MINIMIZING PSEUDO-CONVEX FUNCTIONS ON CONVEX COMPACT SETS

by

J. E. Higgins and E. Polak

Memorandum No. UCB/ERL M88/22

March 1988

MINIMIZING PSEUDO-CONVEX FUNCTIONS ON CONVEX COMPACT SETS

by

J. E. Higgins and E. Polak

Memorandum No. UCB/ERL M88/22

March 1988

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

MINIMIZING PSEUDO-CONVEX FUNCTIONS ON CONVEX COMPACT SETS

by

J. E. Higgins and E. Polak

Memorandum No. UCB/ERL M88/22

March 1988

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

MINIMIZING PSEUDO-CONVEX FUNCTIONS ON CONVEX COMPACT SETS[1]

J. E. Higgins[2] and E. Polak[2]

ABSTRACT

An algorithm is presented which minimizes continuously differentiable pseudo-convex functions on convex compact sets which are characterized by their support functions. If the function can be minimized exactly on affine sets in a finite number of operations and the constraint set is a polytope, the algorithm has finitie convergence. Numerical results are reported which illustrate the performance of the algorithm when applied to a specific search direction problem. The algorithm differs from existing algorithms in that it has proven convergence when applied to any convex compact set, and not just polytopal sets.

KEY WORDS

Search direction problems, Minimization of pseudo-convex functions, Barycentric representation.

[1]This research was supported by the National Science Foundation grant ECS-8517362, the Air Force Office Scientific Research grant 86-0116, the Office of Naval Research contract N00014-86-K-0295, the California State MICRO program, and the Semiconductor Research Corp. Contract SRC-82-11-008.

[2]Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California, Berkeley, CA 94720, U.S.A.

1. INTRODUCTION

The selection of an algorithm for solving problems of the form:

$$\mathbf{P}: \quad \min_{x \in C} f(x) , \tag{1.1}$$

where C is a convex compact subset of a Hilbert space $\mathbf{H}$, and $f: \mathbf{H} \to \mathbb{R}$ is a continuously differentiable pseudo-convex function, depends on the description of the set C. When the set C is described by a finite or infinite number of differentiable inequalities, problem (1.1) can be solved by a large number of algorithms. When the set C is characterized only by its *support function* $\sigma(h) \triangleq \max_{x \in C} \langle h, x \rangle$, the number of possibilities reduces to a very small handful (see [1,2,3,4,5]), all of which can be traced back to the Frank-Wolfe algorithm [2].

Problems of the form (1.1), where the set C is characterized only by its support function, commonly arise as search direction subprocedures in semi-infinite optimization algorithms for engineering design (see, e.g. [6,7]). Quite frequently, in engineering applications, the set C has an infinite number of extreme points. For example (see [8]), the design of a stabilizing controller for a feedback system, which minimizes sensitivity of the closed loop system to disturbances can be cast in the form:

$$\mathbf{SVP}: \quad \min_{x \in \mathbb{R}^n} \max_{\omega \in [\omega', \omega'']} \sigma_1(H(x, j\omega)) , \tag{1.2}$$

where $\sigma_1(\cdot)$ denotes the maximum singular value of its argument, and the disturbance-to-output transfer function $H : \mathbb{R}^n \times \mathbb{C} \to \mathbb{C}^{k \times p}$ is affine in the design vector x and continuously differentiable in ω. By defining the function $\phi : \mathbb{R}^n \times \mathbb{C} \times \mathbb{C}^k \times \mathbb{C}^p \to \mathbb{R}$ as $\phi(x, j\omega, u, v) \triangleq \mathrm{Re}[u^* H(x, j\omega) v]$, we may transcribe **SVP** into the form:

$$\mathbf{SVP}: \quad \min_{x \in \mathbb{R}^n} \max_{\omega \in [\omega', \omega'']} \max_{\substack{\|u\| \le 1 \\ \|v\| \le 1}} \phi(x, j\omega, u, v) . \tag{1.3}$$

The search direction problem which results from applying Algorithm 5.2 (see Example 5.2, p. 65 of [7]) to the problem (1.3) requires the minimization of the convex function $\xi^0 + \tfrac{1}{2}\|\xi\|^2$ on the set

$$\overline{G}\psi(x) \triangleq \operatorname*{co}_{\substack{\|u\| \le 1 \\ \|v\| \le 1 \\ \omega \in [\omega', \omega'']}} \left\{ \begin{bmatrix} \psi(x) - \phi(x, j\omega, u, v) \\ \nabla_x \phi(x, j\omega, u, v) \end{bmatrix} \right\}, \tag{1.4}$$

where $\psi(x) \triangleq \max_{\omega \in [\omega', \omega'']} \sigma_1(H(x, j\omega))$. It is easy to see that, in general, the set $\overline{G}\psi(x)$ has an infinite number of extreme points.

We shall now briefly survey the existing algorithms for solving the problem **P** for the case where the set C is described only by its support function. In [2] Frank and Wolfe proposed an algorithm which solves **P** when $f(\cdot)$ is convex and continuously differentiable, and C is a nonempty bounded polyhedron described by a system of equations and inequalities. Frank and Wolfe showed that the iterates constructed by their algorithm converge to a solution at least as fast as α/k where k is the iteration number and $\alpha > 0$, while in [9], Canon and Cullum showed that under certain conditions, the iterates converge to a solution no faster than $\alpha/k^{1+\epsilon}$, where $\epsilon > 0$ is arbitrary and $\alpha > 0$. In [3], Gilbert makes the observation that the Frank-Wolfe algorithm can be used for solving problems of the form **P**, with the compact convex set C *defined only by its support function*, and evolves an algorithm for the minimization of a class of convex quadratic functions on a general convex, compact set. The main difference between the Gilbert algorithm and the Frank-Wolfe algorithm is that the Gilbert step size is less restrictive. If we set $\delta = 1$ in [3] the algorithms are identical.

Given an $x \in C$, the algorithms in [2,3] approximate the set C by the closed line segment $[x, t]$, where $t \in \arg\min_{\xi \in C} \langle \nabla f(x), \xi \rangle$ is a tangency point of the set C. In [1], Barr modifies the algorithm of [3] by maintaining a "larger" approximation to the set C at each step, and by prescribing rules for updating this larger approximation. At each iteration, the Barr algorithm requires that the quadratic cost function be minimized on the convex hull of $p+2$ points (selected from the set C) for some integer p. The Barr algorithm converges in a finite number of iterations on polytopes. However each iteration requires that a quadratic program be solved. In [5] von Hohenbalken presents an algorithm which minimizes pseudo-convex functions on polytopes. The algorithm in [5] is similar to that of [1] except that instead of maintaining $p+2$ points, the algorithm maintains an affinely independent collection of tangency points which contain the current iterate in the relative interior of their convex hull. The use of affinely independent sets is useful from a numerical standpoint. At each iteration, the von Hohenbalken algorithm attempts to construct a minimum of $f(\cdot)$ on certain affine sets and presents a technique for dealing with the case when $f(\cdot)$ has no minimum on these affine sets. The algorithm is shown to converge in a finite number

of iterations when the set C has a finite number of extreme points. When the set C has an infinite number of extreme points, the von Hohenbalken algorithm may fail to converge, as shown in the appendix. In addition [5], has the implicit assumption that for all affine sets A on which $f(\cdot)$ has a minimizer, $\arg\min_{x \in A} f(x)$ is a bounded set. While this assumption is true in many cases, it is not required for the algorithm presented in this paper. In [4], Wolfe presents a finitely converging algorithm similar to that of [5], for the special case of minimizing the square of the Euclidean norm ($f(x) \triangleq \|x\|^2$) on the convex hull of a finite number of points. In [10] computational results are presented which suggest that in practice Wolfe's method performs much better than the Frank-Wolfe algorithm.

In this paper we present an algorithm for the solution of the *general* case of problem **P**, i.e. for the case where the set C can have an infinite number of extreme points. Our algorithm is a modification of the von Hohenbalken algorithm [5], the main difference being the addition of a *guard step*. Our algorithm converges finitely on problems with polyhedral constraint sets C, but, in addition, it also converges to a solution on general problems of the form **P**. To illustrate the the use of our algorithm, we include implementation details and some numerical experience with the search direction calculation problems of [7].

2. PRELIMINARIES

We begin by reviewing some basic concepts associated with convexity and pseudo-convexity. Let *co* A denote the convex hull of the set A, and, for convenience, let $[x,y] \triangleq co\{x,y\}$.

Definition 2.1: A subset A of a real linear space is *affine* if and only if for all $x,y \in A$, for all $\lambda \in \mathbb{R}$, $\lambda x + (1-\lambda)y \in A$. ■

Definition 2.2: Let A be a subset of a real linear space. We say that *aff* A is the *affine hull* of A if it is the smallest affine set containing A. ■

It follows from Definition 2.2 that the affine hull of a set A is given by: $aff\, A = \{ \sum_{j \in J} \mu^j x_j \mid J \subset \mathbf{N} \text{ finite}, x_j \in A, \sum_{j \in J} \mu^j = 1 \}$.

Definition 2.3: A subset A of a real linear space is said to be *affinely independent* if and only if the

vectors $\{ (1, x) \}_{x \in A}$ are linearly independent. ■

Definition 2.4: Let A be a subset of a real topological linear space. We say that $ri\ A$ is the *relative interior of* A if it is the interior of A relative to *aff* A. ■

Definition 2.5: Let A be a subset of a real topological linear space. We say that $rb\ A$ is the *relative boundary of* A if it is the boundary of A relative to *aff* A. ■

We shall make frequent use of the following obvious results.

Proposition 2.1: Let A be a subset of a real linear space V and let $x \in V$, then $A \cup \{ x \}$ is affinely independent if and only if A is affinely independent and $x \notin aff\ A$. ■

Proposition 2.2: Let A be a subset of a real linear space. Then A is affinely independent if and only if for all $x \in aff\ A$ there exist unique $\{ \lambda_i, a_i \}_{i \in \underline{m}} \subset \mathbb{R} \times A$, $m \in \mathbb{N}$, such that $\sum_{i=1}^{m} \lambda_i = 1$, $\sum_{i=1}^{m} \lambda_i a_i = x$, and $\lambda_i \neq 0$ for all $i \in \underline{m}$. ■

Given an affinely independent set A and a point $x \in A$, the unique multipliers $\{ \lambda_i \}_{i \in \underline{m}}$ whose existence is guaranteed by Proposition 2.2 are known as the *barycentric coordinates* of x with respect to A.

