SIAM J. OPTIM.
Vol. 9, No. 1, pp. 62–83

A FINITE CONTINUATION ALGORITHM FOR BOUND CONSTRAINED QUADRATIC PROGRAMMING*

KAJ MADSEN†, HANS BRUUN NIELSEN†, AND MUSTAFA ÇELEBI PINAR‡

Abstract. The dual of the strictly convex quadratic programming problem with unit bounds is posed as a linear ℓ_1 minimization problem with quadratic terms. A smooth approximation to the linear ℓ_1 function is used to obtain a parametric family of piecewise-quadratic approximation problems. The unique path generated by the minimizers of these problems yields the solution to the original problem for finite values of the approximation parameter. Thus, a finite continuation algorithm is designed. Results of extensive computational experiments are reported.

Key words. bound constrained quadratic programming, Lagrangian duality, linear ℓ_1 estimation, Huber's M-estimator, robust regression

AMS subject classifications. 90C20, 65K05, 65U05, 65F20

PII. S1052623495297820

1. Introduction.

We consider the strictly convex quadratic programming problem (QP) with unit bounds:

[BCQP]

$$\begin{aligned} \min_y \quad & H(y) = -d^T y + \tfrac{1}{2} y^T Q y \\ \text{subject to} \quad & -\mathbf{1} \le y \le \mathbf{1}, \end{aligned}$$

where Q is an $m \times m$ symmetric, positive definite matrix, and y and d are m-vectors.

In this paper we study a dual continuation algorithm for the solution of [BCQP]. We first show that the dual of [BCQP] is an unconstrained minimization problem, where the function is composed of a linear ℓ_1 term and strictly convex quadratic terms. This nondifferentiable function is approximated by a smooth piecewise linear-quadratic Huber function. The resulting smooth problems yield a unique path that converges to the primal-dual optimal solutions. We follow the path using a continuation algorithm based on Newton's method. This algorithm is inspired by our earlier work on linear programming with unit bounds [11]. In this reference, the dual of a linear program is formulated as an ℓ_1 minimization problem. We solve the dual problem using a continuation algorithm based on the piecewise-linear paths generated by a smooth approximation problem. The smooth problem comes from robust statistics, where it was used by Huber as an alternative to the least squares estimation [7]. The most important property of the smooth problems is that they yield primal-dual optimal solutions for sufficiently small values of a continuation parameter. This allows a new finite, numerically stable continuation algorithm for linear programming.

We apply a similar philosophy here to the dual of [BCQP]. We approximate the ℓ_1 term by a Huber function term. This yields a family of problems parameterized by a smoothing parameter γ. This parameter is alternatively referred to as a continuation

*Received by the editors December 3, 1995; accepted for publication (in revised form) February 9, 1998; published electronically October 30, 1998. This research was supported in part by Danish Natural Sciences Research Council grant 11-0505.

http://www.siam.org/journals/siopt/9-1/29782.html

†Institute of Mathematical Modelling, Technical University of Denmark, 2800, Lyngby, Denmark (km@imm.dtu.dk, hbn@imm.dtu.dk).

‡Department of Industrial Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey (mustafap@bilkent.edu.tr).

parameter as in the linear programming case. However, unlike the linear programming case, the path generated by the minimizers of the smooth problem is unique and is no longer piecewise linear. This requires a fresh look at the properties of the path and its behavior for sufficiently small values of the continuation parameter, that is, the analysis of [11] does not apply here. However, we are able to establish that primal-dual optimal solutions are obtained from the path for positive, sufficiently small values of the parameter.

The following properties of the approximation are emphasized as the main contributions of this paper:

P0. The primal-dual minimizers of the smooth problem define a unique path as a function of the smoothing parameter γ.

P1. The primal-dual optimal solutions to [BCQP] are obtained for sufficiently small $\gamma > 0$ using information from the path, that is, γ does not have to be decreased to zero in order to obtain an exact solution to the QP problem (Theorems 2.2 and 2.3).

P2. Although the unique path leading to the primal-dual solutions is nonlinear, a powerful extrapolation result allows computation of primal-dual candidates for optimality (Theorem 2.2).

Furthermore, our main results are obtained without any nondegeneracy assumptions on the problem. In particular, Theorem 2.2 (the description of the extrapolation) and Theorem 2.3 (the behavior of the path for small values of the continuation parameter) are established in the absence of any restrictive assumptions.

These properties suggest an algorithm to trace the path to arrive at a solution of [BCQP]. We refer to the path as the "solution path" throughout the rest of the paper. Our algorithm is best interpreted as a continuation algorithm since it possesses the following main features of continuation algorithms.

1. The solution of a parametrized family of subproblems as a parameter varies over an interval; in our case, the smooth "Huber" problem as a function of the smoothing parameter γ.
2. The use of a local iterative method to solve the subproblems. We use a finite Newton method [10] to solve the smooth Huber problem.
3. The use of an extrapolation technique to guess an optimal primal-dual pair from a point on the path.

As a result of P1 and P2 above, the continuation algorithm is a finite procedure provided that γ is decreased by at least a certain factor after each unconstrained minimization. We make these ideas precise in the forthcoming sections.

In this algorithm, Newton's method is used to locate the path for some value of the smoothing parameter. Unless optimality is reached, Newton's method is invoked for a reduced value of the parameter from a point no longer on the path, and the cycle is repeated. We summarize the algorithmic scheme as follows:

```
Compute initial γ
repeat
    compute a solution of the approximation problem
    decrease γ
until optimality.
```

This scheme closely relates our algorithm to penalty and barrier methods and in general to path-following methods. To the best of our knowledge, from this perspective, both the theoretical analysis of section 2 and the algorithm stand as novel contributions to the quadratic programming literature.

We develop a numerically stable implementation of the new algorithm for dense problems. We also compare the performance of the algorithm to LSSOL, a software system for quadratic programming from Stanford University's Systems Optimization Laboratory, and to an interior point algorithm of Han, Pardalos, and Ye [6].

For a review of the literature on quadratic programming we refer the reader to the paper by Moré and Toraldo [14]. It seems that currently the fastest algorithms for [BCQP] are the active set methods [14]. For problem [BCQP], active set methods can efficiently add or delete many constraints from the active set at one iteration. Primal-dual interior point algorithms have also been recently developed for [BCQP] [6]. Other related ideas have been proposed in more recent papers by Coleman and Hulbert [1] and Li and Swetits [8, 9]. In [1] Coleman and Hulbert reformulate [BCQP] as an unconstrained minimization problem involving an ℓ_1 term. This reformulation is obtained by manipulating the Karush–Kuhn–Tucker conditions of [BCQP]. They apply a superlinearly convergent modified Newton method to this reformulation. In this regard our point of departure is identical to that of [1]. Li and Swetits [8, 9] reformulate the convex quadratic programming problem as an unconstrained minimization of a convex quadratic spline function.

In the rest of the paper we proceed as follows. In section 2 we present a simple derivation of the dual problem, and we explore the relation of the nondifferentiable dual to the approximation problem. We give the details and analysis of Newton's method applied to the approximation problem in section 3. In section 4 we discuss some implementation details and generation of test problems, and we report the results of extensive computational experiments with the new algorithm. Comparisons to competing algorithms are also made. Concluding remarks are offered in section 5.

