On the minimization of quadratic functions subject to box constraints[1]

Ron S. Dembo and Ulrich Tulowitzki

School of Organization and Management

and

Research Center for Scientific Computation,

Department of Computer Science

Yale University

December 1983, Revised September 1984

Yale Research Center for Scientific Computation
TR #302

[1]The work presented in this paper was supported in part by the U.S. Department of Transportation under grant no. CT-06-0011 and by the National Science Foundation under grant no. ECS-81-19513.

Abstract

We study efficient algorithms for very large quadratic problems subject to box constraints and propose some new convergent methods that permit the addition and deletion of many constraints each time a search direction is calculated. Numerical experiments, on problems of up to 10,000 variables, indicate that the performance of the proposed algorithms appears to be insensitive to the number of binding constraints at the optimal solution or to the starting point. Also it appears as if solving the box constrained problem is at worst marginally more expensive than solving the same problem without constraints.

1. Introduction

Box constrained quadratic programming problems have the form

$$(BQP) \qquad \begin{array}{ll} \textit{minimize} & Q(x) = \frac{1}{2}\, x^T H x + c^T x \\ \textit{subject to} & l \le x \le u \end{array} \tag{1}$$

where H is an $(n \times n)$-matrix and c, l and u are given vectors in R^n. We assume throughout this paper that H is positive definite, that is, there exists a constant $\mu > 0$ such that $x^T H x \ge \mu \|x\|^2$ for all x in R^n.

Problems of this type arise very often in numerical analysis applications, optimal control and operations research, as well as subproblems in general nonlinear optimization algorithms. Among the applications in numerical analysis is the solution of partial differential equations arising in Dirichlet problems with obstacles [11], the bar under torsion problem [10] (determining the zone of elasticity and plasticity of a cylindrical bar), or free boundary value problems such as the journal bearing problem [15] or the earthen dam problem [1, 11]. Many discrete time or discretized continuous time optimal control problems lead to quadratic programs with box constraints [39]. Another frequent source of problems of this type is in linear least squares when bounds on the variables are imposed by the nature of the system being modeled. One example of this arises in computerized tomography, where linear least squares problems have to be solved subject to nonnegativity constraints (see [45] for example]). Quadratic programming models also arise frequently in operations research applications and are an important component of portfolio problems in finance (see [18]).

Our primary interest is however in the efficient solution of subproblems that arise when solving general nonlinear optimization problems via successive (recursive) quadratic programming methods [7, 8, 27, 41, 42, 43]. Box-constrained quadratic programming problems arise naturally if the original problem is box-constrained but also if (exact) penalty methods are used where only equality constraints are penalized. Even when solving general problems however, box-constrained quadratic programming problems arise since the dual of a general, linearly-constrained and strictly-convex QP, is box-constrained (see Nickel and Tolle [35] for example). We therefore feel that the box-constrained problem plays a fundamental role in constrained optimization and a thorough analysis of such problems is a necessary prerequisite in the study of algorithms for large-scale optimization.

The importance of such problems has lead many researchers to develop quadratic programming methods which can be readily applied to solve **BQP**. However, often these problems are large ($n = 10{,}000$ is not atypical in numerical analysis or optimal control applications). They also appear commonly as subproblems and therefore need to be solved (perhaps inexactly [43]) many times. Hence efficiency is essential and, in light of the problem dimensions, storage may be at a premium. First it is important to be able to identify a set of constraints that will be binding at an optimal solution (particularly if the QP is not going to be solved exactly [43]). With the exception of some projection methods [4, 32], most existing algorithms use active set strategies which allow one either to add or to drop only one constraint per search direction computation (see [25, 26] for example). This could make them prohibitively slow for large problems for which the starting point and the optimal solution may have vastly different active sets. Algorithms that use projection steps can usually drop many constraints with little overhead. However, they drop constraints only after having solved some reduced problem to optimality and therefore they may be doing a significant amount of work the "wrong" subspace. On the other hand the gradient projection method of Goldstein [29] and Levitin and Polyak [31], which allows one to add and drop many constraints at each iteration, may zigzag between constraint sets, particularly on degenerate problems where the active set might never settle down. Furthermore, it reduces to the unconstrained gradient or steepest descent method once a set of binding constraints at the optimal solution is identified and therefore exhibits a (usually slow) linear rate of convergence. To enhance the asymptotic behavior of this method, Bertsekas [5] proposed a Newton-like projection method for general nonlinear programming problems with box constraints which has a superlinear rate of convergence if the active set settles down. Without assuming strict complementarity Bonnans [9] shows superlinear convergence for the same class of algorithms if the $\epsilon > 0$ which defines the ϵ-active constraints tends to zero fast enough. However, these algorithms require the solution of an equality-constrained quadratic programming problem at each iteration.

In this paper we shall therefore aim to devise algorithms whose efficiency is comparable to that of gradient projection methods when identifying an optimal active constraint set and that of

Newton-like methods once this set of constraints is identified. These algorithms use projected gradient directions when the set of active constraints changes and conjugate gradient directions otherwise. Conjugate reduced gradients are used as restricted directions, which implies efficient finite convergence on a subspace. A bending type one dimensional search permits these algorithms to pick-up many constraints per search direction computation. Finally, a forcing sequence strategy [19] is used to permit inexact subspace minimization.

The organization of the remainder of this paper is as follows. In Section 2 we introduce some notation and definitions. In Section 3 we describe a basic algorithmic framework and show how existing algorithms fit into this framework. A second framework for inexact projection methods is described in Section 4. Extensive computational experience on a variety of large-scale test problems is summarized in Section 5.

2. Notation and Definitions

In order to state the algorithmic framework we need to introduce some terminology. The vector $\boldsymbol{g(x)} = \boldsymbol{Hx} + \boldsymbol{c}$ denotes the gradient of $\boldsymbol{Q(x)}$ at the point $\boldsymbol{x}$. For any feasible point $\boldsymbol{x}$ it is convenient to define the set of **active** constraints as

$$\mathcal{A}(\boldsymbol{x}) = \{\, i \mid x_i = l_i \ \text{ or } \ x_i = u_i\}$$

and the set of **binding** constraints as

$$\mathcal{B}(\boldsymbol{x}) = \{\, i \mid x_i = l_i \ \text{ and } \ g_i(\boldsymbol{x}) \geq 0 \ \text{ or } \ x_i = u_i \text{ and } \ g_i(\boldsymbol{x}) \leq 0 \,\}.$$

The binding constraints are active constraints whose associated Lagrange multiplier estimates have the correct (optimal) sign. The **reduced** gradient $\boldsymbol{g}^R(\boldsymbol{x})$ at a feasible point $\boldsymbol{x}$ is defined as

$$(\boldsymbol{g}^R(\boldsymbol{x}))_i = \begin{cases} 0 & \text{if } i \in \mathcal{A}(\boldsymbol{x}) \\ g_i(\boldsymbol{x}) & \text{if } i \notin \mathcal{A}(\boldsymbol{x}). \end{cases}$$

and the **projected** gradient $\boldsymbol{g}^P(\boldsymbol{x})$ at the feasible point $\boldsymbol{x}$ is defined as

$$(\boldsymbol{g}^P(\boldsymbol{x}))_i = \begin{cases} 0 & \text{if } i \in \mathcal{B}(\boldsymbol{x}) \\ g_i(\boldsymbol{x}) & \text{if } i \notin \mathcal{B}(\boldsymbol{x}). \end{cases}$$

It is easy to see that $\boldsymbol{x}$ is a **stationary** point (a point satisfying the first order necessary conditions for optimality) if and only if $\boldsymbol{x}$ is feasible and $\boldsymbol{g}^P(\boldsymbol{x}) = 0$, that is $g_i(\boldsymbol{x}) = 0$ for $i \notin \mathcal{B}(\boldsymbol{x})$, the non binding constraints. The problem BQP is called ***degenerate*** if at a stationary point $\boldsymbol{x}^*$, $g_i(\boldsymbol{x}^*) = 0$ for some $i \in \mathcal{B}(\boldsymbol{x})$. Finally we denote by $[\boldsymbol{y}]^\#$ the projection of the vector $\boldsymbol{y}$ in $\boldsymbol{R}^n$ onto the feasible box $\{\boldsymbol{x} \in \boldsymbol{R}^n \mid \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}\}$, that is $([\boldsymbol{y}]^\#)_i = \min\{ u_i , \max(l_i , y_i)\}$. Often we omit the argument $\boldsymbol{x}_k$ and write $\boldsymbol{g}_k$ instead of $\boldsymbol{g}(\boldsymbol{x}_k)$, $\mathcal{A}_k$ instead of $\mathcal{A}(\boldsymbol{x}_k)$ etc.

We describe our algorithm using two types of feasible directions; **relaxing** directions, defined as feasible descent directions along which one or more constraints may be dropped from the current active set and **restricted** directions, which are defined as feasible descent directions restricted to lie in subspace containing the current active set. That is, $\boldsymbol{p}$ is a restricted direction at a feasible point $\boldsymbol{x}$ if $p_i = 0$ for all $i \in \mathcal{A}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{p} < 0$.

3. Conjugate gradient projection methods

A basic algorithmic framework can now be defined. Starting at a feasible solution either a gradient related relaxing or restricted direction is determined depending on whether or not a constraint relaxation condition is satisfied (see [19] for a more detailed discussion of this terminology). Then a step is computed such that the new solution is feasible and an Armijo condition is satisfied. The advantage of specifying an algorithm in this way is that it is easy to see how various parts of the algorithm may be replaced while still retaining the convergence characteristics. It also provides a basis for comparing algorithms, as we demonstrate below.

