An infeasible active set method for convex problems with simple bounds

K. Kunisch *
Universität Graz
Institut für Mathematik
A-8010 Graz, Austria
karl.kunisch@kfunigraz.ac.at

F. Rendl
Universität Klagenfurt
Institut für Mathematik
A-9020 Klagenfurt, Austria
franz.rendl@uni-klu.ac.at

August 1, 2000

Key words: Primal-dual active Set Method, Convex Programming.
AMS Subject Classification: 90C06,20,99.

Abstract

A primal-dual active set method for convex quadratic problems with bound constraints is presented. Based on a guess on the active set, a primal-dual pair (x, s) is computed that satisfies the first order optimality condition and the complementarity condition. If (x, s) is not feasible, a new active set is determined, and the process is iterated. Sufficient conditions for the iterations to stop in a finite number of steps with an optimal solution are provided. Computational experience indicates that this approach often requires only a few (less than 10) iterations to find the optimal solution.

1 Introduction

We consider the convex programming problem

$$(P) \qquad \min J(x) \text{ subject to } x - b \leq 0, \tag{1}$$

where

$$J(x) := \frac{1}{2}x^T Q x + d^T x,$$

Q is a positive definite $n \times n$ matrix, and $b, d \in \mathbb{R}^n$. This problem has received considerable interest in the literature. We recall some of the more recent contributions.

*Supported in part by the Fonds zur Förderung der wissenschaftlichen Forschung (FWF), Austria, under SFB 03 "Optimierung und Kontrolle".

Solution methods based on active sets and gradient projection are among the most popular approaches to solve (P), and can be traced back to the 1960's. More recent contributions from Moré and Toraldo [13, 14] indicate that this approach is applicable also for large-scale problems. The key steps here consist in using the conjugate gradient method to investigate a given face of the feasible region, and the gradient projection method to move to a different face. Kočvara and Zowe [11] replace the gradient projection by successive over-relaxation with projection (SORP), thereby gaining efficiency as compared to [14].

Another solution strategy consists in treating the inequalities by the interior-point idea: a sequence of parameterized barrier functions is (approximately) minimized using Newton's method. The main computational effort consists in solving the Newton system to get the search direction. From the vast literature on this topic, we refer to the book by Wright [15]. More recently, Heinkenschloss et al. [9] developed an affine-scaling interior-point approach to general nonlinear bound-constrained problems, which does not assume strict complementarity to hold at local solutions. D'Appuzo et al. [6] present a parallel implementation of an interior-point method for box-constrained quadratic programming.

Finally, trust-region type methods have also been investigated to deal with bound-constrained problems. We refer to Coleman, Li and Liu [4, 5] and Lin and Moré [12] for further details.

Our contribution to solve (P) consists in an infeasible active-set approach that was already successfully applied to constrained optimal control problems, see [2, 1]. The approach from [2, 1] was taylored to deal with specially structured elliptic partial differential equations. Here we investigate this approach for general convex quadratic problems of the form (P). Our approach is iterative. In each step we maintain the first order optimality condition and the complementarity constraint associated to (P), see (2) and (3) below. The iterations are carried out until primal and dual feasibility hold, see (4) and (5) below. Since the proposed method is conceptually very simple, we include a short MATLAB routine in the appendix. Computational experience with our method indicates that typically only few (often only between 5 and 10) iterations are required to reach the optimal solution starting from an arbitrarily chosen initial active set. We succeed to provide a rigorous foundation for this most important property of the algorithm by providing sufficient conditions for finite step convergence.

The paper is organized as follows. We close this section with some notation used throughout. In section 2 we describe the details of our algorithm. The main theoretical contributions are contained in section 3, which presents sufficient conditions for the method to converge in a finite number of iterations. The practical performance is described in Section 4. We consider both randomly generated problems and problems arising in mathematical physics. Specifically, we look at boundary value problems of the discretized biharmonic equation in the unit-square. In the appendix we include a short MATLAB implementation of our method.

Notation: The following notation will be used throughout. For a subset $A \subseteq N := \{1, \ldots, n\}$ and $x \in \mathbb{R}^n$ we write x_A for the components of x indexed by A, i.e. $x_A := (x_i)_{i \in A}$.

The complement of A will be denoted by $\bar{A}$. If Q is a matrix and A and B are subsets of N, then $Q_{A,B}$ is the submatrix of Q, with rows indexed by A and columns indexed by B. If $A = B$ we write Q_A for $Q_{A,A}$. The vector of ones will be denoted by e. For $a, b \in \mathbb{R}^n$ we write $a \circ b$ to denote the vector of element-wise products, $a \circ b := (a_i b_i)_{i \in N}$.

2 The Algorithm

To describe the algorithm that will be investigated analytically and computationally let $b, d \in \mathbb{R}^n$ and $Q = Q^T$ be given, with Q a positive definite $n \times n$ matrix. We consider the *convex quadratic minimization problem* with simple bound constraints (1).

Even though we could assume without loss of generality that $b = 0$, we prefer to maintain a general upper bound on x. The Karush-Kuhn-Tucker (KKT) system for (P) is given by

$$(KKT) \qquad \begin{aligned} Qx + d + s &= 0 && (2)\\ s \circ (x - b) &= 0 && (3)\\ x - b &\le 0 && (4)\\ s &\ge 0. && (5) \end{aligned}$$

It is well known that a vector x together with a vector $s \in \mathbb{R}^n$ of Lagrange multipliers for the inequality constraints furnishes a (global) minimum of (P) if and only if (x, s) satisfies the KKT system.

We now describe in some detail the approach sketched in the introduction. The crucial step in solving (P) is to identify those inequalities which are active, i.e. the set $A \subseteq N$, where the solution to (P) satisfies $x_A = b_A$. Then, with $I := N \backslash A$, we must have $s_I = 0$.

