A TWO-STAGE SUCCESSIVE OVERRELAXATION ALGORITHM FOR SOLVING THE LINEAR COMPLEMENTARITY PROBLEM

by

Karen M. Thompson

Computer Sciences Technical Report #706

July 1987

A Two–Stage Successive Overrelaxation Algorithm for Solving the Linear Complementarity Problem

Karen M. Thompson

Computer Sciences Department

University of Wisconsin, Madison, Wisconsin 53706

Abstract. We propose a two–stage successive overrelaxation (SOR) algorithm for solving the symmetric linear complementarity problem. After the first SOR preprocessing stage this algorithm concentrates on updating a certain prescribed subset of variables which is determined by exploiting the complementarity property. We demonstrate that this algorithm successfully solves problems with as many as 10,000 variables which cannot be tackled by other current algorithms.

Key Words: SOR, linear complementarity problem, large–scale programming

Abbreviated title: A Two–Stage SOR Algorithm

This material is based on research supported by National Science Foundation Grants DCR-8420963 and DCR-8521228 and Air Force Office of Scientific Research Grants AFOSR-86-0172 and AFOSR-86-0255.

1. INTRODUCTION

Successive overrelaxation algorithms are characterized by fast movement into a neighborhood of the solution followed by slow movement close to the solution. An active set strategy would therefore be useful after a few initial iterations of the SOR algorithm. The initial iterates of SOR are designed to identify a good guess for an initial active set. In large–scale programming this step is crucial. Typically active set strategies allow at most one constraint to be added to the active set at each iteration. Lenard, [Lenard79] conducted a computational study of various active set strategies on small quadratic programming problems. In her study, in general the number of changes in the active set appeared to fall in the range of $\frac{n}{3}$ to $\frac{n}{2}$, where n in the number of problem variables. In addition this number approximately corresponds to the number of active constraints at the solution of the test problems. Therefore if the initial active set chosen is not close to the solution active set an enormous number of iterations may be required for large–dimensional problems. Since each iteration may involve solving a large–scale problem this is not feasible for large–scale programming. Discussion of active set methods can be found in numerous sources including [Gill & Murray74],[Goldfarb72],and [Hoyle86]. By contrast our proposed two–stage method provides a good initial guess of the final active constraint set.

In a related work [Bertsekas82] proposes a projected Newton method. Bertsekas [Bertsekas82] proposes a superlinearly convergent scheme for solving

$$\min_{z \geq 0} f(z)$$

using the following iteration,

$$z^{i+1} = (z^i - \lambda^i D_i \nabla f(z^i))_+$$

where λ^i is a stepsize chosen by an Armijo–like rule D_i is a positive definite symmetric matrix which is partly diagonal and f(z) is strictly convex. The portion of D_i which is not diagonal may be chosen to be the inverse of the Hessian of $f(z)$ with respect to the indices corresponding to a subset of the variables. Although Bertsekas uses an active set strategy to partition the variables into two sets strictly speaking this is not an active set method. Active set methods are forced to remain on a linear manifold until a certain criterion is violated. In contrast Bertsekas' method allows the iterates to move off the active constraint set at any given iteration. This allows for multiple changes in the active set at any given iteration hence avoiding the inherent limitations of manifold suboptimization methods.

We propose here a two–stage SOR method. The idea is to use the SOR algorithm until a certain tolerance is reached and then use the current approximation of the solution to identify a set of constrained variables. We use a second SOR scheme for solving a complementary system of equations which leads to a new feasible point at which the process repeats. The use of the SOR algorithm at the beginning helps identify a set of active constraints which may be close to the set of active constraints at a solution. The second step is a Bertsekas–like method which is sparsity preserving.

Section 2 will specify the algorithm and state optimality conditions. Section 3 will establish convergence results. Finally, in Section 4 we discuss implementation of the algorithm and computational experience.

We briefly describe our notation now. For a vector z in the n–dimensional real space $\mathbb{R}^n$, z_+ will denote the vector in $\mathbb{R}^n$ with components $(z_+)_i = \max\{z_i,)\}$, $i = 1, \dots, n$. The scalar product of two vectors x and y in $\mathbb{R}^n$ will be simply denoted by xy. For $z \in \mathbb{R}^n$, $\| z \|_M = (zMz)^{1/2}$. $\mathbb{R}^{m \times n}$ will denote the set of all $m \times n$ real matrices. For $A \in \mathbb{R}^{m\times n}$, A^T will denote the transpose, A_i will denote the ith row, A_{ij} the element in row i and column j, and for $\mathcal{I} \in \{1, \dots, m\}$, $\mathcal{J} \in \{1, \dots, n\}$, $A_{\mathcal{I}}$ will denote the submatrix of A with rows A_i, $i \in \mathcal{I}$, while $A_{\mathcal{I}\mathcal{J}}$ will denote the submatrix of A with elements A_{ij}, $i \in \mathcal{I}, j \in \mathcal{J}$.

2. TWO–STAGE SOR ALGORITHM

We consider the problem

$$\min_{z \geq 0} f(z) = \frac{1}{2} zMz + qz \tag{2.1}$$

If M is symmetric positive semi-definite solving 2.1 is equivalent to solving the symmetric linear complementarity problem.