3.1. Algorithmic Framework #1

```
START with $\boldsymbol{x}_0$, a feasible solution
SET $\boldsymbol{p}_0 = \boldsymbol{0}$, $\beta_0 = 0$, $k = 0$, $\sigma \in (0,1)$ and $\gamma \in (0,\frac{1}{2})$
WHILE "not optimal" DO
    1. (Determine a feasible descent direction $\boldsymbol{p}_k$)
        IF $\|\boldsymbol{g}_k^R\| \le \eta_k$                    (constraint relaxation condition)    (2)
            THEN $\boldsymbol{p}_k = -\boldsymbol{g}_k^P + \beta_k \boldsymbol{p}_{k-1}$          (relaxing direction)    (3)
                where $\beta_k = 0$ if $\mathcal{B}_k \ne \mathcal{A}_{k-1}$ and $\beta_k = \|\boldsymbol{g}_k^R\|^2/\|\boldsymbol{g}_{k-1}^R\|^2$ otherwise
            ELSE $\boldsymbol{p}_k = -\boldsymbol{g}_k^R + \beta_k \boldsymbol{p}_{k-1}$          (restricted direction)    (4)
```

where $\beta_k = 0$ if $\mathcal{A}_k \neq \mathcal{A}_{k-1}$ and $\beta_k = \|g_k^R\|^2/\|g_{k-1}^R\|^2$ otherwise

2. (Determine an **acceptable** step-size α_k)

$$\alpha_k = \sigma^{m_k} \alpha_k^* \quad \text{, where } \alpha_k^* = -\frac{g_k^T p_k}{p_k^T H p_k} \tag{5}$$

and m_k is the smallest nonnegative integer such that

$$Q([x_k + \alpha_k p_k]^{\#}) - Q(x_k) \leq \gamma \alpha_k g_k^T p_k \tag{6}$$

3. (Update)

$$SET \; x_{k+1} = [x_k + \alpha_k p_k]^{\#} \quad \text{and} \quad k \leftarrow k + 1. \tag{7}$$

We refer to methods conforming to this framework as ***conjugate gradient projection methods***, because all iterates will be feasible and the methods simplify to the conjugate gradient method if the problem is unconstrained or if the active set settles down and to projection methods if the active set constantly changes. Note that the forcing sequence η_k in (2) has not been specified. Different choices of sequences will lead us to different algorithms some of which, to our knowledge, have not appeared elsewhere.

The first and simplest algorithm is when $\eta_k = \infty$ for all k. In this case only relaxing directions in (3) will be used. If the set of binding constraints $\mathcal{B}$ changes then this direction is equal to minus the projected gradient and equal to the conjugate projected gradient direction otherwise. We refer to this method as **conjugate gradient projection method (CGP)**.

If only projected gradient directions are used, that is $\eta_k = \infty$ and $\beta_k = 0$ for all k, then we obtain the **Goldstein-Levitin-Polyak** gradient projection method. Bertsekas has analyzed this method in [4] using a modified Armijo step size rule in which $g_k^T p_k$ in (6) is replaced by $g_k^T([x_k + \alpha_k p_k]^{\#} - x_k)$. Since this method allows the algorithm to drop constraints at each iteration it might zigzag on degenerate problems if the active set does not settle down. This motivates the need for restricting search directions until some progress has been made on the current subspace. One mechanism for doing so is to take $\eta_k \rightarrow 0$ as $k \rightarrow \infty$ in (2) as suggested by Dembo and Sahi [19]. We refer to this method as **conjugate reduced gradient projection method (CRGP)**. Here the active constraints are forced to remain active during

Table 3-1: Algorithms conforming to Framework #1

Choice of: η_k	β_k	Algorithm
∞	0	modified Goldstein-Levitin-Polyak
∞	a.s.	CGP
$\eta_k \to 0$ ($=0$)	a.s.	CRG (Polyak)[2]
$\eta_k \to 0$	a.s.	CRGP

Key: a.s. = as specified in Framework #1.

the restricted iterations until the norm of the reduced gradient has been decreased sufficiently, that is **inexact subspace minimization** is used. A further specialization is when $\eta_k = 0$ for all k, that is **exact subspace minimization** is used. This could be viewed as a generalization of Polyak's algorithm [40] which uses a different linesearch, as discussed in Section 3.3.

3.2. Convergence of conjugate gradient projection methods

Before we prove the global convergence of any algorithm satisfying Framework #1 we show that the backtracking line search is well defined.

Lemma 3.1: *Let x be a feasible point of **BQP** and $g=Hx+c$. Assume the direction **p** satisfies $g^Tp<0$ and $p_i=0$ for $i \in \mathcal{B}=\mathcal{B}(x)$. Then*

$$Q'(x;p) := \lim_{\alpha\to 0^+} \frac{1}{\alpha}\left[Q([x+\alpha p]^{\#}) - Q(x)\right] \le g^Tp < 0$$

and hence there exists an $\bar{\alpha}>0$ such that for all $\alpha \in [0,\bar{\alpha}]$

$$Q([x+\alpha p]^{\#}) - Q(x) \le \gamma\alpha g^Tp < 0 \,.$$

Proof: The set of all indices for which $x_i = ([x+\alpha p]^{\#})_i$ for any $\alpha \ge 0$ is given by

$$\mathcal{A}_p = \{\, i \mid x_i = l_i \text{ and } p_i \le 0 \text{ or } x_i = u_i \text{ and } p_i \ge 0 \,\}.$$

[2]In this cases the step size $\alpha_k = \min\{\alpha_k^*, \alpha_k^{\max}\}$ where $\alpha_k^{\max} = \max\{\, \alpha \ge 0 \mid l \le x_k + \alpha\, p_k \le u \,\}$.

Then there exists an $\bar{\alpha}>0$ such that

$$([\boldsymbol{x} + \alpha \boldsymbol{p}]^{\#})_i = x_i + \alpha p_i \text{ for } 0 \leq \alpha \leq \bar{\alpha} \text{ and } i \notin \mathcal{A}_p.$$

Now let $\widetilde{\boldsymbol{p}}$ be given by

$$\widetilde{p}_i = \begin{cases} 0 & \text{if } i \in \mathcal{A}_p \\ p_i & \text{if } i \notin \mathcal{A}_p. \end{cases}$$

Then

$$Q'(\boldsymbol{x};\boldsymbol{p}) = \lim_{\alpha \to 0^+} \frac{1}{\alpha} [Q([\boldsymbol{x} + \alpha \boldsymbol{p}]^{\#}) - Q(\boldsymbol{x})]$$

$$= \lim_{\alpha \to 0^+} \frac{1}{\alpha} [Q(\boldsymbol{x} + \alpha \widetilde{\boldsymbol{p}}) - Q(\boldsymbol{x})] = \boldsymbol{g}^T \widetilde{\boldsymbol{p}}.$$

By the definition of $\mathcal{A}_p$ and the choice of $p_i = 0$ for $i \in \mathcal{B}$ it follows that $\mathcal{B} \subseteq \mathcal{A}_p$ and

$$g_i p_i = 0 \text{ for } i \in \mathcal{B} \quad \text{and} \quad g_i p_i \geq 0 \text{ for } i \in \mathcal{A}_p \backslash \mathcal{B}.$$

Therefore

$$0 > \boldsymbol{g}^T \boldsymbol{p} = \boldsymbol{g}^T \widetilde{\boldsymbol{p}} + \sum_{i \in \mathcal{A}_p \backslash \mathcal{B}} g_i p_i \geq \boldsymbol{g}^T \widetilde{\boldsymbol{p}} = Q'(\boldsymbol{x};\boldsymbol{p}).$$

q.e.d.

Proposition 3.1: *Let $\{\boldsymbol{x}_k\}$ be a sequence generated by any algorithm satisfying Framework #1. Then there either exists a $k \geq 0$ such that $\boldsymbol{x}_k = \boldsymbol{x}^*$ or $\lim_{k \to \infty} \boldsymbol{x}_k = \boldsymbol{x}^*$, where $\boldsymbol{x}^*$ is a stationary point of* BQP.

Proof: First we show that the algorithm is well defined. Assume $\boldsymbol{x}_k$ is not a stationary point, that is $\boldsymbol{g}^P(\boldsymbol{x}_k) \neq 0$. Observe that if the active set does not change $\boldsymbol{g}_k^T \boldsymbol{p}_{k-1} = 0$, since the step size α_k^* minimizes the quadratic in direction $\boldsymbol{p}_{k-1}$. Thus

$$\boldsymbol{g}_k^T \boldsymbol{p}_k = -\|\boldsymbol{g}^P(\boldsymbol{x}_k)\|^2 < 0$$

if $\boldsymbol{p}_k$ is a relaxing direction, or, using (2),

$$\boldsymbol{g}_k^T \boldsymbol{p}_k = -\|\boldsymbol{g}^R(\boldsymbol{x}_k)\|^2 < -\eta_k^2 \leq 0.$$

if $\boldsymbol{p}_k$ is a restricted direction. Hence, $\boldsymbol{g}_k^T \boldsymbol{p}_k < 0$. Since $\mathcal{B}_k \subseteq \mathcal{A}_k$ it follows that $(\boldsymbol{p}_k)_i = 0$ for all $i \in \mathcal{B}_k$. By Lemma 3.1 the backtracking line search and therefore $\boldsymbol{x}_{k+1}$ are well defined.

Now assume that the algorithm does not stop at a stationary point after a finite number of iterations and generates an infinite sequence $\{\boldsymbol{x}_k\}$. Note that since the objective function is quadratic the conjugate reduced gradient method will find an optimal solution in a finite number of steps if the active set remains unchanged. Therefore, for any forcing sequence $\{\eta_k \geq 0\}$, there exists an infininite subsequence of relaxing steps with $p_k = -g^P(x_k)$. Since H is positive definite, the set

$$C_0 = \{ x \mid l \leq x \leq u \text{ and } Q(x) \leq Q(x_0) \}$$

is compact. Hence there exists a subsequence $\{x_{k_j}\}$ such that

$$\lim_{j\to\infty} x_{k_j} = x^* \quad \text{and} \quad p_{k_j} = -g^P(x_{k_j}) \text{ for all j.}$$

We will show now that x^* is a stationary point of BQP.

By the definition of the step size, $\{Q(x_k)\}$ is monotonically decreasing and therefore $Q(x_k) > Q(x_{k+1}) > \ldots > Q(x^*)$ and

$$0 = \lim_{k\to\infty} \{Q([x_k + \alpha_k p_k]^\#) - Q(x_k)\} \leq \lim_{k\to\infty} \gamma\alpha_k g_k{}^T p_k \leq 0.$$

Furthermore we have for the subsequence $\{x_{k_j}\}$ that

$$0 = \lim_{j\to\infty} \alpha_{k_j} g_{k_j}{}^T p_{k_j} = - \lim_{j\to\infty} \sigma^{m_{k_j}} \alpha_{k_j}{}^* \|g^P(x_{k_j})\|^2 \leq - \lim_{j\to\infty} \sigma^{m_{k_j}} \|H\|^{-1} \|g^P(x_{k_j})\|^2 \leq 0.$$

Hence, if $\{m_{k_j}\}$ is bounded, it follows that

$$\|g^P(x^*)\|^2 = \lim_{j\to\infty} \|g^P(x_{k_j})\| = 0.$$

If $\lim_{j\to\infty} m_{k_j} = \infty$ then by definition of m_{k_j} in the backtracking line search (6) it follows that for j large enough

$$[Q([x_{k_j} - (\alpha_{k_j}/\sigma)\, g^P(x_{k_j})]^\#) - Q(x_{k_j})]/(\alpha_{k_j}/\sigma) > - \gamma\|g^P_{k_j}\|^2. \tag{8}$$

Taking limits in (8) and using Lemma 3.1 yields

$$- \|g^P(x^*)\|^2 \geq Q'(x^*;-g^P(x^*)) \geq - \gamma\|g^P(x^*)\|^2$$

and therefore

$$(1-\gamma)\, \|g^P(x^*)\|^2 \leq 0.$$

Since $(1-\gamma) > 0$ it follows that $\|g^P(x^*)\|^2 = 0$. Therefore x^* is a stationary point of BQP and

$$(g(x^*))^T(x - x^*) \geq 0 \quad \text{for all} \quad x \in \{ x \mid l \leq x \leq u \}.$$

Hence, since H_k is positive definite

$$Q(x_k) - Q(x^*) = (g(x^*))^T(x_k - x^*) + \tfrac{1}{2}(x_k - x^*)^T H(x_k - x^*) \geq \tfrac{1}{2}\mu \|x_k - x^*\|^2.$$

Now, since $\lim_{k\to\infty} Q(x_k) = Q(x^*)$ it follows that $\lim_{k\to\infty} x_k = x^*$. q.e.d.

