Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: http://www.tandfonline.com/loi/goms20

Splitting methods for the Eigenvalue Complementarity Problem

Alfredo N. Iusem, Joaquim J. Júdice, Valentina Sessa & Paula Sarabando

To cite this article: Alfredo N. Iusem, Joaquim J. Júdice, Valentina Sessa & Paula Sarabando (2018): Splitting methods for the Eigenvalue Complementarity Problem, Optimization Methods and Software, DOI: 10.1080/10556788.2018.1479408

To link to this article: https://doi.org/10.1080/10556788.2018.1479408

Published online: 08 Jun 2018.

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2018.1479408

Taylor & Francis
Taylor & Francis Group

Splitting methods for the Eigenvalue Complementarity Problem

Alfredo N. Iusem[a], Joaquim J. Júdice[b], Valentina Sessa [c] and Paula Sarabando[d]

[a]Instituto de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, Brazil; [b]Instituto de Telecomunicações, Lisboa, Portugal; [c]Department of Engineering, State University of Rio de Janeiro (UERJ), Rio de Janeiro, Brazil; [d]Polytechnic Institute of Viseu and INESC Centro, Viseu, Portugal

ABSTRACT

We study splitting methods for solving the Eigenvalue Complementarity Problem (EiCP). We introduce four variants, which depend on the properties (symmetry, nonsymmetry, positive definite, negative definite, indefinite) of the matrices included in the definition of EiCP. Convergence analyses for each one of these versions of the splitting method are discussed. Special choices for the splitting matrices associated with these versions are recommended and tested on the solution of small and large symmetric and nonsymmetric EiCPs. These experiments show that the four versions of the splitting method work well at least for some choices of the splitting matrices. Furthermore, these versions of the splitting methods seem to be competitive with the most efficient state-of-the-art algorithms for the solution of EiCP.

ARTICLE HISTORY
Received 12 July 2017
Accepted 17 May 2018

KEYWORDS
Eigenvalue problems; complementarity problems; nonlinear programming; global optimization

MATHEMATICS SUBJECT CLASSIFICATIONS
93B60; 90C33; 90C30; 90C26

1. Introduction

Given a matrix $A \in \mathbb{R}^{n\times n}$ and a positive definite (PD) matrix $B \in \mathbb{R}^{n\times n}$ (i.e. $x^tBx > 0$ for all $x \neq 0$), the *Eigenvalue Complementarity Problem* (**EiCP**) [23,24] consists of finding a real number λ and vectors $x \in \mathbb{R}^n \setminus \{0\}$ and $w \in \mathbb{R}^n$ such that

$$\textbf{EiCP}: \quad w = (\lambda B - A)x \tag{1}$$

$$w \geq 0, x \geq 0 \tag{2}$$

$$x^tw = 0. \tag{3}$$

We also use the notation EiCP(A, B) to represent an EiCP with given matrices A and B. The problem finds many applications in engineering [22,24] and can be seen as a generalization of the well-known Eigenvalue Problem (EiP) [10]. As for the EiP, in any solution of the EiCP, the scalar λ is called an eigenvalue and x is an eigenvector associated to λ. The condition $x^tw = 0$ together with the nonnegative requirements on the variables x_i and w_i implies that $x_i = 0$ or $w_i = 0$ for each $i = 1, 2, \ldots, n$. These two variables are called complementary. It is known [16] that the EiCP always has a solution, as it can be reformulated

CONTACT Valentina Sessa sessa.valentina@gmail.com

as a Variational Inequality Problem on the simplex [8]

$$\Omega = \{x \in \mathbb{R}^n : e^t x = 1,\ x \geq 0\}. \tag{4}$$

The existence of a solution to the EiCP is also guaranteed under the weaker hypothesis that B is Strictly Copositive (SC), that is, when $x^t Bx > 0$ for all $0 \neq x \geq 0$ [16].

If the matrices A and B are both symmetric and B is PD, the EiCP is called symmetric and reduces to the problem of finding a Stationary Point (**SP**) of the so-called Rayleigh Quotient function on the simplex Ω [23,25], that is, an SP of the following Standard Quadratic Fractional Program

$$\textbf{SQFP}: \quad \text{Maximize} \quad \frac{x^t Ax}{x^t Bx} \tag{5}$$

$$\text{subject to} \quad e^t x = 1 \tag{6}$$

$$x \geq 0. \tag{7}$$

A number of techniques have been proposed to solve the EiCP and its extensions [1,2,14–17,19,21,25,29]. As expected, the symmetric EiCP is easier to solve. Projected-gradient type algorithms have been proposed in [4,15] for solving SQFP. The structure of the SQFP is fully exploited for the computation of the gradient and of the projection required by the algorithm in each iteration. Furthermore, it is possible to design an exact line search for finding the stepsize in each iteration that essentially requires the solution of binomial equation. Computational experience illustrates the efficiency of these algorithms for finding a solution for the symmetric EiCP, see [4,15].