To compute the remaining elements x_I and s_A of x and s, we use (2) and partition the equations and variables according to A and I:

$$\begin{pmatrix} Q_A & Q_{A,I} \\ Q_{I,A} & Q_I \end{pmatrix} \begin{pmatrix} x_A \\ x_I \end{pmatrix} + \begin{pmatrix} d_A \\ d_I \end{pmatrix} + \begin{pmatrix} s_A \\ s_I \end{pmatrix} = 0 \qquad (6)$$

The second set of equations can be solved for x_I, because Q_I is by assumption positive definite:

$$x_I = -Q_I^{-1}(d_I + Q_{I,A}\, b_A).$$

Substituting this into the first equation implies $s_A = -d_A - Q_{A,N}\, x$. If our guess for A would have been correct, then $x_I \le b_I$ and $s_A \ge 0$ would have to hold. Suppose this is not the case. Then we need to make a new 'guess' for A, which we denote by A^+. Let us first look at s_A. If $s_i > 0$, this confirms our previous guess $i \in A$, so we include i also in A^+. Consider now x_I. If $x_i > b_i$ we set $x_i = b_i$ in the next iteration. Hence we include i in A^+ also in this case. Formally we arrive at

$$A^+ := \{i : x_i > b_i \text{ or } s_i > 0\}. \qquad (7)$$

Prototype Algorithm

Input: Q symmetric, positive definite $n \times n$ matrix, $b, d \in \mathbb{R}^n$. $A \subseteq N$, e.g. $A = N$.
Output: (x, s) optimal solution

repeat until (x,s) is optimal
  Solve KKT(A), i.e. set $x_A = b_A,\ s_I = 0$ and compute x_I from (8) and s_A from (9);
  $A := \{i : x_i > b_i \text{ or } s_i > 0\}$;

Table 1: Description of the algorithm

This completes an intuitive description of one iteration of the algorithm. It is summarized in Table 1. To simplify notation we introduce for given A the following set $KKT(A)$ of equations:

$$KKT(A) \qquad\qquad Qx + d + s = 0, \quad x_A = b_A, \quad s_I = 0.$$

The solution of $KKT(A)$ is given by $x_A = b_A,\ s_I = 0$ and

$$x_I = -Q_I^{-1}(d_I + Q_{I,A} b_A), \tag{8}$$
$$s_A = -d_A - Q_A\, b_A - Q_{A,I} x_I. \tag{9}$$

We observe that if (x, s) satisfies $KKT(A)$, then (2) and (3) of KKT hold. The iterates of the algorithm are well defined, because $KKT(A)$ has a unique solution for every $A \subseteq N$, due to $Q \succ 0$.

In Lemma 2.1 it is guaranteed that the set A changes in each iteration unless the algorithm stops. Hence the algorithm does not get trapped by generating the same (x, s) in two consecutive iterations.

Lemma 2.1 *Let A and A^+ be the active sets in two consecutive iterations of the algorithm. Then either the current primal and dual variables satisfy (2)–(5) or $A \neq A^+$.*

Proof. Let A and A^+ be the active sets in two consecutive iterations. Suppose that $A = A^+$ and let $i \in A^+$. Then either $s_i > 0$ or $x_i > b_i$. But $x_i = b_i$ for $i \in A$, and $A = A^+$, so we can not have $x_i > b_i$, and hence $x \le b$. Therefore $i \in A^+$ implies that $s_i > 0$, i.e. $s_{A^+} > 0$. But we also have $s_I = s_{I^+} = 0$ because $A = A^+$, so $s \ge 0$. Therefore the solution of $KKT(A)$ is optimal, and the algorithm would have stopped before generating A^+. ∎

Remark 2.2 *To guess the set A at the start, several obvious strategies could be employed. Using $A = N$, we get $x = b$ and $s = -(Qb + d)$. Setting $A = \emptyset$ gives $s = 0$ and $x = -Q^{-1}d$. In the latter case a linear system of order n has to be solved, which may be expensive for large-scale sparse problems. Alternatively, A may be selected at random.*

	s	t	x	y
T	$=0$		$> b$	$= b$
S	≤ 0	$=0$	$= b$	
$I \backslash T$	$=0$	$=0$	$\leq b$	
$A \backslash S$	> 0		$= b$	$= b$

Table 2: Partition of index set N

3 Convergence Analysis

3.1 Index Partition

To investigate the convergence behaviour of the algorithm, we look at two consecutive iterations. Suppose that some iteration, say $k \geq 1$, is carried out with the set $A^k \subseteq N$, yielding $(x^{(k)}, s^{(k)})$ as the solution of $KKT(A^{(k)})$. According to (7), the new active set is

$$A^{(k+1)} = \{i : x_i^{(k)} > b_i \text{ or } s_i^{(k)} > 0\}.$$

Let $(x^{(k+1)}, s^{(k+1)})$ denote the solution of $KKT(A^{(k+1)})$. To avoid too many superscripts, we write

$$(A, x, s) \text{ for } (A^{(k)}, x^{(k)}, s^{(k)}) \text{ and } (B, y, t) \text{ for } (A^{(k+1)}, x^{(k+1)}, s^{(k+1)}).$$

Given A, we find that x, s, B, y, t are determined by

$$x_A = b_A, \; s_{\bar{A}} = 0, \; Qx + d + s = 0 \tag{10}$$

$$B = \{i : x_i > b_i \text{ or } s_i > 0\} \tag{11}$$

$$y_B = b_B, \; t_{\bar{B}} = 0, \; Qy + d + t = 0. \tag{12}$$

The following partition of N into mutually disjoint subsets will be useful in our convergence analysis. We first partition A into

$$S := \{i \in A : s_i \leq 0\}$$

and $A \backslash S$. The set I is partitioned into

$$T := \{i \in I : x_i > b_i\}$$

and $I \backslash T$. In Table 2 we summarize the relevant information about x, s, y, t for this partition. The column for x indicates that $x_T > b_T, x_S = b_S$ and so on. A nonspecified entry, for instance y_S, indicates that y_S is in no specific relation to b_S. Finally, let $K_1 := \{i \in S : y_i > b_i\}$, $K_2 := \{i \in I \backslash T : y_i > b_i\}$, and let $K := K_1 \cup K_2$. The set K contains the indices of primal infeasibility of y.

