A recursive semi-smooth Newton method for linear complementarity problems[*]

Philipp Hungerländer[†] Joaquim Júdice[‡] Franz Rendl[§]

December 16, 2016

Abstract

A primal feasible active set method is presented for finding the unique solution of a Linear Complementarity Problem (LCP) with a P-matrix, which extends the globally convergent active set method for strictly convex quadratic problems with simple bounds proposed by Hungerländer and Rendl [16]. Based on a guess of the active set, a primal-dual pair (x,α) is computed that satisfies stationarity and the complementary condition. If x is not feasible, the variables connected to the infeasibilities are added to the active set and a new primal-dual pair (x,α) is computed. This process is iterated until a primal feasible solution is generated. Then a new active set is determined based on the feasibility information of the dual variable α. We prove that the algorithm stops after a finite number of steps with the unique solution of the LCP. An extension of the algorithm with similar convergence properties is also introduced for finding the unique solution of the Bound Linear Complementarity Problem (BLCP) with a P-matrix. Computational experience indicates that these approaches are very efficient for solving large-scale LCPs and BLCPs in practice.

Key words. primal-dual active set methods, semismooth Newton methods, linear complementarity problem, convex quadratic programming, global convergence.

1 Introduction

Given a $n \times n$ matrix Q and a vector $q \in \mathbb{R}^n$, the Linear Complementarity Problem (LCP) consists of finding vectors $x, \alpha \in \mathbb{R}^n$ such that

$$Qx + q = \alpha, \tag{1a}$$
$$x^\top \alpha = 0, \tag{1b}$$
$$x \geq 0,\ \alpha \geq 0. \tag{1c}$$

We assume that Q is a P-matrix, i.e., all the principal minors of Q are positive. In this case, LCP has a unique solution for each vector q [5]. LCP is equivalent to the so-called Finite-Dimensional Variational Inequality Problem, which consists of finding a vector $\overline{x} \in \mathbb{R}^n_+$ such that

$$(q + Q\overline{x})^T (x - \overline{x}) \geq 0, \quad \forall x \in \mathbb{R}^n_+.$$

[*]This work was partially supported by the Austrian Science Fund (FWF) Project J3793-N95. Research of Joaquim Judice is in the scope of R&D Unit 50008, financed by the applicable financial framework (FCT/MEC through national funds and when applicable co-funded by FEDER PT2020 partnership agreement). Finally partial financial support from Marie-Curie ITN Project MINO, MC-ITN 316647, is gratefully acknowledged.

[†]Laboratory for Information & Decision Systems, MIT, USA, philipp.hungerlaender@aau.at

[‡]Instituto de Telecomunicações, Universidade de Coimbra, Portugal, joaquim.judice@co.it.pt

[§]Department of Mathematics, Alpen-Adria Universität Klagenfurt, Austria, franz.rendl@aau.at

If Q is symmetric the above LCP is equivalent to the strictly convex quadratic programming problem with simple bounds:

$$\begin{aligned} &\min_x \; J(x), \\ &\text{subject to:} \quad x \geq 0,^1 \end{aligned} \tag{2}$$

where in this case Q is a positive-definite $n \times n$ matrix and $J(x) = \frac{1}{2}x^\top Qx + q^\top x$.

LCPs appear as a basic building block of many applications in bimatrix games, market and traffic equilibrium mdels, frictional contact problems, elastoplastic structural analysis, obstacle problems, pricing options and portfolio models among others. We recommend the books [5, 10, 31] and the article [33] for a detailed presentation of the most relevant applications of LCP. In some of these LCPs Q is a nonsymmetric P-matrix.

From a computational perspective, there are several well established techniques for solving an LCP with a nonsymmetric P-matrix. Lemke's method [26] and the so-called principal pivoting algorithms [4, 6, 7, 14, 19, 21, 29, 35, 36] are considered to be the most relevant direct methods for LCP. Interior-Point [18, 22, 23] and semi-smooth [12, 27] algorithms are the most efficient and robust iterative algorithms. An LCP with a nonsymmetric P-matrix can also be solved efficiently by an active-set algorithm or an iterative gradient method for computing the unique stationary point of the following quadratic program:

$$\begin{aligned} &\min_x \; L(x) \\ &\text{subject to:} \quad Qx \geq -q, \\ &\qquad\qquad\qquad x \geq 0, \end{aligned} \tag{3}$$

with $L(x) = \frac{1}{2}x^\top (Q + Q^\top)x + q^\top x$. We suggest the books [5, 10, 31] for good descriptions of all these techniques for the solution of an LCP with a nonsymmetric P-matrix.

In this paper we propose a feasible active set method for an LCP with a nonsymmetric P-matrix. It is inspired by the semi-smooth Newton method (SN-method for short) from Bergounioux et al. [1, 2]. Hungerländer and Rendl [16] propose a recursive variant of the SN-method which is globally convergent for problem (2). Here we adapt this method for LCP (1) and show its finite convergence whenever Q is a P-matrix. Furthermore, an extension of the algorithm is discussed for the solution of the so-called Bound Linear Complementarity Problem (BLCP) [5, 10] with a P-matrix. The extended algorithm also possesses finite convergence to the unique solution of BLCP.

Our new approach has several preferable features, like simplicity (no tuning parameters), finding the exact numerical solution, insensitivity with respect to initialisation and fast restarts. Moreover strict complementary is not required to be satisfied. Computational experience indicates that our approach outperforms other globally convergent approaches for solving an LCP with a symmetric or nonsymmetric P-matrix.

The paper is organized as follows. At the end of this section we summarize notation used throughout the paper. In Section 2 we recall the relevant details of the approach by Hungerländer and Rendl [16] that we denote from now on as recursive SN-method. In Section 3 we suggest an alternative merit function for the recursive SN-method that allows us to prove finite termination of our new method for any LCP with a P-matrix. We extend this convergence result to a Bound Linear Complementarity Problem (BLCP) with a P-matrix in Section 4. In Section 5 we show the efficiency of our new method on a variety of test problems and some conclusions are presented in the last section.

Notation: The following notation will be used throughout. For a subset $\mathcal{A} \subseteq \mathcal{N} = \{1, \dots, n\}$ and $x \in \mathbb{R}^n$ we write $x_\mathcal{A}$ for the components of x indexed by $\mathcal{A}$, i.e. $x_\mathcal{A} = (x_i)_{i \in \mathcal{A}}$. The complement of $\mathcal{A}$ with respect to $\mathcal{N}$ is denoted by $\overline{\mathcal{A}}$. If Q is a matrix and $\mathcal{A}, \mathcal{B} \subseteq \mathcal{N}$, then $Q_{\mathcal{A},\mathcal{B}}$ is the submatrix of Q, given by $Q_{\mathcal{A},\mathcal{B}} = (q_{ij})_{i \in \mathcal{A}, j \in \mathcal{B}}$. We write $Q_\mathcal{A}$ for $Q_{\mathcal{A},\mathcal{A}}$.

[1]The bound constraint $x \geq 0$ is equivalent to the also commonly used constraint $z \leq b$ via the transformation $z = b - x$.

2 The Recursive SN-Method [16]

The starting point for the present paper is the recursive SN-method for (2) introduced in [16]. In this section we briefly describe this method and its convergent properties. In Subsection 2.3 we review further variants of the SN-method and finally in Subsection 2.4 we showcase on small numerical examples that these SN-methods may fail for solving an LCP with a nonsymmetric P-matrix.

2.1 Basic Notation and Description

Let us start with some useful notation. It is well known that x together with a vector $\alpha \in \mathbb{R}^n$ of Lagrange multipliers for the simple bound constraints is the unique global minimizer of (2) if and only if (x, α) satisfies the KKT system (LCP)

$$Qx + q - \alpha = 0, \tag{4a}$$
$$x_i \alpha_i = 0, \; i \in \mathcal{N}, \tag{4b}$$
$$x \geq 0, \tag{4c}$$
$$\alpha \geq 0. \tag{4d}$$

The crucial step in solving (2) is to identify those inequalities which are active, i.e. the set $\mathcal{A} \subseteq \mathcal{N}$, where the solution to (2) satisfies $x_{\mathcal{A}} = 0$.

Given $\mathcal{A}$ and the corresponding inactive set $\mathcal{I} = \mathcal{N} \setminus \mathcal{A}$, in each iteration the $2n$ equations (4a) and (4b) are solved under the condition $x_{\mathcal{A}} = 0, \; \alpha_{\mathcal{I}} = 0$. To simplify notation we introduce for a given $\mathcal{A}$ the following set KKT($\mathcal{A}$) of equations:

$$\text{KKT}(\mathcal{A}) \qquad Qx + q - \alpha = 0, \quad x_{\mathcal{A}} = 0, \quad \alpha_{\mathcal{I}} = 0.$$

The solution of KKT($\mathcal{A}$) is given by

$$x_{\mathcal{A}} = 0, \quad Q_{\mathcal{I}} x_{\mathcal{I}} = -q_{\mathcal{I}}, \tag{5a}$$
$$\alpha_{\mathcal{I}} = 0, \quad \alpha_{\mathcal{A}} = q_{\mathcal{A}} + Q_{\mathcal{A},\mathcal{I}} x_{\mathcal{I}}. \tag{5b}$$

We also write $[x, \alpha] = \text{KKT}(\mathcal{A})$ to indicate that x and α satisfy KKT($\mathcal{A}$). In some cases we also write $x = \text{KKT}(\mathcal{A})$ to emphasize that we only use x and therefore we do not need the backsolve to get α. When we only carry out the backsolve to get α, we write $\alpha = \text{KKT}(\mathcal{A})$, and it is assumed that the corresponding x is available.

Let us call $\mathcal{A} \subseteq \mathcal{N}$ **a primal feasible set**, if the solution x to KKT($\mathcal{A}$) is primal feasible, i.e., $x \geq 0$. The set $\mathcal{A}$ is called **optimal** if $[x, \alpha] = KKT(\mathcal{A})$ satisfies $x \geq 0, \; \alpha \geq 0$.

Now for a given primal feasible set $\mathcal{A}$ the recursive SN-method determines a new primal feasible set $\mathcal{B}$ with

$$J(y) < J(x), \quad x = \text{KKT}(\mathcal{A}), \quad y = \text{KKT}(\mathcal{B}). \tag{6}$$

The objective function $J(x)$ is strictly decreasing during the iteration, hence no set will be visited twice, and finite termination is ensured. It thus serves as a merit function.

Now let us describe the recursive SN-method in more detail. Suppose that $\mathcal{A} \subseteq \mathcal{N}$ is a primal feasible set with $[x, \alpha]$ =KKT($\mathcal{A}$). Then we start a new iteration with the set

$$\mathcal{B}_s = \{i \in \mathcal{A} : \alpha_i \geq 0\}. \tag{7}$$

The set $\mathcal{B}_s$ does not need to be primal feasible, because $[y, \beta] = KKT(\mathcal{B}_s)$ may have $i \in \overline{\mathcal{B}}_s$ such that $y_i < 0$. To turn it into a primal feasible set $\mathcal{B}$, we carry out the following iterations.

```
$\mathcal{B} \leftarrow \mathcal{B}_s$
while $\mathcal{B}$ not primal feasible                                   (8)
    $x = \text{KKT}(\mathcal{B}), \; \mathcal{B} \leftarrow \mathcal{B} \cup \{i \in \overline{\mathcal{B}} : x_i \leq 0\}$
```

This iterative scheme will clearly terminate because $\mathcal{B}$ is augmented by adding only elements of $\overline{\mathcal{B}}_s$. The extension of $\mathcal{B}_s$ to a primal feasible set $\mathcal{B}$ is done by adding either elements from $\mathcal{A} \setminus \mathcal{B}_s$, which are contained in $\mathcal{B}_1$, or from $\mathcal{I}$, which are contained in $\mathcal{B}_2$. We define $\mathcal{J}_1 = \mathcal{A} \setminus (\mathcal{B}_s \cup \mathcal{B}_1)$ and $\mathcal{J}_2 = \mathcal{I} \setminus \mathcal{B}_2$. We depict the partition of the active and inactive set for two consecutive primal feasible active sets in Table 1.