For quadratic objective functions we can state that an algorithm conforming to our framework satisfies the finite termination property if the forcing sequence tends to zero or strict complementarity holds at the solution. This is a direct extension of the finiteness of conjugate gradient methods for unconstrained quadratic programming problems.

Proposition 3.2: *Assume that the forcing sequence in (2) satisfies $\eta_k \to 0$ as $k \to \infty$. Then any algorithm conforming to the Framework #1 stops after a finite number of iterations with $x_k = x^*$, a stationary point of* **BQP**.

Proof: Note that, for quadratic objective functions, the conjugate reduced gradient method finds an optimal solution on any subspace of active constraints in a finite number of steps and hence it is a discrete method on each subspace. Therefore, if the forcing sequence η_k in (2) satisfies $\lim_{k\to\infty} \eta_k = 0$, for k large η_k is small and a relaxing step will be taken only after the minimum on some subspace of active constraints is found. As there are only finitely many different sets of active constraints and since the objective function in any algorithm conforming to Framework #1 is strictly decreasing, the optimal active set will be identified after a finite number of iterations. Thus the algorithm will terminate at a stationary point in a finite number of steps. q.e.d.

Proposition 3.3: *Assume that the quadratic programming problem* **BQP** *is nondegenerate, that is all stationary points x^* satisfy $g_i \neq 0$ for all $i \in \mathcal{B}(x^*)$. Then any algorithm conforming to the Framework #1 stops after a finite number of iterations with $x_k = x^*$, a stationary point of* **BQP**.

Proof: By Proposition 3.1 it follows that $\lim_{k\to\infty} x_k = x^*$ a stationary point, that is, $g^P(x^*) = 0$. We show that, for nondegenerate problems, the set of active constraints $\mathcal{A}(x_k)$ settles down after a finite number of iterations, that is, there exists a $k_0 \geq 0$ such that $\mathcal{A}(x_k) = \mathcal{A}(x^*)$ for all $k \geq k_1$. The result then follows since conjugate gradient directions are used when the active set remains constant.

Note that, for nondegenerate problems, the set $\mathcal{A}(\boldsymbol{x}^*) = \mathcal{B}(\boldsymbol{x}^*)$. Now let $i \in \mathcal{B}(\boldsymbol{x}^*)$, and assume without loss of generality that $x_i^* = l_i$ and $g_i(\boldsymbol{x}^*) > 0$. Since $\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}^*$ there exists a $k_0 \geq 0$ such that

$$g_i(\boldsymbol{x}_k) > 0 \text{ for all } k > k_0.$$

Assume $(\boldsymbol{x}_k)_i > l_i$ for all $k > k_0$. Then $i \notin \mathcal{B}_k$ for all $k > k_0$ and this implies that

$$0 = \lim_{k\to\infty} \|\boldsymbol{g}^P(\boldsymbol{x}_k)\| \geq \lim_{k\to\infty} |g_i(\boldsymbol{x}_k)| = |g_i(\boldsymbol{x}^*)| > 0$$

a contradiction. Therefore there exists a $k_1 \geq k_0$ such that $(\boldsymbol{x}_{k_1})_i = l_i$. Since $g_i(\boldsymbol{x}_k) > 0$ for all $k > k_1$, it follows that

$$(\boldsymbol{x}_k)_i = l_i \text{ and hence } i \in \mathcal{B}_k \subseteq \mathcal{A}_k \text{ for all } k > k_1.$$

q.e.d.

3.3. Relationship to existing algorithms

Many existing algorithms in the literature can be shown to fit into the Framework #1 by specifying the constraint relaxation condition, relaxing or restricted directions and the step size rule. These algorithms include gradient projection methods, algorithms based on pivoting using direct or iterative methods and modified conjugate gradient methods.

Gradient projection methods are a special case of our Framework #1 in which only projected gradient directions are used in (3), that is $\eta_k = \infty$ and $\beta_k = 0$ for all k. An example is the Goldstein-Levitin-Polyak gradient projection method [4]. This method consists of the iteration

$$\boldsymbol{x}_{k+1} = [\boldsymbol{x}_k - \alpha_k \, \boldsymbol{g}_k]^{\#}. \tag{9}$$

In the original algorithms of Goldstein [29] and Levitin and Polyak [31] the step size α_k was choosen to be constant for all k. An alternative method for selecting the step size α_k was proposed later by McCormick [32] in the context of feasible direction methods in order to avoid zigzagging. He extended his results of [32] to the case of general linear constraints and general closed convex sets in [33, 34]. McCormick suggested determining α_k by finding the first minimum of the objective function on the feasible arc $\{[\boldsymbol{x}_k - \alpha \, \boldsymbol{g}_k]^{\#} \mid \alpha \geq 0\}$. Since the exact minimization even in one dimension is impractical, Bertsekas' convergence result in [4], using a modified Armijo one dimensional search, was a significant contribution to making these algorithms useful in practice. Our linesearch is cheaper than the modified Armijo search in

Bertsekas [4], since it does not require the computation of an inner product each time the step is reduced.

Most other feasible descent methods have one aspect in common, namely they perform the one dimensional search along the direction $\boldsymbol{p}_k$ such that the new iterate lies on the ray defined by $\boldsymbol{x}_k$ and $\boldsymbol{p}_k$. In other words, they limit the step size in order to remain feasible without performing a projection onto the feasible region. For quadratic problems this corresponds to a step size $\alpha_k = \min\{\alpha_k^*, \alpha_k^{\max}\}$ where $\alpha_k^{\max} = \max \{ \alpha \geq 0 \mid \boldsymbol{l} \leq \boldsymbol{x}_k + \alpha\, \boldsymbol{p}_k \leq \boldsymbol{u} \}$. Therefore, unless there is some symmetry in the problem, these methods will usually add only one constraint at a time. Note that for such a step size

$$\boldsymbol{x}_{k+1} = [\boldsymbol{x}_k + \alpha_k\, \boldsymbol{p}_k]^{\#} = \boldsymbol{x}_k + \alpha_k\, \boldsymbol{p}_k \tag{10}$$

and no projection takes place. The methods differ only in their choice of descent directions and their rule for relaxing constraints.

Most algorithms for the solution of quadratic programs determine at each iteration one constraint to be dropped from the current active set and then compute a feasible step restricted to the subspace defined by the remaining active constraints. These include the primal methods of Beale [2, 3], Dantzig [17], Fletcher [22], Gill and Murray [24, 26], Goldfarb [28], and Wolfe [47], the principal pivoting methods of Cottle and Dantzig [13], the dual methods of Lemke [30] and Whinston [46, 44]. Under certain conditions most of these algorithms generate the same sequence of points, as shown by Best [6] and Pang [38].

Typically these methods require the solution of a linear system of equations to get a feasible descent direction and then the new iterate is obtained using (10). The difference between these various methods lies then in which constraint is to be dropped, how the linear systems are to be solved and how to take advantage of the special structure of some very large problems. Fletcher and Jackson [23] use a partial $\boldsymbol{LDL}^T$ factorization of $\boldsymbol{H}$ and describe how to update these factors efficiently. For a special class of very large quadratic programming problems **(BQP)**, where the objective functions have nonpositive mixed partial derivatives, iterative methods, like block SOR, have been proposed (see, for example, Céa and Glowinski [10], Cottle and Goheen [14] or Cryer [16]).

Another class of algorithms for problem **BQP** is based on modifications of the conjugate gradient method. Polyak [40] proposed the use of negative projected gradients as relaxing direction and conjugate reduced gradients as restricted directions combined with the step size (10) and solving the restricted subproblems to optimality. In our Framework #1 his algorithm is obtained by setting $\eta_k = 0$ for all k in (2), $\beta_k = 0$ for all k in (3) and using the one dimensional search (10) instead of (6), that is no projection is used. The solution of the restricted, equality-constrained subproblems in Polyak's algorithm may be improved upon using scaled conjugate reduced gradients as shown by O'Leary [36, 37]. It is easy to show that Polyak's algorithm converges in a finite number of iterations. It finds the minimum on each subspace in finitely many steps; there are only finitely many subspaces and being a descent method it never returns to the same subspace twice.

A feature of these algorithms is that they are able to drop many constraints in a single relaxing step and they do so only after having solved some reduced problem to optimality. O'Leary notes in [36] that it is not necessary to solve the subproblems to a high level of accuracy since their primary purpose is to provide some information on the next subspace to consider. However, she does not indicate how the tolerances are to be set for the subspace minimization and does not prove convergence. Furthermore, since all the methods above use the linesearch (10), they usually pick-up only one constraint at a time.

4. Truncated projection methods

Algorithms conforming to Framework #1 take into account all the constraints during restricted iterations. There are other active set methods, that include only a subset of the constraints at each major iteration in order to determine a search direction. In the following framework we solve the subproblems **inexactly** using a forcing sequence.

