Al-Sultan, Khaled Saleh

EXTERIOR POINT ALGORITHMS FOR NEAREST POINTS AND CONVEX QUADRATIC PROGRAMS

EXTERIOR POINT ALGORITHMS FOR NEAREST POINTS AND CONVEX QUADRATIC PROGRAMS

K.S. Al-Sultan and K.G. Murty
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109-2117, U.S.A.

and

Systems Engineering Department
King Fahd University of Petroleum and Minerals
Dhahran-31261, Saudi Arabia

[illegible]

[illegible]

October, 1989

Technical Report 91-6 (revised March 1991)

Partially supported by NSF Grant No. ECS-8521183

Convex programming
Quadratic programming
Algorithms.

Abstract

We consider the problem of finding the nearest point (by Euclidean distance) in a simplicial cone to a given point, and develop an exterior penalty algorithm for it. Each iteration in the algorithm consists of a single Newton step following a reduction in the value of the penalty parameter. Proofs of convergence of the algorithm are given. Various other versions of exterior penalty algorithms for nearest point problems in nonsimplicial polyhedral cones and for convex quadratic programs, all based on a single descent step following a reduction in the value of the penalty parameter per iteration, are discussed. The performance of these algorithms in large scale computational experiments is very encouraging. It shows that the number of iterations grows very slowly, if at all, with the dimension of the problem.

Key words: Exterior penalty function methods, nearest point problems, convex quadratic programs, Newton step, SOR methods.

Abbreviated title: Exterior Point-algorithms

1 Introduction

We consider the *nearest point problem* , which is that of finding the nearest point by Euclidean distance in a convex polyhedral cone $\mathbf{K} \subset \mathbf{R}^n$ to a point $q \in \mathbf{R}^n$. It is a special convex quadratic program with many important applications (see [1, 2, 4, 8, 16, 17]). The KKT optimality conditions for the problem, form the linear complementarity problem (LCP) associated with it. The nearest point problem can be solved by specializations of descent or active set methods for linearly constrained nonlinear programs [3, 15, 16], or through the associated LCP using any of the available algorithms for solving LCPs such as complementary and principal pivoting methods [2, 13, 14, 16], interior point methods [12, 24], and special methods based on its geometry [2, 8, 17, 22, 23]. Thus a variety of methods are already available for the nearest point problem. However, since it appears often as a large scale problem, there has been a continuing motivation to develop new algorithms which are faster. This is our motivation too and our inspiration comes from recent reports of significant improvements in computational performance for solving large scale linear and convex quadratic programs through the use of barrier methods in novel ways. We discuss a family of penalty methods for our problem and report on their encouraging computational performance.

For any matrix D, we denote its ith row, jth column by $D_{i.}$, $D_{.j}$ respectively. The symbol (x_j) denotes the vector whose jth component is x_j. For any real numbers $a_1, \ldots, a_m$, diag $(a_1, \ldots, a_m)$ denotes the $m \times m$ diagonal matrix with entries in the principal diagonal equal to $a_1, \ldots, a_m$. The symbol $\mathbf{S} \backslash \mathbf{T}$ denotes the set of all elements in the set $\mathbf{S}$ which are not in the set $\mathbf{T}$. For any vector x, $||x||$ denotes its Euclidean norm. For a square matrix A, $||A||$ denotes its matrix norm which is max. $\{||Ax|| : \quad ||x|| = 1\}$. Given a matrix Q , Pos $(Q) = \{Qy : y \geqq 0\}$ is the cone which is the nonnegative hull of its column vectors. A square matrix A is said to be positive definite (PD) if $y^T A y > 0$ for all $y \neq 0$, or positive semidefinite (PSD) if $y^T A y \geqq 0$ for all y. Given any differentiable real valued function $h(x)$ defined over $\mathbf{R}^n$, we denote by $\nabla h(x)$ its gradient vector at x, written as a row vector. We use both superscripts and exponents in this paper, and distinguish between them by typeface. Boldface superscripts indicate exponents, for example $\lambda_j^{\mathbf{r}}$ is λ_j raised to the power $\mathbf{r}$. Regular superscripts, as in λ^k, are used to enumerate various vectors, etc.

There are two forms of the nearest point problem, depending on the way the polyhedral cone $\mathbf{K}$ is specified. In one form $\mathbf{K}$ is specified as Pos (Q) for a given matrix Q. In the other form $\mathbf{K}$ is specified as $\{x : Ax \geqq 0\}$ where A is given. Both forms appear in applications. When the problem is given in one of these forms, to transform it into the other form typically requires exponential effort. We develop a family of exterior penalty methods for the nearest point problem when $\mathbf{K}$ is given as Pos(Q). Our convergence analysis and computational experiments treat the problem in this form. However, we show how the methods can be easily adopted to handle problems in the other form, and convex quadratic programs in general.

Without any loss of generality, we assume that the cone $\mathbf{K}$ is of full dimension.

2 Penalty Algorithms When K is a Simplicial Cone Given in the POS Form

Let $\mathbf{K} = \text{Pos}(Q)$ where Q is a nonsingular square matrix of order n. This special case of the nearest point problem itself has many applications [1,4]. In this case, the nearest point problem is: find $\lambda = (\lambda_1, \ldots, \lambda_n)^T$ to

$$\begin{aligned} &\text{minimize } \| q - Q\lambda \|^{\mathbf{2}} \\ &\text{subject to } \lambda \geqq 0. \end{aligned} \tag{1}$$

It is well known that this problem always has a unique optimum solution. If λ^* is the optimum solution of this problem, $x^* = Q\lambda^*$ is the nearest point in $\mathbf{K}$ to q. The penalty approach transforms the problem into one of unconstrained minimization of a *composite function* which is the sum of the original objective function and a penalty function associated with the feasible region. The various penalty methods differ in the choice of the penalty function. In this study we consider penalty functions of the form $P_r(\lambda) = \sum_{j=1}^{n}(\max.\{0, -\lambda_j\})^r$ for $r = 2, 3, 4$. This leads to the following unconstrained minimization problem in λ, where μ is a positive penalty parameter.

$$\text{minimize } f_r(\lambda, \mu) = \| q - Q\lambda \|^2 + \frac{1}{\mu}\sum_{j=1}^{n}(\max.\{0, -\lambda_j\})^r \text{ over } \lambda \in \mathbf{R}^n \tag{2}$$

$f_r(\lambda, \mu)$ is convex in the λ-space for fixed $\mu > 0$. $f_2(\lambda, \mu)$ is continuously differentiable in λ up to the first order, $f_3(\lambda, \mu)$ and $f_4(\lambda, \mu)$ are twice continuously differentiable in λ. Let $g_r(\lambda, \mu), H(f_r(\lambda, \mu)$ be the gradient as a row vector, and the Hessian matrix of $f_r(\lambda, \mu)$ with respect to λ. $g_r(\lambda, \mu)$ for $r = 2, 3, 4$; and $H(f_r(\lambda, \mu))$ for $r = 3, 4$ are given below.