$\mathcal{A}$			$\mathcal{I}$	
$\mathcal{B}_s$	$\mathcal{J}_s$			
$\mathcal{B}_s$	$\mathcal{B}_1$	$\mathcal{J}_1$	$\mathcal{J}_2$	$\mathcal{B}_2$

Table 1: The change from the primal feasible set $\mathcal{A}$ to the primal feasible set $\mathcal{B} = \mathcal{B}_s \cup \mathcal{B}_1 \cup \mathcal{B}_2$.

2.2 Convergence Results

First we recall the basic theoretical properties for two consecutive primal feasible points generated by the simple procedure in (8).

Lemma 1 ([16]) *Let two consecutive primal feasible active sets $\mathcal{A}$ and $\mathcal{B}$ with $[x, \alpha]$ =KKT($\mathcal{A}$), $[y, \beta]$ =KKT($\mathcal{B}$) generated by* (8) *be given. Then we have:*

a) ***Change of the active sets:*** *$\mathcal{B}_s \subseteq \mathcal{A}$ and $\mathcal{B}_s = \mathcal{A}$ if and only if $\mathcal{A}$ is optimal. If $\mathcal{A}$ is not optimal, then $\mathcal{A} \neq \emptyset$, $\mathcal{A} \neq \mathcal{B}$ and $x - y \neq 0$.*

b) ***Change of the merit function:*** *Let x, y, α, β and $\mathcal{B}_2$ be as above. Then*

$$J(y) - J(x) = -\frac{1}{2}(x-y)^\top Q(x-y) - \sum_{i \in \mathcal{B}_2} (x-y)_i \beta_i.$$

Moreover, $(x - y)_{\mathcal{B}_2} > 0$.

Lemma 1b) indicates that we have no means to avoid $J(y) \geq J(x)$ using the described setup. To avoid cycling, we therefore need to take additional measures in case $J(y) \geq J(x)$. In particular we exploit the property outlined in the following lemma.

Lemma 2 (Decrease of the merit function [16]) *Let $\mathcal{A}$ be a primal feasible index set with nonoptimal solution $[x, \alpha]$ =KKT($\mathcal{A}$) and index partition as in Table 1 and assume $|\mathcal{A}| > 1$. Then there exists $\mathcal{A}_0 \neq \emptyset$, $\mathcal{A}_0 \subseteq \mathcal{A}$ and $(\mathcal{B}_1 \cup \mathcal{J}_1) \setminus \mathcal{A}_0 \neq \emptyset$. Let $\mathcal{B}_0$ be the optimal active set for the subproblem with $x_{\mathcal{A}_0} = 0$, $x_{\overline{\mathcal{A}_0}} \geq 0$ and consider $\mathcal{B} = \mathcal{A}_0 \cup \mathcal{B}_0$. Then $J(y) < J(x)$ holds for $[y, \beta]$ =KKT($\mathcal{B}$).*

Remark 1 *There exist many ways to choose $\mathcal{A}_0$. In our implementation we use the following rule to determine $\mathcal{A}_0$:*

$$\mathcal{A}_0 = \begin{cases} \mathcal{B}_s, & \text{if } \mathcal{B}_s \neq \emptyset, \\ \{j\}, \text{ for some } j \in \mathcal{B}_1, & \text{if } \mathcal{B}_s = \emptyset, \ \mathcal{B}_1 \neq \emptyset, \\ \{j\}, \text{ for some } j \in \mathcal{J}_1, & \text{if } \mathcal{B}_s = \emptyset, \ \mathcal{B}_1 = \emptyset. \end{cases} \tag{9}$$

Next let us give an algorithmic outline of the recursive SN-method. Therefore we assume $\mathcal{A}$ is not optimal for the following case distinction. We first recall that in this case $|\mathcal{A}| \geq 1$ due to Lemma 1a). We consider the following cases separately. Let x, y, α, β and $\mathcal{A}$ and $\mathcal{B}$ be as above.

Case 1: $J(y) < J(x)$.

In this case we can set $\mathcal{A} \leftarrow \mathcal{B}$, and continue with the next iteration.

Case 2: $J(y) \geq J(x)$ and $|\mathcal{A}| = 1$.

The set $\mathcal{A} = \{j\}$ is primal feasible. Since x is primal feasible, but not optimal, we must have $\alpha_j < 0$. Thus the merit function would improve by allowing x_j to move away from the boundary. Hence in an optimal solution we have $x_j > 0$ and thus a problem with only $n-1$ constraints which we solve by induction on the number of constraints.

Case 3: $J(y) \geq J(x)$ and $|\mathcal{A}| > 1$.

In this case we solve the subproblem described in Lemma 2 and consequently obtain a new primal feasible active set $\mathcal{B}$ with $J(y) < J(x)$, $[y, \beta] =$KKT($\mathcal{B}$).

We summarize in Table 2 the recursive SN-method, which produces primal feasible iterates in each iteration. For implementation details and their motivation we refer to [16]. The corresponding Matlab code for generating all instances and solving them with the various methods discussed in [16] and also in this paper is available from `http://philipphungerlaender.jimdo.com/qp-code/`.

Recursive SN-method

Input: A symmetric P (positive definite) matrix Q, $q \in I\!R^n$ and a starting active set $\mathcal{A} \subseteq \mathcal{N}$.
Output: The primal optimal solution x of (2), the corresponding dual multiplier α and optimal active set $\mathcal{A}$.

```
$x = $ KKT($\mathcal{A}$).
while $\mathcal{A}$ not primal feasible: $\mathcal{A} \leftarrow \mathcal{A} \cup \{i \in \overline{\mathcal{A}} : x_i \leq 0\}$; $x = $ KKT($\mathcal{A}$); endwhile
$\alpha = $ KKT($\mathcal{A}$).
while $\mathcal{A}$ not optimal
    $\mathcal{B} \leftarrow \{i \in \mathcal{A} : \alpha_i \geq 0\}$, $y = $ KKT($\mathcal{B}$).
    while $\mathcal{B}$ not primal feasible: $\mathcal{B} \leftarrow \mathcal{B} \cup \{i \in \overline{\mathcal{B}} : y_i \leq 0\}$; $y = $ KKT($\mathcal{B}$); endwhile
    Case 1: $J(y) < J(x)$
        $\mathcal{A} \leftarrow \mathcal{B}$.
    Case 2: $J(y) \geq J(x)$ and $|\mathcal{A}| = 1$
        Solve recursively problem (2) with the bound on $\mathcal{A} = \{j\}$ removed.
    Case 3: $J(y) \geq J(x)$ and $|\mathcal{A}| > 1$
        Select $A_0$ according to (9).
        Let $\mathcal{B}_0$ be the optimal active set for problem (2) with $x_{\mathcal{A}_0} = 0$, $x_{\overline{\mathcal{A}}_0} \geq 0$.
        $\mathcal{A} \leftarrow \mathcal{A}_0 \cup \mathcal{B}_0$.
    $\alpha = $ KKT($\mathcal{A}$).
endwhile
```

Table 2: Description of the recursive SN-method.

Since either the objective value or the problem dimension (i.e. number of constraints) strictly decreases in each iteration, the following result holds.

Theorem 3 *[16] For strictly convex quadratic programming problems with simple bounds the recursive SN-method as given in Table 2 terminates in a finite number of iterations with the optimal solution.*

Hence the recursive SN-method is able to solve an LCP with a symmetric P-matrix. Furthermore the unique solution of LCP is given by the optimal primal solution of the quadratic program (2) and the corresponding vector of dual multipliers.

2.3 Review of Further Variants of the SN-Method

The standard active set approach is described in detail in Nocedal and Wright [32, Section 16.5]. It is further investigated and extended by Moré and Toraldo [28] and Dostál and Schöberl [9], who combine it with projected gradients.

A special type of active-set method was introduced for solving an LCP with a symmetric P-matrix (positive definite) by Bergounioux et al. [1, 2] in connection with constrained optimal control problems and tailored to deal with discretisations of specially structured elliptic partial differential equations. Hintermüller et al. [15] show that this primal-dual active set approach can in fact be rephrased as a semi-smooth Newton method (SN-method) that computes the unique optimal solution of the strictly convex quadratic program (2) equivalent to LCP. Practical computational evidence shows that if this SN-method converges at all, it typically takes very few iterations to reach an optimal solution, see for instance Kunisch and Rendl [25] and Júdice and Pires [19].

The SN-method does not need a merit function to operate and therefore can also be used for solving an LCP with a nonsymmetric P-matrix. The algorithm can be viewed as an infeasible active set method, generating a sequence of active sets $\mathcal{A} \subseteq \mathcal{N}$, until an optimal active set is reached. A new iteration is started with the following active set:

$$\mathcal{B} = \{i \in \mathcal{A} : \alpha_i \geq 0\} \cup \{i \in \mathcal{I} : x_i \leq 0\}.$$

Note that the slightly different active set update rule $\mathcal{B} = \{i \in \mathcal{A} : \alpha_i > 0\} \cup \{i \in \mathcal{I} : x_i < 0\}$ has also been used in the original SN-method [25]. Furthermore the SN-method using the update rule $\mathcal{B} = \{i \in \mathcal{A} : \alpha_i \geq 0\} \cup \{i \in \mathcal{I} : x_i < 0\}$ is exactly the simple form of the block principal pivoting algorithm (also called Kostreva's algorithm [24] in [11]) discussed in [11, 19]. All these three versions of the original SN-method have the same convergence properties.

Hence the original SN-method can be applied for the solution of an LCP with a symmetric or nonsymmetric P-matrix and differs from the recursive SN-method with respect to the update rule for the active set and does not use any acceptance criterion for the new active sets, like decrease of a merit function. We summarize the steps of the SN-method in Table 3.

```
SN-method
Input: A $P$-matrix $Q$, $q \in I\!R^n$ and a starting active set $\mathcal{A} \subseteq \mathcal{N}$.
Output: The unique solution $(x, \alpha)$ of LCP and corresponding active-set $\mathcal{A}$.
repeat
    $[x, \alpha] = \mathrm{KKT}(\mathcal{A})$.
    $A \leftarrow \{i \in \mathcal{A} : \alpha_i \geq 0\} \cup \{i \in \mathcal{I} : x_i \leq 0\}$.
until $\mathcal{A}$ is optimal
```

Table 3: Description of the SN-method.

Global convergence of the SN-method can only be guaranteed for special classes of P-matrices. The method possesses finite convergence and is even a polynomial-time algorithm when Q is a symmetric or nonsymmetric M-matrix [4, 5, 20, 31]. In [25] global convergence of the SN-method is shown for a symmetric P-matrix with a strong form of diagonal dominance. However, the SN-method can cycle for a general symmetric or nonsymmetric P-matrix of order $n \geq 3$ [13]. In particular Gharbia and Gilbert [13] show that a cycle made of k nodes is possible, for an arbitrary $k \in \{3, \ldots, n\}$. The SN-method may even cycle for symmetric P-matrix (positive definite) of order $n \geq 3$ [8], see Example 1 below.

In [15] it is shown that the method converges superlinearly if the starting point is sufficiently close to the optimum. However, the radius of the ball where such a convergence rate is guaranteed may be arbitrary small [13]. Hintermüller et al. [15] also show that the SN-method is monotonically and globally convergent if Q is in the set $M_\varepsilon := \{Q : \ Q \text{ is a matrix near an } M\text{-matrix of the same order}\}$.

Júdice and Pires [11, 19] have combined a version of the original SN-method with the so-called Murty's algorithm [29] to guarantee finite convergence for the resulting algorithm to compute the unique solution of an LCP with a symmetric or nonsymmetric P-matrix. Despite not being called for most instances [11, 19], Murty´s algorithm is quite slow and this has recently implied substantial scientific activity to figure out other safeguards for the SN-method to ensure global convergence. Byrd et al. [3] ensure global convergence by maintaining descent directions in combination with the safeguarding procedure proposed by Judice and Pires [19]. Curtis et al. [8] consider a more general setting, which also includes linear equations in the constraints. A special treatment is given to variables which change index sets very often, thereby forcing global convergence. In a recent paper [17] we investigate yet another globalization strategy for the SN-method, which avoids recursion and which aims at maintaining the combinatorial flavour of the original SN-method. But to the best of our knowledge none of these methods is globally convergent for finding the unique solution of an LCP with a nonsymmetric P-matrix.

