

# NAG Toolbox for MATLAB

## e04nf

### 1 Purpose

e04nf solves general quadratic programming problems. It is not intended for large sparse problems.

### 2 Syntax

```
[istate, x, iter, obj, ax, clamda, user, lwsav, iwsav, rwsav, ifail] =  
e04nf(a, bl, bu, cvec, h, qp Hess, istate, x, lwsav, iwsav, rwsav, 'n',  
n, 'nclin', nclin, 'user', user)
```

Before calling e04nf, or the option setting function e04nh, e04wb **must** be called.

### 3 Description

e04nf 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	$f(x)$	Matrix $H$
FP	Not applicable	Not applicable
LP	$c^T x$	Not applicable
QP1	$\frac{1}{2} x^T H x$	symmetric
QP2	$c^T x + \frac{1}{2} x^T H x$	symmetric
QP3	$\frac{1}{2} x^T H^T H x$	$m$ by $n$ upper trapezoidal
QP4	$c^T x + \frac{1}{2} x^T H^T H x$	$m$ by $n$ upper trapezoidal

Table 1

There is no restriction on  $H$  or  $H^T H$  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 function 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,  $\nabla^2 f(x) = H^T H$ ; and for the LP case,  $\nabla^2 f(x) = 0$ . If  $H$  is positive semi-definite, it is usually more efficient to use e04nc. If  $H$  is defined as the zero matrix, e04nf 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 e04mf 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 (sub)program that computes the product  $Hx$  or  $H^T Hx$  for any given vector  $x$ .

In general, a successful run of e04nf 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 semi-definite, e04nf 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 semi-definite, 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 e04nf (see Section 10) 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 1986a 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 1984b 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

### 5.1 Compulsory Input Parameters

- 1: **a(lda,\*)** – double array

The first dimension of the array **a** must be at least  $\max(1, \mathbf{nclin})$

The second dimension of the array must be at least **n** if  $\mathbf{nclin} > 0$  and at least 1 if  $\mathbf{nclin} = 0$

The  $i$ th row of **a** must contain the coefficients of the  $i$ th general linear constraint, for  $i = 1, 2, \dots, m_L$ .

If  $\mathbf{nclin} = 0$ , **a** is not referenced.

- 2: **bl(n + nclin)** – double array

- 3: **bu(n + nclin)** – double array

**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  $\mathbf{bl}(j) \leq -\mathbf{bigbnd}$ , and to specify a nonexistent upper bound (i.e.,  $u_j = +\infty$ ), set  $\mathbf{bu}(j) \geq \mathbf{bigbnd}$ ; the default value of  $\mathbf{bigbnd}$  is  $10^{20}$ , but this may be changed by the optional parameter **Infinite Bound Size**. To specify the  $j$ th constraint as an *equality*, set  $\mathbf{bl}(j) = \mathbf{bu}(j) = \beta$ , say, where  $|\beta| < \mathbf{bigbnd}$ .

*Constraints:*

$\mathbf{bl}(j) \leq \mathbf{bu}(j)$ , for  $j = 1, 2, \dots, \mathbf{n} + \mathbf{nclin}$ ;  
if  $\mathbf{bl}(j) = \mathbf{bu}(j) = \beta$ ,  $|\beta| < \mathbf{bigbnd}$ .

4: **cvec(\*) – double array**

**Note:** the dimension of the array **cvec** must be at least **n** if  $\mathbf{iwsav}(365) = 2$  or  $\mathbf{iwsav}(365) = 4$  or  $\mathbf{iwsav}(365) = 6$ , and at least 1 otherwise.

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.

5: **h(ldh,sdh) – double array**

**ldh**, the first dimension of the array, must be at least

if the problem is of type QP1, QP2 (the default), QP3 or QP4,  $\mathbf{ldh} \geq \mathbf{n}$  or at least the value of the optional parameter **Hessian Rows**;  
if the problem is of type FP or LP,  $\mathbf{ldh} \geq 1$ .

