NAG Library Routine Document

E04NFF/E04NFA

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

E04NFF/E04NFA solves general quadratic programming problems. It is not intended for large sparse problems.

E04NFA is a version of E04NFF that has additional parameters in order to make it safe for use in multithreaded applications (see Section 5). The initialization routine E04WBF **must** have been called before calling E04NFA.

2 Specification

2.1 Specification for E04NFF

```
SUBROUTINE E04NFF (N, NCLIN, A, LDA, BL, BU, CVEC, H, LDH, QPHESS,          &
                   ISTATE, X, ITER, OBJ, AX, CLAMDA, IWORK, LIWORK,         &
                   WORK, LWORK, IFAIL)

INTEGER            N, NCLIN, LDA, LDH, ISTATE(N+NCLIN), ITER,               &
                   IWORK(LIWORK), LIWORK, LWORK, IFAIL
REAL (KIND=nag_wp) A(LDA,*), BL(N+NCLIN), BU(N+NCLIN), CVEC(*),             &
                   H(LDH,*), X(N), OBJ, AX(max(1,NCLIN)),                   &
                   CLAMDA(N+NCLIN), WORK(LWORK)
EXTERNAL           QPHESS
```

2.2 Specification for E04NFA

```
SUBROUTINE E04NFA (N, NCLIN, A, LDA, BL, BU, CVEC, H, LDH, QPHESS,          &
                   ISTATE, X, ITER, OBJ, AX, CLAMDA, IWORK, LIWORK,         &
                   WORK, LWORK, IUSER, RUSER, LWSAV, IWSAV, RWSAV,          &
                   IFAIL)

INTEGER            N, NCLIN, LDA, LDH, ISTATE(N+NCLIN), ITER,               &
                   IWORK(LIWORK), LIWORK, LWORK, IUSER(*), IWSAV(610),      &
                   IFAIL
REAL (KIND=nag_wp) A(LDA,*), BL(N+NCLIN), BU(N+NCLIN), CVEC(*),             &
                   H(LDH,*), X(N), OBJ, AX(max(1,NCLIN)),                   &
                   CLAMDA(N+NCLIN), WORK(LWORK), RUSER(*), RWSAV(475)
LOGICAL            LWSAV(120)
EXTERNAL           QPHESS
```

Before calling E04NFA, or either of the option setting routines E04NGA or E04NHA, 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 = 'E04NFA'. 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 E04NFA, E04NGA, E04NHA and E04WBF.

3 Description

E04NFF/E04NFA is designed to solve a class of quadratic programming problems that are assumed to be stated in the following general form:

$$\underset{x \in R^n}{\text{minimize}} f(x) \quad \text{subject to} \quad l \leq \begin{pmatrix} x \\ Ax \end{pmatrix} \leq u,$$

where A is an m_L by n matrix and $f(x)$ 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 and QP stand for 'feasible point', 'linear programming' and 'quadratic programming' respectively and c is an n-element vector.

Problem type	$\boldsymbol{f(x)}$	**Matrix** $\boldsymbol{H}$
FP	Not applicable	Not applicable
LP	$c^{\mathrm{T}}x$	Not applicable
QP1	$\frac{1}{2}x^{\mathrm{T}}Hx$	symmetric
QP2	$c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Hx$	symmetric
QP3	$\frac{1}{2}x^{\mathrm{T}}H^{\mathrm{T}}Hx$	m by n upper trapezoidal
QP4	$c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}H^{\mathrm{T}}Hx$	m by n upper trapezoidal

There is no restriction on H or $H^{\mathrm{T}}H$ apart from symmetry. If the quadratic function is convex, a global minimum is found; otherwise, a local minimum is found. The default problem type is QP2 and other objective functions are selected by using the optional parameter **Problem Type**. For problems of type FP, the objective function is omitted and the routine attempts to find a feasible point for the set of constraints.

The constraints involving A 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 $\nabla^2 f(x)$ (the *Hessian matrix*) is constant. For QP1 and QP2 (the default), $\nabla^2 f(x) = H$; for QP3 and QP4, $\nabla^2 f(x) = H^{\mathrm{T}}H$; and for the LP case, $\nabla^2 f(x) = 0$. If H is positive semidefinite, it is usually more efficient to use E04NCF/E04NCA. If H is defined as the zero matrix, E04NFF/E04NFA will still attempt to solve the resulting linear programming problem; however, this can be accomplished more efficiently by setting the optional parameter **Problem Type** = LP, or by using E04MFF/E04MFA instead.

You must supply an initial estimate of the solution.

In the QP case, you may supply H either *explicitly* as an m by n matrix, or *implicitly* in a subroutine that computes the product Hx or $H^{\mathrm{T}}Hx$ for any given vector x.

In general, a successful run of E04NFF/E04NFA will indicate one of three situations:

(i) a minimizer has been found;

(ii) the algorithm has terminated at a so-called *dead-point*; or

(iii) the problem has no bounded solution.

If a minimizer is found, and $\nabla^2 f(x)$ is positive definite or positive semidefinite, E04NFF/E04NFA will obtain a global minimizer; otherwise, the solution will be a *local* minimizer (which may or may not be a global minimizer). A dead-point is a point at which the necessary conditions for optimality are satisfied but the sufficient conditions are not. At such a point, a feasible direction of decrease may or may not exist, so that the point is not necessarily a local solution of the problem. Verification of optimality in such instances requires further information, and is in general an NP-hard problem (see Pardalos and Schnitger (1988)). Termination at a dead-point can occur only if $\nabla^2 f(x)$ is not positive definite. If $\nabla^2 f(x)$ is positive semidefinite, the dead-point will be a *weak minimizer* (i.e., with a unique optimal objective value, but an infinite set of optimal x).

The method used by E04NFF/E04NFA (see Section 11) is most efficient when many constraints or bounds are active at the solution.

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 and Murray W (1978) Numerically stable methods for quadratic programming *Math. Programming* **14** 349–372

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, Saunders M A and Wright M H (1989) A practical anti-cycling procedure for linearly constrained optimization *Math. Programming* **45** 437–474

Gill P E, Murray W, Saunders M A and Wright M H (1991) Inertia-controlling methods for general quadratic programming *SIAM Rev.* **33** 1–36

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

Pardalos P M and Schnitger G (1988) Checking local optimality in constrained quadratic programming is NP-hard *Operations Research Letters* **7** 33–35

5 Parameters

1: N – INTEGER *Input*

On entry: n, the number of variables.