From a computational point of view, the nonsymmetric EiCP is much harder to solve. A semi-smooth Newton algorithm has been introduced in [1] for the solution of this case. The algorithm can also be applied to the symmetric EiCP and exploits a formulation of EiCP as a system of nonlinear semi-smooth functions. Despite the efficiency of the algorithm for solving the EiCP in general, only local convergence is guaranteed and the algorithm may fail to solve the EiCP in many instances. An enumerative algorithm has also been recommended in [8] for the symmetric and nonsymmetric EiCP. The algorithm possesses global convergence to a solution of EiCP. This procedure exploits a nonlinear programming (NLP) formulation of EiCP consisting of minimizing a nonnegative objective function on a convex set defined by linear constraints [8]. Since NLP has an optimal solution with zero optimal value corresponding to a solution of EiCP, the enumerative algorithm computes stationary points in a structured way until finding one with a zero optimal value. In [9], this enumerative method is combined with the semi-smooth method mentioned above, in order to enhance its computational efficiency. Among the methods that have been recommended in the literature for the solution of the nonsymmetric EiCP, these semi-smooth Newton and hybrid algorithms are considered to be the most efficient and should be used as a benchmark for testing new algorithms such as those to be introduced in this paper.

In this paper, we discuss splitting methods for the numerical solution of the symmetric and nonsymmetric EiCP. This kind of methods has been applied for a long time for solving Linear Algebra problems, like systems of linear equations (see [27] for an early reference) or Linear Complementarity Problems (LCP; see [6]). On the other hand, to the best of

our knowledge, it is the first time that this approach is used for solving the symmetric and nonsymmetric EiCP. For the case of linear equations $Ax = b$ with given $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, we choose a non-singular matrix D, set $E = A-D$, and assuming that x^k is the kth iterate generated by the method, we solve the linear system $Dx = b'$ with $b' = b - Ex^k$, whose unique solution gives the next iterate x^{k+1}. The matrix D is chosen so that a linear system with the matrix D is much easier to solve than $Ax = b$, and the sequence $\{x^k\} \subset \mathbb{R}^n$ generated in this fashion is expected to converge to a solution of the original linear system. We suitably extend this approach to EiCP, and study four variants. In the first one, called A1, we take $A = C-D$, where D is a symmetric PD (SPD) matrix and the method consists of solving, at each step, an LCP with matrix D. When A and B are symmetric, the direction of the method at iteration k, namely $x^{k+1} - x^k$, turns out to be an ascent direction for the Rayleigh quotient, and, hence, it is possible to add a line search ensuring increase of the quotient, which improves the convergence properties of the algorithm. This version is called A2.

Another option applies to a PD matrix A and consists of using the splitting $A = C-D$, with D a symmetric positive semi-definite (SPSD) matrix and solving in each step an LCP with matrix $\lambda_k B + D$, where $\lambda_k = (x^k)^t Ax^k/(x^k)^t Bx^k$. This method, which we call B1, has the disadvantage that the matrix of each LCP varies along the iterative process, but it turns out to be numerically more efficient for many instances where A is (symmetric or nonsymmetric) positive definite at least when $D = 0$. In the case in which A and B are symmetric, it is also possible to add a line search, ensuring increase of the Rayleigh quotient along the iterative process, improving again the convergence properties of the algorithm. This variant is called B2 in the sequel.

For all these methods, we prove, under mild assumptions on the EiCP, that if the generated sequence $\{x^k\}$ is bounded then all its cluster points are solutions of the EiCP. Under more demanding assumptions on the problem, it is possible to prove that if the sequence is bounded then it has a unique cluster point. Finding conditions on A, B and D ensuring that the generated sequence is indeed bounded is left as an open problem. The choice of the splitting matrix D ($C = A+D$) is a very important issue and is difficult to be done in practice. In this paper, we report some experiments with very simple splittings for the algorithms Ai and Bi, $i = 1,2$. For the methods Ai, if A is a negative definite (ND) matrix, we use $D = -1/2(A + A^t)$. Hence, $D = -A$ if A is a symmetric ND (SND) matrix. If A is not ND, the algorithms are applied to a shifted EiCP($A + \mu B$, B) with $\mu < 0$ so that the matrix $A + \mu B$ is ND. This shifted EiCP turns out to be equivalent to EiCP(A, B). For the algorithms Bi, $i = 1,2$, we propose to use $D = 0$. Other choices for the matrix D have been considered and tested by us, but they seem not to be better than the ones mentioned above for general matrices A and B. However, we believe that other choices for D may be worthwhile for solving large-scale and structured EiCPs that may arise in applications. A good feature of the choices of the matrix D is that the matrix of each LCP to be solved by the algorithms Ai and Bi is always PD. This property enables the use of the so-called Block Principal Pivoting (BPP) algorithm [13] for dealing with these LCPs. This algorithm has shown to be quite efficient for this task and particularly when matrix of the LCP is strictly diagonally dominant with positive diagonal elements (see also [26]).

Computational experiments presented in the paper show that Algorithms A1 and A2 perform quite well for solving EiCP(A, B) with a matrix A (symmetric and nonsymmetric)

negative definite and $D = -1/2(A + A^t)$. When A is a PD matrix, Algorithms B1 and B2 with $D = 0$ are not so robust as the methods Ai, $i = 1,2$ in general, but outperform these latter versions for some instances.

In order to have a better idea of the efficiency of the splitting algorithms for solving EiCP, we decided to compare them with the best state-of-the-art algorithms introduced in the literature. For the symmetric EiCP, we used in these experiments the spectral block active set (SBAS) algorithm [4] and the spectral-projected gradient (SPG) method [15]. Furthermore, for the nonsymmetric EiCP, a hybrid algorithm combining an enumerative algorithm with a semi-smooth Newton method is surely the most robust procedure to be employed. Note that the semi-smooth Newton method is solely used when it is able to compute a solution of EiCP. The numerical experiments showed that splitting algorithms are in general competitive with those alternative methods. Furthermore, splitting methods tend to be more efficient for large-scale EiCPs with a sparse matrix B and a sparse or dense matrix A.

