IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

J. No. QP4
29.4.1997
km/hbn/mcp

BOUND CONSTRAINED QUADRATIC PROGRAMMING VIA PIECEWISE QUADRATIC FUNCTIONS

Kaj Madsen

Hans Bruun Nielsen

Mustafa Çelebi Pınar

TECHNICAL REPORT

IMM-REP-1997-05

IMM

BOUND CONSTRAINED QUADRATIC PROGRAMMING VIA PIECEWISE QUADRATIC FUNCTIONS

Kaj Madsen[*], Hans Bruun Nielsen[*], Mustafa Ç. Pınar[†]

April 30, 1997

Abstract. We consider the strictly convex quadratic programming problem with bounded variables. A dual problem is derived using Lagrange duality. The dual problem is the minimization of an unconstrained, piecewise quadratic function. It involves a lower bound of λ_1, the smallest eigenvalue of a symmetric, positive definite matrix, and is solved by Newton iteration with line search. The paper describes the algorithm and its implementation including estimation of λ_1, how to get a good starting point for the iteration, and up– and downdating of Choleky factorization. Results of extensive testing and comparison with other methods for constrained QP are given.

Key words. Bound constrained quadratic programming. Huber's M–estimator. Condition estimation. Newton iteration. Factorization update.

1. Introduction

The purpose of the present paper is to describe a finite, dual Newton algorithm for the bound constrained quadratic programming problem. Let $c \in \mathbb{R}^n$ and $H \in \mathbb{R}^{n\times n}$ be a given vector and a symmetric, positive definite matrix, respectively. We seek $y^* \in \mathbb{R}^n$ as the solution to the constrained quadratic programming problem

$$\begin{aligned} &\min_y \{\; q(y) \equiv \tfrac{1}{2} y^T H\, y - c^T y \;\} \\ &\text{subject to} \quad -e \le y \le e\;. \end{aligned} \tag{1.1}$$

Here, $e \in \mathbb{R}^n$ is the vector of all ones.

The special form with unit bounds leads to particularly elegant duality results: In a straight forward way we demonstrate that the dual of (1.1) is a Huber M-estimator [7], i.e. a convex quadratic spline function F. This function is minimized using a special version of the Newton iteration with line searches [12].

The duality property has been derived in a more general setting by Li and Swetits [10], [11]. They also propose a Newton iteration, and our testing in Section 5 includes their implementation for the quadratic programming problem with simple bounds.

The main contribution of the present paper is to demonstrate how the implementation problems are overcome. We efficiently compute a guaranteed positive lower bound of the smallest eigenvalue of H, demonstrate how the Newton iteration can be implemented efficiently using factorization updates, and derive an efficient starting procedure. The numerical experiments indicate that the new method is computationally viable. In particular, we demonstrate that it is competitive with established software systems. We substantiate this claim in Section 5 where we present our computational results. In addition to the algorithm of Li and Swetits [10] we

[*]Institute of Mathematical Modelling, Technical University of Denmark, 2800 Lyngby, Denmark (km@imm.dtu.dk and hbn@imm.dtu.dk).

[†]Departm,ent of Industrial Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey (mustafap@Bilkent.EDU.TR).

compare with bqpd, a commercial software system for convex quadratic programming by R. Fletcher [4], and to a primal-dual interior point algorithm by Han *et al.* [6].

In a closely related paper [14] we discuss the solution of the quadratic programming problem via a dual ℓ_1-problem, which in its turn is solved via a series of Huber problems, cf. [13]. In [18] a more detailed account of the implementation of the present paper's algorithm is given.

The literature on quadratic programming is vast. We refer the reader to the paper by Moré and Toraldo [16] for a list of references. Some recent papers include Coleman and Hulbert [2] and Li and Swetits [10], [11]. In [2] Coleman and Hulbert reformulate (1.1) as an uncontrained minimization problem involving an ℓ_1 term. This reformulation is obtained by manipulating the Karush-Kuhn-Tucker conditions of (1.1). They apply a superlinearly convergent modified Newton method to this reformulation. Li and Swetits [10], [11] derive their reformulation of the convex quadratic programming problem by starting from the Karush-Kuhn-Tucker optimality conditions and deriving an unconstrained problem whose minimizer coincides with an optimal solution to (1.1). They use several auxiliary results on monotone mappings to arrive at their equivalence results.

The rest of this paper is organized as follows. First, we give a technical preview of our approach in Section 2. We derive our dual problem in Section 3. Section 4 is devoted to the description of the proposed algorithm and implementational details, and we conclude with a detailed summary of our computational experience in Section 5.

2. Preliminaries

Let $\lambda_1 > 0$ denote the smallest eigenvalue of H, and let γ be a number such that $0 < \gamma < \lambda_1$. Further, let $A \in \mathbb{R}^{n\times n}$ be a matrix that satisfies

$$A^TA = H - \gamma I \ . \tag{2.1}$$

Now, define the function

$$F(x) = \frac{1}{2\gamma} r^T W r + s^T(r - \frac{\gamma}{2}s) + \frac{1}{2}x^T x \ , \tag{2.2a}$$

where

$$r = r(x) = A^T x - c \ , \tag{2.2b}$$

$$s = s(x) = \begin{bmatrix} s_1(x) \\ \vdots \\ s_n(x) \end{bmatrix} \quad \text{with} \quad s_i(x) = \begin{cases} -1 \text{ if } & r_i(x) \le -\gamma \\ \ \ 0 \text{ if } & |r_i(x)| < \gamma \\ \ \ 1 \text{ if } & r_i(x) \ge \gamma \end{cases} \ , \tag{2.2c}$$

$$W = W(x) = \mathrm{diag}(w_1, \ldots, w_n) \quad \text{with} \quad w_i = 1 - s_i^2 \ . \tag{2.2d}$$

In Section 3 we show that (1.1) has the dual problem

$$\max_x\{-F(x)\} = -\min_x\{F(x)\} \ . \tag{2.3}$$

The hyperplanes given by $r_j(x) = \pm\gamma$ divide $\mathbb{R}^n$ into subregions, in each of which $F(x)$ is a quadratic. So the dual problem is to minimize the piecewise quadratic function F. It is easy to show that F is differentiable, and also that the gradient

$$F'(x) = A(\frac{1}{\gamma}W r + s) + x \tag{2.4}$$

varies continuously across the hyperplanes.

The minimizer x_γ of F satisfies $g(x_\gamma) = 0$, or

$$x_\gamma = A\, y_\gamma \ , \tag{2.5a}$$

where we have defined

$$y_\gamma = -(\frac{1}{\gamma}W(x_\gamma)r(x_\gamma) + s(x_\gamma)) \ . \tag{2.5b}$$

In Section 3 we show that $y_\gamma = y^*$ and $q(y_\gamma) = -F(x_\gamma)$. Thus,

$$\begin{aligned}\Gamma(x) &\equiv F(x) + q(-\tfrac{1}{\gamma}W(x)r(x) - s(x)) \\ &\geq F(x_\gamma) + q(y_\gamma) = 0\end{aligned} \tag{2.6}$$

can serve as a gap function.

