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

October, 1989
Technical Report 89-31

*Partially supported by NSF grant No. ECS-8521183

Abstract

We consider the problem of finding the nearest point (by Euclidean distance) in a convex polyhedral cone to a given point, and develop an exterior penalty algorithm for it. Each iteration in the algorithm consists of a single Newton step followed by 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 this problem and for convex quadratic programs, all based on a single descent step followed by 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 $K \subset R^n$, to a point $q \in R^n$. It is a special case of a convex quadratic program with many important applications. In one class of applications in remote sensing we need to find the composition of a mixture of various pure constituents. Signals are obtained from the mixture and each of the pure constituents, and the problem of estimating the composition of the mixture from this data can be posed as a nearest point problem using the least squares formulation. Other applications of the nearest point problem in constructing approximations of functions have been discussed by D.R. Wilhelmsen [1976]. The nearest point problem also appears as a subproblem in each step of the gravitational method for solving linear programs (S.Y. Chang and K.G. Murty [1988]). Also, the nearest point problem has important applications to a variety of problems in robotics (E. G. Gilbert, D. W. Johnson and S. S. Keerthi [1988]).

There is a linear complementarity problem (*LCP*) associated with the nearest point problem, it consists of the *KKT* conditions that provide the necessary and sufficient optimality conditions for it. The nearest point problem can be solved through the associated *LCP* using any of the available algorithms for solving *LCPs* such as Lemke's complementary pivot algorithm, or the principal pivoting methods, etc. (C. Lemke [1965], R. W. Cottle and G. B. Dantzig [1968], K. G. Murty [1988]), or the recently developed polynomial time interior point methods (M. Kojima [1989], M. Kojima, S. Mizuno and A. Yoshise [1989]). The ellipsoid algorithm of linear programming has been extended to solve the nearest point problem in polynomial time, but it is only of theoretical interest so far, as its computational performance is poor (S. J. Chung and K. G. Murty [1981]). Special algorithms for the nearest point problem, based on its geometry are discussed in (E. G. Gilbert [1966], R. O. Barr and E. G. Gilbert [1969], K. G. Murty and Y. Fathi [1982], D. R. Wilhelmsen [1976], P. Wolfe [1976]). The nearest

point problem can also be solved by specializations of any of several descent methods or active set methods for linearly constrained nonlinear programs discussed in the nonlinear programming literature. Thus, a variety of methods are already available for solving the nearest point problem, and some of these are practically efficient.

But the nearest point problem appears in many practical applications, and often as a large scale problem. This has provided a continuing motivation for the development of new algorithms which can solve large scale problems faster. This is also the motivation for the work reported in this paper. Our inspiration comes from recent reports of significant improvements in computational performance for solving large scale linear programs and convex quadratic programs through the use of barrier methods in novel ways. In this paper, we develop several penalty methods for the nearest point problem, and report on their encouraging computational performance.

If the dimension of K is $n_1 < n$, let F be the subspace of R^n which is the linear hull of K, and q^* the orthogonal projection of q in F. Then the nearest point in K to q and q^* is the same, and the problem of finding the nearest point in K to q^* can be carried out completely in the subspace F itself, with K a full dimensional cone in it. Thus, without any loss of generality we assume that the cone K is of full dimension.

For any matrix D, we denote by $D_{i.}$, $D_{.j}$, its ith row, jth column respectively. (x_j) denotes the vector whose jth component is x_j. If S, T are two sets, S\T denotes the set of all elements of S not in T. When x is any vector, || x || denotes its Euclidean norm. When A is a square matrix, || A || denotes its norm which is maximum { || Ax || : over x satisfying || x || = 1 }.

2 The Two Forms of Input Data for the Nearest Point Problem

There are two distinct ways in which the convex polyhedral cone K may be specified.

One is as the *POS* cone of a specified subset of column vectors in R^n, that is, $K = POS(Q) = \{y : y = \sum_{j=1}^{m} \lambda_j Q_{.j}, \lambda \geqq 0\}$ where Q is an nxm data matrix. As mentioned above, when K is given in this form, we assume that the dimension of K is n, i.e., the rank of Q is n. In this form the data for the nearest point problem consists of q and Q.

The second form in which K may be given is as $\{ x : Ax \geqq 0 \}$ where A is an mxn data matrix. Here also we will assume that the dimension of K is n. In this form, the data for the problem consists of q and A.

In practical applications of the nearest point problem, K may appear in either form. When K is given in one of these two forms, to transform it into the other form typically requires exponential effort. A major exception to this occurs when K is simplicial, in this case both Q and A are square and nonsingular and $A = Q^{-1}$, and we can pass from one form to the other in this special case, with just one matrix inversion. We develop special versions of exterior penalty methods for the nearest point problem depending on the form in which K is given.

3 A Penalty Algorithm When K is Given in the POS Form

Let $K = POS(Q)$ where Q is of order nxm and rank n. In this case, the nearest point problem is: find $\lambda = (\lambda_1, ..., \lambda_m)^T$ to

$$\text{minimize} \quad \| q - Q\lambda \|^2$$

$$\text{subject to} \quad \lambda \geqq 0.$$

If λ^* is an optimum solution of this problem, $x^* = Q\lambda^*$ is the nearest point in K to q. It is well known that this problem always has an optimum solution, and that the nearest

point in K to q exists and is unique. The penalty approach solves this problem through an unconstrained minimization problem in λ of the form

$$\text{minimize} \qquad f(\lambda, \mu) = \| q - Q\lambda \|^2 + \frac{1}{\mu} P(\lambda), \text{ over } \lambda \in R^m$$

where $P(\lambda)$ is a penalty function associated with the region $\{ \lambda : \lambda \geq 0\}$, and μ is a positive penalty parameter. The various methods differ in the choice of the penalty function $P(\lambda)$. In this study we will consider the penalty functions that result in the following unconstrained minimizing functions $f(\cdot)$ which are distinguished by a subscript.

$$f_r(\lambda, \mu) = \| q - Q\lambda \|^2 + \frac{1}{\mu} \sum_{j=1}^{m} (\max \{0, -\lambda_j\})^r,$$