Constraint: N > 0.

2: NCLIN – INTEGER *Input*

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

Constraint: NCLIN ≥ 0.

3: A(LDA, $*$) – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array A must be at least N if NCLIN > 0 and at least 1 if NCLIN $= 0$.

On entry: the ith row of A must contain the coefficients of the ith general linear constraint, for $i = 1, 2, \ldots, m_L$.

If NCLIN $= 0$, A is not referenced.

4: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which E04NFF/E04NFA is called.

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

5: BL(N + NCLIN) – REAL (KIND=nag_wp) array *Input*

6: BU(N + 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 m_L elements the bounds for the general linear constraints (if any). To specify a nonexistent lower bound (i.e., $l_j = -\infty$), set BL$(j) \leq -bigbnd$, and to specify a nonexistent 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**. To specify the jth constraint as an *equality*, set BL(j) = BU$(j) = \beta$, say, where $|\beta| < bigbnd$.

Constraints:

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

7: 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 (the default) or QP4, and at least 1 otherwise.

On entry: the coefficients of the explicit linear term of the objective function when the problem is of type LP, QP2 (the default) and QP4.

If the problem is of type FP, QP1, or QP3, CVEC is not referenced.

8: H(LDH, $*$) – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array H must be at least N if it is to be used to store H explicitly, and at least 1 otherwise.

On entry: may be used to store the quadratic term H of the QP objective function if desired. In some cases, you need not use H to store H explicitly (see the specification of subroutine QPHESS). The elements of H are referenced only by subroutine QPHESS. The number of rows of H is denoted by m, whose default value is n. (The optional parameter **Hessian Rows** may be used to specify a value of $m < n$.)

If the default version of QPHESS is used and the problem is of type QP1 or QP2 (the default), the first m rows and columns of H must contain the leading m by m rows and columns of the symmetric Hessian matrix H. Only the diagonal and upper triangular elements of the leading m rows and columns of H are referenced. The remaining elements need not be assigned.

If the default version of QPHESS is used and the problem is of type QP3 or QP4, the first m rows of H must contain an m by n upper trapezoidal factor of the symmetric Hessian matrix $H^{\mathrm{T}}H$. The factor need not be of full rank, i.e., some of the diagonal elements may be zero. However, as a general rule, the larger the dimension of the leading nonsingular sub-matrix of H, the fewer iterations will be required. Elements outside the upper trapezoidal part of the first m rows of H need not be assigned.

If a non-default version of QPHESS is supplied, then in some cases it may be desirable to use a one-dimensional array to transmit data to QPHESS. (This is illustrated in the example program in Section 10 in E04NGF/E04NGA.) H is then declared as an LDH by 1 array, where LDH $\geq$ N $\times$ (N + 1)/2.

In other situations, it may be desirable to compute Hx or $H^{\mathrm{T}}Hx$ without accessing H – for example, if H or $H^{\mathrm{T}}H$ is sparse or has special structure. The parameters H and LDH may then refer to any convenient array.

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

9: LDH – INTEGER *Input*

On entry: the first dimension of the array H as declared in the (sub)program from which E04NFF/E04NFA is called.

Constraints:

if the problem is of type QP1, QP2 (the default), QP3 or QP4, LDH $\geq$ N or at least the value of the optional parameter **Hessian Rows**;
if the problem is of type FP or LP, LDH ≥ 1.

10: QPHESS – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

In general, you need not provide a version of QPHESS, because a 'default' subroutine with name E04NFU/E54NFU is included in the Library. However, the algorithm of E04NFF/E04NFA requires only the product of H or $H^{\mathrm{T}}H$ and a vector x; and in some cases you may obtain increased efficiency by providing a version of QPHESS that avoids the need to define the elements of the matrices H or $H^{\mathrm{T}}H$ explicitly.

QPHESS is not referenced if the problem is of type FP or LP, in which case QPHESS may be the routine E04NFU/E54NFU.

The specification of QPHESS for E04NFF is:

```
SUBROUTINE QPHESS (N, JTHCOL, H, LDH, X, HX)
INTEGER            N, JTHCOL, LDH
REAL (KIND=nag_wp) H(LDH,*), X(N), HX(N)
```

The specification of QPHESS for E04NFA is:

```
SUBROUTINE QPHESS (N, JTHCOL, H, LDH, X, HX, IUSER, RUSER, IWSAV)
INTEGER            N, JTHCOL, LDH, IUSER(*), IWSAV(610)
REAL (KIND=nag_wp) H(LDH,*), X(N), HX(N), RUSER(*)
```

1: N – INTEGER *Input*

On entry: this is the same parameter as supplied to this routine. See the description for the top level parameter N.

2: JTHCOL – INTEGER *Input*

On entry: specifies whether or not the vector x is a column of the identity matrix.

JTHCOL $= j > 0$
The vector x is the jth column of the identity matrix, and hence Hx or $H^{\mathrm{T}}Hx$ is the jth column of H or $H^{\mathrm{T}}H$, respectively. This may in some cases require very little computation and QPHESS may be coded to take advantage of this. However special code is not necessary because x is always stored explicitly in the array X.

JTHCOL $= 0$
x has no special form.

3: H(LDH, $*$) – REAL (KIND=nag_wp) array *Input*

On entry: this is the same parameter as supplied to this routine. See the description for the top level parameter H.

4: LDH – INTEGER *Input*

On entry: this is the same parameter as supplied to this routine. See the description for the top level parameter LDH.

5: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: the vector x.

6: HX(N) – REAL (KIND=nag_wp) array *Output*

On exit: the product Hx if the problem is of type QP1 or QP2 (the default), or the product $H^{\mathrm{T}}Hx$ if the problem is of type QP3 or QP4.

Note: *the following are additional parameters for specific use with E04NFA. Users of E04NFF therefore need not read the remainder of this description*.