In the discussion of the algorithm we say that r_i (and the ith column of A) is *active* at x, if $s_i(x) = 0$ (and therefore $w_i(x) = 1$). The *dual active set* is

$$\mathcal{A} = \mathcal{A}(x) = \{\ i \mid 1 \leq i \leq n \ \wedge s_i(x) = 0\} \ . \tag{2.7}$$

From (2.5b) it follows that an active $r_i(x_\gamma)$ is equivalent with $|y_i^*| < 1$, so the ith constraint in (1.1) is *primal non active*. Note, that if we knew the solution y^*, then it follows from (2.5) that the corresponding $s(x_\gamma)$ is given by

$$s_i(x_\gamma) = \begin{cases} -y_i^* & \text{if } |y_i^*| = 1 \\ \ \ 0 & \text{otherwise} \end{cases} \ , \tag{2.8a}$$

and x_γ is the solution to the linear problem derived from the condition $g(x_\gamma) = 0$,

$$(A\, W(x_\gamma)\, A^T + \gamma I)x_\gamma = A(W(x_\gamma)c - \gamma\, s(x_\gamma)) \ . \tag{2.8b}$$

3. The Dual Problem

In this section we show that (1.1) has the dual problem

$$\max_x\{-F(x)\} = -\min_x\{F(x)\} \ . \tag{3.1}$$

Our reference on duality in convex programming is the book by Rockafellar [19]. Let $M = H - \gamma I$ where $\gamma > 0$ is picked so as to have M positive definite. Clearly, a strict upper bound on the choice of γ is λ_1 where λ_1 denotes the smallest eigenvalue of H. Then (1.1) can be written

$$\begin{aligned}&\min_y \quad -c^T y + \tfrac{1}{2}y^T(M + \gamma I)y \\ &\text{s.t.} \quad -e \leq y \leq e \ .\end{aligned}$$

Since M is symmetric, positive definite, it can be written as $M = A^T A$ where $A \in \mathbb{R}^{n\times n}$ has full rank. Further, let $u = Ay$, and the problem takes the form

$$\begin{aligned}&\min_{u,y} \quad -c^T y + \tfrac{1}{2}u^T u + \tfrac{1}{2}\gamma\, y^T y \\ &\text{s.t.} \quad Ay = u \quad \text{and} \quad -e \leq y \leq e \ .\end{aligned}$$

Now, we are in a position to derive a dual problem to (1.1). Associating dual multipliers $x \in \mathbb{R}^n$ with the equality constraints we form the following Lagrangean max-min problem:

$$\max_x \left\{ \min_{u,\ -e \leq y \leq e} \{\tfrac{1}{2}u^T u + \tfrac{1}{2}\gamma\, y^T y - c^T y + x^T(Ay - u)\} \right\} \ , \tag{3.2}$$

which is equivalent to

$$\max_x \left\{ \min_u \{\tfrac{1}{2}u^T u - u^T x\} + \min_{-e \le y \le e} \{\tfrac{1}{2}\gamma\, y^T y + y^T (A^T x - c)\} \right\} .$$

From [19, Thm. 28.3] it is well-known that for (y, u) to be an optimal solution for (1.1), and for x to be a Lagrange multiplier vector, it is necessary and sufficient that (y, u, x) is a saddlepoint of the Lagrangean function. This holds if and only if the components of (y, u, x) satisfy the stationarity conditions with respect to u and y.

The first minimization problem over u yields the identity

$$u = x \; , \tag{3.3}$$

which, when plugged back in, yields the term $-\frac{1}{2}x^T x$.

The second minimization over y in the unit sphere yields the following cases:

1) If $(A^T x - c)_i \ge \gamma$, then $y_i = -1$.

2) If $(A^T x - c)_i \le -\gamma$, then $y_i = 1$.

3) If $-\gamma < (A^T x - c)_i < \gamma$, then by calculus $y_i = -\frac{1}{\gamma}(A^T x - c)_i$.

Notice that the above is equivalent to the following relation between y and x:

$$y \equiv -\frac{1}{\gamma} W\, r(x) - s \; . \tag{3.4}$$

From this relation we obtain

$$\min_{-1 \le y_i \le 1} \{\tfrac{1}{2}\gamma y_i^2 + y_i (A^T x - c)_i\} = \begin{cases} -\dfrac{1}{2\gamma}(A^T x - c)_i^2 & \text{if } |(A^T x - c)_i| < \gamma \\ -|(A^T x - c)_i| + \frac{1}{2}\gamma & \text{if } |(A^T x - c)_i| \ge \gamma \; . \end{cases}$$

Hence, the triple (y, u, x) is a primal-dual optimal triple if and only if the above three cases and (3.3) hold for them.

An equivalent statement of this duality-optimality relation is summarized in the following theorem:

Theorem 3.1. *Let x be a minimizer of F, and let $s = s(x)$ with W defined accordingly. Then, the unique optimal solution of (1.1) is given by (3.4).*

In the rest of the paper we are concerned with exploiting the above duality correspondence via a Newton-type algorithm.

4. The Algorithm

The algorithm for computing the solution to problem (1.1) can be stated as follows.

1° Compute R as the upper triangular Cholesky factor of H,

$$R^T R = H \; . \tag{4.1}$$

2° Compute γ and A; Sections 4.1 and 4.2.

3° Find the starting point; Section 4.3.

4° Compute the dual solution x_γ; see below.

5° Compute the primal solution (see below) and return.

The problem may be given in terms of R and c, in which case step 1° is omitted. Further, the algorithm is suited for "warm starts": If a problem with the same H has been solved previously, then steps 1° and 2° can be skipped. Also, step 3° can be replaced by information from the previous solution and the new c.

The algorithm for computing the minimizer x_γ of F, (2.2), is described in greater detail in [12] and [17]. We shall briefly repeat it here.

Given x and the corresponding r, s, W. The gradient of F is computed by (2.4), and if this is zero apart from rounding errors, then the algorithm stops. More specifically, we use the test

$$\|F'(x)\|_\infty \le \varepsilon_{\mathrm{M}}(\|A\|_\infty + \|x\|_\infty) \equiv \eta \;, \tag{4.2}$$

where ε_{M} is the machine accuracy (unit round-off).

If this criterion is not satisfied, then the Newton step h aiming at making the gradient zero is found as the solution to

$$(AWA^T + \gamma I)h = -\gamma F'(x) \;. \tag{4.3}$$

Note, that the coefficient matrix is symmetric and positive definite. This implies that h exists and is unique.

If $s(x{+}h) = s(x)$, then $x{+}h$ is a stationary point and hence a global minimizer of F. Otherwise, a line search along h is made. This consists in computing t^* which minimizes $F(x{+}th)$.

This dual algorithm is summarized below.

Dual Algorithm (Given starting point x and corresponding r and s)

```
stop := false
while (not stop) do
   Compute F'(x) by (2.4)
   if ||F'(x)||_∞ ≤ η then stop := true
   else
      Find h from (4.3)
      if (s(x+h) = s(x)) then stop := true
      else x := x + t*h                       {Line search; Section 4.5}
   end
end
```

Theorem 4.1. *The Dual Algorithm terminates in a finite number of iterations with a minimizer x_γ of F.*

A proof of this theorem may be obtained by suitable modification of the proof of Theorem 4.1 of [12]. Alternatively, the proof may be given similarly to the proof of Theorem 3.2 of [10].