2. A nondifferentiable dual problem and its approximation. We begin our study of [BCQP] by deriving a dual problem. Since Q is symmetric positive definite, there exists a full rank matrix $A \in \Re^{m\times m}$ such that $Q = A^TA$. Then the quadratic program can be rewritten

$$\begin{aligned} \min_{y} \quad & -(A^Tb)^Ty + \frac{1}{2}y^TA^TAy \\ \text{subject to} \quad & -\mathbf{1} \le y \le \mathbf{1} \end{aligned}$$

for some $b \in \Re^m$ such that $d = A^Tb$. Let $u = Ay$ and rewrite the problem as

$$\begin{aligned} \min_{y,u} \quad & -b^Tu + \frac{1}{2}u^Tu \\ \text{subject to} \quad & Ay = u \\ & -\mathbf{1} \le y \le \mathbf{1}. \end{aligned}$$

Associating dual multipliers $x \in \Re^m$ with the equality constraints, we form the following Lagrangian max-min problem:

$$\max_x \min_{u,-\mathbf{1}\le y\le\mathbf{1}} \left\{ \frac{1}{2}u^Tu - b^Tu + x^T(Ay - u) \right\},$$

which is equivalent to

$$\max_x \left\{ \min_u \left\{ \frac{1}{2}u^Tu - b^Tu - u^Tx \right\} + \left\{ \min_{-\mathbf{1}\le y\le\mathbf{1}} x^TAy \right\} \right\}.$$

It is easy to see that the first minimization yields the identity

$$Ay = x + b. \tag{2.1}$$

Hence, we get the term

$$-\frac{1}{2}x^T x - b^T x - \frac{1}{2}b^T b.$$

The second minimization over y is also straightforward and yields

$$\min_{-\mathbf{1}\le y_i\le \mathbf{1}} x_i(Ay)_i = \begin{cases} (A^T x)_i & \text{if } (A^T x)_i \le 0, \\ -(A^T x)_i & \text{if } (A^T x)_i \ge 0. \end{cases}$$

However, this is simply the negative of the ℓ_1-norm of $A^T x$. Therefore, our dual problem is

$$\text{minimize} F(x) \;\equiv\; \|A^T x\|_1 + \frac{1}{2}x^T x + b^T x + \frac{1}{2}b^T b. \tag{2.2}$$

As a result of strict convexity, the primal and dual optimal solutions are unique.

Let

$$r(x) = A^T x. \tag{2.3}$$

From the derivation, the conditions for (y_0, x_0) to be optimal can be expressed as

$$\begin{aligned} Ay_0 &= b + x_0\ , \\ r_i(x_0) > 0 \quad &\Longrightarrow \quad y_{0i} = -1, \\ r_i(x_0) < 0 \quad &\Longrightarrow \quad y_{0i} = 1, \\ -1 < y_{0i} < 1 \quad &\Longrightarrow \quad r_i(x_0) = 0, \end{aligned}$$

for all $i = 1, \ldots, m$. From this point on, we use (y_0, x_0) to denote a primal-dual optimal pair.

Let us define a set $\hat{S}$ of "sign vectors" such that $\hat{S} = \{s \in \Re^m \mid s_i \in \{-1, 0, 1\}\}$. Now, define the sign vector $s_0(x)$ such that

$$s_{0i}(x) = \begin{cases} -1 & \text{if } r_i(x) < 0, \\ 0 & \text{if } r_i(x) = 0, \\ 1 & \text{if } r_i(x) > 0, \end{cases} \tag{2.4}$$

and define

$$W_0 = \text{diag}(w_1, \ldots, w_m) \quad \text{with} \quad w_i = 1 - s_{0i}^2. \tag{2.5}$$

Let $s_0 = s_0(x_0)$ and let W_0 be derived from s_0 using (2.5). Now, we can compactly express the optimality conditions as

$$AW_0 y_0 - As_0 = b + x_0. \tag{2.6}$$

Since A has full rank, this implies that the following linear system is consistent:

$$(AW_0 A^T)h = As_0 + b + x_0. \tag{2.7}$$

Since the null space $\mathcal{N}(AW_0A^T)$ coincides with the null space $\mathcal{N}(W_0A^T)$, W_0A^Th is constant no matter which solution h to (2.7) is picked.

2.1. The smooth Huber approximation. Consider the function $\phi : \Re \mapsto \Re$:

$$\phi_\gamma(t) = \begin{cases} \frac{1}{2\gamma}t^2 & \text{if } |t| \le \gamma, \\ |t| - \frac{1}{2}\gamma & \text{if } |t| > \gamma, \end{cases} \tag{2.8}$$

for some scalar parameter $\gamma > 0$. This function is known as Huber's M-estimator function in robust statistics. Now, we replace (2.2) by the following differentiable problem:

$$\min_x \Phi_\gamma(x) + \frac{1}{2}x^T x + b^T x + \frac{1}{2}b^T b, \tag{2.9}$$

where

$$\Phi_\gamma(x) = \sum_{i=1}^{m} \phi_\gamma(r_i(x)). \tag{2.10}$$

We discuss some well-known properties of this function in section 3.1. To view this problem in quadratic programming format, we define a new sign vector s_γ:

$$s_\gamma(x) = [s_{\gamma 1}(x), \ldots, s_{\gamma m}(x)] \quad \text{with} \quad s_{\gamma i}(x) = \begin{cases} -1 & \text{if } r_i(x) < -\gamma, \\ 0 & \text{if } |r_i(x)| \le \gamma, \\ 1 & \text{if } r_i(x) > \gamma, \end{cases} \tag{2.11}$$

and define

$$W_s = \text{diag}(w_1, \ldots, w_m) \quad \text{with} \quad w_i = 1 - s_{\gamma i}^2. \tag{2.12}$$

Therefore, we have the following minimization problem:

$$\text{minimize } F_\gamma(x) \;\equiv\; \frac{1}{2\gamma}r^T W_s r + s_\gamma^T \left[r - \frac{1}{2}\gamma s_\gamma\right] + \frac{1}{2}x^T x + b^T x + \frac{1}{2}b^T b, \tag{2.13}$$

where the argument x of r and $\mathbf{s}_\gamma$ is dropped for notational convenience. We refer to the above problem as the "Huber problem" for ease of expression. Clearly, this problem has a unique minimizer as a result of strict convexity. In the following, we use the notations x_γ for the minimizer of F_γ and $W_\gamma = W_s$, where $s = s_\gamma(x_\gamma)$. For notational convenience, we use W_γ and W_s interchangeably in our analysis when the meaning is clear from the context.

It can be shown using Lagrangian duality that the dual problem to (2.13) is given by
[PBCQP]

$$\begin{aligned} \min_y \quad & H(y) = -d^T y + \frac{1}{2}y^T(Q + \gamma I)y \\ \text{subject to} \quad & -\mathbf{1} \le y \le \mathbf{1}. \end{aligned}$$

We notice that the above problem is simply a quadratically perturbed version of [BCQP]. This relates our analysis to previous studies by Mangasarian [12] and Mangasarian and Meyer [13], where quadratic and nonlinear perturbations of linear programs were addressed.

2.2. The relation between F, F_γ, and [BCQP]. In this section we establish some important properties of the Huber approximation. These properties characterize the proposed algorithm and are used to verify finite convergence.

We begin with some simple results. We can immediately observe the following elementary fact:

$$\lim_{\gamma \to 0} \phi_\gamma(t) = |t| \ , \tag{2.14}$$

for any $t \in \Re$. Now, we have the following simple result.

LEMMA 2.1. *Let x_γ denote the minimizer of the function F_γ. Then,*

$$0 \leq F(x_0) - F_\gamma(x_\gamma) \leq m\frac{\gamma}{2}. \tag{2.15}$$

Proof. From the definitions of F and F_γ, we have for any $x \in \Re^m$

$$0 \leq F(x) - F_\gamma(x) \leq m\frac{\gamma}{2}.$$

Since x_0 and x_γ are minimizers of F and F_γ, we therefore obtain

$$F_\gamma(x_\gamma) \leq F_\gamma(x_0) \leq F(x_0)$$

and

$$F(x_0) - m\frac{\gamma}{2} \leq F_\gamma(x_\gamma) - m\frac{\gamma}{2} \leq F_\gamma(x_\gamma).$$

This proves (2.15). ∎

THEOREM 2.1. *Let x_γ denote the minimizer of the function F_γ. Then,*

$$\lim_{\gamma \to 0} x_\gamma = x_0. \tag{2.16}$$

Proof. Since the functions are continuous and strictly convex, i.e., the minimizers are unique, the result follows using (2.14) and (2.15). ∎

Let $s = s_\gamma(x_\gamma)$. The minimizer x_γ of F_γ satisfies the following necessary condition:

$$A\left[\frac{1}{\gamma}W_s r(x_\gamma) + s\right] + b + x_\gamma = \mathbf{0}, \tag{2.17}$$

which may be written in the form

$$\big(AW_sA^T + \gamma I\big)x_\gamma = -\gamma\big(As + b\big), \tag{2.18}$$

or as

$$Ay_\gamma = b + x_\gamma, \tag{2.19}$$

where we have defined

$$y_\gamma = -\left(\frac{1}{\gamma}W_\gamma r(x_\gamma) + s\right). \tag{2.20}$$