7: IUSER(∗) – INTEGER array *User Workspace*

8: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

QPHESS is called with the parameters IUSER and RUSER as supplied to E04NFF/E04NFA. You are free to use the arrays IUSER and RUSER to supply information to QPHESS as an alternative to using COMMON global variables.

9: IWSAV(610) – INTEGER array *Communication Array*

IWSAV contains information that is required by the default routine E54NFU.

QPHESS must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04NFF/E04NFA is called. Parameters denoted as *Input* must **not** be changed by this procedure.

11: ISTATE(N + 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 m_L elements refer to the general linear constraints (if any). Possible values for ISTATE(j) are as follows:

ISTATE(j)	**Meaning**
0	The corresponding 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 BL(j) = BU(j).

The values -2, -1 and 4 are also acceptable but will be reset to zero by the routine. If E04NFF/E04NFA 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: $-2 \leq \mathrm{ISTATE}(j) \leq 4$, for $j = 1, 2, \ldots, \mathrm{N} + \mathrm{NCLIN}$.

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 This constraint is included in the working set as an equality. This value of ISTATE can occur only when $\mathrm{BL}(j) = \mathrm{BU}(j)$.

4 This corresponds to optimality being declared with $\mathrm{X}(j)$ being temporarily fixed at its current value. This value of ISTATE can occur only when $\mathrm{IFAIL} = 1$ on exit.

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

On entry: an initial estimate of the solution.

On exit: the point at which E04NFF/E04NFA terminated. If $\mathrm{IFAIL} = 0$, 1 or 4, X contains an estimate of the solution.

13: ITER – INTEGER *Output*

On exit: the total number of iterations performed.

14: OBJ – REAL (KIND=nag_wp) *Output*

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

15: AX(max(1, NCLIN)) – REAL (KIND=nag_wp) array *Output*

On exit: the final values of the linear constraints Ax.

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

16: 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 m_L elements contain the multipliers for the general linear constraints (if any). If $\mathrm{ISTATE}(j) = 0$ (i.e., constraint j is not in the working set), $\mathrm{CLAMDA}(j)$ is zero. If x is optimal, $\mathrm{CLAMDA}(j)$ should be non-negative if $\mathrm{ISTATE}(j) = 1$, non-positive if $\mathrm{ISTATE}(j) = 2$ and zero if $\mathrm{ISTATE}(j) = 4$.

17: IWORK(LIWORK) – INTEGER array *Workspace*

18: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which E04NFF/E04NFA is called.

Constraint: $\mathrm{LIWORK} \geq 2 \times \mathrm{N} + 3$.

19: WORK(LWORK) – REAL (KIND=nag_wp) array *Workspace*

20: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which E04NFF/E04NFA is called.

Constraints:

if the problem is of type QP2 (the default) or QP4,

if $\mathrm{NCLIN} > 0$, $\mathrm{LWORK} \geq 2 \times \mathrm{N}^2 + 8 \times \mathrm{N} + 5 \times \mathrm{NCLIN}$;
if $\mathrm{NCLIN} = 0$, $\mathrm{LWORK} \geq \mathrm{N}^2 + 8 \times \mathrm{N}$.;

if the problem is of type QP1 or QP3,

if $\mathrm{NCLIN} > 0$, $\mathrm{LWORK} \geq 2 \times \mathrm{N}^2 + 7 \times \mathrm{N} + 5 \times \mathrm{NCLIN}$;
if $\mathrm{NCLIN} = 0$, $\mathrm{LWORK} \geq \mathrm{N}^2 + 7 \times \mathrm{N}$.;

if the problem is of type LP,

if NCLIN $= 0$, LWORK $\geq 8 \times \mathrm{N} + 1$;
if NCLIN $\geq$ N, LWORK $\geq 2 \times \mathrm{N}^2 + 8 \times \mathrm{N} + 5 \times \mathrm{NCLIN}$;
otherwise LWORK $\geq 2 \times (\mathrm{NCLIN} + 1)^2 + 8 \times \mathrm{N} + 5 \times \mathrm{NCLIN}$.;

if the problem is of type FP,

if NCLIN $= 0$, LWORK $\geq 7 \times \mathrm{N} + 1$;
if NCLIN $\geq$ N, LWORK $\geq 2 \times \mathrm{N}^2 + 7 \times \mathrm{N} + 5 \times \mathrm{NCLIN}$;
otherwise LWORK $\geq 2 \times (\mathrm{NCLIN} + 1)^2 + 7 \times \mathrm{N} + 5 \times \mathrm{NCLIN}$..

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. (E04NFF/E04NFA will then terminate with IFAIL $= 6$.)

21: IFAIL – INTEGER *Input/Output*

Note: *for E04NFA, IFAIL does not occur in this position in the parameter list. See the additional parameters described below.*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameters may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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

E04NFF/E04NFA returns with IFAIL $= 0$ if x is a strong local minimizer, i.e., the reduced gradient (`Norm Gz`; see Section 9.2) is negligible, the Lagrange multipliers (`Lagr Mult`; see Section 9.2) are optimal and H_R (the reduced Hessian of $f(x)$; see Section 11.2) is positive semidefinite.

Note: *the following are additional parameters for specific use with E04NFA. Users of E04NFF therefore need not read the remainder of this description*.

22: IUSER$(*)$ – INTEGER array *User Workspace*
23: RUSER$(*)$ – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by E04NFF/E04NFA, but are passed directly to QPHESS and may be used to pass information to this routine as an alternative to using COMMON global variables.

24: LWSAV(120) – LOGICAL array *Communication Array*
25: IWSAV(610) – INTEGER array *Communication Array*
26: 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 E04NFA, E04NGA or E04NHA.

27: IFAIL – INTEGER *Input/Output*

Note: see the parameter 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: E04NFF/E04NFA may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= 1$

The iterations were terminated at a dead-point. The necessary conditions for optimality are satisfied but the sufficient conditions are not. (The reduced gradient is negligible, the Lagrange multipliers are optimal, but H_R is singular or there are some very small multipliers.) If $\nabla^2 f(x)$ is not positive definite, x is not necessarily a local solution of the problem and verification of optimality requires further information. If $\nabla^2 f(x)$ is positive semidefinite or the problem is of type LP, x gives the global minimum value of the objective function, but the final x is not unique.