The work is dominated by solution of the system (4.3) in each iteration. However, the number of changes in active set between two consecutive iterations often is small compared with n, and in Section 4.4 we show how this can be exploited so that the determination of each Newton step is an $O(n^2)$ process rather than $O(n^3)$.

When the dual algorithm stops, we have the information needed to compute the solution to the QP problem,

$$y = -(\frac{1}{\gamma}W(x)r(x) + s(x)) \;, \tag{4.4}$$

where $x = x_\gamma$, cf. (2.5). If the value of the gap function (2.6) is too big,

$$F(x) + q(y) > \min\{n, 20\} \cdot \varepsilon_{\mathrm{M}} \cdot |F(x)| \;, \tag{4.5}$$

then we refine the solution

$$\begin{aligned} &\text{refactorize} \quad AWA^T + \gamma I \ , \\ &h = \gamma(AWA^T + \gamma I)^{-1} F'(x) \ , \\ &x := x - h; \quad r := r - A^T h \ , \end{aligned} \tag{4.6}$$

and use x, r and the corresponding s to restart the Dual Algorithm. This restart is allowed once, only. The condition (4.5) is rarely satisfied, but when it is, this refinement drastically improves the accuracy of y.

In the remaining parts of this section we describe some important computational modules used in the implementation of the dual algorithm.

4.1. Compute γ

The shift parameter γ in (2.1) should be smaller than λ_1, the smallest eigenvalue of H. We choose it as

$$\gamma = f \cdot \overline{\lambda}_1 \ , \tag{4.7}$$

where $0 < f < 1$, and $\overline{\lambda}_1$ is an estimate of λ_1. In Section 4.3 of [18] we discuss the choice of f. The conclusion – supported by experiments – is that generally $f = 0.5$ is a good choice.

To explain our algorithm for computing $\overline{\lambda}_1$ we introduce the singular value decomposition (SVD) for the Cholesky factor R, (4.1),

$$R = U\Sigma V^T \quad \Longleftrightarrow \quad R^T = V\Sigma U^T \ , \tag{4.8a}$$

where U and V are orthogonal, and

$$\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n) \quad \text{with} \quad \sigma_1 \geq \ldots \geq \sigma_n > 0 \ . \tag{4.8b}$$

Equations (4.1) and (4.8a–b) imply

$$H = R^T R = V\Sigma^2 V^T = V\Lambda V^T, \ \text{ i.e. } \ \lambda_1 = \sigma_n^2 \ . \tag{4.8c}$$

The estimate $\overline{\lambda}_1$ is computed in two major steps:

1° Compute (u, v): estimates of $(U_{:,n}, \sigma_n^{-1} V_{:,n})$.

2° Refine the estimate by one step of simultaneous inverse iterations.

The pair (u, v) is computed by using ideas from some well-known condition estimators; see e.g. [5]. Details are described in the technical report [18]. The total cost of this estimator is about $4n^2$ flops.

Note, that we do not need a high accuracy: We use

$$\gamma = f \cdot \overline{\lambda}_1 = f \cdot (\lambda_1 + \delta) \quad \Longleftrightarrow \quad \lambda_1 - \gamma = (1-f)\lambda_1 - f\delta \ .$$

Thus, $\gamma < \lambda_1$ if $\delta < \dfrac{1-f}{f}\lambda_1$. With the recommend choice $f = 0.5$ it suffices to estimate λ_1 with a relative error less than 100%. In all our experiments we found that this was satisfied, [18]. However, we use a "safety valve": If the matrix $H - \gamma I$ is found to be indefinite – i.e. $\gamma \geq \lambda_1$ – then we replace γ by $\gamma := 0.1 \cdot \gamma$. This is described more precisely at the end of Section 4.2.

4.2. Compute A

The matrix A in the Huber function is defined by

$$A^T A = R^T R - \gamma I \ , \tag{4.9}$$

where R is upper triangular, cf. (4.1). We choose also to let A have this property. We might compute it simply as the upper triangular Cholesky factor of $H - \gamma I$. If, however, the problem is given directly by R, this approach would lead to unnecessary loss of accuracy. Instead we compute A via orthogonal transformation: From (4.9) we see that

$$R^T R = A^T A + \gamma I \ , \tag{4.10a}$$

which is equivalent with

$$\begin{bmatrix} R \\ 0 \end{bmatrix} = Q \begin{bmatrix} A \\ \sqrt{\gamma} I \end{bmatrix} \ , \tag{4.10b}$$

where Q is orthogonal. The transformation is computed via a series of Householder reflections. Before the kth transformation the partly transformed matrix has the structure shown in figure 4.1 for $n = 5$, $k = 3$.

$$\left[\begin{array}{ccccc} * & * & * & * & * \\ & * & * & * & * \\ & & \star & \star & \star \\ & & & \star & \star \\ & & & & \star \\ & & \star & \star & \star \\ & & \star & \star & \star \\ & & \star & & \\ & & & \star & \\ & & & & \star \end{array}\right] \begin{array}{l} \left.\begin{array}{c} \\ \\ \end{array}\right\} \text{Part of } R \ \text{(overwritten by } A) \\ \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} \text{Unknown part of } A \\ \left.\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right\} B = \text{Partly transformed } \sqrt{\gamma} I \end{array}$$

Figure 4.1

The kth reflection involves row k of A and R and rows $1, \ldots, k$ of B. See [18] for details. Here, we only mention that the kth diagonal element in A is computed by

$$a_{kk} = -\mathrm{sign}(r_{kk}) \sqrt{d_k}, \quad \text{where} \quad d_k = r_{kk}^2 - \sum_{i=1}^{k} b_{ik} \ . \tag{4.11a}$$

The matrix $H - \gamma I$ is significantly positive definite only if all d_k are significantly positive. We use the test

$$d_k > \left(\min\{2k, 20\} \cdot \varepsilon_{\mathrm{M}} \cdot r_{kk}\right)^2 \ . \tag{4.11b}$$

The total cost of the transformation (4.10b) is about $\frac{2}{3}n^3$ flops. This is the same as if we computed the upper triangle of $R^T R - \gamma I$ ($\frac{1}{3}n^3$ flops) followed by Cholesky factorization (also $\frac{1}{3}n^3$ flops).

If, for some $k \le n$ the condition (4.11b) is not satisfied, then the process is repeated with $\gamma := 0.1\gamma$. If this also fails, then the QP–algorithm gives an error return: The problem is too ill conditioned.

4.3. Starting Point

In (2.8) it was seen that if the sign vector $s = s(x_\gamma)$ were known, then we could find $x_\gamma = x_s$ as the solution to the linear system

$$(AWA^T + \gamma I)x_s = A(Wc - \gamma s) \ . \tag{4.12}$$

Therefore, we look for a good strategy for choosing the initial s for the Newton iteration in the dual algorithm.

