliers are computed. Otherwise, α is set to α_{M}, the step to the 'nearest' constraint (with index `Jadd`; see Section 13), which is added to the working set at the next iteration.

If A is not input as a triangular matrix, it is overwritten by a triangular matrix R satisfying (8) obtained using the Cholesky factorization in the QP case, or the QR factorization in the LS case. Column interchanges are used in both cases, and an estimate is made of the rank of the triangular factor. Thereafter, the dependent rows of R are eliminated from the problem.

Each change in the working set leads to a simple change to C_{FR}: if the status of a general constraint changes, a *row* of C_{FR} is altered; if a bound constraint enters or leaves the working set, a *column* of C_{FR} changes. Explicit representations are recurred of the matrices T, Q_{FR} and R; and of vectors $Q^{\mathrm{T}}g$, $Q^{\mathrm{T}}c$ and f, which are related by the formulae

$$f = Pb - \begin{pmatrix} R \\ 0 \end{pmatrix} Q^{\mathrm{T}}x, \quad (b \equiv 0 \text{for the } QP \text{ case}),$$

and

$$Q^{\mathrm{T}}g = Q^{\mathrm{T}}c - R^{\mathrm{T}}f.$$

Note that the triangular factor R associated with the Hessian of the original problem is updated during both the optimality *and* the feasibility phases.

The treatment of the singular case depends critically on the following feature of the matrix updating schemes used in E04NCF/E04NCA: if a given factor R_Z is nonsingular, it can become singular during subsequent iterations only when a constraint leaves the working set, in which case only its last diagonal element can become zero. This property implies that a vector satisfying (11) may be found using the single back-substitution $\bar{R}_Z p_Z = e_Z$, where $\bar{R}_Z$ is the matrix R_Z with a unit last diagonal, and e_Z is a vector of all zeros except in the last position. If H is singular, the matrix R (and hence R_Z) may be singular at the start of the optimality phase. However, R_Z will be nonsingular if enough constraints are included in the initial working set. (The matrix with no rows and columns is positive definite by definition, corresponding to the case when C_{FR} contains n_{FR} constraints.) The idea is to include as many general constraints as necessary to ensure a nonsingular R_Z.

At the beginning of each phase, an upper triangular matrix R_1 is determined that is the largest nonsingular leading sub-matrix of R_Z. The use of interchanges during the factorization of A tends to maximize the dimension of R_1. (The rank of R_1 is estimated using the optional parameter **Rank Tolerance**.) Let Z_1 denote the columns of Z corresponding to R_1, and let Z be partitioned as $Z = (Z_1 \quad Z_2)$. A working set for which Z_1 defines the null space can be obtained by including *the rows of* Z_2^{T} as 'artificial constraints'. Minimization of the objective function then proceeds within the subspace defined by Z_1.

The artificially augmented working set is given by

$$\bar{C}_{\mathrm{FR}} = \begin{pmatrix} C_{\mathrm{FR}} \\ Z_2^{\mathrm{T}} \end{pmatrix}, \tag{13}$$

so that p_{FR} will satisfy $C_{\mathrm{FR}}p_{\mathrm{FR}} = 0$ and $Z_2^{\mathrm{T}}p_{\mathrm{FR}} = 0$. By definition of the TQ factorization, $\bar{C}_{\mathrm{FR}}$ *automatically* satisfies the following:

$$\bar{C}_{\mathrm{FR}}Q_{\mathrm{FR}} = \begin{pmatrix} C_{\mathrm{FR}} \\ Z_2^{\mathrm{T}} \end{pmatrix} Q_{\mathrm{FR}} = \begin{pmatrix} C_{\mathrm{FR}} \\ Z_2^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} Z_1 & Z_2 & Y \end{pmatrix} = \begin{pmatrix} 0 & \bar{T} \end{pmatrix},$$

where

$$\bar{T} = \begin{pmatrix} 0 & T \\ I & 0 \end{pmatrix},$$

and hence the TQ factorization of (13) requires no additional work.

The matrix Z_2 need not be kept fixed, since its role is purely to define an appropriate null space; the TQ factorization can therefore be updated in the normal fashion as the iterations proceed. No work is required to 'delete' the artificial constraints associated with Z_2 when $Z_1^{\mathrm{T}}g_{\mathrm{FR}} = 0$, since this simply involves repartitioning Q_{FR}. When deciding which constraint to delete, the 'artificial' multiplier vector associated with the rows of Z_2^{T} is equal to $Z_2^{\mathrm{T}} g_{\mathrm{FR}}$, and the multipliers corresponding to the rows of the 'true' working set are the multipliers that would be obtained if the temporary constraints were not present.