The idea of Two–Stage SOR (TSOR) is to use the SOR algorithm to approximate a solution to 2.1 and use this approximate solution to partition the variables into two sets. Define

$$\mathcal{I} := \mathcal{I}(\bar{z}) = \{j | \bar{z}_j > 0\} \quad \text{for } \bar{z} \ \text{ a solution of (2.1)}$$

Define the set at the ith iterate as follows

$$\mathcal{I}_i := \{j | z^i_j \leq \epsilon^i\} \quad \text{for some} \quad \epsilon^i \geq 0, \quad \epsilon^i \geq \bar{\epsilon} > 0. \tag{2.2}$$

From this we can partition M and z as follows

$$M := \begin{bmatrix} M_{\mathcal{I}_i} \\ M_{\mathcal{I}_i^c} \end{bmatrix}$$

$$z := \begin{bmatrix} z_{\mathcal{I}_i} & z_{\mathcal{I}_i^c} \end{bmatrix}$$

where $\mathcal{I}_i^c$ is the complement of $\mathcal{I}_i$ in $\{1, 2, \ldots, n\}$. We guess that $\mathcal{I}_i^c$ is the set of indices of components of z which will be positive in the solution. Therefore we must solve

$$M_{\mathcal{I}_i^c} z + q_{\mathcal{I}_i^c} = 0 \tag{2.3}$$

to satisfy complementarity. We use an SOR algorithm to approximate a solution to (2.3). Note that if M is positive definite, $\epsilon = 0$, $\mathcal{I}_i^c = \mathcal{I}$, then if we solve (2.3) we solve (2.1) exactly in one step.

We are now ready to specify the TSOR algorithm.

Algorithm 2.1.

Let E be a positive diagonal matrix in $\mathbb{R}^{n \times n}$, let $z^0 \geq 0$. Let $\omega > 0$ such that for some $\gamma_1, \gamma_2 > 0$

$$y((\omega E)^{-1} + K)y \geq \gamma_1 \parallel y \parallel_2^2 \quad , \forall y \in \mathbb{R}^n \tag{2.4}$$

$$y((\omega E)^{-1} + K - \frac{M}{2})y \geq \gamma_2 \parallel y \parallel_2^2 \quad , \forall y \in \mathbb{R}^n \tag{2.5}$$

Step 1

For $i \leq l$ where l is a positive integer

$$z^{i+1} = (z^i - \omega E(Mz^i + q + K(z^{i+1} - z^i)))_+$$

stop if $z^{i+1} = z^i$.

Step 2

Direction Generation

$$d^i := (p(z^i) - z^i)$$

where

$$p(z^i) := \begin{bmatrix} p^k_{\mathcal{I}_i^c}(z^i) \\ p^1_{\mathcal{I}_i}(z^i) \end{bmatrix}$$

where

$$\begin{aligned} p^k_{\mathcal{I}_i^c}(z^i) = p^{k-1}_{\mathcal{I}_i^c}(z^i) - \omega E_{\mathcal{I}_i^c \mathcal{I}_i^c}\big(M_{\mathcal{I}_i^c \mathcal{I}_i^c} p^{k-1}_{\mathcal{I}_i^c}(z^i) + M_{\mathcal{I}_i^c \mathcal{I}_i} z^i_{\mathcal{I}_i} \\ + q_{\mathcal{I}_i^c} + K_{\mathcal{I}_i^c \mathcal{I}_i^c}(p^k_{\mathcal{I}_i^c}(z^i) - p^{k-1}_{\mathcal{I}_i^c}(z^i))) \end{aligned} \tag{2.6}$$

$$p^1_{\mathcal{I}_i}(z^i) = (z_{\bar{\mathcal{I}}_i} - \omega E_{\mathcal{I}_i \mathcal{I}_i}(M_{\mathcal{I}_i} z^i + q_{\mathcal{I}_i}))_+ \tag{2.7}$$

Stop if $p(z^i) = z^i$.

Stepsize Generation

$$z^{i+1} = z^i + \lambda^i d^i$$

where

$$f(z^i + \lambda^i d^i) = \min_{\lambda}\{f(z^i + \lambda d^i) | z^i + \lambda d^i \geq 0\} \tag{2.8}$$

Go to step 2.

We are now ready to state optimality conditions for Algorithm 2.1.

Theorem 2.1. *(TSOR optimality conditions) Let* $M, K, E \in \mathbb{R}^{n\times n}, q \in \mathbb{R}^n, \omega > 0$ *such that* E *is a positive diagonal matrix.*

(a) If z *solves the LCP and* $(\omega E_{\mathcal{F}\mathcal{F}})^{-1} + K_{\mathcal{F}\mathcal{F}}$ *is nonsingular, where* $\mathcal{F}$ *is any subset of* $\{1, 2, \ldots, n\}$*, then* $p(z) = z$ *where* $p(z)$ *is the solution of (2.6) and (2.7).*

(b) If $p(z)$ *solves (2.6) and (2.7) and* $p(z) = z$ *and if* $(\omega E)^{-1} + K - \frac{M}{2}$ *and* $(\omega E)^{-1} + K$ *are positive definite then* z *solves the LCP.*

Proof :

(a) Suppose z solves the LCP. $\mathcal{I} = \{j | z_j \leq \epsilon\}$. It is easy to verify that

$$z_{\mathcal{I}} = (z_{\mathcal{I}} - \omega E_{\mathcal{I}\mathcal{I}}(M_{\mathcal{I}} z + q_{\mathcal{I}}))_+$$