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 (sub)program (sub)program **qp Hess**). The elements of **h** are referenced only by (sub)program **qp Hess**. 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 (sub)program **qp Hess** 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 (sub)program **qp Hess** 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  $H^T H$ . 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 nondefault version of (sub)program **qp Hess** is supplied, then in some cases it may be desirable to use a one-dimensional array to transmit data to **qp Hess**. (This is illustrated in the example program in Section 9 of the document for [missing entity e04ng](#).) **h** is then declared as a vector with dimension (**ldh**), where  $\mathbf{ldh} \geq \mathbf{n} \times (\mathbf{n} + 1)/2$ .

In other situations, it may be desirable to compute  $Hx$  or  $H^T Hx$  without accessing **h** – for example, if  $H$  or  $H^T H$  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.

6: **qp Hess – string containing name of m-file**

In general, you need not provide a version of **qp Hess**, because a ‘default’ (sub)program with name ‘e54nfu’ is included in the Library. However, the algorithm of e04nf requires only the product of

$H$  or  $H^T H$  and a vector  $x$ ; and in some cases you may obtain increased efficiency by providing a version of **qp Hess** that avoids the need to define the elements of the matrices  $H$  or  $H^T H$  explicitly.

**qp Hess** is not referenced if the problem is of type FP or LP, in which case **qp Hess** may be the function 'e54nfu'.

```
[hx, user, iwsav] = qp Hess(n, jthcol, h, ldh, x, user, iwsav)
```

#### Input Parameters

1: **n** – **int32 scalar**

This is the same parameter as supplied to this function. See the description for the top level parameter **n**.

2: **jthcol** – **int32 scalar**

Specifies whether or not the vector  $x$  is a column of the identity matrix.

**jthcol** =  $j > 0$

The vector  $x$  is the  $j$ th column of the identity matrix, and hence  $Hx$  or  $H^T Hx$  is the  $j$ th column of  $H$  or  $H^T H$ , respectively. This may in some cases require very little computation and **qp Hess** 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,sdh)** – **double array**

**ldh**, the first dimension of the array, must be at least

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**  $\geq$  1.

.

This is the same parameter as supplied to this function. See the description for the top level parameter **h**.

4: **ldh** – **int32 scalar**

This is the same parameter as supplied to this function. See the description for the top level parameter **ldh**.

5: **x(n)** – **double array**

The vector  $x$ .

6: **user** – **Any MATLAB object**

**qp Hess** is called from e04nf with **user** as supplied to e04nf

7: **iwsav(610)** – **int32 array**

**iwsav** contains information that is required by the default function **e54nfu**.

**Output Parameters**

1: **hx(n) – double array**

The product  $Hx$  if the problem is of type QP1 or QP2 (the default), or the product  $H^T Hx$  if the problem is of type QP3 or QP4.

2: **user – Any MATLAB object**

**qphess** is called from e04nf with **user** as supplied to e04nf

3: **iwsav(610) – int32 array**

**iwsav** contains information that is required by the default function **e54nfu**.

7: **istate(n + nclin) – int32 array**

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:

<b>istate(j)</b>	<b>Meaning</b>
0	The corresponding constraint should <i>not</i> 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 <b>bl(j) = bu(j)</b> .

The values  $-2$ ,  $-1$  and  $4$  are also acceptable but will be reset to zero by the function. If e04nf 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 function also adjusts (if necessary) the values supplied in **x** to be consistent with **istate**.

*Constraint:*  $-2 \leq \text{istate}(j) \leq 4$ , for  $j = 1, 2, \dots, n + nclin$ .

8: **x(n) – double array**

An initial estimate of the solution.

9: **lwsav(120) – logical array**

10: **iwsav(610) – int32 array**

11: **rwsav(475) – double array**

The arrays **lwsav**, **iwsav** and **rwsav** **must not** be altered between calls to any of the functions e04wb, e04nf, e04nh.