4.1. Algorithmic Framework #2

```
***START*** with $x_0$, a feasible solution
***SET*** k = 0, $\sigma \in (0,1)$ and $\gamma \in (0,\frac{1}{2})$
***WHILE*** "not optimal" ***DO***
```

1. (Determine a feasible descent direction $\boldsymbol{p}_k$)

   Find some ***approximation***, $\boldsymbol{p}_k$, to the optimal solution of the quadratic program $(\boldsymbol{QP})_k$

$$(\boldsymbol{QP})_k \qquad \begin{array}{ll} \textbf{\textit{minimize}} & Q_k(p) = \frac{1}{2}\, p^T H p + g_k{}^T p \\ \textbf{\textit{subject to}} & p_i = 0 \quad \text{for all } i \in \mathcal{B}_k \end{array} \tag{11}$$

   such that $Q_k(p_k) \leq 0$ and[3] $\|[Hp_k + g_k]^R\| \leq \eta_k \|g^P(x_k)\|$ (12)

2. (Determine an **acceptable** step-size α_k)

   $\alpha_k = \sigma^{m_k}\, \alpha_k^*$, where $\alpha_k^* = -\dfrac{g_k{}^T p_k}{p_k{}^T H p_k}$

   where m_k is the smallest nonnegative integer such that

$$Q([x_k + \alpha_k\, p_k]^{\#}) - Q(x_k) \leq \gamma\, \alpha_k\, g_k{}^T p_k \tag{13}$$

3. (Update)

   SET $x_{k+1} = [x_k + \alpha_k\, p_k]^{\#}$ and $k \leftarrow k + 1$.

Note that $Q(x_k + p) = Q(x_k) + Q_k(p)$.

There is no justification to solve the subproblems in Step 1 exactly when far away from the optimal solution. We propose solving the subproblems approximately using a conjugate reduced gradient method which is truncated when the norm of the reduced gradient has been decreased "sufficiently". The resulting method is referred to as a **truncated projection method (TP)**.

Remark 4.1: If a conjugate gradient method is used for solving the subproblems in Framework #2 then the initial step size in the one dimensional search is $\alpha_k^* = 1$ for all k, because conjugate gradient methods determine the minimum of a quadratic on the subspace generated by the previous gradients (see for example [12]) and the new descent direction p_k lies in this subspace.

[3]Here $[Hp_k + g_k]^R$ is the reduced gradient at p_k of the problem $(QP)_k$ where the reduction is taken with respect to $\mathcal{B}_k$.

Remark 4.2: Since the subproblems are equality constrained the one dimensional backtracking search needs to be performed only **once per major iteration.** In Framework #1 this one dimensional search had to be performed **at every iteration**. This might be prohibitively expensive for more general problems where the projection may be expensive to compute.

Proposition 4.1: *Let $\{x_k\}$ be a sequence generated by any algorithm satisfying Framework #2 with $\eta_k \leq \eta < 1$. Then there either exists a $k \geq 0$ such that $x_k = x^*$ or $\lim_{k\to\infty} x_k = x^*$, where x^* is a stationary point of* **BQP**.

Proof: First we show that the algorithm is well defined. If x_k is not a stationary point, that is $g^P(x_k) \neq 0$, then $p_k \neq 0$ because $\eta_k < 1$. Furthermore

$$Q_k(p_k) = \tfrac{1}{2}\, p_k{}^T H_k p_k + g_k{}^T p_k \leq 0.$$

and since H_k is positive definite it follows that

$$g_k{}^T p_k \leq -\tfrac{1}{2}\, p_k{}^T H_k p_k \leq -\tfrac{1}{2}\, \mu\, \|p_k\|^2 < 0\,. \tag{14}$$

By definition of the subproblems $(p_k)_i = 0$ for all i in $\mathcal{B}_k$ and therefore by Lemma 3.1 the backtracking line search and hence x_{k+1} are well defined.

Now assume that there exists a subsequence $\{x_{k_i}\}$ which converges to a nonoptimal point $\bar{x}$ of *BQP*. Since H_k is positive definite the objective function is bounded from below and therefore

$$\lim_{k\to\infty} [Q(x_{k+1}) - Q(x_k)] = 0.$$

Hence the Armijo condition (13) implies that

$$\lim_{k\to\infty} \alpha_k g_k{}^T p_k = 0. \tag{15}$$

It follows from inequality (14) that

$$\limsup_{i\to\infty} g_{k_i}{}^T p_{k_i} < 0,$$

and therefore equation (15) implies that

$$\liminf_{i\to\infty} \alpha_{k_i} = 0.$$

Let $\{k_j\}$ be subsequence such that $\lim_{j\to\infty} \alpha_{k_j} = 0$. Since $\alpha_k = \sigma^{m_k} \alpha_k{}^*$ and by (14) $\alpha_k{}^* \geq \frac{1}{2}$ it follows that $\lim_{j\to\infty} m^{k_j} = \infty$. Therefore the definition of m_k in the backtracking line search (13) implies that, for j large enough,

$$[Q([x_{k_j} + (\alpha_{k_j}/\sigma)\, p_{k_j}]^{\#}) - Q(x_{k_j})]/(\alpha_{k_j}/\sigma) > \gamma g_{k_j}{}^{T} p_{k_j}, \tag{16}$$

that is, the step is reduced at least once. Since H is positive definite $\{p_k\}$ is bounded and there exists a further subsequence $\{k_l\}$ of $\{k_j\}$ such that $\lim_{l\to\infty} p_{k_l} = \bar{p}$. Now taking limits in (16) and using Lemma 3.1 yields

$$\bar{g}^T\bar{p} \geq Q'(\bar{x};\bar{p}) \geq \gamma\bar{g}^T\bar{p} \text{ and hence } (1-\gamma)\, \bar{g}^T\bar{p} \geq 0$$

a contradiction to $(1-\gamma) > 0$ and $\bar{g}^T\bar{p} < 0$. Therefore any limit point of $\{x_k\}$ is a stationary point of **BQP**.

Since H is positive definite the sequence $\{x_k\}$ is bounded and therefore has a limit point x^* which is a stationary point of **BQP** and, as in the proof of Proposintion 3.1, it follows that

$$Q(x_k) - Q(x^*) \geq \tfrac{1}{2}\, \mu \, \|x_k - x^*\|^2$$

and therefore $\lim_{k\to\infty} x_k = x^*$. q.e.d.

Similar to Proposition 3.3 we have for nondegenerate problems that any algorithm conforming to Framework #2 has the finite termination property.

Proposition 4.2: *Assume that the quadratic programming problem* **BQP** *is nondegenerate. Then any algorithm conforming to Framework #2, with $\eta_k \leq \eta < 1$ for all k and using any finte method (such as conjugate gradient method) on the subproblems, stops after a finite number of iterations with $x_k = x^*$, a stationary point of* **BQP**.

5. Numerical Experiments

In order to test the above algorithms we implemented our Framework #1 in a FORTRAN code on a DEC-20 computer running under the TOPS-20 operating system. Through simple switches in the code the following algorithms were obtained.

(a) **CRGP** - conjugate reduced gradient projection method as described in Framework #1;

(b) **CRG** - conjugate reduced gradient method as described in Framework #1 using the optimal feasible step size (9);

(c) **CGP** - conjugate gradient projection method as described in Framework #1 with $\eta_k = \infty$; and

(d) **TP** - truncated projection method as described in Framework #2.

Except for the **CGP** algorithm, in all cases we tested two versions of the algorithms, namely one using exact subspace minimization and one which performed truncated subspace minimization using a forcing sequence. For exact subspace minimization the **CRG** reduces to Polyak's reduced conjugate gradient method [40] and **TP** reduces to Bertsekas' Newton projection method [5] where ϵ-active sets with ϵ=0 are used.

In all algorithms, unless indicated otherwise, we used the forcing sequence[4]

$$\eta_k = \min \{ \eta \, \|g_k^P\|, \|g_k^P\|^2 \} \tag{17}$$

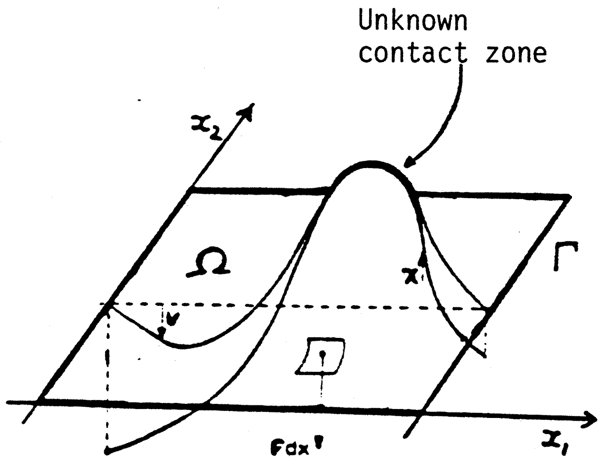
with $\eta = .03$ for controlling the truncated subspace minimization in (2) of Framework #1 and (12) of Framework #2. With the exception of the **CRG**, where the optimal feasible step size (10) was implemented, we used the linesearch parameters $\sigma = .6$ in (5) and $\gamma = .1$ in (6) for controlling the bending one dimensional search. The stopping criterion for all algorithms on all problems tested required the norm of the projected gradient g^P to be less than an error tolerance of 10^{-5}.

5.1. The obstacle problem

There are many partial differential equation problems where the nature of the underlying physical problem imposes some bounds on the solution. Dirichlet problems with obstacles as discussed by Ciarlet [11] are an example of this. The ***obstacle problem*** consists of finding the equilibrium position of an elastic membrane which passes through a curve Γ. That is, the boundary of an open set Ω of a "horizontal" plane, is subjected to the action of a "vertical" force f and must lie over an obstacle which is represented by a function $\chi:\overline{\Omega}\rightarrow R$, and under a second obstacle which is represented by a function $\psi:\overline{\Omega}\rightarrow R$ as illustrated in Figure 5-1.

[4]This forcing sequence worked well on the problems tested but it has a significant drawback in that it is scale dependent.

Figure 5-1: The obstacle problem



The basic mathematical problem may be stated as

$$\textbf{\textit{minimize}} \quad J(v) = \int_{\Omega} \|\text{grad } v\|^2 - fv \, dx$$

$$\textbf{\textit{subject to}} \quad v \in \{ w \in H_0^1(\Omega) \mid \chi \leq w \leq \psi \}$$

where $H_0^1(\Omega)$ is the space of those functions in $L^2(\Omega)$ for which all first partial derivatives (in the distribution sense) belong to the space $L^2(\Omega)$ and the functions are zero on Γ the boundary of Ω. The function f is in $L^2(\Omega)$ and the obstacles χ and ψ are in $H^1(\Omega)$, $\chi \leq 0$ and $\psi \geq 0$ on Γ and $\chi \leq \psi$ almost everywhere on Ω (for more details see Ciarlet [11]).