. Consider $p^1_{\mathcal{I}^c}(z) = z_{\mathcal{I}^c} - \omega E_{\mathcal{I}^c\mathcal{I}^c}(M_{\mathcal{I}^c} z + q_{\mathcal{I}^c} + K(p^1_{\mathcal{I}^c}(z) - z_{\mathcal{I}^c}))$. By rearranging terms we get

$$((\omega E_{\mathcal{I}^c\mathcal{I}^c})^{-1} + K_{\mathcal{I}^c\mathcal{I}^c})p^1_{\mathcal{I}^c}(z) = ((\omega E_{\mathcal{I}^c\mathcal{I}^c})^{-1} + K_{\mathcal{I}^c\mathcal{I}^c})z_{\mathcal{I}^c}.$$

Therefore since $((\omega E_{\mathcal{I}^c\mathcal{I}^c})^{-1} + K_{\mathcal{I}^c\mathcal{I}^c}$ is nonsingular $p^1_{\mathcal{I}^c}(z) = z$. This implies that $p^k_{\mathcal{I}^c}(z) = z$ for every k. Therefore $p(z) = z$.

(b) Now suppose $p(z) = z$. One can easily show that

$$z_{\mathcal{I}} \geq 0, M_{\mathcal{I}}z + q_{\mathcal{I}} \geq 0, z_{\mathcal{I}}(M_{\mathcal{I}}z + q_{\mathcal{I}}) = 0$$

. We now show that

$$z_{\mathcal{I}^c} \geq 0, M_{\mathcal{I}^c}z + q_{\mathcal{I}^c} \geq 0, z_{\mathcal{I}^c}(M_{\mathcal{I}^c}z + q_{\mathcal{I}^c}) = 0 \tag{2.9}$$

is satisfied. It is easy to show that if $p^k_{\mathcal{I}^c}(z) = z_{\mathcal{I}^c}$ and $k = 1$ then (2.5) is satisfies. We will show that this is true for arbitrary k. Assume that $p^k_{\mathcal{I}^c}(z) = z$ for some $k \geq 2$ but $p^1_{\mathcal{I}^c}(z) \neq z_{\mathcal{I}^c}$. Then $z_{\mathcal{I}^c}$ is an accumulation point of the iteration

$$z^{i+1} = z^i - \omega E_{\mathcal{I}^c\mathcal{I}^c}(M_{\mathcal{I}^c\mathcal{I}^c}z^i_{\mathcal{I}^c} + M_{\mathcal{I}^c\mathcal{I}}z_{\mathcal{I}} + q_{\mathcal{I}^c} + K_{\mathcal{I}^c\mathcal{I}^c}(z^{i+1}_{\mathcal{I}^c} - z^i_{\mathcal{I}^c}))$$

Then since $(\omega E_{\mathcal{I}^c\mathcal{I}^c})^{-1} + K_{\mathcal{I}^c\mathcal{I}^c} - \frac{M_{\mathcal{I}^c\mathcal{I}^c}}{2}$ is positive definite and by the argument in Theorem 2.1 of [Mangasarian77],

$$M_{\mathcal{I}^c}z + q_{\mathcal{I}^c} = 0$$

. But then by part (a) $z = p(z)$. In particular $z_{\mathcal{I}^c F} = p^1_{\mathcal{I}^c}(z)$. This contradicts the assumption. Therefore (2.5) is satisfied. ∎

3. CONVERGENCE OF TSOR

In order to prove the convergence result we first need the following two Lemmas.

Lemma 3.1. *Let $f(z) = \frac{1}{2}zMz + qz$. Let the sequence $\{z^i\}$ be generated by the following SOR iteration.*

$$z^{i+1} = z^i - \omega E(Mz + q + K(z^{i+1} - z^i))$$

Then the sequence $\{f(z^i)\}$ is non-increasing.

Proof :

The proof follows from a simple modification of the proof of Theorem 2.1 in [Mangasarian77]. ∎

Suppose the sequence generated by the TSOR algorithm has a convergent subsequence. Lemma 3.2 will establish the boundedness of the directions associated with that convergent subsequence.

Lemma 3.2. *If $z^{i_j} \to \bar{z}$ then d^{i_j} is bounded.*

Proof :

We have $d^{i_j} = (p(z^{i_j}) - z^{i_j})$.

Since the sets $\mathcal{I}^{i_j}$ are finite without loss of generality assume $\mathcal{F} = \mathcal{I}^{i_1} = \mathcal{I}^{i_2} = \ldots = \mathcal{I}^{i_j} = \ldots$. We will show that the components of $p(z^{i_j})$, $p^k_{\mathcal{F}^c}(z^{i_j})$ and $p^1_{\mathcal{F}}(z^{i_j})$ are bounded. We have

$$p^1_{\mathcal{F}}(z^{i_j}) = (z^{i_j}_{\mathcal{F}} - \omega E_{\mathcal{F}\mathcal{F}}(M_{\mathcal{F}} z^{i_j} + q_{\mathcal{F}}))_+$$

Suppose $p^1_{\mathcal{F}}$ is not bounded. Then we get the following contradiction.

$$0 \neq \bar{p}_{\mathcal{F}} = \lim_{j\to\infty} \frac{p^1_{\mathcal{F}}(z^{i_j})}{\| p^1_{\mathcal{F}}(z^{i_j}) \|} = \lim_{j\to\infty} \frac{(z^{i_j} - \omega E_{\mathcal{F}\mathcal{F}}(M_{\mathcal{F}} z^{i_j} + q_{\mathcal{F}}))_+}{\| p^1_{\mathcal{F}}(z^{i_j}) \|} = 0$$

Therefore $p^1_{\mathcal{F}}(z^{i_j})$ is bounded.