The structure of the paper is as follows. In Section 2, we present and analyse Algorithm A1. Section 3 is concerned with Algorithm A2, while the algorithms Bi, $i = 1$, 2 are introduced and analysed in Section 4. Computational experiments are reported in Section 5. Finally, conclusions and a few hints for future research are presented in the last section of the paper.

2. A splitting method for EiCP(A,B)

Given $A, B \in \mathbb{R}^{n\times n}$, EiCP($A$,$B$) consists of finding $\lambda \in \mathbb{R}$ and $0 \neq x \in \mathbb{R}^n$ such that

$$\lambda Bx - Ax \geq 0, \tag{8}$$

$$x \geq 0, \tag{9}$$

$$x^t(\lambda Bx - Ax) = 0. \tag{10}$$

As is common in the literature of EiCP, we assume from now on that B is a PD matrix. It follows from its definition that in any solution of EiCP $(\bar{\lambda}, \bar{x})$, the value of the complementary eigenvalue $\bar{\lambda}$ is equal to the value of the Rayleigh Quotient at the eigenvector $\bar{x}$. Furthermore, this eigenvector $\bar{x}$ is the solution of the Linear Complementarity Problem (LCP) obtained from (8) by fixing λ equal to $\bar{\lambda}$. Since this LCP is very difficult to solve and an LCP with an SPD matrix can be solved efficiently, we design an iterative method requiring in each iteration an EiCP of this last class. In order to achieve this goal, we consider a splitting $A = C-D$, where D is an SPD matrix. Consider a general iteration k of the splitting algorithm to be introduced in this section and let be x^k the corresponding iterate. Then λ_k is computed as the value of the Rayleigh Quotient at x^k. Furthermore, the following LCP(q^k, D):

$$Dx + q^k \geq 0, \quad x \geq 0,$$
$$x^t(Dx + q^k) = 0,$$

is considered, where

$$q^k = (\lambda_k B - C)x^k.$$

Since D is an SPD matrix, then this LCP has a unique solution x^{k+1}, which can be computed efficiently by a direct method [5]. Now, either x^{k+1} equals x^k and (x^{k+1}, λ_k) is a solution of EiCP or a new iteration is performed with the new iterate x^{k+1}.

The formal steps of Algorithm A1 are presented below.

Algorithm A1

▹ **Initialization Step**

(i) Choose an SPD matrix $D \in \mathbb{R}^{n\times n}$ and define $C = A + D$.
(ii) Choose a positive tolerance ϵ and $0 \neq x^0 \in \mathbb{R}^n_+$.

▹ **Iterative Step**

(i) Given $x^k \in \mathbb{R}^n$, define

$$\lambda_k = \frac{(x^k)^t A x^k}{(x^k)^t B x^k}, \tag{11}$$

$$q^k = (\lambda_k B - C)x^k.$$

(ii) Take x^{k+1} as the unique solution of LCP(D, q^k).
(iii) Terminate when $\|x^{k+1} - x^k\| < \epsilon$.

Observe that, as an immediate consequence of Iterative Step, we have, for all $k \geq 1$:

$$\lambda_k B x^k - C x^k + D x^{k+1} \geq 0, \tag{12}$$

$$x^k \geq 0, \tag{13}$$

$$(x^{k+1})^t(\lambda_k B x^k - C x^k + D x^{k+1}) = 0, \tag{14}$$

$$(x^k)^t(\lambda_k B x^k - C x^k + D x^k) = 0, \tag{15}$$

where (12)–(14) are the explicit expression of the fact that x^{k+1} solves LCP(D, q^k) and (15) follows by rewriting (11).

Proposition 2.1: *Let* $\{(x^k, \lambda_k)\}$ *be the sequence generated by Algorithm* A1.

(i) $x^k \neq 0$ *for all* $k \geq 0$.
(ii) *Algorithm* A1 *is well defined.*

Proof: (i) We proceed by induction. $x^0 \neq 0$ by the Initialization Step. Assume by induction that $x^k \neq 0$. If $x^{k+1} = 0$, then $\lambda_k B x^k - C x^k \geq 0$ by (12). Then, by (13),

$$(x^k)^t(\lambda_k B x^k - C x^k) \geq 0. \tag{16}$$

Rewrite (15) as

$$0 = (x^k)^t(\lambda_k B x^k - C x^k) + (x^k)^t D x^k. \tag{17}$$

Since the first term in the right-hand side of (17) is nonnegative by (16), and the second one is strictly positive, because D is PD and $x^k \neq 0$ by the inductive hypothesis, (17) entails a contradiction. Hence $x^{k+1} \neq 0$, completing the inductive step.

(ii) By (i), given x^k and λ_k, x^{k+1} is well defined. It remains to check that λ_{k+1} is also well defined. This is a consequence of item (i), which implies that $x^{k+1} \neq 0$, and the positive definiteness of B, which ensures that the denominator of (11) does not vanish.

■

Define $\langle \cdot, \cdot \rangle_D$, $\|\cdot\|_D$ as $\langle x, y \rangle_D = x^t D y$, $\|x\|_D = \sqrt{\langle x, x \rangle_D} = \sqrt{x^t D x}$. Since D is SPD, $\langle \cdot, \cdot \rangle_D$ is an inner product and $\|\cdot\|_D$ is a norm.

