USER'S GUIDE FOR QPOPT 1.0:
A FORTRAN PACKAGE FOR
QUADRATIC PROGRAMMING

Philip E. GILL

Department of Mathematics
University of California, San Diego
La Jolla, California 92093-0112

Walter MURRAY and Michael A. SAUNDERS

Systems Optimization Laboratory
Department of Operations Research
Stanford University
Stanford, California 94305-4022

August 1995

Abstract

QPOPT is a set of Fortran subroutines for minimizing a general quadratic function subject to linear constraints and simple upper and lower bounds. QPOPT may also be used for linear programming and for finding a feasible point for a set of linear equalities and inequalities.

If the quadratic function is convex (i.e., the Hessian is positive definite or positive semidefinite), the solution obtained will be a global minimizer. If the quadratic is non-convex (i.e., the Hessian is indefinite), the solution obtained will be a local minimizer or a dead-point.

A two-phase active-set method is used. The first phase minimizes the sum of infeasibilities. The second phase minimizes the quadratic function within the feasible region, using a reduced Hessian to obtain search directions. The method is most efficient when many constraints or bounds are active at the solution.

QPOPT is not intended for large sparse problems, but there is no fixed limit on problem size. The source code is suitable for all scientific machines with a Fortran 77 compiler. This includes mainframes, workstations and PCs, preferably with 1MB or more of main storage.

Keywords: Quadratic programming, linear programming, linear constraints, active-set method, inertia-controlling method, reduced Hessian.

Contents

1. Purpose

QPOPT is a collection of Fortran 77 subroutines for solving the *quadratic programming problem*: minimize a quadratic objective function subject to a set of linear constraints and bounds. The problem is assumed to be in the following form:

$$
\begin{array}{lll}
\text{LCQP} & \underset{x \in \mathbb{R}^n}{\text{minimize}} & q(x) \\[2ex]
& \text{subject to} \quad \ell \le r(x) \le u, & r(x) \equiv \begin{pmatrix} x \\ Ax \end{pmatrix}
\end{array}
$$

The vector x is a set of variables, l and u are bounds on the variables and the product Ax, and A is an $m_L \times n$ matrix (absent if m_L is zero).

1.1. Problem types

The objective function $q(x)$ is specified by an optional input parameter of the form `Problem type = a`. The following choices are allowed:

a	$q(x)$	
FP	*None*	Find a feasible point
LP	$c^T x$	Linear program
QP1	$\frac{1}{2} x^T H x$	H symmetric
QP2	$c^T x + \frac{1}{2} x^T H x$	H symmetric
QP3	$\frac{1}{2} x^T G^T G x$	G $m \times n$ upper-trapezoidal
QP4	$c^T x + \frac{1}{2} x^T G^T G x$	G $m \times n$ upper-trapezoidal

The vector c is an n-vector, where n is a parameter of subroutine `qpopt`, and m is specified by another optional parameter, `Hessian rows`. Problems of type `LP` and `QP` are referred to as *linear programs* and *quadratic programs* respectively. Optional parameters such as `Problem type` are defined in Section 6, along with their default values. The default problem type is `QP2`.

Let the first and second derivatives of $q(x)$ be the gradient $g(x) \equiv \nabla q$ and the Hessian $H \equiv \nabla^2 q$. The defining feature of a quadratic function is that the matrix H is constant. There is no restriction on H apart from symmetry. For problems `FP` and `LP`, $H = 0$. For `QP1` and `QP2`, H is a given symmetric matrix. For `QP3` and `QP4`, $H = G^T G$, where the matrix G is given. (When $H = G^T G$ it may be more reliable to use LSSOL [GHM$^+$86], but QPOPT will be more efficient if many constraints or bounds are active at the solution.) If H happens to be zero for any of the `QP` options, QPOPT will solve the resulting linear program; however, it is more efficient to set `Problem type = LP`.

1.2. Bounds

Note that upper and lower bounds are specified for all variables and constraints. This form allows full generality in specifying various types of constraint. In particular, the jth constraint may be defined as an *equality* by setting $\ell_j = u_j$. If certain bounds are not present, the associated elements of ℓ or u may be set to special values that are treated as $-\infty$ or $+\infty$.
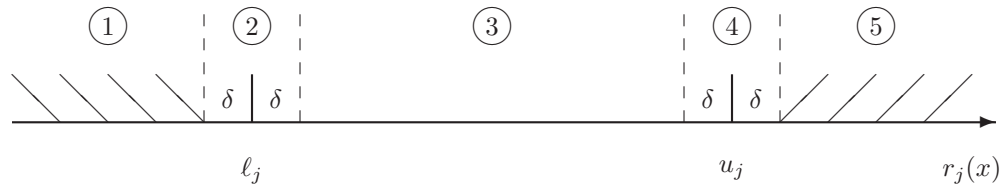
Figure 1: Illustration of the constraints $\ell_j \leq r_j(x) \leq u_j$.

Figure 1 illustrates the jth pair of constraints $\ell_j \leq r_j(x) \leq u_j$ in problem LCQP. The constant δ is the `Feasibility tolerance`. The constraints $\ell_j \leq r_j \leq u_j$ are considered "satisfied" if r_j lies in Regions 2, 3 or 4, and "inactive" if r_j lies in Region 3. The constraint $r_j \geq \ell_j$ is considered "active" in Region 2 and "violated" in Region 1. Similarly, $r_j \leq u_j$ is active in Region 4 and violated in Region 5. For equality constraints ($\ell_j = u_j$), Regions 2 and 4 are the same and Region 3 is empty.

1.3.   Input data

Most of the data for LCQP is supplied as parameters to subroutine `qpopt`. An initial estimate of the solution x must be provided in parameter `x`. For the `QP` options, the user may supply H or G *explicitly* as a matrix (see parameter `H` of subroutine `qpopt`), or *implicitly* via a subroutine that computes the product Hx for any given vector x (see parameter `qpHess` of `qpopt`). An example is given in Section 9.

QPOPT can accept information about which constraints are likely to be active at the solution. This *Warm start* facility may reduce computational effort significantly with a sequence of related problems. For example, NPSOL [GMSW86] uses this feature in a sequential quadratic programming method for nonlinearly constrained optimization.

1.4.   Subroutines

QPOPT is accessed via the following routines:

`qpopt`   (§4) The top-level routine, called by the user.

`qpHess` (§5) Called by `qpopt`. Defines Hx for given vectors x.

`qpprms` (§6.2) Called by the user to read an Options file (if any).

`qpprm, qpprmi, qpprmr` (§6.3) Called by the user to input a single option.

1.5.   Files

QPOPT reads or creates the following files:

Options file. If present, this is input by calling `qpprms`.

Summary file. Intended for output to the screen in an interactive environment. It contains error messages and a brief iteration log, or may be suppressed.

Print file. Intended for a permanent file. It contains error messages, a more detailed iteration log, and optionally the printed solution.

1.6.   Exit conditions

In general, a successful run of QPOPT will indicate one of three situations:

A minimizer was found. If H is positive definite or positive semidefinite, the final solution x is a *global minimizer*. (All other feasible points give a higher objective value.) Otherwise, the solution is a *local minimizer*, which may or may not be global. (All other points in the immediate neighborhood give a higher objective.)

A dead-point was reached. This might occur for problems types `QP1` and `QP2`, if H is not sufficiently positive definite. The necessary conditions for a local minimizer are satisfied but the sufficient conditions are not. If H is positive semidefinite, the solution is a *weak minimizer*. (The objective value is a global optimum, but there may be neighboring points with the same objective value.) If H is indefinite, a feasible direction of decrease may or may not exist (so the point may not be a local or weak minimizer).

The solution is unbounded. The objective can be made arbitrarily negative if some components of x are allowed to become arbitrarily large. Additional constraints may be needed. This cannot occur if H is sufficiently positive definite.

1.7.   Implementation

The source code for QPOPT is about 14,000 lines of Fortran 77 (ANSI Standard X3.9-1978), of which nearly 50% are comments. If there are n variables and m_L general constraints, the storage required is approximately $24n(n + m_L)$ Kbytes.

2.   Description of Method

QPOPT is based on an inertia-controlling method that maintains a Cholesky factorization of the reduced Hessian (see below). The method follows Gill and Murray [GM78] and is described in [GMSW91]. Here we briefly summarize the main features of the method. Where possible, we refer to the following quantities by name: the parameters of subroutine `qpopt`, the optional parameters, and items that appear in the printed output.

2.1.   Overview

QPOPT's method has a *feasibility phase* (finding a feasible point by minimizing the sum of infeasibilities) and an *optimality phase* (minimizing the quadratic objective function within the feasible region). The computations in both phases are performed by the same subroutines, but with different objective functions. The feasibility phase does *not* perform the standard simplex method; i.e., it does not necessarily find a vertex (with n constraints active), except in the LP case if $m_L \leq n$. Once an iterate is feasible, all subsequent iterates remain feasible. Once a vertex is reached, all subsequent iterates are at a vertex.