Definition 2.6: A set S is said to be a *simplex* if and only if it is the convex hull of an affinely independent set A. The set A is referred to as the *affine basis* of S. ■

Definition 2.7: Let S be a simplex, A its basis, and let $x \in S$. We say that a point $a \in A$ *carries* x if and only if the barycentric coordinate of x corresponding to a is strictly positive. ■

Following Rockafellar [11] we say that a convex set C is a *polyhedron* if it is represented as the intersection of a finite number of half-spaces, and we call it a *polytope* if it is the convex hull of a finite number of points. The following results summarize some relevant properties of convex sets in $\mathbb{R}^n$.

Theorem 2.3: ([11, Theorem 17.1]) Let $A \subset \mathbb{R}^n$. Then $x \in co\ A$ if and only if x can be represented as a convex combination of at most $n + 1$ points of A. ■

Theorem 2.4: ([11, Corollary 19.1.1]) A compact polyhedral (or polytopal) subset of $\mathbb{R}^n$ has a finite number of extreme points.

■

As a consequence of Theorem 2.4, the number of simplices which may be formed from the extreme points of a compact polyhedron is finite. Similarly for a polytope, the number of simplices which may be formed from its underlying point set is also finite. This property is utilized in the algorithm presented in Section 5 to guarantee finite termination when the set C in problem **P** is a polytope or a compact polyhedron.

The following is a useful generalization of the concept of a convex function. The definition is essentially that of [12, p. 140] but is slightly more restrictive.

Definition 2.8: Let $f(\cdot)$ be a real valued function on a Hilbert space **H**. The function $f: \mathbf{H} \rightarrow \mathbb{R}$ is said to be *pseudo-convex* at $x \in \mathbf{H}$ if and only if (i) $f(\cdot)$ is (Frechet) differentiable at x and (ii) for all $y \in \mathbf{H}$ such that $\langle \nabla f(x), y - x \rangle \geq 0$ we have $f(y) \geq f(x)$. The function $f(\cdot)$ is said to be pseudo-convex on **H** if and only if it is pseudo-convex for all $x \in \mathbf{H}$. ■

Pseudo-convex functions have properties similar to those of convex functions. These are summarized by the following propositions. The proofs are straightforward and are omitted.

Proposition 2.6: Let $f(\cdot)$ be a pseudo-convex function on a Hilbert space **H**. Then for all $\alpha \in \mathbb{R}$, the level sets $L_\alpha \triangleq \{ x \mid f(x) \leq \alpha \}$ are convex. ■

Proposition 2.7: Suppose $f(\cdot)$ is a pseudo-convex function on a Hilbert space **H**, and let $C \subset \mathbf{H}$ be convex. Then $\hat{x}$ minimizes $f(\cdot)$ on C if and only if $\langle \nabla f(\hat{x}), \xi - \hat{x} \rangle \geq 0$, for all $\xi \in C$. ■

Proposition 2.8: Suppose $f(\cdot)$ is a pseudo-convex function on a Hilbert space **H**, and let $C \subset \mathbf{H}$ be convex. Suppose $\hat{x} \in ri\ C$ minimizes $f(\cdot)$ on C. Then $\hat{x}$ minimizes $f(\cdot)$ on *aff* C.

Proof: Let $z \in aff\ C$. Since $\hat{x} \in ri\ C$, there exists a point $y \in [\hat{x}, z] \cap C$, and a $\lambda \in (0,1)$ such that $y = \lambda\hat{x} + (1-\lambda)z$. By Proposition 2.7 we have

$$\begin{aligned} \langle \nabla f(\hat{x}), y - \hat{x} \rangle &= \langle \nabla f(\hat{x}), \lambda\hat{x} + (1-\lambda)z - \hat{x} \rangle \\ &= (1-\lambda)\langle \nabla f(\hat{x}), z - \hat{x} \rangle \end{aligned} \tag{2.1}$$

≥ 0,

from which we conclude that $\langle \nabla f(\hat{x}), z-\hat{x} \rangle \geq 0$. Applying Proposition 2.7 once more, we conclude that $\hat{x}$ minimizes $f(\cdot)$ on *aff* C. ■

A useful consequence of Proposition 2.6 is that if $f(\cdot)$ is pseudo-convex and $f(x_2) \leq f(x_1)$ then $f(\xi) \leq f(x_1)$ for all $\xi \in [x_1, x_2]$. We use the following notion of a closed map to establish convergence of the algorithm in Section 3.

Definition 2.9: ([13, p. 88]) Let X, Y be topological spaces and let $A(\cdot)$ be a set valued mapping $A : X \rightarrow 2^Y$. Then $A(\cdot)$ is said to be *closed* at $x \in X$ if and only if whenever $x_i \rightarrow \hat{x}$ and $y_i \rightarrow \hat{y}$ with $x_i \in X$ and $y_i \in A(x_i)$, then $\hat{y} \in A(\hat{x})$. The map $A(\cdot)$ is said to be closed if it is closed at each $x \in X$. ■

3. ALGORITHM MODEL

An algorithm model and convergence proof are presented. The algorithm model is an extension of the Frank-Wolfe algorithm in [2]. Let C be a compact convex subset of a real Hilbert space **H**, and let $f(\cdot)$ be a continuously (Frechet) differentiable pseudo-convex function defined on **H**. We define the *contact point function* $T: C \rightarrow 2^C$, the optimality function $\theta: C \rightarrow (-\infty, 0]$ and the algorithm map $A: C \rightarrow 2^C$ as follows:

$$T(x) \triangleq \arg \min_{\xi \in C} \langle \nabla f(x), \xi - x \rangle, \tag{3.1}$$

$$\theta(x) \triangleq \min_{\xi \in C} \langle \nabla f(x), \xi - x \rangle, \tag{3.2}$$

$$A(x) \triangleq \bigcup_{t \in T(x)} \{ \xi \in C \mid f(\xi) \leq \min_{\zeta \in [x,t]} f(\zeta) \}. \tag{3.3}$$

An illustration of the sets $T(x)$ and $A(x)$ is given in Figure 3.1. Informally, $A(x)$ consists of points in C which have cost less than or equal to the cost of the points produced by the Frank-Wolfe algorithm. It is easy to see that $\theta(\cdot)$ (and hence $T(\cdot)$) is related to the support function of C by $\theta(x) = -\sigma_C(-\nabla f(x)) - \langle \nabla f(x), x \rangle$, where $\sigma_C(\cdot)$ is the support function of C. When the set C is a polytope, an element of the set $T(x)$ may be obtained by minimizing a linear function over a finite collection of points. The following result, which characterizes solutions to **P** in terms of $\theta(\cdot)$, follows immediately

from Proposition 2.7.

Proposition 3.1: Let C be a compact convex subset of a real Hilbert space, and let $f(\cdot)$ be a pseudo-convex function defined on some open superset of C. Then $\theta(\hat{x}) = 0$ if and only if $\hat{x}$ minimizes $f(\cdot)$ on C. ■

The algorithm model for solving problem **P** may now be stated.

Algorithm 3.1

Data: $x_0 \in C$.

Step 0: Set $i = 0$.

Step 1: If $\theta(x_i) = 0$ stop.

Step 2: Select $x_{i+1} \in A(x_i)$.

Step 3: Replace i by $i+1$ and go to Step 1. ■

Step 2 implicitly contains the *guard step*, mentioned in the introduction, which ensures that for some $t_i \in T(x_i)$, $f(x_{i+1}) \leq \min_{\zeta \in [x_i, t_i]} f(\zeta)$. This guard step ensures that the algorithm converges on sets with an infinite number of extreme points. To prove global convergence of the algorithm we first show that the algorithm map $A(\cdot)$ is closed.

Proposition 3.2: Let C be a compact convex subset of a real Hilbert space **H**, and let $f(\cdot)$ be a continuously (Frechet) differentiable pseudo-convex function defined on **H**. Then the function $A(\cdot)$ defined by (3.3) is closed.

Proof: Suppose $x_i \to \hat{x}$ and $y_i \to \hat{y}$ with $x_i \in C$ and $y_i \in A(x_i)$. We must show that $\hat{y} \in A(\hat{x})$. It follows from the definition of $A(\cdot)$ that since $y_i \in A(x_i)$, there exists a $t_i \in T(x_i)$ such that $f(y_i) \leq \min_{\zeta \in [x_i, t_i]} f(\zeta)$. By compactness of C the sequence $\{ t_i \}_{i=0}^{\infty}$ will have an accumulation point $\hat{t} \in C$. Since

$$\langle \nabla f(x_i), t_i \rangle \leq \langle \nabla f(x_i), \xi \rangle \quad \text{for any } \xi \in C, \tag{3.4}$$

we may conclude by continuity of $\nabla f(\cdot)$ and of the inner product that

$$\langle \nabla f(\hat{x}), \hat{t} \rangle \leq \langle \nabla f(\hat{x}), \xi \rangle \quad \text{for all } \xi \in C . \tag{3.5}$$

Hence $\hat{t} \in T(\hat{x})$. In addition, for any $\bar{\lambda} \in [0,1]$, we have

$$f(y_i) \leq f(\bar{\lambda} x_i + (1-\bar{\lambda}) t_i) . \tag{3.6}$$

We may conclude by continuity of $f(\cdot)$ that

$$f(\hat{y}) \leq f(\bar{\lambda}\hat{x} + (1-\bar{\lambda})\hat{t}) , \tag{3.7}$$

and so

$$f(\hat{y}) \leq \min_{\zeta \in [\hat{x},\hat{t}]} f(\zeta) . \tag{3.8}$$

Therefore $\hat{y} \in A(\hat{x})$ as required. ■

The following Proposition is similar to Zangwill's Convergence Theorem A [13, p. 91].

Proposition 3.3: Let C be a compact convex subset of a real Hilbert space **H**, and let $f(\cdot)$ be a continuously (Frechet) differentiable pseudo-convex function defined on **H**. Then any accumulation point $\hat{x}$ of an infinite sequence $\{ x_i \}_{i=0}^{\infty}$, generated by Algorithm 3.1, satisfies $\theta(\hat{x}) = 0$.

Proof: Suppose to the contrary that $x_i \overset{K}{\rightarrow} \hat{x}$ where K is some infinite subset of $\mathbb{N}$ and that $\theta(\hat{x}) < 0$. Since $x_{i+1} \in A(x_i) \subset C$, it follows by compactness of C that there exists an infinite subset L of K such that $x_{i+1} \overset{L}{\rightarrow} \bar{x}$, for some $\bar{x} \in C$. Clearly since $A(\cdot)$ is a closed map, $\bar{x} \in A(\hat{x})$. Since $\theta(\hat{x}) < 0$, it follows that $f(\bar{x}) < f(\hat{x})$. By construction $f(x_{i+1}) \leq f(x_i)$, which combined with the fact that $x_i \overset{K}{\rightarrow} \hat{x}$ implies that $f(x_i) \rightarrow f(\hat{x})$. This, however, contradicts $f(\bar{x}) < f(\hat{x})$, and so we may conclude that $\theta(\hat{x}) = 0$. ■

The following theorem shows that any sequence generated by Algorithm 3.1 converges to the set of solutions of **P**. Let Θ denote the set of minimizers of **P**, and define $d_{\Theta}(x) \triangleq \min_{\xi \in \Theta} \|x-\xi\|$.