$$\begin{aligned} g_r(\lambda, \mu) = \nabla f_r(\lambda, \mu) &= -2q^TQ + 2\lambda^TQ^TQ + \frac{r}{\mu}(\delta_1\lambda_1^{r-1}, \ldots, \delta_n\lambda_n^{r-1}) \\ H(f_r(\lambda, \mu)) &= 2Q^TQ + \frac{r(r-1)}{\mu} \text{ diag } (\delta_1\lambda_1^{r-2}, \ldots, \delta_n\lambda_n^{r-2}) \end{aligned} \tag{3}$$

$$\text{where } \delta_j = \delta_j(\lambda_j) = \begin{cases} 0 & \text{if } \lambda_j \geqq 0 \\ (-1)^r & \text{if } \lambda_j < 0. \end{cases}$$

In penalty algorithms, the unconstrained minimization of $f_r(\lambda, \mu)$ in λ is carried out by a descent method. This method begins with an initial point λ^0 and goes through several steps. In each step a descent direction at the current point λ^k for $f_r(\lambda, \mu)$ is generated, and a step is taken from λ^k in that direction. The step length to move in that direction is either 1 (for algorithms based on standard Newton's method) or the optimum step length determined by a line search in that direction. We will now describe the line search routine, and choice of descent directions that we will use.

Line Search Routine for $f_r(\lambda, \mu)$

Let $\overline{\lambda} \in \mathbf{R}^n$ and suppose $d = (d_1, \dots, d_n)^T$ is a descent direction for $f_r(\lambda, \mu)$ at $\overline{\lambda}$. Then

$$\frac{df_r(\overline{\lambda} + \alpha d, \mu)}{d\alpha} = (-2d^T Q^T q + 2d^T Q^T Q\overline{\lambda}) + 2\alpha d^T Q^T Q d + \frac{1}{\mu} \sum_{j=1}^{n} r(-1)^r (\overline{\lambda}_j + \alpha d_j)^{r-1} d_j \zeta_j(\alpha) \tag{4}$$

where $\zeta_j(\alpha) = 0$ if $\overline{\lambda}_j + \alpha d_j \geqq 0$, and 1 if $\overline{\lambda}_j + \alpha d_j < 0$. So $\frac{df_r(\overline{\lambda}+\alpha d,\mu)}{d\alpha}$ is a sum of at most $n + 2$ terms. Determine the set of ratios $\{\frac{-\overline{\lambda}_j}{d_j} : j$ such that either $\overline{\lambda}_j > 0, d_j < 0$, or $\overline{\lambda}_j < 0, d_j > 0\}$, and arrange them in increasing order. Suppose there are s distinct values in this set, $\alpha_1 < \alpha_2 < \dots < \alpha_s$. Define $\alpha_{s+1} = \infty$. For any $u = 1$ to s, in the interval $\alpha_u < a < a_{u+1}$, each of the terms on the right hand side of (4) is monotonic and has the same sign. Since d is a descent direction for $f_r(\lambda, \mu)$ at $\overline{\lambda}$, $\frac{df_r(\overline{\lambda}+\alpha d,\mu)}{d\alpha} < 0$ at $\alpha = 0$, and as $f_r(\lambda, \mu)$ is convex and has a minimum, this derivative becomes 0 at some point in this direction. So, if we begin computing $\frac{df_r(\overline{\lambda}+\alpha d,\mu)}{d\alpha}$ for $\alpha = 0, \alpha_1, \alpha_2, \dots$ there will be a t such that it remains negative until we get to α_t, and at α_{t+1} it becomes $\geqq 0$. Then, we know that the minimum of $f_r(\lambda, \mu)$ in λ, over the half-line $\{\overline{\lambda} + \alpha d : \alpha \geqq 0\}$, is attained by some α in the interval $\alpha_t < \alpha \leqq \alpha_{t+1}$, and it can be found by an efficient search for the zero of $\frac{df_r(\overline{\lambda}+\alpha d,\mu)}{d\alpha}$ in this interval, as all the terms on the right hand side of (4) are monotonic and of the same sign.

Selection of Search Directions

We basically consider two directions.

Steepest descent search direction: Under this rule, the search direction at the current point λ^k for $f_r(\lambda, \mu)$ is $-\nabla f_r(\lambda^k, \mu)$, and step length for the move in this direction is taken to be the optimum step length determined as discussed above. The method based on this strategy is the *steepest descent* version of the method.

Newton search direction: Under this rule the search direction y at the current point λ^k for $f_r(\lambda, \mu)$ is a solution of the system

$$H(f_r(\lambda^k, \mu))y = (\nabla f_r(\lambda^k, \mu))^T \tag{5}$$

Since Q is square and nonsingular, $H(f_r(\lambda, \mu))$ is PD and (5) always has a unique solution. When using the Newton search direction, we can take the step length for the move to be 1 in all the steps, this leads to the *standard Newton method* ; or optimum step lengths leading to Newton method with line searches.

The Classical Exterior Penalty Method

This method, discussed extensively in nonlinear programming literature, goes through several iterations. In each iteration, the value of the penalty parameter μ is fixed, and an algorithm is used to find the unconstrained minimum of $f_r(\lambda, \mu)$ over $\lambda \in \mathbf{R}^n$. This algorithm itself may take several descent steps in this iteration.

Proofs of convergence of the classical penalty method require that the unconstrained minimum of $f_r(\lambda, \mu)$ is obtained in this iteration, but in actual computation a sufficiently small positive tolerance, ε, called the *desired accuracy* , is selected and the unconstrained minimization routine terminated whenever a point satisfying (6) is obtained.

$$\| g_r(\lambda, \mu) \| \leqq \varepsilon \tag{6}$$

Now the penalty parameter is reduced from its present μ to $\overline{\mu}$ say, and the method moves to the next iteration to find the unconstrained minimum of $f_r(\lambda, \overline{\mu})$ beginning with $\overline{\lambda}$ as the initial point.

Let $\lambda(\mu)$ denote an unconstrained minimum of $f_r(\lambda, \mu)$ over λ, as a function of μ. It has been shown [21] that it is possible to select a target value $\mu_* > 0$ for the penalty parameter, such that when μ decreases to this value or becomes smaller, then $\lambda(\mu)$ is within a specified tolerance of an optimum λ for the original problem.

New Exterior Penalty Methods

Our methods are based on the ideas of the classical penalty method, but differ from it in one important aspect. We do not find the unconstrained minimum of $f_r(\lambda, \mu)$ for each fixed μ. Instead we carry out only one descent step of the unconstrained minimization algorithm, then reduce the value of the penalty parameter, and continue the same way.

One can visualize a path in the λ-space, parametrized by μ, $\{\lambda : \nabla f_r(\lambda, \mu) = 0, \mu > 0\}$, called the *exterior path* , and an *envelope* containing this path defined by $\{\lambda :\| \nabla f_r(\lambda, \mu) \| \leqq \varepsilon, \mu > 0\}$ for a certain $\varepsilon > 0$. We will later on prove that the points obtained in this new algorithm, can be interpreted as a sequence of points in this envelop converging to the limit of the point on the exterior path corresponding to μ as μ tends to 0.