We experimented with a number of strategies, [18], and settled for the algorithm given in (4.16) below. This is based on the relation (2.8) between the primal solution y^* and the dual sign vector $s(x_\gamma)$. This relation can be written as $s(x_\gamma) = S(-y^*, 1)$, where the generalized sign vector $S(v, \tau)$ is defined by

$$S_i(v, \tau) = \begin{cases} 0 & \text{if } |v_i| < \tau \\ \mathrm{sign}(v_i) & \text{otherwise} \end{cases} \ . \tag{4.13}$$

An approximation to y^* is found by considering the behaviour of the unconstrained minimizer of q,

$$\widetilde{y} = H^{-1}c \ . \tag{4.14}$$

This vector is easily computed, since we know the factorization (4.1). There are two extreme cases,

1° If $\|\widetilde{y}\|_\infty \le 1$, then $\widetilde{y}$ is the solution also for the constrained problem (1.1), $\widetilde{y} = y^*$. If $\|\widetilde{y}\|_\infty > 1$ but not too big, then $\widetilde{y}/\|\widetilde{y}\|_\infty$ is a good approximation to y^*. In this case $\|H\|_\infty$ and $\|c\|_\infty \simeq \|Hy^*\|_\infty$ are of the same order of magnitude.

2° If $\|c\|_\infty \gg \|H\|_\infty$, then a steepest descent direction from 0, $\widehat{y} = c/\|c\|_\infty$ is a better approximation to the primal solution y^*.

We use the following interpolation between these two extreme cases,

$$y := \frac{\alpha}{\|c\|_\infty}c + \frac{1-\alpha}{\|\widetilde{y}\|_\infty}\widetilde{y} \quad \text{with} \quad \alpha := 0.9 \cdot \left(1 - \frac{\mathcal{N}(-\widetilde{y}, 1)}{n}\right)^4 , \tag{4.15}$$

where the factor 0.9 and the exponent 4 were decided experimentally, and $\mathcal{N}(v, \tau)$ denotes the number of indices i, for which $S_i(v, \tau) = 0$.

The choice between different approximations y is decided by the number of elements in the dual active set for the corresponding starting vector $x = x_s$ with $s = S(-y, 1)$. We aim at having at least $\frac{1}{2}n$ elements in $\mathcal{A}(x)$. This is motivated by experience from similar algorithms for other problems, [13] and [15], and confirmed by experiments with the present problem. If $\mathcal{N}(r(x), \gamma)$ is too small, then the Newton iteration may have too slow initial convergence.

Now, the starting algorithm can be expressed as follows,

$$
\begin{array}{l}
\text{Compute } \widetilde{y} \text{ by (4.14)} \\
\textbf{if } \|\widetilde{y}\|_\infty \le 1 \textbf{ then} \\
\quad y^* := \widetilde{y}; \quad \textsc{Stop} \\
\textbf{end} \\
\textbf{if } \mathcal{N}(-\widetilde{y}, 1) \ge \frac{1}{2}n \textbf{ then} \\
\quad \text{Compute } x \text{ by (4.12) with } s = S(-\widetilde{y}, 1); \quad s := s(x) \\
\textbf{else} \\
\quad \text{Compute } y \text{ by (4.15)}; \quad s := S(-y, \tau_{n/2}) \\
\quad \text{Compute } x \text{ by (4.12)}; \quad \widehat{s} := s(x) \\
\quad \textbf{if } \mathcal{N}(r(x), \gamma) < \frac{1}{2}n \textbf{ then} \\
\quad\quad s := s \oplus \widehat{s}; \\
\quad\quad \text{Compute } x \text{ by (4.12)}; \quad s := s(x) \\
\quad \textbf{else} \\
\quad\quad s := \widehat{s} \\
\quad \textbf{end} \\
\textbf{end}
\end{array}
\tag{4.16}
$$

Here, τ_m denotes the mth smallest $|v_i|$, so that $\mathcal{N}(v, \tau_m) \ge m$, where $0 \le m \le n$. In the innermost part of the algorithm the vector $u := s \oplus \widehat{s}$ has elements $u_i = 0$ if $s_i = \widehat{s}_i = 0$ or $s_i = -\widehat{s}_i$, otherwise, $u_i = s_i = \widehat{s}_i$. This is equivalent with saying that an index is considered to belong to $\mathcal{A}(x)$ only if both $S(-y, \tau_{n/2})$ and $s(x) = S(r(x), \gamma)$ agree on this or if they show opposite sign.

Compared with the other strategies described in [18] we found that (4.16) generally gave the smallest number of iterations with the algorithm of Section 4.1. The dominant part of the computation is one factorization of $AWA^T + \gamma I$ and 2 or 3 updates (or refactorizations), cf. Section 4.4.

4.4. Factorization

During the iterations for computing the Huber solution we have to solve problems of the form

$$(AWA^T + \gamma I)h = -\gamma F'(x) \ , \tag{4.17}$$

where $W = \mathrm{diag}(w_i)$ with $w_i \in \{0, 1\}$. Let $A_{:,j}$ denote the jth column of the matrix A. Then we can write

$$AWA^T = A_{\mathcal{A}} A_{\mathcal{A}}^T \ ,$$

where $A_{\mathcal{A}}$ consists of the columns $\{A_{:,j}\}$ with $j \in \mathcal{A}$, the active set. Between iterations there is usually only a few changes in $\mathcal{A}$, and we are interested in a cheap (but accurate) updating of a triangular factorization of the coefficient matrix in (4.17).

We have chosen to use an untraditional factorization, viz.

$$AWA^T + \gamma I = L^T L \ , \tag{4.18}$$

where L is a lower triangular matrix. This implies that the solution of (4.17) is done by a back substitution followed by a forward substitution.

This choice is made because it leads to simple updatings: Let the active set be augmented by $\mathcal{E}$ (for "enter"). Then

$$\widetilde{L}^T \widetilde{L} = L^T L + A_{\mathcal{E}} A_{\mathcal{E}}^T = \begin{bmatrix} L^T & A_{\mathcal{E}} \end{bmatrix} \begin{bmatrix} L \\ A_{\mathcal{E}}^T \end{bmatrix} \ .$$

This shows that we can compute $\widetilde{L}$ by an orthogonal transformation

$$\begin{bmatrix} \widetilde{L} \\ 0 \end{bmatrix} = Q \begin{bmatrix} L \\ A_{\mathcal{E}}^T \end{bmatrix} \ . \tag{4.19}$$

The structure of the rightmost matrix is shown below for $n = 8$ and columns 2 and 5 entering the active set

$$\left[\begin{array}{cccccccc}
\star &&&&&&& \\
\star&\star &&&&&& \\
\star&\star&\star &&&&& \\
\star&\star&\star&\star &&&& \\
\star&\star&\star&\star&\star &&& \\
\star&\star&\star&\star&\star&\star && \\
\star&\star&\star&\star&\star&\star&\star & \\
\star&\star&\star&\star&\star&\star&\star&\star \\
\star&\star &&&&&& \\
\star&\star&\star&\star&\star &&&
\end{array}\right]
\begin{array}{l}
\left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\\1\\1\end{array}}\right\} L \\
\left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\} A_{\mathcal{E}}^T
\end{array}$$

Figure 4.2

As in Section 4.2 the transformation is done by a series of Householder reflections. Now, however, we start from $k = n$ and go back. For the example shown in figure 4.2 the first change occurs for $k = 5$. For $k = 5, 4, 3$ the transformations involve row k of L and the last row of $A_{\mathcal{E}}^T$. For $k = 2, 1$ the kth row of L and both rows of $A_{\mathcal{E}}^T$ are involved. We use standard Householder reflections in the implementation.