2.4 Numerical Examples

We close this section with the following two small numerical examples to further clarify the workings of both the SN-method and the recursive SN-method and to showcase instances on which they fail.

Example 1 *First let us state the example from Curtis et al. [8] that causes the SN-method to cycle for 6 out of the 8 possible starting active sets. We are given the following problem data:*

$$Q = \begin{pmatrix} 4 & 5 & -5 \\ 5 & 9 & -5 \\ -5 & -5 & 7 \end{pmatrix}, \qquad q = \begin{pmatrix} -2 \\ -1 \\ 3 \end{pmatrix}.$$

Q is a symmetric P-matrix (i.e. symmetric positive definite) and the the active set associated to the unique solution of LCP is $\mathcal{A} = \{2,3\}$. The corresponding active-set-transition-graphs of the SN-method for the 8 starting active sets mentioned in [8] is depicted in Figure 1 and confirms the existence of a cycle for 6 of these sets.

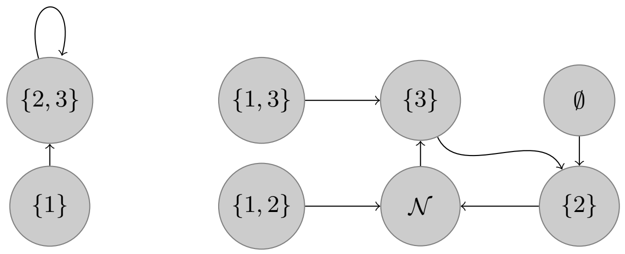


Figure 1: Active-set-transition-graph of the SN-method for the instance from Example 1.

On the other hand the recursive SN-method is able to compute the optimal active set for all these 8 starting points, as shown in Figure 2.

Example 2 *For an LCP with a nonsymmetric P-matrix both the SN-method and the recursive SN-method may fail even for a very small instance. Consider the LCP with*

$$Q = \begin{pmatrix} 1 & -10 & 10 \\ 10 & 1 & 10 \\ -10 & -10 & 1 \end{pmatrix}, \qquad q = \begin{pmatrix} 1 \\ -3 \\ 5 \end{pmatrix}.$$

$$\begin{tikzcd}
\{3\} \arrow[r] & \{2,3\} \arrow[loop above] & \{1,3\} \arrow[l] \\
\emptyset \arrow[r] & \mathcal{N} \arrow[u] & \{1,2\} \arrow[l] \\
\{1\} \arrow[ur] & & \{2\} \arrow[ul]
\end{tikzcd}$$

Figure 2: Active-set-transition-graph of the recursive SN-method for the instance from Example 1.

Then Q is a nonsymmetric P-matrix and the active set corresponding to the unique solution of LCP is $\mathcal{A} = \{3\}$. There exist two primal feasible points $x_1 = KKT(\{1\})$ and $x_2 = KKT(\{1,3\})$ with a smaller value of the merit function than the solution $x^* = KKT(\{3\})$ of the given LCP: $J(x_2) < J(x_1) < J(x^*)$. The recursive SN-method, when started either from $\mathcal{A} = \{1\}$ or $\mathcal{A} = \{1,3\}$, will not reach the set $\mathcal{A} = \{3\}$, as it generates only active sets if $J(x)$ is strictly decreasing. Also the SN-method fails on this instance as can be seen from the active-set-transition-graph depicted in Figure 3.

$$\begin{tikzcd}
\{3\} \arrow[loop above] & \mathcal{N} \arrow[r] & \{1,3\} \arrow[dr, bend left] & \{1\} \arrow[d] \\
\{1,2\} \arrow[u] & \{2\} \arrow[r] & \{2,3\} \arrow[u] & \emptyset \arrow[l]
\end{tikzcd}$$

Figure 3: Active-set-transition-graph of the SN-method for the instance from Example 2.

From the above example we can conclude that both the SN- and the recursive SN-method may easily fail for LCPs with a nonsymmetric P-matrix Q. While the SN-method may run into a cycle like in the symmetric case, the recursive SN-method fails whenever it reaches a primal point x for which the objective value on the symmetric part of Q is smaller than the objective value of the solution x^* of the LCP, i.e., $J(x) < J(x^*)$. Note that for a nonsymmetric P-matrix, LCP and the quadratic program (2) are no longer equivalent. Instead, LCP is equivalent to the quadratic program (3) that contains some further linear constraints $Qx \geq -q$. This explains why the recursive SN-method can find a primal feasible solution of (2) with a smaller value than $J(x^*)$. It is important to add that designing an SN-method for finding an optimal solution of (3) is not an easy task due to the existence of the linear constraints $Qx \geq -q$.

Motivated by the example above, we propose in the next section an alternative version of the recursive SN-method that can be proven to be globally convergent for any LCP with a nonsymmetric P-matrix. In Section 5.1 we compare the SN-method and our two versions of the recursive SN-method for both symmetric and nonsymmetric LCPs with P-matrices on a variety of benchmark instances.

3 Globalization of the Recursive SN-Method for LCPs with P-Matrices

As shown in the previous section, the recursive SN-method is not guaranteed to converge for an LCP with a nonsymmetric P-matrix because the minimizer of the merit function $\frac{1}{2}x^\top Qx + q^\top x$ on $\mathbb{R}^n_+$ is not necessarily the unique solution of (1). In other words the recursive SN-method converges to the minimizer of the symmetric part of the P-matrix Q on the nonnegative orthant, which is not necessarily the unique solution of (1).

In this section we introduce an alternative merit function for the recursive SN-method. In Subsection 3.1 we prove that this version of the recursive SN-method is globally convergent for an LCP with P-matrix. In Subsection 3.2 we suggest an algorithmic enhancement to deal with the recursive call of our method more efficiently.

3.1 Convergence Argument for an Alternative Merit Function

Instead of using $J(x)$ as merit function, we now suggest the cardinal of the set of dual infeasible multipliers as our new merit function

$$m(\alpha) = |\{i \in \mathcal{N} \ : \ \alpha_i < 0\}|, \tag{10}$$

which we call the Cardinal Dual Infeasibility (CDI) merit function. Note that a Cardinal (Primal and Dual) Infeasibiliy function has been used for the implementation of the safeguard of the block principal pivoting algorithm discussed in [11, 19]. For a given primal feasible active set $\mathcal{A}$ (i.e., $x \geq 0$), $[x, \alpha] = \mathrm{KKT}(\mathcal{A})$ is a solution of LCP if $m(\alpha) = 0$. The merit function $m(\alpha)$ is used to decide how the update step for the active set is carried out. We follow the general structure of the recursive SN-method from Table 2. Hence, after having determined the primal feasible set $\mathcal{B}$ from the set B_s defined in (7), we compute $[y, \beta]$ =KKT($\mathcal{B}$) and enter the following case distinction.

Case 1: $m(\beta) < m(\alpha)$.

In this case we can set $\mathcal{A} \leftarrow \mathcal{B}$, and continue with the next iteration.

Case 2: $m(\beta) \geq m(\alpha)$ and $\mathbf{m}(\alpha) = \mathbf{1}$.

In this case there is exactly one dual infeasible multiplier $\alpha_i < 0$. Since a principal submatrix of Q also belongs to the class of P-matrices, the LCP cannot have a solution with $x_i = 0$. Hence, $x_i > 0$ in the unique solution of the LCP and the LCP of order $(n-1)$ with α_i fixed to 0 is considered from now on.

Case 3: $m(\beta) \geq m(\alpha)$ and $\mathbf{m}(\alpha) > \mathbf{1}$.

In this case we can (recursively) solve an LCP of order $(n - |\mathcal{A}_0|)$ with $x_{\mathcal{A}_0} = 0,\ x_{\overline{\mathcal{A}}_0} \geq 0$, where the set $\mathcal{A}_0$ contains at least one and less than $m(\alpha)$ elements and is constructed as follows:

(a) If $0 < |\mathcal{B}_s| < m(\alpha)$, we set $\mathcal{A}_0 = \mathcal{B}_s$.

(b) If $|\mathcal{B}_s| \geq m(\alpha)$, then we choose $\mathcal{A}_0$ to be the set containing the $m(\alpha) - 1$ indices from $\mathcal{B}_s$ with the largest corresponding dual multipliers:

$$\mathcal{A}_0 = \{i \in \mathcal{B}_s : \alpha_i \text{ is one of the } (m(\alpha) - 1) \text{ largest dual multipliers}\}.$$

(c) Finally if $\mathcal{B}_s = \emptyset$, then $|\mathcal{B}_1 \cup \mathcal{J}_1| > 1$, because $|\mathcal{A}| > 1$. In this case we set $A_0 = \{j\}$ for some $j \in \mathcal{B}_1 \cup \mathcal{J}_1$.

In summary we select $\mathcal{A}_0$ according to the following rule:

$$\mathcal{A}_0 = \begin{cases} \mathcal{B}_s, & \text{if } 0 < |\mathcal{B}_s| < m(\alpha), \\ \{i \in \mathcal{B}_s : \alpha_i \text{ is one of the } (m(\alpha) - 1) \text{ largest dual multipliers}\}, & \text{if } |\mathcal{B}_s| \geq m(\alpha), \\ \{j\} \text{ for some } j \in \mathcal{B}_1 \cup \mathcal{J}_1, & \text{if } \mathcal{B}_s = \emptyset. \end{cases} \tag{11}$$

The active set of the solution of the LCP with $x_{\mathcal{A}_0} = 0$, $x_{\overline{\mathcal{A}}_0} \geq 0$ is denoted by $\mathcal{B}_0$. As $\mathcal{B}_0$ contains no dual infeasible multipliers β_i, only the dual multipliers β_i, $i \in \mathcal{A}_0$ could be infeasible. If $\mathcal{B}_s \neq \emptyset$ all infeasible dual multipliers from α are contained in the subproblem and $|A_0| < m(\alpha)$ holds. If $\mathcal{B}_s = \emptyset$ we have $A_0 = \{j\}$, $\alpha_j < 0$ and thus $|\mathcal{A}_0| < 2 \leq m(\alpha)$ also holds. Hence in both cases the set $\mathcal{B} = \mathcal{A}_0 \cup \mathcal{B}_0$ gives a primal-dual pair $[y, \beta]$ =KKT($\mathcal{B}$) with y primal feasible and $m(\beta) < m(\alpha)$. We summarize the discussion in the following lemma.

Lemma 4 (Decrease of the CDI merit function) *Let $\mathcal{A}$ be a primal feasible index set with $[x, \alpha]$ =KKT($\mathcal{A}$) and index partition as in Table 1 and assume $m(\alpha) > 1$. Then there exists $\mathcal{A}_0 \neq \emptyset$, $\mathcal{A}_0 \subseteq \mathcal{A}$ and $(\mathcal{B}_1 \cup \mathcal{J}_1) \setminus \mathcal{A}_0 \neq \emptyset$. Let $\mathcal{B}_0$ be the solution of the LCP with $x_{\mathcal{A}_0} = 0$, $x_{\overline{\mathcal{A}}_0} \geq 0$ and consider $\mathcal{B} = \mathcal{A}_0 \cup \mathcal{B}_0$. Then $m(\beta) < m(\alpha)$ holds for $[y, \beta]$ =KKT($\mathcal{B}$).*

In summary either $m(\alpha)$ or the problem dimension (i.e. the number of complementary pairs (x_i, α_i)) strictly decreases in each iteration of the algorithm and the following result holds:

Theorem 5 *For an LCP with a P-matrix the recursive SN-method with the CDI merit function $m(\alpha)$ terminates in a finite number of iterations with the unique solution of LCP.*

3.2 Dealing with the Recursive Problem Efficiently

Finally we propose an algorithmic enhancement to avoid making slow progress if $m(\alpha)$ is small but > 1. In this case we can only fix very few, i.e., $m(\alpha) - 1$, variables to the bound when making the recursive call. Then it may happen that the subproblem is very similar to the original problem (because we fixed only very few variables) and hence also needs a recursive call. In this case the algorithm may need a large number of recursion levels, which slows down the method. This behaviour can be observed occasionally in the experiments in Subsection 5.1.