QPOPT is designed to be efficient when applied to a *sequence* of related problems—for example, within a sequential quadratic programming method for nonlinearly constrained optimization (e.g., the NPSOL package [GMSW86]). In particular, the user may specify an initial working set (the indices of the constraints believed to be satisfied exactly at the solution); see the discussion of `Warm Start`.

In general, an iterative process is required to solve a quadratic program. Each new iterate $\bar{x}$ is defined by

$$\bar{x} = x + \alpha p, \tag{2.1}$$

where the *step length* α is a non-negative scalar, and p is called the *search direction*. (For simplicity, we shall consider a typical iteration and avoid reference to the iteration index.)

2.2.   The working set

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 `Feasibility tolerance`). The working set is the current prediction of the constraints that hold with equality at a solution of LCQP. Let m_w denote the number of constraints in the working set (including bounds), and let W denote the associated $m_w \times n$ matrix of constraint gradients.

The definition of the search direction ensures that constraints in the working set remain *unaltered* for any value of the step length. Thus,

$$Wp = 0. \tag{2.2}$$

In order to compute p, a *TQ factorization* of W is used:

$$WQ = \begin{pmatrix} 0 & T \end{pmatrix}, \tag{2.3}$$

where T is a nonsingular $m_w \times m_w$ upper-triangular matrix, and Q is an $n \times n$ nonsingular matrix constructed from a product of orthogonal transformations (see [GMSW84]). If the columns of Q are partitioned so that

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

where Y is $n \times m_w$ and Z is $n \times n_z$ (where $n_z = n - m_w$), then the columns of Z form a basis for the null space of W. 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 printed output from `qpopt`). In many cases, Z_R will include *all* the columns of Z. The direction p will satisfy (2.2) if

$$p = Z_R p_R, \tag{2.4}$$

where p_R is any n_R-vector.

2.3. The reduced Hessian

Let g_Q and H_Q denote the *transformed gradient* and *transformed Hessian*:

$$g_Q = Q^T g(x) \quad \text{and} \quad H_Q = Q^T H Q.$$

The first n_R elements of the vector g_Q will be denoted by g_R, and the first n_R rows and columns of the matrix H_Q will be denoted by H_R. The quantities g_R and H_R are known as the *reduced gradient* and *reduced Hessian* of $q(x)$, 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 $D = \text{diag}(1, 1, \ldots, 1, \omega)$, with $\omega \leq 0$.

  In QPOPT, 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 (n_Rth) position. This allows p_R in (2.4) to be computed from a single back-substitution,

$$R p_R = \gamma e_R, \tag{2.5}$$

where γ is a scalar whose definition depends on whether 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 working-set constraints being treated as equalities. If H_R is not positive definite, p_R satisfies

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

allowing the objective function to be reduced by any step of the form $x + \alpha p$, $\alpha > 0$.

2.4. Optimality conditions

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 elsewhere 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 are defined from the equations

$$W^T \lambda = g(x). \tag{2.6}$$

A Lagrange multiplier λ_j corresponding to an inequality constraint in the working set is said to be *optimal* if $\lambda_j \leq \sigma$ when the associated constraint is at its *upper bound*, or if $\lambda_j \geq -\sigma$ when the associated constraint is at its *lower bound*, where σ depends on the `Optimality tolerance`. If a multiplier is non-optimal, the objective function (either the true objective or the sum of infeasibilities) can be reduced by deleting the corresponding constraint from the working set (with index `Jdel`; see Section 7).

  If optimal multipliers occur during the feasibility phase but the sum of infeasibilities is not zero, there is no feasible point. The user can request QPOPT to continue until the

sum of infeasibilities is minimized (see the discussion of `Min sum`). At such a point, the Lagrange multiplier λ_j corresponding to an inequality constraint in the working set will be such that $-(1 + \sigma) \leq \lambda_j \leq \sigma$ when the associated constraint is at its *upper bound*, and $-\sigma \leq \lambda_j \leq 1 + \sigma$ when the associated constraint is at its *lower bound*. Lagrange multipliers for equality constraints will satisfy $|\lambda_j| \leq 1 + \sigma$.

If the reduced gradient is not zero, Lagrange multipliers need not be computed and the search direction p is given by $Z_R p_R$ (see (2.5)). The step length is chosen to maintain feasibility with respect to the satisfied constraints. If H_R is positive definite and $x + p$ is feasible, α is defined to be one. 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 7). This constraint is added to the working set at the next iteration.

If the reduced Hessian H_R is not positive definite and α_M does not exist (i.e., no positive step α_M reaches the boundary of a constraint not in the working set), then QPOPT terminates at x and declares the problem to be unbounded.

3. Further Details of the Method

The following sections are not essential knowledge for normal users. They give background on the active-set strategy and the anti-cycling procedure.

3.1. Treatment of simple upper and lower bounds

Bound constraints $\ell \leq x \leq u$ are treated specially by `qpopt`. The presence of a bound constraint in the working set has the effect of fixing the corresponding component 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. For some permutation P, the working-set matrix satisfies

$$WP = \begin{pmatrix} F & N \\ & I_N \end{pmatrix},$$

where $\begin{pmatrix} F & N \end{pmatrix}$ is part of the matrix A, and I_N corresponds to some of the bounds. The matrices F and N contain the free and fixed columns of the general constraints in the working set. A TQ factorization $FQ_F = \begin{pmatrix} 0 & T_F \end{pmatrix}$ of the smaller matrix F provides the required T and Q as follows:

$$Q = P \begin{pmatrix} Q_F & \\ & I_N \end{pmatrix}, \qquad T = \begin{pmatrix} T_F & N \\ & I_N \end{pmatrix}.$$

The matrix Q_F is implemented as a dense *orthogonal* matrix. Each change in the working set leads to a simple change to F: if the status of a general constraint changes, a *row* of F is altered; if a bound constraint enters or leaves the working set, a *column* of F changes. The matrices T_F, Q_F and R are held explicitly; together with the vectors $Q^T g$, and $Q^T c$. Products of plane rotations are used to update Q_F and T_F as the working set changes. The triangular factor R associated with the reduced Hessian is updated only during the optimality phase.

3.2. The initial working set

For a cold start, the initial working set includes equality constraints and others that are close to being satisfied at the starting point. ("Close" is defined under `Crash tolerance`.) For a warm start, the initial working is specified by the user (and possibly revised to improve the condition of W).

At the start of the optimality phase, QPOPT must ensure that the initial reduced Hessian H_R is positive-definite. It does so by including a suitably large number of constraints (real or artificial) in the initial working set. (When W contains n constraints, H_R has no rows and columns. Such a matrix is positive definite by definition.)

Let H_Z denote the first n_z rows and columns of $H_Q = Q^T H Q$ at the beginning of the optimality phase. A partial Cholesky factorization with interchanges is used to find an upper-triangular matrix R that is the factor of the largest positive-definite leading submatrix of H_Z. The use of interchanges tends to maximize the dimension of R. (The condition of R may be controlled by setting the `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" (with bounds equal to the current value of $Z_A^T x$). Minimization of the objective function then proceeds within the subspace defined by Z_R, as described in Section 2.

The artificially augmented working set is given by

$$\bar{W} = \begin{pmatrix} Z_A^T \\ W \end{pmatrix},$$

so that p will satisfy $Wp = 0$ *and* $Z_A^T p = 0$. By definition of the TQ factors of W, we have

$$\bar{W}Q = \begin{pmatrix} Z_A^T \\ W \end{pmatrix} Q = \begin{pmatrix} Z_A^T \\ W \end{pmatrix} \begin{pmatrix} Z_R & Z_A & Y \end{pmatrix} = \begin{pmatrix} 0 & \bar{T} \end{pmatrix},$$

where

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

Hence the TQ factors of $\bar{W}$ are available trivially.

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 $Z_R^T g = 0$, since this simply involves repartitioning Q. The "artificial" multiplier vector associated with the rows of Z_A^T is equal to $Z_A^T g$, 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 printed output (see Section 7). The multiplier may have either sign.

The number of columns in Z_A and Z_R, the Euclidean norm of $Z_R^T g$, and the condition estimator of R appear in the printed output as Art, Zr, Norm gZ and Cond Rz (see Section 7).

Under some circumstances, a different type of artificial constraint is used when solving a linear program. Although the algorithm of qpopt 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., $m_L \le n$). (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 similarly to normal constraints 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 printed output (see Section 7). Again, the multiplier may have either sign.

3.3. The anti-cycling procedure

The EXPAND procedure [GMSW89] is used to reduce the possibility of cycling at a point where the active constraints are nearly linearly dependent. The main feature of EXPAND is that the feasibility tolerance is increased slightly at the start of every iteration. This allows a positive step to be taken every iteration, perhaps at the expense of violating the constraints slightly.

Suppose that the Feasibility tolerance is δ. Over a period of K iterations (where K is defined by the Expand frequency), the feasibility tolerance actually used by QPOPT—the *working* feasibility tolerance—increases from 0.5δ to δ (in steps of $0.5\delta/K$).

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 K iterations, the resetting procedure is invoked and a new cycle of iterations is started with K 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 QPOPT reaches an apparently optimal, infeasible or unbounded solution, unless this situation has already occurred twice. If any nontrivial adjustments are made, iterations are continued.