Theorem 3.4: Let C be a compact convex subset of a real Hilbert space **H**, and let $f(\cdot)$ be a continuously (Frechet) differentiable pseudo-convex function defined on **H**. Let $\{ x_i \}_{i=0}^{\infty}$ be an infinite sequence generated by Algorithm 3.1. Then

$$\lim_{i \to \infty} d_{\Theta}(x_i) = 0. \tag{3.9}$$

Proof: To obtain a contradiction suppose that $d_{\Theta}(x_i)$ does not converge to 0. Then there exist an $\varepsilon > 0$ and an infinite subset K of $\mathbb{N}$ such that $d_{\Theta}(x_i) \geq \varepsilon$ for all $i \in K$. Let

$$\Delta \triangleq \{ \xi \in C \mid d_{\Theta}(\xi) \geq \varepsilon \}. \tag{3.10}$$

It follows by continuity of $d_{\Theta}(\cdot)$ that Δ is closed and hence compact. Since $\{ x_i \}_{i \in K} \subset \Delta$, there exists an accumulation point $\hat{x} \in \Delta$. However Proposition 3.3 implies that $\theta(\hat{x}) = 0$, and Proposition 3.1 implies that $\hat{x} \in \Theta$ which is a contradiction. ■

Under fairly mild conditions on $f(\cdot)$ (for example if $f(\cdot)$ has strictly convex level sets), the set Θ contains a single point, and the sequence produced by Algorithm 3.1 will converge in the usual sense. In addition, in some special cases this result may be strengthened considerably to give convergence to the set Θ in a *finite* number of iterations.

4. ACTUAL ALGORITHM

An algorithm which falls within the framework of Algorithm 3.1 is described, along with certain implementation details. Let C be a compact convex subset of a real Hilbert space **H**, and let $f(\cdot)$ be a continuously (Frechet) differentiable pseudo-convex function defined on **H**. The functions $T(\cdot)$ and $\theta(\cdot)$ are defined by (3.1) and (3.2). Let $E \subset C$ be a set containing the extreme points of C. The computation of Step 3 will be described subsequently.

Algorithm 4.1 (Minimize $f(\cdot)$ on C).

Data: $x_0 \in E$.

Step 0: Set $i = 0$. Let $B_0 = \{ x_0 \}$.

Step 1: If $\theta(x_i) = 0$ stop.

Step 2: Select $t_i \in T(x_i) \cap E$.

Step 3: Compute x_{i+1}, $B_{i+1} \subset B_i \cup \{ t_i \}$ satisfying the following conditions:

(i) $f(x_{i+1}) \leq \min_{\zeta \in [x_i, t_i]} f(\zeta)$ (*Guard step*).

(ii) B_{i+1} is affinely independent.

(iii) $x_{i+1} \in ri\ co\ B_{i+1}$.

(iv) x_{i+1} minimizes $f(\cdot)$ on $co\ B_{i+1}$.

Step 4: Replace i by i+1 and go to Step 1. ■

It should be evident that Algorithm 4.1 fits into the class of algorithms described by Algorithm 3.1, and that Step 3 is well defined. For example, one may choose $x_{i+1} \in \arg\min_{\zeta \in B} f(\zeta)$, where $B \triangleq B_i \cup \{ t_i \}$, and let B_{i+1} be the set of points carrying x_{i+1}. By combining the fact that $\theta(x_i) < 0$ and Proposition 2.1, we see that B (and hence any subset) is affinely independent. The pair (x_{i+1}, B_{i+1}) clearly satisfies condition (i)-(iv) of Step 3. The requirement that the B_i be affinely independent (Step 3 (ii)) allows condition (iii) of Step 3 to be checked easily, and is usually useful from a numerical standpoint. The convergence result of Theorem 3.4 can be strengthened as follows.

Theorem 4.1: Let C be a compact convex polytopal subset of a real Hilbert space **H**, and let $f(\cdot)$ be a continuously (Frechet) differentiable pseudo-convex function defined on **H**. Suppose that the set $E \subset C$, used in the Data of Algorithm 4.1, is a finite set containing the extreme points of C. Then Algorithm 4.1 solves **P** in a finite number of iterations.

Proof: Obviously, if the algorithm terminates, it is at a solution. Suppose that for some i we have $\theta(x_i) < 0$. Since x_i minimizes $f(\cdot)$ on $co\ B_i$, and $f(x_{i+1}) < f(x_i)$ it follows that $B_{i+1} \neq B_j$ for all $j \leq i$. By construction $B_i \subset E$, and by assumption E is finite. Hence only a finite number of distinct B_i exist, and so we conclude that the algorithm terminates in a finite number of iterations. ■

To continue we need the following definition:

Definition 4.1: Let **H** be a real Hilbert space and let $f : \mathbf{H} \rightarrow \mathbb{R}$ be a pseudo-convex function. Let $\mathbf{A} \triangleq \{ a \subset \mathbf{H} \mid a \text{ is affine} \}$, and define $i_f : \mathbf{A} \rightarrow \{ 0, 1 \}$ as:

$$i_f(a) \triangleq \begin{cases} 0 & \text{if } f(\cdot) \text{ has no minimizer on } a \\ 1 & \text{if } f(\cdot) \text{ has a minimizer on } a \end{cases} \qquad (4.1)$$

Let $\mathbf{M} \triangleq i_f^{-1}(\{ 1 \})$, and for each $a \in \mathbf{M}$ let $m_f(a)$ be an arbitrary element of $\arg\min_{x \in a} f(x)$. ■

The method of computing Step 3, described below, requires that the functions $i_f(\cdot)$ and $m_f(\cdot)$ be explicitly provided. While it is clear that these functions are well defined for any function $f(\cdot)$, in practice this restricts the class of functions to which the algorithm can be applied.

We now consider a subprocedure which implements Step 3 of Algorithm 4.1. The idea underlying the subprocedure is as follows: Suppose the subprocedure is passed a triple (x, B, t), where B is an affinely independent set, $x \in ri\ co\ B$ minimizes $f(\cdot)$ on $co\ B$, and t is a point such that $\langle \nabla f(x), t-x \rangle < 0$. We wish to compute a pair $(\hat{x}, \hat{B})$ satisfying Conditions (i)-(iv) of Step 3. First, we obtain a point $\xi \in \arg\min_{\zeta \in [x,t]} f(\zeta)$. If $\xi = t$ we may return the pair $(\xi, \{ \xi \})$ which clearly satisfy Conditions (i)-(iv) of Step 3 in Algorithm 4.1. If not, we let $x_0 \triangleq \xi$ and $B_0 \triangleq B \cup \{ t \}$. It is straightforward to verify that B_0 is affinely independent (Proposition 2.1), and hence any subset of B_0 will also be affinely independent. If a minimizer $\hat{x}$ of $f(\cdot)$ on $aff\ B_0$ exists and is in $co\ B_0$ then we return $(\hat{x}, \hat{B})$ where $\hat{B} \subset B_0$ is the set of points which carry the minimizer (see Figure 4.1). If no minimizer exists, or the minimizer lies in $aff\ B_0 \cap (co B_0)^C$, a point $m_0 \in aff\ B_0 \cap (co B_0)^C$ is obtained such that $f(m_0) \leq f(x_0)$. Note that by Proposition 2.6, $f(\zeta) \leq f(x_0)$ for all $\zeta \in [m_0, x_0]$. We then obtain the point x_1 (see Figure 4.2) defined by $\{ x_1 \} \triangleq [x_0, m_0] \cap rb\ co\ B_0$ (it should be clear that since $x_0 \in ri\ co\ B_0$ and $m_0 \in aff\ B_0 \cap (co B_0)^C$ that exactly one point of intersection exists). The set B_1 is formed by dropping all points in B_0 which do not carry x_1. Since $f(x_1) \leq f(x_0)$, the pair (x_1, B_1) satisfies Conditions (i)-(iii) of Step 3 in Algorithm 4.1. In order to satisfy Condition (iv), the above procedure is repeated, starting with (x_1, B_1), until a suitable point is obtained. The actual algorithm which implements Step 3 of Algorithm 4.1 is as follows, Step 5 of Subprocedure 4.1 will be elaborated later. Let $f(\cdot)$ be a continuously (Frechet) differentiable pseudo-convex function defined on a real Hilbert space $\mathbf{H}$. Let $i_f(\cdot)$ and $m_f(\cdot)$ be defined as above.

Subprocedure 4.1 (Implementation of Step 3 of Algorithm 4.1).

Data: B affinely independent set, $x \in ri\ co\ B$ such that x minimizes $f(\cdot)$ on $co\ B$, $t \in \mathbf{H}$ such that $\langle \nabla f(x), t-x \rangle < 0$.

Step 0: Let $\xi \in \arg \min_{\zeta \in [x,t]} f(\zeta)$ (*Guard step*).

Step 1: If $\xi = t$ then set $\hat{x} \triangleq \xi, \hat{B} \triangleq \{ \xi \}$ and return the pair $(\hat{x}, \hat{B})$.

Step 2: Let $i = 0, x_0 \triangleq \xi, B_0 \triangleq B \cup \{ t \}$.

Step 3: If $i_f(\mathit{aff}\, B_i) = 1$ then let $m_i \triangleq m_f(\mathit{aff}\, B_i)$. Else go to Step 5.

Step 4: If $m_i \in \mathit{co}\, B_i$, then let $\hat{B} \subset B_i$ be the set of points carrying m_i, let $\hat{x} \triangleq m_i$ and return the pair $(\hat{x}, \hat{B})$. Else go to Step 6.

Step 5: (No minimizer exists) Obtain a $m_i \in \mathit{aff}\, B_i \cap (\mathit{co}\, B_i)^C$ such that $f(m_i) \leq f(x_i)$.

Step 6: Let $x_{i+1} \in [x_i, m_i] \cap \mathit{rb}\ \mathit{co}\, B_i$, and let $B_{i+1} \subset B_i$ be the set of points carrying x_{i+1}.

Step 7: Replace i by i+1 and go to Step 3. ■

The following result shows that Subprocedure 4.1 returns a pair satisfying Conditions (i)-(iv) of Step 3 in Algorithm 4.1.

