An Algorithm for the Fast Solution of Symmetric Linear Complementarity Problems

José Luis Morales* Jorge Nocedal† Mikhail Smelyanskiy‡

August 23, 2008

Abstract

This paper studies algorithms for the solution of mixed symmetric linear complementarity problems. The goal is to compute fast and approximate solutions of medium to large sized problems, such as those arising in computer game simulations and American options pricing. The paper proposes an improvement of a method described by Kocvara and Zowe [19] that combines projected Gauss-Seidel iterations with subspace minimization steps. The proposed algorithm employs a recursive subspace minimization designed to handle severely ill-conditioned problems. Numerical tests indicate that the approach is more efficient than interior-point and gradient projection methods on some physical simulation problems that arise in computer game scenarios.

1 Introduction

Linear complementarity problems (LCPs) arise in a variety of applications [12, 13, 16] and several methods have been proposed for their solution [19, 17]. In this paper we are interested in applications where linear complementarity problems must be solved with strict time limitations and where approximate solutions are often acceptable. Applications of this kind arise in physical simulations for computer games [3] and in the pricing of American options [30].

*Departamento de Matemáticas, Instituto Tecnológico Autónomo de México, México. This author was supported by Asociación Mexicana de Cultura AC and CONACyT-NSF grant J110.388/2006.

†Department of Electrical Engineering and Computer Science, Northwestern University, USA. This author was supported by National Science Foundation grant CCF-0514772, Department of Energy grant DE-FG02-87ER25047-A004 and a grant from the Intel Corporation.

‡The Intel Corporation, Santa Clara, California, USA.

The form of the linear complementarity problem considered here is

$$Au + Cv + a = 0 \tag{1.1a}$$
$$C^T u + Bv + b \geq 0 \tag{1.1b}$$
$$v^T(C^T u + Bv + b) = 0 \tag{1.1c}$$
$$v \geq 0, \tag{1.1d}$$

where the variables of the problem are u and v; the matrices A, B, C and the vectors a, b are given. We assume that the matrix

$$Q = \begin{bmatrix} A & C \\ C^T & B \end{bmatrix}, \tag{1.2}$$

is a square symmetric matrix of dimension n, that the diagonal elements of B are positive and that the diagonal elements of A are nonzero. A problem of the form (1.1) is called a mixed linear complementarity problem.

The goal of this paper is to study algorithms that can quickly approximate the solution of medium-to-large LCPs. Although the need for real time solutions arises in various applications, our testing is driven by rigid body simulations in computer games, which must run at a rate of 30 frames per second. For each frame, several operations must be executed, including the physics simulation that invokes the solution of an LCP. Hence there are strict time limitations in the solution of the LCP (say 0.005 seconds) but the accuracy of the solution need not be high provided a significant proportion of the optimal active set components are identified.

The earliest methods for solving linear complementarity problems were pivotal methods, such as Lemke's algorithm [12]. Although these methods are efficient for small problems, they do not exploit sparsity effectively and are unsuitable for large-scale applications [22]. Projected iterative methods, including the projected Gauss-Seidel (PGS) and projected SOR methods, have become popular for physics simulations in computer games [9, 3]. They have a low computational cost per iteration and have been shown to be convergent under certain conditions [13]. Interior-point methods [31] constitute an important alternative as they are effective for solving large and ill-conditioned LCPs; their main drawbacks are the high cost of each iteration and the fact that they may not yield a good estimate of the solution when terminated early. The last class of methods for solving large LCPs studied in this paper are gradient projection methods [11, 8, 4] for bound constrained optimization—a problem that is closely related to the linear complementarity problem (1.1) when Q is positive definite.

There are some useful reformulations of problem (1.1). Introducing a slack variable w we obtain the system

$$Au + Cv + a = 0 \tag{1.3a}$$
$$C^T u + Bv + b = w \tag{1.3b}$$
$$v^T w = 0 \tag{1.3c}$$
$$v, w \geq 0. \tag{1.3d}$$

Note also that (1.3) are the first-order optimality conditions of the bound constrained quadratic program

$$\begin{array}{ll} \min\limits_{z} & \phi(z) = \dfrac{1}{2} z^T Q z + q^T z \\ \text{s.t.} & v \geq 0, \end{array} \tag{1.4}$$

where Q is defined by (1.2) and

$$z = \begin{bmatrix} u \\ v \end{bmatrix}, \qquad q = \begin{bmatrix} a \\ b \end{bmatrix}.$$

If Q is positive definite, (1.4) is a strictly convex quadratic program and its (unique) primal-dual solution (u^*, v^*) can be computed by solving the LCP (1.3), and vice versa. This establishes the equivalence between the two problems in the symmetric positive definite case. The quadratic programming formulation (1.4) has the advantage that many efficient algorithms have been developed to compute its solution, such as gradient projection methods.

Ferris, Wathen and Armand [17] have recently studied the performance of various LCP algorithms in the context of computer game simulations. They recommend an interior-point method with a primal/dual switching mechanism designed to reduce the memory requirements during the solution of linear systems. Ferris et al. did not, however, study projected Gauss-Seidel methods, which are perhaps the most popular techniques for solving LCPs arising in computer game simulations.