## 5.2 Optional Input Parameters

1: **n – int32 scalar**

*Default:* The dimension of the array **x**.

$n$ , the number of variables.

*Constraint:*  $n > 0$ .

2: **nclin – int32 scalar**

$m_L$ , the number of general linear constraints.

*Constraint:*  $nclin \geq 0$ .

3: **user** – Any MATLAB object

**user** is not used by e04nf, but is passed to **qphe**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

### 5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldh, sdh, iwork, liwork, work, lwork

### 5.4 Output Parameters

1: **istate**(**n** + **nclin**) – int32 array

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:

<b>istate</b> ( <i>j</i> )	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 <b>istate</b> can occur only when <b>bl</b> ( <i>j</i> ) = <b>bu</b> ( <i>j</i> ).
4	This corresponds to optimality being declared with <b>x</b> ( <i>j</i> ) being temporarily fixed at its current value. This value of <b>istate</b> can occur only when <b>ifail</b> = 1 on exit.

2: **x**(**n**) – double array

The point at which e04nf terminated. If **ifail** = 0, 1 or 3, **x** contains an estimate of the solution.

3: **iter** – int32 scalar

The total number of iterations performed.

4: **obj** – double scalar

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.

5: **ax**(\*) – double array

**Note:** the dimension of the array **ax** must be at least max(1, **nclin**).

The final values of the linear constraints *Ax*.

If **nclin** = 0, **ax** is not referenced.

6: **clamda**(**n** + **nclin**) – double array

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<sub>L</sub>* 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 nonnegative if **istate**(*j*) = 1, nonpositive if **istate**(*j*) = 2 and zero if **istate**(*j*) = 4.

7: **user** – Any MATLAB object

**user** is not used by e04nf, but is passed to **qphe**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

- 8: **lwsav(120)** – logical array  
 9: **iwsav(610)** – int32 array  
 10: **rwsav(475)** – double array

The arrays **lwsav**, **iwsav** and **rwsav** **must not** be altered between calls to any of the functions e04wb, e04nf, e04nh.

- 11: **ifail** – int32 scalar

0 unless the function detects an error (see Section 6).

## 6 Error Indicators and Warnings

**Note:** e04nf may return useful information for one or more of the following detected errors or warnings.

### **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 semi-definite 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 8.2) 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  $\sigma$ , you should ensure that the value of the optional parameter **Feasibility Tolerance** (default value =  $\sqrt{\epsilon}$ , where  $\epsilon$  is the *machine precision*) is *greater* than  $\sigma$ . 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 e04nf or, alternatively, rerun e04nf 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 e04nf 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 e04nf 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  $j$ th constraint, it may be possible to avoid the difficulty by increasing the magnitude of the **Feasibility Tolerance** (default value =  $\sqrt{\epsilon}$ , where  $\epsilon$  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.

## 7 Accuracy

e04nf 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 Further Comments

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

### 8.1 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.

### 8.2 Description of the Printed Output

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  $\|Z_R^T g_{FR}\|$ , 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 10.2 and 10.3.)

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 $j = 1, 2, \dots, n$ , 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 <b>Feasibility Tolerance</b> , State will be ++ or -- respectively.  A key is sometimes printed before State.  A <i>Alternative optimum possible</i> . 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 <i>might</i> 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 <i>Degenerate</i> . The variable is free, but it is equal to (or very close to) one of its bounds.  I <i>Infeasible</i> . The variable is currently violating one of its bounds by more than the <b>Feasibility Tolerance</b> .
Value	is the value of the variable at the final iteration.
Lower Bound	is the lower bound specified for the variable. None indicates that $\mathbf{bl}(j) \leq -bigbnd$ .
Upper Bound	is the upper bound specified for the variable. None indicates that $\mathbf{bu}(j) \geq bigbnd$ .
Lagr Mult	is the Lagrange multiplier for the associated bound. This will be zero if State is FR unless $\mathbf{bl}(j) \leq -bigbnd$ and $\mathbf{bu}(j) \geq bigbnd$ , in which case the entry will be blank. If $x$ is optimal, the multiplier should be nonnegative if State is LL and nonpositive if State is UL.
Slack	is the difference between the variable Value and the nearer of its (finite) bounds $\mathbf{bl}(j)$ and $\mathbf{bu}(j)$ . A blank entry indicates that the associated variable is not bounded (i.e., $\mathbf{bl}(j) \leq -bigbnd$ and $\mathbf{bu}(j) \geq bigbnd$ ).