Proposition 4.2: Let **H** be a real Hilbert space, and let $f: \mathbf{H} \to \mathbb{R}$ be a continuously (Frechet) differentiable pseudo-convex function. Suppose that (i) B is a finite, affinely independent subset of **H**, (ii) $x \in \mathit{ri}\ \mathit{co}\, B$ such that x minimizes $f(\cdot)$ on $\mathit{co}\, B$ and (iii) $t \in \mathbf{H}$ is such that $\langle \nabla f(x), t-x \rangle < 0$. Then Subprocedure 4.1 will terminate in a finite number of iterations returning a pair $(\hat{x}, \hat{B})$ satisfying the following conditions:

(i) $f(\hat{x}) \leq \min_{\zeta \in [x,t]} f(\zeta)$.

(ii) $\hat{B}$ is affinely independent.

(iii) $\hat{x} \in \mathit{ri}\ \mathit{co}\, \hat{B}$.

(iv) $\hat{x}$ minimizes $f(\cdot)$ on $\mathit{co}\, \hat{B}$.

(v) $\hat{B} \subset B \cup \{ t \}$.

Proof: Since B is finite and Step 6 always removes at least one point, it is clear that the algorithm terminates in a finite number of steps. In addition, the Algorithm returns only from Steps 1 and 4, and consequently Conditions (iii) and (iv) are satisfied. As remarked earlier, the set $B \cup \{ t \}$ is affinely

independent, and hence so is $\hat{B}$. By construction $f(x_{i+1}) \leq f(x_i)$, and $f(x_0) = \min_{\zeta \in [x,t]} f(\zeta)$, and so Condition (i) is satisfied. Condition (v) is trivially true. ■

Some remarks regarding the implementation of Subprocedure 4.1 are in order. From a computational standpoint, the Hilbert space in question is invariably $\mathbb{R}^n$, and the affinely independent sets B_i can have cardinality at most $n+1$ (Theorem 2.3). Thus the sets B_i may be represented by a matrix $\mathbf{B}_i$ whose columns are the elements of B_i, and where $\mathbf{B}_i$ has at most $n+1$ columns. Any point $x \in \mathit{aff}\, B_i$ can be represented by a multiplier μ of suitable dimension such that $x = \mathbf{B}_i\mu$ and $\langle e, \mu \rangle = 1$, where $\langle \cdot, \cdot \rangle$ is the usual inner product on $\mathbb{R}^n$ and e is a vector of ones.

One method of implementing Step 5 is as follows: Suppose that $B_i = \{ b_j \}_{j=1}^{k}$, and define the matrix $L_i \in \mathbb{R}^{n \times (k-1)}$ by $L_i \triangleq [b_1 - b_k \cdots b_{k-1} - b_k]$. It is trivial to verify that $\mathit{aff}\, B_i = \mathbf{R}(L_i) + \{ b_k \}$ where $\mathbf{R}(\cdot)$ denotes the range space of a linear operator, and set addition is defined in the usual manner. Define $\phi_i : \mathbb{R}^{k-1} \rightarrow \mathbb{R}$ by $\phi_i(y) \triangleq f(L_i y + b_k)$. It is possible to show that by applying a method of steepest descent (see, for example, Algorithm 37 in Section 2.1 of [14]) to $\phi_i(\cdot)$, a point in $\mathit{aff}\, B_i \cap (\mathit{co}\, B_i)^C$ with cost lower than $f(x_i)$ may be obtained in a finite number of iterations (under the assumption that $i_f(\mathit{aff}\, B_i) = 0$). For many functions (see Section 5 for an example) it is possible to compute such a point directly without iteration. Alternatively, the procedure of [5] may be modified (by the addition of a suitable guard step) to compute Step 5. Using this procedure requires the additional assumption that on all affine sets A such that $i_f(A) = 1$ (i.e. $A \in \mathbf{M}$), $\arg\min_{x \in A} f(x)$ is a bounded set.

The computation in Step 6 is straightforward. Suppose that $x_i = \mathbf{B}_i\mu_i$ and $m_i = \mathbf{B}_i\nu_i$, where $\langle e, \mu_i \rangle = \langle e, \nu_i \rangle = 1$, $\mu_i > 0$ (componentwise) and $\nu_i^j < 0$ for some j (since $m_i \notin \mathit{co}\, B_i$). Then a simple calculation shows that

$$x_{i+1} = \mathbf{B}_i(\mu_i + \lambda_i(\nu_i - \mu_i)) \tag{4.2}$$

where

$$\lambda_i \triangleq \min\{ \frac{\mu_i^j}{\mu_i^j - \nu_i^j} \mid \mu_i^j - \nu_i^j > 0 \}. \tag{4.3}$$

The set B_{i+1} is computed by dropping all points in B_i for which the associated multiplier $\mu_i^j + \lambda_i(\nu_i^j - \mu_i^j)$ is zero (at least one such point exists).

In practice, the stopping rule (Step 1) in Algorithm 4.1 is replaced by a rule which stops when the point x_i is "close enough" to a desired point. For example, if the function $f(\cdot)$ is convex and continuously differentiable (hence trivially pseudo-convex), a suitable stopping criteria is to stop when $|\theta(x_i)| \leq \varepsilon$ where $\varepsilon > 0$ is suitably small. In this case, when the algorithm terminates the following condition holds:

$$\min_{x \in C} f(x) \geq f(x_i) - \varepsilon . \tag{4.4}$$

In many instances more appropriate stopping rules can be used for specific $f(\cdot)$.

5. APPLICATION

To illustrate Algorithm 4.1 we consider the application of Algorithm 4.1 to a specific cost function. Relevant properties of the cost function $f(\cdot)$ are developed, an alternative stopping criterion is presented, and some numerical implementation details are discussed. This cost function arises when a modified version of Algorithm 5.2 of [7] is applied to the problem:

$$\textbf{SVP:} \qquad \min_{x \in \mathbb{R}^n} \sigma_1(A(x)) , \tag{5.1}$$

where $\sigma_1(\cdot)$ denotes the maximum singular value of its argument, and $A : \mathbb{R}^n \rightarrow \mathbb{R}^{k \times p}$ is a continuously differentiable function. By defining the function $\phi : \mathbb{R}^n \times \mathbb{R}^k \times \mathbb{R}^p \rightarrow \mathbb{R}$ as $\phi(x,u,v) \triangleq u^T A(x) v$, we may transcribe SVP as:

$$\textbf{SVP:} \qquad \min_{x \in \mathbb{R}^n} \max_{\substack{\|u\| \leq 1 \\ \|v\| \leq 1}} \phi(x,u,v). \tag{5.2}$$

Algorithm 5.2 is modified by replacing the optimality function and the augmented search direction by those given in Proposition 5.5 of [7]. A suitable a.c.d.f. map (see [7] for details) for problem SVP is given by

$$\overline{G}\psi(x) \triangleq \underset{\substack{\|u\| \leq 1 \\ \|v\| \leq 1}}{co} \left\{ \begin{bmatrix} \sigma_1(A(x)) - \phi(x,u,v) \\ \nabla_x \phi(x,u,v) \end{bmatrix} \right\}. \tag{5.3}$$

The search direction computation of the modified algorithm requires the solution of the problem

$$\min_{\xi \in \bar{C}} \xi^0 + \tfrac{1}{2}\|\xi\|^2 , \tag{5.4}$$

where $\bar{\xi} \triangleq (\xi^0, \xi^T)^T$, $\xi^0 \in \mathbb{R}$, $\xi \in \mathbb{R}^n$, and $\bar{C} \triangleq \bar{G}\psi(x)$. Given a direction $\bar{h} \triangleq (h^0, h^T)^T$, a tangency point to the set $\bar{C}$ may be computed by solving the problem

$$\begin{aligned} \min_{\xi \in \bar{C}} \langle \bar{h}, \bar{\xi} \rangle &= \min_{\substack{\|u\| \le 1 \\ \|v\| \le 1}} h^0(\sigma_1(A(x)) - \phi(x,u,v)) + \langle h, \nabla_x \phi(x,u,v) \rangle , \\ &= \min_{\substack{\|u\| \le 1 \\ \|v\| \le 1}} h^0 \sigma_1(A(x)) + u^T \Big(\sum_{i=1}^{n} h^i \frac{\partial A(x)}{\partial x^i} - h^0 A(x)\Big) v , \\ &= h^0 \sigma_1(A(x)) - \max_{\substack{\|u\| \le 1 \\ \|v\| \le 1}} u^T \Big(h^0 A(x) - \sum_{i=1}^{n} h^i \frac{\partial A(x)}{\partial x^i}\Big) v . \end{aligned} \tag{5.5}$$

Hence a tangency point to $\bar{C}$ may be computed by evaluating the maximum singular value and corresponding right and left singular vectors of the matrix

$$h^0 A(x) - \sum_{i=1}^{n} h^i \frac{\partial A(x)}{\partial x^i} . \tag{5.6}$$

From a numerical standpoint it is sometimes useful to scale the search direction calculation by replacing the $\frac{1}{2}\|\xi\|^2$ term in (5.4) by $\frac{1}{2}\langle \xi, Q\xi \rangle$ where Q is a symmetric positive definite scaling matrix. The resulting search direction problem becomes:

$$\mathbf{SDP}_1: \qquad \min_{\xi \in \bar{C}} f_{pp}(\bar{\xi}) , \tag{5.7}$$

where

$$f_{pp}(\bar{\xi}) \triangleq \xi^0 + \tfrac{1}{2}\langle \xi, Q\xi \rangle , \tag{5.8}$$

Q is a positive definite symmetric matrix, and $\bar{C}$ is a convex compact subset of $[0,\infty)\times\mathbb{R}^n$. The function $f_{pp}(\cdot)$ has some useful properties, which are summarized in the following proposition. Define $e_0 \triangleq (1,0,\ldots,0)^T$, and let e_i denote the vector with 1 in the $i+1$ position and zero elsewhere.

Proposition 5.1: Let $f_{pp}(\cdot)$ be as in (5.8), and let $\bar{C}$ be a convex compact subset of $[0,\infty)\times\mathbb{R}^n$. Let $\bar{B} \subset \mathbb{R}^{n+1}$ be an affinely independent set and define $B \triangleq \{ \xi \mid \bar{\xi} \in \bar{B} \}$. Define $\bar{\mathbf{B}}$ to be a matrix whose columns are the elements of $\bar{B}$. Partition $\bar{\mathbf{B}}$ as follows (where $\mathbf{B}$ has row dimension n):

$$\overline{\mathbf{B}} = \begin{bmatrix} (\mathbf{b}^0)^T \\ \mathbf{B} \end{bmatrix}. \tag{5.9}$$

Then the following are true:

(i) $f_{pp}(\cdot)$ is convex, smooth (hence pseudo-convex) and is non-negative on $\overline{C}$.

(ii) For all $\lambda \in \mathbb{R}$, $f_{pp}(\overline{\xi}+\lambda e_0) - f_{pp}(\overline{\xi}) = \lambda$.