Proposition 2.2: *Let* $\{x^k\}$ *be the sequence generated by Algorithm* A1. *Then, for all* $k \in \mathbb{N}$,

(i) $\langle x^{k+1}, x^k \rangle_D \geq \left\|x^k\right\|_D^2$,

(ii)

$$\left\|x^{k+1} - x^k\right\|_D^2 \leq \left\|x^{k+1}\right\|_D^2 - \left\|x^k\right\|_D^2, \tag{18}$$

(iii) $\{\left\|x^k\right\|_D\}$ *is strictly increasing.*

Proof: (i) By (12) and (13), $(x^k)^t(\lambda_k B x^k - C x^k + D x^{k+1}) \geq 0$. This implies that

$$\langle x^k, x^{k+1} \rangle_D = (x^k)^t D x^{k+1} \geq (x^k)^t (C x^k - \lambda_k B x^k) = (x^k)^t D x^k = \left\|x^k\right\|_D^2,$$

using (12) and (13) in the inequality, and (15) in the second equality.

(ii)

$$\begin{aligned} \left\|x^{k+1} - x^k\right\|_D^2 &= \left\|x^{k+1}\right\|_D^2 + \left\|x^k\right\|_D^2 - 2\langle x^{k+1}, x^k \rangle_D \\ &\leq \left\|x^{k+1}\right\|_D^2 + \left\|x^k\right\|_D^2 - 2\left\|x^k\right\|_D^2 \qquad (19) \\ &= \left\|x^{k+1}\right\|_D^2 - \left\|x^k\right\|_D^2, \qquad (20) \end{aligned}$$

using item (i) in the inequality.

(iii) Immediate from item (ii).

■

Proposition 2.3: *Let* $\{(x, \lambda_k)\}$ *be the sequence generated by Algorithm* A1. *Then for all* $k \in \mathbb{N}$,

(i) *The sequence* $\{\lambda_k\}$ *is bounded.*

(ii) *Assume that* $\{x^k\}$ *is bounded. Then,*

(a) $\lim_{k \to \infty} \left\|x^{k+1} - x^k\right\|_D = 0$.

(b) *If* $(\bar{x}, \bar{\lambda})$ *is a cluster point of* $\{(x^k, \lambda_k)\}$, *then the pair* $(\bar{x}, \bar{\lambda})$ *solves EiCP*(A, B).

(c) *The whole sequence* $\{\lambda_k\}$ *converges to some complementary eigenvalue* λ^* *of EiCP*(A, B).

Proof: (i) Define $\tilde{x}^k = \left\|x^k\right\|^{-1} x^k$, which is well defined by Proposition 2.1(i). Dividing the numerator and the denominator of (11) by $\left\|x^k\right\|$, we have

$$\lambda_k = \frac{(\tilde{x}^k)^t A \tilde{x}^k}{(\tilde{x}^k)^t B \tilde{x}^k}. \tag{21}$$

Since $\left\|\tilde{x}^k\right\| = 1$ for all k, both the numerator and the denominator of the right-hand side of (21) are bounded. Since B is PD and $\tilde{x}^k \neq 0$ for all k by Proposition 2.1(i), the denominator of (21) is bounded away from 0. The conclusion follows.

(ii-a) Since $\{x^k\}$ is bounded, we may take γ such that $\left\|x^k\right\|_D^2 \leq \gamma$ for all k. By applying Proposition 2.2(ii) and summing (18) with k between 0 and ℓ, we obtain

$$\sum_{k=0}^{\ell} \left\|x^{k+1} - x^k\right\|_D^2 \leq \left\|x^{\ell+1}\right\|_D^2 - \left\|x^0\right\|_D^2 \leq \left\|x^{\ell+1}\right\|_D^2 \leq \gamma \tag{22}$$

for all ℓ. Hence, the series $\sum_{k=0}^{\infty} \left\|x^{k+1} - x^k\right\|_D^2$ is summable, implying the result.

(ii-b) Consider a subsequence $\{(x^{j_k}, \lambda_{j_k})\}$ of $\{(x^k, \lambda_k)\}$ such that $\lim_{k\to\infty}(x^{j_k}, \lambda_{j_k}) = (\bar{x}, \bar{\lambda})$. By item (ii-a), $\lim_{k\to\infty} x^{j_k+1} = \bar{x}$. Now, we take limits as $k \to \infty$ along the chosen subsequence in (12), (13) and (15). Taking into account that $A = C - D$, we obtain respectively,

$$0 \leq \bar{\lambda} B\bar{x} - C\bar{x} + D\bar{x} = \bar{\lambda} B\bar{x} - A\bar{x}, \tag{23}$$

$$\bar{x} \geq 0, \tag{24}$$

$$0 = \bar{x}^t(\bar{\lambda} B\bar{x} - C\bar{x} + D\bar{x}) = \bar{x}^t(\bar{\lambda} B\bar{x} - A\bar{x}). \tag{25}$$

Comparing (8)–(10) to (23)–(25), we observe that in order to establish that $(\bar{x}, \bar{\lambda})$ solves EiCP(A,B) it only remains to prove that $\bar{x} \neq 0$. It follows from Proposition 2.2(iii) that $0 < \left\|x^0\right\|_D < \left\|x^k\right\|_D$ for all k, so that $0 < \|\bar{x}\|_D$ and hence $\bar{x} \neq 0$.