Let $$P_r(\lambda) = \sum_{j=1}^{m} (\max \{ 0, - \lambda_j \})^r$$

for $r = 2, 3, 4$. $f_2(\lambda, \mu)$ 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. All the functions $f_2(\lambda, \mu)$, $f_3(\lambda, \mu)$, $f_4(\lambda, \mu)$ are convex in the λ-space for fixed $\mu > 0$. We denote the row vector of partial derivatives of $f_r(\lambda, \mu)$ with respect to λ by $g_r(\lambda, \mu) = \nabla f_r(\lambda, \mu)$, and the hessian matrix of $f_r(\lambda, \mu)$ with respect to λ, for $r = 3, 4$, by $H(f_r(\lambda, \mu))$. We have, for $r = 2, 3, 4$,

$$g_r(\lambda, \mu) = \nabla f_r(\lambda, \mu) = -2q^TQ + 2\lambda^TQ^TQ + \frac{1}{\mu} P'_r(\lambda),$$

where $P'_r(\lambda) = r(\delta_1 \lambda_1^{r-1}, \ldots, \delta_m \lambda_m^{r-1})$ with δ_j defined by

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

for $j = 1$ to m. Also, for $r = 3, 4$, we have

$$H(f_r(\lambda, \mu)) = 2Q^TQ + \frac{1}{\mu} r(r-1)\, \text{diag}(\delta_1 \lambda_1^{r-2}, \ldots, \delta_m \lambda_m^{r-2}),$$

where for any $a_1, \ldots, a_m$, diag$(a_1, \ldots, a_m)$ is the diagonal matrix of order mxm with entries in the principal diagonal equal to $a_1, \ldots, a_m$.

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

Let $\bar{\lambda} \in R^m$ and suppose $d = (d_1, \ldots, d_m)^T$ is a descent direction for $f_r(\lambda, \mu)$ at $\bar{\lambda}$. Then

$$\frac{d\, f_r(\bar{\lambda} + \alpha d, \mu)}{d\alpha} = (-2d^TQ^Tq + 2d^TQ^TQ\bar{\lambda}) + 2\alpha d^TQ^TQd$$

$$+ \frac{1}{\mu} \sum_{j=1}^{m} r(-1)^r (\bar{\lambda}_j + \alpha d_j)^{r-1}\, d_j\, \zeta_j(\alpha) \qquad (1)$$

where $\zeta_j(\alpha) = 0$ if $\bar{\lambda}_j + \alpha d_j \geq 0$, and 1 if $\bar{\lambda}_j + \alpha d_j < 0$. So $\frac{d\, f_r}{d\alpha}(\bar{\lambda} + \alpha d, \mu)$ is a sum of at most $m + 2$ terms. Determine the set of ratios $\{ \frac{-\bar{\lambda}_j}{d_j} : j$ such that either $\bar{\lambda}_j > 0$, $d_j < 0$, or $\bar{\lambda}_j < 0$, $d_j > 0\}$, and arrange them in increasing order. Suppose there are s distinct values in this set, $\alpha_1 < \alpha_2 < \ldots < \alpha_s$. Define $\alpha_{s+1} = \infty$. For any $u = 1$ to s, in the interval $\alpha_u < \alpha < \alpha_{u+1}$, each of the terms on the right hand side of (1) is monotonic and has the

same sign. Since d is a descent direction for $f_r(\lambda, \mu)$ at $\bar{\lambda}$, $\frac{d f_r(\bar{\lambda} + \alpha d, \mu)}{d\alpha} < 0$ 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 $\frac{d f_r(\bar{\lambda} + \alpha d, \mu)}{d\alpha}$ for $\alpha = 0, \alpha_1, \alpha_2, \ldots$ 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 $\{ \bar{\lambda} + \alpha d : \alpha \geqq 0 \}$, 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 $\frac{d f_r}{d\alpha}(\bar{\lambda} + \alpha d, \mu)$ in this interval, as all the terms on the right hand side of (1) are monotonic and of the same sign.

Thus for the problem of minimizing $f_r(\bar{\lambda} + \alpha d, \mu)$ over $\alpha \geqq 0$, when d is a descent direction for $f_r(\lambda, \mu)$ at $\bar{\lambda}$, the optimum step length can be determined efficiently by the above method.

Selection of Search Directions

Given a fixed positive value for the penalty parameter μ, a descent method can be used for finding an unconstrained minimum of $f_r(\lambda, \mu)$ in λ. This method begins with an initial point $\lambda = \lambda^0$ and goes through several steps. If λ^0 is such that $q = Q\lambda^0$, then λ^0 is a global minimum of the problem when $\mu = \infty$ and thus could be used as a starting solution for $\mu > 0$ and large. In each step, a descent direction at the current point, λ^k, is generated, and a step is taken from λ^k in that direction. We get a variety of methods by varying the selection of search directions in each step. We basically consider two search directions.

Steepest descent search direction: Under this rule, the search direction at the current point λ^k 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 is a solution of the system

$$H(f_r(\lambda^k, \mu))y = -\nabla f_r(\lambda^k, \mu). \tag{2}$$

If K is simplicial (i.e., if $m = n$ and Q is of order nxn, since we already assumed that Q has rank n) then $H(f_r(\lambda, \mu))$ is positive definite and (2) always has a unique solution.

If K is not simplicial (i.e., Q is of order nxm, $m > n$), $H(f_r(\lambda^k, \mu))$ is positive semidefinite, and may be singular, and (2) 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. If this happens, we will replace (2) by

$$(H(f_r(\lambda^k, \mu)) + \tau I)\, y = -\nabla f_r(\lambda^k, \mu) \tag{3}$$

where $\tau > 0$ and I is the unit matrix of order m. The matrix on the left hand side of (3) is positive definite and hence (3) again has a unique solution which is a descent direction for $f_r(\lambda, \mu)$ at the current point λ^k.

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, we can take the step length for the move to be the optimum step length determined as discussed above. The method based on this strategy is the *Newton method with line searches*.

Termination

In practice we select a tolerance ε, positive and sufficiently small, and terminate the methods when the current point λ^k satisfies $\frac{1}{\mu} P_r(\lambda) < \varepsilon$, or if $\lambda^k \geqq -\varepsilon e$, where e is the column vector of all 1's in R^m. At that stage, the point $Q\lambda^k$ is accepted as the nearest point in K to q.

The Classical Exterior Penalty Method

This method, discussed extensively in nonlinear programming literature, goes through many iterations. In each iteration, the value of the penalty parameter μ is fixed, and an unconstrained minimization algorithm is used to find the global minimum $f_r(\lambda, \mu)$ over $\lambda \in R^m$. This algorithm itself may take several descent steps in this iteration. A commonly used termination criteria for this unconstrained minimization is

$$\| g_r(\lambda, \mu) \| \leqq \varepsilon^* \tag{4}$$

where ε^* is a positive and sufficiently small tolerance. Then the value of the penalty parameter μ is decreased, and the method moves to the next iteration.