Basic Algorithm

Initialization Let ε be the desired accuracy, and μ_* the target value for the penalty parameter value. Select an initial point λ^o satisfying $Q\lambda^o = q$, and an initial value μ_o for the penalty parameter (this could be large). Select a value for the factor ρ between 0 and 1. With (λ^o, μ_o) as the initial pair, go to Step 1.

General step Let (λ^k, μ_k) be the pair at the end of the previous step. Let $\mu_{k+1} = \rho\mu_k$. Find the descent direction (either steepest descent or Newton's) d^k at λ^k for $f_r(\lambda, \mu_{k+1})$. Select the step length α_k (1 for standard Newton method, or the optimum step length for other methods). Let $\lambda^{k+1} = \lambda^k + \alpha_k d^k$. If $\lambda_j^{k+1} \geqq -\varepsilon$ for all j, or if $\mu_{k+1} \leqq \mu_*$, λ^{k+1} is a point within the specified tolerance of the optimum solution for (1), terminate. Otherwise, with the new pair $(\lambda^{k+1}, \mu_{k+1})$ go to the next step.

This is the basic algorithm. Details on how to select ρ, μ_o etc. are specified later on.

Convergence Results

Classical penalty methods are known to converge to an optimum solution of the original problem [3, 7, 15, 19, 21]. However, these proofs assume that an actual optimum solution for the problem of minimizing $f(\lambda, \mu)$ over $\lambda \in \mathbf{R}^n$ is obtained for each fixed value of μ in a sequence converging to zero. We do not really obtain the unconstrained minimum of $f_r(\lambda, \mu)$ over $\lambda \in \mathbf{R}^n$ for any value of μ before we change it. Hence standard proofs of penalty methods do not apply directly to our algorithm. We give a convergence proof for Standard Newton version of the algorithm (one Newton step following a decrease in the penalty parameter μ in every step), using the composite function $f_4(\lambda, \mu)$, this has been selected because each component of its gradient is twice continuously differentiable, and our present proofs use this property. We denote $f_4(\lambda, \mu)$ by $f(\lambda, \mu)$ for the sake of simplicity. Likewise we denote the gradient vector and the Hessian matrix of $f_4(\lambda, \mu)$ with respect to λ by $g(\lambda, \mu), H(\lambda, \mu)$. So

$$\begin{aligned} g(\lambda, \mu) &= -2(q - Q\lambda)^T Q + \frac{4}{\mu}(\delta_1 \lambda_1^3, \dots, \delta_n \lambda_n^3) \\ H(\lambda, \mu) &= 2Q^T Q + \frac{12}{\mu} \text{ diag } (\delta_1 \lambda_1^2 \dots, \delta_n \lambda_n^2) \end{aligned} \qquad (7)$$

where $\delta = (\delta_1, \ldots, \delta_n)$ is a function of λ defined as in (3). We denote by $g_j(\lambda, \mu)$ the jth component of $g(\lambda, \mu)$; and by $H_{.j}(\lambda, \mu)$ the jth column vector of $H(\lambda, \mu)$.

Since Q is nonsingular, $Q^T Q$ is PD. Let σ_1 be the smallest eigenvalue of $Q^T Q$. Also, from (3), $(12/\mu)$ diag$(\delta_1 \lambda_1^2, \ldots \delta_n \lambda_n^2)$ is PSD for every $\mu > 0$. These facts imply that $H(\lambda, \mu)$ is PD for every $\mu > 0$, and that its smallest eignevalue is $\geqq 2\sigma_1$. Hence $f(\lambda, \mu)$ is not only strictly convex , but is a strongly convex function in λ for all $\mu > 0$. They also imply that $\| (H(\lambda, \mu))^{-1} \| \leqq 1/(2\sigma_1)$, for every $\mu > 0$ and $\lambda \in \mathbf{R}^n$.

Theorem 1: For each $\mu > 0$, the system $g(\lambda, \mu) = 0$ has a unique solution, $\lambda(\mu)$ say, in λ. $\lambda(\mu)$ is a continuous function of μ in $\{\mu : \mu > 0\}$, and as $\mu \to 0$ through positive values, $\| \lambda(\mu) \|$ remains bounded above by a constant which depends only on q and Q.

Proof: Fix $\mu > 0$. Select any point $x \in$ Pos(Q), let $\varphi = \| q - x \|, \mathbf{S} = \{y : \| q - y \| \leqq \varphi\}, \mathbf{\Gamma} = \{\lambda : Q\lambda \in \mathbf{S}\}$. For all $\lambda \notin \mathbf{\Gamma}, \| q - Q\lambda \|^2 > \varphi^2$ and hence $f(\lambda, \mu) > \varphi^2$. Let $\overline{\lambda} = Q^{-1}x$, then $\overline{\lambda} \geqq 0$ since $x \in$ Pos(Q), and hence $f(\overline{\lambda}, \mu) = \varphi^2$; and $\overline{\lambda} \in \mathbf{\Gamma}$. $\mathbf{\Gamma}$ is a compact set and $f(\lambda, \mu)$ is a continuous function in λ, so it attains its minimum over $\mathbf{\Gamma}$, at a point $\tilde{\lambda} \in \mathbf{\Gamma}$ say. By the above $f(\tilde{\lambda}, \mu) \leqq \varphi^2$ and since $f(\lambda, \mu) > \varphi^2$ for all $\lambda \notin \mathbf{\Gamma}$, $\tilde{\lambda}$ is the unconstrained minimizer of $f(\lambda, \mu)$ over $\lambda \in \mathbf{R}^n$. Since $f(\lambda, \mu)$ is strictly convex in λ, its unconstrained minimum $\tilde{\lambda}$ is unique and is attained at the solution of $g(\lambda, \mu) = 0$. Hence, for any $\mu > 0$, $g(\lambda, \mu) = 0$ has a unique solution in λ, $\lambda(\mu)$ say.

$\frac{\partial g(\lambda, \mu)}{\partial \lambda} = H(\lambda, \mu)$ is nonsingular for all λ whenever $\mu > 0$. Hence, by the implicit function theorem, $\lambda(\mu)$ is continuous in μ in the region $\mu > 0$.

If x^* is the nearest point in Pos(Q) to q, and $\lambda^* = Q^{-1}x^*$, then from penalty function theory we know that $\lambda(\mu) \to \lambda^*$ as $\mu \to 0$ through positive values. This and the continuity of $\lambda(\mu)$ implies that $\|\lambda(\mu)\|$ remains bounded above by a constant as $\mu \to 0$ through positive values, where this constant depends on $\|\lambda^*\|$ which itself depends on q and Q. ∎

From Theorem 1 we know that $\{\lambda(\mu) : \mu > 0\}$ is a path in the λ-space which we call the *exterior path.* For given $\epsilon > 0$, $\mu > 0$ define

$$\Omega(\mu, \epsilon) = \{\lambda : \|g(\lambda, \mu)\| \leqq \epsilon\} \tag{8}$$

The set $\cup_{\mu>0}\Omega(\mu, \epsilon)$ defines an *envelope around the exterior path.* We will show that it is possible to interpret the points obtained in our algorithm, as a sequence of points contained in this envelope, converging to λ^* as $\mu \to 0$.