In this paper, we study a method for LCP that alternates PGS iterations and subspace minimization steps. It can be viewed as a compromise between the inexpensive PGS method, which can be very slow, and interior-point methods, which require the solution of expensive linear systems. The PGS iteration is able to quickly produce a good estimate of the optimal active set, while the subspace minimization steps refine this estimate and permit a fast rate of convergence. The proposed method is an improvement of the algorithm by Kocvara and Zowe [19] in that we perform a more thorough subspace minimization phase. By repeatedly minimizing in a subspace of decreasing dimension (until the active set estimate no longer changes), our algorithm is able to make a quick identification of the optimal active set, even on very ill-conditioned problems.

The paper is organized as follows. Section 2 discusses the projected Gauss-Seidel method and presents some numerical results that illustrate its strengths and weaknesses. The proposed method is described in §3. Numerical experiments are reported in §4 and some concluding remarks are made in §5.

Notation. The elements of a matrix B are denoted by B_{ij} and the components of a vector b are denoted by b_i. We use $b \geq 0$ to indicate that all components of b satisfy $b_i \geq 0$. Superindices denote iteration numbers. The matrix A in (1.1) is of dimension $n_a \times n_a$, the matrix B is of dimension $n_b \times n_b$, and we define $n = n_a + n_b$. Throughout the paper, $||\cdot||$ denotes the infinity norm.

2 The Projected Gauss-Seidel Method

Iterative projection methods for linear complementarity were proposed in the 1970s and their interest has increased recently with their application in physics simulations for computer games and in finance. A comprehensive treatment of projected iterative methods is given in Cottle, Pang and Stone [13], but for the sake of clarity we derive them here in the context of the mixed linear complementarity problem (1.1).

We first define matrix splittings of the matrices A and B in (1.1). Let

$$A = A_L + A_D + A_U \quad \text{and} \quad B = B_L + B_D + B_U,$$

where A_L is a strictly lower triangular matrix whose nonzero elements are given by the lower triangular part of A; A_D is a diagonal matrix whose nonzero entries are given by the diagonal of A, and A_U is a strictly upper triangular matrix containing the remaining elements of A (and similarly for B_L, B_D and B_U).

Given u^k and $v^k \geq 0$, an approximate solution to (1.1), consider the auxiliary linear complementarity problem

$$(A_L + A_D)u + A_U u^k + Cv^k + a = 0 \tag{2.1a}$$
$$C^T u + (B_L + B_D)v + B_U v^k + b \geq 0 \tag{2.1b}$$
$$v^T(C^T u + (B_L + B_D)v + B_U v^k + b) = 0 \tag{2.1c}$$
$$v \geq 0. \tag{2.1d}$$

We define the new iterate (u^{k+1}, v^{k+1}) as a solution of (2.1). We have from (2.1a)

$$A_D u^{k+1} = -a - A_L u^{k+1} - A_U u^k - Cv^k,$$

or in scalar form, for $i = 1, 2, \ldots, n_a$,

$$\begin{aligned} u_i^{k+1} &= u_i^k - \Delta u_i, \\ \Delta u_i &= \frac{1}{A_{ii}} \left(a_i + \sum_{j<i} A_{ij} u_j^{k+1} + \sum_{j \geq i} A_{ij} u_j^k + \sum_{j=1}^{n_b} C_{ij} v_j^k \right). \end{aligned} \tag{2.2}$$

Formula (2.2) is well defined since by assumption $A_{ii} \neq 0, i = 1, ..., n$. Having thus determined u^{k+1}, we now find a solution v^{k+1} of (2.1). Let us define

$$\hat{b} = b + C^T u^{k+1} + B_U v^k.$$

Then we can write (2.1b)-(2.1d) as

$$\hat{b} + (B_L + B_D)v \geq 0 \tag{2.3a}$$
$$v_i[\hat{b} + (B_L + B_D)v]_i = 0, \quad i = 1, 2, \ldots, n_b \tag{2.3b}$$
$$v \geq 0. \tag{2.3c}$$

Assuming that v_j^{k+1} has been computed for $j < i$, we can satisfy (2.3a)-(2.3b) by defining the ith component of the solution v_i^{k+1} so that

$$[(B_L + B_D)v^{k+1}]_i = -\hat{b}_i;$$

or

$$\begin{aligned} v_i^{k+1} &= v_i^k - \Delta v_i \\ \Delta v_i &= \frac{1}{B_{ii}} \left(b_i + \sum_{j<i} B_{ij} v_j^{k+1} + \sum_{j \geq i} B_{ij} v_j^k + \sum_{j=1}^{n_a} C_{ji} u_j^{k+1} \right). \end{aligned} \tag{2.4}$$

We cannot guarantee that $v_i^{k+1} \geq 0$, but if it is not, we can simply set $v_i^{k+1} = 0$. By doing so, we still satisfy the ith equation in (2.3a) by positive definiteness of the diagonal matrix B_D and the fact that we increased v_i^{k+1} from a level at which the ith eqution in (2.3a) was zero. Clearly, we will also satisfy the ith equation in (2.3b). In summary, we define

$$v_i^{k+1} = \max\left(0, v_i^k - \Delta v_i\right), \quad i = 1, 2, \ldots, n_b. \tag{2.5}$$

Combining (2.2), (2.4) and (2.5) we obtain the Projected Gauss-Seidel (PGS) method for solving the mixed LCP (1.1).