The changes in active set can also imply that columns of A leave (4.18). Therefore we are also interested in downdating the factorization; i.e. to compute

$$\widetilde{L}^T \widetilde{L} = L^T L - A_{\mathcal{L}} A_{\mathcal{L}}^T \ . \tag{4.20}$$

This is done using the update procedure and ideas from Section 4.2. Here, we check for severe loss of accuracy and signal an error return in that case. Remember that the columns of A that are removed have contributed to the current L.

Typically a change in active set involves that some columns of A leave and some enter. This means that we seek

$$\widetilde{L}^T\widetilde{L} = L^TL - A_{\mathcal{L}}A_{\mathcal{L}}^T + A_{\mathcal{E}}A_{\mathcal{E}}^T \ . \qquad (4.21)$$

The transformation is made in two steps. The details are discussed in [18].

For both up- and downdating the cost of entering (deleting) m columns to (from) the active set is between $\frac{2}{3}m^3$ flops (the first m columns) and $2mn^2$ flops (the last m columns). Thus, in the typical case, where $m_{\mathcal{E}}+m_{\mathcal{L}} \ll n$, the cost of adjusting the factorization is $O(n^2)$ flops.

4.5. Line Search

Given x and $r=r(x)$ and a search direction h, that satisfies

$$(AWA^T + \gamma I)h = -\gamma F'(x) = -A(Wr + \gamma s) - \gamma x \ .$$

We seek the minimum of the function $\varphi(t) = F(x+th)$. It is easily verified that φ' is a continuous, piecewise linear function, whose coefficients change at *kink values*, where one or more residual components pass the threshold γ. The kink values are the positive α_j–values defined by

$$|r_k + \alpha_j{\cdot}(A^Th)_k| = \gamma \quad \text{for some} \quad k = k_j \ .$$

The line search algorithm is similar to the algorithm of [12], and details can be found in [18].

5. Testing

The algorithm has been tested on a large number of problems generated as described in Section 5.1. In Section 5.2 we present three competing algorithms, and in Section 5.3 we give computational results.

5.1. Test Problems

The test problem generator is based on the Kuhn–Tucker condition

$$Hy^* - c + u = 0 \ , \qquad (5.1)$$

where $u_i \neq 0$ only if the ith constraint is primal active. This is the background for the widely used test problem generator of Moré and Toraldo [16]. The vector c is found as

$$c = Hy^* + u \ , \qquad (5.2a)$$

where H, y^* and u are generated so that a prescribed number of constraints are active and

$$H = 10^{-desc}H_0, \quad u_i = \begin{cases} 0 & \text{if } |y_i^*| < 1 \\ -y_i^* \cdot 10^{-\nu\cdot deg} & \text{otherwise} \end{cases} \ . \qquad (5.2b)$$

Here, ν is uniform random in [0,1], H_0 is a symmetric, positive definite matrix with $\|H_0\|_2 = 10^{ncond}$, and *desc*, *ncond*, *deg* are chosen, nonnegative parameter values. It follows that

$$\|y^*\|_\infty = 1, \quad \|u\|_\infty \leq 1 \ ,$$
$$\|Hy^*\|_\infty \leq \|H\|_\infty = 10^{-desc}\|H_0\|_\infty \sim 10^{ncond-desc} \ .$$

The original Moré–Toraldo generator corresponds to $desc=0$. In that case we are sure that Hy^* has a dominating influence on c, and the starting point given by $s=S(-\widetilde{y},1)$ (cf. Section 4.3), has been found to work well. The choice of $desc>0$ leads to more difficult problems.

The matrices have the form

$$H = M^T M \quad \text{with} \quad M = D^{\frac{1}{2}} Z \ , \tag{5.3a}$$

where

$$D = 10^{-desc}\text{diag}(d_1, \ldots, d_n) \quad \text{with} \quad \log_{10} d_i = \frac{i-1}{n-1} ncond \ . \tag{5.3b}$$

Here *desc* and *ncond* are prescribed numbers. Further, Z is a Householder matrix,

$$Z = I - \frac{2}{z^T z} z\, z^T \tag{5.3c}$$

where $z \in \mathbb{R}^n$ has elements that are uniform random in]-1,1[. Since Z is orthogonal, it follows that the condition number $\kappa_2(H) = \kappa_2(D) = 10^{ncond}$.

A little algebra shows that

$$H = D - 2(uv^T + vu^T), \tag{5.3d}$$

where

$$u = z/\|z\|_2, \quad v = Du - (u^T D\, u) u \ . \tag{5.3e}$$

The generator involves two more parameters:

nb Controls the number of “large” components in y^*, i.e. $|y_i^*| = 1$.

deg Controls near–degeneracy: The non active residuals are computed as $r_i = s_i \cdot 10^{-\nu \cdot deg}$, where ν is uniform random in $[0, 1]$.

5.2. Competing Methods

First, we compare with an interior point method. As a typical example we take the primal-dual approach used by Han et al. [6]. They use the standard formulation

$$\begin{aligned} &\min\{\ f(x) \equiv \tfrac{1}{2} x^T H x + d^T x\ \} \\ &\text{subject to} \ \ 0 \le x \le e\ . \end{aligned} \tag{5.4}$$

This is equivalent with our formulation (1.1) when we set

$$d = -\tfrac{1}{2}(c + He)\ , \tag{5.5a}$$

and

$$y = 2x - e, \quad q(y) = f(x) + \tfrac{1}{2} e^T (2c + He)\ . \tag{5.5b}$$

The method of Han et al. is derived from the reformulation

$$\begin{aligned} &\min\{f(x) \equiv \tfrac{1}{2} x^T H x + d^T x\} \\ &\text{subject to} \ \ x + z = e \quad \text{and} \quad x, z \ge 0\ , \end{aligned} \tag{5.6a}$$

and the dual

$$\begin{aligned} &\max\{-e^T v - \tfrac{1}{2} x^T H x\} \\ &\text{subject to} \ \ v \ge 0 \quad \text{and} \quad u \equiv Hx + d + v \ge 0\ . \end{aligned} \tag{5.6b}$$

The duality gap is

$$\Delta \equiv x^T u + z^T v \ge 0\ . \tag{5.6c}$$

The solution is found by minimizing the potential function

$$\varphi(x, z, u, v) \equiv \rho \log(\Delta) - \sum_{i=1}^{n} \log(x_i u_i) - \sum_{i=1}^{n} \log(z_i v_i)\ , \tag{5.7}$$

where the scalar $\rho \ge 2n + \sqrt{2n}$.