Lemma 1: For given $\epsilon > 0$, the set $\Omega(\mu, \epsilon)$ is bounded for each $\mu > 0$.

Proof: Fix $\mu > 0$. Since $f(\lambda, \mu)$ is strongly convex in λ, for all λ, $\lambda^1 \in \mathbf{R}^n$, there exists a positive constant γ_0 such that $\|g(\lambda, \mu) - g(\lambda^1, \mu)\| \geqq \gamma_0\|\lambda - \lambda^1\|$ (see [19]). Substituting $\lambda^1 = \lambda(\mu)$ in this inequality yields $\|g(\lambda, \mu)\| \geqq \gamma_0\|\lambda - \lambda(\mu)\|$ for all λ. Hence, if $\lambda \in \Omega(\mu, \epsilon)$ we have $\|\lambda - \lambda(\mu)\| \leqq \frac{\epsilon}{\gamma_0}$. The result in this lemma now follows by using Theorem 1. ∎

From Lemma 1, we can establish a bound for $\lambda \in \Omega(\mu, \epsilon)$ which depends only on μ, ϵ, and the input data. Let $\gamma(\mu, \epsilon)$ denote this bound, i.e., $\|\lambda\| \leqq \gamma(\mu, \epsilon)$ for all $\lambda \in \Omega(\mu, \epsilon)$. The algorithm will be initiated with a value for μ, $\mu_0 > 0$, and the value of μ will be decreased in each iteration, until it reaches a target value μ_* say. The algorithm will terminate when μ reaches or decreases below μ_*. Define $\gamma = \max\{\gamma(\mu, \epsilon) : \mu_* \leqq \mu \leqq \mu_0\}$. Then $\|\lambda\| \leqq \gamma$ for all $\lambda \in \Omega(\mu, \epsilon)$, and for all $\mu_* \leqq \mu \leqq \mu_0$; and this interval itself is called *the range for the penalty parameter.* γ depends on ϵ, and clearly it decreases with ϵ.

Theorem 2: For given $\epsilon > 0$, and $\mu > 0$ in its range , let $\lambda \in \Omega(\mu, \epsilon)$, i.e., $\|g(\lambda, \mu)\| \leqq \epsilon$. Let ρ be a number satisfying $0 < \rho < 1$ and $\bar{\mu} = \rho\mu$. Then there exists a constant β depending only on the input data, and the bound γ defined above, such that $\|g(\lambda, \bar{\mu})\| \leqq \frac{1-\rho}{\rho}\beta + \frac{\epsilon}{\rho}$.

Proof: Let $a = -2(q - Q\lambda)^T Q$ and $b = 4(\delta_1\lambda_1^3, \cdots, \delta_n\lambda_n^3)$. Then $g(\lambda, \bar{\mu}) = a + \frac{b}{\bar{\mu}}$. So $\|g(\lambda, \bar{\mu})\| = \|a + \frac{b}{\rho\mu}\| = \frac{1}{\rho}\|(\rho - 1)a + a + \frac{b}{\mu}\| \leqq \frac{(1-\rho)}{\rho}\|a\| + \frac{1}{\rho}\|a + \frac{b}{\mu}\| \leqq \frac{1-\rho}{\rho}\|a\| + \frac{\epsilon}{\rho}$. Now $\|a\| = \| - 2(q - Q\lambda)^T Q\| \leqq 2\|q^T Q\| + 2\|Q^T Q\| \cdot \|\lambda\| \leqq 2\|q^T Q\| + 2\|Q^T Q\|\gamma = \beta$, by the definition of γ given above. So $\|g(\lambda, \bar{\mu})\| \leqq \frac{(1-\rho)}{\rho}\beta + \frac{\epsilon}{\rho}$. ∎

Theorem 3: For given $\epsilon > 0$, and $\mu > 0$ in its range, let $\lambda \in \Omega(\mu, \epsilon)$. Select a ρ satisfying $\frac{1}{2} < \rho < 1$, and let $\bar{\mu} = \rho\mu$, $\bar{\lambda} = \lambda - (H(\lambda, \bar{\mu}))^{-1}(g(\lambda, \bar{\mu}))^T$. Then $\|g(\bar{\lambda}, \bar{\mu})\| \leq \bar{\alpha}\|g(\lambda, \bar{\mu})\|^2$

where $\bar{\alpha}$ is a constant depending on ρ, ϵ, γ, β, μ, and σ_1 defined earlier.

Proof: Since the function $g(\cdot, \bar{\mu})$ is twice continuously differentiable in λ, from Taylor's theorem we have

$$\begin{aligned} g_j(\bar{\lambda}, \bar{\mu}) = \; & g_j(\lambda, \bar{\mu}) + (\bar{\lambda} - \lambda)^T H_{\cdot j}(\lambda, \bar{\mu}) \\ & + \tfrac{1}{2}(\bar{\lambda} - \lambda)^T W(\tilde{\lambda}, \bar{\mu})(\bar{\lambda} - \lambda) \end{aligned} \tag{9}$$

where $W(\tilde{\lambda}, \bar{\mu})$ is the square matrix $\left(W_{u,v}(\tilde{\lambda}, \bar{\mu})\right)$ of order n with $W_{u,v}(\tilde{\lambda}, \bar{\mu}) = 0$ for all $(u, v) \neq (j, j)$, and $= \frac{24}{\bar{\mu}}\tilde{\delta}_j \tilde{\lambda}_j$ for $(u, v) = (j, j)$; $\tilde{\lambda} = (\tilde{\lambda}_j) = (1 - \theta)\lambda + \theta\bar{\lambda}$ for some $0 \leqq \theta \leqq 1$, and $\tilde{\delta}_j = \delta_j(\tilde{\lambda}_j)$ as defined in (3) for $r = 4$. From the definition of $\bar{\lambda}$ we verify that $g_j(\lambda, \bar{\mu}) + (\bar{\lambda} - \lambda)^T H_{\cdot j}(\lambda, \bar{\mu}) = 0$. So from (9)

$$\begin{aligned} g_j(\bar{\lambda}, \bar{\mu}) &= \frac{12}{\bar{\mu}}(\bar{\lambda}_j - \lambda_j)^2 \tilde{\delta}_j \tilde{\lambda}_j \\ \|g(\bar{\lambda}, \bar{\mu})\|^2 &= (\frac{12}{\bar{\mu}})^2 \sum_{j=1}^{n} (\bar{\lambda}_j - \lambda_j)^4 \tilde{\delta}_j^2 \tilde{\lambda}_j^2 \\ &\leqq (\frac{12}{\bar{\mu}})^2 \sum_{j=1}^{n} (\bar{\lambda}_j - \lambda_j)^4 \tilde{\lambda}_j^2 \end{aligned} \tag{10}$$