Algorithm 2.1 Projected Gauss-Seidel (PGS) Method

Initialize $u^0, v^0 \geq 0$*; set* $k \leftarrow 0$.
repeat *until a stop test for (1.1) is satisfied:*
  for $i = 1, \ldots, n_a$
    $\Delta u_i = \frac{1}{A_{ii}} \left(a_i + \sum_{j<i} A_{ij} u_j^{k+1} + \sum_{j \geq i} A_{ij} u_j^k + \sum_{j=1}^{n_b} C_{ij} v_j^k \right)$;
    $u_i^{k+1} = u_i^k - \Delta u_i$;
  end
  for $i = 1, \ldots, n_b$
    $\Delta v_i = \frac{1}{B_{ii}} \left(b_i + \sum_{j<i} B_{ij} v_j^{k+1} + \sum_{j \geq i} B_{ij} v_j^k + \sum_{j=1}^{n_a} C_{ji} u_j^{k+1} \right)$;
    $v_i^{k+1} = \max\{0, v_i^k - \Delta v_i\}$;
  end
  $k \leftarrow k + 1$
end repeat

One way to define the error in the solution of (1.1) is through the residuals

$$\rho_a^k = ||Au^k + Cv^k + a|| \tag{2.6a}$$

$$\rho_b^k = \min_i \{v_i^k, (C^T u^k + Bv^k + b)_i\} \tag{2.6b}$$

$$\rho_c^k = \max_i \{0, -(C^T u^k + Bv^k + b)_i\}. \tag{2.6c}$$

It has been shown (see e.g., [13]) that the projected Gauss-Seidel method is globally convergent if Q is symmetric positive definite.

In order to assess the effectiveness of the PGS method, we conducted experiments using several problems generated with the Open Dynamics Engine [29], an open source package for game physics simulations. We extracted mixed LCPs that correspond to a well known problem of destruction. The PGS method was implemented in Fortran 77, in double precision; we terminate it when

$$r_1(k) = \max\left(\frac{\rho_a^k}{1+||a||}, \frac{\rho_b^k}{1+||b||}, \frac{\rho_c^k}{1+||b||^2}, \right) \leq tol, \tag{2.7}$$

for some positive constant tol.

The results of the experiments are given in Table 1. For each problem we report: a) n, the number of variables; b) nz(Q), the number of nonzeros in Q; c) cond(Q), the condition number of Q; d) the number of iterations of the projected Gauss-Seidel method for two values of the convergence tolerance: $tol = 10^{-1}, 10^{-2}$. The name of each problem indicates the number of bodies and constraints in the physics simulation; for example, the first problem has 7 bodies and 18 constraints. In all the problems, Q is positive definite. It is clear from Table 1 that the projected Gauss-Seidel method requires an excessive number of iterations for $tol = 10^{-2}$ on some of the large problems (we report CPU times in §4).

name	n	nz(Q)	cond(Q)	10^{-1}	10^{-2}
nb7_nc18	18	162	5.83e+01	4	6
nb8_nc45	45	779	2.92e+03	17	120
nb8_nc48	48	868	2.38e+03	17	111
nb235_nc1044	1 044	14 211	4.58e+04	61	312
nb449_nc1821	1 821	28 010	4.22e+04	132	414
nb907_nc5832	5 832	176 735	5.11e+07	21	16 785
nb948_nc7344	7 344	269 765	9.02e+07	3 123	>50 000
nb966_nc8220	8 220	368 604	9.19e+07	1 601	39 103
nb976_nc8745	8 745	373 848	6.45e+07	7 184	>50 000
nb977_nc9534	9 534	494 118	1.03e+08	1 246	>50 000

Table 1: Number of iterations (last two columns) of the Projected Gauss-Seidel Method for two values of the stopping tolerance: $tol = 10^{-1}, 10^{-2}$.

Given this poor performance, it may seem surprising that the projected Gauss-Seidel method is popular in computer game software. A possible explanation is given in Table 2, which shows that this method often provides a good estimate of the active set after only a few iterations. In Table 2, we report the ratio of active-set components correctly identified after k iterations of the projected Gauss-Seidel iteration over the total number of components in the optimal active set, for 5 values of k. Note that the first few iterations identify a large percentage of the components, but for the ill-conditioned problems the improvement is very slow (and sometimes erratic).

name	$k = 2$	$k = 20$	$k = 100$	$k = 1\,000$	$k = 10\,000$
nb7_nc18	3/4	4/4			
nb8_nc45	7/8	7/8	8/8		
nb8_nc48	8/10	8/10	10/10		
nb235_nc1044	12/58	40/58	52/58	58/58	
nb449_nc1821	156/254	233/254	248/254	254/254	
nb907_nc5832	1 253/1 512	1 301/1 512	1 333/1 512	1 399/1 512	1 471/1 512
nb948_nc7344	1 504/1 828	1 523/1 828	1 542/1 828	1 614/1 828	1 707/1 828
nb966_nc8220	2 112/2 321	2 106/2 321	2 109/2 321	2 178/2 321	2 253/2 321
nb976_nc8745	1 728/2 158	1 743/2 158	1 758/2 158	1 870/2 158	1 976/2 158
nb977_nc9534	2 513/2 728	2 495/2 728	2 505/2 728	2 578/2 728	2 670/2 728

Table 2: Fraction of correct active constraints identified by the PGS method after k iterations.