It has been shown (see, for example, A. V. Fiacco and G. P. McCormick [1968]. K. Truemper [1975]) that this method converges to the optimum solution of the original problem. If $\lambda(\mu)$ denotes the unconstrained minimum of $f_r(\lambda, \mu)$ over λ, as a function of μ, then $\lambda(\mu)$ converges to λ^*, an optimum λ for the original problem, as $\mu \to 0$. In fact, as shown in K. Truemper [1975], 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 λ^*.

The New Exterior Penalty Method

We will now describe an algorithm based on the ideas of the classical penalty method, but differs from it in one important aspect. In this algorithm 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 μ, and repeat in 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 envelope 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 a prespecified accuracy, ρ be a multiplier satisfying $0 < \rho < 1$. Select $\mu^0 > 0$ and large, select λ^0 satisfying $Q\lambda^0 = q$, let $k = 0$.

Main step

1. λ^k is the current point. Find the direction d^k at λ^k (either the steepest descent direction or Newton's direction).
2. Select step length α_k. This is 1 for standard Newton method, or the optimum step length for other methods.
3. Let $\lambda^{k+1} = \lambda^k + \alpha_k d^k$. . If $\lambda^{k+1} \geqq - \varepsilon e$ stop. Otherwise, let $\mu^{k+1} = \rho \mu^k$, $k = k + 1$, return to 1.

4 Theoretical Results on Convergence of the Algorithm

Classical penalty methods are known to converge to an optimum solution of the original problem. Proofs can be found in nonlinear programming literature (for example, D. G. Luenberger [1984], M. S. Bazaraa, and C. M. Shetty [1979], A. V. Fiacco, and G. P. McCormick [1968], K. Truemper [1975], J. P. Evans, F. J. Gould, and J. W. Tolle [1973], T. Pietrzykowski [1968]). However, these proofs assume that an actual optimum solution for the problem of minimizing $f(\lambda, \mu)$ over $\lambda \in R^m$ is obtained for each fixed value of μ in a sequence converging to zero. We do not really obtain the unconstrained minimum of $f(\lambda, \mu)$ over $\lambda \in R^m$ for any value of μ before we change it. Hence, standard proofs of penalty methods do not apply directly to our algorithm.

In this section we give a convergence proof for standard Newton version of our algorithm in the simplicial case, that is, when Q is a nonsingular square matrix. So, $m = n$, and the algorithm considered is the one-step Newton method (step length = 1) attended by a decrease in the penalty parameter μ after every step. We restrict our attention to the case $r = 3$, i.e., the unconstrained minimizing function is $f_3(\lambda, \mu)$, which we denote by $f(\lambda, \mu)$ for the sake of simplicity. We denote by $g(\lambda, \mu)$, $H(\lambda, \mu)$ the gradient vector (as a row), and the Hessian matrix of $f_3(\lambda, \mu)$ with respect to λ. So,

$$g(\lambda, \mu) = -2(q - Q\lambda)^T Q - \frac{3}{\mu}(\delta_1\lambda_1^2, \ldots, \delta_n\lambda_n^2)$$

$$H(\lambda, \mu) = 2Q^T Q - \frac{6}{\mu}\,\mathrm{diag}\,(\delta_1\lambda_1, \ldots, \delta_n\lambda_n) \tag{5}$$

where $\delta = (\delta_1, \ldots, \delta_n)$ is a function of λ defined by : for j=1 to n

$$\delta_j = \begin{cases} 0 & \text{if } \lambda_j \geq 0 \\ 1 & \text{if } \lambda_j < 0 \end{cases} \qquad (6)$$

THEOREM 1: $f(\lambda, \mu)$ is a strictly convex function in $\lambda \in R^n$, for all $\mu > 0$.

PROOF: Since Q is nonsingular, Q^TQ is PD (positive definite). Also, from (6), we verify that $-(6/\mu)$ diag $(\delta_1\lambda_1, \ldots, \delta_n\lambda_n)$ is a positive semidefinite matrix for every $\mu > 0$ and $\lambda \in R^n$. So $H(\lambda, \mu)$ is PD for every $\mu > 0$ and $\lambda \in R^n$. Hence $f(\lambda, \mu)$ is a strictly convex function in $\lambda \in R^n$, for every $\mu > 0$. □

THEOREM 2: For each $\mu > 0$, the system

$$g(\lambda, \mu) = 0 \qquad (7)$$

has a unique solution in λ, say $\lambda(\mu)$. $\lambda(\mu)$ is a continuous function of μ in $\{ \mu : \mu > 0 \}$. Also, 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 from K = POS (Q). Let $\varphi = || q - x ||$, $S = \{ y : || q - y || \leq \varphi \}$, $\Gamma = \{ \lambda : \lambda = Q^{-1}y, y \in S \}$. Γ is a compact set, and as $f(\lambda, \mu)$ is a continuous function in λ, there is a point $\tilde{\lambda} \in \Gamma$ where $f(\lambda, \mu)$ attains its minimum over Γ. For all $y \notin S$, $|| q - y || > \varphi$, and hence for all $\lambda \notin \Gamma$, $|| q - Q\lambda ||^2 > \varphi^2$, which implies that $f(\lambda, \mu) > \varphi^2$ since $P_3(\lambda, \mu) \geq 0$ for all λ. Let $\bar{\lambda} = Q^{-1}x$. As $x \in K$ we have $\bar{\lambda} \geq 0$, thus $P_3(\bar{\lambda}, \mu) = 0$, and $f(\bar{\lambda}, \mu) = \varphi^2$. Hence for all $\lambda \notin \Gamma$, we have $f(\lambda, \mu) > \varphi^2$, and for at least one $\lambda \in \Gamma$ we have $f(\lambda, \mu) \leq \varphi^2$. This implies that $\tilde{\lambda}$, the minimizer of $f(\lambda, \mu)$ over Γ is its global minimum. Hence $f(\lambda, \mu)$ attains its minimum in λ, and since it is strictly convex in λ by Theorem 1, it has a unique global minimum in λ, which is attained at the solution to $g(\lambda, \mu) = 0$. Hence the system $g(\lambda, \mu) = 0$ has a unique solution, $\lambda(\mu)$, say. $\dfrac{\partial g(\lambda, \mu)}{\partial \lambda} =$

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

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

From Theorem 2 we know that $\{ \lambda (\mu) : \mu > 0 \}$ is a path in the λ-space, called the exterior path. The set $\{ \lambda : || g(\lambda, \mu) || \leqq \varepsilon, \mu > 0 \}$ for some $\varepsilon > 0$ defines an envelope containing the exterior path. We will now show that it is possible to interpret the points obtained in our algorithm, as a sequence of points in this envelope leading to λ^* as $\mu \rightarrow 0$.