From the definition of $\bar{\lambda}$, $\|\bar{\lambda}\| \leqq \|\lambda\| + \| \left(H(\lambda, \bar{\mu})\right)^{-1} \| \cdot \|g(\lambda, \bar{\mu})\| \leq \gamma + \eta$ where γ is the constant defined earlier and $\eta = \frac{1}{2\sigma_1}\left(\frac{1-\rho}{\rho}\beta + \frac{\epsilon}{\rho}\right)$ from Lemma 1 and Theorem 2. So $\|\bar{\lambda}\|^2 \leqq (\gamma + \eta)^2$. Since $\tilde{\lambda} = (\tilde{\lambda}_j) = (1 - \theta)\lambda + \theta\bar{\lambda}$ for some $0 \leqq \theta \leqq 1$, $\tilde{\lambda}_j^2 \leqq \max \cdot \{\lambda_j^2, \bar{\lambda}_j^2\} \leqq \max \cdot \{\|\lambda\|^2, \|\bar{\lambda}\|^2\} \leqq (\gamma + \eta)^2$. Substituting this in (10) yields

$$\|g(\bar{\lambda}, \bar{\mu})\|^2 \leqq \left(\frac{12(\gamma + \eta)}{\bar{\mu}}\right)^2 \sum_{j=1}^{n} (\bar{\lambda}_j - \lambda_j)^4 \tag{11}$$

Now, $\sum_{j=1}^{n}(\bar{\lambda}_j-\lambda_j)^4 \leqq \left(\sum_{j=1}^{n}(\bar{\lambda}_j-\lambda_j)^2\right)^2 = (\|\bar{\lambda}-\lambda\|^2)^2 \leqq (\|(H(\lambda,\bar{\mu}))^{-1}\|^2 \cdot \|g(\lambda,\bar{\mu})\|^2)^2 \leqq (\frac{1}{4\sigma_1^2})^2\|g(\lambda,\bar{\mu})\|^4$. Substituting in (11) we get

$$\|g(\bar{\lambda},\bar{\mu})\| \leqq \frac{3(\gamma+\eta)}{\bar{\mu}\sigma_1^2}\|g(\lambda,\bar{\mu})\|^2 \tag{12}$$

Now, $\eta = \frac{1}{2\sigma_1}\left(\frac{1-\rho}{\rho}\beta + \frac{\epsilon}{\rho}\right) \leq \frac{1}{2\sigma_1}(\beta+2\epsilon)$, since $\frac{1}{2} < \rho < 1$. So, if we take $\bar{\alpha} = \frac{3}{\bar{\mu}\sigma_1^2}\left(\gamma + \frac{\beta+2\epsilon}{2\sigma_1}\right)$, then from (12) $\|g(\bar{\lambda},\bar{\mu})\| \leqq \bar{\alpha}\|g(\lambda,\bar{\mu})\|^2$. ∎

We will now formally describe the detailed steps in the algorithm and show that there exists a value for the factor ρ strictly < 1 which ensures that all the points λ obtained in the algorithm are within the envelope around the exterior path mentioned earlier, and that the algorithm terminates in a finite number of steps. Let $\hat{\epsilon}$ be the desired accuracy and $\hat{\mu}$ the specified target value for the penalty parameter. The aim is to find λ which satisfies $\|g(\lambda,\mu)\| \leqq \epsilon$ for some $\mu \leqq \hat{\mu}$, $\epsilon \leqq \hat{\epsilon}$.

Let $\lambda^0 = Q^{-1}q$. If $\lambda^0 \geqq 0$, $q \in \text{Pos}(Q)$, hence it is itself the nearest point in $\text{Pos}(Q)$, and λ^0 is optimal to (1), terminate. If $\lambda^0 \ngeqq 0$, define

$$\mu_0 = \frac{4}{\hat{\epsilon}}\sqrt{\Sigma\left((\lambda_j^0)^6 : \text{ over } j \text{ such that } \lambda_j^0 < 0\right)}$$

Define $\gamma = \max\cdot\{\gamma(\mu,\hat{\epsilon}) : \hat{\mu} \leqq \mu \leqq \mu_0\}$ as before. Define $\beta = 2\|q^TQ\| + 2\|Q^TQ\|\gamma$ as before, using this value for γ. Define

$$\begin{aligned}
\hat{\alpha} &= \max\cdot\{\tfrac{1}{2\hat{\epsilon}},\ \tfrac{3}{\hat{\mu}\sigma_1^2}\left(\gamma + \tfrac{\beta+2\hat{\epsilon}}{2\sigma_1^2}\right)\} \\
\epsilon^* &= \tfrac{1}{2\hat{\alpha}} \\
\mu_* &= \tfrac{3}{\hat{\alpha}\sigma_1^2}\left(\gamma + \tfrac{\beta+2\hat{\epsilon}}{2\sigma_1^2}\right) \\
\rho^* &= \tfrac{(\beta+\frac{1}{2\hat{\alpha}})}{(\beta+\frac{1}{\hat{\alpha}\sqrt{2}})}
\end{aligned} \tag{13}$$

Since $\hat{\alpha} \geqq \frac{1}{2\hat{\epsilon}}$, $\hat{\epsilon} \geqq (\frac{1}{2\hat{\alpha}}) = \epsilon^*$. Similarly, we can verify that $\hat{\mu} \geqq \mu_*$, $\rho^* < 1$, and that $\|g(\lambda^0, \mu_0)\| = \hat{\epsilon}$.

The Algorithm

Initialization Initiate the algorithm with the pair (λ^0, μ_0).

General Step Let (λ^k, μ_k) be the pair from the end of the previous step. Define $\mu_{k+1} = \rho^* \mu_k$. Take one Newton step from λ^k for $f(\lambda, \mu_{k+1})$, i.e.

$$\lambda^{k+1} = \lambda^k - \left(H(f(\lambda^k, \mu_{k+1}))\right)^{-1} \left(g(\lambda^k, \mu_{k+1})\right)^T$$

If $\mu_{k+1} \leq \mu_*$, λ^{k+1} is a point within desired tolerance of the optimum solution of (1), terminate. Otherwise, with λ^{k+1}, μ_{k+1} as the new pair, go to the next step.

We will now show that every pair (λ^k, μ_k) obtained in this algorithm satisfies $\|g(\lambda^k, \mu_k)\| \leq \hat{\epsilon}$. Suppose this inequality holds for $k = p$. From the definition of $\hat{\alpha}$ in (13), we see that $\hat{\alpha} \geqq$ the constant $\bar{\alpha}$ of Theorem 3 for $\epsilon = \hat{\epsilon}$, this leads to $\|g(\lambda^{p+1}, \mu_{p+1})\| \leqq \hat{\alpha}\|g(\lambda^p, \mu_{p+1})\|^2 \leqq \hat{\alpha}\left(\frac{1-\rho^*}{\rho^*}\beta + \frac{\hat{\epsilon}}{\rho^*}\right)^2$ since $\|g(\lambda^p, \mu_p)\| \leqq \hat{\epsilon}$ and by using Theorem 2. So

$$\begin{aligned}
\|g(\lambda^{p+1}, \mu_{p+1})\| &\leqq \tfrac{\hat{\alpha}}{(\rho^*)^2}\left((1-\rho)^*\beta + \epsilon^*\right)^2 \\
&= \tfrac{1}{2\epsilon^*(\rho^*)^2}\left((1-\rho)^*\beta + \epsilon^*\right)^2 \\
&= \tfrac{(\epsilon^*)^2}{2\epsilon^*(\rho^*)^2}\left(\tfrac{(\sqrt{2}-1)\beta}{\beta+\sqrt{2}\epsilon^*} + 1\right)^2 \\
&= \tfrac{\epsilon^*}{2(\rho^*)^2}\left(\tfrac{\sqrt{2}(\beta+\epsilon^*)}{\beta+\sqrt{2}\epsilon^*}\right)^2 \\
&= \tfrac{\epsilon^*}{(\rho^*)^2}\left(\tfrac{\beta+\frac{1}{2\hat{\alpha}}}{\beta+\frac{1}{\sqrt{2}\hat{\alpha}}}\right)^2 \\
&= \epsilon^* \leqq \hat{\epsilon}
\end{aligned}$$

Since $\|g(\lambda^0, \mu_0)\| \leqq \hat{\epsilon}$, by induction $\|g(\lambda^k, \mu_k)\| \leqq \hat{\epsilon}$ for all k.