We therefore ask whether it is possible to design an algorithm that exploits the active-set identification qualities of the projected Gauss-Seidel iteration, but is able to return a highly accurate estimate solution (if needed) in a reasonable amount of time. The method described in the next section aims to achieve these goals.

3 The Proposed Method

The algorithm we discuss in this section performs two types of iterations. First, the projected Gauss-Seidel (PGS) method generates an estimate of the optimal active set. Starting with this estimate, a sequence of subspace minimization steps is performed; these steps aim at optimality and also produce a new estimate of the active set. The cycle of PGS and subspace minimization iterations is repeated until an acceptable solution of the linear complementarity problem (1.1) is found.

To describe this approach more precisely, suppose that after a few iterations of the projected Gauss-Seidel method, we find that the first m components of v are zero. Following an active-set approach, we fix these variables at zero (i.e., we temporarily remove them from the problem) and retain the remaining components, which we denote as $\hat{v}$. We then ask what, if any, equations or inequalities can be eliminated from (1.1). An answer is that to preserve symmetry, we should eliminate the first m inequalities from (1.1b). This yields the reduced problem

$$Au + \hat{C}\hat{v} + a = 0 \tag{3.1a}$$

$$\hat{C}^T u + \hat{B}\hat{v} + \hat{b} \geq 0 \tag{3.1b}$$

$$\hat{v}^T(\hat{C}^T u + \hat{B}\hat{v} + \hat{b}) = 0 \tag{3.1c}$$

$$\hat{v} \geq 0, \tag{3.1d}$$

where

$$\hat{C} = CP^T, \quad \hat{B} = PBP^T, \quad \hat{b} = Pb, \qquad \hat{v} = Pv, \tag{3.2}$$

and $P = [\,0\,|\,I_{n_b-m}]$. Since the active set approach has $\hat{v} > 0$, we set the term $\hat{C}^T u + \hat{B}\hat{v} + \hat{b}$ in (3.1c) to zero. Thus (3.1) reduces to the square system

$$Au + \hat{C}\hat{v} + a = 0 \tag{3.3a}$$

$$\hat{C}^T u + \hat{B}\hat{v} + \hat{b} = 0, \tag{3.3b}$$

together with the condition $\hat{v} \geq 0$. We compute an approximate solution of this problem by solving (3.3) and then projecting the solution $\hat{v}$ onto the nonnegative orthant through the assignment

$$\hat{v} \leftarrow \max(0, \hat{v}). \tag{3.4}$$

The components of $\hat{v}$ that are set to zero by the projection operation (3.4) are then removed to obtain a new subspace minimization problem of smaller dimension. The corresponding reduced system (3.3) is formed and solved. We repeat this subspace minimization procedure until the solution of (3.3) contains no negative components in v or until a prescribed maximum number of subspace iterations is performed. In either case the procedure terminates with a point that we denote as $z^s = (u^s, v^s)$. (Here and in what follows, z denotes a vector in $\mathbb{R}^n$ where the componenent v is obtained from the reduced vector $\hat{v}$ by appending an appropriate number of zeros to it.)

One extra ingredient is added in the algorithm to guarantee global convergence (in the positive definite case). We compute a safeguarding point z^b by backtracking (instead of projecting) from the first subspace solution to the feasible region $v \geq 0$. More precisely, we compute

$$z^b = z^0 + \alpha(z^1 - z^0), \tag{3.5}$$

where z^0 is the initial point in the subspace minimization phase, z^1 is the point obtained by solving the system (3.3) for the first time (and before applying the projection (3.4)), and $0 < \alpha \leq 1$ is the largest steplength such that $v^b \geq 0$. We store z^b and perform the repeated subspace minimization phase described above to obtain the point z^s. We then choose between z^b and z^s, depending on which one is preferable by some metric, such as the optimality measure (2.6).

However, in the case when Q is positive definite, it is natural to use the quadratic function ϕ given in (1.4) as the metric. Recall from §1, that the linear complementarity problem (1.1) and the bound constrained quadratic program (1.4) are equivalent when Q is positive definite. It is easy to explain the utility of the point z^b by interpreting the solution of (3.3)-(3.4) in the context of this quadratic program. Suppose, once more, that we have removed the first m components of v and that $\hat{v}$ are the remaining components. We then define the reduced quadratic program

$$\begin{aligned} \min_{\hat{z}} \quad & \hat{\phi}(\hat{z}) = \frac{1}{2}\hat{z}^T\hat{Q}\hat{z} + \hat{q}^T\hat{z} \\ \text{s.t.} \quad & \hat{v} \geq 0, \end{aligned} \tag{3.6}$$

where $\hat{z} = (u, \hat{v})$ and $\hat{Q}, \hat{q}$ are defined accordingly by means of the matrix P. The unconstrained minimizer of the objective in (3.6) coincides with the solution of (3.3), which justifies

the use of the term "subspace minimization" in our earlier discussion. Note, however, that by projecting the solution of (3.3) through (3.4) the value of the quadratic function $\hat{\phi}$ may be higher than at the initial point z^0 of the subspace minimization, which can cause the algorithm to fail. This difficulty is easily avoided by falling back on the point z^b, which is guaranteed to give $\phi(z^b) \leq \phi(z^0)$. In this way, we ensure that the subspace minimization does not undo the progress made in the PGS phase. In this case we replace z^s by z^b if $\phi(z^b) < \phi(z^s)$.