The EXPAND procedure not only allows a positive step to be taken at every iteration, but also provides a potential *choice* of constraints to be added to the working set. Let α_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 distance α ($\alpha \leq \alpha_M$) 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. This strategy helps keep the working-set matrix W well-conditioned.

4.   Subroutine qpopt

Problem LCQP is solved by a call to subroutine `qpopt`, whose parameters are defined here. Note that most machines use `double precision` declarations as shown, but some machines use `real`. The same applies to the user routine `qpHess`.

Specification:

```
 subroutine qpopt ( n, nclin, ldA, ldH,
$                   A, bl, bu, cvec, H,
$                   qpHess, istate, x,
$                   inform, iter, obj, Ax, clamda,
$                   iw, leniw, w, lenw )

 external          qpHess
 integer           leniw, lenw
 integer           istate(n+nclin)
 integer           iw(leniw)
 double precision  A(ldA,*), Ax(*), bl(n+nclin), bu(n+nclin)
 double precision  clamda(n+nclin), cvec(*)
 double precision  H(ldH,*), x(n)
 double precision  w(lenw)
```

On entry:

n (> 0) is n, the number of variables in the problem.

nclin (≥ 0) is m_L, the number of general linear constraints.

ldA $(\geq 1$ and $\geq$ `nclin`) is the row dimension of the array `A`.

ldH $(\geq 1$ and $\geq$ `n`) is the row dimension of the array `H`. (`ldH` must be at least the value of `Hessian Rows` if that parameter is set.)

A   is an array of dimension (`ldA`, k) for some $k \geq$ `n`. It contains the matrix A for the linear constraints. If `nclin` is zero, `A` is not referenced. (In that case, `A` may be dimensioned (`ldA,1`) with `ldA` = 1, or it could be any convenient array.)

bl   is an array of dimension at least `n` + `nclin` containing the lower bounds ℓ in problem LCQP. To specify a non-existent bound $(\ell_j = -\infty)$, set `bl`$(j) \leq -$`bigbnd`, where `bigbnd` is the `Infinite Bound` (default value 10^{20}). To specify an *equality* constraint $r_j(x) = \beta$, set `bl`$(j) =$ `bu`$(j) = \beta$, where $|\beta| <$ `bigbnd`.

bu   is an array of dimension at least `n`+`nclin` containing the upper bounds u in problem LCQP. To specify a non-existent bound $(u_j = \infty)$, set `bu`$(j) \geq$ `bigbnd`. The bounds must satisfy `bl`$(j) \leq$ `bu`(j) for all j.

cvec   is an array of dimension at least `n` that contains the explicit linear term c of the objective. If the problem is of type `FP`, `QP1`, or `QP3`, `cvec` is not referenced. (In that case, `cvec` may be dimensioned (`1`), or it could be any convenient array.)

H   is an array of dimension (`ldH`, k) for some $k \geq$ `n`. `H` may be used to store the matrix H associated with the quadratic term of the QP objective. It is not referenced if the problem type is `FP` or `LP`. (In that case, `H` may be dimensioned (`ldH,1`) with `ldH` = 1, or it could be any convenient array.)

Let the number of `Hessian rows` be m (with default value $m = n$). For problems `QP1` or `QP2`, the first m rows and columns of `H` must contain the leading $m \times m$ 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.

For problems `QP3` and `QP4`, the first m rows of `H` must contain an $m \times n$ upper-trapezoidal matrix G such that $H = G^T G$. The factor G need not be of full rank, i.e., some of the diagonals may be zero. However, as a general rule, the larger the dimension of the leading non-singular submatrix of G, the fewer iterations will be required. Elements outside the upper-triangular part of the first m rows of `H` are assumed to be zero and need not be assigned.

In some cases, `H` need not be used; see the next parameter `qpHess`.

`qpHess` is the name of a subroutine that defines the product Hx for a given vector x. It must be declared as `external` in the routine that calls `qpopt`. In general, the user need not provide this parameter, because a "default" subroutine named `qpHess` is distributed with QPOPT. It uses the array `H` defined above. In some cases, a specialized routine may be desirable. For a detailed description of `qpHess`, see Section 5.

`istate` is an integer array of dimension at least `n + nclin`. It need not be initialized if `qpopt` is called with a `Cold Start` (the default option).

For a `Warm Start`, `istate` must be set. It is used to choose the first working set. If `qpopt` has just been called on a problem with the same dimensions, `istate` already contains valid values. In general, $\texttt{istate}(j)$ should indicate whether either of the constraints $r_j(x) \geq \ell_j$ or $r_j(x) \leq u_j$ is expected to be active at a solution of LCQP.

The ordering of `istate` is the same as for `bl`, `bu` and $r(x)$, i.e., the first `n` components refer to the bounds on the variables, and the last `nclin` refer to the bounds on Ax. Possible values for $\texttt{istate}(j)$ follow.

0   Neither $r_j(x) \geq \ell_j$ nor $r_j(x) \leq u_j$ is expected to be active.

1      $r_j(x) \geq \ell_j$ is expected to be active.

2      $r_j(x) \leq u_j$ is expected to be active.

3   This may be used if $\ell_j = u_j$. Normally an equality constraint $r_j(x) = \ell_j = u_j$ is active at a solution.

The values 1, 2 and 3 have the same effect when $\texttt{bl}(j) = \texttt{bu}(j)$. If necessary, `qpopt` will override the given values, so a poor choice will not cause the algorithm to fail.

`x`   is an array of dimension at least `n`. It contains an initial estimate of the solution.

`iw`   is an array of dimension `leniw` that provides integer workspace.

`leniw` is the dimension of `iw`. It must be at least $2\,\texttt{n} + 3$.

`w`   is an array of dimension `lenw` that provides real workspace.

`lenw`   is the dimension of `w`. It depends on the Problem type and `nclin` as shown in the following table.

Problem type	nclin	Minimum value of lenw
FP	$\geq$ n	$2\,n^2 + 7\,n + 5\,\texttt{nclin}$
	$0 < \texttt{nclin} < \texttt{n}$	$2\,(\texttt{nclin} + 1)^2 + 7\,n + 5\,\texttt{nclin}$
	$= 0$	$7\,n + 1$
LP	$\geq$ n	$2\,n^2 + 8\,n + 5\,\texttt{nclin}$
	$0 < \texttt{nclin} < \texttt{n}$	$2\,(\texttt{nclin} + 1)^2 + 8\,n + 5\,\texttt{nclin}$
	$= 0$	$8\,n + 1$
QP1 or QP3	> 0	$2\,n^2 + 7\,n + 5\,\texttt{nclin}$
	$= 0$	$n^2 + 7\,n$
QP2 or QP4	> 0	$2\,n^2 + 8\,n + 5\,\texttt{nclin}$
	$= 0$	$n^2 + 8\,n$

If insufficient workspace is provided, minimum acceptable values of `lenw` and `leniw` are printed on the Summary file and Print file. In this event, minimum values of `leniw` and `lenw` are stored in `iw(1)` and `iw(2)` respectively. Thus, appropriate values may be obtained from a preliminary run with `lenw = 1`. (The values will be printed before `qpopt` terminates with `inform = 6`.)

On exit:

`inform` reports the result of the call to `qpopt`. (If `printlevel > 0`, a short description of `inform` is printed.) Specific values of `inform` follow.

0   `x` is a unique local minimizer. This means that `x` is *feasible* (it satisfies the constraints to the accuracy requested by the `Feasibility tolerance`), the reduced gradient is negligible, the Lagrange multipliers are optimal, and the reduced Hessian H_R is positive definite. If H is positive definite or positive semidefinite, `x` is a *global minimizer*. (All other feasible points give a higher objective value.) Otherwise, the solution is a *local minimizer*, which may or may not be global. (All other points in the immediate neighborhood give a higher objective.)

1   A dead-point was reached. This might occur for problems types `QP1` and `QP2`, if H is not sufficiently positive definite. If H is positive semidefinite, the solution is a *weak minimizer*. (The objective value is a global optimum, but there may be infinitely many neighboring points with the same objective value.) If H is indefinite, a feasible direction of decrease may or may not exist (so the point may not be a local or weak minimizer).

         At a dead-point, the necessary conditions for optimality are satisfied (`x` is feasible, the reduced gradient is negligible, the Lagrange multipliers are optimal, and H_R is positive semidefinite.) However, H_R is nearly singular, and/or there are some very small multipliers. If H is indefinite, `x` is not necessarily a local solution of the problem. Verification of optimality requires further information, and is in general an NP-hard problem [PS88].

2   The solution appears to be unbounded. The objective is not bounded below in the feasible region, if the elements of `x` are allowed to be arbitrarily large. This occurs if a step larger than `Infinite Step` would have to be taken in order to continue the algorithm, or the next step would result in a component of `x` having magnitude larger than `Infinite Bound`. It should not occur if H is sufficiently positive definite.

3   The constraints could not be satisfied. The problem has no feasible solution. See Section 8.4 for further comments.

4   One of the iteration limits was reached before normal termination occurred. See `Feasibility Phase Iterations` and `Optimality Phase Iterations`.