Using (2.17) we see that y_γ is feasible in [BCQP] and optimal in [PBCQP]. Clearly, using (2.1), (2.16), and (2.19) we have

$$\lim_{\gamma \to 0} y_\gamma = y_0. \tag{2.21}$$

In the remainder of this section, we study the behavior of the solution paths $\{x_\gamma\}$ and $\{y_\gamma\}$ as $\gamma \searrow 0$. For fixed s (and therefore W_s) we introduce the singular value decomposition (SVD) of the matrix $W_s A^T$:

$$W_s A^T = U \Sigma V^T. \tag{2.22}$$

Here, the matrices U and V with columns $\{u_j\}_{j=1}^m$ and $\{v_j\}_{j=1}^m$ are orthogonal, and the singular values are given in Σ:

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_m) \quad \text{with} \quad \sigma_1 \geq \cdots \geq \sigma_q > 0, \quad \sigma_{q+1} = \cdots = \sigma_m = 0. \tag{2.23}$$

The number q is the rank of the matrix $W_s A^T$, the vectors $\{u_j\}_{j=1}^q$ and $\{v_j\}_{j=1}^q$ form an orthonormal basis of the range of $W_s A^T$ and $A W_s A^T$, respectively, and $\{v_j\}_{j=1}^m$ is an orthonormal basis of $\Re^m$. This means that we can write

$$As + b = \sum_{j=1}^m \alpha_j v_j = V\alpha, \tag{2.24}$$

and by inserting (2.22) into (2.18) we get

$$\big(V\Sigma^2 V^T + \gamma I\big) x_\gamma = -\gamma V \alpha,$$

from which we find

$$x_\gamma \;=\; -\gamma \sum_{j=1}^m \frac{\alpha_j}{\sigma_j^2 + \gamma}\, v_j \;=\; -\gamma \sum_{j=1}^q \frac{\alpha_j}{\sigma_j^2 + \gamma}\, v_j \;-\; \sum_{j=q+1}^m \alpha_j v_j. \tag{2.25}$$

Furthermore, from (2.20) and (2.22) we get

$$y_\gamma = \sum_{j=1}^q \frac{\sigma_j \alpha_j}{\sigma_j^2 + \gamma}\, u_j \;- s. \tag{2.26}$$

As we shall see in Theorem 2.3, $s_\gamma(x_\gamma)$ and therefore, W_s are constant for γ small enough. When the SVD factorization (2.22) corresponds to this W_s, it follows that

$$x_0 = \lim_{\gamma \to 0} x_\gamma = -\sum_{j=q+1}^m \alpha_j v_j \quad \text{and} \quad y_0 = \lim_{\gamma \to 0} y_\gamma = \sum_{j=1}^q \frac{\alpha_j}{\sigma_j}\, u_j \;- s. \tag{2.27}$$

In the algorithm of section 3 we do not compute the SVD, but the following theorem provides us with an extrapolation formula that is used in our algorithm to test for optimality. To the best of our knowledge, this is a new result in the path-following literature.

THEOREM 2.2. *Let x_δ be the minimizer of F_δ for $0 < \delta \leq \gamma$ with $s = s_\delta(x_\delta)$ and $W = W_s$. Assume that $s_\delta(x_\delta) = s$ for $0 < \delta \leq \gamma$. Then,*

$$x_0 = x_\delta + \delta d_\delta^{(\delta)} \quad \textit{and} \quad y_0 = W A^T d_\delta^{(0)} - s\ , \tag{2.28}$$

where $d_\delta^{(\delta)}$ and $d_\delta^{(0)}$ are the minimum-norm solutions to the linear systems

$$(AWA^T)d = As + b + x_\delta \quad \textit{and} \quad (AWA^T)d = As + b + x_0\ , \tag{2.29}$$

respectively.

Proof. From (2.24)–(2.25) we get

$$As + b + x_\delta \;=\; \sum_{j=1}^{q}\left(1 - \frac{\delta}{\sigma_j^2 + \delta}\right)\alpha_j v_j \;=\; \sum_{j=1}^{q}\frac{\sigma_j^2}{\sigma_j^2 + \delta}\,\alpha_j v_j \tag{2.30}$$

(the contributions for $j = q{+}1, \ldots, n$ cancel). Thus, the first of the rank-deficient systems in (2.29) is consistent, and the minimum-norm solution is

$$d_\delta^{(\delta)} = \sum_{j=1}^{q}\frac{\alpha_j}{\sigma_j^2 + \delta}\,v_j. \tag{2.31}$$

By adding $\delta d_\delta^{(\delta)}$ to x_δ (given by (2.25)) we get x_0, as expressed in (2.27). For the other system, we find

$$As + b + x_0 = \sum_{j=1}^{q}\alpha_j v_j. \tag{2.32}$$

Thus, the second system in (2.29) is also consistent. The minimum-norm solution is

$$d_\delta^{(0)} = \sum_{j=1}^{q}\frac{\alpha_j}{\sigma_j^2}\,v_j, \tag{2.33}$$

and by inserting this into (2.28) we get y_0 as expressed in (2.27). ∎

In general, let $(\hat{x}_0, \hat{y}_0)$ denote the quantities computed by (2.28). They provide practical termination criteria for the algorithm defined in section 3.

In Theorem 2.3 we show that $s_\gamma(x_\gamma)$ is constant when γ is small enough. For some of the components of s_γ this is almost trivial. The components which cause difficulty are those for which $r_i(x_0) = 0$ and $|y_{0i}| = 1$. This set is denoted by $\mathcal{D}$, and the set of sign vectors for which the "easy" components equal those of s_0 is denoted by $\mathcal{S}$. More precisely, $\mathcal{D}$ and $\mathcal{S}$ are defined as follows. Let $s \in \hat{S}$, $\kappa_s^+ = \{i : s_i = 1\}$, and $\kappa_s^- = \{i : s_i = -1\}$ with $\kappa_s = \kappa_s^+ \cup \kappa_s^-$ and $\kappa_s^0 = \{i : s_i = 0\}$. Let $\mathcal{D} = \{i : |y_{0i}| = 1\} \cap \kappa_s^0$ and $\mathcal{S} = \{s \in \hat{S} \mid s_i = s_{0i} \text{ for } i \notin \mathcal{D}\}$.

THEOREM 2.3. *Let $s_0 = s_0(x_0)$. There exists γ^* such that $s_\gamma(x_\gamma)$ is constant, with $\kappa_{s_0}^+ \subseteq \kappa_{s_\gamma}^+$, $\kappa_{s_0}^- \subseteq \kappa_{s_\gamma}^-$ for $0 < \gamma \le \gamma^*$.*

Proof. Since the number of different sign vectors is finite, there must exist a sequence of positive numbers $\gamma_1, \gamma_2, \ldots$, with $\gamma_k \searrow 0$ for $k \to \infty$ such that $s_\gamma(x_\gamma)$ is constant for $\gamma = \gamma_k$, $k = 1, 2, \ldots$. Denote this constant sign vector by s.

According to (2.3) and (2.11), the elements of s are defined by the values of $r_i(x_\gamma) = a_i^T x_\gamma$. Since $x_\gamma \to x_0$, we have $|a_i^T x_\gamma| > \gamma$ for $i \in \kappa_s^0$ and γ small enough. Furthermore, since $y_\gamma \to y_0$, we have from (2.20) that $|a_i^T x_\gamma|/\gamma < 1$ for $i \in \kappa_{s_0}^0 \setminus \mathcal{D}$, and γ small enough. Therefore, since $\gamma_k \searrow 0$, it must be the case that $s \in \mathcal{S}$.

Now, let $W = W_s$ and let (2.22) be the SVD factorization of WA^T. Furthermore, let d_γ be the solution to

$$(AWA^T + \gamma I)d_\gamma = As + b + x_0.$$

By inserting (2.32), we see that

$$d_\gamma = \sum_{j=1}^{q}\frac{\alpha_j}{\sigma_j^2 + \gamma}\,v_j. \tag{2.34}$$

We introduce

$$\psi_i(\gamma) \equiv a_i^T d_\gamma = \sum_{j=1}^{q} \frac{\alpha_j}{\sigma_j^2 + \gamma}\, a_i^T v_j$$

for $i = 1, 2, \ldots, m$. Since ψ_i is a rational function for $\gamma{>}0$, it can only have a finite number of oscillations as $\gamma{\to}0$, and hence there exists $\gamma_1^* > 0$ such that for each i

$$\begin{array}{ll} \text{either} & |\psi_i(\gamma)| > 1 \quad \text{for } 0 < \gamma \le \gamma_1^* \\ \text{or} & |\psi_i(\gamma)| \le 1 \quad \text{for } 0 < \gamma \le \gamma_1^*. \end{array}$$

If $i \notin \kappa_{s_0}$, then $r_i(x_0) = 0$ and

$$r_i(x_0 - \gamma d_\gamma) = -\gamma \psi_i(\gamma).$$

Hence, the ith component of $s_\gamma(x_0 - \gamma d_\gamma)$ is constant for $0 < \gamma \le \gamma_1^*$. Since d_γ is bounded (see (2.34)) the other components of $s_\gamma(x_0 - \gamma d_\gamma)$ must also be constant in some interval $0 < \gamma \le \gamma_2^*$. Therefore, $s_\gamma(x_0 - \gamma d_\gamma)$ is constant for $0 < \gamma \le \gamma_3^* \equiv \min\{\gamma_1^*, \gamma_2^*\}$.