The proposed algorithm is summarized as follows for the case when Q is positive definite.

Algorithm 3.1 Projected Gauss-Seidel with Subspace Minimization

Initialize $u, v \geq 0$. *Choose a constant* $tol > 0$.
repeat *until a convergence test for problem* (1.1) *is satisfied*
  Perform k_{gs} *iterations of the projected Gauss-Seidel iteration (see Algorithm 2.1) to obtain an iterate* $z^0 = (u, v)$;
  Set `isub` $\leftarrow 1$;
  repeat *at most* k_{sm} *times (subspace minimization)*
    Define $\hat{v}$ *to be the subvector of* v *whose components satisfy* $v_i > tol$;
    Form and solve the reduced system (3.3) to obtain z^1;
    if `isub` $= 1$ *compute* z^b *by (3.5);*
    Project the solution of (3.3) by means of (3.4);
    Denote the new iterate of the subspace minimization by $z^s = (u^s, v^s)$;
    If the solution component $\hat{v}$ *satisfies* $\hat{v} > tol$, **break**;
    Set `isub` $\leftarrow 0$;
  end repeat
  If $\phi(z^s) > \phi(z^b)$ *then set* $z^s \leftarrow z^b$;
  Define the new iterate of the algorithm as $z = (u, v) \leftarrow z^s$
end repeat

Practical ranges for the parameters are $k_{gs} \in [2, 5]$ and $k_{sm} \in [3, 5]$. We could replace the projected Gauss-Seidel iteration by the projected SOR iteration [13], but even when the SOR parameter is tuned to the application, we can expect only a marginal improvement of performance, for the reasons discussed below.

The algorithm of Kocvara and Zowe [19] is a special case of Algorithm 3.1 where only one subspace minimization step is performed, i.e., $k_{sm} = 1$. Another important difference is that Kocvara and Zowe always use the backtracking procedure (3.5), instead of first trying the projection point (3.4), which is often much more effective in the identification of the optimal active set. The refinements introduced in our algorithm are crucial; without them the method can be very inefficient on ill-conditioned problems (as we report in §4). Kocvara and Zowe observed these inefficiencies and attempt to overcome them by devising a procedure for computing a good initial point and employing certain heuristics in the line search procedure.

The subspace minimization could be performed using a procedure that guarantees a steady decrease in ϕ. For example, Lin and Moré [21] perform an approximate minimization of ϕ along the path generated by projecting the subspace minimization point z^1 onto the nonnegative orthant $v \geq 0$. Their procedure can produce a better iterate than the subspace minimization of Algorithm 3.1, but our nonmonotone approach has the advantage that the projection (3.4) can improve the active set estimate very quickly and that the computational cost of the iteration can easily be controlled through the parameter k_{sm}. In our tests, it was never necessary for Algorithm 3.1 to revert to the safeguarding point z^b defined in (3.5), despite the fact that, on occasion, ϕ increased after the first step of the subspace minimization. Thus, our approach is indeed nonmonotone.

Algorithm 3.1 is well defined even if Q is not positive definite; all we need to assume is that the diagonal elements of B are positive and the diagonal elements of A are nonzero. The function ϕ would then be defined as the optimality measure of the problem, as mentioned above. However, when Q is not positive definite the global convergence of the algorithm cannot be guaranteed.

4 Numerical experiments

We first compare three methods that work directly on the mixed linear complementarity problem: the projected Gauss-Seidel method (Algorithm 2.1), the method proposed in this paper (Algorithm 3.1) and an interior-point method.

The interior-point method is a primal-dual algorithm for LCPs applied to the system (1.3). We use separate steplengths for the variables v and w. The centering parameter is set to $\sigma = 0.25$ throughout the iteration and the fraction to the boundary parameter is defined as $\tau = 0.9995$; see Wright [31] for a description of the algorithm and the meaning of these parameters. The stopping condition for the interior-point method is defined in terms of the residuals

$$r_a^k = Au^k + Cv^k + a, \qquad r_b^k = C^T u^k + Bv^k - w^k + b, \qquad r_c^k = V^k w^k, \tag{4.1}$$

where V is a diagonal matrix whose nonzero entries are given by the components of v.

In order to make the numerical comparisons as accurate as possible, we have implemented these three methods in a Fortran 77 package (using double precision), and employ the same linear algebra solver, namely PARDISO [26, 27], to solve the linear systems arising in Algorithms 3.1 and the interior-point method. The latter performs the symbolic factorization only once, whereas Algorithm 3.1 does so at every subspace iteration. The package was compiled using GNU f77 and all computations were performed on a Pentium 4 DELL PRECISION 340.