IFAIL $= 2$

The solution appears to be unbounded, i.e., the objective function is not bounded below in the feasible region. This value of IFAIL occurs if a step larger than **Infinite Step Size** (default value $= 10^{20}$) would have to be taken in order to continue the algorithm, or the next step would result in an element of x having magnitude larger than **Infinite Bound Size** (default value $= 10^{20}$).

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 A 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 was reached before normal termination occurred.

The values of the optional parameters **Feasibility Phase Iteration Limit** (default value $= \max(50, 5(n + m_L))$) and **Optimality Phase Iteration Limit** (default value $= \max(50, 5(n + m_L))$) may be too small. If the method appears to be making progress (e.g., the objective function is being satisfactorily reduced), either increase the iterations limit and rerun E04NFF/E04NFA or, alternatively, rerun E04NFF/E04NFA using the **Warm Start** facility to specify the initial working set.

IFAIL $= 5$

The reduced Hessian exceeds its assigned dimension. The algorithm needed to expand the reduced Hessian when it was already at its maximum dimension, as specified by the optional parameter **Maximum Degrees of Freedom** (default value $= n$).

The value of the optional parameter **Maximum Degrees of Freedom** is too small. Rerun E04NFF/E04NFA with a larger value (possibly using the **Warm Start** facility to specify the initial working set).

IFAIL $= 6$

An input parameter is invalid.

IFAIL $= 7$

The designated problem type was not FP, LP, QP1, QP2, QP3 or QP4. Rerun E04NFF/E04NFA with the optional parameter **Problem Type** set to one of these values.

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 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

E04NFF/E04NFA 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

E04NFF/E04NFA is not threaded by NAG in any implementation.

E04NFF/E04NFA 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 E04NFF/E04NFA. 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 E04NFF, by default no output is produced by E04NFA).

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, `Step` will be the step to the nearest constraint. When the problem is of type LP, the step can be greater than one during the optimality phase.

`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 $\left\|Z_R^{\mathrm{T}} g_{\mathrm{FR}}\right\|$, the Euclidean norm of the reduced gradient with respect to Z_R. 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 quadratic function $f(x) = c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Hx$, where

$$c = (-0.02, -0.2, -0.2, -0.2, -0.2, 0.04, 0.04)^{\mathrm{T}}$$

$$H = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 & -2 & -2 \end{pmatrix}$$

subject to the bounds

$$\begin{aligned} -0.01 &\leq x_1 \leq 0.01 \\ -0.1 &\leq x_2 \leq 0.15 \\ -0.01 &\leq x_3 \leq 0.03 \\ -0.04 &\leq x_4 \leq 0.02 \\ -0.1 &\leq x_5 \leq 0.05 \\ -0.01 &\leq x_6 \\ -0.01 &\leq x_7 \end{aligned}$$

and to the general constraints

$$\begin{array}{rcrcrcrcrcrcrcr} & & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & = & -0.13 \\ & & 0.15x_1 & + & 0.04x_2 & + & 0.02x_3 & + & 0.04x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.03x_7 & \leq & -0.0049 \\ & & 0.03x_1 & + & 0.05x_2 & + & 0.08x_3 & + & 0.02x_4 & + & 0.06x_5 & + & 0.01x_6 & & & \leq & -0.0064 \\ & & 0.02x_1 & + & 0.04x_2 & + & 0.01x_3 & + & 0.02x_4 & + & 0.02x_5 & & & & & \leq & -0.0037 \\ & & 0.02x_1 & + & 0.03x_2 & & & & & + & 0.01x_5 & & & & & \leq & -0.0012 \\ -0.0992 & \leq & 0.70x_1 & + & 0.75x_2 & + & 0.80x_3 & + & 0.75x_4 & + & 0.80x_5 & + & 0.97x_6 & & & & \\ -0.003 & \leq & 0.02x_1 & + & 0.06x_2 & + & 0.08x_3 & + & 0.12x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.97x_7 & \leq & 0.002 \end{array}$$

The initial point, which is infeasible, is

$$x_0 = (-0.01, -0.03, 0.0, -0.01, -0.1, 0.02, 0.01)^{\mathrm{T}}.$$

The optimal solution (to five figures) is

$$x^* = (-0.01, -0.069865, 0.018259, -0.24261, -0.62006, 0.013805, 0.0040665)^{\mathrm{T}}.$$

One bound constraint and four general constraints are active at the solution.

The document for E04NGF/E04NGA includes an example program to solve the same problem using some of the optional parameters described in Section 12.

10.1 Program Text

the following program illustrates the use of E04NFF. An equivalent program illustrating the use of E04NFA is available with the supplied Library and is also available from the NAG web site.

```
    Program e04nffe

!     E04NFF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: e04nff, e04nfu, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: obj
      Integer                          :: i, ifail, iter, lda, ldh, liwork,    &
                                          lwork, n, nclin, sda
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), ax(:), bl(:), bu(:),         &
                                          clamda(:), cvec(:), h(:,:), work(:), &
                                          x(:)
      Integer, Allocatable             :: istate(:), iwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max
!     .. Executable Statements ..
      Write (nout,*) 'E04NFF Example Program Results'
      Flush (nout)

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

      Read (nin,*) n, nclin
      liwork = 2*n + 3
      lda = max(1,nclin)

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

!     This particular example problem is of type QP2 with H stored explicitly,
!     so we allocate CVEC(N) and H(LDH,N), and define LDH and LWORK as below

      ldh = n

      If (nclin>0) Then
        lwork = 2*n**2 + 8*n + 5*nclin
      Else
        lwork = n**2 + 8*n
      End If

      Allocate (istate(n+nclin),ax(max(1,nclin)),iwork(liwork),h(ldh,n),bl(n+ &
```

```
        nclin),bu(n+nclin),cvec(n),x(n),a(lda,sda),clamda(n+nclin), &
        work(lwork))

      Read (nin,*) cvec(1:n)
      Read (nin,*)(a(i,1:sda),i=1,nclin)
      Read (nin,*) bl(1:(n+nclin))
      Read (nin,*) bu(1:(n+nclin))
      Read (nin,*) x(1:n)
      Read (nin,*)(h(i,1:n),i=1,n)

!     Solve the problem

      ifail = 0
      Call e04nff(n,nclin,a,lda,bl,bu,cvec,h,ldh,e04nfu,istate,x,iter,obj,ax, &
        clamda,iwork,liwork,work,lwork,ifail)

    End Program e04nffe
```