(iii) The set $\Theta \triangleq \arg\min_{\overline{\xi} \in \overline{C}} f_{pp}(\overline{\xi})$ is a singleton.

(iv) $f_{pp}(\cdot)$ has a minimizer on $aff\ \overline{B}$ (equivalently $i_{f_{pp}}(aff\ \overline{B}) = 1$) $\Leftrightarrow$ B is affinely independent.

(v) B is affinely independent $\Leftrightarrow$ the matrix A defined by:

$$A \triangleq \begin{bmatrix} 0 & e^T \\ e & \mathbf{B}^T Q \mathbf{B} \end{bmatrix} \tag{5.10}$$

is invertible.

(vi) If $i_{f_{pp}}(aff\ \overline{B}) = 1$, then $m_{f_{pp}}(aff\ \overline{B}) = \overline{\mathbf{B}}\hat{w}$, where $\hat{w}$ solves the system

$$A \begin{bmatrix} \hat{\lambda} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} 1 \\ -\mathbf{b}^0 \end{bmatrix}. \tag{5.11}$$

Proof: Notice that $aff\ \overline{B} = \{\ \overline{\mathbf{B}} w \mid \langle e, w \rangle = 1\ \}$, where e is a vector of ones and dimension equal to the cardinality of $\overline{B}$. Properties (i) and (ii) are obvious. To prove (iii) note that

$$f_{pp}(\lambda\overline{\xi}_1+(1-\lambda)\overline{\xi}_2)-(\lambda f_{pp}(\overline{\xi}_1)+(1-\lambda)f_{pp}(\overline{\xi}_2)) = \frac{\lambda(1-\lambda)}{2}\langle \xi_1-\xi_2, Q(\xi_1-\xi_2)\rangle. \tag{5.12}$$

Suppose that $\overline{\xi}_1, \overline{\xi}_2 \in \Theta$. It follows from (5.12) that $\xi_1 = \xi_2$, and since $f_{pp}(\overline{\xi}_1) = f_{pp}(\overline{\xi}_2)$ we conclude that $\overline{\xi}_1 = \overline{\xi}_2$ which shows that Θ contains a single point.

(iv) ($\Rightarrow$) To obtain a contradiction suppose $f_{pp}(\cdot)$ has a minimizer on $aff\ \overline{B}$ and that B is affinely dependent. Hence by suitably numbering the elements of B, we may obtain a multiplier λ such that

$$\xi_1 = \sum_{i \neq 1} \lambda_i \xi_i, \quad \sum_{i \neq 1} \lambda_i = 1, \quad \xi_i \in B. \tag{5.13}$$

Since $\overline{B}$ is affinely independent, we have $\xi_1^0 \neq \sum_{i \neq 1} \lambda_i \xi_i^0$. Since $\overline{\xi}_1, \sum_{i \neq 1} \lambda_i \overline{\xi}_i \in aff\ \overline{B}$ it follows that $\mathbb{R} \times \{\ \xi_1\ \} \subset aff\ \overline{B}$. Since $f_{pp}(\cdot)$ is unbounded from below on $\mathbb{R} \times \{\ \xi_1\ \}$ we obtain a contradiction.

($\Leftarrow$) Suppose B is affinely independent. It is straightforward to verify that the matrix A (defined by (5.10)) is non-singular, and so the following system has a solution:

$$A\begin{bmatrix}\hat{\lambda}\\ \hat{w}\end{bmatrix} = \begin{bmatrix}1\\ -b^0\end{bmatrix}. \tag{5.14}$$

The system (5.14) is a first order necessary and sufficient optimality condition for the convex problem

$$\min\{\, f_{pp}(\overline{B}w) \mid \langle w, e\rangle = 1 \,\}. \tag{5.15}$$

Hence $\overline{B}\hat{w}$ minimizes $f_{pp}(\cdot)$ on $aff\ \overline{B}$, with $\hat{\lambda}$ being the Lagrange multiplier associated with the equality constraint.

(v) This follows from the definition of affine independence and the fact that Q is positive definite.

(vi) Since the point $\hat{w}$ defined by (5.14) is the unique solution of (5.15), it is clear that $m_{f_{pp}}(aff\ \overline{B}) = \overline{B}\hat{w}$. ■

The following is an alternative stopping criterion which is appropriate for this search direction calculation. This stopping rule is similar to that in [10], and utilizes the fact that $f_{pp}(\cdot)$ is non-negative on $\overline{C}$. Two numbers $\varepsilon_1, \varepsilon_2 > 0$ are selected, respectively representing an absolute and relative accuracy. Step 1 of Algorithm 4.1 is replaced by the following rule, where x_i is the current iterate. If $f_{pp}(x_i) \le \varepsilon_1$ stop, otherwise if $\dfrac{|\theta(x_i)|}{f_{pp}(x_i)} \le \varepsilon_2$ then stop. If the Algorithm stops by satisfying the second inequality the following condition holds:

$$\frac{\min_{\zeta \in \overline{C}} f_{pp}(\zeta)}{f_{pp}(x_i)} \ge 1 + \frac{\theta(x_i)}{f_{pp}(x_i)} \ge 1 - \varepsilon_2, \tag{5.16}$$

and so $f_{pp}(x_i)$ is within $100\varepsilon_2\%$ of the optimal value.

Consider the implementation of Subprocedure 4.1 for the function $f_{pp} : \mathbb{R}^{n+1} \to \mathbb{R}$. Each affinely independent set B_i of Subprocedure 4.1 may be represented by a matrix $\overline{B}_i \in \mathbb{R}^{(n+1)\times k_i}$, where k_i is the cardinality of B_i, and whose columns are the elements of B_i (a subset of the t_i generated by Step 2 of Algorithm 4.1). Proposition 5.1 provides a means of computing $i_{f_{pp}}(aff\ B_i)$ and $m_{f_{pp}}(aff\ B_i)$ in terms of operations on the matrix $\overline{B}_i$. Partition the matrix $\overline{B}_i$ as in (5.9). By determining the rank of the matrix A (5.10), $i_{f_{pp}}(aff\ B_i)$ may be computed. If A has full rank, then $m_{f_{pp}}(aff\ B_i)$ may be obtained by solving the system (5.11). If A is singular, Step 5 requires a point $m_i \in aff\ B_i \cap (co\ B_i)^C$ satisfying

$f(m_i) \leq f(x_i)$. Step 6 the obtains the successor x_{i+1} by intersecting $[x_i, m_i]$ with $rb\ co\ B_i$. For the function $f_{pp}(\cdot)$ these two steps may be combined by computing a suitable direction of descent for $f_{pp}(\cdot)$ and then moving in this direction until a point in $rb\ co\ B_i$ is obtained (i.e. x_{i+1}). Proposition 5.1 (ii) shows that $-e_0$ is a suitable direction. A convenient way of computing x_{i+1} is to solve the system

$$\begin{bmatrix} e^T \\ \overline{\mathbf{B}}_i \end{bmatrix} \eta = \begin{bmatrix} 0 \\ -e_0 \end{bmatrix}. \tag{5.17}$$

It is straightforward to verify that (5.17) has a unique solution since $\mathbf{B}_i$ is affinely dependent and $\overline{\mathbf{B}}_i$ is affinely independent. Let μ be the barycentric coordinates of the current iterate x_i. Clearly $\overline{\mathbf{B}}_i(\mu + \lambda\eta) \in aff\ B_i$ for all $\lambda \in \mathbb{R}$, and by choosing

$$\lambda = \min\{ -\frac{\mu^j}{\eta^j} \mid \eta^j < 0 \}, \tag{5.18}$$

the point $x_{i+1} \triangleq \overline{\mathbf{B}}_i(\mu + \lambda\eta)$ is obtained. Since at least one component of $\mu + \lambda\eta$ is zero, $x_i \in rb\ co\ B_i$ and proposition 5.1 (ii) ensures that $f_{pp}(x_{i+1}) < f_{pp}(x_i)$. The set B_{i+1} is computed by dropping all elements whose corresponding multiplier is zero.

Some numerical details relevant to the preceeding discussion will now be considered. Of particular concern are the issues of rank determination and speed of execution. Since either the system (5.11) or (5.17) must be solved at each iteration, the latter concern suggests that any decompositions used to solve these systems should be updated incrementally rather than performing a complete decomposition at each stage. It may be readily verified that the matrix A (5.10) is invertible if and only if the matrix $H \triangleq \mathbf{B}^T Q \mathbf{B} + e e^T$ is invertible. Working with the positive semi-definite matrix H instead of A has some advantages which will be described subsequently. It should be noticed at this stage that H (and hence A) can only drop rank by one. This follows since Subprocedure 4.1 is always initialized with an affinely independent set B which satisfies $i_{f_{pp}}(B) = 1$, and the only time the corresponding H may drop rank is when B is augmented by the vector t in Step 2. Subprocedure 4.1 always returns an affinely independent set $\hat{B}$ satisfying $i_{f_{pp}}(\hat{B}) = 1$ (and hence the corresponding H has full rank).

The four types of operations performed on H are, (i) to determine rank (or some approximation to its numerical rank), (ii) to augment H to form H_+, when the matrix $\overline{\mathbf{B}}$ is replaced by the augmented

matrix $\overline{\mathbf{B}}_+ \triangleq \left[\overline{\mathbf{B}} \quad \overline{t}\right]$ (where $\overline{t} = (t^0, t^T)^T$), (iii) to downdate H by removing a column of $\overline{\mathbf{B}}$ and (iv) to solve systems of the form $Hx = b$ (to solve (5.11) or (5.17)). The Cholesky decomposition may be used to perform the operations (i)-(iv) in a convenient manner. Suppose that H is positive definite and that R^TR is its Cholesky decomposition, where R is upper triangular. Given the factorization of H, the factorization of $H_+ \triangleq \mathbf{B}_+^T Q \mathbf{B}_+ + e e^T$ (operation (ii) above), where $\mathbf{B}_+$ is the "lower" part of $\overline{\mathbf{B}}_+$ as in (5.9)), is given by:

$$R_+ \triangleq \begin{bmatrix} R & r \\ 0 & \gamma \end{bmatrix} \tag{5.19}$$

where $r \triangleq (R^T)^{-1}(\mathbf{B}^T Q t + e)$ and $\gamma \triangleq \sqrt{t^T Q t + 1 - r^T r}$. It is straightforward to show that (theoretically) γ is well defined and that H_+ is indefinite if and only if $\gamma = 0$. An approximate determination of the numerical rank may be made by examinining γ^2. If γ^2 is less than some specified tolerance, then it is replaced by 0, and $i_{f_{pp}}(B \cup \{ t \})$ is taken to be 0 (B is the affinely independent set used to generate H). To remove an element from the set B (i.e. remove a column from the matrix $\overline{\mathbf{B}}$), remove the corresponding column from R and use Givens rotations ([15, p 45]) to restore the reduced R to upper triangular form. To solve the system (5.11) (assuming that A and hence H are invertible), let

$$\mu_0 \triangleq H^{-1}\mathbf{b}^0, \qquad \eta \triangleq H^{-1}e, \tag{5.20}$$

and define

$$\alpha \triangleq \frac{1 - \langle \mu_0, e \rangle}{\langle \eta, e \rangle}. \tag{5.21}$$