For $\mu > 0, \varepsilon > 0$, define

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

All the eigenvalues of Q^TQ are real and positive since it is symmetric and PD. Let σ_1, σ_n be the smallest and largest eigenvalues respectively, of the symmetric PD matrix Q^TQ. Hence $\sigma_1 > 0$.

THEOREM 3: For $\mu > 0$ and every $\lambda \in R^n$, the smallest eigenvalue of $H(\lambda, \mu)$ is $\geqq 2\sigma_1$. Hence $f(\lambda, \mu)$ is a strongly convex function in λ for all $\mu > 0$.

PROOF: Let σ_s be the smallest eigenvalue of $H(\lambda, \mu)$. Then

$$\sigma_s = \min \{ z^T H(\lambda, \mu) z : || z || = 1 \} \tag{9}$$

From (5), we have

$$z^T H(\lambda, \mu)\, z = 2\, z^T Q^T Q z - \frac{6}{\mu} z^T (\mathrm{diag}\,(\delta_1\lambda_1, \ldots, \delta_n\lambda_n))z$$

$$\geq 2\, z^T Q^T Q z$$

from the definition of δ_js in (6). Hence, from (9),

$$\sigma_s \geq \min \{\, 2\, z^T Q^T Q z : \| z \| = 1 \,\}$$

$$= 2\sigma_1$$

This automatically implies that $f(\lambda, \mu)$ is strongly convex in λ for every $\mu > 0$. □

THEOREM 4: The set $\Omega(\mu, \varepsilon)$ is bounded for all $\varepsilon > 0$, whenever $\mu > 0$.

PROOF: Given $\mu > 0$, $\lambda(\mu)$ is the solution of $g(\lambda, \mu) = 0$ in λ. In Theorem 3 we established that $f(\lambda, \mu)$ is a strongly convex function in λ. Hence, there exists a constant γ such that for all λ, $\bar{\lambda}$,

$$\| g(\lambda, \mu) - g(\bar{\lambda}, \mu) \| \geq \gamma \| \lambda - \bar{\lambda} \| \qquad (10)$$

See B. T. Polyak [1987]. Substituting $\bar{\lambda} = \lambda(\mu)$ in (10) leads to

$$\| g(\lambda, \mu) \| \geq \gamma \| \lambda - \lambda(\mu) \| \qquad (11)$$

for all λ, since $g(\lambda(\mu), \mu) = 0$. From (11) we conclude that if $\lambda \in \Omega(\mu, \varepsilon)$, then

$$|| \lambda - \lambda(\mu) || \leq \varepsilon / \gamma \qquad (12)$$

The fact that $\lambda(\mu)$ is bounded, and (12) together imply the result in this theorem. □

We will now investigate what happens in one iteration of our algorithm.

THEOREM 5: Let μ, λ be the penalty parameter, and the vector respectively, at the end of one iteration. $\bar{\mu} = \rho\mu$ is the value of the penalty parameter for the next iteration. If $\varepsilon > 0$ is such that $|| g(\lambda, \mu) || \leq \varepsilon$, then $|| g(\lambda, \bar{\mu}) || \leq \frac{(1 - \rho)}{\rho} \beta + \frac{\varepsilon}{\rho}$ where β is a constant.

PROOF: $g(\lambda, \bar{\mu}) = a + \frac{1}{\bar{\mu}} b$ where $a = -2(q - Q\lambda)^T Q$ and $b = -3(\delta_1 \lambda_1^2, \ldots, \delta_n \lambda_n^2)$. So

$$\begin{aligned} || g(\lambda, \bar{\mu}) || &= || a + \frac{b}{\rho\mu} || \\ &= \frac{1}{\rho} || (\rho - 1)a + a + \frac{b}{\mu} || \\ &\leq \frac{(1 - \rho)}{\rho} || a || + \frac{1}{\rho} || a + \frac{b}{\mu} || \\ &\leq \frac{(1 - \rho)}{\rho} || a || + \frac{\varepsilon}{\rho} \qquad (13) \end{aligned}$$

Now $$\begin{aligned} || a || &= || -2(q - Q\lambda)^T Q || \\ &\leq 2|| q^T Q || + 2|| Q^T Q || \; || \lambda || \end{aligned}$$

$$\leq \quad \beta$$

where β is some positive constant since $\| \lambda \|$ is bounded on $\Omega(\mu, \varepsilon)$ by Theorem 4. Hence, from (13), we have

$$\| g(\lambda, \bar{\mu}) \| \quad \leq \quad \frac{1 - \rho}{\rho} \beta + \frac{\varepsilon}{\rho} \qquad \square$$

LEMMA 1: $\| (H(\lambda, \mu))^{-1} \| \leq 1/2\sigma_1$, where σ_1 is the smallest eigenvalue of Q^TQ.

PROOF: This follows from the arguments in the proof of Theorem 3. $\square$

THEOREM 6: Let μ, λ be the penalty parameter, and the vector at the end of one iteration. In the next iteration set the penalty parameter to $\bar{\mu} = \rho\mu$, and carry out a single Newton step leading to the new vector $\bar{\lambda}$. Then

$$\| g(\bar{\lambda}, \bar{\mu} \| \quad \leq \quad \bar{\alpha} \| g(\lambda, \mu) \|^2$$

where $\bar{\alpha} = \dfrac{3}{4\bar{\mu}\sigma_1^2}$

PROOF: Define

$$B = -\frac{3}{\bar{\mu}} \operatorname{diag}(\delta_1\lambda_1, \ldots, \delta_n\lambda_n)$$

$$H = H(\lambda, \bar{\mu}) = 2Q^TQ + 2B \qquad (14)$$

$$g = g(\lambda, \bar{\mu}) = -2q^TQ + 2\lambda^TQ^TQ - \frac{3}{\bar{\mu}}(\delta_1\lambda_1^2, \ldots, \delta_n\lambda_n^2)$$