10.2 Program Data

```
E04NFF Example Program Data
  7  7                                              :Values of N and NCLIN
 -0.02  -0.20  -0.20  -0.20  -0.20   0.04   0.04   :End of CVEC
  1.00   1.00   1.00   1.00   1.00   1.00   1.00
  0.15   0.04   0.02   0.04   0.02   0.01   0.03
  0.03   0.05   0.08   0.02   0.06   0.01   0.00
  0.02   0.04   0.01   0.02   0.02   0.00   0.00
  0.02   0.03   0.00   0.00   0.01   0.00   0.00
  0.70   0.75   0.80   0.75   0.80   0.97   0.00
  0.02   0.06   0.08   0.12   0.02   0.01   0.97   :End of matrix A
 -0.01  -0.10     -0.01     -0.04     -0.10     -0.01     -0.01
 -0.13  -1.0D+25  -1.0D+25  -1.0D+25  -1.0D+25  -9.92D-02 -3.0D-03 :End of BL
  0.01   0.15      0.03      0.02      0.05      1.0D+25   1.0D+25
 -0.13  -4.9D-03  -6.4D-03  -3.7D-03  -1.2D-03   1.0D+25   2.0D-03 :End of BU
 -0.01  -0.03   0.00  -0.01  -0.10   0.02   0.01   :End of X
  2.00   0.00   0.00   0.00   0.00   0.00   0.00
  0.00   2.00   0.00   0.00   0.00   0.00   0.00
  0.00   0.00   2.00   2.00   0.00   0.00   0.00
  0.00   0.00   2.00   2.00   0.00   0.00   0.00
  0.00   0.00   0.00   0.00   2.00   0.00   0.00
  0.00   0.00   0.00   0.00   0.00  -2.00  -2.00
  0.00   0.00   0.00   0.00   0.00  -2.00  -2.00   :End of matrix H
```

10.3 Program Results

```
 E04NFF Example Program Results

 *** E04NFF

 Parameters
 ----------

 Problem type...........       QP2

 Linear constraints.....         7       Feasibility tolerance..  1.05E-08
 Variables..............         7       Optimality tolerance...  1.05E-08
 Hessian rows...........         7       Rank tolerance.........  1.11E-14

 Infinite bound size....  1.00E+20       COLD start.............
 Infinite step size.....  1.00E+20       EPS (machine precision)  1.11E-16

 Check frequency........        50       Expand frequency.......         5
 Minimum sum of infeas..        NO       Crash tolerance........  1.00E-02

 Max degrees of freedom.         7       Print level............        10
 Feasibility phase itns.        70       Monitoring file........        -1
 Optimality  phase itns.        70

 Workspace provided is     IWORK(      17),  WORK(     189).
 To solve problem we need  IWORK(      17),  WORK(     189).
```

```
 Itn     Step Ninf Sinf/Objective  Norm Gz
   0  0.0E+00    3   1.038000E-01  0.0E+00
   1  4.1E-02    1   3.000000E-02  0.0E+00
   2  4.2E-02    0   0.000000E+00  0.0E+00
Itn     2 -- Feasible point found.
   2  0.0E+00    0   4.580000E-02  0.0E+00
   3  1.3E-01    0   4.161596E-02  0.0E+00
   4  1.0E+00    0   3.936227E-02  4.2E-17
   5  4.1E-01    0   3.758935E-02  1.2E-02
   6  1.0E+00    0   3.755377E-02  2.4E-17
   7  1.0E+00    0   3.703165E-02  4.2E-17


 Varbl State     Value       Lower Bound   Upper Bound    Lagr Mult   Slack

 V   1    LL  -1.000000E-02 -1.000000E-02  1.000000E-02  0.4700         .
 V   2    FR  -6.986465E-02 -0.100000      0.150000            .      3.0135E-02
 V   3    FR   1.825915E-02 -1.000000E-02  3.000000E-02        .      1.1741E-02
 V   4    FR  -2.426081E-02 -4.000000E-02  2.000000E-02        .      1.5739E-02
 V   5    FR  -6.200564E-02 -0.100000      5.000000E-02        .      3.7994E-02
 V   6    FR   1.380544E-02 -1.000000E-02      None            .      2.3805E-02
 V   7    FR   4.066496E-03 -1.000000E-02      None            .      1.4066E-02


 L Con State     Value       Lower Bound   Upper Bound    Lagr Mult   Slack

 L   1    EQ  -0.130000     -0.130000     -0.130000      -1.908         .
 L   2    FR  -5.879898E-03      None     -4.900000E-03        .      9.7990E-04
 L   3    UL  -6.400000E-03      None     -6.400000E-03  -0.3144      8.6736E-19
 L   4    FR  -4.537323E-03      None     -3.700000E-03        .      8.3732E-04
 L   5    FR  -2.915996E-03      None     -1.200000E-03        .      1.7160E-03
 L   6    LL  -9.920000E-02 -9.920000E-02      None        1.955      1.3878E-17
 L   7    LL  -3.000000E-03 -3.000000E-03  2.000000E-03    1.972      8.6736E-19

 Exit E04NFF - Optimal QP solution.

 Final QP objective value =   0.3703165E-01

 Exit from QP problem after     7 iterations.
```

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 E04NGF/E04NGA and/or E04NHF/E04NHA. 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 E04NFF/E04NFA.

11.1 Overview

E04NFF/E04NFA is based on an inertia-controlling method that maintains a Cholesky factorization of the reduced Hessian (see below). The method is based on that of Gill and Murray (1978), and is described in detail by Gill *et al.* (1991). Here we briefly summarise the main features of the method. Where possible, explicit reference is made to the names of variables that are parameters of E04NFF/E04NFA or appear in the printed output. E04NFF/E04NFA has two phases:

(i) finding an initial feasible point by minimizing the sum of infeasibilities (the *feasibility phase*), and

(ii) minimizing the quadratic objective function within the feasible region (the *optimality phase*).