In our implementation of Algorithm 3.1, we performed $k_{gs} = 5$ PGS iterations before invoking the subspace minimization phase, which was repeated at most $k_{sm} = 3$ times. The initial point in Algorithm 3.1 and the PGS method was chosen as $u^0 = 0$, $v^0 = 0$, and for the interior-point method it was set to $u^0 = 0$, $v^0 = 0.1$, $w^0 = 1$. To terminate the interior-point

method, we define the error

$$r_2(k) = \max\left(\frac{||r_a^k||}{1+||a||}, \frac{||r_b^k||}{1+||b||}, \frac{||r_c^k||}{1+||b||^2},\right) \tag{4.2}$$

where r_a, r_b, r_c are given in (4.1). For the PGS method and Algorithm 3.1 we use the error measure (2.7). In the first set of experiments, the algorithms were terminated when $r_1(k)$ and $r_2(k)$ were less than 10^{-8}.

Table 3 compares the performance of the interior-point method and Algorithm 3.1. Column 2 shows CPU time in seconds; column 3 reports the maximum number of nonzeros in the Cholesky factors throughout the run; and column 4 reports the number of times that a Cholesky factorization was computed. The first number in columns 2-4 corresponds to the interior-point method and the second to Algorithm 3.1. We report results only for the five largest problems in Table 1. Algorithm 3.1 is clearly faster and requires less storage than the interior-point algorithm in these tests.

name	cpu time	nz(L)	# Chol. fact.
nb907_nc5832	0.50/0.22	216 080/114 864	17/7
nb948_nc7344	1.02/0.46	406 152/218 522	18/9
nb966_nc8220	2.40/0.63	797 989/398 821	16/7
nb976_nc8745	1.79/0.66	646 929/341 911	19/9
nb977_nc9534	4.67/0.87	1 222 209/604 892	17/6

Table 3: Performance of the interior-point method (first number) and the proposed method (second number)

To illustrate the importance of performing a thorough subspace minimization, we solved problem nb_907_5832 using only one subspace minimization cycle in Algorithm 3.1, i.e., setting $k_{sm} = 1$. The algorithm required 387 iterations (compared with 7 iterations for Algorithm 3.1); the first 385 iterations fall back on the safeguarding point (3.5). Even greater inefficiencies were observed for the other problems in Table 3.

Next, we compare the performance of the PGS method, the interior-point method (IPM) and Algorithm 3.1 for various levels of accuracy in the stopping test. Figures 1-3 plot the errors (2.7), (4.2) as a function of CPU time for the last 3 problems in Table 3. Note that Algorithm 3.1 is quite effective when terminated early; it seems to strike the right balance between identifying the active set and moving toward optimality.