5   The `Maximum degrees of freedom` is too small. The reduced Hessian must expand if further progress is to be made.

6   An input parameter was invalid.

7   The `Problem type` was not recognized.

`iter`   is the total number of iterations performed in the feasibility phase and the optimality phase.

`istate` describes the status of the constraints $\ell \leq r(x) \leq u$ in problem LCQP. For the jth lower or upper bound, $j = 1$ to $\mathtt{n} + \mathtt{nclin}$, the possible values of $\mathtt{istate}(j)$ are as follows (where δ is the `Feasibility tolerance`; see Figure 1).

-2 (Region 1) The lower bound is violated by more than δ.

-1 (Region 5) The upper bound is violated by more than δ.

  0   (Region 3) Both bounds are satisfied by more than δ.

  1   (Region 2) The lower bound is active (to within δ).

  2   (Region 4) The upper bound is active (to within δ).

  3   (Region 2 = Region 4) The bounds are equal and the equality constraint is satisfied (to within δ).

  4   (Region 2, 3 or 4) The quantity $r_j(x)$ is temporarily fixed at its current value, which may or may not be equal to a bound.

These values are labeled in the printed solution according to the following table.

Region	1	2	3	4	5	$2 \equiv 4$	2–4
$\mathtt{istate}(j)$	-2	1	0	2	-1	3	4
Printed solution	--	LL	FR	UL	++	EQ	TF

`Ax`   is an array of dimension at least `nclin` that contains the linear constraint functions Ax at the final iterate. If $\mathtt{nclin} = 0$, `Ax` is not referenced. (In that case, `Ax` may be dimensioned `(1)`, or it could be any convenient array.)

`clamda` contains the Lagrange multipliers at `x`. At an optimal solution, $\mathtt{clamda}(j)$ will be non-negative if $\mathtt{istate}(j) = 1$ and non-positive if $\mathtt{istate}(j) = 2$.

`obj`   is the final value of the QP objective if `x` is feasible (zero for problem FP), or the sum of infeasibilities if `x` is infeasible.

`x`   contains the final estimate of the solution.

5. Subroutine qpHess

Quadratic programs have a matrix H in the objective function. The method employed by qpopt requires products of the form Hx for given vectors x. These are provided by the parameter qpHess (an external subroutine).

QPOPT contains a standard subroutine qpHess to compute such products from the input parameter H, as described in Section 4. In some cases, it may be more efficient to use a special version of qpHess.

Subroutine qpHess is not accessed if the problem type is FP or LP.

Specification:

```
subroutine qpHess( n, ldH, jthcol, H, x, Hx, iw, leniw, w, lenw )

integer           n, ldH, jthcol
integer           leniw, lenw
integer           iw(leniw)
double precision  H(ldH,*), Hx(n), x(n)
double precision  w(lenw)
```

On entry:

n   is the same as the input parameter of qpopt. It must not be altered. Similarly for the parameters ldH, H, iw, leniw, w and lenw.

jthcol   may be ignored if it is convenient to treat all vectors x the same way. In general, jthcol is an integer j. If $j = 0$, x contains a general vector x. If $1 \le j \le$ n, x contains the jth column of the identity matrix, and it may be easy to code the product Hx specially without referencing x. (The product is the jth column of H.)

x   contains a vector x such that the product Hx should be returned in Hx.

On exit:

Hx   should contain the product Hx for the vector stored in x.

Note that the array H is never touched by qpopt; it is just passed to qpHess. The default version of qpHess uses H as a two-dimensional array. In some cases it may be desirable to use a one-dimensional array to transmit data or workspace to special versions of qpHess. H should then be declared as double precision H(ldH).

In other situations, it may be desirable to compute Hx without accessing H at all. For example, H may be sparse or have a regular structure. (See subroutine qpHes1 in Section 9.2.) The parameters H and ldH may then refer to any convenient array.

6. The Options File

Several choices in QPOPT's algorithm logic may be defined by various *optional parameters* (more briefly known as *options* or *parameters*).

In order to reduce the number of subroutine parameters for qpopt, the options have *default values* that are appropriate for most problems. Options need be specified only if their values should be different from the default.

New values may be specified by calling subroutines qpprms, qpprm, qpprmi or qpprmr (Sections 6.2 and 6.3). Each such option is listed on the Print file unless the first option is either Nolist or Print file 0.

Options are not altered by QPOPT, so that any changes are *cumulative*. The option Defaults may be used to reset all options to their default values.

6.1. Format of option strings

Each optional parameter is defined by an *option string* of up to 72 characters, containing one or more *items* separated by spaces or equal signs (=). Alphabetic characters may be in upper or lower case. An example option string is Print level = 5. In general, an option string contains the following items:

1. A *keyword* such as Print.

2. A *phrase* such as level that qualifies the keyword. (Typically 0, 1 or 2 words.)

3. A *number* that specifies either an integer or a real value (only for some options). Such numbers may be up to 16 contiguous characters in Fortran 77's F, E or D formats, terminated by a space.

Blank strings and comments may be used to improve readability. A *comment* begins with an asterisk (∗) and all subsequent characters are ignored. Synonyms are recognized for some of the keywords, and abbreviations may be used if there is no ambiguity.

The following are examples of valid option strings for QPOPT:

```
NOLIST
COLD START
Warm start
Problem type  = LP
Problem type  = Quadratic Program        * Same as QP or QP2
Problem Type    QP4
Min sum         Yes
Feasibility Phase iteration limit   100
Feasibility tolerance               1.0e-8  * for IEEE double precision
Crash tolerance                     0.002
Defaults
* This string will be ignored.          So will a blank line.
```

6.2.   Subroutine qpprms (to read an Options file)

Subroutine `qpprms` provided with QPOPT reads options from an external Options file.

Specification:

```
      subroutine qpprms( ioptns, inform )
      integer            ioptns, inform
```

On entry:

`ioptns` is the unit number of the Options file, in the range $[0, 99]$. It is not changed.

On exit:

`inform` reports the result of the call to `qpprms` as follows.

  0   A valid Options file was found.

  1   `ioptns` is out of range.

  2   The file does not begin with `Begin` or end with `End`.

Each line of the Options file defines a single optional parameter. The file must be delimited by `Begin` and `End`. For example:

```
      Begin
         Problem type LP
         Print file    9
         Print level = 5
      End
```

If this Options file is on unit number 4, it can be input as follows:

```
      ioptns = 4
      call qpprms( ioptns, inform )
```

In some cases, the file associated with unit `ioptns` may need to be opened before the call to `qpprms`. It may also need to be closed and reopened if it is to be re-read.

6.3. Subroutines qpprm, qpprmi, qpprmr (to define a single option)

The second method of setting the optional parameters is through a series of calls to the following subroutines. Each call sets one option.

Specification:

```
subroutine qpprm ( string )
character*(*)      string


subroutine qpprmi( string, ivalue )
character*(*)      string
integer            ivalue


subroutine qpprmr( string, rvalue )
character*(*)      string
double precision   rvalue
```

On entry:

string must be a valid option string.

ivalue is the required integer value.

rvalue is the required real value.

On exit:

All parameters are unchanged.

The following examples illustrate setting options within the calling program. Note that on most machines, `featol` must be declared `double precision`.

```
maxitn = 200
featol = 1.0d-6
call qpprm ( 'Problem type              QP3' )
call qpprmi( 'Feasibility Phase iterations', maxitn )
call qpprmr( 'Feasibility tolerance       ', featol )
```

6.4.   Description of the optional parameters

Permissible options are defined below in alphabetical order. For each option, we give the keyword, any essential qualifiers, the default value, and the definition. The minimum abbreviation of each keyword and qualifier is underlined. If no characters of a qualifier are underlined, the qualifier may be omitted. The letters i and r denote `integer` and `real` values required for certain options. The letter a denotes a character string value. The number **u** represents unit roundoff for floating-point arithmetic (typically about 10^{-16}).

<u>Chec</u>k frequency i 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 working-set matrix is refactorized and the variables are recomputed to satisfy the constraints more accurately.

<u>Col</u>d start   Default $=$ `Coldstart`
<u>Warm</u> start

This option specifies how the initial working set is chosen. With a cold start, QPOPT chooses the initial working set based on the values of the variables and constraints at the initial point. Broadly speaking, the first working set will include all equality constraints and also any bounds or inequality constraints that are "nearly" satisfied (to within the `Crash tolerance`).
    With a warm start, the user must provide a valid definition of every element of the array `istate` (see Section 4). The specification of `istate` will be overridden if necessary, so that a poor choice of the working set will not cause a fatal error. A warm start will be advantageous if a good estimate of the initial working set is available—for example, when `qpopt` is called repeatedly to solve related problems.

<u>Cra</u>sh tolerance r Default $= 0.01$

This value is used for cold starts when QPOPT selects an initial working set. Bounds and inequality constraints are selected if they are satisfied to within r. More precisely, 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.

<u>Def</u>aults

This is a special option to reset all options to their default values.

<u>Exp</u>and frequency i Default $= 5$

This defines the initial value of an integer K that is used in an anti-cycling procedure designed to guarantee progress even on highly degenerate problems. See Section 3.3.
    If $i \geq 9999999$, no anti-cycling procedure is invoked.