The meaning of the printout for general constraints is the same as that given above for variables, with ‘variable’ replaced by ‘constraint’,  $\mathbf{bl}(j)$  and  $\mathbf{bu}(j)$  are replaced by  $\mathbf{bl}(n+j)$  and  $\mathbf{bu}(n+j)$  respectively, and with the following change in the heading:

L Con gives the name (L) and index  $j$ , for  $j = 1, 2, \dots, n_L$ , 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.

## 9 Example

e04nf\_qphess.m

```

function [hx, user, iwsav] = qphess(n, jthcol, h, ldh, x, user, iwsav)

    if (iwsav(365) == 3 || iwsav(365) == 4)
        % Problem type QP1 and QP2
        hx = h*x;
    elseif(iwsav(365) == 5 || iwsav(365) == 6)
        hx = transpose(h)*h*x;
    else
        hx = zeros(n,1);
    end

a = [1, 1, 1, 1, 1, 1, 1;
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;
0.02, 0.04, 0.01, 0.02, 0.02, 0, 0;
0.02, 0.03, 0, 0, 0.01, 0, 0;
0.7, 0.75, 0.8, 0.75, 0.8, 0.97, 0;
0.02, 0.06, 0.08, 0.12, 0.02, 0.01, 0.97];
bl = [-0.01; -0.1; -0.01; -0.04; -0.1; -0.01; -0.01; -0.13; -
9.999999999999999e+24; -9.999999999999999e+24; -9.999999999999999e+24; -
9.999999999999999e+24; -0.0992; -0.003];
bu = [0.01; 0.15; 0.03; 0.02; 0.05; 9.999999999999999e+24;
9.999999999999999e+24; -0.13; -0.0049000000000000001; -0.0064; -0.0037; -
0.0012; 9.999999999999999e+24; 0.002];
cvec = [-0.02; -0.2; -0.2; -0.2; -0.2; 0.04; 0.04];
h = [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];
istate = zeros(14, 1, 'int32');
x = [-0.01; -0.03; 0; -0.01; -0.1; 0.02; 0.01];
[cwsav,lwsav,iwsav,rwsav,ifail] = e04wb('e04nf');
[istateOut, xOut, iter, obj, ax, clamda, user, lwsavOut, iwsavOut,
rwsavOut, ifail] = ...
    e04nf(a, bl, bu, cvec, h, 'e04nf_qphess', istate, x, lwsav, iwsav,
rwsav)

istateOut =
    1
    0
    0
    0
    0
    0
    0
    0
    3
    0
    2
    0
    0
    1
    1
xOut =
   -0.0100
   -0.0699
    0.0183
   -0.0243
   -0.0620
    0.0138
    0.0041
iter =
     7
obj =
    0.0370
ax =

```

```

-0.1300
-0.0059
-0.0064
-0.0045
-0.0029
-0.0992
-0.0030
clamda =
  0.4700
    0
    0
    0
    0
    0
    0
-1.9082
    0
-0.3144
    0
    0
  1.9545
  1.9716
user =
  0
lwsavOut =
  array elided
iwsavOut =
  array elided
rwsavOut =
  array elided
ifail =
    0

```

**Note:** the remainder of this document is intended for more advanced users. Section 10 contains a detailed description of the algorithm which may be needed in order to understand Sections 11 and 12. Section 11 describes the optional parameters which may be set by calls to e04mh. Section 12 describes the quantities which can be requested to monitor the course of the computation.