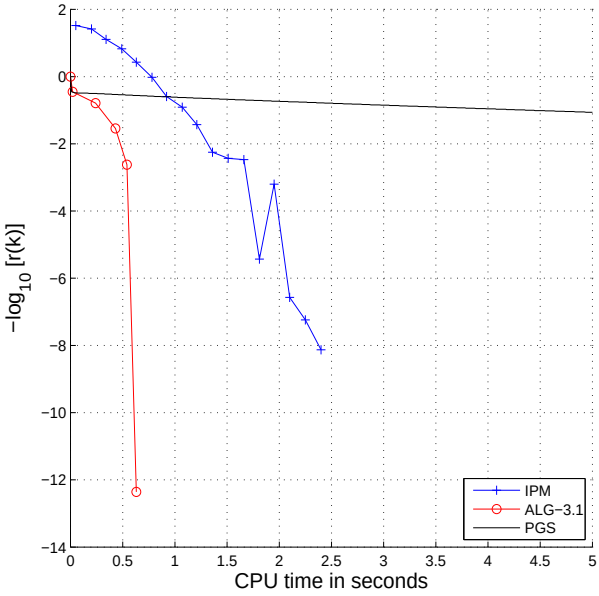

Figure 1: Problem nb976_nc8220

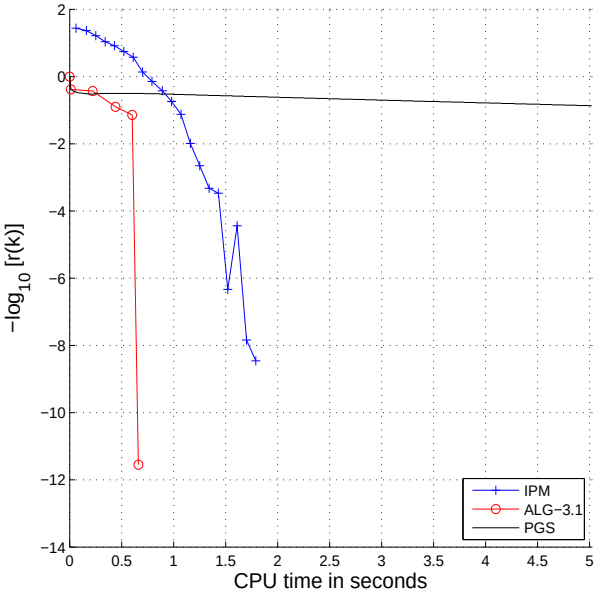

Figure 2: Problem nb976_nc8745

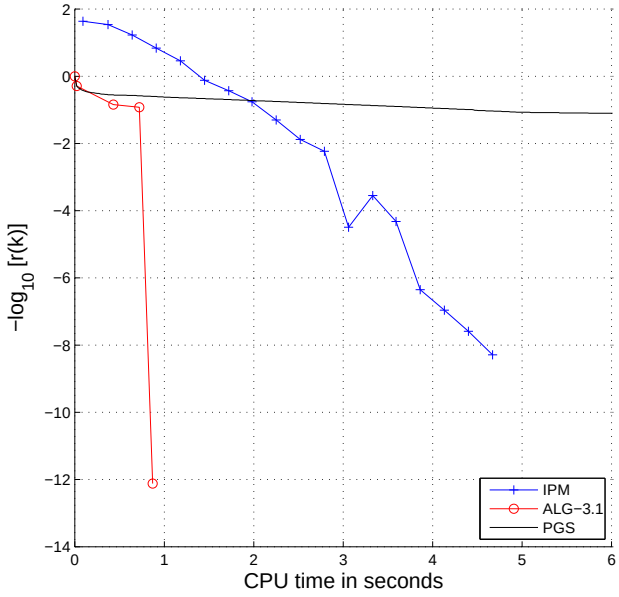

Figure 3: Problem `nb977_nc9534`. Error as a function of CPU time.

In a second set of experiments, we compare Algorithm 3.1 and the PGS method to a gradient projection method. Since Q is positive definite for all the problems in Table 1, we can solve the LCP by applying the gradient projection method to the quadratic program (1.4). A great variety of gradient projection methods have been developed in the last 20 years; some variants [6, 7, 5, 14, 18] perform only gradient projection steps, while others [24, 10, 23, 32, 21] alternate gradient projection and subspace minimization steps.

For our tests we have chosen TRON, a well established code by Lin and Moré [21] that has been tested against other gradient projection methods [18, 5]. The algorithm in TRON alternates one gradient projection step and a (repeated) subspace minimization phase, which is performed using a conjugate gradient iteration preconditioned by a (modified) incomplete Cholesky factorization.

Numerical results comparing TRON, Algorithm 3.1 and the PGS method are displayed in Table 4. For TRON, we report 3 numbers: the number of factorizations, CPU time, and an adjusted CPU time that excludes Hessian evaluation times after the first iteration. TRON is designed for nonlinear objective functions and evaluates the Hessian at every iteration, However, for quadratic programs this evaluation needs to be done only once and therefore we must account for this extra cost. For Algorithm 3.1 we used three parameter settings: $(k_{gs} = 2, k_{sm} = 3)$, $(k_{gs} = 5, k_{sm} = 3)$, and $(k_{gs} = 5, k_{sm} = 2)$. We report only CPU time for the PGS method.

name	TRON	PGS(2)-SM(3)	PGS(5)-SM(3)	PGS(5)-SM(2)	PGS
	nfact/cpu1/cpu2	nfact/cpu	nfact/cpu	nfact/cpu	cpu
nb235_nc1044	9/0.50/.03	6/.03	6/.03	5/.02	1.41
nb449_nc1821	8/0.11/.06	6/.04	5/.03	5/.03	0.72
nb907_nc5832	100/35.0/22.3	9/.30	7/.22	7/.24	>300
nb948_nc7344	182/133./91.0	9/.48	9/.46	7/.39	>300
nb966_nc8220	191/143./67.8	7/.64	7/.63	6/.59	>300
nb976_nc8745	491/440./260.	10/.73	9/.66	7/.55	>300
nb977_nc9534	229/326./186.	6/.88	6/.87	6/.89	>300
DPJB_1_10 000	19/1.36/1.12	7/.33	6/.30	6/.30	6.49

Table 4: Performance of TRON, three variants of Algorithm 3.1, and the PGS method.

We include one additional problem in Table 4, namely DPJB_1 from the COPS collection [15], with $n = 10^4$ variables. Although DPJB_1 is the most difficult version ($\epsilon = .1$) of the journal bearing problem in the COPS collection, we observe from Table 4 that it is easily solved by all methods.

We note from Table 4 that the performance of Algorithm 3.1 is not very sensitive to the choice of the parameters k_{gs} and k_{sm}. In our experience, $k_{gs} \in [2, 5]$ yields good results, and we have observed that the subspace iteration limit $k_{sm} = 3$ was reached only occasionally, on the most ill-conditioned problems. Nevertheless, the parameters k_{gs}, k_{sm} should be tuned for the application, particularly on those requiring real-time solutions.

These results also indicate that little would be gained by replacing the projected Gauss-Seidel iteration by the projected SOR iteration, given that we have found it best to perform at most 5 iterations of the projected iterative method. Since the Gauss-Seidel method does not require the selection of parameters, we find it preferable to the SOR method.

Note that TRON requires a very large number of iterations for the ill-conditioned problems (i.e. the last 5 problems in Table 1) and requires significantly more time than Algorithm 3.1. This is due to the fact that the subspace minimization in TRON is not able to produce an accurate estimate of the active set and, in addition, that it requires between 4 and 10 times more CPU time than the subspace minimization in Algorithm 3.1.

We have not made comparisons with the algorithm of Kocvara and Zowe because their numerical results report only moderate improvements in performance over the gradient projection method GPCG [23]) that does not employ preconditioning. Our results, on the other hand, show dramatic improvements over the more powerful gradient projection code TRON which uses an incomplete Cholesky preconditioner.