3.2 Merit Function and Finite Step Convergence

We recall the well-known [3] augmented Lagrangian merit function for (P) given by

$$\mathcal{L}_c(x,s) = J(x) + s^T \hat{g}_c(x,s) + \frac{c}{2}\|\hat{g}_c(x,s)\|^2,$$

where $c > 0$ and $\hat{g}_c(x,s) = \max(g(x), -\frac{s}{c})$, with $g(x) = x - b$ and the max–operation acting componentwise. Here we shall employ a variation of $\mathcal{L}_c(x,s)$ given by

$$L_c(x,s) = \mathcal{L}_c(x, \max(0,s)).$$

Let us note that

$$L_c(x,s) = J(x) + \frac{c}{2}\|g(x)\|^2,$$

provided that (x,s) satisfies the complementarity condition $s \circ (x-b) = 0$, which is the case for the iterates of our algorithm. In the remainder of this section we shall establish sufficient conditions for the decay of L_c along the iterates of the algorithm, i.e.

$$L_c(y,t) - L_c(x,s) < 0$$

holds for any two consecutive pairs (x,s) and (y,t). In particular this implies convergence of the algorithm in finitely many steps. We start with some preliminary results. First, we investigate how the objective function changes during consecutive iterations. One can not expect a monotone decrease of $J(x)$ as the iterates may be infeasible.

Lemma 3.1 *Let (x,s), (y,t) and T be given as above. Then, we have*

$$J(y) - J(x) = \frac{1}{2}(y-x)^T \begin{pmatrix} Q_T & 0 \\ 0 & -Q_{\bar{T}} \end{pmatrix} (y-x).$$

Proof. We use the Q-inner product, $\langle a,b\rangle_Q := a^T Q b$, with associated norm $\|a\|_Q^2 := \langle a,a\rangle_Q$ and get

$$J(y) - J(x) = \frac{1}{2}\|y\|_Q^2 - \frac{1}{2}\|x\|_Q^2 + z^T d,$$

where $z = y - x$. Using the following identity

$$\|a\|_Q^2 - \|b\|_Q^2 = 2\langle a-b, a\rangle_Q - \|a-b\|_Q^2,$$

the right hand side can now be rewritten to obtain

$$J(y) - J(x) = -\frac{1}{2}z^T Q z + z^T(Qy + d) = -\frac{1}{2}z^T Q z - z^T t.$$

The last equation follows from $Qy + d = -t$. Now $t_i = 0$ for $i \in S \cup (I \backslash T)$ and $z_i = 0$ on $A \backslash S$. Therefore $z^T t = \sum_{i \in T} z_i t_i$. Furthermore $t - s = -Qz$ and $t_i - s_i = t_i$ for $i \in T$, and hence

$$-\sum_{i \in T} z_i t_i = \sum_{i \in T} z_i (Qz)_i = z^T \begin{pmatrix} Q_{T,N} \\ 0 \end{pmatrix} z.$$

Summarizing, we see that

$$J(y) - J(x) = -\frac{1}{2} z^T \begin{pmatrix} Q_T & Q_{T,\bar{T}} \\ Q_{\bar{T},T} & Q_{\bar{T}} \end{pmatrix} z + z^T \begin{pmatrix} Q_T & \frac{1}{2}Q_{T,\bar{T}} \\ \frac{1}{2}Q_{\bar{T},T} & 0 \end{pmatrix} z.$$

■

Lemma 3.2 *Let (x, s), (y, t) as well as T and K be given as above. Then we have*

$$\|g(y)\|^2 - \|g(x)\|^2 = \sum_{i \in K} |y_i - b_i|^2 - \sum_{i \in T} |x_i - b_i|^2.$$

Proof. The claim follows from the fact that x is infeasible precisely on T, see Table 2. Moreover, by the definition of the sets K_1 and K_2, the variable y is infeasible on K. ■

In summary we have proved the following result.

Proposition 3.3 *For every two consecutive pairs (x, s) and (y, t) we have*

$$L_c(y, t) - L_c(x, s) = \frac{1}{2} z^T \begin{pmatrix} Q_T & 0 \\ 0 & -Q_{\bar{T}} \end{pmatrix} z + \frac{c}{2} \sum_{i \in K} |y_i - b_i|^2 - \frac{c}{2} \sum_{i \in T} |x_i - b_i|^2.$$

■

We can now state the following sufficient conditions (C1) and (C2) for decrease of the merit function. Some more notation is required. Let $\mu := \lambda_{\min}(Q) > 0$ denote the smallest eigenvalue of Q. Further, let

$$\nu := \max\{\|Q_{A,\bar{A}}\| : A \subset N, A \neq 0, A \neq N\}.$$

We also use the diagonal matrix $D := \mathrm{diag}(q_{11}, \ldots, q_{nn})$ consisting of the main diagonal elements of Q and define

$$q := \min\{q_{ii} : i \in N\}.$$

Finally let $r := \|Q - D\|$ denote the norm of Q with the elements from the main diagonal removed.

$$\text{condition (C1)} \qquad \mathrm{cond}(Q) < \left(\frac{\mu}{\nu}\right)^2 - 1 \tag{13}$$

$$\text{condition (C2)} \qquad \mathrm{cond}(Q) < \left(\frac{q}{r}\right)^2 - 1, \tag{14}$$

where $\mathrm{cond}(Q) = \frac{\lambda_{\max}(Q)}{\lambda_{\min}(Q)}$. We will show that either of (C1) and (C2) insures strict decrease of the merit function L_c.