Finally, let $\gamma = \gamma_k$, $\gamma_k \le \gamma_3^*$ denote a value for which $s_\gamma(x_\gamma) = s$. It follows from (2.25), (2.27), and (2.34) that the unique minimizer x_γ is equal to $x_0 - \gamma d_\gamma$. ∎

Notice that $\mathcal{S}$ may be a singleton, in which case it is possible to establish a stronger result. This depends on a certain nondegeneracy assumption stated below.

THEOREM 2.4. *Let x_0 be the minimizer of F with $s = s_0(x_0)$ and $W = W_s$. Assume there exists $\gamma_1 > 0$ such that the solution d_γ to the system*

$$(AWA^T + \gamma I)d = As + b + x_0 \tag{2.35}$$

has the property

$$\|WA^T d_\gamma\|_\infty \le 1 \quad \text{for } \gamma \in (0, \gamma_1]\ . \tag{2.36}$$

Then, there exists $\gamma^* > 0$ such that $s_\gamma(x_\gamma)$ is constant for $\gamma \in (0, \gamma^*]$. Furthermore, $s_\gamma(x_\gamma) = s$ for $\gamma \in (0, \gamma^*]$.

Proof. Let $\delta = \min\{|r_i(x_0)| : r_i(x_0) \ne 0\}$. Choose $\gamma_2 < \delta$ such that, for $0 < \gamma \le \gamma_2$,

$$r_i(x_0) - \gamma a_i^T d_\gamma > \gamma_2 \text{ for } i \in \kappa_s^+, \tag{2.37}$$

$$r_i(x_0) - \gamma a_i^T d_\gamma < -\gamma_2 \text{ for } i \in \kappa_s^-. \tag{2.38}$$

Using (2.36), $s_\gamma(x_0 - \gamma d_\gamma) = s(x_0)$. Now, from (2.35) and using the fact that $WA^T x_0 = 0$, we get

$$\begin{aligned} (AWA^T + \gamma I)(-\gamma d_\gamma) &= -\gamma(As + b + x_0), \\ (AWA^T + \gamma I)(-\gamma d_\gamma) &= -AWA^T x_0 - \gamma(As + b + x_0), \\ (AWA^T + \gamma I)(x_0 - \gamma d_\gamma) &= -\gamma(As + b). \end{aligned}$$

Hence, $x_0 - \gamma d_\gamma$ is the minimizer of F_γ, and the theorem is proved with $\gamma^* = \min\{\gamma_1, \gamma_2\}$. ∎

DEFINITION 2.1. *A primal-dual optimal pair (y, x) is nondegenerate if the following condition holds for each zero component $r_i(x)$ of $r(x)$:*

$$r_i(x) = 0 \quad \text{and} \quad -1 < y_i < 1\ . \tag{2.39}$$

COROLLARY 2.1. *Let (y_0, x_0) be a nondegenerate primal-dual optimal pair for [BCQP] with $s = s_0(x_0)$ and $W = W_0(x_0)$. Then, there exists $\gamma^* > 0$ with $\gamma^* < \min\{|r_i(x_0)| : i \in \sigma_s\}$ such that $s_\gamma(x_\gamma) = s$ for $\gamma \in (0, \gamma^*]$.*

Proof. Since A has full rank, under the nondegeneracy assumption on (y_0, x_0) any solution d to the optimality system (2.7)

$$(AWA^T)d = As + b + x_0$$

satisfies

$$\|WA^T d\|_\infty < 1.$$

Now, using the fact that $\lim_{\gamma \to 0} d_\gamma = d^*$, where d^* denotes the minimum-norm solution to (2.7) and the continuity of the norm in its argument, there exists $\gamma_1^* > 0$ such that for $\gamma \in (0, \gamma_1^*]$ the unique solution d_γ of (2.35) satisfies

$$\|WA^T d_\gamma\|_\infty < 1.$$

The rest of the proof follows from Theorem 2.4. ∎

Hence, under a nondegeneracy assumption, the Huber problem is guaranteed to generate a sign vector identical to the sign vector corresponding to the dual optimal point x_0 for a sufficiently small value γ^* of γ. The magnitude of γ^* is related to the smallest nonzero component of $r(x_0)$ as stated in Corollary 2.1.

3. The algorithm. The new algorithm is based on minimizing the function F_γ for a set of decreasing values of γ. It can be described as follows. Starting from a point x, we find a minimizer of F_γ for some $\gamma > 0$, i.e., we locate the solution path for some value of γ. Utilizing Theorem 2.2 we compute $(\hat{y}_0, \hat{x}_0)$, estimates of primal-dual solutions. If optimality is not reached at $(\hat{y}_0, \hat{x}_0)$, we reduce the value of γ. Starting from a new point corresponding to the reduced value of γ, we compute the exact minimizer of F_γ using a Newton-type algorithm. Hence, we follow the solution path closely without having to stay on it. Based on Theorem 2.2, this process terminates when the duality gap is closed and primal feasibility is obtained.

The algorithm has two main components: (1) the solution of the smooth problem, i.e., minimization of F_γ for a given value of γ; (2) the check for optimality and the reduction of γ with the computation of an initial point for the solution of the subsequent Huber problem. We now consider these two components in detail.

3.1. Solving the Huber problem.

3.1.1. Properties of F_γ. In this section we describe some essential properties of F_γ.

Clearly, F_γ is composed of a finite number of quadratic functions. In each domain $D \subseteq \Re^m$, where $s_\gamma(x)$ is constant, F_γ is equal to a specific quadratic function. These domains are separated by the following union of hyperplanes:

$$B_\gamma = \{x \in \Re^m \mid \exists i \ : \ |r_i(x)| = \gamma\}.$$

A sign vector s is γ-*feasible* at x if

$$\text{for all } \varepsilon > 0 \ \exists z \in \Re^m \setminus B_\gamma \ : \ \|x - z\| < \varepsilon \ \wedge \ s = s_\gamma(z).$$

If s is a γ-feasible sign vector at some point x, then let Q_s be the quadratic function which equals F_γ on the subset

$$\mathcal{C}_s^\gamma = \text{cl}\{z \in \Re^m \mid s_\gamma(z) = s\}. \tag{3.1}$$

$\mathcal{C}_s^\gamma$ is called a *Q-subset* of $\Re^m$. Notice that any $x \in \Re^m \setminus B_\gamma$ has exactly one corresponding Q-subset ($s = s_\gamma(x)$), whereas a point $x \in B_\gamma$ belongs to two or more Q-subsets. Therefore, in general we must give a sign vector s in addition to x in order to specify which quadratic function we are currently considering as representative of F_γ. However, the gradient of F_γ is independent of the choice of s.

Q_s can be defined as follows:

$$Q_s(z) = \frac{1}{2\gamma}(z-x)^T(AW_sA^T + I)(z-x) + F_\gamma'^T(x)(z-x) + F_\gamma(x). \tag{3.2}$$

The gradient of the function F_γ is given by

$$F_\gamma'(x) = A\left[\frac{1}{\gamma}W_s r(x) + s\right] + b + x, \tag{3.3}$$

where s is a γ-feasible sign vector at x. For $x \in \Re^m \setminus B_\gamma$, the Hessian of F_γ exists and is given by

$$F_\gamma''(x) = \frac{1}{\gamma}AW_sA^T + I. \tag{3.4}$$

The set of indices corresponding to "small" residuals

$$\mathcal{A}_\gamma(z) = \{i \mid 1 \le i \le m \ \wedge \ s_{\gamma i}(z) = 0\} \tag{3.5}$$

is called the γ-*active* set at z.