Now suppose $p^k_{\mathcal{F}^c}(z^{i_j})$ is not bounded. Consider the case when $k = 1$. Then we have

$$0 \neq \bar{p} = \lim_{j\to\infty} \frac{p^1_{\mathcal{F}^c}(z^{i_j})}{\| p^1_{\mathcal{F}^c}(z^{i_j}) \|} = -\omega E_{\mathcal{F}^c\mathcal{F}^c} K_{\mathcal{F}^c\mathcal{F}^c} \bar{p}$$

Therefore $((\omega E_{\mathcal{F}^c\mathcal{F}^c})^{-1} + K_{\mathcal{F}^c\mathcal{F}^c})\bar{p} = 0, \bar{p} \neq 0$ This contradicts the positive definiteness of $(\omega E)^{-1} + K$. Therefore $\{p^1_{\mathcal{F}^c}(z^{i_j})\}$ is bounded. The result follows for arbitrary k by induction. ∎

We are now ready to prove the main result of this section.

Theorem 3.1. *(TSOR convergence) Let M be symmetric and positive semidefinite. Either the sequence $\{z^i\}$ generated by Algorithm 2.1 terminates at a solution of the LCP or each accumulation point of $\{z^i\}$ solves the LCP.*

Proof :

The sequence $\{z^i\}$ terminates only if for some i, $p(z^i) = z^i$, in which case by Theorem 2.1, z^i solves the LCP. Suppose now $\{z^i\}$ does not terminate and the $\bar{z}$ is an accumulation point of $\{z^i\}$. Let

$$f_{\mathcal{I}^c_i}(z_{\mathcal{I}^c}) := f(z_{\mathcal{I}^c_i}, z^i_{\mathcal{I}_i})$$

and

$$f_{\mathcal{I}_i}(z_{\mathcal{I}}) := f(z^i_{\mathcal{I}_i^c}, z_{\mathcal{I}_i})$$

We have

$$\begin{aligned}
-\nabla_{\mathcal{I}_i^c} f(z^i) d^i_{\mathcal{I}_i^c} &= -\nabla_{\mathcal{I}_i^c} f(z^i)(p^k_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c}) \\
&= f_{\mathcal{I}_i^c}(z_{\mathcal{I}^c}) - f_{\mathcal{I}_i^c}(p^k_{\mathcal{I}_i^c}(z^i)) + \frac{1}{2} \parallel p^k_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c} \parallel^2_{M_{\mathcal{I}_i^c \mathcal{I}_i^c}} \\
&\geq f_{\mathcal{I}_i^c}(z_{\mathcal{I}^c}) - f_{\mathcal{I}_i^c}(p^1_{\mathcal{I}_i^c}(z^i)) + \frac{1}{2} \parallel p^k_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c} \parallel^2_{M_{\mathcal{I}_i^c \mathcal{I}_i^c}} \\
&\qquad \text{(By Lemma 3.1)} \\
&\geq f_{\mathcal{I}_i^c}(z^i_{\mathcal{I}^c}) - f_{\mathcal{I}_i^c}(p^1_{\mathcal{I}_i^c}(z^i)) \\
&\qquad \text{(By positive semidefiniteness of M)} \\
&= -\Big((p^1_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c}) + \omega E_{\mathcal{I}_i^c \mathcal{I}_i^c}(M_{\mathcal{I}_i^c} z^i + q_{\mathcal{I}_i^c} \\
&\quad + K_{\mathcal{I}_i^c \mathcal{I}_i^c}(p^1_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c})) \\
&\qquad \times (\omega E_{\mathcal{I}_i^c \mathcal{I}_i^c})^{-1}(p^1_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c})\Big) \\
&\qquad - \parallel p^1_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c} \parallel^2_{M_{\mathcal{I}_i^c \mathcal{I}_i^c}/2 - K_{\mathcal{I}_i^c \mathcal{I}_i^c} - (\omega E_{\mathcal{I}_i^c \mathcal{I}_i^c})^{-1}} \\
&\qquad \text{(By Taylor Series Expansion)} \\
&= \parallel p^1_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c} \parallel^2_{(\omega E_{\mathcal{I}_i^c \mathcal{I}_i^c})^{-1} + K_{\mathcal{I}_i^c \mathcal{I}_i^c} - M_{\mathcal{I}_i^c \mathcal{I}_i^c}/2} \\
&\qquad \text{(By (2.6))} \\
&\geq \tilde{\gamma} \parallel p^1_{\mathcal{I}_i^c}(z^i) - z^i_{\mathcal{I}_i^c} \parallel^2 \\
&\qquad \text{(By (2.5))}
\end{aligned}$$

Now, we find a similar result for the second case.