Theorem 3.4 *Let $(x, s), (y, t)$ be two consecutive primal–dual iterates of the algorithm, and $c = \|Q\| + \mu$. If (C1) holds, then we have*

$$2(L_c(y, t) - L_c(x, s)) \leq c_1 \|y - x\|^2 < 0,$$

with $c_1 := (\|Q\| + \mu)(\frac{\nu}{\mu})^2 - \mu < 0$. *Similarly, (C2) implies that*

$$2(L_c(y,t) - L_c(x,s)) \leq c_2\|y - x\|^2 < 0,$$

with $c_2 := (\|Q\| + \mu)(\frac{r}{q})^2 - \mu < 0$. *In both cases the algorithm stops after a finite number of iterations for every* b *and* d *in* $\mathbb{R}^n$.

Proof. As before we set $z = y - x$. We first note that for $i \in K$ we have $x_i \leq b_i$, hence $0 < y_i - b_i \leq y_i - x_i$, and therefore

$$\sum_{i \in K}(y_i - b_i)^2 \leq \|z_K\|^2.$$

Similarly, $y_T = b_T$, and hence

$$\sum_{i \in T}(x_i - b_i)^2 = \|z_T\|^2.$$

Using Proposition 3.3 we get

$$2(L_c(y,t) - L_c(x,s)) \leq \|Q\|\|z_T\|^2 - \mu\|z_{\bar{T}}\|^2 + c\|z_K\|^2 - c\|z_T\|^2.$$

The next goal is to bound $\|z_K\|$ in terms of $\|z\|$. On K_1 we have $s_{K_1} \leq 0, t_{K_1} = 0$, and therefore $(Qz)_{K_1} = s_{K_1} \leq 0$. Similarly $s_{K_2} = t_{k_2} = 0$ on K_2, and thus $(Qz)_{K_2} = 0$. It follows that $(Qz)_K = Q_K z_K + Q_{K,\bar{K}} z_{\bar{K}} = \begin{pmatrix} s_{K_1} \\ 0 \end{pmatrix}$. Taking the inner product with z_K and noting that $z_{K_1} \geq 0$ we find

$$z_K^T(Qz)_K = z_K^T Q_K z_K + z_K^T Q_{K,\bar{K}} z_{\bar{K}} = s_{K_1}^T z_{K_1} \leq 0. \tag{15}$$

We can now bound $\|z_K\|$ in the following two ways.
Variant 1 to bound $\|z_K\|$.
We have

$$\mu\|z_K\|^2 \leq \|z_K^T Q_K z_K\| \leq \|z_K^T Q_{K,\bar{K}} z_{\bar{K}}\| \leq \nu\|z_K\|\|z_{\bar{K}}\|.$$

The first inequality follows from the definition of μ, the second uses (15), and the last follows from the definition of ν. So we get:

$$\|z_K\|^2 \leq \frac{\nu^2}{\mu^2}\|z_{\bar{K}}\|^2 \leq \frac{\nu^2}{\mu^2}(\|z_T\|^2 + \|z_{\bar{T}}\|^2).$$

Variant 2 to bound $\|z_K\|$.
Using again (15) and the definition of D we can alternatively write

$$0 \geq z_K^T(Qz)_K = z_K^T D_K z_K + z_K^T(Q_{K,N} - D_{K,N})z.$$

From this we get $q\|z_K\|^2 \leq z_K^T D_K z_K \leq r\|z_K\|\|z\|$, and therefore

$$\|z_K\|^2 \leq \frac{r^2}{q^2}\|z\|^2 = \frac{r^2}{q^2}(\|z_T\|^2 + \|z_{\bar{T}}\|^2).$$

Variant 1 yields

$$2(L_c(y,t) - L_c(x,s)) \leq (\|Q\| - c + c\frac{\nu^2}{\mu^2})\|z_T\|^2 + (c\frac{\nu^2}{\mu^2} - \mu)\|z_{\bar{T}}\|^2, \tag{16}$$

while variant 2 gives

$$2(L_c(y,t) - L_c(x,s)) \leq (\|Q\| - c + c\frac{r^2}{q^2})\|z_T\|^2 + (c\frac{r^2}{q^2} - \mu)\|z_{\bar{T}}\|^2. \tag{17}$$

Note that setting $c := \|Q\| + \mu$ makes the coefficients of $\|z_T\|^2$ and $\|z_{\bar{T}}\|^2$ in (16) as well as (17) equal to one another.

Up to this point we have not yet used either of the conditions (C1) or (C2). Suppose now that (C1) holds. Then the coefficients in (16) are both equal to $(\|Q\| + \mu)\frac{\nu^2}{\mu^2} - \mu = c_1$. Moreover condition (C1) implies that $\|Q\| + \mu < \mu(\frac{\mu}{\nu})^2$, and hence $c_1 < 0$.

Similarly we find that in (17) the coefficients are both equal to c_2. If we assume that (C2) holds, then $c_2 < 0$.