## 10 Algorithmic Details

This section contains a detailed description of the method used by e04nf.

### 10.1 Overview

e04nf 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 summarize the main features of the method. Where possible, explicit reference is made to the names of variables that are parameters of e04nf or appear in the printed output. e04nf 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 (sub)programs. 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.

e04nf 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., e04wd or e04uf). 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

$$\bar{x} = x + \alpha p \quad (1)$$

where the *step length*  $\alpha$  is a nonnegative 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 e04nf; see Section 12). Similarly, let  $n_{FR}$  ( $n_{FR} = n - n_{FX}$ ) 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.

## 10.2 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

$$A_{FR}p_{FR} = 0. \quad (2)$$

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

$$A_{FR}Q_{FR} = \begin{pmatrix} 0 & T \end{pmatrix}, \quad (3)$$

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.* 1984b). If the columns of  $Q_{FR}$  are partitioned so that

$$Q_{FR} = \begin{pmatrix} Z & Y \end{pmatrix},$$

where  $Y$  is  $n_{FR}$  by  $m_W$ , then the  $n_Z$  ( $n_Z = n_{FR} - m_W$ ) 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 e04nf. In many cases,  $Z_R$  will include *all* the columns of  $Z$ .) The direction  $p_{FR}$  will satisfy (2) if

$$p_{FR} = Z_R p_R, \quad (4)$$

where  $p_R$  is any  $n_R$ -vector.

Let  $Q$  denote the  $n$  by  $n$  matrix

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

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*

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

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,  $H_R = R^T R$ , where  $R$  is the upper triangular Cholesky factor of  $H_R$ . If  $H_R$  is not positive-definite,  $H_R = R^T D R$ , where  $D = \text{diag}(1, 1, \dots, 1, \mu)$ , 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.,  $n_R$ th) position. This allows the vector  $p_R$  in (4) to be computed from a single back-substitution

$$R p_R = \gamma e_R \quad (5)$$

where  $\gamma$  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

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

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

### 10.3 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  $\lambda_C$  and  $\lambda_B$  for the general and bound constraints are defined from the equations

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

Given a positive constant  $\delta$  of the order of the **machine precision**, a Lagrange multiplier  $\lambda_j$  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 12) 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 e04nf 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  $\lambda_j$  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  $|\lambda_j| \leq 1 + \delta$ .

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,  $\alpha$  will be taken as unity. In this case, the reduced gradient at  $\bar{x}$  will be zero, and Lagrange multipliers are computed. Otherwise,  $\alpha$  is set to  $\alpha_M$ , the step to the ‘nearest’ constraint (with index `Jadd`; see Section 12), 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  $Q^T g$ , and  $Q^T c$ . The triangular factor  $R$  associated with the reduced Hessian is only updated during the optimality phase.

One of the most important features of e04nf 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 12). In constructing the initial working set, constraints are excluded that would result in a large value of Cond T.

e04nf 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  $\alpha_M$  denote the maximum step at which  $x + \alpha_M p$  does not violate any constraint by more than its feasibility tolerance. All constraints at a distance  $\alpha$  ( $\alpha \leq \alpha_M$ ) 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.

#### 10.4 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  $H_Q = Q^T H Q$  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  $Z = (Z_R \ Z_A)$ . 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 10.2.

The artificially augmented working set is given by

$$\bar{A}_{FR} = \begin{pmatrix} Z_A^T \\ A_{FR} \end{pmatrix}, \quad (7)$$

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

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

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  $Z_R^T g_{FR} = 0$ , since this simply involves repartitioning  $Q_{FR}$ . The ‘artificial’ multiplier vector associated with the rows of  $Z_A^T$  is equal to  $Z_A^T g_{FR}$ , 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 Jdcl column of the monitoring file output (see Section 12).

The number of columns in  $Z_A$  and  $Z_R$ , the Euclidean norm of  $Z_R^T g_{FR}$ , and the condition estimator of  $R$  appear in the monitoring file output as Art, Zr, Norm Gz and Cond Rz respectively (see Section 12).