Since $\rho^* < 1$, the value of the penalty parameter decreases strictly in the algorithm. So, after at most $\lceil(\log \mu_* - \log \mu_0)/(\log \rho^*)\rceil$ steps of the algorithm, the value of the penalty parameter decreases to μ_* or below, i.e. to $\leqq$ the target value of $\hat{\mu}$. We terminate with the vector λ at that time as the point near to the optimum solution of (1) to the desired accuracy.

3 Computational Results

We have so far solved several randomly generated nearest point problems in simplicial cones of dimension n ranging from 10 to 1500 with implementations of these exterior point methods. In these problems, each element of q was generated from the uniform distribution between -5 and +5 (of double word length of 8 bytes) and each element of Q was drawn from the uniform distribution between -20 and +20. The vector q and the matrix Q were fully dense in all the problems generated. None of the matrices Q was checked for singularity, but system (5) always had a solution in methods which used it to generate descent directions. Even though our convergence results have been proved for the standard Newton method using the composite function $f_4(\lambda, \mu)$, in our computational tests we tried a variety of methods to compare their performance. The various methods tried are listed below.

VERSION 1: **Using Newton direction with step lengths 1**: In this version, exactly one descent step is carried out following a decrease in the penalty parameter in each iteration. $f_3(\lambda, \mu)$ and $f_4(\lambda, \mu)$ both have second derivatives at all λ but $f_2(\lambda, \mu)$ has second derivatives only at points λ in which all components are nonzero. It has been observed that all the λ^k generated in the algorithm tended to have all components nonzero, even though some of the components were clearly converging to zero (as explained in [15], this might be due to the fact that exterior methods approach the optimum from outside the feasible region). So, we implemented this version also with the composite function $f_2(\lambda, \mu)$ using the formula for the hessian given in (3) and the method never encountered any problems in thousands of runs in early experiments.

VERSION 2: **Using Newton direction with line searches**: Unlike Version 1, here we implemented this version only using the composite function $f_2(\lambda, \mu)$, because of the simplicity of the line search routine for $r = 2$. Again, only one descent step was carried out per iteration.

VERSION 3: **Using steepest descent strategy**: We implemented this version using the composite functions $f_2(\lambda, \mu)$ and also $f_1(\lambda, \mu)$. Although $f_1(\lambda, \mu)$ is not differentiable at points λ with some components zero, the previous discussion on second order differentiability applies here. Unfortunately, this version based on $f_1(\lambda, \mu)$ did not converge in most cases. A close look at the gradient reveals that this divergence is expected since the gradient does not carry much information about negative λ_js. In this version we also experimented with carrying out several descent steps per iteration (i.e., several descent steps between changes in the value of the penalty parameter).

Our convergence theory showed that there exists a value for the factor ρ strictly < 1 (this is ρ^* defined in (13)) which ensures that Version 1 with $r = 4$ (i.e., with the composite function $f_4(\lambda, \mu)$) terminates with a desired point in a finite number of iterations. To actually implement the methods, we experimented with several trial values for ρ and pursued that one which seemed to give the best performance . This is a very standard practice in implementing optimization algorithms. In the same vein, in our computational studies we experimented with all composite functions $f_r(\lambda, \mu)$ for $r = 2, 3, 4$, even though our convergence analysis focussed on the composite function $f_4(\lambda, \mu)$, since the current proof depended on the twice continuous differentiability of the gradient. We got significantly better computational results with $r = 2$. In most mathematical programming implementations this phenomenon is typical.

For all versions, the following criteria have been tried as signals for convergence in early experimentation. Small positive tolerances ϵ_1, ϵ_2 have been selected, and the algorithm is terminated in iteration k if the penalty parameter μ_k and the vector λ^k satisfied both the following conditions:

$$\text{(i)} \quad \lambda_j^k \geq -\epsilon_1 \ , \quad \text{for all} \ \ j = 1 \ \ \text{to} \ \ n$$

$$\text{(ii)} \quad \frac{1}{\mu_k} P_r(\lambda^k) < \epsilon_2$$

Condition (i) relates to "near feasibility" (within the tolerance ϵ_1). Both conditions have been used in early experiments, and it always turned out that when one held the other too held in the same iteration (depending on the choice of suitable values for ϵ_1 and ϵ_2). Hence, in later tests, we used condition (i) as the sole termination criterion.

In Figure 1 we provide the plot of $\sum(|\lambda_j| :$ over j such that $\lambda_j < 0)$, which is a measure of infeasibility of the current λ-vector, in each iteration of the standard Newton vesion based on $f_2(\lambda, \mu)$, for a problem of dimension $n = 40$. It can be seen that this infeasibility measure drops very sharply and becomes almost zero in about 6 iterations.

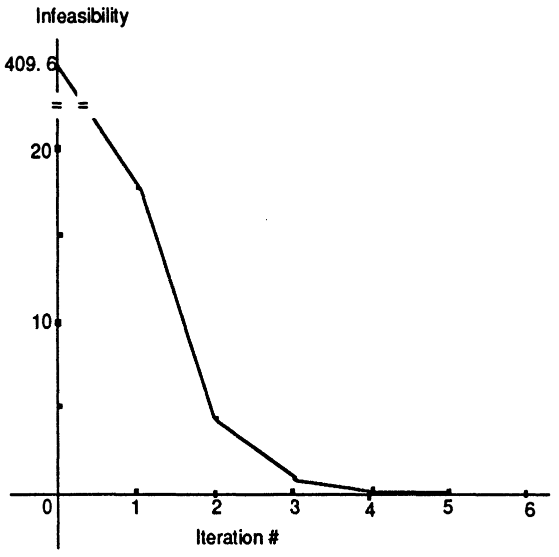

Figure 1 Infeasibility versus # iterations

For the algorithm to converge, we observed that the penalty parameter μ must reach a sufficiently small value (like 10^{-12}, this depends on the desired accuracy ϵ_1). A lot of experimentation has been done to determine the best value for μ_0, the initial penalty parameter value. The smaller the value of μ_0, the less the number of iterations to drive μ to its terminal value. However, since the initial λ, $\lambda^0 = Q^{-1}q$ is the unconstrained minimum of $f_r(\lambda, \mu)$ only at $\mu = \infty$, choosing a small value for μ_0 puts the unconstrained minimum of $f_r(\lambda, \mu_0)$ far away from λ^0, resulting in slow convergence. In most experiments, selecting $\mu_0 = 10^{-2}$ yielded excellent results.