It is clear that under the above conditions it is impossible that some active set A is reproduced twice. Since there is only a finite number, namely 2^n, of different sets, the algorithm stops with an optimal solution for every choice of b and d. ■

Remark 3.5 *We conclude this section with several remarks.*

1. *The main computational effort per iteration is the solution of the linear equation (8). The size of this system varies, but typically it is much smaller than n, since it needs only be solved for the inactive variables.*

2. *We emphasize that the iterates of the algorithm do not observe primal and dual feasibility. Thereby big changes to the active set are possible and occur in numerical practice. This is an extremely useful feature especially for large scale problems.*

3. *Let us comment on scale invariance. Utilizing pre- and post-multiplication by a diagonal matrix, it is always possible to scale Q such that all its diagonal elements are equal to 1. However, this type of scaling does not improve the conditioning of the algorithm, since the iterates remain unaffected by diagonal scaling. The easy details are left to the reader.*

4. *It is straightforward to extend the algorithm to both lower an upper bounds, i.e. to constraints of the form*
$$l \leq x \leq u.$$
We experimented also with this type of problem, and did not find a significant difference in performance to the unilaterally constrained problem.

5. *It is also straightforward to extend the present approach to strictly convex cost functions f. In this case the algorithm can be included in an outer SQP type iteration.*

3.3 Equality Constraints

We close this section with a discussion on why our approach may have difficulties if we allow equality constraints. For this purpose we consider the problem:

$$\text{minimize } J(x) \text{ such that } Rx - r = 0,\ x \leq b.$$

Denoting by u the multipliers for the equality constraints, the KKT system for this problem is given by

$$Qx + d + s + R^T u = 0,\ Rx = r,\ s \circ (x - b) = 0,\ x \leq b,\ s \geq 0.$$

Solving the system $Qx + d + s + R^T u = 0,\ Rx = r$ under the additional constraint that $x_A = b_A,\ s_I = 0$ leads to

$$\begin{pmatrix} Q_I & R_I^T \\ R_I & 0 \end{pmatrix} \begin{pmatrix} x_I \\ u \end{pmatrix} = \begin{pmatrix} -d_I - Q_{I,A} x_A \\ r - R_A x_A \end{pmatrix}.$$

This system need not be solvable even if R has full rank. R_I could for instance be the zero matrix for some $I \subseteq N$. On the other hand, this system is solvable for any I if there is only a single equation $\sum r_i x_i = r_0$ and $r_i \neq 0$ for all i. This occurs for instance if x represents a convex combination, $x \geq 0,\ \sum_i x_i = 1$. Numerical experiments for this case are given in Section 4.5.

4 Computational Experience

In section 3 we gave sufficient conditions for convergence of our algorithm. In this section we look at the practical behaviour of the algorithm by considering a variety of test problems.

4.1 A randomly generated problem

To get a first impression, we study in some detail a randomly generated problem, where we vary Q by making it increasingly ill-conditioned. In order for the reader to be able to reproduce some of the following results, we provide the MATLAB commands that we used to generate the data Q, d and b.

```
>> n=500;
>> rand('seed',n);
>> p=sprandn(n,n,.1)+speye(n);
>> p=tril(triu(p,-100));
>> Q0=p*p';
>> d=rand(n,1)*20*n-10*n;
>> b=ones(n,1);
>> Astart=find(rand(n,1)>rand);
```

5	6	7	8	9	10	11	12	cond(Q)	ε
540	460	0	0	0	0	0	0	795	1
0	67	575	343	15	0	0	0	9.3e+03	1e-1
0	0	11	306	470	176	35	2	8.7e+06	1e-4
0	0	14	266	443	220	47	10	7.4e+09	1e-7
0	0	13	244	437	235	59	12	5.1e+12	1e-10

Table 3: Iteration counts for a random problems of size $n = 500$. $Q = Q0 + \varepsilon I$, where $Q0 \succeq 0$ is singular.

The problem of size $n = 500$ was generated so that $Q0$ is a sparse positive semidefinite matrix, which is singular. The matrices Q are obtained from $Q0$ by adding a small multiple of the identity matrix I:

$$Q = Q0 + \varepsilon I,$$

where $\varepsilon \in \{1, 10^{-1}, 10^{-4}, 10^{-7}, 10^{-10}\}$. The estimated condition numbers of these matrices are given in Table 3 below. We used the MATLAB command `condest` to compute them. The only nontrivial input to our algorithm is the initial guess A for the optimal active set. The algorithm is started by a randomly generated initial active set Astart, see the last command line above.

In Table 3 we summarize the iteration counts of our algorithm. Each line represents 1000 runs on the same problem with different starting sets A. The columns labeled with numbers *it* ranging from 5 to 12 indicate how often the algorithm stopped after *it* iterations. We note that the algorithm never took more than 12 iterations in all of the 5000 runs. For the well-conditioned problem with $\varepsilon = 1$, the algorithm in fact always stopped after no more than 6 iterations.

4.2 Biharmonic Equation

A rich source of applications for our algorithm comes for applications in mathematical physics. Here we consider a model problem describing small vertical deformations u of a horizontal, elastic thin plate occupying a domain $\Omega \subset \mathbb{R}^2$, which is clamped along its boundary Γ, under the influence of a vertical force of density $f \in L^2(\Omega)$, with the plate constrained to remain below an obstacle $\psi \in L^\infty(\Omega)$:

$$\left.\begin{array}{l} \min \frac{1}{2} \int_\Omega |\Delta u(x)|^2 dx - \int_\Omega f(x) u(x) dx \\ \text{over } \ u \in H^2(\Omega) \ \text{ satisfying} \\ u = 0, \ \frac{\partial u}{\partial n} = 0 \ \text{ on } \Gamma \\ u(x) \le \psi(x) \text{ a.e. in } \Omega. \end{array}\right\} \tag{18}$$