3.1.2. Computing a minimizer of F_γ. The algorithm for computing a minimizer x^* of F_γ is based on a modified Newton algorithm given in [10]. This algorithm becomes simpler in our case as a result of strict convexity of the objective function. The algorithm consists of applying Newton's method to the function F_γ followed by a piecewise linear one-dimensional search. The idea is to locate the Q-subset of $\Re^m$ which contains its own minimizer using Newton's method. A search direction h is computed by minimizing the quadratic Q_s, where $s = s_\gamma(x)$ and x is the current iterate. More precisely, we consider the equation

$$Q_s''h = -Q_s'(x),$$

where Q_s'' and Q_s' denote the Hessian and gradient of Q_s, respectively. From (3.2)–(3.4) we obtain

$$(AW_sA^T + \gamma I)h = -AW_sr - \gamma(As + b + x). \tag{3.6}$$

The next iterate is found by a line search aiming for a zero of the directional derivative [10]. More precisely, the next iterate is the point $x + \alpha h$, $\alpha > 0$, for which the function

$$\rho(\alpha) = F_\gamma(x + \alpha h)$$

is minimized. Since ρ is a convex univariate function, the problem is to find a zero of the increasing piecewise-linear smooth function ρ'. The solution α to this problem is positive since $\rho'(0) < 0$ by the definition of h.

Let $\{\alpha_k\}$, $k = 1, \ldots, n$ be the set of positive breakpoints where ρ' has kinks, i.e., the set of points where an $s_{\gamma i}(x + \alpha h)$ changes value:

$$\mathcal{K} = \{\alpha > 0 \mid \exists i \in E: \ |(A^T(x + \alpha h))_i| = \gamma\},$$

where $E = \{i \mid 1 \le i \le m \ \wedge \ (A^T h)_i \neq 0\}$. Assume that the points α_k, $k = 1, \ldots, n$ are given in ascending order. Then the line search procedure is as follows:

$j := 0$
$\alpha_0 = 0$
repeat
    $j \leftarrow j + 1$
    find $\rho'(\alpha_j)$
until $\rho'(\alpha_j) \ge 0$
find the zero α of the linear function ρ' in the interval $[\alpha_{j-1}, \alpha_j]$.

This procedure is computationally cheap as a result of the piecewise-linear nature of F'_γ. First, the elements of the set $\mathcal{K}$ need not be sorted in practice. It suffices to pick the smallest element among the elements that remain in the set as the search proceeds. Furthermore, the quantity $\rho'(\alpha_j)$ is easily obtained from $\rho'(\alpha_{j-1})$, since the move from α_{j-1} to α_j only affects one term in the defining equation of ρ'. A more detailed description of this procedure is given in [10].

We summarize below the modified Newton algorithm:

repeat
    $s = s_\gamma(x)$
    find h from (3.6)
    if $x + h \in \mathcal{C}_s^\gamma$ then
        $x \leftarrow x + h$
        stop = true
    else
        $x \leftarrow x + \alpha h$ (line search)
    endif
until stop.

The algorithm stops when we have $x + h \in \mathcal{C}_{s(x)}^\gamma$, i.e., we have found the local quadratic which contains its own minimum. Therefore, $x + h$ is a minimizer of F_γ as a result of (3.1), (3.2), and the convexity of F_γ. Now, we show that this occurs in a finite number of iterations. First, we notice that the line searches made in the algorithm are well defined. This follows from two observations. First, since A has full rank, there exists an index j for which $(A^T h)_j \neq 0$. Hence, the set E of breakpoints is always nonempty. Furthermore, $\rho(\alpha)$ is a strictly convex quadratic function of α, which implies that the line search must terminate at a minimum along the half-line.

THEOREM 3.1. *The Newton algorithm stops at a minimizer of F_γ after a finite number of iterations.*

Proof. The set of iterates is bounded since the method is descent. Suppose that the iteration is infinite. Then, the set of iterates must have an accumulation point, z^*, say. We consider two cases:

(i) $F'_\gamma(z^*) \neq \mathbf{0}$: Since F'_γ is continuous and since F_γ is composed of a finite number of quadratics, all directions are found via a finite set of positive definite matrices $AW_sA^T + \gamma I$. Hence, there exists $\epsilon > 0$ and $\delta > 0$ such that $\|z^* - x\| < \epsilon$ implies $F_\gamma(x) - F_\gamma(x_{next}) > \delta$, where x_{next} is the successor of x in the iteration. Since this happens infinitely often, the function values must tend to $-\infty$, which contradicts the boundedness of F_γ from below.

(ii) $F'_\gamma(z^*) = \mathbf{0}$: In this case, z^* is the minimizer of F_γ because of convexity. Let x be an iterate with $z^* \in \mathcal{C}_{s(x)}^\gamma$. Since z^* minimizes the quadratic Q_s and h is found by (3.6), $x + h = z^*$, and the algorithm stops. ∎

3.2. Checking optimality and reducing γ. Let x_γ be a minimizer of F_γ computed using the Newton algorithm of the previous section. Then, either the continuation algorithm terminates or the Newton algorithm is restarted using a reduced value of γ.

The stopping test is based on Theorem 2.2. It consists of checking the duality gap $H(\hat{y}_0) - F(\hat{x}_0)$ and the feasibility of $\hat{y}_0$, where $(\hat{y}_0, \hat{x}_0)$ are as given in Theorem 2.2. If the duality gap is zero (within the roundoff tolerance), then the algorithm is stopped provided the components of $\hat{y}_0$ satisfy

$$-1 \le y_i \le 1.$$

Otherwise, γ is decreased as

$$\gamma^{new} = \beta \cdot \gamma^{old},$$

where $\beta \in (0, 1)$. The precise description of this procedure is as follows:

$s = s_\gamma(x_\gamma)$
compute the minimum norm solution $d_\gamma^{(\gamma)}$ to $(AWA^T)d = As + b + x_\gamma$
compute $\hat{x}_0 = x_\gamma + \gamma d_\gamma^{(\gamma)}$
compute the minimum norm solution $d_\gamma^{(0)}$ to $(AWA^T)d = As + b + \hat{x}_0$
compute $\hat{y}_0 = WA^T d_\gamma^{(0)} - s$
if $H(\hat{y}_0) - F(\hat{x}_0) = 0$ and $\hat{y}_0$ is feasible then
    stop = true
else
    $\gamma \leftarrow \beta \cdot \gamma$
endif

To compute an advantageous starting point for the subsequent Newton iteration with γ^{new}, we use the following linear system derived from necessary conditions (2.17):

$$(AWA^T + \gamma^{new} I)x = -\gamma^{new}(As + b), \tag{3.7}$$

where $s = s_\gamma(x_\gamma)$ and $W = W_\gamma(x_\gamma)$. The solution x^{new} of (3.7) is used as the starting point for the Newton iteration.

We note that this procedure guarantees that, unless the duality gap is closed, γ is decreased by a nonzero factor after each unconstrained minimization. Hence, we have the following theorem.

THEOREM 3.2. *The continuation algorithm described in sections* 3.1.2 *and* 3.2 *stops at a primal-dual optimal pair* (y_0, x_0) *after a finite number of iterations.*

Proof. As a result of the above observation, γ is reduced by a certain factor after each unconstrained minimization phase unless optimality is reached. Hence, using Theorem 2.3, γ can only be decreased a finite number of times. Since the Newton algorithm of section 3.1.2 is finite (Theorem 3.1), the result follows. ∎

4. Implementation and testing. The major effort in the dual algorithm of section 3.1.2 is spent in solving systems (3.6) and (2.29). We use the AAFAC package of [15] to perform this. The solution is obtained via an LDL^T factorization of the matrix $C_k = AW_sA^T + \gamma I$ (where γ is zero in the case of (2.29)), so D and L are computed directly from the γ-active columns of A, i.e., without squaring the condition number as would be the case if C_k was first computed. The efficiency of the Newton algorithm depends critically on the fact that the difference between the γ-active set $\mathcal{A}_\gamma(x_k)$ and $\mathcal{A}_\gamma(x_{k-1})$ is caused by a few elements. This implies

that the factorization of C_k can be obtained by relatively few up- and downdates of the factorization of C_{k-1}. Therefore, the computational cost of a typical iteration step is $O(m^2)$. Occasionally, a refactorization is performed. This consists of the successive updating of $LDL^T \leftarrow LDL^T + a_j a_j^T$ for all j in the γ-active set (starting with $L = I, D = \gamma I$). It is considered only when some columns of A leave the active set, i.e., when downdating is involved. If many columns leave, we may refactorize because it is cheaper. This part of the algorithm combines ideas from [3, 4]. For details see section 2 in [15]. The refactorization is an $O(m^3)$ process.