The following algorithmic enhancement succeeds in avoiding the behaviour described above. The most natural choice for the set $\mathcal{A}_0$ of variables to be fixed for the subproblem is the set $\mathcal{B}_0$ of variables that are currently at the bound and are dual feasible. But the choice $\mathcal{A}_0 = \mathcal{B}_0$ does not guarantee a decrease of the merit function $m(\alpha)$, if $|\mathcal{A}_0| \geq m(\alpha)$. Nonetheless our experiments indicate that this choice usually works very well in practice. Hence we suggest to first try the natural choice $\mathcal{A}_0 = \mathcal{B}_0$ and only if the CDI merit function does not decrease, we discard $\mathcal{A}_0 = \mathcal{B}_0$ and the associated primal-dual pair and instead choose $\mathcal{A}_0$ safely according to (11).

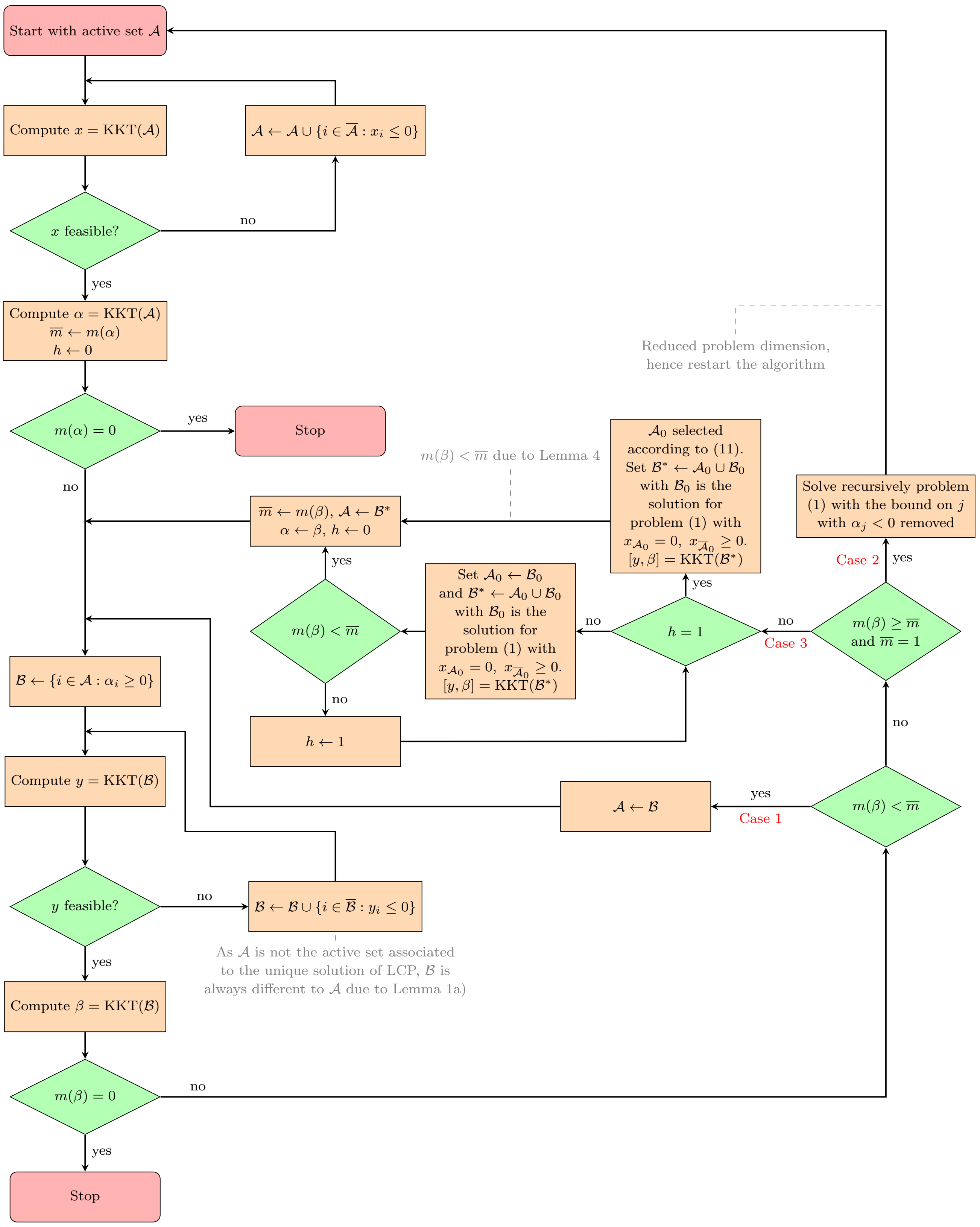
We summarize the recursive SN-method with CDI merit function $m(\alpha)$ and algorithmic enhancement in Table 4 and alternatively in the flow chart in Figure 4. Note that the only difference between this new and the previous version of the recursive SN-method is the possible repetition of Case 3. Since the CDI merit function strictly decreases with or without repetition, the following result holds.

Theorem 6 *For an LCP with a P-matrix the recursive SN-method with the CDI merit function $m(\alpha)$ and algorithmic enhancement as given in Table 4 terminates in a finite number of iterations with the unique solution of LCP.*

4 Extension to BLCPs with P-Matrices

In this section we show that the convergence argument for the recursive SN-method with CDI merit function $m(\alpha)$ presented in the previous section can be extended to a Bound Linear Complementarity Problem (BLCP) [5, 10] with a P-matrix.[2]

[2]Note that the convergence argument of the original recursive SN-method with merit function $J(x)$ can also be generalized to a BLCP with a symmetric P-matrix, as this problem is equivalent to a strictly convex programming problem with box constraints, for details see [16, Appendix].



Start with active set $\mathcal{A}$
Compute $x = \mathrm{KKT}(\mathcal{A})$
$\mathcal{A} \leftarrow \mathcal{A} \cup \{i \in \overline{\mathcal{A}} : x_i \leq 0\}$
x feasible?
no
yes
Compute $\alpha = \mathrm{KKT}(\mathcal{A})$
$\overline{m} \leftarrow m(\alpha)$
$h \leftarrow 0$
Reduced problem dimension, hence restart the algorithm
$m(\alpha) = 0$
yes
Stop
no
$m(\beta) < \overline{m}$ due to Lemma 4
$\mathcal{A}_0$ selected according to (11). Set $\mathcal{B}^* \leftarrow \mathcal{A}_0 \cup \mathcal{B}_0$ with $\mathcal{B}_0$ is the solution for problem (1) with $x_{\mathcal{A}_0} = 0,\ x_{\overline{\mathcal{A}}_0} \geq 0$. $[y, \beta] = \mathrm{KKT}(\mathcal{B}^*)$
Solve recursively problem (1) with the bound on j with $\alpha_j < 0$ removed
$\overline{m} \leftarrow m(\beta)$, $\mathcal{A} \leftarrow \mathcal{B}^*$
$\alpha \leftarrow \beta$, $h \leftarrow 0$
yes
Case 2
yes
yes
Set $\mathcal{A}_0 \leftarrow \mathcal{B}_0$ and $\mathcal{B}^* \leftarrow \mathcal{A}_0 \cup \mathcal{B}_0$ with $\mathcal{B}_0$ is the solution for problem (1) with $x_{\mathcal{A}_0} = 0,\ x_{\overline{\mathcal{A}}_0} \geq 0$. $[y, \beta] = \mathrm{KKT}(\mathcal{B}^*)$
$m(\beta) < \overline{m}$
no
$h = 1$
no
Case 3
$m(\beta) \geq \overline{m}$ and $\overline{m} = 1$
$\mathcal{B} \leftarrow \{i \in \mathcal{A} : \alpha_i \geq 0\}$
no
$h \leftarrow 1$
no
Compute $y = \mathrm{KKT}(\mathcal{B})$
$\mathcal{A} \leftarrow \mathcal{B}$
yes
Case 1
$m(\beta) < \overline{m}$
y feasible?
no
$\mathcal{B} \leftarrow \mathcal{B} \cup \{i \in \overline{\mathcal{B}} : y_i \leq 0\}$
As $\mathcal{A}$ is not the active set associated to the unique solution of LCP, $\mathcal{B}$ is always different to $\mathcal{A}$ due to Lemma 1a)
yes
Compute $\beta = \mathrm{KKT}(\mathcal{B})$
$m(\beta) = 0$
no
yes
Stop

Figure 4: Outline of the workings of the recursive SN-method with the CDI merit function $m(\alpha)$ and algorithmic enhancement.

Recursive SN-method with CDI merit function $m(\alpha)$ and algorithmic enhancement

Input: A P-matrix Q, $q \in I\!R^n$ and a starting active set $\mathcal{A} \subseteq \mathcal{N}$.
Output: Unique solution (x, α) of LCP and corresponding active set $\mathcal{A}$.

```
$x = \mathrm{KKT}(\mathcal{A})$, $h = 0$, $\overline{m} = n + 1$.
while $\mathcal{A}$ not primal feasible:  $\mathcal{A} \leftarrow \mathcal{A} \cup \{i \in \overline{\mathcal{A}} : x_i \leq 0\}$;  $x = \mathrm{KKT}(\mathcal{A})$; endwhile
$\alpha = \mathrm{KKT}(\mathcal{A})$.
Set $\overline{m} = m(\alpha)$.
while  $m(\alpha) > 0$
    $\mathcal{B}_s \leftarrow \{i \in \mathcal{A} : \alpha_i \geq 0\}$,  $\mathcal{B} \leftarrow \mathcal{B}_s$,    $y = \mathrm{KKT}(\mathcal{B})$.
    while $\mathcal{B}$ not primal feasible:  $\mathcal{B} \leftarrow \mathcal{B} \cup \{i \in \overline{\mathcal{B}} : y_i \leq 0\}$,  $y = \mathrm{KKT}(\mathcal{B})$; endwhile
    $\beta = \mathrm{KKT}(\mathcal{B})$.
    Case 1: $m(\beta) < \overline{m}$
        Set $\mathcal{A} \leftarrow \mathcal{B}$, $\alpha \leftarrow \beta$ and $h = 0$.
    Case 2: $m(\beta) \geq \overline{m}$ and $\overline{m} = 1$
        Solve recursively problem (1) with the bound on $j$ with $\alpha_j < 0$ removed.
    Case 3: $m(\beta) \geq \overline{m}$ and $\overline{m} > 1$
        if $h = 1$
            Select $\mathcal{A}_0$ according to (11).
        else
            Set $\mathcal{A}_0 \leftarrow \mathcal{B}_s$.
        endif
        Let $\mathcal{B}_0$ be the active set of the solution of problem (1) with $y_{\mathcal{A}_0} = 0$,  $y_{\overline{\mathcal{A}_0}} \geq 0$.
        $\mathcal{B}^* \leftarrow \mathcal{A}_0 \cup \mathcal{B}_0$.
        $[y, \beta] = \mathrm{KKT}(\mathcal{B}^*)$.
        if $m(\beta) < \overline{m}$
            Set $\overline{m} = m(\beta)$, $\mathcal{A} \leftarrow \mathcal{B}^*$, $\alpha \leftarrow \beta$ and $h = 0$.
        else
            Set $h = 1$ and repeat Case 3.
        endif
endwhile
```

Table 4: Description of the recursive SN-method with the CDI merit function $m(\alpha)$ and algorithmic enhancement.