The number of columns in Z_2 and Z_1, the Euclidean norm of $Z_1^{\mathrm{T}} g_{\mathrm{FR}}$, and the condition estimator of R_1 appear in the monitoring file output as `Art`, `Zr`, `Norm Gz` and `Cond Rz` respectively (see Section 13).

Although the algorithm of E04NCF/E04NCA does not perform simplex steps in general, there is one exception: a linear program with fewer general constraints than variables (i.e., $n_L \leq n$). Use of the simplex method in this situation leads to savings in storage. At the starting point, the 'natural' working set (the set of constraints exactly or nearly satisfied at the starting point) is augmented with a suitable number of 'temporary' bounds, each of which has the effect of temporarily fixing a variable at its current value. In subsequent iterations, a temporary bound is treated as a standard constraint until it is deleted from the working set, in which case it is never added again.

One of the most important features of E04NCF/E04NCA is its control of the conditioning of the working set, whose nearness to linear dependence is estimated by the ratio of the largest to smallest diagonals of the TQ factor T (the printed value `Cond T`; see Section 13). In constructing the initial working set, constraints are excluded that would result in a large value of `Cond T`. Thereafter, E04NCF/E04NCA allows constraints to be violated by as much as a user-specified optional parameter **Feasibility Tolerance** in order to provide, whenever possible, a *choice* of constraints to be added to the working set at a given iteration. Let α_{M} denote the maximum step at which $x + \alpha_{\mathrm{M}} p$ does not violate any constraint by more than its feasibility tolerance. All constraints at distance $\alpha (\alpha \leq \alpha_{\mathrm{M}})$ along p from the current point are then viewed as acceptable candidates for inclusion in the working set. The constraint whose normal makes the largest angle with the search direction is added to the working set. In order to ensure that the new iterate satisfies the constraints in the working set as accurately as possible, the step taken is the exact distance to the newly added constraint. As a consequence, negative steps are occasionally permitted, since the current iterate may violate the constraint to be added by as much as the feasibility tolerance.

12 Optional Parameters

Several optional parameters in E04NCF/E04NCA define choices in the problem specification or the algorithm logic. In order to reduce the number of formal arguments of E04NCF/E04NCA these optional parameters have associated *default values* that are appropriate for most problems. Therefore, you need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 12.1.

Cold Start
Crash Tolerance
Defaults
Feasibility Phase Iteration Limit
Feasibility Tolerance
Hessian
Infinite Bound Size
Infinite Step Size
Iteration Limit
Iters
Itns
List
Monitoring File
Nolist
Optimality Phase Iteration Limit

Print Level
Problem Type
Rank Tolerance
Warm Start

Optional parameters may be specified by calling one, or both, of the routines E04NDF/E04NDA and E04NEF/E04NEA before a call to E04NCF/E04NCA.

E04NDF/E04NDA 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 E04NDF (IOPTNS, INFORM)
```

can then be used to read the file on unit IOPTNS. INFORM will be zero on successful exit. E04NDF/E04NDA should be consulted for a full description of this method of supplying optional parameters.

E04NEF/E04NEA can be called to supply options directly, one call being necessary for each optional parameter. For example,