When a minimizer x_γ is at hand, a refactorization is needed to compute the minimum-norm solutions in system (2.29).

The stopping criteria in the Newton algorithm are implemented as follows. The iterate $x + h$ is considered to be in $\mathcal{C}_s^\gamma$ if

$$[\text{for all } i \in \mathcal{A}_\gamma(x) : \ |r_i + (A^T h)_i| \le \gamma + \tau\,] \ \text{ and}$$
$$[\text{for all } i \notin \mathcal{A}_\gamma(x) : \ s_{\gamma i} \cdot (r_i + (A^T h)_i) > \gamma - \tau\,].$$

Here, $\tau \approx O(\varepsilon_M \|A\|_\infty \|x\|_\infty)$ is used to take into account effects of rounding errors; ε_M denotes unit roundoff of the computer. We refer to the subroutine that implements the algorithm as QPASL1. With the exception of some internal tolerance parameters (e.g., tolerances used for numerical checks for zero) QPASL1 does not allow any control over the execution of the algorithm. Hence, all the results reported in this study were obtained under identical algorithmic choices. Further implementation details are given in [16].

4.1. Test problems. We generate test problems using ideas described in [1, 6, 14].

A symmetric positive definite matrix Q is generated as $Q = M^T M$, where $M = D^{1/2} Y$ and $Y = I - (2/\|y\|_2) y y^T$ for some vector $y \in \Re^m$ randomly generated in the interval $(-1, 1)$. The matrix D is diagonal with components d_i:

$$\log d_i = \frac{(i-1)}{(n-1)} ncond \quad \text{for } i = 1, \ldots, m.$$

It is easy to verify that *ncond* specifies the condition number of the matrix Q. The matrix A is obtained as the Cholesky factor of Q. This implies that A is triangular, and it is easy to recover the dual optimal solution from the generated "residual" vector r using (2.3).

The components of vectors y and r are generated simultaneously in accordance with a randomly generated sign vector s as follows.

```
for  i = 1 : m  do
     Generate μ uniformly in (−1, 1)
     if |m · μ| < nb then
          s_i = (−1)^{i−1}
          Generate ν uniformly in (0, 1)
          r_i = s_i 10^{−ν·ndeg}
     else
          y_i = μ
          r_i = 0
          s_i = 0
     endif
end
```

To introduce near-degeneracy, we use the following identity to define r_i if $s_i = 1$ or -1:

$$r_i = s_i 10^{-\nu \cdot ndeg}.$$

Near-degeneracy is turned off by choosing $ndeg = 1$. Furthermore, the parameter nb in the above procedure is chosen as a fraction of m. Knowing r, x is computed from definition (2.3) by solving the linear system

$$A^T x = r.$$

Finally, using the necessary condition for a minimizer (2.17) of F_γ we obtain b from the identity:

$$b = Ay - x.$$

4.2. Competing algorithms. The main competitors of the proposed algorithm are active set methods and interior point methods.

Active set methods choose a subset of the set of variables to be fixed at their lower and upper bounds. The resulting quadratic problem is solved over the free variables. The algorithm generates a descent direction keeping the variables in the active set fixed at their bounds, and performs a line search restricted by the largest step that can be used before one of the free variables reaches a bound. This scheme is repeated until a unit step length is found. At the end of this phase the Karush–Kuhn–Tucker optimality conditions are checked at the candidate point. If there is a variable which fails to satisfy the optimality conditions, it is removed from the active set. The algorithm repeats by solving a new quadratic problem over the updated set of free variables. The software system LSSOL contains a numerically stable and efficient implementation of the active set algorithm [5].

In [14], Moré and Toraldo propose a modification of the active set algorithm. The modification consists of taking projected gradient steps starting from a point obtained from solving the quadratic problem over the free variables as described above. This way, the proposed algorithm is able to make bigger changes to the active set than the original active set algorithm which makes a single change at a time. Unfortunately, an implementation of this algorithm was not available for comparison.

Our algorithm makes significant changes to the active set at each iteration and also when γ is reduced. In this regard, it is closer to the Moré–Toraldo algorithm than the pure active set strategy.

In [6], Han, Pardalos, and Ye develop a primal-dual potential reduction algorithm for bound constrained quadratic programming problems. The main computational effort in their algorithm is the solution of a linear system of the form

$$(I + R^T D^{-1} R)p = g,$$

where R is an $m \times n$ matrix, D is a diagonal $n \times n$ positive definite matrix, and p and g are m-vectors. As this algorithm was simple to program, we developed an efficient implementation making extensive use of BLAS routines for comparison to QPASL1. We refer to this code as HPY.

In [1], Coleman and Hulbert propose a superlinearly convergent Newton algorithm for bound constrained quadratic programs with unit bounds. The main effort in this algorithm is also the solution of a linear system

$$(|Y| + R^{1/2} H R^{1/2})v = g,$$

where Y is a diagonal matrix with nonzero entries, R is a nonsingular matrix, and H is the matrix of the quadratic term in [BCQP]. Clearly, both linear systems have a structure similar to (3.6). The algorithm by Coleman and Hulbert also uses a one-dimensional search which is similar to that described in section 3.1.2. However, in the algorithms of [6] and [1] a numerical refactorization needs to be performed at each iteration, whereas we only perform a refactorization when it is cheaper or numerically advisable to so. Hence, our average iteration is cheaper than any iteration of these algorithms. An implementation of the Coleman–Hulbert algorithm is not available for comparison. However, a close inspection of the results of [1] reveals that our algorithm uses consistently much smaller numbers of iterations to solve test problems with similar characteristics. To give an example, the Coleman–Hulbert algorithm requires between 10.8 and 17.0 iterations (varying *lcnd* and *ndeg*) on the average for m = 100, whereas our algorithm only requires between 3.8 and 9.6 for the same size for a similar degree of accuracy.

4.3. Initialization. We tested both QPASL1 and LSSOL with different starting points based on the recommendation of an anonymous referee. For LSSOL, we use the following starting points: (1) we choose a starting point y^0 as $y_j^0 = 0$ for all $j = 1, \ldots, m$; (2) we compute $\bar{y} = Q^{-1}d$ and select the initial point as

$$y_i = \begin{cases} -1 & \text{if } \bar{y}_i \leq -1, \\ 1 & \text{if } \bar{y}_i \geq +1, \\ \bar{y}_i & \text{otherwise.} \end{cases}$$

For QPASL1 we also use two different starting points. The first starting point is computed as follows. We fix a value of γ and use the following procedure, based on treating the objective function as

$$\frac{1}{2\gamma} r^T(x) r(x) + b^T x + \frac{1}{2} x^T x + \frac{1}{2} b^T b.$$

The necessary condition for a minimizer is

$$(AA^T + \gamma I)x = -\gamma b.$$

We compute a solution x to the above linear system and use $x^0 = x$. This is referred to as the *least squares starting point*. The second starting point is inspired by the second starting point used for LSSOL. We fix a value of γ and compute $\bar{y} = Q^{-1}d$. Then we set

$$s_i = \begin{cases} -1 & \text{if } \bar{y}_i \leq -1, \\ 1 & \text{if } \bar{y}_i \geq +1, \\ 0 & \text{otherwise.} \end{cases}$$

We compute x_0 as the solution to the system

$$(AWA^T + \gamma I)x = -\gamma(As + b),$$

where W is the diagonal matrix associated with s.

For HPY we use the initial point suggested in [6].

4.4. Numerical results. In this section we report our numerical experience with a Fortran 77 implementation of the new algorithm, which does not exploit sparsity. We have three goals when we perform numerical experiments. The first is to examine the growth in solution time and iteration count of the new algorithm as the problem size is increased. The second is to test the numerical accuracy of the algorithm. The third is to estimate the relative standing of the algorithm vis-á-vis other software systems. We compare our results to a library routine, E04NCF, from the NAG subroutine library. E04NCF is based on LSSOL from the Stanford Systems Optimization Library. We also offer comparisons with our own implementation of the interior point algorithm of Han, Pardalos, and Ye [6].