We assume throughout that ψ is uniformly positive in a neighborhood of Γ. This implies, in particular that the set of feasible solutions to (18) is nonempty. It is then well known that (18) admits a unique solution u^* in $H_0^2(\Omega) = \{v \in H^2(\Omega) : v|\Gamma = 0, \ \frac{\partial v}{\partial n}|\Gamma = 0\}$, see e.g. [7], Section 4.4. The Lagrange multiplier λ associated to the inequality constraint is only a measure in $L^\infty(\Omega)^*$, the dual of $L^\infty(\Omega)$. The KKT–system characterizing the solution to (18) is given by

$$\left.\begin{array}{l}(\Delta u^*, \Delta v)_{L^2} + \langle \lambda, v\rangle_{L^{\infty,*},L^\infty} = (f,v)_{L^2} \text{ for all } v \in H_0^2(\Omega),\\ \langle \lambda, v\rangle_{L^{\infty,*},L^\infty} \geq 0 \text{ for all } v \in H_0^2(\Omega), \text{ with } v \geq 0\\ \langle \lambda, u^* - \psi\rangle_{L^{\infty,*},L^\infty} = 0,\ u^* - \psi \leq 0 \text{ a. e. in } \Omega,\end{array}\right\} \qquad (19)$$

where $(\cdot,\cdot)$ denotes the inner product in $L^2(\Omega)$, $\langle\cdot,\cdot\rangle_{L^{\infty,*},L^\infty}$ the duality pairing between $L^\infty(\Omega)^*$ and $L^\infty(\Omega)$, and we recall that $H_0^2(\Omega) \subset L^\infty(\Omega)$.
Formally (19) can be expressed as

$$\left.\begin{array}{l}\Delta^2 u^* + \lambda = f, \text{ in } \Omega, \text{ with } u|\Gamma = \frac{\partial u}{\partial n}|\Gamma = 0,\\ \lambda \geq 0, u^* \leq \psi, \lambda(x)(u^* - \psi)(x) = 0 \text{ in } \Omega.\end{array}\right\} \qquad (20)$$

To realize (18) discretization of the biharmonic Δ^2 with homogenous Dirichlet and Neumann boundary conditions, as well as of u, λ, f and ψ is required. For our numerical tests we chose Ω as the unit square which was discretized by a uniform axiparallel grid of meshsize $h = \frac{1}{m+1}$, for fixed $m \in \mathbb{N}$, resulting in m^2 interior nodes. The biharmonic operator, including the boundary conditions was discretized on the basis of a 13–point finite difference stencil as described e.g. in [8] page 105, resulting after proper ordering of the nodes in a positive definite $m^2 \times m^2$–matrix Q_h. It is worthwhile to point out that Q_h is not an M-matrix. The functions u, f and ψ were approximated by their interior nodal values and are denoted by u_h, f_h, ψ_h. Approximating the integral by cuboids centered at the nodes of the grid, the discretization of (18) turns out to be

$$\left.\begin{array}{l}\min \frac{1}{2} u_h^T Q_h u_h + f_h^T u_h\\ \text{subject to } u_h \leq \psi_h,\end{array}\right\} \qquad (21)$$

which is of the form (P). For the discretized problem a Lagrange multiplier $\lambda_h \geq 0$ in $\mathbb{R}^{m^2}$ clearly exists. We solved (21) for different choices of f and ψ. Numerical results are presented for a typical case with

$$f(x,y) = -60(1-x^2)\ y\ e^{-7(x-.9)^2-4(y-.1)^2} + 100x(1-y)\ e^{-3(x-.2)^2-6(y-.8)^2},$$

see Table 5, and

$$\psi(x,y) = 4 \cdot 10^{-5}.$$

A straightforward application of our algorithm with discretization $m = 128$ requires 71 iterations and 172 seconds to reach the exact solution of (21). All computation times given are seconds on a DEC ALPHA workstation.

To reduce the number of iterations and to utilize the advantage of fast solution on coarse meshes a multilevel approach turned out to be very efficient. For this purpose we choose a sequence of grids characterized by mesh-sizes $h_i = 2^{-i}h$, $i = 0, 1, \cdots$, where h is an initial coarse meshsize. The nested algorithm then consists in a coarse to fine sweep. The optimal active set for meshsize h_i is interpolated to the grid h_{i+1}, and utilized as initial active set for the problem with gridsize h_{i+1}.

m	8	16	32	64	128	256
iter	6	7	10	6	7	8
time (sec.)	0.01	0.08	0.6	2.7	26.7	364
largest system	26	139	647	2738	11251	45580
dimension (n)	64	256	1024	4096	16384	65536

Table 4: Number of iterations during grid refinement.

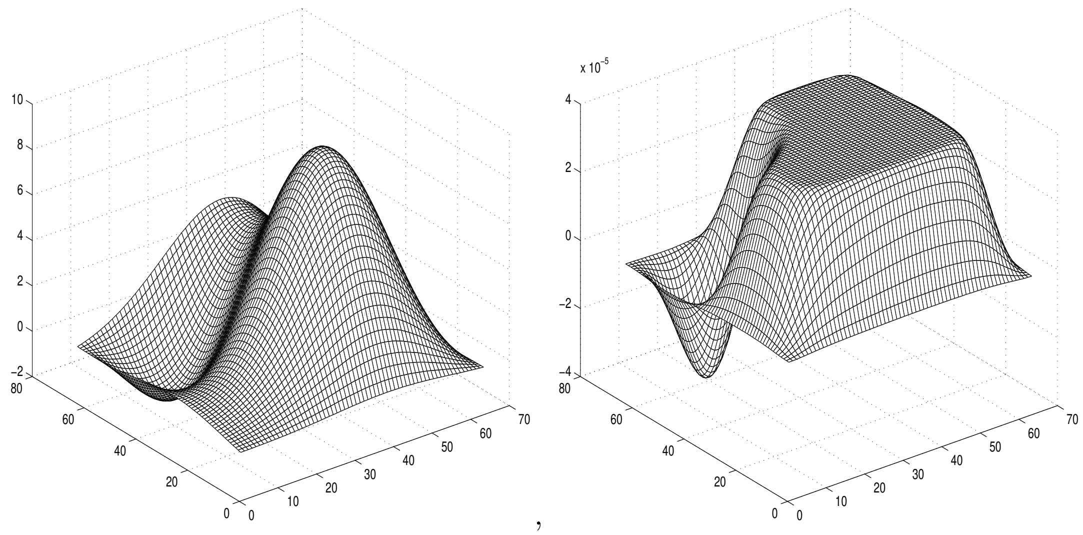

Table 5: Force f acting on plate (left), deformation u of plate (right)

Utilizing this strategy for our test problem resulted in a significant reduction in the total number of iterations. In Table 4 we give the number of required iterations on each grid level, starting with $m = 8$. In our computational experiments we noted that the algorithm typically maintained dual feasibility throughout ($s \geq 0$), and only primal feasibility was violated. This phenomenon is known to hold if Q is an M-matrix, see [10].