From the Newton step we know that $\bar{\lambda} = \lambda - H^{-1} g(\lambda, \bar{\mu}) = \lambda - H^{-1}g$. Hence

$$g(\bar{\lambda}, \bar{\mu}) = -2q^TQ + 2\bar{\lambda}^TQ^TQ - \frac{3}{\bar{\mu}}(\delta_1\bar{\lambda}_1^2, \ldots, \delta_n\bar{\lambda}_n^2)$$

$$= -2q^TQ + 2(\lambda - H^{-1}g)^TQ^TQ - \frac{3}{\bar{\mu}}(\delta_i (\lambda_i - (H^{-1})_{i\cdot} g)^2)$$

$$= -2q^TQ + (\lambda - H^{-1}g)^T (2Q^TQ + 2B - 2B)$$

$$- \frac{3}{\bar{\mu}}(\delta_i (\lambda_i^2 - 2\lambda_i (H^{-1})_{i\cdot} g + ((H^{-1})_{i\cdot} g)^2))$$

$$= -2q^TQ + 2\lambda^TQ^TQ - g^TH^{-1} (H - 2B) \qquad (15)$$

$$- \frac{3}{\bar{\mu}}(\delta_i (\lambda_i^2 - 2\lambda_i (H^{-1})_{i\cdot} g + ((H^{-1})_{i\cdot} g)^2))$$

$$= -2q^TQ + 2\lambda^TQ^TQ - g^T + 2g^TH^{-1}B - \frac{3}{\bar{\mu}}(\delta_i \lambda_i^2)$$

$$-2g^TH^{-1}B - \frac{3}{\bar{\mu}}\left(((H^{-1})_{i\cdot} g)^2\right)$$

$$= -\frac{3}{\bar{\mu}}\left(((H^{-1})_{i\cdot} g)^2\right), \quad \text{by substituting for } g^T \text{ from}$$

(14) and cancelling terms.

So,

$$\| g(\bar{\lambda}, \bar{\mu}) \|^2 = \frac{9}{\bar{\mu}^2} \sum_{i=1}^{m} ((H^{-1})_i \cdot g)^4$$

$$\leq \frac{9}{\bar{\mu}^2} \left(\sum_{i=1}^{m} ((H^{-1})_i \cdot g)^2 \right)^2$$

Hence,

$$\| g(\bar{\lambda}, \bar{\mu}) \| \leq \frac{3}{\bar{\mu}} \| H^{-1} g \|^2$$

$$\leq \frac{3}{\bar{\mu}} \| H^{-1} \|^2 \| g \|^2$$

$$\leq \frac{3}{4\bar{\mu}\, \sigma_1^2} \| g \|^2 \quad \text{, by lemma 1}$$

That is, $$\| g(\bar{\lambda}, \bar{\mu}) \| \leq \frac{3}{4\bar{\mu}\, \sigma_1^2} \| g(\lambda, \bar{\mu}) \|^2$$

Now we select a sufficiently small target value for the penalty parameter μ, say $\hat{\mu}$, and a tolerance $\hat{\varepsilon}$, and show that the algorithm converges to a point λ satisfying (4), i.e., $\| g(\lambda, \mu^*) \| \leq \varepsilon^*$, for some $\mu^* \leq \hat{\mu}$, $\varepsilon^* \leq \hat{\varepsilon}$. By the results on the classical penalty method mentioned above, this will imply that point λ at termination is within the specified tolerance of λ^*, the optimum λ. Define

$$\alpha^* = \max \{ \frac{1}{2\hat{\varepsilon}}, \frac{3}{4\hat{\mu}\, \sigma_1^2} \} \qquad (16)$$

$$\mu^* = \frac{3}{4\alpha^* \sigma_1^2}, \quad \varepsilon^* = \frac{1}{2\alpha^*}$$

Therefore $\mu^* \leqq \hat{\mu}$, $\varepsilon^* \leqq \hat{\varepsilon}$. Since $\hat{\mu}$, $\hat{\varepsilon}$ are small, $\hat{\alpha}$ is a large positive number. Let $\beta^* > 1$ be a large positive number which is an upper bound for $|| \lambda ||$ over the set $\Omega(\mu^*, \varepsilon^*)$, guaranteed by Theorem 4. Select

$$\rho = 1 - \frac{1}{6\alpha^*\beta^*} \tag{17}$$

We will now denote by μ^k, λ^k, the penalty parameter, and the λ-vector at the end of iteration k in the algorithm, for k = 1, 2, ... $\mu^{k+1} = \rho\mu^k < \mu^k$ for all k, and we will show that when k is such that $\mu^k \leqq \mu^*$, then $|| g(\lambda^k, \mu^k) || \leqq \varepsilon^*$ will hold.

LEMMA 2: If the algorithm did not terminate in iteration k, then we will have

$$|| g(\lambda^{k+1}, \mu^{k+1}) || \leqq \alpha^* || g(\lambda^k, \mu^{k+1}) ||^2$$

PROOF: Since the algorithm did not terminate, we have $\mu^{k+1} > \mu^*$, and this lemma follows from Theorem 6. □

THEOREM 7: If $|| g(\lambda^k, \mu^k) || \leqq \varepsilon^*$, then $|| g(\lambda^{k+1}, \mu^{k+1}) || \leqq \varepsilon^*$ too.

PROOF: From Lemma 2, we have

$$|| g(\lambda^{k+1}, \mu^{k+1}) || \leqq \alpha^* || g(\lambda^k, \mu^{k+1}) ||^2$$

$$\leq \quad \alpha^* \left(\left(\frac{1 - \rho}{\rho} \right) \beta^* + \frac{\varepsilon^*}{\rho} \right)^2, \text{ by Theorem 5}$$

$$= \quad \alpha^* \left(\frac{(1 - \rho)^2 (\beta^*)^2}{\rho^2} + \frac{(\varepsilon^*)^2}{\rho^2} + \frac{2\varepsilon^*\beta^*(1 - \rho)}{\rho^2} \right)$$

$$= \quad \varepsilon^* \left(\frac{1}{18} + \frac{1}{2} + \frac{1}{3} \right) \left(\frac{1}{\rho^2} \right) \qquad (18)$$

by substituting for $1 - \rho$ from (17).