(ii-c) Define $W = \{x \in \mathbb{R}^n_+ : \|x\|_D \geq \left\|x^0\right\|_D\}$. By Proposition 2.2(iii), $\{\left\|x^k\right\|_D\}$ is non-decreasing, so that $\{x^k\} \subset W$ for all k. Observe that the Rayleigh Quotient function $\varphi : \mathbb{R}^n_+ \to \mathbb{R}$ defined as

$$\varphi(x) = \frac{x^t A x}{x^t B x} \tag{26}$$

is continuous on W because B is PD and $\left\|x^0\right\|_D > 0$. In view of item (ii-a), it follows from (11) and the continuity of φ on W that $\lim_{k\to\infty}(\lambda_{k+1} - \lambda_k) = 0$. Since $\{\lambda_k\}$ is bounded by item (i), we can apply Ostrowsky's Theorem [20] for establishing that the set of cluster points of $\{\lambda_k\}$ is compact and connected. On the other hand, by item (ii-b) such set of cluster points is contained in the set of complementary eigenvalues of EiCP(A,B), known to be finite. Since the only finite and connected subsets of $\mathbb{R}$ are the singletons, we conclude that the set of cluster points of $\{\lambda_k\}$ is a singleton $\{\lambda^*\}$,

so that $\{\lambda_k\}$ converges to λ^*, which is a complementary eigenvalue for EiCP(A,B) by item (ii-b).

■

Next we present sharper convergence results for the sequence $\{x^k\}$ generated by Algorithm A1, assumed to be bounded, under some additional conditions on $\lambda^* := \lim_{k\to\infty} \lambda_k$. We will say that a complementary eigenvalue λ of EICP(A, B) is *strict* if for any solution (λ, x) of EiCP(A,B) it holds that $x + (\lambda B - A)x > 0$ (i.e. there exists no index i for which both x_i and $[(\lambda B - A)x]_i$ vanish). For $x \in \mathbb{R}^n_+$, we define

$$I(x) := \{i : x_i > 0\}. \tag{27}$$

As it is usual in the complementary eigenvalue literature, we will use the complementary variables w. We define $w^k \in \mathbb{R}^n$ as $w^k = (\lambda_k B - A)x^k$. It is easy to check that (12) and (14) can be rewritten in terms of w^k as

$$D(x^{k+1} - x^k) + w^k \geq 0, \tag{28}$$

$$(x^{k+1})^t [D(x^{k+1} - x^k) + w^k] = 0. \tag{29}$$

Proposition 2.4: *Let $I(x)$ be the set defined by* (27). *If the sequence $\{x^k\}$ generated by Algorithm* A1 *is bounded, and $\lambda^* = \lim_{k\to\infty} \lambda_k$ is a strict complementary eigenvalue for EiCP(A, B), then $I(\bar{x}) = I(\bar{y})$ for any two cluster points $\bar{x}, \bar{y}$ of $\{x^k\}$.*

Proof: Assume that $\bar{x}, \bar{y}$ are two cluster points of $\{x^k\}$. By Proposition 2.3(ii-b), both $(\bar{x}, \lambda^*)$ and $(\bar{y}, \lambda^*)$ solve EiCP(A,B). Let $\bar{w}, \bar{v}$ be the corresponding complementary variables, i.e. $\bar{w} = (\lambda^* B - A)\bar{x}, \bar{v} = (\lambda^* B - A)\bar{y}$. Note that both $\bar{w}$ and $\bar{v}$ are cluster points of the sequence $\{w^k\}$. Furthermore, Proposition 2.3(1-a) and the definition of w^k imply that

$$\lim_{k\to\infty} (w^{k+1} - w^k) = 0. \tag{30}$$

By strict complementarity of λ^*, $I(\bar{x}) = \{i : \bar{w}_i = 0\}$, $I(\bar{y}) = \{i : \bar{v}_i = 0\}$. Suppose that $I(\bar{x}) \neq I(\bar{y})$. Then, there exists i such that $0 = \bar{v}_i < \bar{w}_i$ or $0 = \bar{w}_i < \bar{v}_i$. Without loss of generality, assume that the first case occurs. Now we consider subsequences $\{x^{j_k}\}$, $\{x^{\ell_k}\}$ of $\{x^k\}$ such that $\lim_{k\to\infty} x^{j_k} = \bar{x}$, $\lim_{k\to\infty} x^{\ell_k} = \bar{y}$. Redefining the subsequences if needed, we may assume, without loss of generality, that they are interlaced, i.e. that $j_k < \ell_k < j_{k+1}$ for all k. Fix any ϵ such that

$$0 < \epsilon < \bar{w}_i. \tag{31}$$

In view of Proposition 2.3(ii-a) and (30), we can choose $\hat{k}$ such that

$$\left| [D(x^{k+1} - x^k)]_i \right| \leq \epsilon, \tag{32}$$

$$\left| w_i^{k+1} - w_i^k \right| \leq \frac{\epsilon}{2}, \tag{33}$$

for all $k \geq \hat{k}$. Furthermore,

$$\left| w_i^{\ell+k} \right| = \left| w_i^{\ell_k} - \bar{v}_i \right| \leq \frac{\epsilon}{2} \tag{34}$$

for all k such that $\ell_k > \hat{k}$, using the fact that $\lim_{k\to\infty} w^{\ell_k} = \bar{v}$ and $\bar{v}_i = 0$. It follows from (31) and (32) that for $j_k \geq \hat{k}$, $[D(x^{j_k+1} - x^{j_k}]_i + w_i^{j_k} > 0$, so that, by (29) and (13), we