A BCLP with a P-matrix has a unique solution [10] and can be written as follows:

$$\begin{aligned} Qx + q + \alpha - \gamma &= 0, \\ \alpha^\top (b - x) &= 0, \\ \gamma^\top (x - a) &= 0, \\ b - x &\geq 0, \\ x - a &\geq 0, \\ \alpha &\geq 0, \\ \gamma &\geq 0, \end{aligned} \tag{12}$$

where $a, b \in \mathbb{R}^n$ are given vectors of lower and upper bounds respectively such that $a_i < b_i$, $i \in \mathcal{N}$. As for the LCP, the variables x_i are called primal while the multipliers α_i and γ_i are the dual variables. The order (or dimension) of a BLCP is the number n of primal variables. As for LCP, let us denote $\mathrm{KKT}(\mathcal{A}, \mathcal{C})$

the following set of equations:

$$\mathrm{KKT}(\mathcal{A},\mathcal{C}) \qquad Qx + q + \alpha - \gamma = 0, \quad x_{\mathcal{A}} = b_{\mathcal{A}}, \quad x_{\mathcal{C}} = a_{\mathcal{C}}, \quad \alpha_{\mathcal{I}} = \gamma_{\mathcal{I}} = 0,$$

where $\mathcal{A}$ and $\mathcal{C}$ are the active sets corresponding to the upper and lower bounds respectively. Note that $A \cap C = \emptyset$, due to the definition of the bounds. So, if $\mathcal{I} = \mathcal{N} - (\mathcal{A} \cup \mathcal{C})$, then $\mathrm{KKT}(\mathcal{A},\mathcal{C})$ reduces to:

$$x_{\mathcal{A}} = b_{\mathcal{A}}, \quad x_{\mathcal{C}} = a_{\mathcal{C}}, \quad Q_{\mathcal{I},\mathcal{I}} x_{\mathcal{I}} = -\left(q_{\mathcal{I}} + Q_{\mathcal{I},\mathcal{C}} a_{\mathcal{C}} + Q_{\mathcal{I},\mathcal{A}} b_{\mathcal{A}}\right), \tag{13a}$$

$$\alpha_{\mathcal{I}} = 0, \qquad \alpha_{\mathcal{C}} = 0, \qquad \alpha_{\mathcal{A}} = q_{\mathcal{A}} + Q_{\mathcal{A},\mathcal{I}} x_{\mathcal{I}} + Q_{\mathcal{A},\mathcal{C}} a_{\mathcal{C}}, \tag{13b}$$

$$\gamma_{\mathcal{I}} = 0, \qquad \gamma_{\mathcal{A}} = 0, \qquad \gamma_{\mathcal{C}} = q_{\mathcal{C}} + Q_{\mathcal{C},\mathcal{I}} x_{\mathcal{I}} + Q_{\mathcal{C},\mathcal{A}} b_{\mathcal{A}}. \tag{13c}$$

Since Q is a P-matrix, $\mathrm{KKT}(\mathcal{A},\mathcal{C})$ has a unique solution for each pair of sets $(\mathcal{A},\mathcal{C})$. Furthermore, similarly to the LCP, the main computational effort for computing this solution relies on a linear system with the matrix $Q_{\mathcal{I},\mathcal{I}}$. As before, we use the notations $[x,\alpha,\gamma] = \mathrm{KKT}(\mathcal{A},\mathcal{C})$ for representing this unique solution and $[\alpha,\gamma] = \mathrm{KKT}(\mathcal{A},\mathcal{C})$ when only the dual variables are computed and x is available.

Let us call the pair $(\mathcal{A},\mathcal{C})$ a primal feasible pair, if the solution x to $\mathrm{KKT}(\mathcal{A},\mathcal{C})$ is primal feasible, i.e., $a \le x \le b$. Now suppose that $(\mathcal{A},\mathcal{C})$ is a primal feasible pair. Then we start a new iteration with the sets

$$\mathcal{B}_s = \{i \in \mathcal{A} : \alpha_i \ge 0\}, \qquad \mathcal{D}_s = \{i \in \mathcal{C} : \gamma_i \ge 0\}. \tag{14}$$

The pair $(\mathcal{B}_s,\mathcal{D}_s)$ does not need to be primal feasible, because $[x,\alpha,\gamma] = \mathrm{KKT}(\mathcal{B}_s,\mathcal{D}_s)$ may have $i \in \overline{\mathcal{B}}_s$ such that $x_i > b_i$ or $i \in \overline{\mathcal{D}}_s$ such that $x_i < a_i$. To turn it into a primal feasible pair $(\mathcal{B},\mathcal{D})$, we carry out the following iterations.

```
$\mathcal{B} \leftarrow \mathcal{B}_s,\ \mathcal{D} \leftarrow \mathcal{D}_s$
while $(\mathcal{B},\mathcal{D})$ not primal feasible
  $[x] = \mathrm{KKT}(\mathcal{B},\mathcal{D}),\ \mathcal{B} \leftarrow \mathcal{B} \cup \{i \in \overline{\mathcal{B}} : x_i \ge b_i\},\ \mathcal{D} \leftarrow \mathcal{D} \cup \{i \in \overline{\mathcal{D}} : x_i \le a_i\}.$
```

This iterative scheme will clearly terminate because $\mathcal{B}$ and $\mathcal{D}$ are augmented by adding only elements of $\mathcal{J}_s = \mathcal{N} \setminus \{\mathcal{B}_s \cup \mathcal{D}_s\}$. Our method starts with a primal feasible pair $(\mathcal{A},\mathcal{C})$ and generates a new primal feasible pair $(\mathcal{B},\mathcal{D})$ as described above. Then a new iteration is started with $(\mathcal{B},\mathcal{D})$ in place of $(\mathcal{A},\mathcal{C})$.

As for the LCP, we suggest to use the recursive SN-method with the Cardinal Dual Infeasibility (CDI) merit function:

$$m(\alpha,\gamma) = |\{i \in \mathcal{N} \ : \ \alpha_i < 0\}| + |\{i \in \mathcal{N} \ : \ \gamma_i < 0\}|. \tag{15}$$

Due to (13), $\alpha_i \gamma_i = 0,\ i \in \mathcal{N}$, and $0 \le m(\alpha,\gamma) \le n$. Now we take a closer look at how $(\mathcal{B}_s,\mathcal{D}_s)$ and $(\mathcal{B},\mathcal{D})$ change relative to $(\mathcal{A},\mathcal{C})$. Formally, the situation looks as indicated in Table 5.

$\mathcal{A}$				$\mathcal{C}$				$\mathcal{I}$		
$\mathcal{B}_s$	$\mathcal{J}_s$			$\mathcal{D}_s$	$\mathcal{J}_s$					
$\mathcal{B}_s$	$\mathcal{B}_1$	$\mathcal{D}_2$	$\mathcal{J}_1$	$\mathcal{D}_s$	$\mathcal{D}_1$	$\mathcal{B}_2$	$\mathcal{J}_2$	$\mathcal{J}_3$	$\mathcal{B}_3$	$\mathcal{D}_3$

Table 5: The change from the primal feasible pair $(\mathcal{A},\mathcal{C})$ to the primal feasible pair $(\mathcal{B},\mathcal{D})$ with $\mathcal{B} = \mathcal{B}_s \cup \mathcal{B}_1 \cup \mathcal{B}_2$ and $\mathcal{D} = \mathcal{D}_s \cup \mathcal{D}_1 \cup \mathcal{D}_2$.

To explain this diagram, we note that by definition we have $\mathcal{B}_s \subseteq \mathcal{A}$ and $\mathcal{D}_s \subseteq \mathcal{C}$. The extension of $\mathcal{B}_s$ to $\mathcal{B}$ is done by adding elements from $\mathcal{A} \setminus \mathcal{B}_s$, which are contained in $\mathcal{B}_1$, elements from $\mathcal{C} \setminus \mathcal{D}_s$, which are contained in $\mathcal{B}_2$ and elements from $\mathcal{I}$, which are contained in $\mathcal{B}_3$. Along the same lines $\mathcal{D}_s$ is extended to $\mathcal{D}$ and $(\mathcal{B},\mathcal{D})$ then form a primal feasible pair. Finally we have $\mathcal{J}_1 = \mathcal{A} \setminus (\mathcal{B}_s \cup \mathcal{B}_1 \cup \mathcal{D}_2)$, $\mathcal{J}_2 = \mathcal{C} \setminus (\mathcal{D}_s \cup \mathcal{B}_2 \cup \mathcal{D}_1)$ and $\mathcal{J}_3 = \mathcal{I} \setminus (\mathcal{B}_3 \cup \mathcal{D}_3)$.

Now let us recall some additional useful properties for two consecutive iterations given by the feasible pairs $(\mathcal{A},\mathcal{C})$ and $(\mathcal{B},\mathcal{D})$.

Lemma 7 ([16]) *Let (A, C) be a primal feasible pair. Then $\mathcal{B}_s \subseteq \mathcal{A}$, $\mathcal{D}_s \subseteq \mathcal{C}$ and $(\mathcal{B}_s = \mathcal{A}) \wedge (\mathcal{D}_s = \mathcal{C})$ if and only if $(\mathcal{A}, \mathcal{C})$ is the unique solution of a BLCP with a P-matrix. If $(\mathcal{A}, \mathcal{C})$ is not the unique solution of a BCLP with a P-matrix, then $(\mathcal{A} \neq \emptyset) \vee (\mathcal{C} \neq \emptyset)$, $(\mathcal{A} \neq \mathcal{B}) \vee (\mathcal{C} \neq \mathcal{D})$ and $x - y \neq 0$.*

For the following case distinction we assume $(\mathcal{A}, \mathcal{C})$ is not the unique solution of a BLCP with a P-matrix. We consider the following cases separately. Let x, y, α, β, γ, δ and $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ be as above.

Case 1: $m(\beta, \delta) < m(\alpha, \gamma)$.
In this case we can set $\mathcal{A} \leftarrow \mathcal{B}$, $\mathcal{C} \leftarrow \mathcal{D}$, and continue with the next iteration.

Case 2: $m(\beta, \delta) \geq m(\alpha, \gamma)$ and $m(\alpha, \gamma) = 1$.
In this case there is exactly one dual infeasible multiplier $\alpha_i < 0$ or $\gamma_i < 0$. Since a principal submatrix of Q also belongs to the class of P-matrices, the BCLP cannot have a solution with $x_i = b_i$ or $x_i = a_i$ respectively. Hence $a_i < x_i < b_i$ holds at the unique solution of the BLCP and a BLCP of order $(n - 1)$ with both α_i and γ_i fixed to zero is considered from now on.

Case 3: $m(\beta, \delta) \geq m(\alpha, \gamma)$ and $m(\alpha, \gamma) > 1$.
In this case we can (recursively) solve a BLCP of order $(n - |\mathcal{A}_0| - |\mathcal{C}_0|)$ with $x_{\mathcal{A}_0} = b_{\mathcal{A}_0}$, $x_{\mathcal{C}_0} = a_{\mathcal{C}_0}$, $x_{\overline{\mathcal{A}}_0} \leq b_{\overline{\mathcal{A}}_0}$, $x_{\overline{\mathcal{C}}_0} \geq a_{\overline{\mathcal{C}}_0}$, where the set $(\mathcal{A}_0 \cup \mathcal{C}_0)$ contains at least one and less than $m(\alpha, \gamma)$ elements and is constructed as follows:

(a) If $0 < |\mathcal{B}_s| + |\mathcal{D}_s| < m(\alpha, \gamma)$, we set $\mathcal{A}_0 = \mathcal{B}_s$ and $\mathcal{C}_0 = \mathcal{D}_s$.

(b) If $|\mathcal{B}_s| + |\mathcal{D}_s| \geq m(\alpha, \gamma)$, then we choose A_0 and C_0 such that $A_0 \cup C_0$ contains exactly $m(\alpha, \gamma) - 1$ indices from $\mathcal{B}_s \cup \mathcal{D}_s$ corresponding to the largest dual multipliers:

$$\mathcal{A}_0 = \{i \in \mathcal{B}_s : \alpha_i \text{ is one of the } (m(\alpha, \gamma) - 1) \text{ largest dual multipliers}\},$$
$$\mathcal{C}_0 = \{i \in \mathcal{D}_s : \gamma_i \text{ is one of the } (m(\alpha, \gamma) - 1) \text{ largest dual multipliers}\}.$$

(c) Finally if $\mathcal{B}_s = \mathcal{D}_s = \emptyset$, then $|\mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{J}_1 \cup \mathcal{J}_2| > 1$, because $|\mathcal{A}| + |\mathcal{C}| > 1$. In this case we set either $A_0 = \{j\}$ for some $j \in \mathcal{B}_1 \cup \mathcal{D}_2 \cup \mathcal{J}_1$ or $C_0 = \{j\}$ for some $j \in \mathcal{D}_1 \cup \mathcal{B}_2 \cup \mathcal{J}_2$.