Let (x,z,u,v) denote the current iterate, and let X,Z,U,V denote the diagonal matrices with $(X)_{ii}=x_i$ etc. The next iterate is found as follows,

1° Find h_x as the solution to

$$(DHD+UZ+VX)(D^{-1}h_x) = \tfrac{\Delta}{\rho}(DX^{-1}-DZ^{-1})e - D(u-v) \tag{5.8}$$

with $D=\mathrm{diag}(\sqrt{x_i z_i})$, and compute

$$h_v = Z^{-1}(Vh_x + \frac{\Delta}{\rho}e) - v\ .$$

2° Line search: Find $\overline{\theta}$, the largest value for which

$$x+\theta h_x \ge 0,\quad z-\theta h_x \ge 0,\quad v+\theta h_v \ge 0 \quad \text{and} \quad u+\theta(Hh_x+h_v)\ge 0\ .$$

3° Update:

$$\theta = \beta\overline{\theta}$$
$$x := x+\theta h_x;\qquad z := e-x;\quad v := v+\theta h_v;\quad u := Hx+d+v$$
$$\Delta := x^T u + z^T v$$

The iteration is stopped when

$$\Delta \le \min\{\epsilon_1, \epsilon_2(1+|f(x)|)\}\ . \tag{5.9a}$$

The algorithm involves four iteration parameters: ρ, β, ϵ_1 and ϵ_2. Han et al. [6] found that the choices $\rho=n^{1.5}$, $\beta=0.99$ were close to optimal, and these are the values used in our comparisons. Further, they recommend to use the stopping parameters: $\epsilon_1=10^{-5}$, $\epsilon_2=10^{-8}$. In an attempt to get more accurate results we use $\epsilon_1=10^{-8}$, $\epsilon_2=10^{-12}$, but have found that often this results in an infinite loop: After a certain number of steps the computed value for Δ is not decreased further. To cure this problem we supplement the stopping criterion (5.9a) with

$$\textbf{if}\ \Delta_{\text{new}} \ge \Delta_{\text{old}}\ \textbf{then}\ \text{STOP}\ . \tag{5.9b}$$

We also compare with a Simplex type method, viz. the `bqpd` package of Fletcher (1993). This package has a broader range of applications. The option used in our comparisons addresses a generalized version of (1.1),

$$\begin{aligned}&\min_y\{\ q(y) \equiv \tfrac{1}{2}y^T Hy - c^T y\ \}\\ &\text{subject to}\ \ b_l \le y \le b_u \ \ \text{and}\ \ \widehat{b}_l \le Gy \le \widehat{b}_u\ ,\end{aligned} \tag{5.10}$$

where G is an $m\times n$ matrix, and $\widehat{b}_l$, $\widehat{b}_u$ are m-vectors. By choosing $m=0$ and $b_l=-e$, $b_u=e$ we see that (5.10) is identical with (1.1).

The method used is an active set strategy, and demands an initial, feasible point $y_{(0)}$. In the comparisons we use $y_{(0)}=0$. Further, we use the parameter values

tol	*tolmin*	*fmin*	*nrep*	*npiv*
10^{-10}	10^{-14}	$-9.0\cdot 10^{99}$	2	3

Finally, we compare with the algorithm of Li and Swetits [10]. They treat problem (1.1) with general box constraints, $l\le y\le u$. The method is based on minimizing the following convex quadratic spline

$$\Phi(x) = \tfrac{1}{2}x^T Bx - \tfrac{1}{2}\|(\rho(x))_l^u\|_2^2 - l^T(\rho(x))^l - u^T(\rho(x))_u\ , \tag{5.11a}$$

where

$$B = I-\alpha H \quad \text{with} \quad 0<\alpha<\|H\|_2^{-1}\ , \tag{5.11b}$$

$$\rho(x) = x - \alpha(Hx - c) \ , \tag{5.11c}$$

and $(z)^w$ (or $(z)_w$) is the vector whose ith component is $\max\{z_i, w_i\}$ (or $\min\{z_i, w_i\}$). The gradient of Φ is the piecewise linear function

$$\Phi'(x) = B(x - (\rho(x))_l^u) \ ,$$

and Li and Swetits use Newton's method with line search to find the minimizer x^*: $\Phi'(x^*) = 0$. The Newton direction h is found by solving

$$(B - BDB)h = -\Phi'(x) \ , \tag{5.12}$$

where $D = \mathrm{diag}(d_1, \ldots, d_n)$ with $d_i = 1$ if $l_i \le \rho_i(x) \le u_i$, otherwise $d_i = 0$. The iteration is started with $x = \frac{1}{2}(l + u)$.

It is interesting to note some similarities between our method and the method of Li and Swetits [10]:

1° We have to compute γ so that $0 < \gamma < \min\{\lambda_j(H)\}$, and Li and Swetits must find α so that $0 < \alpha < \|H\|_2^{-1} = 1/\max\{\lambda_j(H)\}$. They use $\alpha = \|H\|_\infty^{-1}$ with a cost of about n^2 flops, i.e. about one quarter of the cost of computing γ, cf. Section 4.1.

2° Both methods operate with a dual active set: D in (5.12) is equivalent with W in (4.17). As described in Section 4.4 we use an efficient updating of the factorization. Li and Swetits [10] use Cholesky factorization of $B - BDB$ in each step of the iteration. The possibility of updating the factorization is mentioned in [11], dealing with general linear constraints. There is no specific indication of how it should be done, however.

5.3. Computational Results

We implemented the new algorithm and the algorithm of Han et al. in Fortran77 with extensive use of BLAS, [3]. We used Fletcher's own Fortran77 implementation of `bqpd` except that we changed it to double precision. Also for the method of Li and Swetits [10] we used their own implementation, `simpbd`.

The tests we performed on an HP9000/800–K460, and timings were done with the `-O` option of the `f77` compiler. The machine accuracy is $\varepsilon_{\mathrm{M}} = 2^{-52} \simeq 2.22 \cdot 10^{-16}$.

Below we give results for varying values of the parameters of the problem generator described in Section 5.1. In e.g. [6] such results are presented in the form of tables, where each entry is the average over 10 problems with fixed value of the parameters, with a few, selected values of the parameter under discussion (e.g. the size of the problem with $n = 100, 200, \ldots, 500$). Instead, we have chosen to show a "more continuous" variation ($n = 100, 110, \ldots, 500$) with one instance of each. This presentation illustrates both the influence of the parameter and the stochasticity in the problem generation.

As regards accuracy of the computed results, we introduce

$$q_{err} = \frac{|q(y) - q(y^*)|}{|q(y^*)|} \quad \text{and} \quad y_{err} = \|y - y^*\|_\infty \ , \tag{5.13}$$

where y^* is the solution generated as described in Section 5.1 and y is the computed solution. Since $\|y^*\|_\infty = 1$, both q_{err} and y_{err} are relative errors.

In Figures 5.1–5 we use the symbols

o Results from `pqf`: the new method based on minimizing a piecewise quadratic function, and described in Section 4.

x Results from `hpy`: the interior point method of Han et al. [6] as described in Section 5.2.

+ Results from `bqpd`: the Simplex type method of Fletcher [4] as described in Section 5.2. Here "iterations" is $\frac{1}{40}$(the number of Simplex bases). (The scaling factor 40 was chosen to get nice figures).

* Results from `lisw`: the Newton method of Li and Swetits [10] as described in Section 5.2.

First, we show the influence of the size of the problem. Note, that for `pqf`, `hpy` and `lisw` the number of iterations is almost constant, while in `bqpd` the number of Simplex steps seems to grow linearly with n. In none of the cases a refactorization was needed during the iterations in `pqf`.