The computations in both phases are performed by the same subroutines. 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 $m_L \leq n$. Once any iterate is feasible, all subsequent iterates remain feasible.

E04NFF/E04NFA 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

$$\bar{x} = x + \alpha p \tag{1}$$

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 the solution of a linearly constrained QP problem. 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 A corresponding to fixed variables may be ignored.

Let m_{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 (m_{W} and n_{FX} are the quantities `Lin` and `Bnd` in the monitoring file output from E04NFF/E04NFA; see Section 13). Similarly, let n_{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.

11.2 Definition of Search Direction

Let A_{FR} denote the m_{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

$$A_{\mathrm{FR}} p_{\mathrm{FR}} = 0. \tag{2}$$

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

$$A_{\mathrm{FR}} Q_{\mathrm{FR}} = (0 \quad T), \tag{3}$$

where T is a nonsingular m_{W} by m_{W} upper triangular matrix (i.e., $t_{ij} = 0$ if $i > j$), 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

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

where Y is n_{FR} by m_{W}, then the n_Z ($n_Z = n_{\mathrm{FR}} - m_{\mathrm{W}}$) columns of Z form a basis for the null space of A_{FR}. Let n_R be an integer such that $0 \leq n_R \leq n_Z$, and let Z_R 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 output from E04NFF/E04NFA. In many cases, Z_R will include *all* the columns of Z.) The direction p_{FR} will satisfy (2) if

$$p_{\mathrm{FR}} = Z_R p_R, \tag{4}$$

where p_R is any n_R-vector.

Let Q denote the n by n matrix

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

where I_{FX} is the identity matrix of order n_{FX}. Let H_Q and g_Q denote the n by n *transformed Hessian* and *transformed gradient*

$$H_Q = Q^{\mathrm{T}} H Q \quad \text{and} \quad g_Q = Q^{\mathrm{T}}(c + Hx)$$

and let the matrix of first n_R rows and columns of H_Q be denoted by H_R and the vector of the first n_R elements of g_Q be denoted by g_R. The quantities H_R and g_R are known as the *reduced Hessian* and *reduced gradient* of $f(x)$, respectively. Roughly speaking, g_R and H_R describe the first and second derivatives of an *unconstrained* problem for the calculation of p_R.

At each iteration, a triangular factorization of H_R is available. If H_R is positive definite, $H_R = R^{\mathrm{T}} R$, where R is the upper triangular Cholesky factor of H_R. If H_R is not positive definite, $H_R = R^{\mathrm{T}} D R$, where $D = \mathrm{diag}(1, 1, \ldots, 1, \mu)$, with $\mu \leq 0$.

The computation is arranged so that the reduced-gradient vector is a multiple of e_R, a vector of all zeros except in the last (i.e., n_Rth) position. This allows the vector p_R in (4) to be computed from a single back-substitution

$$R p_R = \gamma e_R \tag{5}$$

where γ is a scalar that depends on whether or not the reduced Hessian is positive definite at x. In the positive definite case, $x + p$ is the minimizer of the objective function subject to the constraints (bounds and general) in the working set treated as equalities. If H_R is not positive definite p_R satisfies the conditions

$$p_R^{\mathrm{T}} H_R p_R < 0 \quad \text{and} \quad g_R^{\mathrm{T}} p_R \leq 0,$$

which allow the objective function to be reduced by any positive step of the form $x + \alpha p$.

11.3 Main Iteration

If the reduced gradient is zero, x is a constrained stationary point in the subspace defined by Z. During the feasibility phase, the reduced 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 reduced 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

$$A_{\mathrm{FR}}^{\mathrm{T}} \lambda_C = g_{\mathrm{FR}} \quad \text{and} \quad \lambda_B = g_{\mathrm{FX}} - A_{\mathrm{FX}}^{\mathrm{T}} \lambda_C. \tag{6}$$

Given a positive constant δ of the order of the ***machine precision***, a 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 you can force E04NFF/E04NFA to continue until the minimum value of the sum of infeasibilities has been found; see the discussion of the optional parameter **Minimum Sum of Infeasibilities**. At such a 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 $\left|\lambda_j\right| \leq 1 + \delta$.

If the reduced gradient is not zero, Lagrange multipliers need not be computed and the nonzero elements of the search direction p are given by $Z_R p_R$ (see (4) and (5)). The choice of step length is influenced by the need to maintain feasibility with respect to the satisfied constraints. If H_R is positive definite and $x + p$ is feasible, α will be taken as unity. In this case, the reduced 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.

Each change in the working set leads to a simple change to A_{FR}: if the status of a general constraint changes, a *row* of A_{FR} is altered; if a bound constraint enters or leaves the working set, a *column* of A_{FR} changes. Explicit representations are recurred of the matrices T, Q_{FR} and R; and of vectors $Q^{\mathrm{T}}g$, and $Q^{\mathrm{T}}c$. The triangular factor R associated with the reduced Hessian is only updated during the optimality phase.

One of the most important features of E04NFF/E04NFA 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 diagonal elements 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`.

E04NFF/E04NFA includes a rigorous procedure that prevents the possibility of cycling at a point where the active constraints are nearly linearly dependent (see Gill *et al.* (1989)). The main feature of the anti-cycling procedure is that the feasibility tolerance is increased slightly at the start of every iteration. This not only allows a positive step to be taken at every iteration, but also provides, whenever possible, a *choice* of constraints to be added to the working set. 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 a distance α ($\alpha \leq \alpha_{\mathrm{M}}$) 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.

11.4 Choosing the Initial Working Set

At the start of the optimality phase, a positive definite H_R can be defined 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 A_{FR} contains n_{FR} constraints.) The idea is to include as many general constraints as necessary to ensure that the reduced Hessian is positive definite.