In Versions 1 and 2, computationally the most expensive part in each iteration is solving a system of equations ((5) here) to get the descent direction. Since many optimization algorithms require a step like this in every iteration, this has been the object of intense research activity over the years. For our computational runs we experimented with several schemes and found that the iterative SOR methods gave the best results, however the best value for the relaxation paramter ω^* in these methods seems to increase with n (see Table 2). The performance of Version 1, 2 can be improved substantially by using better schemes to solve (5) in each iteration.

All versions were coded in FORTRAN 77 using double precision arithmetic and run on an IBM 3033, or on an APOLLO series 4000. Problems with $n > 700$ were run on the Cornell Supercomputer (IBM 3090).

In Version 1 based on Newton directions and constant step length of 1, it can be seen from Table 1 that the number of iterations grows extremely slowly, if at all, with the problem dimension. As mentioned earlier, there is tremondous scope for improving the computer time taken for solving (5) in each iteration, hence the number of iterations is a more reliable guide of algorithm performance, than computer time, and this is almost independent of problem dimension. This almost constant number of iterations was about 6 for versions based on $f_2(\lambda, \mu)$, 38 for versions based on $f_3(\lambda, \mu)$, and 70 for versions based on $f_4(\lambda, \mu)$. The main reason for this may be the fact that $f_2(\lambda, \mu)$ is very nearly quadratic in λ , thus making it well suited for Newton method. Since $f_2(\lambda, \mu)$ is almost quadratic, one Newton step almost always leads very close to the unconstrained minimum of this function in each iteration, thus enabling a much faster reduction in the value of μ. The barrier terms based on logarithmic functions employed by interior point methods do not share this nice property.

In Version 2 based on Newton's method with line searches, the step length ranged between 0.8 to 1.2, and was very close to 1 in most iterations. The total number of iterations was only marginally less than that for Versions 1 based on constant step length of 1.

We compared the performance of Version 1 with M.J.D. Powell's implementation of A. Idnani and D. Goldfarb's dual quadratic programming algorithm (available through IMSL as subroutine QPROG), K. Haskell and R. Hanson's [10] routine available through ACM algorithm 587 for linearly constained least squares (called HH in Table 2) , and with our implementation of D.R. Wilhelmsen's nearest point algorithm [22]. Our algorithm was superior to each of these, but we display comparative figures only for K. Haskell and R. Hanson's code and QPROG to conserve space (comparative timings for the other algorithm can be obtained from the authors).

In Version 1 we found that the total number of iterations depends critically on the value of the factor ρ used. We found that once a λ_j becomes > 0, it remains > 0 in almost all subsequent iterations. This may explain the excellent performance.

For Version 3 based on steepest descent directions, we found that it is better to do many descent moves between consecutive updates of the penalty parameter. In each iteration we continue making descent moves as long as there is significant change in the solution vector λ. When this becomes small by the L_∞ norm we update μ and go to the next iteration. In comparing Version 3 with those based on Newton directions one should bear in mind that each move in these methods is computationally cheaper as there is no equation solving involved.

Version 3 is more sensitive than Newton based versions to the strategy for updating the penalty parameter. In general, the rate of reduction of μ has to be slower, more so as n increases. The number of descent moves per iteration was almost constant at 15 independent of problem dimension. The number of iterations itself averaged around 37 in problems with n up to 250. From Tables 1, 2, and 3 we see that our current implementation of Version 3 does not compare favorably with the implementations of Newton based versions, or even with QPROG, in terms of computer time. However, there is tremendous scope to improve the coding of this version. There is also the possibility of using conjugate gradient based directions rather than steepest descent directions, this has not been tested yet.

It has already been mentioned earlier that if λ_j becomes > 0, it tended to remain > 0 in subsequent iterations. In this problem, if we know the set $\mathbf{J} = \{j : \lambda_j > 0$ in the optimum solution$\}$, it is well known that the nearest point itself can be found by orthogonally projecting q onto the linear hull of the face of $\mathbf{K}$ corresponding to $\mathbf{J}$. In some experiments we selected a tolerance $\epsilon_3 > 0$ (about 10^{-3}) and when the current solution vector λ^k satisfied $\lambda_j^k \geqq -\epsilon_3$ for all j, we have taken $\{j : \lambda_j^k > -\epsilon_3\}$ as an estimate for $\mathbf{J}$ and used the projection strategy. This has cut the number of iterations by about 1/3 on an average.

Table 1 : Results with Version 1(time in IBM 3033 seconds)

based on $f_r(\lambda, \mu)$

n	# problems	$r = 2$		$r = 3$		$r = 4$		
		*	time	*	time	*	time	time for QPROG
10	200	5.80	0.65	37.40	.710	69.50	0.77	0.65
20	200	6.01	0.68	38.00	.877	70.10	1.07	0.68
30	200	6.03	0.72	38.10	1.170	70.50	1.62	0.73
40	200	6.04	0.80	38.13	1.610	70.60	2.44	0.82
50	200	6.04	0.89	38.40	2.240	70.79	3.59	0.96
100	100	6.08	1.98	39.10	9.270	71.20	16.35	2.88
700	1	7.00	339.70	+	+	+	+	652.00

Accuracy = 10^{-8}. System (5) solved by direct factorization using IMSL routines LFCSF and LSLSF.

*Average number of iterations.

+Not tried

Best $\rho = .02, .30, .40$ for $r = 2, 3, 4$ respectively.

Table 2 : Results with Version 1 based on $f_2(\lambda, \mu)$ on APOLLO series 4000. Time in seconds.

n	# problems	Avg. # of iterations	time	optimal ρ	ω^*	time for HH
10	10	6.0	6.12	.02	1.02	6.07
50	10	6.0	4.05	.02	1.05	4.50
100	10	6.0	24.0	.02	1.25	34.0
200	10	6.1	150.0	.02	1.30	259.6
300	10	6.3	405.1	.02	1.30	890.0
400	10	6.3	902.3	.02	1.30	2123.0
700	2	6.5	4302.0	.02	1.45	11768.0

Accuracy = 10^{-7}. System (5) solved by SOR with relaxation parameter ω^* (found best by experimentation). In all cases, $\sqrt{n}$ SOR iterations gave an accurate solution .

Problems with $700 < n \leqq 1500$ were run on a different computer (IBM 3090) and for these the average number of iterations was also 6. 5. But we do not show those results in the table because the times on the computers are not comparable, and the algorithm HH was not available on that system.

Table 3 : Results with Version 3 (time in IBM 3033 seconds)

based on $f_2(\lambda, \mu)$

n	#	Avg. # of		time	optimal	time for
	problems	iterations	descent steps per iteration		ρ	QPROG
10	200	12	14.66	1.807	.1	.565
20	200	12	15.16	1.170	.1	.670
30	200	35	14.90	3.760	5	.727
40	200	36	15.56	6.260	.5	.815
50	200	36	13.38	8.080	.5	.960
100	100	36	15.58	32.90	.5	2.33
250	1	37	14.90	197.30	.5	31.7

Accuracy = 10^{-3}.