$$
\begin{aligned}
-\nabla f_{\mathcal{I}_i}(z^i)d^i_{\mathcal{I}_i} &= -\nabla f_{\mathcal{I}_i}(z^i)(p_{\mathcal{I}_i}(z^i) - z_{\mathcal{I}_i}) \\
&= -(M_{\mathcal{I}_i} z^i + q_{\mathcal{I}_i})(p^1_{\mathcal{I}_i}(z^i) - z^i_{\mathcal{I}^i}) \\
&= -\Big((p^1_{\mathcal{I}_i}(z^i) - z_{\mathcal{I}_i}) + (\omega E_{\mathcal{I}_i\mathcal{I}_i})(M_{\mathcal{I}_i} z^i + q_{\mathcal{I}_i})\Big) \\
&\quad \times \Big((\omega E_{\mathcal{I}_i\mathcal{I}_i})^{-1}(p^1_{\mathcal{I}_i}(z^i) - z^i_{\mathcal{I}_i})\Big) + \| p^1_{\mathcal{I}_i}(z^i) - z^i_{\mathcal{I}^i} \|_{(\omega E_{\mathcal{I}_i\mathcal{I}_i})^{-1}} \\
&\geq \| p^1_{\mathcal{I}_i}(z^i) - z^i_{\mathcal{I}^i}) \|^2_{(\omega E_{\mathcal{I}_i\mathcal{I}_i})^{-1}} \\
&\qquad\qquad \text{(By Lemma 2.2. of [Mangasarian77])} \\
&\geq \hat{\gamma} \| p^1_{\mathcal{I}_i}(z^i) - z^i_{\mathcal{I}_i} \|^2 \\
&\qquad\qquad \text{(Since } E \text{ is a positive diagonal)}
\end{aligned}
$$

Therefore we have

$$
\begin{aligned}
-\nabla f(z^i)d^i &= -\nabla_{\mathcal{I}^c_i} f(z^i) d_{\mathcal{I}^c_i} - \nabla_{\mathcal{I}_i} f(z^i) d_{\mathcal{I}_i} \\
&\geq \tilde{\gamma} \| p^1_{\mathcal{I}^c_i}(z^i) - z^i_{\mathcal{I}^c_i} \|^2 + \hat{\gamma} \| p^1_{\mathcal{I}_i}(z^i) - z^i_{\mathcal{I}_i} \|^2 \\
&\geq \gamma \| p(z^i) - z^i \|^2
\end{aligned}
$$

where $\gamma = \min\{\tilde{\gamma}, \hat{\gamma}\}$.

Now, suppose that $\{z^{i_j}\} \to \bar{z}$. By Lemma 3.2 we have $\{p(z^{i_j})\}$ is bounded. Therefore without loss of generality we can assume $\{z^{i_j}, p(z^{i_j})\} \to (\bar{z}, \bar{p})$. We have $\bar{d} = \bar{p} - \bar{z}$. Then

$$z^{i_j} + \lambda d^{i_j} \geq 0 \text{ for } 0 \leq \lambda \leq \bar{\lambda}^{i_j}$$

where

$$\bar{\lambda}^{i_j} = \min_{k \in \mathcal{I}^c_{i_j}} \{1, z_k^{i_j} / d_k^{i_j} | z_k^{i_j} + d_k^{i_j} < 0\}.$$

Since $\{\bar{\lambda}^{i_j}\}$ is bounded without loss of generality we can assume that $\bar{\lambda}^{i_j} \to \bar{\lambda}$. Therefore

$$f(z^{i_j} + \lambda d^{i_j}) \geq f(z^{i_j+1}) \geq f(z^{i_{j+1}}) \quad \text{for } 0 \leq \lambda \leq \bar{\lambda}^{i_j}$$

Therefore

$$f(\bar{z} + \lambda \bar{d}) \geq f(\bar{z}) \quad \text{for} \quad 0 \leq \lambda \leq \bar{\lambda} \tag{3.1}$$

We claim that $\bar{\lambda} \neq 0$. Since each $\mathcal{I}^c_{i_j}$ is a finite set at least one set $\mathcal{F}$ occurs infinitely often. Therefore without loss of generality assume $\mathcal{I}^c_{i_j} = \mathcal{F}$ for all j. Now, suppose $\bar{\lambda}^{i_j} \to 0$. Then

$$\lim_{j \to \infty} \frac{-z_*^{i_j}}{d_*^{i_j}} = 0$$

where

$$\frac{-z_*^{i_j}}{d_*^{i_j}} = \min_{k \in \mathcal{F}} \{-z_k^{i_j} / d_k | z_k^{i_j} + d_k < 0\}$$

since $\{d^{i_j}\}$ is bounded then $\bar{\lambda}^{i_j} \to 0$ implies that $-z_*^{i_j} \to 0$ However $z_*^{i_j} \in \mathcal{F}$ and therefore $z_*^{i_j} > \bar{\epsilon}$ for all j by (2.2). This contradiction implies that $\bar{\lambda} \neq 0$. Therefore from (3.1) we have

$$\nabla f(\bar{z}) \bar{d} \geq 0$$

. Then

$$0 \geq -\nabla f(\bar{z}) \bar{d} = \lim_{j \to \infty} -\nabla f(z^{i_j}) d^{i_j} \geq \lim_{j \to \infty} \gamma \| p(z^{i_j}) - z^{i_j} \|^2 \geq 0$$

Therefore $\| p(\bar{z}) - \bar{z} \|^2 = 0$. By Theorem 2.1 we have $\bar{z}$ solves the LCP. ∎

We have the following special case of TSOR.