Then the pair $((1-\alpha), (\mu_0 + \alpha\eta)^T)^T$ solves the system (5.11). If the matrix H_+ (formed by operation (ii)) drops rank (i.e. $\gamma = 0$), then the system (5.17) may be solved by obtaining a solution η to the system

$$H_+\eta = 0, \qquad \langle \mathbf{b}_+^0, \eta \rangle = -1, \tag{5.22}$$

where $\mathbf{b}_+^0 \triangleq ((\mathbf{b}^0)^T, t^0)^T$. This is equivalent to solving the system

$$u \triangleq R^{-1}r, \tag{5.23}$$

(where R and r are as in (5.19)), in which case the solution to (5.17) is given by

$$\eta \triangleq \frac{-\begin{bmatrix} u \\ -1 \end{bmatrix}}{\langle b_+^0, \begin{bmatrix} u \\ -1 \end{bmatrix} \rangle}. \tag{5.24}$$

The algorithm was coded in C and tested by solving a number of problems with known solutions. To examine the numerical behavior of the Cholesky decomposition scheme described above, each iteration was performed in parallel using LINPACK routines to solve the systems (5.10) and (5.17), and the results compared. No significant difference in the results was observed.

In general, the algorithm performed significantly better than the Frank-Wolfe algorithm, in terms of time and number of iterations required to solve a given problem. To give an indication of the relative performance, both algorithms were applied to the minimization of $f_{pp}(\cdot)$ on the set

$$\overline{C} \triangleq \{ x \in \mathbb{R}^3 \mid x^0 \geq 1 + \tfrac{1}{2}\left[\frac{(x^1)^2}{10} + \frac{(x^2)^2}{1000}\right];\ x^0 \leq 10^6 \}, \tag{5.25}$$

with the scaling matrix Q (5.8) taken to be the identity. The solution is easily seen to be $\hat{x} \triangleq (1,0,0)^T$. This set is similar to Example 3 in [3]. Figures 5.1 and 5.2 show the cost versus time and iteration number respectively, starting from $x_0 \triangleq (6.0005, 10, -1)^T$. The times were measured on a VAXstation II/GPX running Ultrix V2.0. The solid line represents the Frank-Wolfe method, and the dashed line represents Algorithm 4.1. To interpret the performance of the algorithm when applied to other sets $\overline{C}$, the time required to compute a tangency point to the set must be considered. The computation of the tangency point (Step 2, Algorithm 4.1) to the set $\overline{C}$ (5.25) requires a negligible amount of time compared with that required to execute the algorithm. Consequently, if the tangency point routine for a different set also requires a negligible amount of time, it is reasonable to expect that the cost versus time graph would be similar to Figure 5.1. However, if the tangency point routine requires a significant amount of time, the the cost versus iteration graph (Figure 5.2) would give a more realistic estimate of the time required to solve the problem. An empirical estimate of the rate of convergence of the cost can be made by examining Figure 5.3, which graphs $\log(f_{pp}(x_i) - f_{pp}(\hat{x}))$ versus the iteration number i. Algorithm 4.1 is seen to have a linear rate of convergence in this example.

If the set $\bar{C}$ has the form $\bar{C} \triangleq co\ \{\ \bar{x}_i\ \}_{i \in m}$, then the performance may be compared with that of a quadratic program solver in the following manner. Define the matrix $\bar{X}$ as

$$\bar{X} = \begin{bmatrix} (x^0)^T \\ X \end{bmatrix} \triangleq \begin{bmatrix} \bar{x}_1 \dots \bar{x}_m \end{bmatrix}. \tag{5.26}$$

Then problem $\mathbf{SDP_1}$ (5.7) is equivalent to:

$$\mathbf{QP:} \quad \min\{\ (x^0)^T\mu + \mu^T X^T Q X \mu \mid \mu \in \mathbb{R}^m;\ \ \mu \geq 0;\ \ e^T\mu = 1\ \}. \tag{5.27}$$

As before, the matrix Q is taken to be the identity. The quadratic program solver used was the LSSOL package [16] (default options, problem type QP2, print level 10). The set C was formed by taking m points at random, distributed uniformly on the set $[0,5]\times[-10,10]^{n-1} \subset \mathbb{R}^n$. In all cases the Frank-Wolfe algorithm required more than 200 iterations and more time than either LSSOL or Subprocedure 4.1 To provide a resonable basis for comparison, the time taken to compute the matrix X^TX was added to the time taken to solve the quadratic program. The times (in seconds) required to solve the problem for varying dimension and number of points are given in Tables 5.1 and 5.2.

Time (s) using Algorithm 4.1 on fpp				
	number of points			
n	10	25	50	100
10	0.13	0.53	0.72	1.13
25	0.92	2.03	4.79	7.38
50	2.89	8.91	15.42	38.93
100	11.98	30.62	63.59	114.03

Table 5.1. Time required to solve $\mathbf{SDP_1}$ with Algorithm 4.1.

Time (s) LSSOL on fpp				
	number of points			
n	10	25	50	100
10	0.68	2.15	5.26	12.6
25	0.85	3.49	10.75	37.69
50	0.8	3.26	15.15	76.88
100	0.98	4.08	16.16	98.62

Table 5.2. Time required to solve $\mathbf{SDP_1}$ with LSSOL.

The tables suggest that the algorithms have comparable performance when the number of points m is equal to the dimension of the underlying space n, and that Algorithm 4.1 performs considerably better when $m > n$. In the search direction calculations of many engineering design problems, the parameter space dimension n is considerably smaller than the number of points, and consequently we would expect that Algorithm 4.1 performs better that LSSOL on these problems. In the case where $m < n$, the number of iterations required by Algorithm 4.1 to solve the problem is approximately equal to the number of points carrying the final solution (and also approximately equal to m). Since the algorithm only picks up one extra point per iteration, we observe that the time in Figure 4.1 is that required to "collect" all m points via the tangency point routine. Since LSSOL has the full description of the set $\overline{C}$ available initially through the set $\overline{X}$, it does not suffer from this disadvantage.

6. CONCLUSION

We have presented an extension of the von Hohenbalken algorithm [5], which maintains the desirable property of finite convergence when the set C (in (1.1)) is a polytope, and also converges when the set C is a general convex compact set. Such (non-polytopal) sets arise in many engineering design problems. We have also presented an implementation for a specific search direction problem, along with some numerical results. We observe that our modified algorithm exhibits good performance in most cases. So far we have not encountered a case where our modified algorithm has not converged linearly. We are therefore led to suspect that is should be possible to establish theoretically that, under suitable conditions, the algorithm converges to a solution with a linear rate.

APPENDIX: A counter example

In this appendix, a convex compact (non-polytopal) set C is constructed on which the von Hohenbalken algorithm [5] does not converge when applied to the function $f(x) \triangleq \frac{1}{2}\|x\|^2$, for certain initial starting points. Strictly speaking, the algorithm [5] *maximizes* pseudo-concave functions, so to minimize a pseudo-convex function $f(\cdot)$, the algorithm is applied to the function $-f(\cdot)$.

Given the set C a sequence $\{ \xi_i \}_{i=0}^{\infty}$ is created which satisfies $\xi_{i+1} \in \arg\min_{\zeta \in C} \langle \zeta, \nabla f(\xi_i) \rangle$, $f(\xi_{i+1}) < f(\xi_i)$, and ξ_i does *not* converge to the (unique) solution. As a result, when the algorithm of [5] is used (starting from the point ξ_0) it will cycle indefinitely between Basic step 1 and Basic step 2(a), generating the sequence $\{ \xi_i \}_{i=0}^{\infty}$, which does not converge to the solution.

Define the function $\lambda : \mathbb{R} \rightarrow \mathbb{R}$ as

$$\lambda(x) \triangleq e^{\alpha(|x|-1)} + \kappa \tag{A.1}$$

(where $\alpha \triangleq 10/11$, $\kappa \triangleq 1/10$) and the set C as

$$C \triangleq \{ x \in \mathbb{R}^2 \mid \lambda(x_2) \leq x_1 \leq \lambda(11) \}. \tag{A.2}$$

Since $\lambda(\cdot)$ is convex, it is clear that the set is convex. Compactness of C is obvious. Note in passing that $\lambda(\cdot)$ is differentiable everywhere except at 0. Define the cone Λ as

$$\Lambda \triangleq \{ x \in \mathbb{R}^2 \mid x_1 > 0; \quad \alpha e^{-\alpha} x_1 < |x_2| < \alpha e^{10\alpha} x_1 \}. \tag{A.3}$$

It is straightforward to show that for all $h \in \Lambda$,

$$\min_{\zeta \in C} \langle \zeta, h \rangle = \min_{\zeta \in epi\, \lambda} \langle \zeta, h \rangle = \min_{y \in \mathbb{R}} \lambda(y) h_1 + y h_2 , \tag{A.4}$$

where *epi* λ denotes the epigraph of $\lambda(\cdot)$. Define the function $y : \Lambda \rightarrow \mathbb{R}$ by

$$\arg\min_{\zeta \in C} \langle \zeta, h \rangle = \{ (\lambda(y(h)), y(h)) \} , \tag{A.5}$$

where $y(h)$ satisfies

$$\alpha e^{\alpha(|y(h)|-1)} = \frac{|h_2|}{h_1} , \qquad y(h) h_2 < 0. \tag{A.6}$$

Define a collection of neighborhoods of the points 1 and -1 by $U_\varepsilon \triangleq \{ x \in \mathbb{R} \mid |\,|x| - 1| < \varepsilon \}$. Since

Λ is open, and $(\lambda(1),1),(\lambda(-1),-1) \in \Lambda$, it is clear that $(\lambda(x),x) \in \Lambda$ for all $x \in U_{\varepsilon_1}$, for some $\varepsilon_1 > 0$. For convenience, define

$$\phi(x) \triangleq \frac{|x|/\alpha}{e^{\alpha(|x|-1)} + \kappa}, \qquad \theta(x) \triangleq \frac{\phi(x)}{e^{\alpha(|x|-1)}}. \tag{A.7}$$

Direct calculation shows that $\phi(1) = \theta(1) = 1$, $\phi(-1) = \theta(-1) = 1$, $\phi'(1) > 0$, $\phi'(-1) < 0$, $\theta'(1) < 0$, and $\theta'(-1) > 0$. Hence there exists an $\varepsilon_2 > 0$ such that if $x \in U_{\varepsilon_2}$ and $|x| > 1$, then $\phi(x) > 1$ and $\theta(x) < 1$. Suppose that $x \in U_\varepsilon$, where $\varepsilon \triangleq \min\{ \varepsilon_1, \varepsilon_2 \}$, and $y((\lambda(x),x))$ is defined by A.5, A.6. Then the following hold

$$e^{\alpha(|y((\lambda(x),x))|-1)} = \phi(x) > 1, \qquad \frac{e^{\alpha(|y((\lambda(x),x))|-1)}}{e^{\alpha(|x|-1)}} = \theta(x) < 1, \tag{A.8}$$

and we may conclude that $|y((\lambda(x),x))| - 1 > 0$, $|y((\lambda(x),x))| < |x|$, and hence $y((\lambda(x),x)) \in U_\varepsilon$.