Below we report the results of the following experiments:

1. The effect of near-degeneracy.
2. The effect of the condition number.
3. The effect of the number of variables at their bounds at the optimal solution.
4. The impact of the problem size.

We solve 10 problems of each size. The parameter nb is kept at the value $m/2$ unless otherwise indicated. The tests were performed on a SPARC 4 Workstation running Solaris with the `-O` switch of the F77 compiler. In all tables below, each line reports the average over 10 problems of the following QPASL1 statistics: number of iterations, run time in CPU seconds, number of refactorizations, and number of γ reductions. The column "it" refers to the total number of iterations of the Newton method and the total number of optimality checks during the execution of the algorithm. The column "rf" refers to the total number of refactorizations in connection with the computations of the factors L and D. The column "rd" refers to the total number of times the optimality check was performed and/or γ was reduced. The heading QPASL1(2) refers to the second starting point for QPASL1, whereas QPASL1(1) refers to the least squares starting point. Similarly, LSSOL(2) indicates the second initial point, while LSSOL(1) refers to the use of the origin as the initial point. The columns under the heading LSSOL contain the run time statistics of LSSOL averaged over 10 problems for each line. All runs with QPASL1, LSSOL, and HPY were performed using default parameters, i.e., no fine tuning of the codes was done for any test problem.

QPASL1 is stopped when the relative duality gap

$$(H(\hat{y}_0) - F(\hat{x}_0))/(1 + F(\hat{x}_0))$$

is less than or equal to 10^{-8} and the primal feasibility measure $\|\hat{y}_0\|_\infty$ is less than or equal to $1 + \epsilon_y$ with $\epsilon_y = 10^{-5}$. The final accuracy obtained in QPASL1 is measured using the accuracy in the objective function and the primal solution with respect to the known optimal value and optimal solution vector. The accuracy in the optimal value is checked using

$$q_1 = \frac{H(y_0) - H(\hat{y}_0)}{H(y_0)},$$

where $H(y_0)$ is the known optimal value, and the accuracy in the solution is checked using

$$q_2 = \|y_0 - \hat{y}_0\|_2 / \|y_0\|_2,$$

where y_0 and $\hat{y}_0$ denote the known and computed optimal values, respectively. In all test problems solved in this study, we have

$$10^{-16} \leq q_1 \leq 10^{-12}.$$

Depending on the conditioning of the problem, we also obtain

$$10^{-12} \leq q_2 \leq 10^{-9}.$$

This indicates that we achieve high accuracy in the computed optimal solution. Regarding other parameters, we use $\gamma^0 = 10^{-3}$ as the starting value of γ, and $\beta = 1/100$.

LSSOL yields objective function values accurate to machine precision in all cases. For HPY, the quantities q_1 and q_2 vary as follows:

$$10^{-9} \leq q_1 \leq 10^{-8},$$

$$10^{-8} \leq q_2 \leq 10^{-5}.$$

4.4.1. Experiment 1: The effect of near-degeneracy. In Table 4.1 we give computational results obtained when the near-degeneracy parameter *ndeg* is increased.

We make the following observations.

- QPASL1 is competitive with LSSOL for small values (1,3) of the parameter *ndeg*, whereas for larger values it loses its advantage. It is also substantially faster than HPY.
- The iteration number of QPASL1 remains very small and almost constant with the increasing problem size for small values of *ndeg*.
- The parameter *ndeg* has almost no effect on the performance of LSSOL.
- The two starting points for QPASL1 tend to perform similarly when near-degeneracy is increased.

The reason for the deterioration in performance of QPASL1 for larger values of *ndeg* is precisely related to Corollary 2.1. It is shown in this corollary that the value of

TABLE 4.1
Solution statistics of QPASL1 and LSSOL when near-degeneracy is increased.

$m, lcnd, ndeg$	QPASL1(2)				QPASL1(1)			
	it	rf	rd	CPU	it	rf	rd	CPU
100, 1, 1	3.8	2	1	0.4	4.1	3	1	0.6
100, 1, 3	5.2	2.1	1.1	0.5	5.9	3.1	1.1	0.7
100, 1, 6	9.6	3.1	2.1	1.1	10.3	3.4	2.1	1.3
200, 1, 1	4.2	2	1	2.3	5.1	3	1	4.0
200, 1, 3	5.1	2.1	1.1	3.0	6	3.1	1.1	4.8
200, 1, 6	9.5	3.1	2.1	6.9	10.2	3.3	2.1	8.6
300, 1, 1	4	2	1	6.8	3.8	3	1	13.1
300, 1, 3	4.8	2.2	1.2	8.7	5.6	3.2	1.2	15.2
300, 1, 6	9.3	3.3	2.3	22.1	10.9	3.8	2.3	27.5

$m, lcnd, ndeg$	LSSOL(2)		LSSOL(1)		HPY	
	it	CPU	it	CPU	it	CPU
100, 1, 1	14.5	0.5	50	0.9	18	2.6
100, 1, 3	21.6	0.6	50	0.9	16.9	2.3
100, 1, 6	23.5	0.6	45.7	0.8	14.9	2.0
200, 1, 1	27.8	3.8	100.6	6.0	16	15.9
200, 1, 3	39.9	4.1	100.6	6.0	16.2	15.6
200, 1, 6	46.4	4.4	91.7	5.6	17.5	16.7
300, 1, 1	16	10.1	152.4	19.7	16.8	51.5
300, 1, 3	34.6	10.9	152.4	19.5	18.6	56.9
300, 1, 6	44.5	11.7	140.2	18.7	18.2	55.5

TABLE 4.2
Solution statistics of QPASL1 and LSSOL when the condition number is increased.

$m, lcnd, ndeg$	QPASL1(2)				QPASL1(1)			
	it	rf	rd	CPU	it	rf	rd	CPU
100, 4, 1	3.8	2	1	0.4	3.9	3.1	1	0.6
100, 8, 1	3.8	2	1	0.4	4.1	3.1	1	0.6
200, 4, 1	4	2	1	2.2	5.2	3	1	4.0
200, 8, 1	8.5	2.2	1	2.9	5.5	3	1	4.0
300, 4, 1	3.9	2	1	6.8	3.8	3	1	12.3
300, 8, 1	3.9	2	1	6.9	4.1	3	1	12.4

$m, lcnd, ndeg$	LSSOL(2)		LSSOL(1)		HPY	
	it	CPU	it	CPU	it	CPU
100, 4, 1	12.8	0.5	50	0.9	14.6	1.9
100, 8, 1	13.9	0.5	49.7	0.8	17.2	2.3
200, 4, 1	32.2	3.9	100.6	5.9	14.9	14.2
200, 8, 1	29.2	3.8	100.2	5.8	17.3	16.6
300, 4, 1	18	10.2	152.6	19.1	17.3	52.9
300, 8, 1	32.4	11.4	152.4	19.1	17	51.9

γ^* is affected by the magnitude of nonzero residuals $r(x_0)$ at the optimal solution x_0. The smaller the residuals, the more γ should be reduced in order to reach the optimal solution. This increases the number of reduction steps and the total number of iterations, thereby causing a degradation in performance.

4.4.2. Experiment 2: The effect of the condition number. In Table 4.2 we summarize the average performance of the three codes when the conditioning parameter *lcond* is increased.

It is observed that all three codes handle problems with increasing condition number equally well.

4.4.3. Experiment 3: The effect of the number of variables at bounds. The number of variables at a bound at an optimal solution can be controlled by varying the parameter nb. We do so in this experiment and report the results in Table 4.3.

We notice that the performance of LSSOL improves significantly when nb becomes smaller than $m/2$ and worsens when it exceeds that value. This improvement is more marked when the zero starting point is used. A similar improvement occurs with HPY, whereas the opposite is true of QPASL1.

4.4.4. Experiment 4: The effect of the problem size. To illustrate the effect of increasing problem size on the performance of the three codes, we provide some results in Table 4.4.

We notice that LSSOL consumes about 1.5 times more CPU time than QPASL1 as we increase the problem size, while HPY uses approximately 10 times more CPU compared to QPASL1.

5. Summary and concluding remarks. In this paper, we presented a dual approach to strictly convex quadratic programming with unit bounds.