Let H_Z denote the matrix of the first n_Z rows and columns of the matrix $H_Q = Q^{\mathrm{T}}HQ$ at the beginning of the optimality phase. A partial Cholesky factorization is used to find an upper triangular matrix R that is the factor of the largest positive definite leading sub-matrix of H_Z. The use of interchanges during the factorization of H_Z tends to maximize the dimension of R. (The condition of R may be controlled using the optional parameter **Rank Tolerance**.) Let Z_R denote the columns of Z corresponding to R, and let Z be partitioned as $Z = \begin{pmatrix} Z_R & Z_A \end{pmatrix}$. A working set for which Z_R defines the null space can be obtained by including *the rows of* Z_A^{T} as 'artificial constraints'. Minimization of the objective function then proceeds within the subspace defined by Z_R, as described in Section 11.2.

The artificially augmented working set is given by

$$\bar{A}_{\mathrm{FR}} = \begin{pmatrix} Z_A^{\mathrm{T}} \\ A_{\mathrm{FR}} \end{pmatrix}, \tag{7}$$

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

$$\bar{A}_{\mathrm{FR}}Q_{\mathrm{FR}} = \begin{pmatrix} Z_A^{\mathrm{T}} \\ A_{\mathrm{FR}} \end{pmatrix} Q_{\mathrm{FR}} = \begin{pmatrix} Z_A^{\mathrm{T}} \\ A_{\mathrm{FR}} \end{pmatrix} \begin{pmatrix} Z_R & Z_A & Y \end{pmatrix} = \begin{pmatrix} 0 & \bar{T} \end{pmatrix},$$

where

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

and hence the TQ factorization of (7) is available trivially from T and Q_{FR} without additional expense.

The matrix Z_A is not 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_A when $Z_R^{\mathrm{T}}g_{\mathrm{FR}} = 0$, since this simply involves repartitioning Q_{FR}. The 'artificial' multiplier vector associated with the rows of Z_A^{T} is equal to $Z_A^{\mathrm{T}}g_{\mathrm{FR}}$, and the multipliers corresponding to the rows of the 'true' working set are the multipliers that

would be obtained if the artificial constraints were not present. If an artificial constraint is 'deleted' from the working set, an `A` appears alongside the entry in the `Jdel` column of the monitoring file output (see Section 13).

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

Under some circumstances, a different type of artificial constraint isused when solving a linear program. Although the algorithm of E04NFF/E04NFA does not usually perform simplex steps (in the traditional sense), there is one exception: a linear program with fewer general constraints than variables (i.e., $m_L \leq n$). 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. If a temporary bound is 'deleted' from the working set, an `F` (for 'Fixed') appears alongside the entry in the `Jdel` column of the monitoring file output (see Section 13).

12 Optional Parameters

Several optional parameters in E04NFF/E04NFA define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of E04NFF/E04NFA 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.

Check Frequency
Cold Start
Crash Tolerance
Defaults
Expand Frequency
Feasibility Phase Iteration Limit
Feasibility Tolerance
Hessian Rows
Infinite Bound Size
Infinite Step Size
Iteration Limit
Iters
Itns
List
Maximum Degrees of Freedom
Minimum Sum of Infeasibilities
Monitoring File
Nolist
Optimality Phase Iteration Limit
Optimality Tolerance
Print Level
Problem Type
Rank Tolerance
Warm Start

Optional parameters may be specified by calling one, or both, of the routines E04NGF/E04NGA and E04NHF/E04NHA before a call to E04NFF/E04NFA.

E04NGF/E04NGA 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 = 5
End
```

The call

```
CALL E04NGF (IOPTNS, INFORM)
```

can then be used to read the file on unit IOPTNS. INFORM will be zero on successful exit. E04NGF/E04NGA should be consulted for a full description of this method of supplying optional parameters.

E04NHF/E04NHA can be called to supply options directly, one call being necessary for each optional parameter. For example,