```
CALL E04NEF ('Print Level = 1')
```

E04NEF/E04NEA should be consulted for a full description of this method of supplying optional parameters.

All optional parameters not specified by you are set to their default values. Optional parameters specified by you are unaltered by E04NCF/E04NCA (unless they define invalid values) and so remain in effect for subsequent calls unless altered by you.

12.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords, where the minimum abbreviation of each keyword is underlined (if no characters of an optional qualifier are underlined, the qualifier may be omitted);

a parameter value, where the letters a, i and r denote options that take character, integer and real values respectively;

the default value, where the symbol ϵ is a generic notation for ***machine precision*** (see X02AJF).

Keywords and character values are case and white space insensitive.

Cold Start Default
Warm Start

This option specifies how the initial working set is chosen. With a **Cold Start**, E04NCF/E04NCA chooses the initial working set 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**).

With a **Warm Start**, you must provide a valid definition of every element of the array ISTATE. E04NCF/E04NCA will override your 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 E04NCF/E04NCA 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 E04NCF/E04NCA 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 $c_j^{\mathrm{T}} x \geq l$ will be included in the initial working set if $\left| c_j^{\mathrm{T}} x - l \right| \leq r(1 + |l|)$. 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.

Feasibility Phase Iteration Limit i_1 Default $= \max(50, 5(n + n_L))$
Optimality Phase Iteration Limit i_2 Default $= \max(50, 5(n + n_L))$

The scalars i_1 and i_2 specify the maximum number of iterations allowed in the feasibility and optimality phases. Optional parameter **Optimality Phase Iteration Limit** is equivalent to optional parameter **Iteration Limit**. Setting $i_2 = 0$ and **Print Level** > 0 means that the workspace needed will be computed and printed, but no iterations will be performed. If $i_1 < 0$ or $i_2 < 0$, the default value is used.

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

If $r > \epsilon$, r defines the maximum acceptable *absolute* violation in each constraint at a 'feasible' point. For example, if the variables and the coefficients in the general constaints are of order unity, and the latter are correct to about 6 decimal digits, it would be appropriate to specify r as 10^{-6}. If $0 \leq r < \epsilon$, the default value is used.

Note that a 'feasible solution' is a solution that satisfies the current constraints to within the tolerance r.

Hessian **No** Default $=$ NO

This option controls the contents of the upper triangular matrix R (see the description of A in Section 5). E04NCF/E04NCA works exclusively with the transformed and reordered matrix H_Q (8), and hence extra computation is required to form the Hessian itself. If **Hessian** $=$ NO, A contains the Cholesky factor of the matrix H_Q with columns ordered as indicated by KX (see Section 5). If **Hessian** $=$ YES, A contains the Cholesky factor of the matrix H, with columns ordered as indicated by KX.

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 $+\infty$ (and similarly any lower bound less than or equal to $-bigbnd$ will be regarded as $-\infty$). If $r < 0$, the default value is used.

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

If $r > 0$, r specifies the magnitude of the change in variables that will be considered a step to an unbounded solution. (Note that an unbounded solution can occur only when the Hessian is singular and the objective contains an explicit linear term.) 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 $= \max(50, 5(n + n_L))$
Iters
Itns

See optional parameter **Feasibility Phase Iteration Limit**.

List Default for E04NCF = **List**
Nolist Default for E04NCA = **Nolist**

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

Monitoring File i Default $= -1$

If $i \geq 0$ and **Print Level** ≥ 5, monitoring information produced by E04NCF/E04NCA at every iteration is sent to a file with logical unit number i. If $i < 0$ and/or **Print Level** < 5, no monitoring information is produced.

Print Level i Default for E04NCF $= 10$
Default for E04NCA $= 0$

The value of i controls the amount of printout produced by E04NCF/E04NCA, as indicated below. A detailed description of the printed output is given in Section 9.2 (summary output at each iteration and the final solution) and Section 13 (monitoring information at each iteration).

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 9.2) for each iteration (no printout of the final solution).
≥ 10	The final solution and one line of summary output for each iteration.

The following printout is sent to the logical unit number by the optional parameter **Monitoring File**:

i	**Output**
< 5	No output.
≥ 5	One long line of output ($>$ 80 characters; see Section 13) for each iteration (no printout of the final solution).
≥ 20	At each iteration, the Lagrange multipliers, the variables x, the constraint values Cx and the constraint status.
≥ 30	At each iteration, the diagonal elements of the matrix T associated with the TQ factorization (4) (see Section 11.2) of the working set, and the diagonal elements of the upper triangular matrix R.

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

Problem Type a Default = LS1

This option specifies the type of objective function to be minimized during the optimality phase. The following are the nine optional keywords and the dimensions of the arrays that must be specified in order to define the objective function:

LP	A and B not referenced, CVEC(N);
QP1	A(LDA, N) symmetric, B and CVEC not referenced;
QP2	A(LDA, N) symmetric, B not referenced, CVEC(N);
QP3	A(LDA, N) upper trapezoidal, KX(N), B and CVEC not referenced;
QP4	A(LDA, N) upper trapezoidal, KX(N), B not referenced, CVEC(N);
LS1	A(LDA, N), B(M), CVEC not referenced;
LS2	A(LDA, N), B(M), CVEC(N);

LS3	A(LDA, N) upper trapezoidal, KX(N), B(M), CVEC not referenced;
LS4	A(LDA, N) upper trapezoidal, KX(N), B(M), CVEC(N).

For problems of type FP, the objective function is omitted and A, B and CVEC are not referenced.

The following keywords are also acceptable. The minimum abbreviation of each keyword is underlined.

a	**Option**
<u>Le</u>ast	LS1
<u>Qu</u>adratic	QP2
<u>Li</u>near	LP

In addition, the keywords LS and LSQ are equivalent to the default option LS1, and the keyword QP is equivalent to the option QP2.

If $A = 0$, i.e., the objective function is purely linear, the efficiency of E04NCF/E04NCA may be increased by specifying a as LP.

<u>R</u>ank Tolerance r Default $= 100\epsilon$ or $10\sqrt{\epsilon}$ (see below)

Note that this option does not apply to problems of type FP or LP.

The default value of r depends on the problem type. If A occurs as a least squares matrix, as it does in problem types QP1, LS1 and LS3, then the default value of r is 100ϵ. In all other cases, A is treated as the 'square root' of the Hessian matrix H and r has the default value $10\sqrt{\epsilon}$.

This parameter enables you to control the estimate of the triangular factor R_1 (see Section 11.3). If ρ_i denotes the function $\rho_i = \max\{|R_{11}|, |R_{22}|, \ldots, |R_{ii}|\}$, the rank of R is defined to be smallest index i such that $\left|R_{i+1,i+1}\right| \leq r|\rho_{i+1}|$. If $r \leq 0$, the default value is used.

13 Description of Monitoring Information

This section describes the long line of output (> 80 characters) which forms part of the monitoring information produced by E04NCF/E04NCA. (See also the description of the optional parameters **Monitoring File** and **Print Level**.) You can control the level of printed output.

To aid interpretation of the printed results, the following convention is used for numbering the constraints: indices 1 through n refer to the bounds on the variables, and indices $n + 1$ through $n + n_L$ refer to the general constraints. When the status of a constraint changes, the index of the constraint is printed, along with the designation `L` (lower bound), `U` (upper bound), `E` (equality), `F` (temporarily fixed variable) or `A` (artificial constraint).

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

`Itn`	is the iteration count.
`Jdel`	is the index of the constraint deleted from the working set. If `Jdel` is zero, no constraint was deleted.
`Jadd`	is the index of the constraint added to the working set. If `Jadd` is zero, no constraint was added.
`Step`	is the step taken along the computed search direction. If a constraint is added during the current iteration (i.e., `Jadd` is positive), `Step` will be the step to the nearest constraint. During the optimality phase, the step can be greater than one only if the factor R_Z is singular.
`Ninf`	is the number of violated constraints (infeasibilities). This will be zero during the optimality phase.

`Sinf/Objective`	is the value of the current objective function. If x is not feasible, `Sinf` gives a weighted sum of the magnitudes of constraint violations. If x is feasible, `Objective` is the value of the objective function of (1). The output line for the final iteration of the feasibility phase (i.e., the first iteration for which `Ninf` is zero) will give the value of the true objective at the first feasible point. During the optimality phase the value of the objective function will be nonincreasing. During the feasibility phase the number of constraint infeasibilities will not increase until either a feasible point is found or the optimality of the multipliers implies that no feasible point exists. Once optimal multipliers are obtained the number of infeasibilities can increase, but the sum of infeasibilities will either remain constant or be reduced until the minimum sum of infeasibilities is found.
`Bnd`	is the number of simple bound constraints in the current working set.
`Lin`	is the number of general linear constraints in the current working set.
`Art`	is the number of artificial constraints in the working set, i.e., the number of columns of Z_2 (see Section 11.3).
`Zr`	is the number of columns of Z_1 (see Section 11.2). `Zr` is the dimension of the subspace in which the objective function is currently being minimized. The value of `Zr` is the number of variables minus the number of constraints in the working set; i.e., $\texttt{Zr} = n - (\texttt{Bnd} + \texttt{Lin} + \texttt{Art})$. The value of n_Z, the number of columns of Z (see Section 11.2) can be calculated as $n_Z = n - (\texttt{Bnd} + \texttt{Lin})$. A zero value of n_Z implies that x lies at a vertex of the feasible region.
`Norm Gz`	is $\left\| Z_1^{\mathrm{T}} g_{\mathrm{FR}} \right\|$, the Euclidean norm of the reduced gradient with respect to Z_1. During the optimality phase, this norm will be approximately zero after a unit step.
`Norm Gf`	is the Euclidean norm of the gradient function with respect to the free variables, i. e., variables not currently held at a bound.
`Cond T`	is a lower bound on the condition number of the working set.
`Cond Rz`	is a lower bound on the condition number of the triangular factor R_1 (the first `Zr` rows and columns of the factor R_Z). If the problem is specified to be of type LP or the estimated rank of the data matrix A is zero then `Cond Rz` is not printed.