Under some circumstances, a different type of artificial constraint is used when solving a linear program. Although the algorithm of e04nf 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 12).

## 11 Optional Parameters

Several optional parameters in e04nf define choices in the problem specification or the algorithm logic. In order to reduce the number of formal parameters of e04nf 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. A complete list of optional parameters and their default values is given in Section 11.1.

Optional parameters may be specified by calling e04nh before a call to e04nf.

e04nh can be called to supply options directly, one call being necessary for each optional parameter. For example,

```
[lwsav, iwsav, rwsav, inform] = e04nh('Print Level = 5', lwsav, iwsav,
rwsav);
```

e04nh 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 e04nf (unless they define invalid values) and so remain in effect for subsequent calls unless altered by you.

### 11.1 Optional Parameter Checklist and Default Values

The following list gives the valid options. For each option, we give the keyword, any essential optional qualifiers and the default value. A description for each option can be found in Section 11.2. The minimum abbreviation of each keyword is underlined. If no characters of an optional qualifier are underlined, the qualifier may be omitted. The letter *a* denotes a phrase (character string) that qualifies an option. The letters *i* and *r* denote integer and double values required with certain options. The number  $\epsilon$  is a generic notation for *machine precision* (see x02aj).

Optional Parameters	Default Values
<u>C</u> heck F <u>r</u> equency	Default = 50
<u>C</u> old S <u>t</u> art	Default
<u>C</u> rash T <u>o</u> lerance	Default = 0.01
<u>D</u> efaults	
<u>E</u> xpand F <u>r</u> equency	Default = 5
<u>F</u> easibility P <u>h</u> ase I <u>te</u> ration L <u>im</u> it	Default = $\max(50, 5(n + m_L))$
<u>F</u> easibility T <u>o</u> lerance	Default = $\sqrt{\epsilon}$
<u>H</u> essian R <u>o</u> ws	Default = $n$
<u>I</u> nfinite B <u>o</u> und S <u>i</u> ze	Default = $10^{20}$
<u>I</u> nfinite S <u>t</u> ep S <u>i</u> ze	Default = $\max(bigbnd, 10^{20})$
<u>I</u> teration L <u>im</u> it	Default = $\max(50, 5(n + m_L))$
<u>I</u> ters	See <u>I</u> teration L <u>im</u> it

<u>Itns</u>	See <b>Iteration Limit</b>
<u>List</u>	Default for e04nf = <b>List</b>
<u>Maximum Degrees of Freedom</u>	Default = $n$
<u>Minimum Sum of Infeasibilities</u>	Default = No
<u>Monitoring File</u>	Default = -1
<u>Nolist</u>	Default for e04nf = <b>Nolist</b> . See <b>List</b> .
<u>Optimality Phase Iteration Limit</u>	Default = $\max(50, 5(n + m_L))$ . See <b>Feasibility Phase Iteration Limit</b> .
<u>Optimality Tolerance</u>	Default = $\epsilon^{0.8}$
<u>Print Level</u>	Default for e04nf = 10
	Default for e04nf = 0
<u>Problem Type</u>	Default = QP2
<u>Rank Tolerance</u>	Default = $100\epsilon$
<u>Warm Start</u>	See <b>Cold Start</b>

## 11.2 Description of the Optional Parameters

**Check Frequency**  $r$  Default = 50

Every  $i$ th 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**, e04nf 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**. e04nf 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 e04nf 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 e04nf selects an initial working set. If  $0 \leq r \leq 1$ , the initial working set will include (if possible) bounds or general inequality constraints that lie within  $r$  of their bounds. In particular, a constraint of the form  $a_j^T x \geq l$  will be included in the initial working set if  $|a_j^T x - l| \leq r(1 + |l|)$ . If  $r < 0$  or  $r > 1$ , the default value is used.