For our tests we let Ω be the unit square in R^2, discretized with parameter $h = (m+1)^{-1}$. The finite difference approximation to this problem using the five point symmetric Laplace operator yields a quadratic programming problem of the form **BQP** where the Hessian H is block tridiagonal with minus the identity off the diagonal and tridiagonal matrices on the diagonal which have fours on the diagonal and minus ones off the diagonal. The linear term in the quadratic objective c has as elements $c_i = -(m+1)^{-2}$ if we choose the force to be equal to 1 on Ω, that is $f(x) = 1$ for all x in the unit square of R^2. The lower bound l and the upper bound

$\boldsymbol{u}$ are defined by the discretization of the obstacles χ and ψ respectively. The number of variables in this problem is $n = m^2$ since the functions $\boldsymbol{v}$ are zero on $\boldsymbol{\Gamma}$ the boundary of $\boldsymbol{\Omega}$.

5.2. Computational Results on the Obstacle Problem

In our first series of test problems, Tables 5-2, 5-3 and 5-4, we only consider the obstacle from below. The lower bound $\boldsymbol{l}$ is the discretization of $\chi(\boldsymbol{x}) = p_1 \, [\, \sin(3.2x_1) \sin(3.3x_2) \,]^{p_2}$ where the numbers p_1 and p_2 specify the parameters of the obstacle χ. The upper bound is redundant for this problem and we therefore set $u_i = 2000.$ for all i. The initial starting points are either $\boldsymbol{x}_0 = \boldsymbol{l}$, where many constraints are binding and have to be dropped during the iterations, or $\boldsymbol{x}_0 = \mathbf{1}$, the vector with all elements equal to 1, with no constraints binding.

In the second series of test problems, Tables 5-5 and 5-6, both obstacles are active at the optimal solution. In Table 5-5 similar obstacles are used. Namely, the lower bound $\boldsymbol{l}$ is the discretization of $\chi(\boldsymbol{x}) = [\, \sin(9.2x_1)\sin(9.3x_2) \,]^{p_1}$ and the upper bound $\boldsymbol{u}$ is the discretization of $\psi(\boldsymbol{x}) = [\, \sin(9.2x_1)\sin(9.3x_2) \,]^{p_2} + .02$. For the obstacles in Table 5-6 we chose polonomial functions. Namely, the lower bound $\boldsymbol{l}$ is the discretization of the lower obstacle $\chi(\boldsymbol{x}) = [\, 16 \, x_1(1-x_1) \, x_2(1-x_2) \,]^{p_1}$ and the upper bound $\boldsymbol{u}$ is the discretization of the upper obstacle $\psi(\boldsymbol{x}) = [\, 16 \, x_1(1-x_1) \, x_2(1-x_2) \,]^{p_2} + .01$. The pattern of the constraints active at the optimal solution for both these problems are shown in Figure 5-2 and 5-4.

In all tables, $\#\mathcal{B}(\boldsymbol{x})$ indicates the number of binding constraints at the point $\boldsymbol{x}$. The column $\|g_0^P\|$ gives the norm of the projected gradient at the initial solution $\boldsymbol{x}_0$, which is a measure of how close the initial estimate is to the optimal solution. As a **benchmark** we use the number of conjugate gradient iterations it takes to solve the problem **if the correct set of active constraints $\mathcal{A}(\boldsymbol{x}^*)$ would have been known**, that is, $(\boldsymbol{x}_0)_i = (\boldsymbol{x}^*)_i$ for all $i \in \mathcal{A}(\boldsymbol{x}^*)$ (see column **CAS**). We also use the number of conjugate gradient iterations needed to solve the unconstrained problem from the the same starting point $\boldsymbol{x}_0$ (see column **UC**). The remaining columns show the number of minor iterations needed by the four different algorithms tested to reduce the norm of the projected gradient to the error tolerance indicated above. Minor iterations refer to the number of restricted steps taken in Framework #1 and to the number of iterations on the quadratic subproblem in Framework #2. With the exception of Table 5-2, the numbers in brackets indicate the number of major (relaxing) iterations for the **CRGP**, **TP** and the **CRG** algorithms. Note that the **CGP** algorithm uses only relaxing steps.

5.2.1. The effect of the bending one dimensional search

The **CRG** method uses the optimal feasible step size (9) and therefore can usually only add one constraint at a time. All tables indicate immediately that if the optimal active set contains many more constraints than the initial active set, then the **CRG** algorithm requires at least that many iterations. Hence, for example, in Table 5-3 for the 10,000 variable obstacle problem over 6,000 constraints are binding at x^* and the starting point is totally unconstrained. It takes over 7,000 iterations for **CRG** to identify an optimal binding set. All the other three methods however use the bending one dimensional search (7) which allows them, using a projection onto the feasible region, to pick-up many constraints per search direction computation. This leads to radical improvements over **CRG** when constraints have to be added. However, when constraints are only to be dropped from the binding set, as is the case for the initial guess $x_0 = l$ in Tables 5-2, 5-4 and 5-3, then the bending one dimensional search yields no improvement over the linesearch in the **CRG** algorithm. The **CRG** algorithm with $\eta_k = 0$ for all k is Polyak's algorithm [40] and use of the forcing sequence (inexact subspace minimization) does improve its performance, although not radically so.

5.2.2. Effect of the forcing sequence

After having discussed the problem of adding constraints we now examine the problem of dropping constraints from the active set. In Table 5-2 the number in brackets for the **CRGP**, **TP** and the **CRG** is the number of minor iterations that were required to solve the problems when exact subspace minimization was peformed. This means the forcing sequence $\eta_k = 0$ for all k and **CRG** corresponds to the original Polyak algorithm [40] and **TP** reduces to a special case of Bertsekas' Newton projection method. It is easily seen from Table 5-2 that the effect of inexact subspace minimization the forcing sequence (17) is to reduce the total number of minor iterations, and therefore the work required to solve the problem, by a factor of two. This remains true for **CRG** only in the case when $x_0 = l$, which means that many constraints are binding at x_0 and that constraints only need to be dropped. For the starting point $x_0 = \mathbf{1}$, where all the constraints that are binding at the optimal solution have to be added, the poor performance of an optimal feasible linesearch overshadows the gains made by inexact subspace minimization.

Other forcing sequences have been tried as well, for example $\eta = .1$ in (17) as used in the runs for Tables 5-5 and 5-6 or the sequence $\eta_k = \eta^k \|g_0^P\|$. Generally speaking the performance of the algorithms is **not very sensitive to the forcing sequence**, as long as it does not go to zero too fast. This is particularly important if many constraints are to be dropped, as the example of **CGP** shows. This method consists solely of relaxing iterations ($\eta_k = \infty$ for all k) and therefore is superior to all the other algorithms for the starting point $x_0 = l$, (as can be seen in Tables 5-2, 5-3 and 5-4).

5.2.3. Comparison of CRGP, TP and CGP

As noted above for problems in which constraints are only dropped from the initial active set **CGP** outperforms any of the other algorithms tested. This remains true on average if many constraints are to be dropped and added during the iterations. When many constraints are to be added, as for the starting point $x_0 = \mathbf{1}$ in Tables 5-2, 5-3 and 5-4 or for the starting point $x_0 = \frac{1}{2}(l+u)$ in Tables 5-5 and 5-6, the performance of the **CGP** lies between that of **CRGP** and that of **TP** for most of the larger problems.

The performance of **CGP** on the obstacle problem with the parameters $p_1 = .3$ and $p_2 = 1$ in Tables 5-3 and 5-4. is uncharacteristically very poor. It appears as if the reason for this is that the problem is (numerically) degenerate as can be seen in Figure 5-5 where there are binding constraints with very small multipliers. Degeneracy affects **CGP** more than the other algorithms because the zigzagging between constraints results in many steps being projected gradient steps. Hence **CGP** never gets the benefit of conjugate gradient directions. The other algorithms are less affected by this because they are required to take restricted steps, which have a stabilizing effect. Perhaps it is best to use a very mild forcing sequence.

For the larger problems the difference between in the performance of **CRGP** and **TP** may be summarized as follows. If many constraints are binding at the optimal solution then **TP** seems to be able to add constraints more easily. This may be due to the fact that the solution of the unconstrained problem is computed and projected back onto the feasible region. However, if fewer constraints are binding at x^* then **TP** appears to add too many constraints at a time which then have to be dropped again. Therefore **CRGP** outperforms **TP** when the starting point is strictly feasible (see Tables 5-3 , 5-4, 5-5 and 5-6).

5.3. A network flow problem of 916 variables

For the second series of tests we used the test problem generator for large scale unconstrained optimization of Dembo and Steihaug [20] to generate quadratic programming problems. We then added box constraints to these problems so as to create some problems with widely differing optimal active sets. Some of our experiments are reported in Table 5-1. Again **CRGP, TP and CGP are essentially insensitive to the number of constraints binding at the optimal solution**, whereas the **CRG** algorithm, as expected needs as least as many iterations as there are constraints to be added to the set of binding constraints at the optimum.

Conclusions

An important aspect of the new algorithms that have been introduced in this paper is the projection linesearch. We have been able to show that a standard Armijo-type termination test (6) and (13), respectively, is sufficient to guarantee convergence. This result extends to more general objective functions and will be presented in a subsequent paper. The termination rules (6) and (13) are more convenient and cheaper to evaluate than those introduced by Bertsekas [4, 5].

Although we have assumed throughout that the matrix H is positive definite there exists simple modifications to these algorithms that would handle indefinite matrices in a practical implementation. One such idea, which involves an additional test in the conjugate gradient algorithm, is described in Dembo and Steihaug [21].

On the empirical side, the algorithms **CRGP**, **TP** and **CGP** appear to be all remarkably insensitive to the number of constraints that are active at the optimum. In all cases they required on average between two and four times the work needed to solve the same problem with **the optimal active set known a priori**. This was not the case for **CRG** which uses a traditional active set strategy and is very sensitive to the number of constraints active at the optimum.

The conjugate gradient projection algorithm (**CGP**) was by far the best method overall and its only drawback appears to be its sensitivity to degeneracy. It definitely merits further attention. The conjugate reduced gradient projection method (**CRGP**) appears to be less sensitive to degeneracy and not much slower than **CGP** in many cases. For general purpose codes it would probably be preferable to **CGP**.

We did not attempt to test the affect of preconditioning on the behavior of these algorithms since the choices of preconditioner would have made it more difficult to interpret the outcomes. Undoubtedly, a careful choice of preconditioner will enhance their behavior and possibly alter their relative performance.

There are a number of factors in our numerical experiments that caution against extrapolating from these results. Firstly, only a limited class of test problems was solved. Secondly, **for more general constraint sets** the use of conjugate gradient iterations is likely to be a poor measure of performance because the overhead will vary considerably among algorithms.