get $x_i^{j_k+1} = 0$. On the other hand, by (34) there exists a first index $p_k \geq j_k$ such that $w_i^{p_k} \leq \epsilon$, and it satisfies $p_k \leq \ell_k$. By (32) and the definition of p_k, we have $[D(x^{r+1} - x^r)]_i + w_i^r > 0$ for all r such that $j_k < r < p_k$, so that $x_i^{r+1} = 0$ by (28), (29) and (11). Since $j_k \leq p_k \leq \ell_k$ for all k, $\{p_k\}$ is an infinite sequence. We look now at the subsequences $\{x^{p_k}\}$, $\{w^{p_k}\}$, with p_k larger than $\hat{k}$. We have $x_i^{p_k} = 0$, $\epsilon/2 \leq w^{p_k} \leq \epsilon$, using (33) in the left inequality. Let $(\bar{z}, \bar{u})$ be a cluster point of $\{(x^{p_k}, w^{p_k})\}$. We conclude that $\bar{z}_i = 0$, $|\bar{u}_i| \leq \epsilon$. Since ϵ is arbitrary, we have shown that for all small enough $\epsilon > 0$, there exists a cluster point (z^ϵ, u^ϵ) of $\{(x^k, w^k)\}$ with $z_i^\epsilon = 0$, $u_i^\epsilon \in [0, \epsilon)$. Since the set of cluster points of $\{(x^k, w^k)\}$ is closed, again by Ostrowski's Theorem, we may take limits with $\epsilon \to 0$ and obtain a cluster point $(\hat{z}, \hat{u})$ of $\{(x^k, w^k)\}$, with $0 = \hat{z}_i = \hat{u}_i$, associated to the strictly complementary eigenvalue λ^*, which entails a contradiction. This contradiction comes from assuming that $I(\bar{x}) \neq I(\bar{y})$, and so $I(\bar{x}) = I(\bar{y})$ for any two cluster points $\bar{x}, \bar{y}$ of $\{x^k\}$. ■

Still under the assumption of boundedness of $\{x^k\}$, we present next an additional property of $\lambda^* = \lim_{k\to\infty} \lambda_k$, besides strict complementarity, which guarantees convergence of the whole sequence $\{x^k\}$.

For a given subset $I \subset \{1, 2, \ldots, n\}$, we denote as x_I the vector with components x_i ($i \in I$) and by A_I, B_I the principal minors of A,B respectively associated to the index set I, i.e. $A_I = \{A_{ij} : i, j \in I\}$, $B_I = \{B_{ij} : i, j \in I\}$. Assume now that (x, λ) solves EiCP(A,B), and take $I = I(x) = \{i : x_i > 0\}$. Note that x_I fully determines x, because $x_i = 0$ for all $i \notin I$. It is well known that in such a situation λ and x_I are a generalized eigenvalue and eigenvector respectively for the pair A_I, B_I, i.e. it holds that $\lambda B_I x_I - A_I = 0$, or equivalently, since B is non-singular and the same holds for all its principal minors, λ, x_I are a standard eigenvalue and eigenvector respectively of the matrix $B_I^{-1} A_I$.

Proposition 2.5: *Assume that the sequence* $\{x^k\}$ *generated by Algorithm* A1 *is bounded, and that* $\lambda^* = \lim_{k\to\infty} \lambda_k$ (*which exists by Proposition* 2.3(ii-c)), *is a strictly complementary eigenvalue for EiCP*(A, B). *Let* $I = I(\bar{x})$ *be the set defined in* (27) *for* $x = \bar{x}$ *a cluster point of* $\{x^k\}$. *If* λ^* *has multiplicity 1 as a root of the characteristic polynomial of* $B_I^{-1} A_I$, *then the whole sequence* $\{x^k\}$ *converges to a complementary eigenvector* x^* *of EiCP*(A, B) *associated to* λ^*, *and* x_I^* *is an eigenvector of* $B_I^{-1} A_I$ *with associated eigenvalue* λ^*.

Proof: By Proposition 2.4, the set I is uniquely determined, since it does not depend on the choice of a particular cluster point of $\{x^k\}$. Take any cluster point $\bar{x}$ of $\{x^k\}$, which exists by Proposition 2.3(ii-b). Since $\{\|x^k\|_D\}$ is non-decreasing by Proposition 2.2(ii), and bounded by assumption, it converges, say to σ. Note that $\|x^k\|_D = \|x_I^k\|_{D_I}$, so that $\|\bar{x}_I\|_{D_I} = \sigma$. In view of the comments above, $\bar{x}_I$ is an eigenvector of $B_I^{-1} A_I$ with associated eigenvalue λ^* such that $\|\bar{x}_I\|_{D_I} = \sigma$. The assumption on the multiplicity of λ^* as an eigenvalue of $B_I^{-1} A_I$ implies that the set of its associated eigenvectors is a halfline, and hence there exists a unique one with a given prescribed norm. We have proved that $\bar{x}_I$ is uniquely determined, i.e. that $\{x_I^k\}$ has a unique cluster point, say x_I^*. Since $I(x) = I$ for any cluster point x of $\{x^k\}$ by Proposition 2.4, it follows that $x_i = 0$ for any $i \notin I$ and any cluster point x of $\{x^k\}$, i.e. $\lim_{k\to\infty} x_i^k = 0$ for all $i \notin I$. Defining $x_i^* = 0$ for $i \notin I$, we conclude that $\lim_{k\to\infty} x^k = x^*$, and x^* is a complementary eigenvector of EiCP(A,B) with associated complementary eigenvalue λ^*, again by Proposition 2.3(ii-b). ■