Corollary 3.1. *Let M be symmetric positive semidefinite. In addition let $K = L$ and $E = D^{-1} > 0$ where L is the strictly lower triangular part of M and D is the diagonal part of M. Then conditions (2.4) and (2.5) are satisfied for $0 < \omega < 2$.*

4. IMPLEMENTATION AND COMPUTATIONAL RESULTS

There are two obvious questions that come to mind when implementing the TSOR Algorithm. The first is what criterion should be used to switch from Step 1 to Step 2 of the algorithm. The second question is how do we choose k when computing $p^k_{\mathcal{I}^c_i}(z^i)$.

The criterion for switching must be such that it provides a good initial guess of the active set at a solution. The criterion we implemented checks whether the set $\mathcal{I}^c_i$ has changed every l iterations. When that set has not changed over l iterations then the algorithm proceeds to Step 2. The size of l is important. If we wait too long to make the switch we lose the full advantage of moving to Step 2. If we switch too early we may have failed to identify an appropriate set and may spend a lot of time switching sets. Since the iterations in Step 1 are much cheaper than those in Step 2 we would prefer to stay in Step 1 under these conditions. We found that checking every 10 iterations was appropriate for very sparse problems, such as less than 1% density. Checking after every 5 iterations seemed to improve results on the denser problems.

The second question involves deciding on how much work to do to find the direction in Step 2. Notice, that if M is positive definite and we solve exactly the system

$$M_{\mathcal{I}_i^c \mathcal{I}_i^c} z_{\mathcal{I}_i^c} + M_{\mathcal{I}_i^c \mathcal{I}_i} z^i_{\mathcal{I}_i} + q_{\mathcal{I}_i^c} = 0 \tag{4.1}$$

then we are essentially using Bertsekas' method without explicitly defining D_i. However, this would involve an infinite number of SOR iterates. We could solve (4.1) to a stringent tolerance. This would provide a good direction if we have identified the set correctly. If we have failed to identify the set correctly it may not be worthwhile to do so much work for one iteration. We suggest the following scheme

0. Start with a loose tolerance, γ^i and a stringent tolerance $\bar{\gamma}$. Define final tolerance ϵ.
1. Solve until $\| p^k_{\mathcal{I}_i^c}(z^i) - p^{k-1}_{\mathcal{I}_i^c}(z^i) \| < \gamma^i$
2. Compute z^{i+1}. If $\| (-Mz - q)_+, z(Mz + q)) \| < \epsilon$ then go to 5.
3. Identify $\mathcal{I}_{i+1}$. If $\mathcal{I}_{i+1} = \mathcal{I}_i$ then $\gamma^{i+1} = \bar{\gamma}$ else $\gamma^{i+1} = \alpha\gamma^i$ for $0 < \alpha < 1$.
4. Go to 1.
5. Stop

In addition, we have included an upper bound on the size of k to avoid to much time being spent on a given iteration.

The algorithm was tested on randomly generated LCPs. We randomly generate a matrix A and define

$$M := AA^T$$

A random solution $\bar{z}$ is generated and q is selected such that

$$\bar{z}_i > 0 \Longrightarrow q_i = -M_i\bar{z}$$

$$\bar{z}_i = 0 \Longrightarrow q_i > -M_i\bar{z}$$

The performance of TSOR is compared with the standard SOR algorithm,

$$z^{i+1} = (z^i - \omega E(Mz^i + q + K(z^{i+1} - z^i)))_+$$

where

$$E = D^{-1}, K = L, \omega > 0$$

The convergence criteria used is

$$\| (-Mz^i - q)_+, z^i(Mz^i + q) \| < .5 \times 10^{-4}$$

Tables 4.1, 4.2, 4.3,and 4.4 display the results for varying dimensions of a positive definite M. Tables 4.5, 4.6, and 4.7 display results for positive semidefinite M. The problem density indicates the fraction of non-zero elements in the matrix M. The solution density indicates the fraction of non-zero elements in the generated solution $\bar{z}$. We note that in Step 2 of the algorithm the time to complete one iteration will increase as the number of non-zero elements in the solution increases. Therefore we tested various densities of the solution vector. The columns *iter* and *time* indicate the number of iterations and the total time in seconds for convergence using the standard SOR Algorithm. The next four columns describe the results from the

TSOR Algorithm applied to the same problem. The first column labeled *total iter* indicates the total number of iterations to reach convergence. The next column labeled *SOR iter* indicates the total number of SOR iterations before switching to Step 2. The column labeled *inner iter* indicates the cumulative number of SOR iterations needed to calculate the directions in Step 2 on the algorithm. The column labeled *time* indicates the total time in seconds for the TSOR Algorithm to satisfy the convergence criteria. Finally the column labeled *factor* compares the two algorithms by measuring

$$\frac{\text{Time for SOR Algorithm}}{\text{Time for TSOR Algorithm}}$$

We make the following observations about the results for positive definite M shown in Tables 4.1, 4.2, 4.3 and 4.4.