Acknowledgements

The authors are indebted to Stanley C. Eisenstat for suggestions and discussions on the obstacle test problem. We would also like to thank the Research Center for Scientific Computation in the Department of Computer Science at Yale University for providing the material support for this paper.

Table 5-1: The network problem (n = 916).

						Benchmarks		Algorithms			
x_0	$\#\mathcal{B}(x^*)$	$\#\mathcal{B}(x_0)$	l	u	$\|g_0^P\|$	CAS	UC	CRGP	TP	CRG	CGP
						number of minor iterations (number of major iterations)[3]					
1	688	0	-1.1	100.	10^5	31	50	$\underline{45}$ (6)	$\underline{41}$ (6)	746 (7)	$\underline{45}$
1	493	0	-1.3	100.	10^5	36	50	57 (6)	$\underline{49}$ (6)	550 (6)	56
1	222	0	-1.6	100.	10^5	40	50	63 (6)	$\underline{49}$ (6)	262 (6)	62

Table 5-2: The obstacle problem (n = 2,601): Lower bounds only.

						Benchmarks		Algorithms			
x_0	$\#\mathcal{B}(x^*)$	$\#\mathcal{B}(x_0)$	p_1	p_2	$\|g_0^P\|$	CAS	UC	CRGP	TP	CRG	CGP
					number of minor iterations (number minor iterations if η_k=0 for all k)[3]						
l	1671	2276	1.	1	.84	52	103	151 (305)	151 (305)	154 (306)	$\underline{79}$
l	1255	1846	.3	1	.84	58	93	191 (335)	191 (335)	193 (335)	$\underline{142}$
l	365	843	1.	2	.17	85	105	241 (496)	238 (495)	243 (496)	$\underline{126}$
l	197	554	1.	3	.15	90	107	275 (563)	270 (561)	279 (565)	$\underline{115}$
1	1671	0	1.	1	14.	70	84	$\underline{212}$(334)	$\underline{196}$(376)	1765 (1846)	253
1	1255	0	.3	1	14.	81	84	277 (458)	$\underline{232}$(418)	1570 (1744)	508
1	365	0	1.	2	14.	108	84	$\underline{200}$(360)	288 (583)	570 (611)	$\underline{202}$
1	197	0	1.	3	14.	116	84	$\underline{203}$(382)	318 (632)	457 (505)	$\underline{201}$

The lower obstacle is $\chi(x) = p_1[\sin(3.2x_1)\sin(3.3x_2)]^{p_2}$ and the upper obstacle is $\psi(x) = 2000$.

Table 5-3: The obstacle problem (n = 10000): Lower bounds only.

						Benchmarks		Algorithms			
x_0	$\#\mathcal{B}(x^*)$	$\#\mathcal{B}(x_0)$	p_1	p_2	$\|g_0^P\|$	CAS	UC	CRGP	TP	CRG	CGP
						number of minor iterations (number of major iterations)[3]					
l	6157	8548	1.	1	.4	104	195	479 (19)	475 (19)	483 (19)	$\underline{310}$
l	4638	6954	.3	1	.4	110	176	558 (19)	551 (19)	616 (18)	$\underline{352}$
l	1321	3220	1.	2	.1	163	203	714 (15)	704 (15)	851 (15)	$\underline{470}$
l	704	2159	1.	3	.1	167	197	756 (15)	734 (15)	794 (15)	$\underline{464}$
1	6157	0	1.	1	20.	134	164	789 (12)	$\underline{504}$(14)	7231 (20)	681
1	4638	0	.3	1	20.	157	164	960 (10)	$\underline{873}$(17)	6060 (12)	1733
1	1321	0	1.	2	20.	208	164	661 (5)	$\underline{483}$(10)	1956 (5)	596
1	704	0	1.	3	20.	221	164	$\underline{423}$(6)	833 (15)	1527 (5)	661

The lower obstacle is $\chi(x) = p_1[\sin(3.2x_1)\sin(3.3x_2)]^{p_2}$ and the upper obstacle is $\psi(x) = 2000$.

[3]The underlined numbers indicate runs within 10% of the best.

Table 5-4: The obstacle problem (n = 5,041): Lower bounds only.

						Benchmarks		Algorithms			
x_0	$\#\mathcal{B}(x^*)$	$\#\mathcal{B}(x_0)$	p_1	p_2	$\|g_0^P\|$	CAS	UC	CRGP	TP	CRG	CGP
						number of minor iterations (number of major iterations)[4]					
l	3150	4328	1.	1	.99	76	142	268 (14)	266 (14)	271 (14)	$\underline{147}$
l	2389	3543	.3	1	.33	80	126	340 (15)	329 (15)	339 (15)	$\underline{204}$
l	679	1609	1.	2	.14	117	145	397 (11)	393 (11)	434 (12)	$\underline{237}$
l	371	1074	1.	3	.10	122	145	451 (11)	442 (11)	467 (12)	$\underline{193}$
1	3150	0	1.	1	17.	96	117	501 (9)	338 (15)	3356 (5)	$\underline{243}$
1	2389	0	.3	1	17.	112	117	532 (13)	$\underline{397}$ (14)	3124 (8)	701
1	679	0	1.	2	17.	149	117	$\underline{334}$ (5)	461 (12)	1010 (5)	$\underline{366}$
1	371	0	1.	3	17.	158	117	$\underline{285}$ (4)	504 (12)	7687 (4)	418

The lower obstacle is $\chi(x) = p_1[\sin(3.2x_1)\sin(3.3x_2)]^{p_2}$ and the upper obstacle is $\psi(x) = 2000$.

Table 5-5: The obstacle problem (n = 5,041).

						Benchmarks		Algorithms			
x_0	$\#\mathcal{B}(x^*)$	$\#\mathcal{B}(x_0)$	p_1	p_2	$\|g_0^P\|$	CAS	UC	CRGP	TP	CRG	CGP
						number of minor iterations (number of major iterations)[4]					
u	1339	3041	3	2	1.5	87	145	232 (12)	$\underline{202}$ (12)	526 (13)	$\underline{216}$
l	1339	2348	3	2	1.3	86	138	268 (12)	210 (12)	1197 (12)	$\underline{198}$
$\frac{l+u}{2}$	1339	0	3	2	1.3	81	148	$\underline{120}$ (5)	$\underline{132}$ (6)	1539 (6)	143

The lower obstacle is $\chi(x) = [\ \sin(9.2x_1)\sin(9.3x_2)\ ]^{p_1}$
and the upper obstacle is $\psi(x) = [\ \sin(9.2x_1)\sin(9.3x_2)\ ]^{p_2} + .02$.

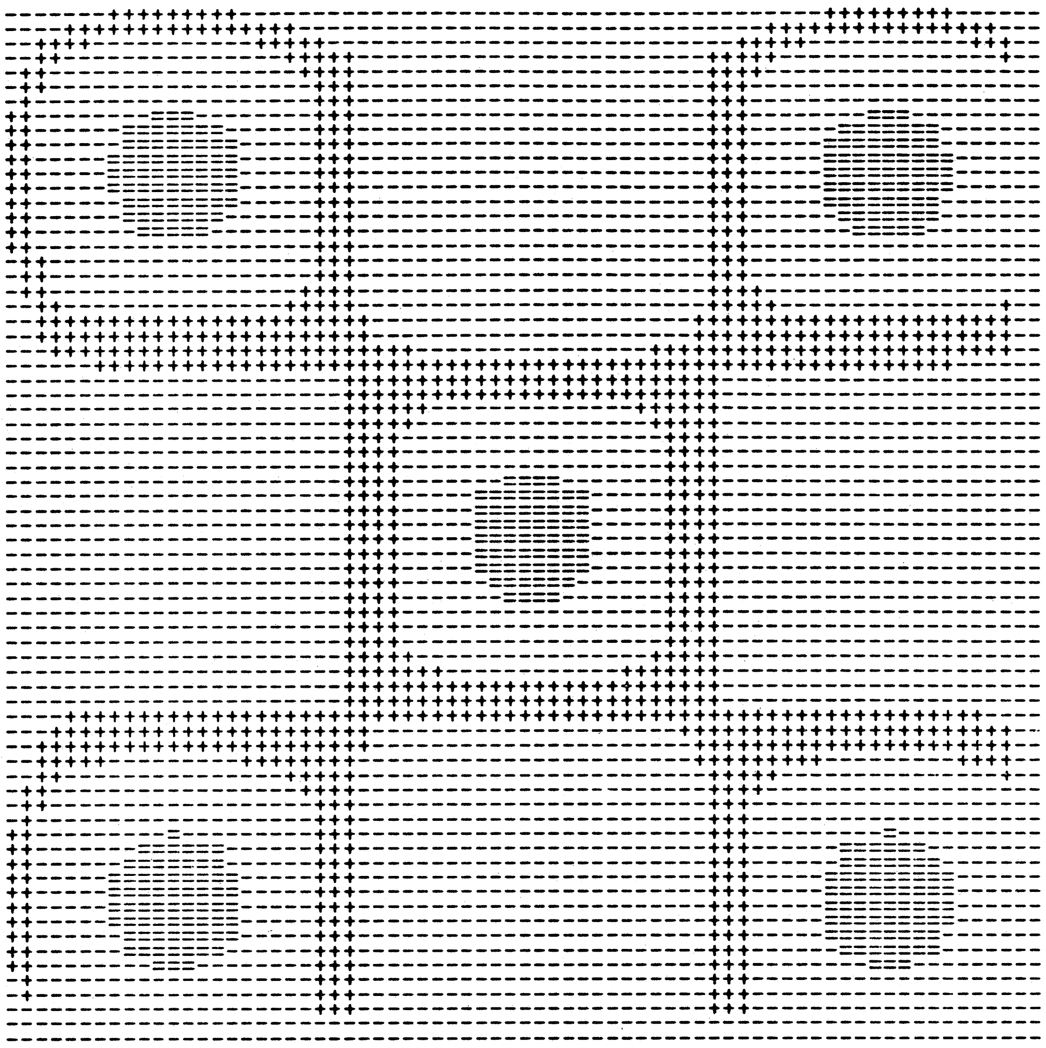
Table 5-6: The obstacle problem (n = 5,041).

						Benchmarks		Algorithms			
x_0	$\#\mathcal{B}(x^*)$	$\#\mathcal{B}(x_0)$	p_1	p_2	$\|g_0^P\|$	CAS	UC	CRGP	TP	CRG	CGP
						number of minor iterations (number of major iterations)[4]					
u	1781	2708	3	2	.18	68	100	289 (13)	267 (12)	378 (13)	$\underline{207}$
l	1781	1345	3	2	.13	68	88	255 (11)	213 (11)	388 (11)	$\underline{144}$
$\frac{l+u}{2}$	1781	0	3	2	.2	65	97	$\underline{110}$ (5)	191 (7)	294 (4)	$\underline{122}$

The lower obstacle is $\chi(x) = [\ 16\ x_1(1\text{-}x_1)\ x_2(1\text{-}x_2)\ ]^{p_1}$
and the upper obstacle is $\psi(x) = [\ 16\ x_1(1\text{-}x_1)\ x_2(1\text{-}x_2)\ ]^{p_2} + .01$.