To continue, notice that $\nabla f(0) = 0$, and so the Initial step of the algorithm may select any extreme point of the set C as an initial point. Choose any $x \in U_\varepsilon$, with $|x| > 1$, and define $\xi_0 \triangleq (\lambda(x),x)$ (it is straightforward to verify that ξ_0 is an extreme point of C). Define the sequence inductively by $\xi_{i+1} \triangleq (\lambda(y(\xi_i)), y(\xi_i))$. Observe that $|(\lambda(x),x)| > |(\lambda(z),z)|$ whenever $|x| > |z|$. Hence by A.8, and the associated remarks we see that $|\xi_{i+1}| < |\xi_i|$ (i.e. $f(\xi_{i+1}) < f(\xi_i)$). In addition, since

$$\xi_{i+1} \in \arg\min_{\zeta \in C} \langle \zeta, \nabla f(\xi_i) \rangle = \arg\min_{\zeta \in C} \langle \zeta, \xi_i \rangle, \tag{A.9}$$

it follows that if the algorithm is initialized at ξ_0, then it will generate the sequence $\{ \xi_i \}_{i=0}^{\infty}$. Finally, since $|\xi_i^2| > 1$ for all i, we observe that the sequence does not converge to the solution $(\lambda(0),0)$.

REFERENCES

[1] Barr, R. O., "An Efficient Computational Procedure for a Generalized Quadratic Programming Problem," *SIAM J. Control* 7(3)(1969).

[2] Frank, M. and Wolfe, P., "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly* 3 pp. 95-110 (1956).

[3] Gilbert, E. G., "An Iterative Procedure for Computing the Minimum of a Quadratic Form on a Convex Set," *SIAM J. Control* 4(1)(1966).

[4] Wolfe, P., "Finding the Nearest Point in a Polytope," *Mathematical Programming* 11 pp. 128-149 (1976).

[5] von Hohenbalken, B., "A Finite Algorithm to Maximize Certain Pseudoconcave functions on Polytopes," *Mathematical Programming* 9 pp. 189-206 (1975).

[6] Pironneau, O. and Polak, E., "On the Rate of Convergence of Certain Methods of Centers," *Mathematical Programming* 2(2) pp. 230-258 (1972).

[7] Polak, E., "On the Mathematical Foundations of Nondifferentiable Optimization in Engineering Design," *SIAM Review* 29(1) pp. 21-89 (March 1987).

[8] Polak, E. and Salcudean, S. E., "On the Design of Linear Multivariable Feedback Systems via Constrained Nondifferentiable Optimization in H^∞ Spaces," *IEEE Transactions on Automatic Control*, (in press).

[9] Canon, M. D. and Cullum, C. D., "A Tight Upper Bound on the Rate of Convergence of the Frank-Wolfe Algorithm," *SIAM J. Control* 6(4)(1968).

[10] Hauser, J. E., "Proximity Algorithms: Theory and Implementation," Memo No. UCB/ERL M86/53, Electronics Research Laboratory, University of California, Berkeley, California (20 May 1986).

[11] Rockafellar, R. T., *Convex Analysis*, Princeton University Press, Princeton (1970).

[12] Mangasarian, O. L., *Non-linear Programming*, McGraw-Hill, New York (1969).

[13] Zangwill, W. I., *Nonlinear Programming: A Unified Approach*, Prentice-Hall, Englewood Cliffs, N.J. (1969).

[14] Polak, E., *Computational Methods in Optimization*, Academic Press, New York, N.Y. (1971).

[15] Golub, G. H. and Van Loan, C. F., *Matrix Computations*, John Hopkins University Press, Baltimore, Maryland (1983).

[16] Gill, P. E., Hammarling, S. J., Murray, W., Saunders, M. A., and Wright, M. H., "User's Guide for LSSOL (Version 1.0): A Fortran Package for Constrained Linear Least-squares and Convex Quadratic Programming," Technical report SOL 86-1, Department of Operations Research, Stanford University, Stanford (January 1986).

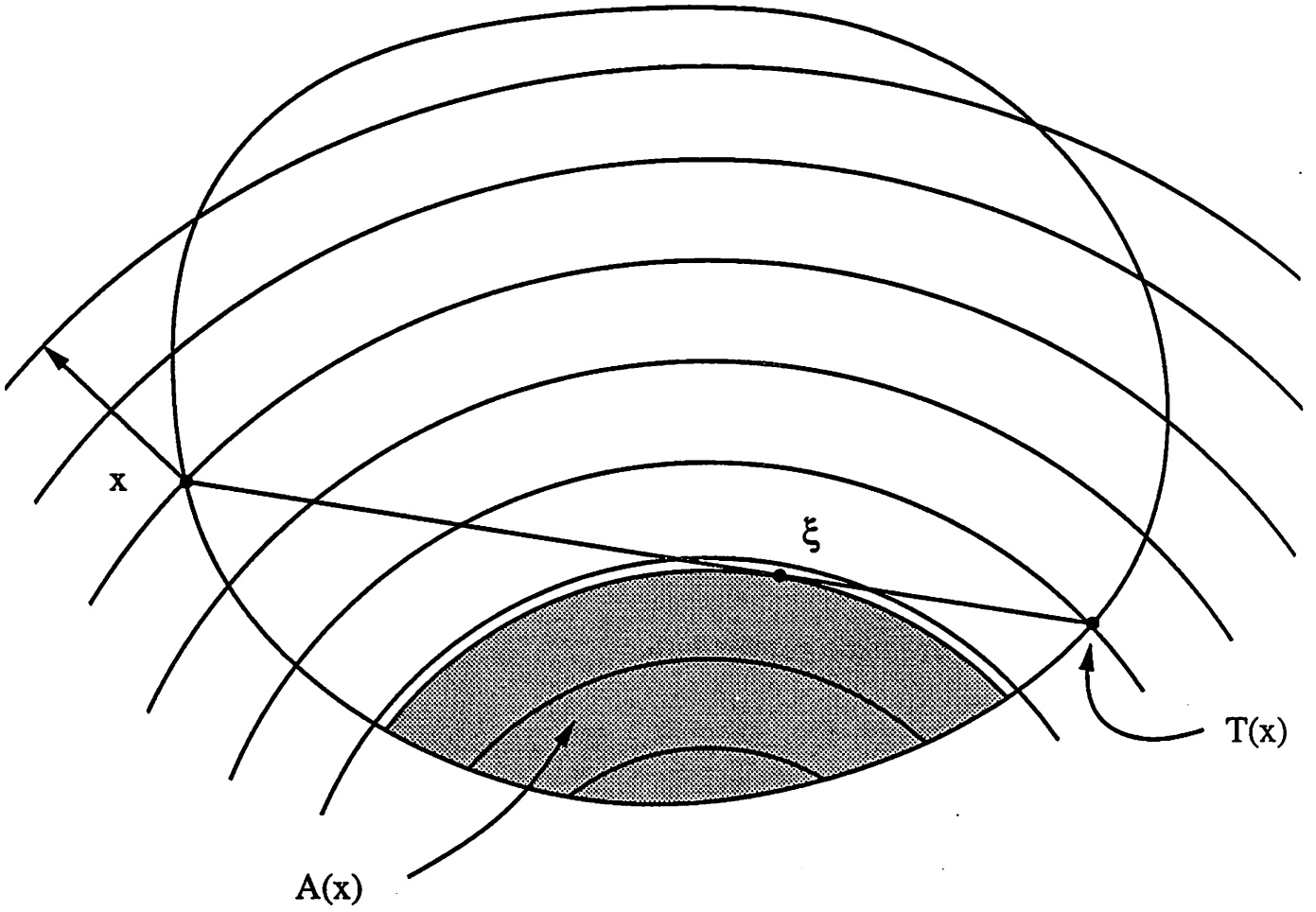

Figure 3.1. Illustration of the sets T(x) and A(x).

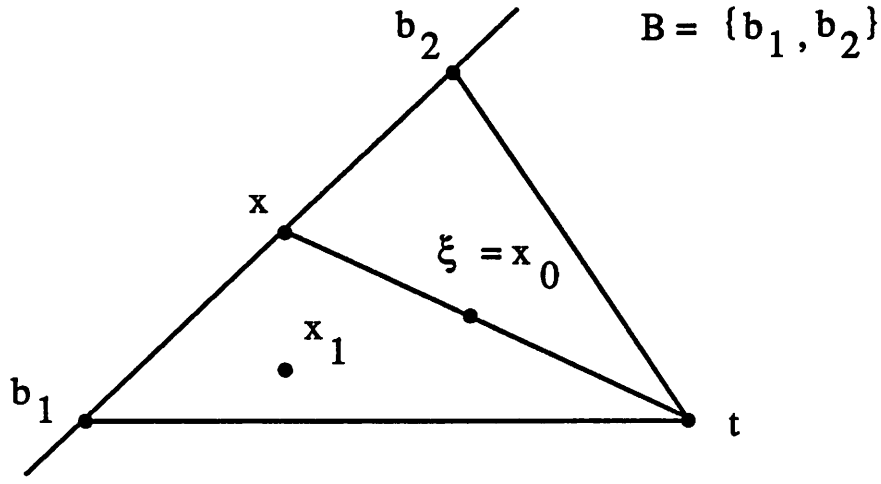

Figure 4.1. Minimizer inside convex hull.