(1) The most dramatic improvements occur when the solution density is small. The bulk of the time spent on one iteration in Step 2 is calculating $p^k_{\mathcal{I}^c_i}(z^i)$. We note that when the set $\mathcal{I}^c_i$ has few elements, determining the direction at the ith iteration is much faster. Therefore we might expect the most dramatic improvements to occur in this case.

(2) The largest number of Step 2 iterations needed for convergence was 19,and often we need less than five extra iterations. However the number of inner SOR iterations needed to generate these iterations rises as the density of the solution

rises. This can be attributed to the complementarity condition which causes the system of equations

$$M_{\mathcal{I}_i^c \mathcal{I}_i^c} z_{\mathcal{I}_i^c} + q_{\mathcal{I}_i^c} + M_{\mathcal{I}_i^c \mathcal{I}_i} z_{\mathcal{I}_i} = 0$$

to be of much larger dimension for problems with denser solutions. Therefore the number of SOR iterations needed to solve the system to a certain tolerance is larger than a smaller size problem.

(3) SOR is not as effective on very sparse problems. TSOR carries some of the same difficulties as regular SOR since we are using the SOR iterate as the basis for generating the direction. However in TSOR we consider

$$M_{\mathcal{I}_i^c \mathcal{I}_i^c} z_{\mathcal{I}_i^c} + q_{\mathcal{I}_i^c} + M_{\mathcal{I}_i^c \mathcal{I}_i} z^i_{\mathcal{I}_i}$$

The matrix $M_{\mathcal{I}_i^c \mathcal{I}_i^c}$ may not be as sparse as M and therefore the SOR iterate to generate the direction may be more effective. Note that when we exclude the dramatic improvements in the first two lines of Table 4.2 and the first line in Table 4.3 the improvement factor tends to be better for sparse problems.

To test for the positive semidefinite case we generated a matrix of rank $\frac{4}{5} \times n$. Tables 4.5, 4.6 and 4.7 display results from these tests. In tables 4.5 and 4.6 we use the same stopping criterion as previously stated with the exception of problems with solution density .6 and .8 in table 4.6. These problems were run to a tolerance of $.5 \times 10^{-3}$ and $.5 \times 10^{-2}$ respectively.

We note that although table 4.5 seems to mirror the positive definite results, a comparison of tables 4.6 and 4.2 finds the positive semidefinite case requites more second stage iterations. This can be explained by the fact that in the positive semidefinite case the LCP has infinitely many solutions and as a result there may be more changes in the active sets. In addition the resulting system of equations may not be of full rank and in fact is not even guaranteed to be consistent.

Finally, we note that the principle advantage of this method over a direct implementation of Bertsekas method is the ability to solve huge problems without storage difficulties and in a reasonable amount of time. The largest problem we have tested is a positive semi–definite LCP of dimension 10,000. Table 4.7 displays the results of that run and provides a comparison with the SOR algorithm. Bertsekas has solved special positive definite problems with 10,000 variables using his algorithm. The special nature of his problems allows him to use Riccati equations to efficiently compute the Newton direction. However, we know of no method other than TSOR, that can handle general positive semi-definite problems of this size.

Table 4.1

COMPARISON OF SOR AND TSOR

M POSITIVE DEFINITE, DIMENSION 1500

		SOR		**TSOR**				
problem	*solution*		*time*	*total*	*SOR*	*inner*	*time*	*speedup*
density	*density*	*iter*	*seconds*	*iter*	*iter*	*iter*	*seconds*	*factor*
.03131	.25	34	91.76	19	15	28	67.36	1.3622
.03131	.5	50	138.68	29	25	19	108.86	1.2738
.03131	.7	90	250.13	45	40	38	198.68	1.2589
.03131	.8	150	419.81	60	55	86	358.46	1.1711
.00427	.25	317	167.76	34	30	200	34.85	4.8139
.00427	.5	1000*	516.50	55	40	750	179.96	2.8699
.00427	.7	1499	783.40	79	60	907	348.05	2.2508

Table 4.2

COMPARISON OF SOR AND TSOR

M POSITIVE DEFINITE, DIMENSION 2000

		SOR		**TSOR**				
problem	*solution*		*time*	*total*	*SOR*	*inner*	*time*	*speedup*
density	*density*	*iter*	*seconds*	*iter*	*iter*	*iter*	*seconds*	*factor*
.02443	.25	1000*	3877.65	31	30	4	124.30	31.1958
.02443	.4	1000*	3884.70	41	40	5	168.92	22.9977
.02443	.6	75	297.75	43	40	22	216.43	1.3757
.02443	.7	143	565.20	74	70	62	442.02	1.2786
.02443	.8	300	1202.33	95	90	100	671.87	1.7895
.00799	.25	50	70.53	22	20	20	36.95	1.9088
.00799	.4	300	424.95	54	50	169	133.80	3.1760
.00799	.6	1000*	1426.93	59	50	401	338.10	4.2204
.00799	.8	1000*	1436.06	81	70	550	624.46	2.2996

* failed to converge after maximum iterations is reached

Table 4.3

COMPARISON OF SOR AND TSOR

M POSITIVE DEFINITE, DIMENSION 2500

		SOR		**TSOR**				
problem density	*solution density*	*iter*	*time seconds*	*total iter*	*SOR iter*	*inner iter*	*time seconds*	*speedup factor*
.01429	.25	1000*	3711.53	31	30	3	119.63	31.0251
.01429	.4	50	188.40	32	30	32	145.83	1.2919
.01429	.6	100	379.86	44	40	37	237.56	1.5989
.01429	.7	150	570.90	51	40	124	488.05	1.1697
.01429	.8	250	955.08	77	70	174	743.12	1.2852
.00477	.25	100	142.48	22	20	49	42.65	3.3407
.00477	.5	1000	1415.73	73	70	150	175.97	8.0454
.00477	.7	1000	1437.76	98	80	807	793.97	1.8108