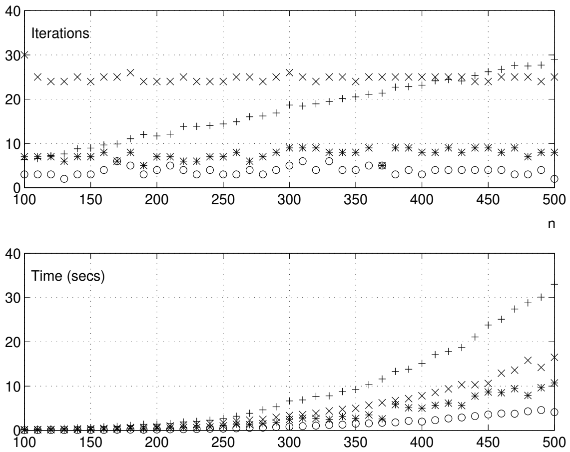

Figure 5.1. *Varying size. (ncond, deg, nb, desc) = (3, 1, 50%, 0).*

In all the cases reported in Figure 5.1 we found that `pqf`, `bqpd` and `lisw` all gave $q_{err} \simeq 10^{-16}$ and $y_{err} \simeq 10^{-15}$, i.e. full precision. With `hpy` we found $q_{err} \simeq 10^{-10}$ and y_{err} in the range $[10^{-7}, 10^{-3}]$. Interior point methods for LP problems use "extrapolation" to the boundary, see e.g. [1, Section 7]. Similarly, from the results of the interior point method it should be possible to identify the primal active set and find a more accurate solution. Han et al. [6] do not consider this kind of "extrapolation", however.

Next, in Figure 5.2 we consider the influence of the condition number $\kappa_2(H)$. Here, the number of iterations grows slightly with *ncond* for `pqf`, `hpy` and `lisw`, but is constant for `bqpd`. At most one refactorization was used by `pqf`, and it is seen that the increasing number of iterations is not reflected in the computing time. Each iteration in the dual algorithm is an $O(n^2)$ proces, wheras each iteration with `hpy` and `lisw` is an $O(n^3)$ proces, cf. (5.8) and (5.12).

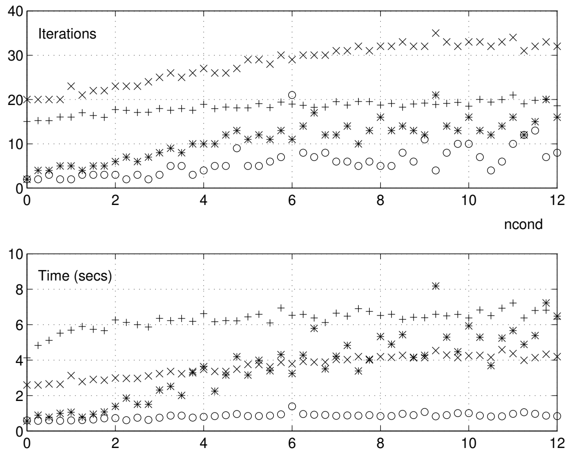

Figure 5.2a. *Varying condition. Iterations and timing.*
(n, deg, nb, desc) = (300, 1, 50%, 0).

In Figure 5.2b we show the accuracy obtained. `pqf`, `bqpd` and `lisw` all determine the minimum value of q with a relative error which is small multiple of the machine accuracy, and the error in the computed y grows proportional with $\kappa_2(H)$, which is to be expected. The results from `lisw` could probably be improved by one final step of iterative refinement. The results from `hpy` are orders of magnitude worse.

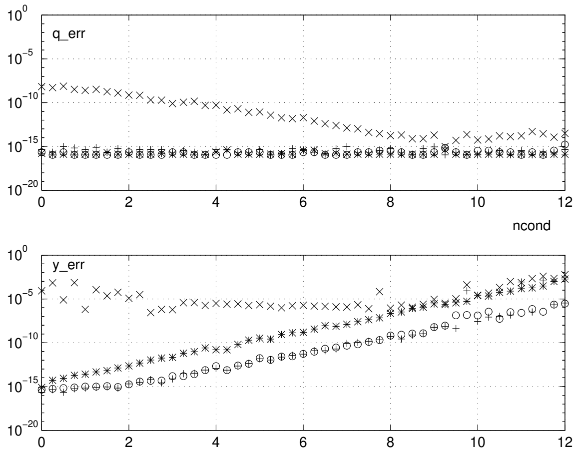

Figure 5.2b. *Varying condition. Accuracy.*
(n, deg, nb, desc) = (300, 1, 50%, 0).

In Figure 5.3 we give results for the influence of the parameter *deg*, i.e. how well active are distinguished from non active equations. Both `hpy` and `bqpd` show very little sensitivity, whereas the the number of `pqf`– and `lisw`–iterations grows slightly with *deg*. In none of the cases a refactorization was needed in `pqf`, and the accuracy is as described in connection with Figure 5.1.

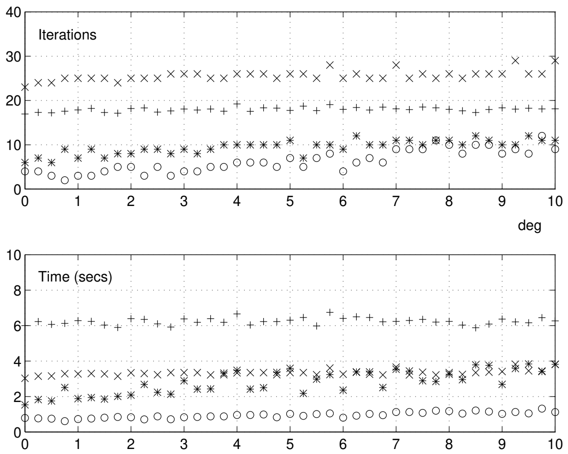

Figure 5.3. *Varying "near-degeneracy".*
(n, cond, nb, desc) = (300, 3, 50%, 0).

Figure 5.4 shows the effect of the parameter *nb*, i.e. the number of active constraints. The number of iterations seems to be independent of *nb* for `hpy`, but grows slightly with this parameter for `pqf` and `bqpd`, although it does not reflect in the timings. In 8 (3) of the 41 cases shown one (two) refactorizations were used in `pqf`. For `lisw` the number of iterations is almost constant, but the computing time increases significantly when *nb* decreases, i.e. when the number of nonzero diagonal elements in D, (5.12), increases.

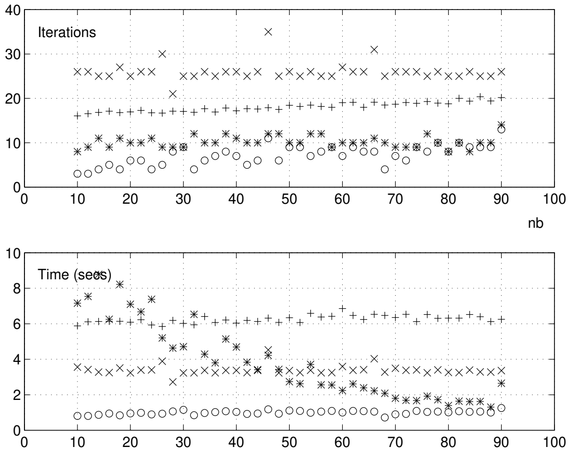

Figure 5.4. *Varying number of active constraints.*
(n, cond, deg, desc) = (300, 3, 6, 0).