3. A splitting method with line search for the symmetric case

In this section, we assume that A and B are symmetric matrices. Hence B is an SPD matrix. Consider $\varphi : \mathbb{R}^n \to \mathbb{R}$ as defined by (26). Observe that

$$\nabla\varphi(x) = \frac{2}{x^tBx}\left[A - \frac{x^tAx}{x^tBx}B\right]x = \frac{2}{x^tBx}\left[A - \varphi(x)B\right]x. \tag{35}$$

If $\{x^k\}$ is the sequence generated by Algorithm A1, then, in view of (11), we get $\lambda_k = \varphi(x^k)$ and then, using (35),

$$\nabla\varphi(x^k) = \frac{2}{(x^k)^tBx^k}\left[A - \lambda_k B\right]x^k. \tag{36}$$

Define $\sigma_k \in \mathbb{R}, u^k \in \mathbb{R}^n$ as $\sigma_k = 2/(x^k)^tBx^k$, $u^k = (\lambda_k B - A)x^k$. Note that $\sigma_k > 0$ and that (36) can be rewritten as

$$\nabla\varphi(x^k) = -\sigma_k u^k. \tag{37}$$

It turns out that $(x^{k+1} - x^k)^t\nabla\varphi(x^k) > 0$ for all k, which suggests the introduction of a variant of Algorithm A1, to be called Algorithm A2, which adds a line search in the direction $(x^{k+1} - x^k)$ so as to insure that φ increases along the algorithm, in which case $\{\lambda_k\}$ turns out to be also increasing, and hence convergent, without assuming boundedness of $\{x^k\}$. We introduce Algorithm A2 before establishing the above announced result.

Algorithm A2

▹ **Initialization Step**
Same as the Initialization Step of Algorithm A1 in Section 2.

▹ **Iterative Step**

(i) Given $x^k \in \mathbb{R}^n$, define

$$\lambda_k = \frac{(x^k)^tAx^k}{(x^k)^tBx^k}. \tag{38}$$

$$q^k = (\lambda_k B - C)x^k.$$

(ii) Define y^k as the unique solution of LCP(D, q^k),

$$d^k = y^k - x^k, \tag{39}$$

$$x^{k+1} = x^k + \alpha_k d^k, \tag{40}$$

with $\alpha_k \in (0, 1]$ obtained through a line search for maximizing φ, thus guaranteeing that $\varphi(x^k) < \varphi(x^{k+1})$.

(iii) Terminate when $\|x^{k+1} - x^k\| < \epsilon$.

Proposition 3.1: (i) *Algorithm* A2 *is well defined, and* $0 \neq x^k \geq 0$ *for all* k.
(ii) *If* $\{x^k\}$ *be the sequence defined by Algorithm* A2, *then the sequence* $\{\|x^k\|_D\}$ *is non-decreasing.*

Proof: (i) We proceed by induction. The result holds for $k=0$ by the Initialization Step. Assume now that x^k is well defined and nonnegative. We claim first that d^k is an ascent direction for φ, which is essential for performing the line search. Thus we must prove that

$$\nabla\varphi(x^k)^t d^k > 0 \tag{41}$$

for all k. Note that

$$\lambda_k Bx^k - Cx^k + Dy^k \geq 0, \tag{42}$$

$$y^k \geq 0, \tag{43}$$

$$(y^k)^t(\lambda_k Bx^k - Cx^k + Dy^k) = 0, \tag{44}$$

which hold by the same argument as given for (12)–(14) in the case of Algorithm A1, i.e. they are a consequence of (38) and the fact that y^k solves LCP(D, q^k). Since $x^k \geq 0$ by the inductive hypothesis, $y^k \geq 0$ by (43), and α_k belongs to $(0, 1]$, it follows from (39) and (40) that $x^{k+1} \geq 0$. We still need to prove that $x^{k+1} \neq 0$. Using the same argument as in the proof of Proposition 2.1(i) with y^k substituting for x^{k+1}, we conclude that $y^k \neq 0$. Since x^k is in the segment between x^k and y^k which are both nonnull and nonnegative, it follows that $x^{k+1} \neq 0$. It remains to be seen that α_k can be chosen so that $\varphi(x^k) < \varphi(x^{k+1})$, i.e. that d^k is an ascent direction for φ at x^k. We rewrite (42) and (44) as

$$u^k + Dd^k = (\lambda_k B - A)x^k + D(y^k - x^k) \geq 0, \tag{45}$$

$$\begin{aligned}(y^k)^t(u^k + Dd^k) &= (y^k)^t[(\lambda_k B - A)x^k + D(y^k - x^k)] \\ &= (y^k)^t(\lambda_k Bx^k - Cx^k + Dy^k) = 0,\end{aligned} \tag{46}$$

using the definition of u^k and the fact that $D = C - A$. In view of (45) and the fact that x^k is nonnegative by the inductive hypothesis, we obtain

$$(x^k)^t(u^k + Dd^k) \geq 0. \tag{47}$$

Subtracting (46) from (47), we get

$$-(d^k)^t(u^k + Dd^k) = (x^k - y^k)^t(u^k + Dd^k) \geq 0. \tag{48}$$

It follows from (48) that $(d^k)^t(u^k + Dd^k) \leq 0$. Then,

$$0 < (d^k)^t Dd^k \leq (-u^k)^t d^k = \sigma_k^{-1}[\nabla\varphi(x^k)^t d^k], \tag{49}$$

using (37) and the fact that D is PD in the leftmost inequality. Since $\sigma_k > 0$, (41) follows from (49), establishing the claim. Hence, we have proved that d^k is an ascent direction for φ at x^k. It follows that the line search in the Iterative Step will provide a value of $\alpha_k > 0$ so that $\varphi(x^{k+1}) = \varphi(x^k + \alpha_k d^k) > \varphi(x^k)$, and therefore Algorithm A2 is well defined.