Table 4.4

COMPARISON OF SOR AND TSOR

M POSITIVE DEFINITE, DIMENSION 5000

		SOR		**TSOR**				
problem density	*solution density*	*iter*	*time seconds*	*total iter*	*SOR iter*	*inner iter*	*time seconds*	*speedup factor*
.00376	.25	50	213.70	31	30	4	138.00	1.5485
.00376	.4	650	2743.88	47	40	350	501.98	5.4661
.00376	.6	450	1938.53	65	50	750	1548.47	1.2519
.00376	.8	400	1749.70	94	80	648	2210.8	.7914
.00191	.25	150	361.83	32	30	100	111.30	3.2509
.00191	.4	150	364.95	52	50	77	173.53	2.1031
.00191	.6	550	1343.76	100	80	920	1219.266	1.1021

* failed to converge after maximum iterations is reached

Table 4.5

COMPARISON OF SOR AND TSOR

***M* POSITIVE SEMIDEFINITE, DIMENSION 1000**

		SOR		**TSOR**				
problem density	*solution density*	*iter*	*time seconds*	*total iter*	*SOR iter*	*inner iter*	*time seconds*	*speeup factor*
.07106	.25	50	133.55	31	30	4	85.52	1.5616
.07106	.4	1000	2655.43	32	30	15	97.98	27.1009
.07106	.6	100	272.91	57	50	39	224.26	1.2169
.07106	.8	2000*	5509.71	121	110	453	1190.95	4.6263
.03162	.25	50	62.33	22	20	16	31.38	1.9861
.03162	.4	50	63.11	41	40	6	54.10	1.1665
.03162	.6	150	190.66	44	40	87	105.3833	1.8092
.03162	.8	1000*	1282.88	100	90	454	524.083	2.4478

Table 4.6

COMPARISON OF SOR AND TSOR

***M* POSITIVE SEMIDEFINITE, DIMENSION 2000**

		SOR		**TSOR**				
problem density	*solution density*	*iter*	*time seconds*	*total iter*	*SOR iter*	*inner iter*	*time seconds*	*speedup factor*
.01023	.25	300	522.08	51	50	42	98.21	5.3156
.01023	.4	350	611.50	65	60	214	189.78	3.2221
.01023	.6	1550	2751.82	106	90	751	725.35	3.7938
.01023	.8	4000*	7172.35	237	180	2801	3677.75	1.9502
.00393	.25	550	434.45	28	20	289	66.6166	6.5216
.00393	.4	1950	1558.08	89	50	1950	511.97	3.0433
.00393	.6	2000*	1611.56	112	80	1680	673.13	2.3941
.00393	.8	2000	1629.33	259	210	2439	1584.98	1.0279

* failed to converge after maximum iterations is reached

Table 4.7

COMPARISON OF SOR AND TSOR

M POSITIVE SEMIDEFINITE, DIMENSION 10000

		SOR		**TSOR**				
problem density	*solution density*	*iter*	*time seconds*	*total iter*	*SOR iter*	*inner iter*	*time seconds*	*speedup factor*
.00038	.25	4000*	10480.83	111	40	3533	2024.30	5.1770
.00038	.25	1900	4969.20	78	40	1900	1132.95	4.3860
.00038	.4	2000	5205.53	107	60	2350	2215.06	2.3500
.00129	.25	10000*	61560.50	73	40	1650	1716.06	35.8731
.00129	.40	7400	45667.65	164	120	2200	3994.11	11.4337

* failed to converge after maximum iterations is reached

References

Bertsekas, D. P. (1982) Projected Newton Methods for Optimization Problems with Simple Constraints, SIAM J. Control and Optimization, 20, 221–246.

Gill, P. E. and Murray, W. (1974) Methods for Large–scale Linearly Constrained Problems. *In* "Numerical Methods for Constrained Optimization" (Gill, Murray, eds.), 93–148. Academic Press, London and New York.

Goldfarb, D. (1972) Extensions of Newton's Method and Simplex Methods for Solving Quadratic Programs, *In* "Numerical Methods for Non–Linear Optimization" (F. A. Lootsma, ed.), 239–254. Academic Press, London and New York.

Hoyle, S. C. (1986) A Single–Phase Method for Quadratic Programming, Systems Optimization Laboratory Technical Report #86–9, Stanford University.

Lenard, M. L. (1979) A Computational Study of Active Set Strategies in Nonlinear Programming with Linear Constraints, Math. Prog. 16, 81-97.

Mangasarian, O. L. (1977) Solution of Symmetric Linear Complementarity Problems by Iterative Methods, Journal of Optimization Theory and Applications, 22,465–484.

Mangasarian, O. L. and De Leone, R. (1986) Parallel Gradient Projection Successive Overrelaxation for Symmetric Linear Complementarity Problems and Linear Programs, Computer Sciences Technical Report #659, University of Wisconsin–Madison.