4.3 Lack of strict complementarity

A solution pair (x, s) of KKT is called strictly complementary, if $s_i > 0$ if and only if $x_i = b_i$. In other words, (x, s) lacks strict complementarity if for some i we have $s_i = x_i - b_i = 0$. The lack of strict complementarity may be a potential source of computational difficulties. Our convergence analysis of section 3 holds independently of any assumptions about strict complementarity.

In finite precision computation, the test $s_i > 0$ is sensitive to round-off errors. Therefore we replace it by $s_i > -tol$, where $tol > 0$ is a small tolerance. Similarly, we allow small violations of primal feasibility and set our stopping condition to

$$s_i \geq -tol, \quad x_i \leq b_i + tol \quad \forall i.$$

We used $tol = 10^{-6}$ in our computational experiments. With this modification, we tested our method on problems lacking strict complementarity, which we generated as follows.

First we partition N into three nonempty sets A, B, I. We select b and $Q \succ 0$ at random. Setting $x_A = b_A$, $x_B = b_B$, $x_I < b_I$ and $s_B = 0$, $s_I = 0$, $s_A > 0$ and defining $d := -Qx - s$ yields a solution pair (x, s) satisfying KKT, with $s_B = x_B - b_B = 0$. By construction the solution thus lacks strict complementarity on B.

Our computational tests with this type of problem did not indicate a performance difference to problems where strict complementarity holds.

4.4 Regularization

The conditions (13) or (14) are sufficient for our method to converge. In practice we noticed however, that the method also converges when these conditions are not satisfied, provided that Q is positive definite and not too ill-conditioned.

To make the method work independent of the conditioning of Q, we propose to use the following regularization term in the first few iterations of the algorithm. In iteration k of the algorithm we use

$$Q' := Q + \frac{t}{2^{k-1}} I \text{ for } k \leq k_0,$$

and set $Q' = Q$ after iteration k_0. Here the number t depends on the scaling of Q. In our tests we set $t = 1$ and $k_0 = 4$. With this modification we never ran into computational difficulties, and managed to handle even problems with singular Q. A specific class of examples is given in the next subsection.

4.5 Projection onto Polytope

Suppose we are given $n + 1$ points $a_0, \ldots, a_n$ in $\mathbb{R}^m$. If P denotes the convex hull of $a_1, \ldots, a_n$, we consider the problem of finding the point $a \in P$ closest to a_0. Let $A = (a_1, \ldots, a_n)$. Then $a \in P$ if and only if $a = Ax$, $x \geq 0$, $\sum_i x_i = 1$. Thus we arrive at the following problem:

$$\text{minimize } \|Ax - a_0\|^2 \text{ such that } x \geq 0, \ \sum_i x_i = 1.$$

We generate our problems as follows. A is chosen to be a sparse random matrix of order $m \times n$, where $n > m$, with nonzero entries uniformly distributed in $(0, 1)$. We set $a_0 = 0$, and measure the distance of the polytope $P \subseteq \mathbb{R}^n_+$, contained in the positive orthant, from the origin. Under our assumptions, $Q = A^T A$ is positive semidefinite with dimension of the nullspace at least $n - m$, so the matrix is highly singular. To insure that the iterates are well-defined, we have added a multiple of $(\frac{1}{2})^{k-1}$ times the identity for iterations $k = 1, \ldots, 4$. After iteration 4 we continued with the singular matrix Q, without encountering any computational difficulties. In Table 6 we present again accumulated results for 100 testruns with $n = 500$ and $n = 1000$. In both cases $m = n/2$. The meaning of the numbers in Table 6 is the same as that in Table 3. The number of iterations was never more than 10, despite the fact, that we work with a singular matrix after iteration 4. Obviously, after the first 4

n	m	6	7	8	9	10	11
500	250	1	23	61	12	3	0
1000	500	0	6	56	33	5	0

Table 6: Distance of polytope from origin (iteration counts).

iteration	$n - \|A\|$	common	difference
1	11097	10928	169
2	11192	11092	100
3	11225	11191	34
4	11245	11225	20
5	11249	11245	4
6	11251	11249	2
7	11251	11251	0

Table 7: Cardinality of inactive set during iterations for $m = 128$. The third column provides the number of common elements to the subsequent inactive set.

iterations the approximation to the true active set was already close enough to the optimal solution, so that no computational difficulties occurred.

4.6 Cholesky Up/Down-date

The algorithm which we use for the computational tests can be improved by utilizing the fact that as the method progresses, the current estimate of the active set gets closer to the active set at the optimum. Hence from a certain iteration on this set changes only in a few variables from one iteration to the next when compared to the total number of variables. To give some intuition, we provide in Table 7 the cardinalities of the inactive set (= size of linear system to be solved) and the number of common elements to the subsequent inactive set for the biharmonic equation with $m = 128$, starting from the extrapolated optimal active set from $m = 64$, see Table 5.