From the choice of ρ it is clear that the right hand side of (18) is $\leqq \varepsilon^*$, proving the theorem. □

How to initiate the algorithm?

If $Q^{-1}q \geqq 0$, then $q \in$ POS (Q), and hence q is itself the nearest point in POS (Q) to q, terminate. So, assume $Q^{-1}q \ngeqq 0$.

We will now show how to select an initial penalty parameter μ^o and point λ^o, so that $\| g(\lambda^o, \mu^o) \| \leqq \varepsilon^*$. Select

$$\lambda^o \quad = \quad Q^{-1}q \qquad (19)$$

$$\mu^o \quad = \quad \frac{3}{\varepsilon^*} \sqrt{\Sigma \left((\lambda_j^o)^4 : \text{over j such that } \lambda_j^o < 0 \right)}$$

Then it can be verified that $\| g(\lambda^o, \mu^o) \| = \varepsilon^*$. When initiated with λ^o, μ^o as given in (19), with ρ selected as in (17), it is clear that the method terminates after at most $(\log \mu^o - \log \mu^*) / (- \log \rho)$ iterations, with a λ close to the optimum point λ^*.

5 The exterior penalty method for the nearest point problem when *K* is specified by homogeneous linear inequalities.

Let $K = \{ x : Ax \geq 0 \}$ where $A = (a_{ij})$ is of order mxn. We assume that K is of full dimension in R^n. In this case the nearest point problem is: find $x \in R^n$ to

$$\text{minimize} \quad \| q - x \|^2$$

$$\text{subject to} \quad Ax \geqq 0$$

In the penalty approach we solve this problem through the unconstrained minimization problem of the form

$$\text{minimize } h_r(x, \mu) = \| q - x \|^2 + \frac{1}{\mu} \sum_{i=1}^{m} (\max \{ 0, -A_i. x \})^r \text{ over } x \in R^n$$

for r = 2, 3, or 4, where μ is a positive penalty parameter as before. For $\mu > 0$, $h_r(x, \mu)$ is convex in x for all these values of r. For r = 2, 3, 4, $\nabla h_r(\bar{x}, \mu)$, the gradient vector (as a row) of $h_r(x, \mu)$ with respect to x at $\bar{x}$, is

$$\nabla h_r(\bar{x}, \mu) = -2(q - \bar{x})^T + \frac{r}{\mu} \sum_{i=1}^{m} \upsilon_i (\max \{ 0, - A_i. \bar{x} \})^{r-1} A_i.$$

where $$\upsilon_i = \begin{cases} 0 & \text{if } A_i. \bar{x} \geqq 0 \\ (-1)^r & \text{if } A_i. \bar{x} < 0 \end{cases} \qquad (20)$$

for i = 1 to m. Also, for r = 3, 4, we have $H(h_r(x, \mu))$ = the hessian matrix of $h_r(x, \mu)$ with respect to x,

$$H(h_r(x, \mu)) = 2I + E$$

where I is the unit matrix of the order n, and $E = (e_{ij})$ of order nxn is given by

$$e_{ij} = \frac{r(r-1)}{\mu} \sum_{t=1}^{m} \upsilon_t (\max\{0, A_t. x\})^{r-2} a_{ti} a_{tj}$$

for i, j = 1 to n, with υ_t as defined in (20).

Using these, the new exterior penalty algorithm for this problem, based a single descent step followed by a decrease in the penalty parameter per iteration, is constructed in a way similar to that discussed in Section 3.

6 Extension to convex quadratic programs

Consider the convex quadratic program,

$$\text{minimize } z(x) = cx + \frac{1}{2} x^T D x$$

$$\text{subject to } Ax \geqq b$$

$$x \geqq 0$$

where D is a symmetric positive semidefinite matrix and A is of order mxn. In the penalty approach we solve this problem through the unconstrained minimization problem of the form

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

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

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

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

7 Results from computational experiments

So far, we have conducted a computational experiment with the new exterior penalty algorithm described in Section 3, for solving randomly generated nearest point problems in simplicial cones of dimension n ranging from 10 to 700.

DATA GENERATION: Results are reported for the case of problems in which q and Q are generated randomly as follows. Each element of q is generated from the uniform distribution between -5 to +5, of double word length (8 bytes). Likewise elements of Q are generated from the uniform distribution between -20 and +20, of the same word length. No nonsingularity checking was ever carried out on any of the matrices Q generated, but the system of equation (2) always had a solution in methods which used it to generate descent directions. The vector q and the matrix Q are fully dense in all the problems generated.

We have solved various other problems in which q and Q are randomly generated according to different distributions, but the performance of the algorithms turned out to be the same.

Implementation details

Experiments were carried out using the following versions of the algorithm.

VERSION 1: **Using Newton direction with step lengths 1:** In this version, exactly one descent step is carried out, followed by a decrease in the penalty parameter, per each iteration. $f_3(\lambda, \mu)$ and $f_4(\lambda, \mu)$ both have second derivatives at all $\lambda \in R^n$. 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 D. G. Luenberger [1984], 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 $f_2(\lambda, \mu)$, using the formula for the hessian given in Section 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 $f_2(\lambda, \mu)$, 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 $f_2(\lambda, \mu)$ 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).

Termination Criteria

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

i) $\lambda_j^k \geq -\varepsilon_1$, for all j = 1 to n

ii) $\frac{1}{\mu^k} P_r(\lambda^k) < \varepsilon_2$

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 Σ ($|\lambda_j|$: 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 version 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.