<u>F</u>easibility tolerance r Default $= \sqrt{\mathbf{u}}$

This 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 $r < \mathbf{u}$, the default value is used.

Before optimizing the objective function, QPOPT must find a feasible point for the constraints. If the sum of infeasibilities cannot be reduced to zero and `Min sum = Yes` is requested, QPOPT will 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.

<u>F</u>easibility <u>P</u>hase Iteration Limit	i_1	Default $= \max(50, 5(n + m_L))$
<u>O</u>ptimality <u>P</u>hase Iteration Limit	i_2	Default $= \max(50, 5(n + m_L))$

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 `PrintLevel > 0` means that the workspace needed will be computed and printed, but no iterations will be performed.

<u>Hes</u>sian rows	i	Default $= 0$ or n

This specifies m, the number of rows in the Hessian matrix H or its trapezoidal factor G (as used by the default subroutine `qpHess`).

For problem type FP or LP, the default value is $m = 0$.

For problems QP1 or QP2, the first m rows and columns of H are obtained from H, and the remainder are assumed to be zero. For problems QP3 or QP4, the factor G is assumed to have m rows and n columns. They are obtained from the associated rows of H.

If a nonstandard subroutine `qpHess` is provided, it may access the problem type and m via the lines

```
    integer         lqptyp, mHess
    common    /sol1qp/ lqptyp, mHess
```

For example, `Problem type FP, LP` or `QP4` sets `lqptyp` $= 1$, 2 or 6 respectively, and `Hessian rows 20` sets `mHess` $= 20$.

<u>In</u>finite <u>B</u>ound size	r	Default $= 10^{20}$

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

<u>In</u>finite <u>S</u>tep size	r	Default $= \max(\text{bigbnd}, 10^{20})$

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 `Infinite Step`, the objective function is considered to be unbounded below in the feasible region. If $r \le 0$, the default value is used.

<u>It</u>eration limit	i	Default $= \max(50, 5(n + m_L))$
<u>It</u>ers		
<u>It</u>ns		

This is equivalent to `Optimality Phase iteration limit`. See `Feasibility Phase`.

<u>Lis</u>t

If `Nolist` was previously specified, `List` restores output to the Print file whenever an optional parameter is reset.

<u>Max</u>imum <u>degrees of freedom</u> i Default $= n$

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. (See Sections 2.2 and 2.3.) 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.

<u>Min</u> <u>sum</u> a Default $= $ `No`

This option comes into effect if the constraints cannot be satisfied. If `Min sum = No`, QPOPT terminates as soon as it is evident that no feasible point exists. The final point will generally not be the point at which the sum of infeasibilities is minimized. If `Min sum = Yes`, QPOPT will continue until either the sum of infeasibilities is minimized or the iteration limit is reached, whichever occurs first.

<u>No</u>lis<u>t</u>

This suppresses output to the Print file whenever an optional parameter is reset.

<u>O</u>ptimality <u>tolerance</u> r Default $= \sqrt{\mathbf{u}}$

This affects the tolerance used to determine if the Lagrange multipliers associated with the bounds and general constraints have the right "sign" for the solution to be judged optimal. Increasing r tends to cause earlier termination. For example, if $r = $ `1.0e − 4`, the final objective value will probably agree with the true optimum to about 4 digits.

<u>Pri</u>nt <u>file</u> i Default $= 9$

This specifies the unit number for the Print file (see Section 8).

   If $i > 0$ and `PrintLevel` > 0, a full log in 132-column format is output to unit i. `Print file = 0` suppresses all output, *including error messages* and the QPOPT banner.

<u>Pri</u>nt <u>level</u> i Default $= 10$

This controls the amount of printing produced by QPOPT as follows.

   i

   0   No output.

   1   The final solution only, sent to the Print file.

   5   One line of output for each iteration (no printout of the final solution).

≥ 10 The final solution and one line of output for each iteration (Print file only).

≥ 20 At each iteration, the Lagrange multipliers, the variables x, the constraint values Ax and the constraint status (Print file only).

≥ 30 At each iteration, the diagonal elements of the upper-triangular matrix T associated with the TQ factorization (2.3) of the working set, and the diagonal elements of the upper-triangular matrix R (Print file only).

<u>Prob</u>lem type a Default = `QP2`

This option specifies the type of objective function to be minimized during the optimality phase. The following are the six values of a and the dimensions of the arrays that must be specified to define the objective function:

FP	H and `cvec` not accessed;
LP	H not accessed, `cvec(n)` required;
QP1	`H(ldH,*)` symmetric, `cvec` not referenced;
QP2	`H(ldH,*)` symmetric, `cvec(n)`;
QP3	`H(ldH,*)` upper-trapezoidal, `cvec` not referenced;
QP4	`H(ldH,*)` upper-trapezoidal, `cvec(n)`;

`Linear program` is equivalent to `LP`. `Quadratic program` and `QP` are equivalent to the default option `QP2`. For the QP options, the default subroutine `qpHess` requires array `H(ldH,*)` as shown. If a non-standard `qpHess` is provided, `H(*,*)` may be used in any convenient way.

<u>Ran</u>k tolerance r Default = $100\mathbf{u}$

This parameter enables the user to control the condition number of the triangular factor R (see Section 2). 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 $|R_{i+1,i+1}| \leq \sqrt{r}|\rho_{i+1}|$. If $r \leq 0$, the default value is used.

<u>Sum</u>mary file i Default = 6

This specifies the unit number for the Summary file (see Section 7).

     If $i > 0$ and `PrintLevel` > 0, a brief log in 80-column format is output to unit i. On many systems, the default value refers to the screen. `Summary file = 0` suppresses output, *including error messages*.

Warm start

See `Cold start`.

6.5. Optional parameter checklist and default values

For easy reference, the following list shows all valid options and their default values. The quantity $\mathbf{u}$ represents floating-point precision ($\approx 1.1 \times 10^{-16}$ in IEEE double-precision arithmetic).

```
Check frequency                   50     *
Cold start                               *
Crash tolerance                  .01     *
Expand frequency                   5     *
Feasibility tolerance          1.1e-8    *  √u
Feasibility Phase iteration limit  50    *  or 5(n + mₗ)
Optimality Phase iteration limit   50    *  or 5(n + mₗ)
Hessian rows                       n     *
Infinite bound size           1.0e+20    *  Plus infinity
Infinite step size            1.0e+20    *
Iteration limit                   50     *  or 5(n + mₗ)
List                                     *
Maximum degrees of freedom         n     *
Min sum                           No     *
Optimality tolerance           1.1e-8    *  √u
Print file                         9     *
Print level                       10     *
Problem type                      QP     *  or QP2
Rank tolerance                1.1e-14    *  100u
Summary file                       6     *
```

Other options may be set as follows:

```
Defaults
Nolist
Warm start
```

7. The Summary File

The Summary file records an iteration log and error messages. It is intended for screen output, but may be directed to a permanent file or suppressed, using the `Summary file` and `Print level` options. The maximum record length is 63 characters.

By default, a Summary file is produced on unit 6. In general, output is produced if `Summaryfile` > 0 and `PrintLevel` > 0.

To suppress the Summary file, specify `Summary file 0`.

7.1. Constraint numbering and status

For items `Jdel` and `Jadd` in the iteration log, indices 1 through `n` refer to the bounds on the variables, and indices `n + 1` through `n + nclin` 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).

7.2. The iteration log

The following items are printed *after* each iteration.

`Itn`   is the iteration count (including those from the feasibility phase).

`Jdel`   is the index of the constraint deleted from the working set. If `Jdel` is zero, no constraint was deleted.

`Jadd`   is the index of the constraint added to the working set. If `Jadd` is zero, no constraint was added.

`Step`   is the step taken along the computed search direction. If a constraint is added during the current iteration (i.e., `Jadd` is positive), `Step` will be the step to the nearest constraint. During the optimality phase, the step can be greater than one only if the reduced Hessian is not positive definite.

`Ninf`   is the number of violated constraints (infeasibilities). This number 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. 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 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. Note that the *sum* of the infeasibilities may increase or decrease during this part of the feasibility phase. However, once optimal phase-one multipliers are obtained, the number of infeasibilities can increase, but the sum of infeasibilities must either remain constant or be reduced until the minimum sum of infeasibilities is found.

In the optimality phase, the value of the objective is non-increasing.

`Norm gZ`   is $\|Z_R^T g\|$, 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.

Zr	is the number of columns of Z_R (see Section 2). `Zr` is the dimension of the subspace in which the objective is currently being minimized. The value of `Zr` is the number of variables minus the number of constraints in the working set.
Art	is the number of artificial constraints in the working set, i.e., the number of columns of Z_A (see Section 3). At the start of the optimality phase, `Art` provides an estimate of the number of nonpositive eigenvalues in the reduced Hessian.