```
CALL E04NHF ('Print Level = 5')
```

E04NHF/E04NHA 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 E04NFF/E04NFA (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.

Check Frequency r Default $= 50$

Every ith iteration, a numerical test is made to see if the current solution x satisfies the constraints in the working set. If the largest residual of the constraints in the working set is judged to be too large, the current working set is refactorized and the variables are recomputed to satisfy the constraints more accurately. If $i \leq 0$, the default value is used.

Cold Start Default
Warm Start

This option specifies how the initial working set is chosen. With a **Cold Start**, E04NFF/E04NFA 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. E04NFF/E04NFA 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 E04NFF/E04NFA 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 E04NFF/E04NFA 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 $a_j^{\mathrm{T}}x \geq l$ will be included in the initial working set if $\left|a_j^{\mathrm{T}}x - l\right| \leq r(1 + |l|)$. 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.

Expand Frequency i Default $= 5$

This option is part of an anti-cycling procedure designed to guarantee progress even on highly degenerate problems.

The strategy is to force a positive step at every iteration, at the expense of violating the constraints by a small amount. Suppose that the value of the optional parameter **Feasibility Tolerance** is δ. Over a period of i iterations, the feasibility tolerance actually used by E04NFF/E04NFA (i.e., the *working* feasibility tolerance) increases from 0.5δ to δ (in steps of $0.5\delta/i$).

At certain stages the following 'resetting procedure' is used to remove constraint infeasibilities. First, all variables whose upper or lower bounds are in the working set are moved exactly onto their bounds. A count is kept of the number of nontrivial adjustments made. If the count is positive, iterative refinement is used to give variables that satisfy the working set to (essentially) ***machine precision***. Finally, the working feasibility tolerance is reinitialized to 0.5δ.

If a problem requires more than i iterations, the resetting procedure is invoked and a new cycle of i iterations is started with i incremented by 10. (The decision to resume the feasibility phase or optimality phase is based on comparing any constraint infeasibilities with δ.)

The resetting procedure is also invoked when E04NFF/E04NFA reaches an apparently optimal, infeasible or unbounded solution, unless this situation has already occurred twice. If any nontrivial adjustments are made, iterations are continued.

If $i \leq 0$, the default value is used. If $i \geq 9999999$, no anti-cycling procedure is invoked.

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

For problems of type FP, the scalar i_1 specifies the maximum number of iterations allowed before temination. Setting $i_1 = 0$ and **Print Level** > 0 means that the workspace needed will be computed and printed, but no iterations will be performed.

For problems of type LP, the maximum number of iterations allowed before temination is taken as $\max(i_1, i_2)$. Setting $i_1 = 0$, $i_2 = 0$ and **Print Level** > 0 means that the workspace needed will be computed and printed, but no iterations will be performed.

For problems of type QP, the scalars i_1 and i_2 specify the maximum number of iterations allowed in the feasibility and optimality phases. **Optimality Phase Iteration Limit** is equivalent to **Iteration Limit**. Setting $i_1 = 0$ 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 $r \geq \epsilon$, 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 constraints 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.

E04NFF/E04NFA attempts to find a feasible solution before optimizing the objective function. If the sum of infeasibilities cannot be reduced to zero, the optional parameter **Minimum Sum of Infeasibilities** can

be used to find the minimum value of the sum. Let `Sinf` be the corresponding sum of infeasibilities. If `Sinf` is quite small, it may be appropriate to raise r by a factor of 10 or 100. Otherwise, some error in the data should be suspected.

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

Hessian Rows i Default $= n$

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

This specifies m, the number of rows of the Hessian matrix H. The default value of m is n, the number of variables of the problem.

If the problem is of type QP then m will usually be n, the number of variables. However, a value of m less than n is appropriate for QP3 or QP4 if H 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 this case the last $n - m$ rows and columns of H 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 H are ignored.

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

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 not positive definite.) If the change in x during an iteration would exceed the value of r then 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 $= \max(50, 5(n + m_L))$
Iters
Itns

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

List Default for E04NFF = **List**
Nolist Default for E04NFA = **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.

Maximum Degrees of Freedom i Default $= n$

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

This places a limit on the storage allocated for the triangular factor R of the reduced Hessian H_R. Ideally, i should be set slightly larger than the value of n_R expected at the solution. It need not be larger than $m_{\mathrm{N}} + 1$, where m_{N} is the number of variables that appear nonlinearly in the quadratic objective function. For many problems it can be much smaller than m_{N}.

For quadratic problems, a minimizer may lie on any number of constraints, so that n_R may vary between 1 and n. The default value of i is therefore the number of variables n. If **Hessian Rows** m is specified, the default value of i is the same number, m.

Minimum Sum of Infeasibilities a Default = NO

If no feasible point exists for the constraints then this option is used to control whether or not E04NFF/E04NFA will calculate a point that minimizes the constraint violations. If **Minimum Sum of Infeasibilities** = NO, E04NFF/E04NFA will terminate as soon as it is evident that no feasible point exists for the constraints. The final point will generally not be the point at which the

sum of infeasibilities is minimized. If **Minimum Sum of Infeasibilities** = YES, E04NFF/E04NFA will continue until the sum of infeasibilities is minimized.

Monitoring File i Default $= -1$

If $i \geq 0$ and **Print Level** ≥ 5, monitoring information produced by E04NFF/E04NFA 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.

Optimality Tolerance r Default $= \epsilon^{0.5}$

If $r \geq \epsilon$, r defines the tolerance used to determine if the bounds and general constraints have the right 'sign' for the solution to be judged to be optimal.

If $0 \leq r < \epsilon$, the default value is used.

Print Level i Default for E04NFF $= 10$
Default for E04NFA $= 0$

The value of i controls the amount of printout produced by E04NFF/E04NFA, 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). If $i < 0$, the default value is used.

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 defined 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 Ax and the constraint status (see ISTATE).
≥ 30	At each iteration: the diagonal elements of the upper triangular matrix T associated with the TQ factorization (3) (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 **Monitoring File** is the same as that defined by X04ABF then the summary output is suppressed.

Problem Type a Default = QP2

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

LP H not referenced, CVEC(N) required;

QP1 H(LDH, $*$) symmetric, CVEC not referenced;

QP2 H(LDH, $*$) symmetric, CVEC(N) required;

QP3 H(LDH, $*$) upper trapezoidal, CVEC not referenced;

QP4 H(LDH, $*$) upper trapezoidal, CVEC(N) required.

For problems of type FP the objective function is omitted and neither H nor CVEC are referenced.

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

a	**Option**
Quadratic	QP2
Linear	LP
Feasible	FP

In addition, the keyword QP is equivalent to the default option QP2.

If $H = 0$ (i.e., the objective function is purely linear), the efficiency of E04NFF/E04NFA may be increased by specifying a as LP.

Rank Tolerance r Default $= 100\epsilon$

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

This optional parameter enables you to control the condition number of the triangular factor R (see Section 11). If ρ_i denotes the function $\rho_i = \max\{|R_{11}|, |R_{22}|, \ldots, |R_{ii}|\}$, the dimension of R is defined to be smallest index i such that $\left|R_{i+1,i+1}\right| \leq \sqrt{r}|\rho_{i+1}|$. 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 E04NFF/E04NFA. (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 $n+m_L$ 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 the **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, `Step` will be the step to the nearest constraint. When the problem is of type LP, the step can be greater than one during the optimality phase.

`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_A (see Section 11.4).

`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 $\left\|Z_R^{\mathrm{T}} g_{\mathrm{FR}}\right\|$, the Euclidean norm of the reduced gradient with respect to Z_R. During the optimality phase, this norm will be approximately zero after a unit step.

`NOpt` is the number of nonoptimal Lagrange multipliers at the current point. `NOpt` is not printed if the current x is infeasible or no multipliers have been calculated. At a minimizer, `NOpt` will be zero.

`Min Lm` is the value of the Lagrange multiplier associated with the deleted constraint. If `Min Lm` is negative, a lower bound constraint has been deleted, if `Min Lm` is positive, an upper bound constraint has been deleted. If no multipliers are calculated during a given iteration `Min Lm` will be zero.

`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 (the Cholesky factor of the current reduced Hessian; see Section 11.2). If the problem is specified to be of type LP then `Cond Rz` is not printed.

`Rzz` is the last diagonal element μ of the matrix D associated with the $R^{\mathrm{T}}DR$ factorization of the reduced Hessian H_R (see Section 11.2). `Rzz` is only printed if H_R is not positive definite (in which case $\mu \neq 1$). If the printed value of `Rzz` is small in absolute value then H_R is approximately singular. A negative value of `Rzz` implies that the objective function has negative curvature on the current working set.