5 Final Remarks

In this paper we have proposed an algorithm for solving symmetric mixed linear complementarity problems (and as a special case, convex bound constrained quadratic programs) that places emphasis on the speed and accuracy of the subspace minimization phase. The algorithm is designed for medium to large real-time applications and our tests suggest that it has far better performance on ill-conditioned problems than competing methods. The proposed algorithm has much flexibility since the number of projected Gauss-Seidel and subspace minimization iterations can be adapted to the requirements of the application at hand.

Another advantage of the proposed method is that it can exploit parallelism well. The projected Gauss-Seidel method, which is popular in computer game software and American options pricing, scales poorly with the number of processors due to dependencies between the equations. For example, Kumar et al. simulated the performance of a PGS step on a 64-core INTEL chip multiprocessor simulator and observed less than 15% utilization [20]. On the other hand, Algorithm 3.1 typically spends only 10% of the time in the PGS phase and 90% in the subspace minimization. Hence it effectively offloads the parallelization task to a direct linear solver that is known to parallelize well [28]. Specifically, our adaptation of PARDISO, the Cholesky solver used for the experiments in this paper, yields more than 65% utilization on a 64-core chip multiprocessor simulator (see also Schenk [25] for a scalability study for a moderate number of processors on a coarsely coupled shared memory system).

There has been a recent emergence of accelerated physics solutions in the computer industry. Ageia has developed a stand-alone physics processing unit [1], while graphics processing unit makers ATI and Nvidia have introduced new physics engines that run on their chips [2]. These approaches use massively parallel architectures that can benefit from the features of Algorithm 3.1. In future work, we intend to perform further quantitative studies of Algorithm 3.1 on various emerging chip multiprocessors.

References

[1] Ageia physx. Web page, `http://www.ageia.com/ physx/` , 2006.

[2] HavokFX. Web page, `http://www.havok.com/ content/view/187/77/` , 2006.

[3] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14::231–247, 1997.

[4] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2):221–246, 1982.

[5] E G. Birgin and J. M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Comput. Optim. Appl.*, 23:101—125, 2002.

[6] E G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIOPT*, 10:1196—1211, 2000.

[7] E G. Birgin, J. M. Martínez, and M. Raydan. Algorithm 813: Spg–software for convex-constrained optimization. *ACM Trans. Math. Software*, 27:340—349, 2001.

[8] J. V. Burke and J. J. Moré. On the identification of active constraints. *SIAM Journal on Numerical Analysis*, 25(5):1197–1211, 1988.

[9] E. Catto. Iterative dynamics with temporal coherence. Technical report, Crystal Dynamics, Menlo Park, California, 2005.

[10] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(182):433–460, 1988. See also same journal 26:764–767, 1989.

[11] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for Large-scale Nonlinear Optimization (Release A)*. Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[12] R.W. Cottle and G.B. Dantzig. Complementarity pivot theory of mathematical programming. *J. Linear Algebra Applns.*, 1:103–125, 1968.

[13] R.W. Cottle, J-S Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[14] Y. H. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numer. Math.*, 100, 2005.

[15] E. D. Dolan, J. J. Moré, and T. S. Munson. Benchmarking optimization software with COPS 3.0. Technical Report ANL/MCS-TM-273, Argonne National Laboratory, Argonne, Illinois, USA, 2004.

[16] M. C. Ferris and J. S. Pang. Engineering and economic applications of complementarity problems. *SIAM Review*, 39(4):669–713, 1997.

[17] M.C. Ferris, A.J. Wathen, and P. Armand. Limited memory solution of bound constrained convex quadratic problems arising in video games. *RAIRO- Operations Research*, 41:19–34, 2007.

[18] W. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIOPT*, 17(2):526–557, 2007.

[19] M. Kocvara and J. Zowe. An iterative two-step algorithm for linear complementarity problems. *Numerische Mathematik*, 68:95–106, 1994.

[20] S. Kumar, C. J. Hughes, and A. Nguyen. Carbon: Architectural support for fine-grained parallelism on chip multiprocessors. In *Proceedings of IEEE/ACM International Symposium on Computer Architecture (ISCA), San Diego, California*, 2007.

[21] C. J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

[22] J.L Morales and R.W.H. Sargent. Computational experience with several methods for large-scale convex quadratic programming. *Aportaciones Matemáticas. Comunicaciones*, 14:141–158, 1994.

[23] J. J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113, 1991.

[24] D. P. O'Leary. A generalized conjugate gradient algorithm for solving a class of quadratic programming problems. *Linear Algebra and its Applications*, 34:371–399, 1980.

[25] O. Schenk. *Scalable parallel sparse LU factorization methods on shared memory multiprocessors*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 2000.

[26] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Journal of Future Generation Computer Systems*, 20(3):475–487, 2004.

[27] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite systems. *Elec. Trans. Numer. Anal.*, 23:158–179, 2006.

[28] M. Smelyanskiy, V. W. Lee, D. Kim, A. D. Nguyen, and P. Dubey. Scaling performance of interior-point methods on a large-scale chip multiprocessor system. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, 2007.

[29] R. Smith. Open dynamics engine. Technical report, 2004. http://www.ode.org.

[30] P. Wilmott. *Quantitative Finance*. J. Wiley and sons, 2006.

[31] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, USA, 1997.

[32] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 78: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.