### Defaults

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

**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  $\delta$ . Over a period of  $i$  iterations, the feasibility tolerance actually used by e04nf (i.e., the *working* feasibility tolerance) increases from  $0.5\delta$  to  $\delta$  (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  $0.5\delta$ .

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  $\delta$ .)

The resetting procedure is also invoked when e04nf 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.

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

For problems of type FP, the scalar  $i_1$  specifies the maximum number of iterations allowed before termination. 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 termination 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.

<b>Feasibility Tolerance</b>	$r$	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.

e04nf 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  $\text{Sinf}$  be the corresponding sum of infeasibilities. If  $\text{Sinf}$  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  $r$ .

<b>Hessian Rows</b>	$i$	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 *infbnd* in the definition of the problem constraints. Any upper bound greater than or equal to *infbnd* will be regarded as plus infinity (and similarly any lower bound less than or equal to  $-infbnd$  will be regarded as minus infinity). If  $r < 0$ , the default value is used.

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

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 e04nf = **List**  
**Nolist** Default for e04nf = **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  $m_n + 1$ , 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 e04nf will calculate a point that minimizes the constraint violations. If **Minimum Sum of Infeasibilities** = No, e04nf 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, e04nf will continue until the sum of infeasibilities is minimized.

**Monitoring File**  $i$  Default =  $-1$

If  $i \geq 0$  and **Print Level**  $\geq 5$ , monitoring information produced by e04nf 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 =  $\epsilon^{0.8}$

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 e04nf = 10

Default for e04nf = 0

The value of  $i$  controls the amount of printout produced by e04nf, as indicated below. A detailed description of the printed output is given in Section 8.2 (summary output at each iteration and the final solution) and Section 12 (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 x04ab):

$i$	Output
0	No output.
1	The final solution only.
5	One line of summary output ( < 80 characters; see Section 8.2) for each iteration (no printout of the final solution).
$\geq 10$	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.
$\geq 5$	One long line of output ( > 80 characters; see Section 12) for each iteration (no printout of the final solution).
$\geq 20$	At each iteration: the Lagrange multipliers, the variables $x$ , the constraint values $Ax$ and the constraint status.
$\geq 30$	At each iteration: the diagonal elements of the upper triangular matrix $T$ associated with the $TQ$ factorization (3) (see Section 10.2) of the working set and the diagonal elements of the upper triangular matrix $R$ .

If **Print Level**  $\geq 5$  and the unit number defined by the **Monitoring File** is the same as that defined by x04ab 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
<u>Q</u> uadratic	QP2
<u>L</u> inear	LP
<u>F</u> easible	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 e04nf 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 10). If  $\rho_i$  denotes the function  $\rho_i = \max\{|R_{11}|, |R_{22}|, \dots, |R_{ii}|\}$ , the dimension of  $R$  is defined to be smallest index  $i$  such that  $|R_{i+1,i+1}| \leq \sqrt{r}|\rho_{i+1}|$ . If  $r \leq 0$ , the default value is used.

## 12 Description of Monitoring Information

This section describes the long line of output ( $> 80$  characters) which forms part of the monitoring information produced by e04nf. (See also the description of the optional parameters **Monitoring File** and **Print Level**.) You can control the level of printed output can be controlled by you.

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**  $\geq 5$  and **Monitoring File**  $\geq 0$ , 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 10.4).
Zr	is the number of columns of $Z_1$ (see Section 10.2). 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., $Zr = n - (Bnd + Lin + Art)$ .

The value of  $n_Z$ , the number of columns of  $Z$  (see Section 10.2) can be calculated as  $n_Z = n - (Bnd + Lin)$ . A zero value of  $n_Z$  implies that  $x$  lies at a vertex of the feasible region.

Norm Gz	is $\ Z_R^T g_{FR}\ $ , 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 10.2). If the problem is specified to be of type LP then Cond Rz is not printed.
Rzz	is the last diagonal element $\mu$ of the matrix $D$ associated with the $R^T D R$ factorization of the reduced Hessian $H_R$ (see Section 10.2). 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.

---