In the current version of the algorithm, see the Appendix, we do not use this feature and compute the Cholesky factor of the system from scratch in each iteration. It would certainly be worth testing to use the Cholesky factor from the previous iteration, and to perform the required update and downdate steps to add or remove elements as necessary. Since version 5.3 of MATLAB does not support this feature for sparse matrices, we have no computational results for this variant of the algorithm yet.

5 Discussion

In this paper we presented an infeasible active set method. Under certain conditions we were able to prove global convergence of the method. Moreover we demonstrated the efficiency of the method by considering a diverse set of test problems. Among the special features of the algorithm are its ability to find the exact numerical solution of the problem, and the fact that at each iteration level the size of the linear system which must be solved is determined by the currently inactive set, which can be significantly smaller than the total set of variables. As a consequence the proposed algorithm differs significantly from interior point methods, for example. From the numerical experiments we observe that the algorithm has the feature of "correcting" many active variables to inactive ones and vice versa during the early stages of the iterations. This is certainly one of its distinguishing practical features. The total number of iterations is frequently quite insensitive with respect to initialization. Lack of strict complementarity as well as singularity of the system matrix do not inhibit the efficiency of the algorithm. In the context of the obstacle problem for the plate equation we demonstrated that it can be useful to combine our algorithm with a multi-level approach.

References

[1] M. BERGOUNIOUX, M. HADDOU, M. HINTERMÜLLER, and K. KUNISCH. A comparison of interior point methods and a Moreau-Yosida based active set strategy for constrained optimal control problems. *SIAM Journal on Optimization*, to appear, 2000.

[2] M. BERGOUNIOUX, K. ITO, and K. KUNISCH. Primal-Dual Strategy for Constrained Optimal Control Problems. *SIAM Journal on Control and Optimization*, 37:1176–1194, 1999.

[3] D.P. BERTSEKAS. *Constrained optimization and Lagrange multipliers*. Academic Press, 1982.

[4] T.F. COLEMAN and Y. LIN. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6:418–445, 1996.

[5] T.F. COLEMAN and J. LIU. An interior Newton method for quadratic programming. *Mathematical Programming*, 85:491–523, 1999.

[6] M. D'APUZZO, M. MARINO, P.M. PARDALOS, and G. TORALDO. A parallel implementation of a potential reduction algorithm for box-constrained quadratic programming. Technical report, 2000.

[7] R. GLOWINSKI, J.L. LIONS, and R. TRÉMOLIÈRES. *Numerical analysis of variational inequalities*. North Holland, Amsterdam, 1981.

[8] W. HACKBUSCH. *Elliptic partial differential equations*. Springer Berlin, 1992.

[9] H. HEINKENSCHLOSS, M. ULRICH, and S. ULRICH. Superlinear and quadratic convergence of affine-scaling interior-point Newton methods for problems with simple bounds without strict complementarity assumption. *Mathematical Programming*, 86:615–635, 1999.

[10] T. KÄRKÄINEN, K. KUNISCH, and P. TARVAINEN. Primal-dual active set methods for obstacle problems. Technical report, 2000.

[11] M. KOČVARA and J. ZOWE. An iterative two-step algorithm for linear complementarity problems. *Numerische Mathematik*, 68:95–106, 1994.

[12] C.-J. LIN and J.J. MORÉ. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9:1100–1127, 1999.

[13] J.J. MORÉ and G. TORALDO. Algorithms for bound constrained quadratic problems. *Numerische Mathematik*, 55:377–400, 1989.

[14] J.J. MORÉ and G. TORALDO. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1:93–113, 1991.

[15] S.J. WRIGHT. *Primal-dual interior point methods*. SIAM, Philadelphia, 1997.

6 Appendix: A Matlab function

We include a short MATLAB function that solves problem (P). The routine takes as input the problem data Q, d and b and an initial guess A for the active set.

It outputs the optimal solution x, together with the dual variables s, the number of iterations and the optimal active set.

```
function [ x, s, iter, Aopt] = qp_bnd( Q, d, b, A);

% solves: min d'x + (1/2) x'Qx  subject to: x  <= b
% Q is assumed to be symmetric positive definite
% KKT:    Qx + s + d = 0;  x <= b, s >=0, s'(x-b) = 0
% input:  (Q,d,b) problem data,
%         A ... guess on initial active set, e.g. A=(1:n);
% output: (x,s) optimal solution, iter= # iterations
%         Aopt = active set at opt. sol.
% call:   [ x, s, iter, Aopt] = qp_bnd( Q, d, b, A);

n = length( d);                  % problem size
k = 0;                           % iteration count
done = 0;                        % not yet done
disp(' iter inact '  );          % display some info on screen

% main loop
while done < 1;                  % while not optimal
     k = k + 1;                  % start a new iteration
% solve system KKT(A):
     I = ones(n,1);              % compute I, the complement of A
     I(A) = zeros( length(A),1);
     I = find(I>0);              % complement of A
     x = b;                      % x( A) = b( A);
     s = zeros( n,1);            % s( I) = 0
     if length( I) > 0;
         ltri = sparse(chol(full(Q( I,I))));  % cholesky factor
         rhs = -d( I);
         if length(A) > 0;       % update right hand side rhs
           rhs = rhs - Q( I,A)*x( A);
         end;
         xtmp = (ltri') \ rhs;
         x( I) = ltri \  xtmp;   % solve for inactive variables
     end;
     if length( A)>0;            % backsubsitute for s(A), if |A|>0
         s( A) = -d( A) - Q( A, :)* x;
     end;
     A = [find(x>b); find(s>0)]; % active constraints of new solution
     done = (max(x-b) <= 0) & (min(s) >= 0);
     fprintf(' %3.0d %6.0f \n',[k  (n-length(A))]);
     if k > 100; done = 1;       % emergency exit to avoid cycling
         disp(' max number of iterations reached.');
     end;
end;                 % end while
iter = k;
Aopt = A;
```