Finally, Figure 5.5a–b shows the influence of the descaling factor. Here, `bqpd` needs an almost constant number of iterations, and its timings show a marked decrease as *desc* grows. The interior point method `hpy` also performs faster for increasing *desc,* while `pqf` needs considerably more iterations (and up to three refactorizations). This reflects in the computing times, which are up to 0.7 times the `hpy`-time. Refering to the discussion in Section 4.3 we see that `lisw` performs well both when $\|H\|_\infty$ and $\|c\|_\infty$ are of the same order of magnitude ($desc \le 2$) and when $\|H\|_\infty \ll \|c\|_\infty$ ($desc \ge 5$). In the range $2 < desc < 5$ the number of iterations and the computing time increases significantly.

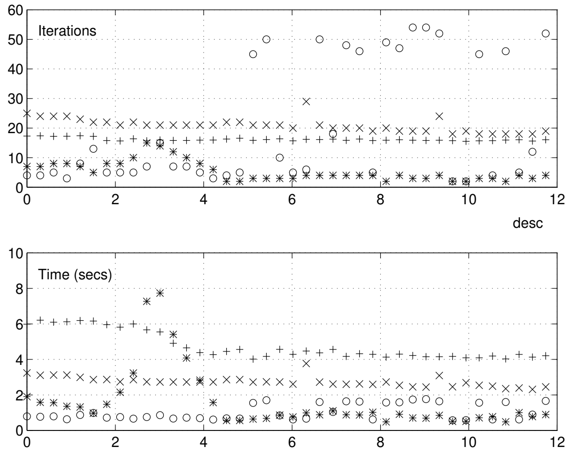

Figure 5.5a. *Varying descaling. Iterations and timing.*
(n, cond, deg, nb) = (300, 3, 1, 50%).

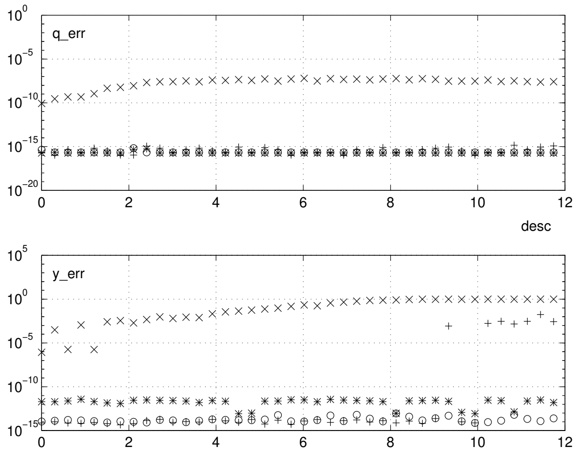

Figure 5.5b. *Varying descaling. Accuracy.*
(n, cond, deg, nb) = (300, 3, 1, 50%).

As regards accuracy, Figure 5.5b shows that it is doubtful whether the solution can be obtained by "extrapolation" from the results of `hpy`: For $desc \geq 9$ we find $0.967 \leq y_{err} \leq 1$. Also `bqpd` has trouble finding y with full accuracy, while the extra effort pays off with `pqf`, and `lisw` supplemented with a step of iterative refinement would also give the solution to full accuracy.

6. Conclusion

We have described a new method for solving quadratic programming problems with unit constraints. Careful attention to computational details has led to an efficient and accurate algorithm that compares favourably with both interior point and Simplex type methods.

The method is easily modified to non unit box constraints, and we are currently working on a sparse implementation of the algorithm (together with Wolfgang Hartmann from SAS). The results of the sparse code will be reported elsewhere. Future projects include modification of the algorithm to general linear constraints.

7. Acknowledgements

The authors are grateful to Professor Roger Fletcher and Professors Wu Li and John Swetits for making their programs available to us.

References

[1] E.D. Andersen, J. Gondzio, C. Mészáros and X. Xu. 1996. *Implementation of Interior Point Methods for Large Scale Linear Programming.* Chapter 6 (pp 189–252) in T. Terlaky (ed.) *Interior Point Methods in Mathematical Programming,* Kluwer Academic Publishers.

[2] T. Coleman and L. Hulbert. 1993. *A Globally and Superlinearly Convergent Algorithm for Quadratic Programming with Simple Bounds,* SIAM J. on Optimization **3**, 298–321.

[3] J. Dongarra, C.B. Moler, J.R. Bunch and G.W. Stewart. 1988. *An Extended Set of Fortran Basic Linear Algebra Subprograms,* ACM Trans. Math. Soft. **14**, 1–17.

[4] R. Fletcher. 1993. *Resolving Degeneracy in Quadratic Programming,* Annals of OR **47**, 307–334.

[5] N.J. Higham. 1987. *A Survey of Condition Number Estimation for Triangular Matrices,* SIAM Review **29**, 575–596.

[6] C-G. Han, P. Pardalos and Y. Ye. 1990. *Computational Aspects of an Interior Point Algorithm for Quadratic Programming Problems with Box Constraints,* pp 92–112 in T. Coleman and Y. Li (eds), *Large Scale Numerical Optimization*, SIAM, Philadelphia.

[7] P. Huber. 1981. *Robust Statistics.* John Wiley, New York.

[8] W. Li. 1995. *Linearly convergent descent algorithms for strictly convex quadratic programming,* J. of Optimization Theory and Appl., to appear.

[9] W. Li. 1995. *Conjugate gradient algorithms for convex quadratic splines,* Math. Programming, to appear.

[10] W. Li and J. Swetits. 1993. *A Newton method for convex regression, data smoothing, and quadratic programming with bounded constraints,* SIAM J. on Optimization **3**, 466–468.

[11] W. Li and J. Swetits. 1995. *A new algorithm for solving strictly convex quadratic programs,* SIAM J. on Optimization, to appear in 1997.

[12] K. Madsen and H.B. Nielsen. 1990. *Finite Algorithms for Robust Linear Regression,* BIT **30**, 682–699.

[13] K. Madsen, H.B. Nielsen. 1993. *A Finite Smoothing Algorithm for Linear L_1 Estimation,* SIAM J. on Optimization **3**, 223–235.

[14] K. Madsen, H. B. Nielsen and M.Ç. Pınar. 1995. *A New Finite Continuation Algorithm for Bound Constrained Quadratic Programming.* Report IMM-REP-95-22, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[15] K. Madsen, H.B. Nielsen and M.Ç. Pınar. 1996. *A New Finite Continuation Algorithm for Linear Programming,* SIAM J. on Optimization **6**, 600–616.

[16] J. Moré and G. Toraldo. 1989. *Algorithms for Bound Constrained Quadratic Programming Problems,* Numer. Math. **55**, 377–400.

[17] H.B. Nielsen. 1991. *Implementation of a Finite Algorithm for linear ℓ_1 Estimation,* Report NI-91-01, Institute for Numerical Analysis, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[18] H.B. Nielsen. 1996. *Bound Constrained Quadratic Programming Solved via Piecewise Quadratic Functions: Implementation.* Report IMM–1996–21, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark. Available as http://www.imm.dtu.dk/~hbn/publ/TR9621.ps

[19] R.T. Rockafellar. 1970. *Convex Analysis.* Princeton University Press, Princeton, New Jersey.