7.3. Summary file from the example problem

The following Summary file is produced by the example program described in Section 9.

```
                    QPOPT  ---  Version 1.0-10     Sep  1995
                    =========================================

Itn Jdel  Jadd     Step Ninf  Sinf/Objective Norm gZ    Zr  Art
  0   0      0   0.0E+00    0  0.00000000E+00 0.0E+00    0    6
Itn     0 -- Feasible point found.
  0   0      0   0.0E+00    0  1.51638000E+03 9.8E+01    1    5
  1   0     8U   2.8E-01    0  1.72380000E+02 0.0E+00    0    5
  2   1L   10L   3.1E-03    0  1.68083225E+02 0.0E+00    0    5
  3   5A   11L   1.2E-02    0  1.57176475E+02 0.0E+00    0    4


Itn Jdel  Jadd     Step Ninf  Sinf/Objective Norm gZ    Zr  Art
  4   4A   12L   3.2E-02    0  1.38528925E+02 0.0E+00    0    3
  5   3A   13L   6.9E-02    0  1.11295925E+02 0.0E+00    0    2
  6   2A   14L   1.3E-01    0  7.41228000E+01 0.0E+00    0    1
  7   1A    1U   8.4E-01    0 -5.85162625E+01 0.0E+00    0    0
  8  13L     0   1.0E+00    0 -8.72144740E+01 1.3E-15    1    0


Itn Jdel  Jadd     Step Ninf  Sinf/Objective Norm gZ    Zr  Art
  9   1U    6U   2.5E+00    0 -3.12744888E+02 1.4E+02    1    0
 10   0     1L   1.4E-01    0 -5.62265012E+02 0.0E+00    0    0
 11  14L    7U   1.3E-01    0 -6.21487825E+02 0.0E+00    0    0

Exit from QP problem after   11 iterations.  Inform =  0


                    QPOPT  ---  Version 1.0-10     Sep  1995
                    =========================================

Itn Jdel  Jadd     Step Ninf  Sinf/Objective Norm gZ    Zr  Art
  0   0      0   0.0E+00    3  2.35500000E+01 1.7E+00    0    3
  1   2U   10L   4.0E+00    2  1.96000000E+01 1.4E+00    0    3
  2   4U   12L   7.8E+00    1  1.17500000E+01 1.0E+00    0    3
  3   6U   14L   1.2E+01    0  0.00000000E+00 0.0E+00    0    3
Itn     3 -- Feasible point found.
  3   0      0   0.0E+00    0  8.66526437E+02 1.5E+02    1    2


Itn Jdel  Jadd     Step Ninf  Sinf/Objective Norm gZ    Zr  Art
  4   0     9L   1.0E-01    0  4.98244375E+01 0.0E+00    0    2
  5   2A   11L   4.5E-01    0 -5.62265013E+02 0.0E+00    0    1
  6   1A    6U   5.7E-13    0 -5.62265013E+02 0.0E+00    0    0
  7  14L    7U   1.3E-01    0 -6.21487825E+02 0.0E+00    0    0

Exit from QP problem after    7 iterations.  Inform =  0
```

8.   The Print File

The Print file records specified options, error messages, a detailed iteration log, and the final solution. It is intended for output to a permanent file, but may be directed to the screen or suppressed. The maximum record length is 114 characters.

By default, a Print file is produced on unit 9. In general, output is produced if `Print file` > 0, `PrintLevel` > 0, and the file number is different from the Summary file.

To suppress the Print file, specify `Print file 0` as the *first* option before or after a call to `qpopt`. If an Options file is specified, `Print file 0` must be the first option after the `begin` (no blank lines).

8.1.   Constraint numbering and status

Items `Jdel` and `Jadd` in the iteration log are the same as in the Summary file. Please see Section 7.1.

8.2.   The iteration log

When `PrintLevel` ≥ 5, a line of output is produced at every iteration. The quantities printed are those in effect *on completion* of the iteration. Several items are the same as in the Summary file. Please see Section 7.2.

`Itn`	Same as Summary file.
`Jdel`	Same as Summary file.
`Jadd`	Same as Summary file.
`Step`	Same as Summary file.
`Ninf`	Same as Summary file.
`Sinf/Objective`	Same as Summary file.
`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`	Same as Summary file.
`Zr`	Same as Summary file. $\mathtt{Zr} = \mathtt{n} - (\mathtt{Bnd} + \mathtt{Lin} + \mathtt{Art})$.
	The number of columns of Z (see Section 2) can be calculated as $\mathtt{Nz} = \mathtt{n} - (\mathtt{Bnd} + \mathtt{Lin}) = \mathtt{Zr} + \mathtt{Art}$. If $\mathtt{Nz}$ is zero, x lies at a vertex of the feasible region.
`Norm gZ`	Same as Summary file.
`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 the `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 matrix W.

Cond Rz	is a lower bound on the condition number of the triangular factor R (the Cholesky factor of the current reduced Hessian H_R, whose dimension is Zr). If the problem type is LP, Cond Rz is not printed.
Rzz	is the last diagonal element ω of the matrix D associated with the $R^T D R$ factorization of the reduced Hessian H_R (see Section 2). Rzz is only printed if H_R is not positive definite (in which case $\omega \neq 1$). 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.

8.3. Printing the solution

When PrintLevel $= 1$ or PrintLevel ≥ 10, the final output from qpopt includes a listing of the status of every variable and constraint. Numerical values that are zero are printed as ".". In the "Variables" section, the following output is given for each variable x_j ($j = 1$ to n).

Variable	gives j, the number of the variable.
State	gives the state of the variable. The possible states are as follows (see Fig. 1), where δ is the Feasibility tolerance.

> FR   The variable lies between its upper and lower bound.
>
> EQ   The variable is a fixed variable, with x_j equal to its upper and lower bound.
>
> LL   The variable is active at its lower bound (to within δ).
>
> UL   The variable is active at its upper bound (to within δ).
>
> TF   The variable is temporarily fixed at its current value.
>
> --   The lower bound is violated by more than δ.
>
> ++   The upper bound is violated by more than δ.

A key is sometimes printed before the State to give some additional information about the state of a variable.

> A   *Alternative optimum possible.* The variable is active at one of its bounds, but its Lagrange multiplier is essentially zero. This means that if the variable were allowed to start moving away from its bound, there would be no change to the objective function. The values of the other free variables *might* change, giving a genuine alternative solution. However, if there are any degenerate variables (labeled 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 δ.

Value	is the final value of the variable x_j.
Lower bound	is the lower bound specified for x_j. "None" indicates that $bl(j) \leq -$bigbnd.

Upper bound is the upper bound specified for x_j. "None" indicates that $\mathtt{bu}(j) \geq$ bigbnd.

Lagr multiplier is the Lagrange multiplier for the associated bound. This will be zero if State is FR. 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 $\mathtt{bl}(j)$ and $\mathtt{bu}(j)$. A blank entry indicates that the associated variable is not bounded (i.e., $\mathtt{bl}(j) \leq -\mathtt{bigbnd}$ and $\mathtt{bu}(j) \geq \mathtt{bigbnd}$).

In the "Constraints" section, similar output is given for each constraint $a_i^T x$, $i = 1$ to nclin. The word "variable" must be replaced by "constraint", and x_j should be changed to $a_i^T x$, and (j) should be changed to $(\mathtt{nclin} + i)$. "Movement off a constraint" means allowing the entry in the slack column to become positive.

8.4. Interpreting the printout

The input data for qpopt should always be checked (even if it terminates with $\mathtt{inform} = 0$!). Two common sources of error are uninitialized variables and incorrectly dimensioned array arguments. The user should check that all components of A, bl, bu and x are defined on entry to qpopt, and that qpHess computes all relevant components of Hx.

In the following, we list the different ways in which qpopt terminates abnormally and discuss what further action may be necessary.

Underflow A single underflow will always occur if machine constants are computed automatically (as in the distributed version of QPOPT). Other floating-point underflows may occur occasionally, but can usually be ignored.

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 Feasibility tolerance. If the message recurs, the offending linearly dependent constraint (with index "j") must be removed from the problem. If a warning message did not precede the fatal overflow, contact the authors.

$\mathtt{inform} = 3$ The problem appears to have no feasible point. Check that there are no conflicting constraints, such as $x_1 \geq 1$, $x_2 \geq 2$ and $x_1 + x_2 = 0$. If the data for the constraints are accurate to the absolute precision σ, make sure that the Feasibility tolerance is *greater* than σ. For example, if all elements of A are of order unity and are accurate to only three decimal places, the Feasibility tolerance should be at least 10^{-3}.

$\mathtt{inform} = 4$ One of the iteration limits may be too small. (See Feasibility Phase and Optimality Phase.) Increase the appropriate limit and rerun qpopt.

$\mathtt{inform} = 5$ The Maximum Degrees of Freedom is too small. Rerun qpopt with a larger value (possibly using the warm start facility to specify the initial working set).

$\mathtt{inform} = 6$ An input parameter is invalid. The printed output will indicate which parameter(s) must be redefined. Rerun with corrected values.

$\mathtt{inform} = 7$ The specified problem type was not FP, LP, QP1, QP2, QP3, or QP4. Rerun qpopt with Problem type set to one of these values.

9.   Example

This section describes an example QP problem, and shows how the Hessian may be coded implicitly. It then gives a main program that calls `qpopt` (twice) to solve the problem, and the Print files that are generated. The Summary file is shown in Section 7.