Our dual approach consisted of posing the problem [BCQP] as an unconstrained ℓ_1 minimization problem and approximating this nondifferentiable problem by a smooth Huber problem. The minimizers of the smooth problem define a unique path that converges to the primal-dual optimal solutions as a function of a scalar parameter

TABLE 4.3
Solution statistics of QPASL1 and LSSOL when nb is varied.

$m, lcnd, ndeg, nb$	QPASL1(2)				QPASL1(1)			
	it	rf	rd	CPU	it	rf	rd	CPU
$100, 1, 1, m/2$	3.8	2	1	0.4	4.1	3	1	0.6
$100, 1, 1, m/10$	3.7	2	1	0.5	4.2	2	1	0.5
$100, 1, 1, 3m/4$	5	3.1	1	0.4	3.5	3	1	0.5
$200, 1, 1, m/2$	4.2	2	1	2.3	5.1	3	1	4.0
$200, 1, 1, m/10$	4.1	2	1	3.6	6	4.6	2	4.0
$200, 1, 1, 3m/4$	8.1	3	1	2.8	4.5	3	1	3.1
$300, 1, 1, m/2$	4	2	1	6.8	3.8	3	1	13.1
$300, 1, 1, m/10$	3.9	2	1	11.4	4.2	2	1	12.7
$300, 1, 1, 3m/4$	9.3	3	1	8.8	3.9	3	1	10.0

$m, lcnd, ndeg, nb$	LSSOL(2)		LSSOL(1)		HPY	
	it	CPU	it	CPU	it	CPU
$100, 1, 1, m/2$	14.5	0.5	50	0.9	18	2.6
$100, 1, 1, m/10$	13	0.3	10.6	0.21	13.6	1.9
$100, 1, 1, 3m/4$	14.4	0.4	76.1	1.1	15.6	2.1
$200, 1, 1, m/2$	27.8	3.8	100.6	6.0	16	15.9
$200, 1, 1, m/10$	17.2	1.9	19.4	1.4	14.4	14.0
$200, 1, 1, 3m/4$	28.5	3.0	150.4	7.5	16	15.3
$300, 1, 1, m/2$	16	10.1	152.4	19.7	16.8	51.5
$300, 1, 1, m/10$	26.1	6.19	30	4.5	15.5	47.7
$300, 1, 1, 3m/4$	25.3	9.1	223.3	24.0	16	49.9

TABLE 4.4
Solution statistics of QPASL1 and LSSOL when the problem size is increased.

$m, lcnd, ndeg$	QPASL1(2)				QPASL1(1)			
	it	rf	rd	CPU	it	rf	rd	CPU
100, 1, 1	3.8	2	1	0.4	4.1	3	1	0.6
200, 1, 1	4.2	2	1.0	2.3	5.1	3	1	4
300, 1, 1	4	2	1	6.8	3.8	3	1	13.1
400, 1, 1	4.2	2	1	16.0	4.9	3.1	1	29.7
500, 1, 1	4.3	2	1	30.4	5.3	3.1	1	58.7

$m, lcnd, ndeg$	LSSOL(2)		LSSOL(1)		HPY	
	it	CPU	it	CPU	it	CPU
100, 1, 1	14.5	0.5	50	0.9	18	2.6
200, 1, 1	27.8	3.8	100.6	6.0	16	15.9
300, 1, 1	16	10.1	152.4	19.7	16.8	51.5
400, 1, 1	30.8	25.4	203.7	45.1	18.8	139.4
500, 1, 1	40.8	48.1	253	85.9	20.2	288.4

γ. This suggested a continuation algorithm, where we follow this path to arrive at primal-dual optimal solutions.

On the theoretical front, we established an extrapolation property of the solution path and a constant sign property (for sufficiently small γ), which formed the pillar of finite convergence of the continuation algorithm. We also gave a finite Newton algorithm to solve the Huber problems.

On the practical front, we developed a stable and efficient implementation of the algorithm for dense problems. We compared our results to an established software system for quadratic programming, LSSOL, and to more recent algorithms for [BCQP]. The following picture emerged from our experiments. The new algorithm is competitive with a state-of-the-art implementation of active set methods for problems

with low degree of near-degeneracy. It also handles problems with increasing condition number very well. It is also substantially faster than an interior point algorithm proposed for [BCQP].

Finally we remark that the duality framework of section 2 can be easily extended to problems where bounds are different from unity and/or where one of the bounds is missing; see [2]. Nonunit bounds simply change the slope of the nondifferentiable function arising in the dual problem. By way of illustration, consider the following case:

$$\begin{aligned} \min_{y} \quad & H(y) = -d^T y + \tfrac{1}{2} y^T Q y \\ \text{subject to} \quad & l \le y. \end{aligned}$$

The nondifferentiable dual problem corresponding to the program above is

$$\text{minimize } F(x) \;\equiv\; \sum_{i=1}^{m} \rho_i(r_i(x)) + \frac{1}{2} x^T x + b^T x + \frac{1}{2} b^T b,$$

where

$$\rho_i(r_i) = \begin{cases} l_i r_i & \text{if } r_i \ge 0 \\ \infty & \text{otherwise,} \end{cases}$$

and the vectors r and b are defined as in section 2. The nondifferentiable function ρ can be approximated by the following smooth Huber function

$$\psi_\gamma(r_i) = \begin{cases} l_i r_i - \frac{1}{2}\gamma & \text{if } r_i \ge \gamma, \\ \frac{1}{2\gamma} r_i^2 & \text{if } r_i < \gamma, \end{cases}$$

for some scalar parameter $\gamma > 0$. The properties and the algorithm derived in this paper apply to the above approximation as well.

Acknowledgments. The helpful comments and suggestions of two anonymous referees are gratefully acknowledged.

REFERENCES

[1] T. Coleman and L. Hulbert, *A globally and superlinearly convergent algorithm for quadratic programming with simple bounds*, SIAM J. Optim., 3 (1993), pp. 298–321.
[2] O. Edlund, *Private communication*, Luleå University of Technology, Luleå, Sweden, 1997.
[3] R. Fletcher and M. J. D. Powell, *On the modification of LDL^T factorizations*, Math. Comp., 28 (1974), pp. 1067–1087.
[4] W. M. Gentleman, *Least squares computation by Givens transformations without square roots*, J. Inst. Math. Appl., 12 (1973), pp. 329–336.
[5] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright, *User's Guide for LSSOL* (*Version* 1.0): *A Fortran Package for Constrained Linear Least Squares and Convex Quadratic Programming*, Technical Report SOL 86-1, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA, 1986.
[6] C.-G. Han, P. Pardalos, and Y. Ye, *Computational aspects of an interior point algorithm for quadratic programming problems with box constraints*, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., SIAM, Philadelphia, PA, 1990, pp. 92–112.
[7] P. Huber, *Robust Statistics*, John Wiley, New York, 1981.
[8] W. Li and J. Swetits, *A Newton method for convex regression, data smoothing, and quadratic programming with bounded constraints*, SIAM J. Optim., 3 (1993), pp. 466–488.
[9] W. Li and J. Swetits, *A new algorithm for solving strictly convex quadratic programs,* SIAM J. Optim., 7 (1997), pp. 595–619.

[10] K. Madsen and H. B. Nielsen, *Finite algorithms for robust linear regression*, BIT, 30 (1990), pp. 682–699.
[11] K. Madsen, H. B. Nielsen, and M.Ç. Pinar, *A new finite continuation algorithm for linear programming*, SIAM J. Optim., 6 (1996), pp. 600–616.
[12] O. L. Mangasarian, *Normal solution of linear programs*, Math. Programming Stud., 22 (1984), pp. 206–216.
[13] O. L. Mangasarian and R. R. Meyer, *Nonlinear perturbation of linear programs*, SIAM J. Control Optim., 17 (1979), pp. 745–752.
[14] J. Moré and G. Toraldo, *Algorithms for bound constrained quadratic programming problems*, Numer. Math., 55 (1989), pp. 377–400.
[15] H. B. Nielsen, *AAFAC: A Package of Fortran* 77 *Subprograms for Solving* $A^T Ax = c$, Report NI-90-11, Institute of Mathematical Modelling, Numerical Analysis Group, Technical University of Denmark, DK-2800 Lyngby, Denmark, 1990.
[16] H. B. Nielsen, *Implementation of a Finite Algorithm for Linear* ℓ_1 *Estimation*, Report NI-91-01, Institute of Mathematical Modelling, Numerical Analysis Group, Technical University of Denmark, DK-2800 Lyngby, Denmark, 1991.