In summary we select $\mathcal{A}_0$ and $\mathcal{C}_0$ according to the following rules:

$$\mathcal{A}_0 = \begin{cases} \mathcal{B}_s, & \text{if } 0 < |\mathcal{B}_s| + |\mathcal{D}_s| < m(\alpha, \gamma), \\ \{i \in \mathcal{B}_s : \alpha_i \text{ is one of the } (m(\alpha, \gamma) - 1) \\ \qquad\qquad \text{largest dual multipliers}\}, & \text{if } |\mathcal{B}_s| + |\mathcal{D}_s| \geq m(\alpha, \gamma), \\ \{j\} \text{ for some } j \in \mathcal{B}_1 \cup \mathcal{D}_2 \cup \mathcal{J}_1, & \text{if } \mathcal{B}_s = \mathcal{D}_s = \mathcal{C}_0 = \emptyset, \\ \emptyset, & \text{if } \mathcal{B}_s = \mathcal{D}_s = \emptyset \text{ and } \mathcal{C}_0 \neq \emptyset. \end{cases} \tag{16}$$

$$\mathcal{C}_0 = \begin{cases} \mathcal{D}_s, & \text{if } 0 < |\mathcal{B}_s| + |\mathcal{D}_s| < m(\alpha, \gamma), \\ \{i \in \mathcal{D}_s : \gamma_i \text{ is one of the } (m(\alpha, \gamma) - 1) \\ \qquad\qquad \text{largest dual multipliers}\}, & \text{if } |\mathcal{B}_s| + |\mathcal{D}_s| \geq m(\alpha, \gamma), \\ \{j\} \text{ for some } j \in \mathcal{D}_1 \cup \mathcal{B}_2 \cup \mathcal{J}_2, & \text{if } \mathcal{B}_s = \mathcal{D}_s = \mathcal{A}_0 = \emptyset, \\ \emptyset, & \text{if } \mathcal{B}_s = \mathcal{D}_s = \emptyset \text{ and } \mathcal{A}_0 \neq \emptyset. \end{cases} \tag{17}$$

The active sets of the solution of the subproblem with $x_{\mathcal{A}_0} = b_{\mathcal{A}_0}$, $x_{\mathcal{C}_0} = a_{\mathcal{C}_0}$, $x_{\overline{\mathcal{A}}_0} \leq b_{\overline{\mathcal{A}}_0}$, $x_{\overline{\mathcal{C}}_0} \geq a_{\overline{\mathcal{C}}_0}$ are denoted by $\mathcal{B}_0$ and $\mathcal{D}_0$. As $\mathcal{B}_0$ and $\mathcal{D}_0$ contain no dual infeasible multipliers β_i and δ_i respectively, only the dual multipliers β_i, $i \in \mathcal{A}_0$ and δ_i, $i \in \mathcal{C}_0$ could be infeasible. If $\mathcal{B}_s \cup \mathcal{D}_s \neq \emptyset$ all infeasible dual multipliers from α and γ are contained in the subproblem and $|A_0| + |C_0| < m(\alpha, \gamma)$ holds. If $\mathcal{B}_s = \mathcal{D}_s = \emptyset$

we have either $A_0 = \{j\}$ for some $j \in \mathcal{B}_1 \cup \mathcal{D}_2 \cup \mathcal{J}_1$ or $C_0 = \{j\}$ for some $j \in \mathcal{D}_1 \cup \mathcal{B}_2 \cup \mathcal{J}_2$ and thus $|\mathcal{A}_0| + |\mathcal{C}_0| < 2 \leq m(\alpha, \gamma)$ also holds. Hence in both cases the sets $\mathcal{B} = \mathcal{A}_0 \cup \mathcal{B}_0$ and $\mathcal{D} = \mathcal{C}_0 \cup \mathcal{D}_0$ give a primal-dual triple $[y, \beta, \delta]$ =KKT$(\mathcal{B}, \mathcal{D})$ with y primal feasible and $m(\beta, \delta) < m(\alpha, \gamma)$.

We summarize in Table 6 the recursive SN-method for BLCPs with CDI merit function $m(\alpha, \gamma)$ and algorithmic enhancement, which produces primal feasible iterates in each iteration As either $m(\alpha, \gamma)$ or the problem dimension (i.e. the number of complementary triples (x, α, γ)) strictly decreases in each iteration of the algorithm, the following result holds:

Theorem 8 *For a BLCP with a P-matrix the recursive SN-method with the CDI merit function $m(\alpha, \gamma)$ and algorithmic enhancement as given in Table 6 terminates in a finite number of iterations with the unique solution of LCP.*

In the following section we report the experiments with the SN-method and our two variants of the recursive SN-method on a large variety of benchmark instances, showcasing that this version of the recursive SN-method with the CDI merit function and algorithmic enhancement performs very well in practice for all the tested LCPs with symmetric and nonsymmetric P-matrices.

5 Computational Experience

In this section we compare the original SN-method [SN] [1, 2, 25] and the recursive SN-method with merit function $J(x)$ [RSN] [16] with our newly proposed recursive SN-method with the CDI merit function $m(\alpha)$ without algorithm enhancement [RSNd] and with it [RSNde]. Standard conjugate and projected gradient, active set and interior point algorithms can also be used to solve an LCP with a symmetric P-matrix (positive definite) by finding the optimal solution of the quadratic program (2). Furthermore, interior-point algorithms can also be used to solve an LCP with a nonsymmetric P-matrix [23]. According to a computational study reported in [16], these methods are in general less efficient than the recursive SN-method for solving large-scale LCPs with symmetric P-matrices. Since the recursive SN-method with the CDI merit fucntion with algorithm enhancement [RSNde] will be shown in this paper to be competitive with the recursive SN-method [RSN] for LCPs with symmetric P-matrices, it is expected that interior-point algorithms will perform in general worse than [RSNde] for the solution of large-scale LCPs with nonsymmetric P-matrices. An LCP with a symmetric or a nonsymmetric P-matrix can aslo be solved by Lemke's method [26] or the simple principal pivoting algorithms [6, 7, 14, 21, 29, 35, 36] described in [5, 31]. However, these methods only perform updates of the active set of one element in each iteration and are quite slow for large-scale problems. The same reason rules out the use of an active set method [32] for solving a large-scale LCP with a nonsymmetric P-matrix by computing the unique stationary point of the quadratic program (3).

In the tables presented in this section, the column labels have the following meaning:

- The condition number of Q is estimated with the MATLAB command `condest`. Its average is given in the column labeled *cond*(Q).
- In the column labeled *solve* we show how often on average we solve x =KKT$(\mathcal{A})$ to determine the primal inactive variables. We recall that this reflects the main computational effort of the methods considered in our study.
- The column labeled *fail* contains the number of trials, where [SN] or [RSN] failed to solve the problem.
- ℓ gives the maximal depth of levels of subproblems (recursive calls).
- r denotes the number of dimension reductions performed over all runs, i.e. the number of times **Case 2** is entered.

Recursive SN-method for BLCPs with CDI merit function and algorithmic enhancement

Input: A P-matrix Q, $q, a, b \in I\!R^n$, $a < b$, and starting active sets $\mathcal{A} \cup \mathcal{C} \subseteq \mathcal{N}$, $\mathcal{A} \cap \mathcal{C} = \emptyset$.
Output: Unique solution (x, α, γ) of BLCP and corresponding active sets $(\mathcal{A}, \mathcal{C})$.

```
$x = \mathrm{KKT}(\mathcal{A},\mathcal{C})$, $h = 0$, $\overline{m} = n + 1$.
while $(\mathcal{A},\mathcal{C})$ not primal feasible:
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{i \in \overline{\mathcal{A}} : x_i \geq b_i\}$.
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{i \in \overline{\mathcal{C}} : x_i \leq a_i\}$.
    $x = \mathrm{KKT}(\mathcal{A},\mathcal{C})$.
endwhile
$[\alpha, \gamma] = \mathrm{KKT}(\mathcal{A},\mathcal{C})$.
Set $\overline{m} = m(\alpha, \gamma)$.
while $m(\alpha, \gamma) > 0$
    $\mathcal{B}_s \leftarrow \{i \in \mathcal{A} : \alpha_i \geq 0\}$, $\mathcal{B} \leftarrow \mathcal{B}_s$.
    $\mathcal{D}_s \leftarrow \{i \in \mathcal{C} : \gamma_i \geq 0\}$, $\mathcal{D} \leftarrow \mathcal{D}_s$.
    $y = \mathrm{KKT}(\mathcal{B},\mathcal{D})$.
    while $(\mathcal{B},\mathcal{D})$ not primal feasible
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{i \in \overline{\mathcal{B}} : y_i \geq b_i\}$.
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{i \in \overline{\mathcal{D}} : y_i \leq a_i\}$.
        $y = \mathrm{KKT}(\mathcal{B},\mathcal{D})$.
    endwhile
    $[\beta, \delta] = \mathrm{KKT}(\mathcal{B},\mathcal{D})$.
    Case 1: $m(\beta, \delta) < \overline{m}$
        Set $\mathcal{A} \leftarrow \mathcal{B}$, $\mathcal{C} \leftarrow \mathcal{D}$, $\alpha \leftarrow \beta$, $\gamma \leftarrow \delta$ and $h = 0$.
    Case 2: $m(\beta, \delta) \geq \overline{m}$ and $m(\alpha, \gamma) = 1$
        Solve recursively problem (12) with the bound on $j$ with either $\alpha_j < 0$ or $\gamma_j < 0$ removed.
    Case 3: $m(\beta, \delta) \geq \overline{m}$ and $\overline{m} > 1$
        if $h = 1$
            Select $\mathcal{A}_0$ and $\mathcal{C}_0$ according to (16) and (17) respectively.
        else
            Set $\mathcal{A}_0 \leftarrow \mathcal{B}_s$ and $\mathcal{C}_0 \leftarrow \mathcal{D}_s$.
        endif
        Let $(\mathcal{B}_0, \mathcal{D}_0)$ be the active sets of the solution of problem (12)
        with $x_{\mathcal{A}_0} = b_{\mathcal{A}_0}, x_{\mathcal{C}_0} = a_{\mathcal{C}_0}, x_{\overline{\mathcal{A}}_0} \leq b_{\overline{\mathcal{A}}_0}, x_{\overline{\mathcal{C}}_0} \geq a_{\overline{\mathcal{C}}_0}$.
        $\mathcal{B}^* \leftarrow \mathcal{A}_0 \cup \mathcal{B}_0$, $\mathcal{D}^* \leftarrow \mathcal{C}_0 \cup \mathcal{D}_0$.
        $[y, \beta, \delta] = \mathrm{KKT}(\mathcal{B}^*,\mathcal{D}^*)$.
        if $m(\beta, \delta) < \overline{m}$
            Set $\overline{m} = m(\beta, \delta)$, $\mathcal{A} \leftarrow \mathcal{B}^*$, $\mathcal{C} \leftarrow \mathcal{D}^*$, $\alpha \leftarrow \beta$, $\gamma \leftarrow \delta$ and $h = 0$.
        else
            Set $h = 1$ and repeat Case 3.
        endif
endwhile
```

Table 6: Description of the recursive SN-method for BLCPs with the CDI merit function $m(\alpha, \gamma)$ and algorithmic enhancement.

- *dens* denotes the average density of the matrix Q.

All experiments were conducted on an Intel Xeon 5160 processor with 3 GHz and 2 GB RAM. The corresponding Matlab code for generating all instances and solving them with the various methods discussed is available from `http://philipphungerlaender.jimdo.com/qp-code/`. In Subsection 5.1 we demonstrate on a variety of benchmark sets that the recursive SN-method with the CDI merit function $m(\alpha)$ and algorithmic enhancement [RSNde] as given in Table 4 is quite competitive with the recursive SN-method [RSN] for the solution of LCPs and BLCPs with symmetric P-matrices (i.e., strictly convex quadratic programs on the nonnegative orthant and with bounds). Furthermore, in Subsection 5.2, we report a similar performance of [RSNde] for the solution of LCPs and BLCPs with nonsymmetric P-matrices.