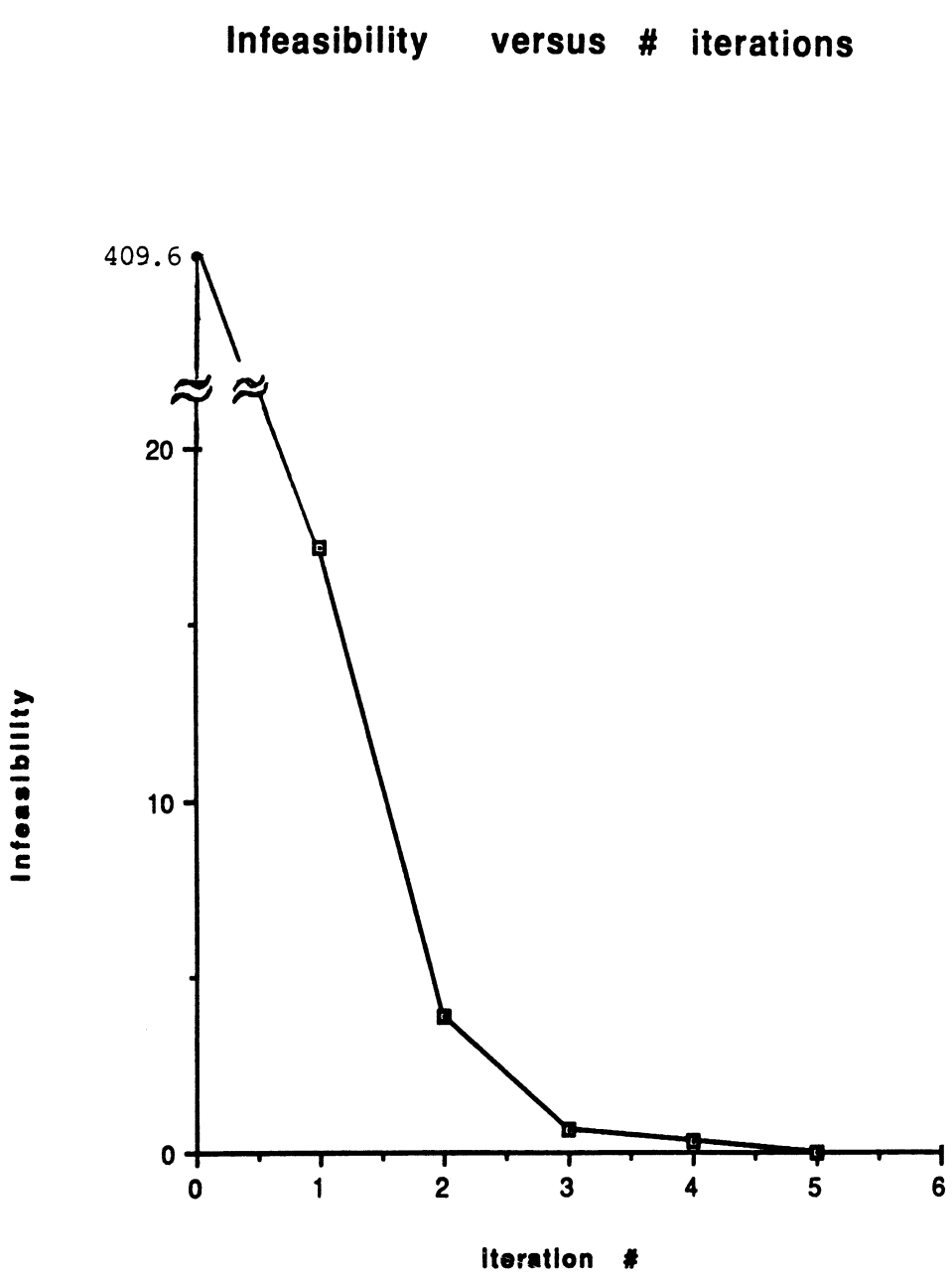
Infeasibility versus # iterations
409.6
20
10
0
Infeasibility
0
1
2
3
4
5
6
Iteration #

Figure 1

Updating the Penalty Paramter

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). The penalty parameter is updated using the formula $\mu^{k+1} = \rho\mu^k$. Clearly, the smaller the value of ρ, the more rapid is the decrease in μ. However if ρ is taken too small, the unconstrained minimum of $f_r(\lambda, \mu^{k+1})$ tends to be far away from the unconstrained minimum of $f_r(\lambda, \mu^k)$, thus ruining the chance of gaining advantage from using steps of Newton's method which is only locally convergent. So, we experimented with different values of ρ, and provided the best observed values of ρ in the tables.

A lot of experimentation has been done to determine the best value of μ^o, the initial penalty parameter value. The smaller the value of μ^o, the less the number of iterations to drive μ to its terminal value. However, since the starting λ, which is $\lambda^o = Q^{-1}q$, is the unconstrained minimum of $f_r(\lambda, \mu)$ only at $\mu = \infty$, choosing a small value for μ^o puts the unconstrained minimum of $f_r(\lambda, \mu^o)$ far away from λ^o, resulting in slow convergence. In most experiments, selecting $\mu^o = 0.01$ yielded excellent results.

Solving (2) For Finding Newton Directions

Several strategies have been tried for this. One is to solve this system directly using matrix factorizations. This provides accurate solutions but is expensive in computer time when the dimension n is large (particularly because all our problems are fully dense). Iterative methods based on preconditioned conjugate gradient methods turned out to be even more expensive computationally than direct matrix factorization methods. We also tried iterative successive overrelaxation (SOR) method, which turned out to be the best among all the methods we tried. In fact approximately $\sqrt{n}$ iterations of these SOR methods each with $0(n^2)$ effort yielded results comparable in accuracy to those obtained with $0(n^3)$ effort in other methods. The best value for the

relaxation parameter, ω^*, seems to increase with n (Table 2). However, a lot more work remains to be done to obtain good implementation for this aspect of the algorithm. Since almost all the work in algorithms based on Newton directions goes into solving system (2) in various steps, optimizing this aspect is very important.

The algorithms were coded in FORTRAN 77 using double precision arithmetic and run on an IBM 3033 or on an APOLLO Series 4000.

Summary of Computational Results: Number of Iterations

NEWTON BASED METHODS: In all versions based on Newton directions, it can be seen from Table 1 that the number of iterations grows extremely slowly, if at all, with problem dimension. As mentioned earlier, there is tremendous scope for improving the computer time taken for solving (2) in each iteration, hence the number of iteration is a more reliable guide of algorithm performance, than the computer time, and this is about 6 for version 1 (constant step length) and marginally less for version 2 (Newton with line searches), almost independent of problem dimension. Even with our current tentative equation solving routines for direction finding, it is clear that these algorithms are superior to implementations of other existing algorithms.

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.

We compared the performance of our algorithms 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 [1981] routine available through ACM algorithm 587 for linearly constrained least squares, and with our implementation of D.R. Wilhelmsen's nearest point algorithm. 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).

We found that the total number of iterations depends critically on the value of 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 of the algorithm.

STEEPEST DESCENT BASED METHODS: 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.

There is the possibility of using conjugate gradient based moves rather than steepest descent moves in each iteration. Implementing conjugate gradient moves may be tricky as μ becomes smaller, hence this has not been tested yet.

In comparing these methods 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.