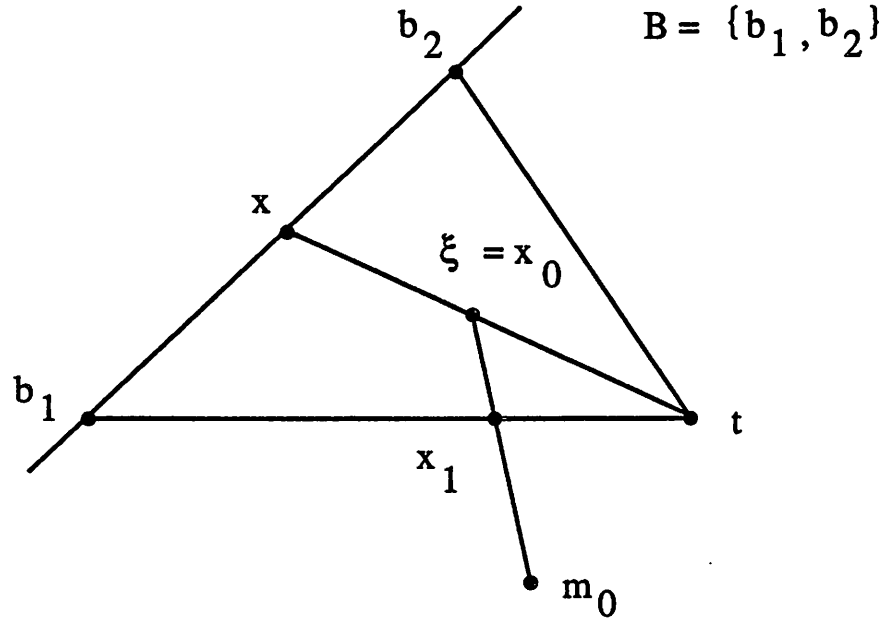

Figure 4.2. Minimizer does not exist or is outside convex hull.

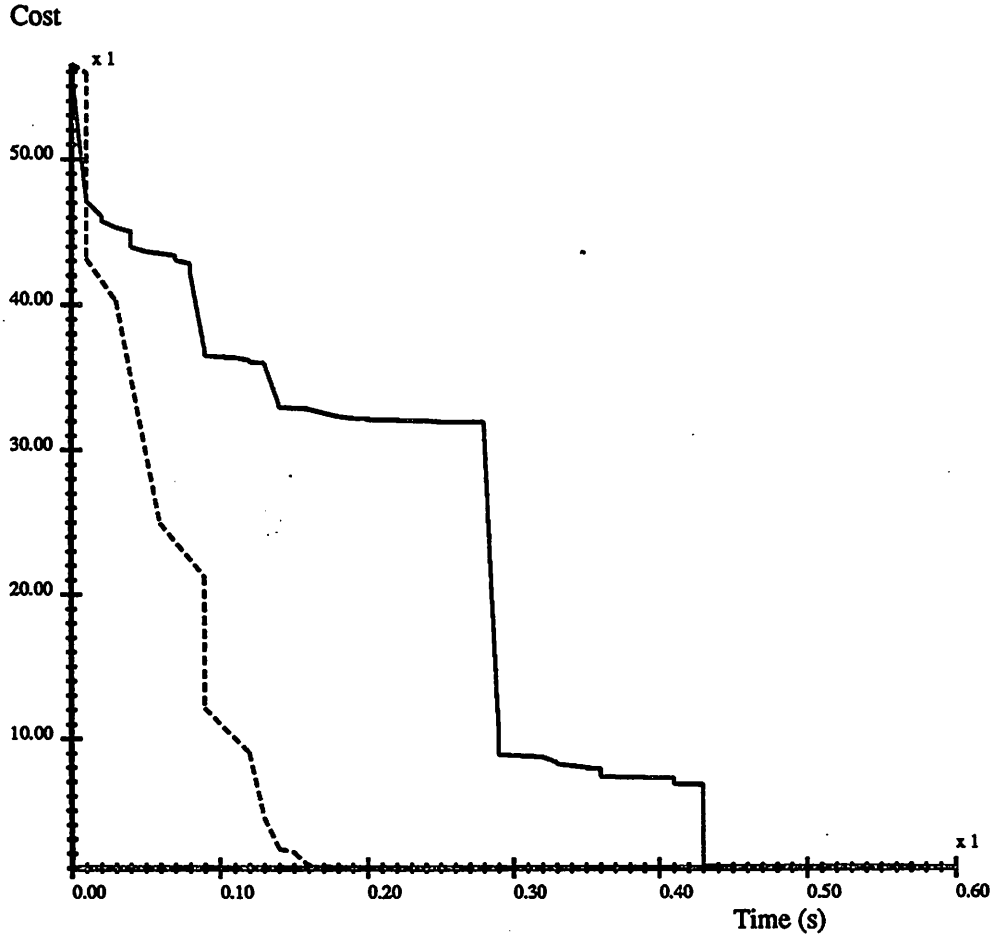

Figure 5.1. Cost vs. time plots for the Frank-Wolfe algorithm (solid) and Algorithm 4.1 (dashed).

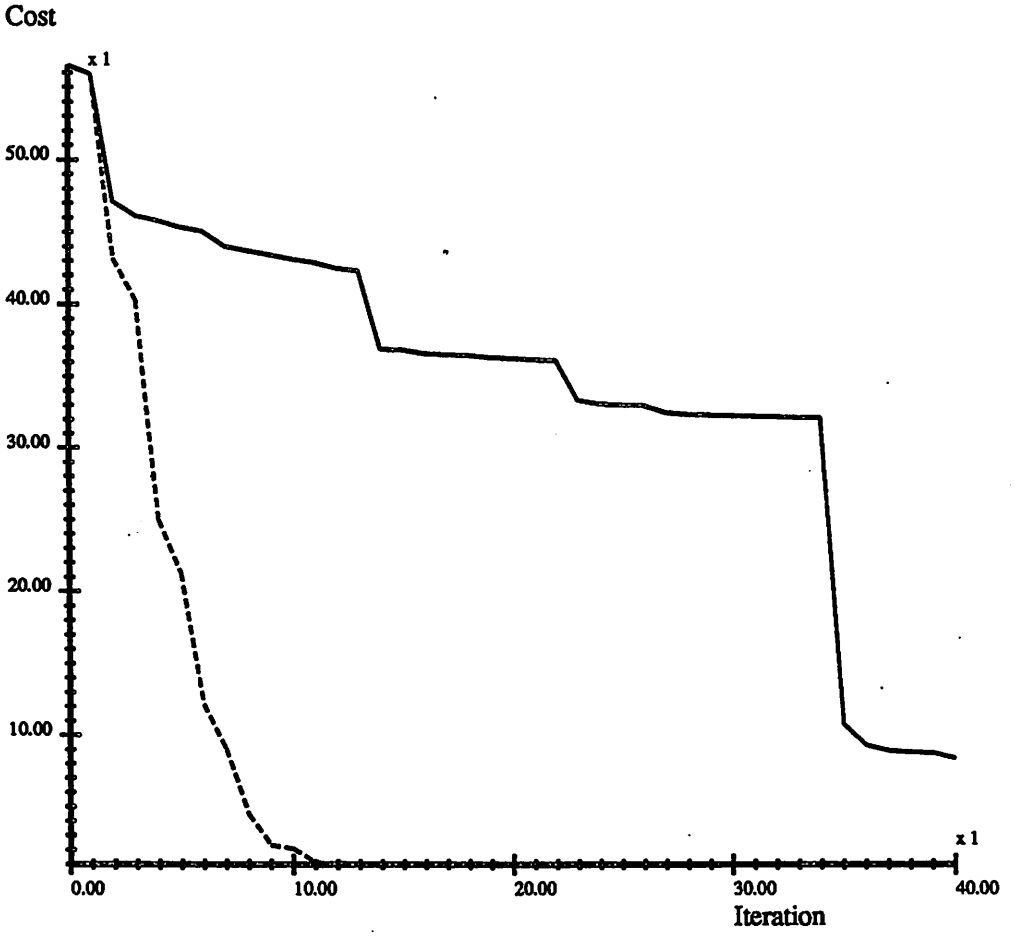

Figure 5.2. Cost vs. iteration plots for the Frank-Wolfe algorithm (solid) and Algorithm 4.1 (dashed).

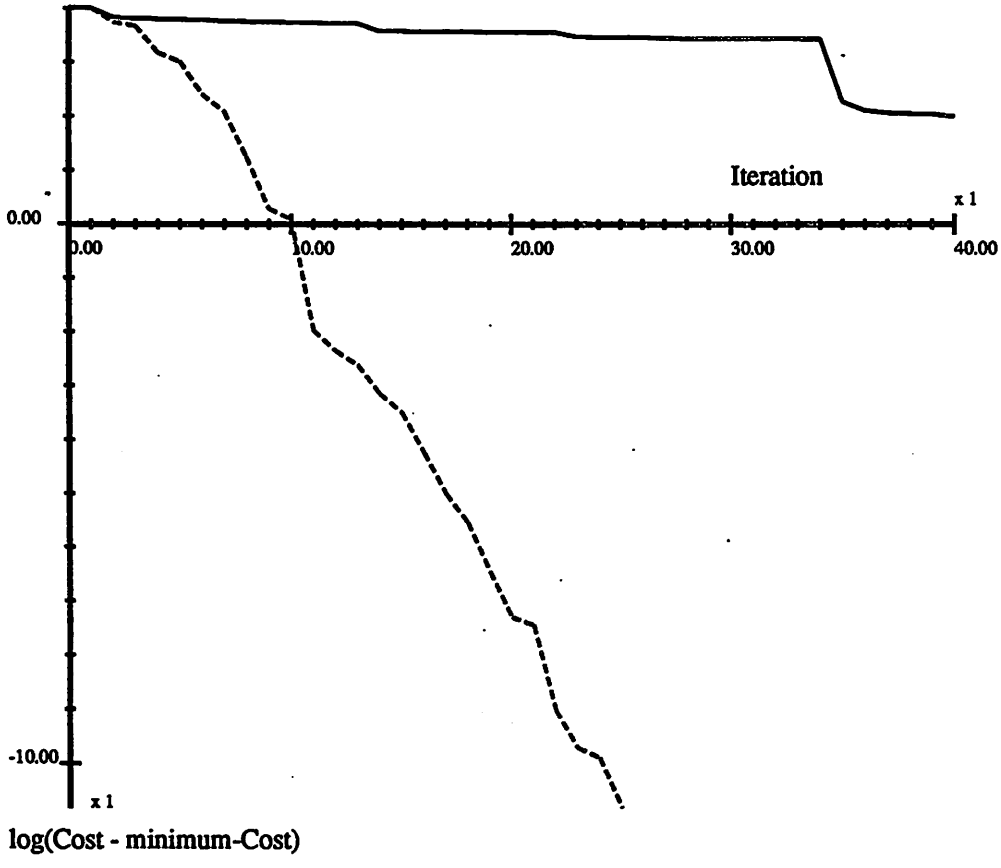

Figure 5.3. Rate of convergence of the Frank-Wolfe algorithm (solid) and Algorithm 4.1 (dashed).