5.1 LCPs and BLCPs with Symmetric P-Matrices

In this subsection we use part of the benchmark set from [16] to compare the SN-method [SN] and the recursive SN-method [RSN] with the new versions [RSNd] and [RSNde] of the recursive SN-method introduced in this paper on the solution of LCPs and BLCPs with symmetric P-matrices (positive definite matrices).

First we compare our methods on randomly generated problems (see [25, Section 4.1.]) of size $n = 2000$, whose matrices Q have the same structure but different condition numbers. The random instances are generated starting with a sparse singular positive semidefinite matrix Q_0. The matrices Q are obtained from Q_0 by adding a small multiple of the identity matrix I,

$$Q = Q_0 + \varepsilon I,$$

where $\varepsilon \in \{1, 10^{-5}, 10^{-10}, 10^{-14}\}$. Hence all these matrices are symmetric P (positive definite) and their condition numbers increase with the reduction of ε.

In Table 7 we summarize the performance of our methods on these random instances. Each line represents the number of linear systems (5a) required to be solved and CPU time in average over 2000 trials for each value of ε. We used the initial active set $\mathcal{A}^0 = \mathcal{N}$ (initial active set contains all indices) to avoid solving a linear system in the first iteration. The effort for the backsolve in (5b) to get the dual variables is negligible.

	[SN]		[RSN]		[RSNd]		[RSNde]	
cond(Q)	time	solve	time	solve	time	solve	time	solve
$\approx 10^3$	0.13	4.84	0.15	9.48	0.15	9.48	0.15	9.48
$\approx 10^8$	0.17	8.03	0.24	12.10	0.25	12.94	0.24	12.15
$\approx 10^{13}$	0.17	8.05	0.24	15.06	0.25	16.30	0.24	15.08
$\approx 10^{17}$	0.17	8.05	0.24	15.08	0.25	16.31	0.24	15.08

Table 7: Comparison of variants of the SN-method with initial active set is $\mathcal{N}$ for LCPs of order $n = 2000$.

We note that the SN-method never cycles in these experiments even though in principle this is possible. For ill-conditioned problems the number of linear systems (5a) to be solved is slightly smaller for [SN] than for the other methods at the cost of risking to run into a cycle. [RSN] and [RSNde] show basically the same performance and the performance of [RSNd] is slightly worse than the other two methods on these random instances.

Next we consider another benchmark set of random LCPs with symmetric P-matrices (i.e., strictly convex quadratic programs) generated according to the instructions presented in [8]. Specifically, we generated Q via Matlab's `sprandsym` routine. Both the average density and the average condition number are given in the following two tables. Then we generated the unique solution x of LCP via the Matlab `randn` routine. With the solution in hand, we choose the components of the right-hand side vector q of the LCP and, for the case of a BLCP, lower and upper bounds so that at the unique solution the number of

- the active variables and the inactive variables are roughly equal for the LCP and
- the active variables on the upper bound, the active variables on the lower bound and the inactive variables are all roughly equal for the BLCP.

In Table 8 we compare the performance of the four SN-methods mentioned before. Each line represents the average over 10 trials. We consider the same sizes n as in [8], but we allow the condition number to go up to $\approx 10^{10}$ as compared to $\approx 10^{6}$ in [8]. Since the code from [8] is not available to us, we can not make a direct comparison.

These special random instances are the most difficult benchmark set for any type of SN-method that we are aware of. In particular [SN] fails completely once the condition number and the problem size are very large. So, for these instances the globalizations of [SN] show their full power. But still the number *solve* of linear systems (5a) to be solved is much larger for [RSN], [RSNde] and [RSNd] than for all other classes of instances considered in both this paper and [16].

As a first conclusion, we see a clear dominance from [RSN] and [RSNde] to [SN] and [RSNd]. [SN] and [RSNd] show similar results as the other two methods for well-conditioned problems. But if the condition number of the matrix Q is large [SN] fails completely and [RSNd] performs poorly.

The numerical results with this new set of test problems clearly indicate that the number of dimension reductions and recursions indicated by the values of ℓ and r can be very large for the version [RSNd]. For ill-conditioned instances [RSNd] suffers from its large number of recursions and in some cases needs more than 100 times longer than [RSNde]. This goes in line with our intuition that asking for a decrease of the CDI merit function $m(\alpha)$ in each iteration is more challenging than demanding a decrease of the quadratic merit function $J(x)$ in each iteration. It is quite impressive that the inclusion of our simple algorithmic enhancement leads to such a substantial drop in recursions and iterations. Hence from now on we do not consider the version [RSNd] for comparison with other methods.

As before, the two versions [RSN] and [RSNde] perform similarly for this new set of test problems and there is no clear winner with respect to running time. Both methods go into recursions, but the depth of the recursion is no larger than 3. The total number of linear systems (5a) to be solved stays typically below 70 for problems with $cond(Q) \approx 10^{6}$ and it may rise up to 200 for problems with large condition number $cond(Q) \approx 10^{10}$.

The corresponding results for BLCPs on the second set of test problems are displayed in Table 9 and lead to the same conclusions that have been stated for the LCP.

Finally let us point out that for all other benchmark instances from [16], which were motivated by applications in mathematical physics and combinatorial optimization, the four SN-methods exhibit the same performance as neither of them ever has to enter into a recursion or dimension reduction. So, the random instances considered above are clearly the most challenging test problems for the four SN-methods that we have been faced so far. Therefore we refrain from restating additional test results and we refer to [16, Section 6] for further details.

5.2 LCPs with Nonsymmetric P-Matrices

As discussed before, the version [RSN] cannot be employed for solving an LCP with a nonsymmetric P-matrix and it will not be considered in our experiments. The original SN-method [SN] has shown to be very useful to solve this LCP in this case [11], despite there is not guarantee that the algorithm converges to the solution of LCP unless in very special cases [31, 34]. Since the version [RSNd] does not perform well in the symmetric case, we will only report the results of our experiments with the algorithms [SN] and [RSNde].

n	dens	cond(Q)	[SN]			[RSN]					[RSNd]					[RSNde]				
			time	solve	fail	time	iter	solve	ℓ	r	time	iter	solve	ℓ	r	time	iter	solve	ℓ	r
5000	10^{-1}	$\approx 10^2$	9.23	4.6	0	15.96	4.4	6.9	0	0	15.96	4.4	6.9	0	0	15.96	4.4	6.9	0	0
5000	10^{-1}	$\approx 10^6$	15.35	13.7	0	33.77	9.6	45.6	1	0	35.20	4.0	63.8	3	12	34.02	9.9	48.5	1	6
5000	10^{-1}	$\approx 10^{10}$	33.06	33.4	0	66.17	12.3	124.4	2	0	157.93	4.3	519.8	31	72	72.68	13.1	152.8	2	28
5000	10^{-2}	$\approx 10^2$	3.79	4.4	0	6.51	4.1	6.3	0	0	6.51	4.1	6.3	0	0	6.51	4.1	6.3	0	0
5000	10^{-2}	$\approx 10^6$	4.71	16.8	4	10.64	10.9	47.0	2	0	36.39	5.8	211.7	6	19	11.25	10.3	49.2	2	13
5000	10^{-2}	$\approx 10^{10}$	-	-	10	15.05	9.7	96.0	3	0	5520.86	4.2	1136008.7	1543	12971	16.69	10.3	127.9	3	22
5000	10^{-3}	$\approx 10^2$	0.09	5.1	0	0.17	4.5	6.7	1	0	0.16	4.5	6.5	0	1	0.16	4.5	6.5	0	1
5000	10^{-3}	$\approx 10^6$	0.10	8.6	5	0.22	6.3	13.9	1	0	4.95	6.8	582.3	455	2	0.21	6.7	16.6	2	8
5000	10^{-3}	$\approx 10^{10}$	-	-	10	0.37	8.3	21.3	1	0	24.14	8.3	2308.1	1102	56	0.34	9.9	27.8	2	9
10000	10^{-3}	$\approx 10^2$	3.09	4.9	0	4.66	4.1	6.4	0	0	4.66	4.1	6.4	0	0	4.66	4.1	6.4	0	0
10000	10^{-3}	$\approx 10^6$	1.77	11.0	9	9.31	8.3	25.3	1	0	381.47	9.1	3283.1	1424	9	9.91	11.3	37.7	2	10
10000	10^{-3}	$\approx 10^{10}$	-	-	10	16.81	10.0	52.8	2	0	651.04	9.3	7363.3	1380	12	16.27	12.5	61.0	2	15

Table 8: Comparison of four variants of the SN-method for random LCPs generated according to [8].

n	dens	cond(Q)	[SN]			[RSN]					[RSNde]				
			time	solve	fail	time	iter	solve	ℓ	r	time	iter	solve	ℓ	r
5000	10^{-1}	$\approx 10^2$	3.04	6.1	0	16.68	4.2	12.0	0	0	17.76	4.1	15.4	1	9
5000	10^{-1}	$\approx 10^6$	43.97	26.1	0	43.21	14.0	66.5	1	0	45.19	6.2	181.5	3	25
5000	10^{-1}	$\approx 10^{10}$	-	-	10	104.20	14.3	258.1	2	0	118.80	7.1	751.4	12	136
5000	10^{-2}	$\approx 10^2$	4.57	6.2	0	5.83	3.6	10.6	0	0	6.03	3.5	11.1	1	7
5000	10^{-2}	$\approx 10^6$	8.64	27.6	5	8.83	12.0	57.1	2	0	8.95	11.6	59.6	2	25
5000	10^{-2}	$\approx 10^{10}$	-	-	10	21.11	10.5	157.9	3	0	21.54	10.8	162.0	7	44
5000	10^{-3}	$\approx 10^2$	0.15	5.8	0	0.15	3.6	9.4	0	0	0.17	3.5	9.4	1	2
5000	10^{-3}	$\approx 10^6$	0.25	12.8	4	0.23	6.7	19.2	1	0	0.22	6.8	19.8	1	2
5000	10^{-3}	$\approx 10^{10}$	-	-	10	0.30	7.4	25.6	2	0	0.28	7.7	25.9	2	5
10000	10^{-3}	$\approx 10^2$	3.28	6.0	0	3.11	3.6	9.6	0	0	3.11	3.6	9.6	0	2
10000	10^{-3}	$\approx 10^6$	-	-	10	5.07	8.9	31.5	2	0	4.64	9.5	32.1	1	8
10000	10^{-3}	$\approx 10^{10}$	-	-	10	10.18	10.7	62.7	4	0	8.79	11.9	66.0	3	14

Table 9: Comparison of three variants of the SN-method for random BLCPs generated according to [8].

In the experiments to be reported in this section, we first consider the LCP presented in [30] that has proven to be difficult for the simple and block principal pivoting methods, see [11, 30, 31]. For its definition, let

$$Q = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 2 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 2 & \dots & 2 & 1 \end{pmatrix} \tag{18}$$

be a dense lower triangular $n \times n$ matrix with diagonal elements equal to one and off-diagonal elements equal to two. Furthermore we set $q = -e$, where e is the vector of all ones. It is easy to verify that $x = e_1$, and $\alpha = e - e_1$ is the unique solution of LCP, where e_1 is a unit vector with 1 in the first component. The corresponding active set is $\mathcal{A}^* = \{2, \dots, n\}$.

We investigated the transition graph of [SN] for $n \leq 5$ and noted that in all cases it turned out to be a complete binary tree rooted at $\mathcal{A}^*$. We conjecture that this is true for all n. Since a complete binary tree on 2^n nodes has 2^{n-1} leaves, we would expect that for half of all possible starting active sets, the number of SN iterations should be $n + 1$, if $\mathcal{N}$ is the initial active set. This is confirmed by our computational results given below. For instance starting with $\mathcal{A} = \mathcal{N}$, [SN] required exactly $n + 1$ iterations and n linear systems (5a) to reach the unique solution.