These methods are more sensitive than Newton based methods 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 this version 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.

LINE SEARCHES IN NEWTON DIRECTIONS: In versions based on Newton's method with line searches, almost always the step length ranged between 0.8 to 1.2, with many of them very close to 1. Moreover, the line search faces numerical difficulties when the penalty parameter gets small. Thus, the following strategy was implemented: apply Newton's method with line search till the penalty parameter reaches a certain value, and then start applying standard Newton's method (i.e. with a constant step length of 1). Unfortunately, Newton's method with line search does not reduce the number of iterations significantly (this may be attributed to the surprising small number of iterations, and the size of the optimum step length which is close to 1).

SAVING ITERATIONS BY PROJECTIONS: It has already been mentioned earlier that if λ_j becomes > 0, it tended to remain > 0 in subsequent iterations. In the nearest point problem, if we know the set $J = \{ j : \lambda_j > 0$ in the optimum solution $\}$, it is well known that the nearest point itself can be found by orthoganally projecting q into the linear hull of the face of K corresponding to J. In some experiments we selected a tolerance ε_3 (about 10^{-3}) and when the current solution vector λ^k satisfied $\lambda_j \geq -\varepsilon_3$ for all j, we have taken $\{ j : \lambda_j^k > 0 \}$ as an estimate for J and used the projection strategy. This has cut the number of iterations by about 1/3 on an average.

μ Fixed At A Small Value

We experimented with selecting μ at the small target value initially itself and carrying out several descent steps keeping μ fixed. This scheme worked well and converged with the same (sometimes less) number of iterations as the usual scheme. However, numerical difficulties have been encountered as the product $\frac{1}{\mu}\lambda_j^2$ was large causing the hessian to be unstable. This does not happen when μ is gradually reduced because as μ^k gets smaller, $|\lambda_j^k|$ for j such that $\lambda_j^k < 0$ also gets smaller, and the numerical difficulty is avoided.

TABLE 1: Time in Seconds on IBM 3033 for Version 1 (Standard Newton Steps) Based on $f_r(\lambda, \mu)$

n	# problems	r = 2			r = 3			r = 4			Time for QPROG*
		Avg. # iteration	time	optimal ρ	Avg. # iteration	time	optimal ρ	Avg. # iteration	time	optimal ρ	
10	200	5.80	.658	.02	37.4	.71	.29	69.5	.77	.4	.656
20	200	6.01	.681	.02	38	.877	.29	70.1	1.075	.4	.675
30	200	6.03	.722	.02	38.1	1.17	.29	70.5	1.62	.4	.73
40	200	6.04	.7983	.02	38.13	1.61	.29	70.6	2.44	.4	.819
50	200	6.04	.895	.02	38.4	2.24	.29	70.79	3.59	.4	.964
100	100	6.08	1.98	.02	39.1	9.27	.29	71.2	16.35	.4	2.88
700	1	7	339.7	.02	+	+	+	+	+	+	652

Accuracy = 10^{-8}. System (2) solved by direct factorization using IMSL routines LFCSF and LSLSF

* This is M.J.D. Powell's inplementation of A. Idnani and D. Goldfarb's dual QP algorithm (IMSL routine QPROG).

+ Not tried.

TABLE 2 : Time in Seconds on APOLLO Series 4000 for Version 1 (Standard Newton Method) based on $f_2(\lambda, \mu)$

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 (2) solved by SOR with relaxation parameter ω* (found best by experimentation).

Almost in all cases $\sqrt{n}$ SOR iterations give an accurate solution to (2).

*This is the K. Haskell and R. Hanson's routine available through ACM algorithm 587.

TABLE 3: Time in Seconds on IBM 3033 for Version 3 (Steepest Descent Steps) based on $f_2(\lambda, \mu)$

n	# problems	Avg. # of iterations	Avg. # steps per iteration	Time	Optimal ρ	Time for QPROG
10	200	12	14.66	1.807	.1	.565
20	200	12	15.16	1.17	.1	.67
30	200	35	14.9	3.76	.5	.727
40	200	36	15.56	6.26	.5	.815
50	200	36	13.38	8.08	.5	.96
100	100	36	15.58	32.9	.5	2.33
250	1	37	14.9	197.3	.5	31.7

Accuracy = 10^{-3}

Conclusion

In our experiments so far we found that the best performance is obtained by the version using standard Newton steps and based on $f_2(\lambda, \mu)$. Of course, several other versions are still under investigation.

Acknowledgements

We acknowledge the many discussions we had with R. Saigal. We are also grateful to O. L. Mangasarian [1989] for bringing the result on strongly convex functions to our attention which greatly simplified the proof of Theorem 4.

References

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

2. S.Y. Chang and K.G. Murty, "The Steepest Descent Gravitational Method for Linear Programming," *Discrete Applied Mathematics*, to appear.

3. S.J. Chung and K.G. Murty, "Polynomially Bounded Ellipsoid Algorithms for Convex Quadratic Programming," in O.L. Mangasarian, R.R. Meyer, and S.M. Robinson (Eds), *Nonlinear Programming* 4, (Academic Press, 1981).

4. M.B. Daya and C.M. Shetty, "Polynomial Barrier Function Algorithms for Convex Quadratic Programming," School of ISE, Georgia Tech (Atlanta, GA, 1988).

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

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

7. 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.

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

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

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

11. M. Kojima, "A Polynomial-Time Algorithm for a Class of Linear Complementarity Problems", to appear in Mathematical Programming [1989].

12. M. Kojima, S. Mizuno, and A. Yoshise, "A Polymonial-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. O.L. Mangararian, Private Communication (1989).

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

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

19. T. Pietrzykowski, "An Exact Potential Method for Constrained Maxima," *SIAM J. Numerical Analysis*, 6, 2 (1968) 299-304.

20. B. T. Polyak, Introduction to Optimization, (Optimization Software, Inc., NY, 1987).

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

22. J. Renegar and M. Shub, "Simplified Complexity Analysis for Newton LP Methods," Tech Report No. 807, SORIE, Cornell University (Ithaca, NY, 1988).

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

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

25. P. Wolfe, "Algorithm for a Least Distance Programming," *Mathematical Programming Study 1* (1974) 190-205.

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

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

28. D.M. Young, "Convergence Properties of the Symmetric and Unsymmetric Successive Overrelaxation Methods and Related Methods," *Mathematics of Computation*, 24(112), (1970) 793-807.