9.1.   Definition of the example problem

The example problem is an indefinite quadratic program (see [BK80]). It has eight variables and seven general constraints. The vector c and the Hessian H are given by

$$
c = \begin{pmatrix} 7 \\ 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \\ 0 \end{pmatrix}
\quad \text{and} \quad
H = \begin{pmatrix}
1.69 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
1 & 1.69 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 1 & 1.69 & 1 & 2 & 3 & 4 & 5 \\
3 & 2 & 1 & 1.69 & 1 & 2 & 3 & 4 \\
4 & 3 & 2 & 1 & 1.69 & 1 & 2 & 3 \\
5 & 4 & 3 & 2 & 1 & 1.69 & 1 & 2 \\
6 & 5 & 4 & 3 & 2 & 1 & 1.69 & 1 \\
7 & 6 & 5 & 4 & 3 & 2 & 1 & 1.69
\end{pmatrix}.
$$

The general constraint matrix A and bound vectors ℓ and u are

$$
\ell = \begin{pmatrix}
-1.0 \\ -2.1 \\ -3.2 \\ -4.3 \\ -5.4 \\ -6.5 \\ -7.6 \\ -8.7 \\ -1.0 \\ -1.05 \\ -1.1 \\ -1.15 \\ -1.2 \\ -1.25 \\ -1.3
\end{pmatrix}, \quad
A = \begin{pmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{pmatrix}, \quad
u = \begin{pmatrix}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ \infty \\ \infty \\ \infty \\ \infty \\ \infty \\ \infty \\ \infty
\end{pmatrix},
$$

and the starting point (which is infeasible) is

$$
x_0 = \begin{pmatrix} -1 & -2 & -3 & -4 & -5 & -6 & -7 & -8 \end{pmatrix}^T.
$$

Three local minimizers are (to five figures)

$$
x_1^* = \begin{pmatrix} -1 \\ -2 \\ -3.05 \\ -4.15 \\ -5.3 \\ 6 \\ 7 \\ 8 \end{pmatrix}, \quad
x_2^* = \begin{pmatrix} -1 \\ -2.1 \\ -3.15 \\ -4.25 \\ -5.4 \\ 6 \\ 7 \\ 8 \end{pmatrix} \quad \text{and} \quad
x_3^* = \begin{pmatrix} 1 \\ 2 \\ 1.880144 \\ .780144 \\ -.369856 \\ -1.569856 \\ -2.819856 \\ -4.119856 \end{pmatrix}.
$$

9.2.   Implicit definition of H for the example problem

In the example main program, the problem is first solved with H defined explicitly using the default version of qpHess. The problem is then solved again with H defined implicitly by the following subroutine qpHes1. The name qpHes1 is passed as a parameter to qpopt.

```
      subroutine qpHes1( n, ldH, jthcol, H, x, Hx, iw, leniw, w, lenw )

      implicit           double precision(a-h,o-z)

      integer            iw(leniw)
      double precision   H(ldH,*), Hx(n), x(n)
      double precision   w(lenw)

*     ================================================================
*     qpHes1    computes the vector  Hx = (H)*x  for some matrix H
*     that defines the Hessian of the required QP problem.
*
*     In this version of  qpHess  the Hessian matrix is implicit.
*     The array  H  is not accessed.  There is no special coding
*     for the case jthcol .gt. 0.
*     ================================================================
      do 200, i = 1, n
         sum = 1.69d+0*x(i)
         do 100, j = 1, n
            sum = sum + dble( abs(i-j) )*x(j)
  100    continue
         Hx(i) = sum
  200 continue

*     end of qpHes1
      end
```

9.3.   Main program for the example problem

```
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*
*     File qpmain.f
*
*     Sample program for QPOPT Version 1.0-10   Sept 1995.
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      program           qpmain
      implicit          double precision (a-h,o-z)

*     Set the declared array dimensions.
*     ldH   = the declared row dimension of H.
*     ldA   = the declared row dimension of A.
*     maxn  = maximum no. of variables allowed for.
*     maxbnd = maximum no. of variables + linear constraints.
*     leniw = the length of the integer work array.
*     lenw  = the length of the double precision work array.

      parameter        ( ldH   =   8,   ldA  =   7,
     $                   maxn  =   8,
     $                   leniw =  20,   lenw = 500,
     $                   maxbnd = maxn + ldA )

      integer           istate(maxbnd)
      integer           iw(leniw)
      double precision  H(ldH,maxn)
      double precision  bl(maxbnd), bu(maxbnd), clamda(maxbnd)
      double precision  cvec(maxn)
      double precision  A(ldA,maxn), Ax(ldA), x(maxn)
      double precision  w(lenw)
      external          qpHess, qpHes1

      double precision  bigbnd
      character*20      lFile
      logical           byname, byunit

      parameter        ( point1 = 0.1d+0, zero = 0.0d+0, one = 1.0d+0 )

*     ----------------------------------------------------------------
*     Assign file numbers and open files by various means.
*     (Some systems don't need explicit open statements.)
*     iOptns = unit number for the Options file.
*     iPrint = unit number for the Print file.
*     iSumm  = unit number for the Summary file.
*     ----------------------------------------------------------------
      iOptns = 4
      iPrint = 10
      iSumm  = 6
      byname = .true.
      byunit = .false.
```

```
         if ( byname ) then
            lFile = 'qpopt.opt'
            open( iOptns, file=lFile, status='OLD',     err=800 )

            lFile = 'qpopt.out'
            open( iPrint, file=lFile, status='UNKNOWN', err=800 )

         else if ( byunit ) then
            lUnit = iOptns
            open( lUnit, status='OLD',      err=900 )

            lUnit = iPrint
            open( lUnit, status='UNKNOWN', err=900 )
         end if

*     ==================================================================
*     Set the actual problem dimensions.
*     n     = the number of variables.
*     nclin = the number of general linear constraints (may be 0).
*     bigbnd = the Infinite Bound size.
*     ==================================================================
      n      = 8
      nclin  = 7
      bigbnd = 1.0d+21

*     ------------------------------------------------------------------
*     Define H, A, bl, bu, cvec and the initial x.
*     This example is due to Bunch and Kaufman,
*     'A computational method for the indefinite quadratic programming
*     problem', Linear Algebra and its Applications, 34, 341-370 (1980).
*     ------------------------------------------------------------------
      do 200, j = 1, n
         do 120, i = 1, nclin
            A(i,j) = zero
  120    continue

         do 150, i = 1, j-1
            H(i,j) = abs(i - j)
  150    continue

         H(j,j)  = 1.69d+0
         bl(j)   = - j - point1*dble(j - 1)
         bu(j)   =   j
         cvec(j) =   dble(8 - j)
         x(j)    = - dble(j)
  200 continue

      do 220, i = 1, nclin
         A(i,i)    = - one
         A(i,i+1)  =   one
         bl(n+i)   = - one - 0.05d+0*dble(i - 1)
         bu(n+i)   =   bigbnd
  220 continue
```

```
*      ------------------------------------------------------------------
*      Set a few options in-line.
*      The Print file   will be on unit iPrint.
*      The Summary file will be on the default unit 6
*      (typically the screen).
*      ------------------------------------------------------------------
       call qpprmi( 'Print file          =', iPrint )
       call qpprmr( 'Infinite Bound size =', bigbnd )

*      Read the Options file.

       call qpprms( iOptns, inform )
       if (inform .ne. 0) then
          write(iPrint, 3000) inform
          stop
       end if


*      ------------------------------------------------------------------
*      Solve the QP problem.
*      ------------------------------------------------------------------
       call qpopt ( n, nclin, ldA, ldH,
      $             A, bl, bu, cvec, H, qpHess,
      $             istate, x,
      $             inform, iter, obj, Ax, clamda,
      $             iw, leniw, w, lenw )

*      Test for an error condition.

       if (inform .gt. 1) go to 999


*      ==================================================================
*      Re-solve the problem with the Hessian defined by a subroutine.
*      ==================================================================

*      Set some new options in-line,
*      but stop listing them on the Print file.

       call qpprm ( 'Nolist'                         )
       call qpprm ( 'Problem Type  QP2'              )
       call qpprm ( 'Feasibility Tolerance  =  1.0e-10' )
       call qpprmr( 'Optimality tolerance   ', 1.0d-5   )
       call qpprmi( 'Print level            ',     10  )
```

```
*     ----------------------------------------------------------------
*     Define a new starting point.
*     ----------------------------------------------------------------
      x(1) =  -1.0
      x(2) =  12.0
      x(3) =  -3.0
      x(4) =  14.0
      x(5) =  -5.0
      x(6) =  16.0
      x(7) =  -7.0
      x(8) =  18.0

      call qpopt ( n, nclin, ldA, ldH,
     $             A, bl, bu, cvec, H, qpHes1,
     $             istate, x,
     $             inform, iter, obj, Ax, clamda,
     $             iw, leniw, w, lenw )

      if (inform .gt. 1) go to 999
      stop

*     ----------------------------------------------------------------
*     Error conditions.
*     ----------------------------------------------------------------
  800 write(iSumm , 4000) 'Error while opening file', lFile
      stop

  900 write(iSumm , 4010) 'Error while opening unit', lUnit
      stop

  999 write(iPrint, 3010) inform
      stop


 3000 format(/ '  QPPRMS terminated with  inform =', i3)
 3010 format(/ '  QPOPT   terminated with  inform =', i3)
 4000 format(/  a, 2x, a  )
 4010 format(/  a, 2x, i6 )

*     end of the example program for QPOPT
      end
```