[4] The underlined numbers indicate runs within 10% of the best.

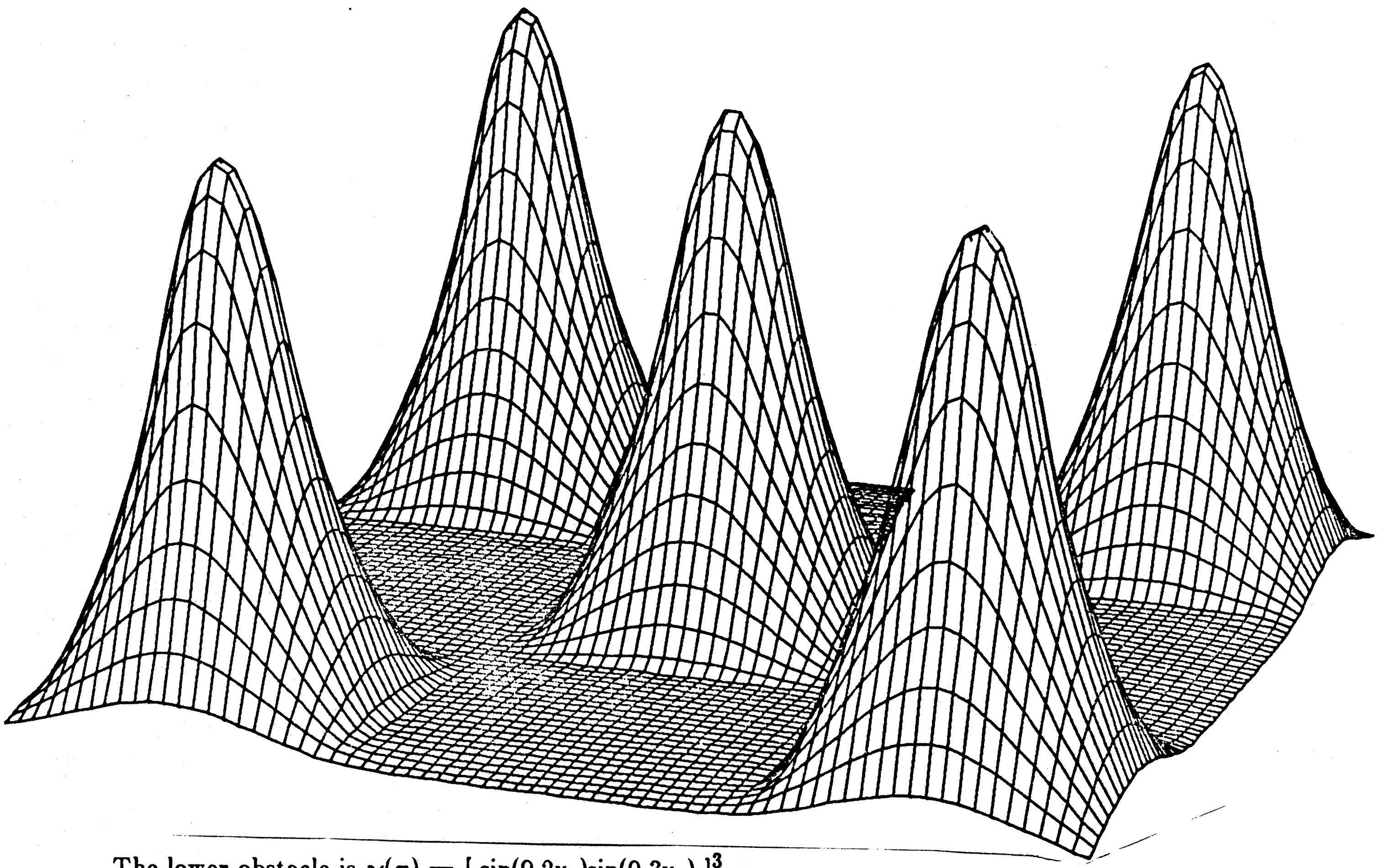
Figure 5-2: The obstacle problem (n = 5,041): Pattern of active constraints #1.



- : free varibles, + : upper bound is binding, = : lower bound is binding

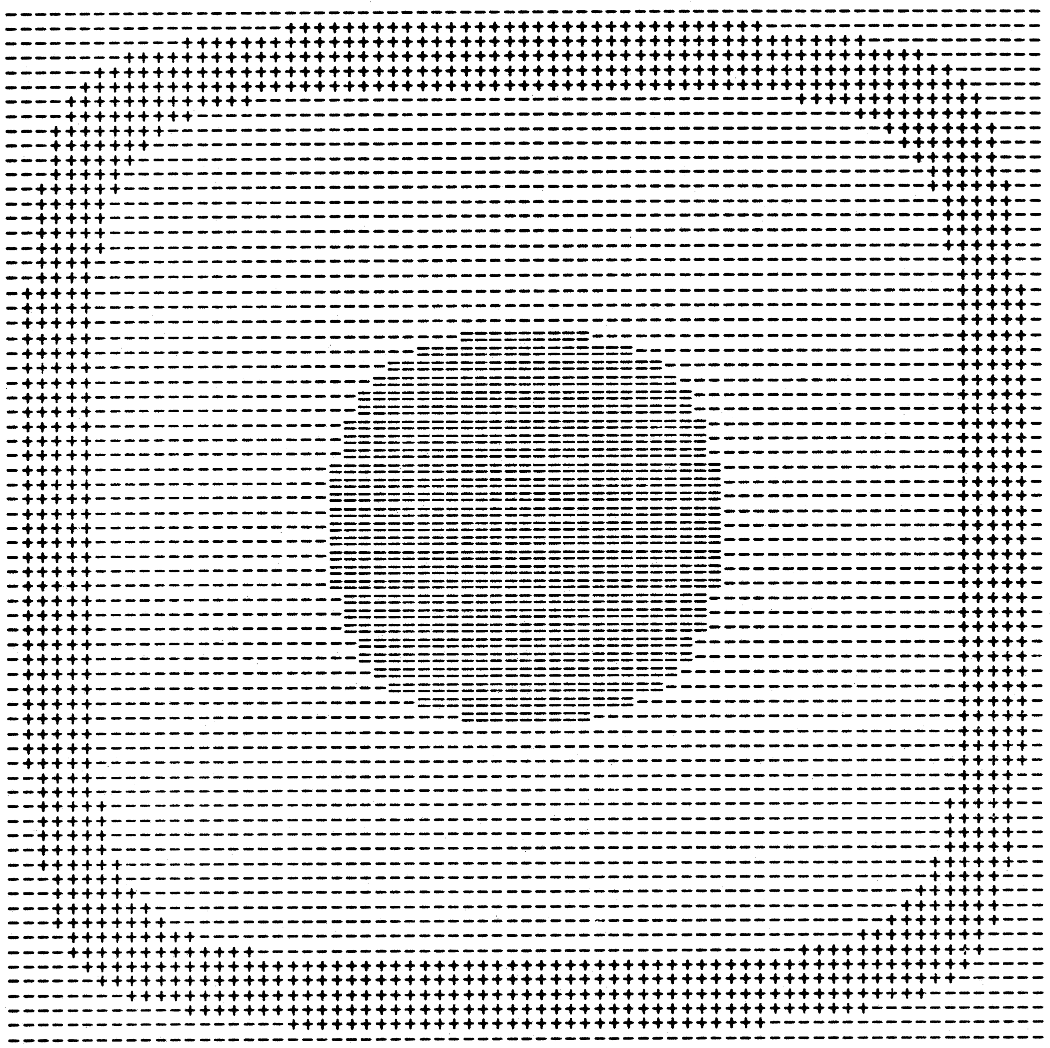
The lower obstacle is $\chi(\boldsymbol{x}) = [\ \sin(9.2x_1)\sin(9.3x_2)\ ]^3$
and the upper obstacle is $\psi(\boldsymbol{x}) = [\ \sin(9.2x_1)\sin(9.3x_2)\ ]^2 + .02$.

Figure 5-3: The solution (m = 71, n = 5,041)



The lower obstacle is $\chi(\boldsymbol{x}) = [\ \sin(9.2x_1)\sin(9.3x_2)\ ]^3$
and the upper obstacle is $\chi(\boldsymbol{x}) = [\ \sin(9.2x_1)\sin(9.3x_2)\ ]^2 + .2$.