4 Exterior Penalty Methods for Nearest Point Problems in Nonsimplicial Cones and Convex Quadratic Programs.

Consider the problem of finding the nearest point in Pos(Q) to $q \in \mathbf{R}^n$ where Q is an $n \times m$ matrix of rank n, with $m > n$. This is the problem of the same form as (1) with the present Q, however an optimum λ in this case may not be unique. If λ^* is an optimum solution, then $x^* = Q\lambda^*$ is the nearest point in Pos(Q) to q, x^* is always unique. We use the same composite functions $f_r(\lambda, \mu)$ as before, for $r = 2, 3$, or 4 and the formulas for $g_r(\lambda, \mu)$, and $H(f_r(\lambda, \mu))$ are the same as before. However, in this case the hessian $H(f_r(\lambda, \mu))$ of order $m \times m$ is PSD and may be singular, and (5) may not have a solution. In this case, we can use some of the standard modifications in the literature for defining the Newton search direction, one of which replaces (5) by

$$\Big(H(f_r(\lambda^k, \mu)) + \tau I\Big)\, y = - \nabla f_r(\lambda^k, \mu) \tag{14}$$

where $\tau > 0$, and I is the unit matrix of order m. The matrix on the left-hand side of (14) is PD and hence (14) again has a unique solution which is a descent direction for $f_r(\lambda, \mu)$ at λ^k.

The steepest descent direction, and the line search routine for $f_r(\lambda, \mu)$ discussed earlier, remain valid as they are, in this case. So, all the exterior point methods discussed earlier for the simplicial cone case; can be implemented to handle this case directly, with only minor modifications. We conducted some computational experiments in which Q was rectangular, and our results were almost the same as those reported earlier.

Now consider the general convex quadratic program

$$\begin{aligned} \text{minimize } z(x) &= cx + \tfrac{1}{2} x^T D x \\ \\ \text{subject to } Ax &\geqq b \\ x &\geqq 0 \end{aligned}$$

where D is a symmetric PSD matrix and A is of order $m \times n$. In the penalty approach we solve this problem through the unconstrained minimization problem of the form

$$\text{minimize } w_r(x,\mu) = cx + \tfrac{1}{2}x^T Dx + \tfrac{1}{\mu}\textstyle\sum_{i=1}^{m}(\max\{0, b_i - A_{i.}x\})^r$$

$$+\tfrac{1}{\mu}\textstyle\sum_{j=1}^{n}(\max\{0, -x_j\})^r$$

$$\text{over } x \in \mathbf{R}^n$$

Using the function $w_r(x,\mu)$, versions of the new exterior penalty algorithms for this problem are constructed exactly as before.

5 Acknowledgements

We acknowledge the many discussions we had with R. Saigal. Our thanks to O.L. Mangararian for bringing the result on strongly convex functions to our attention which greatly simplified the proof of Lemma 1. Finally, we are grateful to an anonymous referee for pointing out errors in an earlier version of this paper.

6 References

1. K.S. Al-Sultan, "Nearest Point Problems: Theory and Algorithms," Ph.D. Dissertation, University of Michigan (Ann Arbor, MI, 1990).

2. K.S. Al-Sultan and K.G. Murty, "Nearest Points in Nonsimplicial Cones and LCPs with PSD Symmetric Matrices," in S. Kumar (ed.) *Recent Developments in Mathematical Programming*, Australian Society for O.R. special Publication, Gordon & Greach, Camberwell, Victoria, Australia, 1991.

3. M.S. Bazaraa and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, (Wiley, NY, 1979).

4. S.Y. Chang and K.G. Murty, "The Steepest Descent Gravitational method for Linear Programming," *Discrete Applied Mathematics*, 25 (1989) 211-239.

5. M.B. Daya and C.M. Shetty, "Polynomial Barrier Function Algorithms for Convex Quadratic Programming," *Arabian Journal for Science and Engineering*, 15, no. 4B (October 1990) 658-670.

6. J.P. Evans, F.J. Gould, and J.W. Tolle, "Exact Penalties Functions in Nonlinear Programming," *Mathematical Programming*, 4, 2 (1973) 72-97.

7. A.V. Fiacco and G.P. McCormich, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, (Wiley, NY, 1968).

8. E.G. Gilbert, D.W. Johnson, and S.S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," *IEEE Journal of Robotics and Automation*, 4, 2 (1988) 193-203.

9. G.H. Golub and C.F. Van Loan, *Matrix Computations*, (John Hopkins University Press, Baltimore, MD, 1989).

10. K. Haskell and R. Hanson, "An Algorithm for Linear Least Squares Problems with Equality and Nonnegativity Constraints," *Mathematical Programming*, 21 (1981) 98-118.

11. N. Karmarkar, "A New Polynomial Algorithm for Linear Programming," *Combinatorica*, 4 (1984) 373-395.

12. M. Kojima, S. Mizuno, and A. Yoshise, "A Polynomial-Time Algorithm for a Class of Linear Complementarity Problems," *Mathematical Programming*, 44, 1 (1989) 1-26.

13. C.E. Lemke, "Bimatrix Equilibrium Points and Mathematical Programming," *Management Science*, 11 (1965) 681-689.

14. C.E. Lemke, “On Complementary Pivot Theory” in G.B. Dantzig and A. Veinott (eds) *Mathematics of the Decision Sciences* (1968).

15. D.G. Luenberger, “Linear and Nonlinear Programming,” 2nd edition (Addison-Wesley, Melo Park, CA, 1984).

16. K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*, (Heldermann Verlag, West Berlin, 1988).

17. K.G. Murty and Y. Fathi, “A Critical Index Algorithm for the Nearest Point Problem on Simplicial Cones,” *Mathematical Programming*, 23 (1982) 206-215.

18. T. Pietrzykowski, “An Exact Potential Method for Constrained Maxima,” *SIAM J. Numerical Analysis*, 26, 2 (1968) 299-304.

19. B.T. Polyak, “Introduction to Optimization,” (Optimization Software, Inc., NY, 1987).

20. J. Renegar, “A Polynomial-Time Algorithm Based on Newton's Method for Linear Programming,” *Mathematical Programming*, 40, 1 (1988) 59-95.

21. K. Truemper, “Note on Finite Convergence of Exterior Functions,” *Management Science*, 21, 5 (1975) 600-606.

22. D.R. Wilhelmsen, “A Nearest Point Algorithm for Convex Polyhedral Cones and Applications to Positive Linear Approximations,” *Mathematics of Computation*, 30 (1976) 48-57.

23. P. Wolfe, “Finding Nearest Point in a Polytope,” *Mathematical Programming*, 11 (1976) 128-149.

24. Y. Ye and E. Tse, “An Extension of Karmarkar's Projective Algorithm for Convex Quadratic Programming,” *Mathematical Programming*, 44, 2 (1989) 157-181.