9.4.   Print file from the example problem

```
Optional Parameters
-------------------

    Print file         =              10
    Infinite Bound size =  1.00000000E+21
    Begin optional parameters read from a file
       Optimality  phase iterations     50
       Feasibility phase iterations     50
       Print level                       5
    End  of optional parameters read from a file




                    QPOPT  ---  Version 1.0-10    Sep  1995
                    =========================================


Parameters
----------


Problem type...........      QP2
Linear constraints.....        7   Cold start.............            Min. Sum of Infeas.....       No
Variables..............        8   Infinite bound size....  1.00E+21   Feasibility tolerance..  1.05E-08
Hessian rows...........        8   Infinite step size.....  1.00E+21   Optimality tolerance...  1.72E-13
Check frequency........       50   Expand frequency.......         5   Crash tolerance........  1.00E-02
Max degrees of freedom.        8   Max active constraints.         7   Rank tolerance.........  1.11E-14
Max free variables.....        8

Print level............        5   Print file.............        10   Feasibility phase itns.       50
Unit round-off.........  1.11E-16   Summary file...........         6   Optimality  phase itns.       50

Workspace provided is    iw(      20), w(      500).
To solve problem we need  iw(      19), w(      227).


Itn Jdel  Jadd     Step Ninf  Sinf/Objective Norm gZ   Zr  Art  Bnd  Lin NOpt     Min Lm  Cond T  Cond Rz      Rzz
  0    0     0  0.0E+00    0  0.00000000E+00 0.0E+00    0    6    1    1                     1.E+00
Itn    0 -- Feasible point found.
  0    0     0  0.0E+00    0  1.51638000E+03 9.8E+01    1    5    1    1                     1.E+00  1.0E+00
  1    0    8U 2.8E-01    0  1.72380000E+02 0.0E+00    0    5    2    1                     1.E+00
  2   1L   10L 3.1E-03    0  1.68083225E+02 0.0E+00    0    5    1    2     7 -8.61E+01  1.E+00
  3   5A   11L 1.2E-02    0  1.57176475E+02 0.0E+00    0    4    1    3     5  6.33E+01  1.E+00
  4   4A   12L 3.2E-02    0  1.38528925E+02 0.0E+00    0    3    1    4     4  6.32E+01  1.E+00
  5   3A   13L 6.9E-02    0  1.11295925E+02 0.0E+00    0    2    1    5     3  6.31E+01  1.E+00
  6   2A   14L 1.3E-01    0  7.41228000E+01 0.0E+00    0    1    1    6     3  6.48E+01  1.E+00
  7   1A    1U 8.4E-01    0 -5.85162625E+01 0.0E+00    0    0    2    6     4  6.94E+01  1.E+00
  8  13L     0 1.0E+00    0 -8.72144740E+01 1.3E-15    1    0    2    5     6 -1.76E+01  1.E+00  1.0E+00
  9   1U    6U 2.5E+00    0 -3.12744888E+02 1.4E+02    1    0    2    5     2  1.03E+02  1.E+00  1.0E+00
 10    0    1L 1.4E-01    0 -5.62265012E+02 0.0E+00    0    0    3    5                     1.E+00
 11  14L    7U 1.3E-01    0 -6.21487825E+02 0.0E+00    0    0    4    4     1 -2.82E+01  1.E+00


Exit QPOPT - Optimal QP solution.


Final QP objective value =    -621.4878


Exit from QP problem after   11 iterations.  Inform =  0
```

```
                        QPOPT  ---  Version 1.0-10     Sep  1995
                        =========================================


Parameters
----------


Problem type..........     QP2
Linear constraints.....      7   Cold start.............            Min. Sum of Infeas.....      No
Variables..............      8   Infinite bound size....  1.00E+21  Feasibility tolerance..  1.00E-10
Hessian rows...........      8   Infinite step size.....  1.00E+21  Optimality tolerance...  1.00E-05
Check frequency........     50   Expand frequency.......         5  Crash tolerance........  1.00E-02
Max degrees of freedom.      8   Max active constraints.         7  Rank tolerance.........  1.11E-14
Max free variables.....      8

Print level............     10   Print file.............        10  Feasibility phase itns.      50
Unit round-off........  1.11E-16 Summary file...........         6  Optimality  phase itns.      50

Workspace provided is   iw(     20), w(    500).
To solve problem we need  iw(     19), w(    227).


Itn Jdel  Jadd    Step Ninf  Sinf/Objective Norm gZ  Zr  Art  Bnd  Lin NOpt    Min Lm  Cond T  Cond Rz     Rzz
  0   0     0  0.0E+00    3  2.35500000E+01 1.7E+00   0    3    5    0                   1.E+00
  1  2U   10L  4.0E+00    2  1.96000000E+01 1.4E+00   0    3    4    1        1.00E+00   1.E+00
  2  4U   12L  7.8E+00    1  1.17500000E+01 1.0E+00   0    3    3    2        1.00E+00   1.E+00
  3  6U   14L  1.2E+01    0  0.00000000E+00 0.0E+00   0    3    2    3        1.00E+00   1.E+00
Itn    3 -- Feasible point found.
  3   0     0  0.0E+00    0  8.66526437E+02 1.5E+02   1    2    2    3                   1.E+00  1.0E+00
  4   0     9L  1.0E-01    0  4.98244375E+01 0.0E+00   0    2    2    4                   2.E+00
  5  2A   11L  4.5E-01    0 -5.62265013E+02 0.0E+00   0    1    2    5   5 -6.86E+01      2.E+00
  6  1A    6U  5.7E-13    0 -5.62265013E+02 0.0E+00   0    0    3    5   2 -2.20E+01      2.E+00
  7 14L    7U  1.3E-01    0 -6.21487825E+02 0.0E+00   0    0    4    4   1 -2.82E+01      2.E+00


Variable          State     Value      Lower bound    Upper bound    Lagr multiplier      Slack

variable     1    LL    -1.000000      -1.000000       1.000000        304.4550           .
variable     2    FR    -2.000000      -2.100000       2.000000            .               0.1000
variable     3    FR    -3.050000      -3.200000       3.000000            .               0.1500
variable     4    FR    -4.150000      -4.300000       4.000000            .               0.1500
variable     5    FR    -5.300000      -5.400000       5.000000            .               0.1000
variable     6    UL     6.000000      -6.500000       6.000000       -0.6100000           .
variable     7    UL     7.000000      -7.600000       7.000000       -24.42000            .
variable     8    UL     8.000000      -8.700000       8.000000       -34.23000            .


Linear constrnt  State     Value      Lower bound    Upper bound    Lagr multiplier      Slack

lincon       1    LL    -1.000000      -1.000000          None         212.8950           .
lincon       2    LL    -1.050000      -1.050000          None         131.5250           0.2220E-15
lincon       3    LL    -1.100000      -1.100000          None          64.42950         -0.4441E-15
lincon       4    LL    -1.150000      -1.150000          None          17.79300         -0.4441E-15
lincon       5    FR    11.30000       -1.200000          None             .              12.50
lincon       6    FR     1.000000      -1.250000          None             .               2.250
lincon       7    FR     1.000000      -1.300000          None             .               2.300


Exit QPOPT - Optimal QP solution.


Final QP objective value =   -621.4878


Exit from QP problem after    7 iterations.  Inform =  0
```

Acknowledgement

We are grateful to Alan Brown of NAG Ltd for many helpful comments on earlier versions of QPOPT.

References

[BK80]   J. R. Bunch and L. Kaufman. A computational method for the indefinite quadratic programming problem. *Linear Algebra Appl.*, 34, 341–370, 1980.

[GHM+86]   Philip E. Gill, S. J. Hammarling, Walter Murray, M. A. Saunders, and Margaret H. Wright. User's guide for LSSOL (Version 1.0): a Fortran package for constrained linear least-squares and convex quadratic programming. Report SOL 86-1, Department of Operations Research, Stanford University, Stanford, CA, 1986.

[GM78]   Philip E. Gill and Walter Murray. Numerically stable methods for quadratic programming. *Math. Program.*, 14, 349–372, 1978.

[GMSW84]   Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Trans. Math. Softw.*, 10, 282–298, 1984.

[GMSW86]   Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for NPSOL (Version 4.0): a Fortran package for nonlinear programming. Report SOL 86-2, Department of Operations Research, Stanford University, Stanford, CA, 1986.

[GMSW89]   Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. A practical anti-cycling procedure for linearly constrained optimization. *Math. Program.*, 45, 437–474, 1989.

[GMSW91]   Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Inertia-controlling methods for general quadratic programming. *SIAM Rev.*, 33(1), 1–36, 1991.

[PS88]   P. M. Pardalos and G. Schnitger. Checking local optimality in constrained quadratic programming is NP-hard. *Operations Research Letters*, 7, 33–35, 1988.