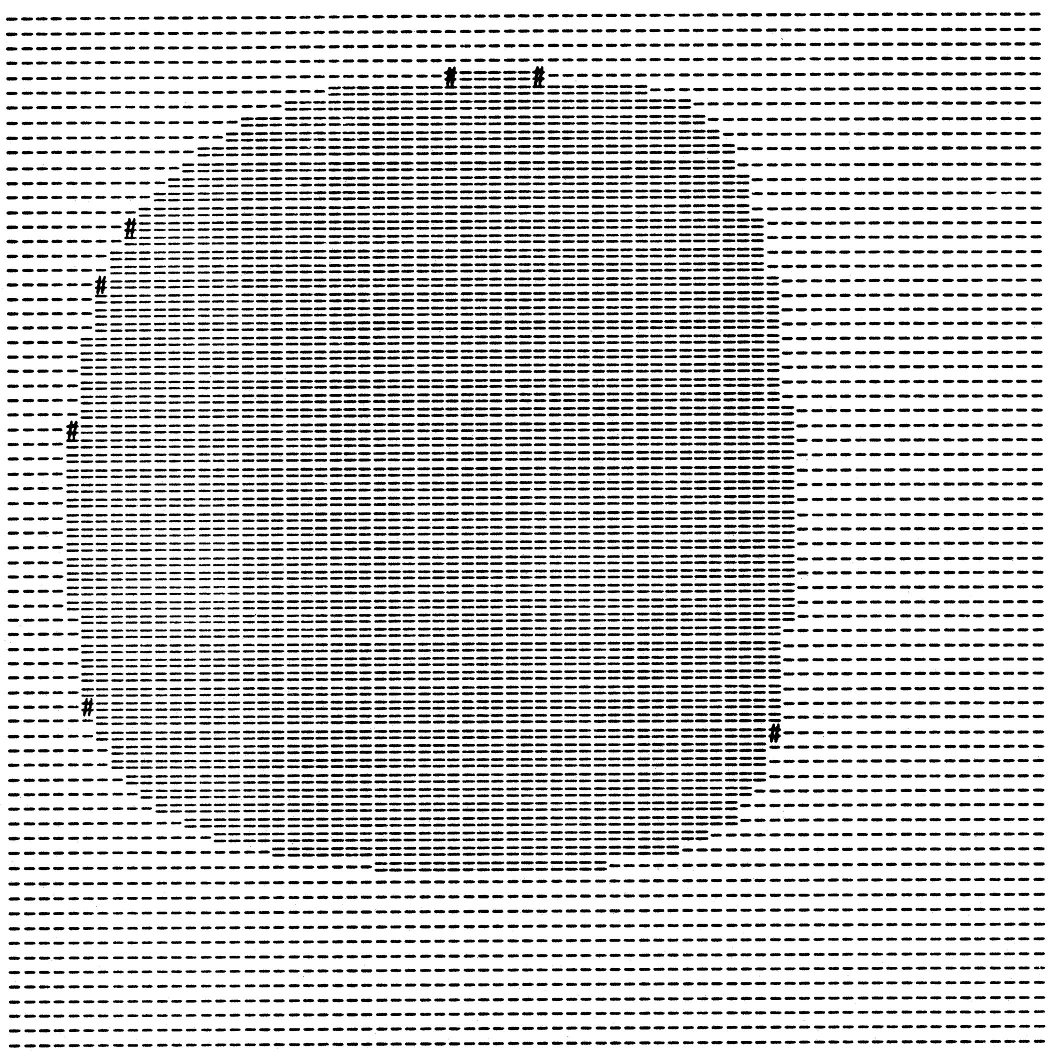
Figure 5-4: The obstacle problem (n = 5,041): Pattern of active constraints #2.



- : free varibles, + : upper bound is binding, = : lower bound is binding

The lower obstacle is $\chi(\boldsymbol{x}) = [\ 16\ x_1(1\text{-}x_1)\ x_2(1\text{-}x_2)\ ]^3$
and the upper obstacle is $\psi(\boldsymbol{x}) = [\ 16\ x_1(1\text{-}x_1)\ x_2(1\text{-}x_2)\ ]^2 + .01$.

Figure 5-5: The obstacle problem (n = 5,041): Numerical degeneracy.



- : free varibles,   = : lower bound is binding,

: lower bound is binding and $|g_i(\boldsymbol{x}^*| < 10^{-5}$.

The lower obstacle is $\chi(\boldsymbol{x}) = .3\ [\ \sin(9.2x_1)\sin(9.3x_2)\ ]$ and the upper obstacle is $\psi(\boldsymbol{x}) = 2000$.

References

[1] C. Baiocchi, V. Comineioli, E. Magenes and G.A. Pozzi.
Free boundary problems in the theory of fluid flow through porous media.
Ann. Mat. Pura. Appl. 97:1-82, 1973.

[2] E.M.L. Beale.
On minimizing a convex function subject to linear inequalities.
Journal of the Royal Statistical Society 17:173-184, 1955.

[3] E.M.L. Beale.
On quadratic programming.
Naval Research Logistics Quarterly 6:227-243, 1959.

[4] D.P. Bertsekas.
On the Goldstein-Levitin-Polyak gradient projection method.
IEEE Transactions on Automatic Control AC-21:174-184, 1976.

[5] D.P. Bertsekas.
Projected Newton methods for optimization problems with simple constraints.
SIAM Journal of Control and Optimization 20:221-246, 1980.

[6] M.J. Best.
Equivalence of some quadratic programming algorithms.
Mathematical Programming 30:71-87, 1984.

[7] M.C. Biggs.
Constrained minimization using recursive quadratic programming.
In F.A. Lootsma (editor), *Numerical methods for nonlinear optimization*, pages 411-428. Academic Press, London, 1972.

[8] M.C. Bartholomew-Biggs.
An improved implementation of the recursive quadratic programming method for constrained minimization.
Technical report 105, Numerical Optimization Center, The Hatfield Polytechnic, Hatfield, England, 1979.

[9] J.F. Bonnans.
A variant of a projected variable metric method for bound constraint optimization problems.
Rapport de recherche 242, INRIA, Le Chesnay, France, 1983.

[10] J. Céa and R. Glowinski.
Sur les méthodes d'optimization par relaxation.
RAIRO R-3:5-32, 1973.

[11] P.G. Ciarlet.
The finite element method for elliptique problems.
North-Holland, Amsterdam, 1978.

[12] P.G. Ciarlet.
Introduction à l'Analyse Numérique Matricielle et à l'Optimisation.
Masson, Paris, 1982.

[13] R.W. Cottle and G.B. Dantzig.
Complementary pivot theory of mathematical programming.
Linear Algebra and Its Applications 1:103-125, 1968.

[14] R.W. Cottle and M.S. Goheen.
A special class of large quadratic programs.
In O.L. Mangasarian, R.R. Meyer and S.M. Robinson (editor), *Nonlinear Programming 3*, pages 361-390. Academic Press, London, 1978.

[15] C.W. Cryer.
The method of Christopherson for solving free boundary problems for infinite journal bearing by means of finite differences.
Mathematics of Computation 25:435-443, 1971.

[16] C.W. Cryer.
The solution of a quadratic programming problem using systematic overrelaxation.
Journal of SIAM Control 9:385-392, 1971.

[17] G.B. Dantzig.
Linear programming and extensions.
Princeton University Press, Princeton, 1963.

[18] R.S. Dembo.
Large scale nonlinear optimization.
In M.J.D. Powell (editor), *Nonlinear Optimization 1981*, pages 361-372. Academic Press, London, 1982.

[19] R.S. Dembo and S. Sahi.
A Convergent Active Set Strategy for Linearly Constrained Optimization.
Working Paper Series B #80, School of Organization and Management, Yale University, New Haven, Connecticut, USA, 1984.

[20] R.S. Dembo and T. Steihaug.
A test problem generator for large scale unconstrained optimization.
Working Paper Series B #64, School of Organization and Management, Yale University, New Haven, Connecticut, USA, 1983.

[21] R.S. Dembo and T. Steihaug.
Truncated Newton algorithms for large-scale unconstrained optimization.
Mathematical Programming 26:190-212, 1983.

[22] R. Fletcher.
A general quadratic programming algorithm.
Journal of the Institute of Mathematics and Applications 7:76-91, 1971.

[23] R. Fletcher and M.P. Jackson.
Minimization of a quadratic function of many variables subject only to lower and upper bounds.
Journal of the Institute of Mathematics and Applications 14:159-174, 1974.

[24] P.E. Gill and W. Murray.
Minimization subject to bounds on the variables.
Report NAC-71, National Physical Laboratory, England, 1976.

[25] P.E. Gill and W. Murray.
Linearly constrained problems including linear and quadratic programming.
In D. Jacobs (editor), *The State of the Art in Numerical Analysis*, pages 313-363. Academic Press, London, 1977.

[26] P.E. Gill and W. Murray.
Numerically stable methods for quadratic programming.
Mathematical Programming 14:343-379, 1978.

[27] P.E. Gill , W. Murray, M.A. Saunders and M.H. Wright.
User's guide for SOL/NLSOL: a Fortran package for nonlinear programming.
Report SOL 83-12, Department of Operations Research, Stanford University, Stanford, California, USA, 1983.

[28] D. Goldfarb.
Extensions of Newton's method and simplex methods for solving quadratic programs.
In F.A. Lootsma (editor), *Numerical methods for nonlinear optimization*. Academic Press, London, 1972.

[29] A.A. Goldstein.
Convex programming in Hilbert space.
Bulletin of the American Mathematical Society 70:709-710, 1964.

[30] C.E. Lemke.
Bimatrix equilibrium points in mathematical programming.
Management Science 11:681-689, 1965.

[31] E.S. Levitin and B.T. Polyak.
Constrained minimization problems.
USSR Computational Mathematics and Mathematical Physics 6:1-50, 1966.

[32] G.P. McCormick.
Anti-zigzagging by bending.
Management Science 15:315-320, 1969.

[33] G.P. McCormick.
The variable reduction method for nonlinear programming.
Management Science 17:146-160, 1970.

[34] G.P. McCormick and R.A. Tapia.
The gradient projection method under mild differentiability conditions.
SIAM Journal of Control 10:93-98, 1972.

[35] R.H. Nickel and J.W. Tolle.
A sequential quadratic programming algorithm for solving large, sparse nonlinear programs .
Paper 416, Center for Naval Analyses, Alexandria, Virginia, USA, 1984.

[36] D. O'Leary.
Hybrid conjugate gradient algorithms.
Ph.D. Thesis, Computer Science Department, Stanford University, Stanford, California, 1976.

[37] D. O'Leary.
A generalized conjugate gradient algorithm for solving a class of quadratic programming problems.
Working Paper Stan-CS-77-638, Computer Science Department, Stanford University, Stanford, California, 1977.

[38] J.-S. Pang.
An equivalence between two algorithms for quadratic programming.
Mathematical Programming 20:152-165, 1981.

[39] E. Polak.
Methods in optimization: A unified approach.
Academic Press, London, 1971.

[40] B.T. Polyak.
The conjugate gradient method in extremal problems.
USSR Computational Mathematics and Mathematical Physics 9:94-112, 1969.

[41] M.J.D. Powell.
A fast algorithm for nonlinearly constrained optimization calculations.
In G.W. Watson (editor), *Numerical Analysis, Dundee, 1977, Lecture notes in mathematics 630,* pages 144-157. Springer Verlag, Berlin, 1978.

[42] K. Schittkowski.
The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. Part 2: An efficient implementation with linear least square subproblems.
Numerische Mathematik 38:115-127, 1981.

[43] U. Tulowitzki.
Successive Inexact Quadratic Programming for Nonlinear Optimization Problems.
Ph.D. Dissertation, Yale University, New Haven, Connecticut, USA, 1984.

[44] C. van de Panne and A. Whinston.
The symmetric formulation of the simplex method for quadratic programming.
Econometrica 37:507-527, 1969.

[45] Y. Vardi, L.A. Shepp and L. Kaufman.
A statistical model for positron emission tomography.
Working paper, Bell Laboratories, Murray Hill, New Jersey, USA, 1983.

[46] A. Whinston.
The bounded variable problem - an application of the dual method for quadratic programming.
Naval Research Logistics Quarterly 12:173-179, 1965.

[47] P. Wolfe.
The simplex method for quadratic programming.
Econometrica 27:382-398, 1959.