This is an interesting example where the original SN-method does not to cycle and takes a number of iterations proportional to n. In Table 10 we compare our algorithms using random active starting sets and state the average values over 10 runs. It is interesting to note that [RSNde] terminates with significantly fewer iterations than [SN]. While [SN] needs approximately n iterations on all test problems, the number of linear systems (5a) always stays below 20 for [RSNde] and proves to be quite insensitive to the dimension n.

n	[SN]			[RSNde]				
	time	iter	fail	time	iter	solve	ℓ	r
500	1.34	500.1	0	0.00	1.3	9.9	0	1
1000	15.81	1000.1	0	0.01	1.5	11.7	0	2
2000	146.22	1999.3	0	0.08	1.5	12.6	0	0
5000	2767.23	4999.3	0	0.70	1.7	14.6	0	5

Table 10: Comparison of [SN] and [RSNde] on LCPs with the nonsymmetric P-matrix (18) suggested by Murty [31].

Next we revisit the random instances from the previous subsection. In order to make our benchmark instances totally nonsymmetric we randomly either multiply $Q_{i,j}$ or $Q_{j,i}$, $i < j$ by -1. Hence the matrix Q is of the form $D + E$, where D is a diagonal matrix with positive diagonal elements and E is a skew symmetric matrix. So, Q is a nonsymmetric positive definite matrix and hence a nonsymmetric P-matrix.

In Table 11 we summarize the results of our experiments with the nonsymmetric version of the random instances from Table 7. As for the results with symmetric P-matrices, each line represents the average over 2000 trials. [SN] with $\mathcal{N}$ as the initial active set works very well on this benchmark set. [RSNde] with the same initial active set also faces no problems and in fact shows a similar performance on all instances.

Next we consider the nonsymmetric version of the random instances from Table 8 that have been generated in a way similar to the previous test problems. We summarize the corresponding results in Table 12. As for the previous experiments, $\mathcal{N}$ is always the initial active set for both methods and each line represents the average over 2000 trials. [SN] again performs quite well for these test problems. Furthermore [RSNde] shows the exactly same good performance on all instances and does not have to go into recursion once.

	[SN]			[RSNde]			
ε	time	solve	fail	time	solve	ℓ	r
1	0.10	4.13	0	0.17	7.94	0	735
10^{-5}	0.11	4.32	0	0.18	8.64	0	596
10^{-10}	0.11	4.39	0	0.19	8.79	1	587
10^{-14}	0.11	4.39	0	0.19	8.80	1	587

Table 11: Comparison of [SN] and [RSNde] on LCPs with random nonsymmetric P-matrices and $n = 2000$.

n	dens	cond(Q)	[SN]			[RSNde]				
			time	solve	fail	time	iter	solve	ℓ	r
5000	10^{-1}	$\approx 10^2$	21.98	5.5	0	48.98	6.1	14.2	0	1
5000	10^{-1}	$\approx 10^6$	23.14	6.8	0	51.25	7.1	18.7	0	2
5000	10^{-1}	$\approx 10^{10}$	24.81	7.5	0	52.82	7.5	20.9	0	1
5000	10^{-2}	$\approx 10^2$	10.15	5.4	0	19.66	5.8	12.2	0	2
5000	10^{-2}	$\approx 10^6$	9.75	6.3	0	18.21	6.5	14.8	0	4
5000	10^{-2}	$\approx 10^{10}$	9.20	6.4	0	18.10	6.8	16.1	0	6
5000	10^{-3}	$\approx 10^2$	0.28	5.1	0	0.44	5.7	9.7	0	5
5000	10^{-3}	$\approx 10^6$	0.25	5.5	0	0.38	5.5	10.3	0	1
5000	10^{-3}	$\approx 10^{10}$	0.23	5.3	0	0.34	5.4	10.1	0	4
10000	10^{-3}	$\approx 10^2$	7.68	5.3	0	13.15	5.5	10.8	0	4
10000	10^{-3}	$\approx 10^6$	6.22	5.7	0	11.36	6.0	13.0	0	0
10000	10^{-3}	$\approx 10^{10}$	6.48	6.3	0	11.34	6.2	14.0	0	3

Table 12: Comparison of [SN] and [RSNde] for LCPs with random nonsymmetric P-matrices generated according to [8].

Finally let us summarize the main findings of our computational experiments:

- For most benchmark large-scale LCP instances with symmetric P-matrices, all the algorithms [SN], [RSN], [RSNd] and [RSNde] are very efficient. The only exception that we found were those LCPs with ill-conditioned random data generated according to the instructions presented in [8]. For these instances [SN] and [RSNd] run into serious difficulties while [RSN] and [RSNde] perform equally well.
- For most benchmark instances with nonsymmetric P-matrices [SN] and [RSNde] are very efficient. The only exception found are the instances suggested by Murty [31]. For such instances [SN] needs $\approx n$ instances, while [RSNde] performs very well.
- [RSN] and [RSNde] are the methods of choice for solving an LCP and a BLCP with a symmetric P-matrix. For an LCP and a BLCP with a nonsymmetric P-matrix we suggest to use [RSNde].

6 Conclusion

We introduce a new recursive variant of the semi-smooth Newton (SN) method for finding the unique solution of a Linear Complementarity Problem (LCP) with a (symmetric or nonsymmetric) P-matrix. The algorithm is an extension of the recursive SN-method that has been proposed by Hungerländer and Rendl [16] for an LCP with a symmetric P-matrix (i.e., positive definite). The new recursive SN-method uses a Cardinal Dual Infeasibility merit function to guarantee finite convergence. This merit function can be used for a symmetric or nonsymmetric LCP with a P-matrix, contrary to the quadratic function employed in the old SN-method that may only be used for symmetric LCPs. An extension of the new recursive SN-method to the so-called Bound Linear Complementarity Problem (BLCP) is also introduced with similar properties.

Computational experiments reported in this paper indicate that the new approach is quite efficient for the solution of large-scale LCPs and BLCPs with symmetric and nonsymmetric P-matrices. This computational study also confirms that the original SN-method (or block principal pivoting method) [11, 15, 25] is quite efficient for finding the unique solution of these problems. This last algorithm does not use recursions but unfortunately can cycle. The development of conditions for finite convergence of the SN-method is an important topic of our future research. The use of SN- and recursive SN-methods in splitting algorithms [5] for solving LCPs and BLCPs with more general classes of matrices is another interesting area that should deserve our attention in the future. These algorithms will allow the solution of more difficult LCPs and BLCPs that appear quite often in applications [5, 10, 31].

References

[1] M. Bergounioux, K. Ito, and K. Kunisch. Primal-Dual Strategy for Constrained Optimal Control Problems. *SIAM Journal on Control and Optimization*, 37:1176–1194, 1999.

[2] M. Bergounioux, M. Haddou, M. Hintermüller, and K. Kunisch. A comparison of interior point methods and a Moreau-Yosida based active set strategy for constrained optimal control problems. *SIAM Journal on Optimization*, 11(2):495–521, 2000.

[3] R. H. Byrd, G. M. Chin, J. Nocedal, and F. Oztoprak. A family of second-order methods for convex ℓ_1 regularized optimization. *Mathematical Programming*, 159(1):435–467, 2016.

[4] R. Chandrasekaran. A special case of the complementary pivot problem. *Opsearch*, 7:263–268, 1970.

[5] R. Cottle, J. S. Pang, and R. Stone. *The Linear Complementarity Problem.* Academic Press, New York, 1992.

[6] R. W. Cottle and G. B. Dantzig. Complementarity pivot theory of mathematical programming. In G. B. Dantzig and A. F. Veinott, editors, *Mathematics of Decision Sciences: Part I*, pages 115–136. American Mathematical Society, Rhode Island, 1968.

[7] Z. Csizmadia and T. Illés. New criss-cross type algorithms for linear complementarity problems with sufficient matrices. *Optimization Methods and Software*, 21(2):247–266, 2006.

[8] F. E. Curtis, Z. Han, and D. P. Robinson. A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization. *Computational Optimization and Applications*, 60(2): 311–341, 2015.

[9] Z. Dostál and J. Schöberl. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Computational Optimization and Applications*, 30:23–43, 2005.

[10] F. Facchinei and J. S. Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems*, volume 1 and 2. Springer, New York, 2002.

[11] L. Fernandes, J. Júdice, and J. Patricio. An investigation of interior-point and block pivoting algorithms for large-scale symmetric monotone linear complementarity problems. *Computational Optimization and Applications*, 5(1):49–77, 1996.

[12] A. Fischer. A special Newton-type optimization method. *Optimization*, 24:269–284, 1992.

[13] I. B. Gharbia and C. J. Gilbert. Nonconvergence of the plain Newton-min algorithm for linear complementarity problems with a P-matrix. *Mathematical Programming*, 134(2):349–364, 2012.

[14] R. L. Graves. A principal pivoting simplex algorithm for linear and quadratic programming. *Operations Research*, 15:482–494, 1967.

[15] M. Hintermüller, K. Ito, and K. Kunisch. The primal-dual active set strategy as a semi-smooth newton method. *SIAM Journal on Optimization*, 13(3):865–888, 2003.

[16] P. Hungerländer and F. Rendl. A feasible active set method for strictly convex problems with simple bounds. *SIAM Journal on Optimization*, 25(3):1633–1659, 2015.

[17] P. Hungerländer and F. Rendl. An infeasible active set method with combinatorial line search for

convex quadratic problems with bound constraints. Technical report, Alpen-Adria Universität Klagenfurt, Mathematics, Optimization Group, TR-AAUK-M-O-16-08-03, 2016.

[18] T. Illés, J. Peng, C. Roos, and T. Terlaky. A strongly polynomial rounding procedure yielding a maximally complementary solution for $P_*(\kappa)$ linear complementarity problems. *SIAM Journal on Optimization*, 11(2):320–340, 2000.

[19] J. J. Júdice and F. M. Pires. A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers and Operations Research*, 21(5):587–596, 1994.

[20] C. Kanzow. Inexact semismooth newton method for large-scale complementarity problems. *Optimization Methods and Software*, 19:309–325, 2004.

[21] E. L. Keller. The general quadratic optimization problem. *Mathematical Programming*, 5:311–337, 1973.

[22] M. Kojima, S. Mizuno, and A. Yoshise. A polynomial-time algorithm for a class of linear complementarity problems. *Mathematical Programming*, 44:1–26, 1989.

[23] M. Kojima, N. Meggido, T. Noma, and A. Yoshise. *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems.* Lecture Notes in Computer Science 538. Springer-Verlag, Berlin, 1991.

[24] M. M. Kostreva. Block pivot methods for solving the complementarity problem. *Linear Algebra and its Applications*, 21:207–215, 1978.

[25] K. Kunisch and F. Rendl. An infeasible active set method for convex problems with simple bounds. *SIAM Journal on Optimization*, 14(1):35–52, 2003.

[26] C. E. Lemke. Recent results on complementarity problems. In G. B. Dantzig and A. F. Veinott, editors, *Mathematics of Decision Sciences: Part I*, pages 95–114. American Mathematical Society, Rhode Island, 1968.

[27] T. D. Luca., F. Facchinei, and C. Kanzow. A semi-smooth equation approach to the solution of nonlinear complementarity problems. *Mathematical Programming*, 75:407–439, 1996.

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

[29] K. Murty. Note on a Bard-type scheme for solving the complementarity problem. *Opsearch*, 11: 123–130, 1974.

[30] K. Murty. Computational complexity of complementary pivot methods. *Mathematical Programming Study*, 7:61–73, 1978.

[31] K. Murty. *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, 1988.

[32] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, New York, second edition, 2006.

[33] U. Schäfer. A linear complementarity problem with a P-matrix. *SIAM Review*, 46(2):189–201, 2004.

[34] S. Takriti and K. G. Murty. On the convergence of the block principal pivotal algorithm for the LCP. *European Journal of Operational Research*, 102(3):657–666, 1997.

[35] C. Van de Panne and A. Whinston. The simplex and dual method for quadratic programming. *Operational Research Quarterly*, 15:355–388, 1964.

[36] L. Van der Heyden. A variable dimension algorithm for the linear complementarity problem. *Mathematical Programming*, 19:328–346, 1980.