(ii) Since the definition of y^k in Algorithm A2 coincides with the definition of x^{k+1} in Algorithm A1, Proposition 2.2(ii) holds for Algorithm A2 with y^k substituting for x^{k+1}, namely

$$\left\|y^k - x^k\right\|_D^2 \leq \left\|y^k\right\|_D^2 - \left\|x^k\right\|_D^2,$$

so that $\left\|x^k\right\|_D \leq \left\|y^k\right\|_D$. From (39)–(40) and the fact that $\alpha_k \in (0, 1]$, we obtain that x^{k+1} belongs to the segment between y^k and x^k, and then the conclusion follows from the convexity of $\|\cdot\|_D$.

■

Corollary 3.2: *The sequence* $\{\lambda_k\}$ *defined by Algorithm* A2 *is convergent.*

Proof: By (38), $\lambda_k = \varphi(x^k)$. By Proposition 3.1, $\{\lambda_k\}$ is increasing. On the other hand, $\{\lambda_k\}$ is bounded, with the same argument as used in the proof of Proposition 2.3(i) for Algorithm A1. The conclusion follows. ■

Next, we prove a convergence result for Algorithm A2 similar to the one established for Algorithm A1 in Proposition 2.3, with a rather different proofline.

Proposition 3.3: *Let* $\{x^k\}, \{\lambda_k\}$ *be the sequences generated by Algorithm* A2. *If the sequence* $\{x^k\}$ *is bounded and* $(\bar{x}, \bar{\lambda})$ *is a cluster point of* $\{(x^k, \lambda_k)\}$, *then the pair* $(\bar{x}, \bar{\lambda})$ *solves EiCP*(A, B) *and the sequence* $\{\lambda_k\}$ *converges to some complementary eigenvalue* $\bar{\lambda}$.

Proof: The proof consists of observing that Algorithm A2 is an instance of a projected descent direction method applied to the problem

$$\min -\varphi(x) \quad \text{s.t. } x \geq 0. \tag{50}$$

Such method generates a sequence $\{z^k\}$ of the form

$$z^{k+1} = P(z^k - \beta_k w^k), \tag{51}$$

where P is the orthogonal projection onto $\mathbb{R}^n_+$, u^k is a descent direction for $-\varphi$ at x^k and $\beta_k > 0$ is determined through a line search. Indeed, (39) and (40) imply that x^{k+1} is obtained through a line search in the direction d^k, which is a descent direction for $-\varphi$ by (41). On the other hand, by Proposition 3.1, $x^k \geq 0$ for all k, so that $x^{k+1} = P(x^{k+1}) = P(x^k - \alpha_k d^k)$. Hence, Algorithm A2 has the prescribed form for a projected descent direction method. The well-known convergence theory for the projected descent direction method establishes that if $\{z^k\}$ given by (51) is bounded, and

$$-\nabla\varphi(z^k)^t w^k \leq -\eta \left\|w^k\right\|^2 \tag{52}$$

for some $\eta > 0$, then all cluster points of $\{z^k\}$ are stationary points for problem (50). Since $\{x^k\}$ is assumed to be bounded, there exists $\rho > 0$ such that $\left\|x^k\right\| \leq \rho$ for all k. Let μ be

the smallest eigenvalue of D and ν the largest eigenvalue of B. In view of (49) and the fact that both B and D are both SPD, we have

$$-\nabla\varphi(x^k)^t d \leq -\sigma_k d^k D d^k \leq -\sigma_k \mu \left\| d^k \right\|^2 = -\frac{2}{(x^k)^t B x^k}\mu \left\| d^k \right\|^2 \leq -\frac{2}{\nu\rho^2}\mu \left\| d^k \right\|^2.$$

Hence, (52) holds with $z^k = x^k$, $w^k = d^k$ and $\eta = 2\mu/\nu\rho^2$. We conclude from the convergence theorem for the projected descent direction method that all cluster points of $\{x^k\}$ are stationary points for problem (50). By Proposition 3.1(ii), $\left\|x^k\right\|_D \geq \left\|x^0\right\|_D$, so that $\|\bar{x}\|_D \geq \left\|x^0\right\|_D > 0$, using the fact that $x^0 \neq 0$. Hence, $\bar{x} \neq 0$. It is well known that if $\bar{x}$ is a nonnull stationary point for problem (50), then the pair $(\bar{x}, \varphi(\bar{x}))$ is a solution of EiCP(A,B). Note that $\{\lambda^k\}$ converges by Corollary 3.2, to some real number, say $\bar{\lambda}$, and in view of (38) and the continuity of φ, we have that $\bar{\lambda} = \varphi(\bar{x})$, completing the proof. ■

Remark 3.4: If A is an ND matrix, then $D = -1/2(A + A^t)$ is an SPD matrix and can be used in the splitting of A. So, $D = -A$ if A is a symmetric ND matrix. If A is not an ND matrix there exists a $\mu < 0$ such that $A + \mu B$ is ND. In this case, Algorithms Ai, $i = 1,2$, should find a solution for EiCP (A,B) by solving the shifted EiCP ($A + \mu B$, B). These two problems are equivalent, as (x, λ) is a solution of EiCP(A, B) if and only if $(x, \lambda + \mu)$ is a solution of EiCP($A + \mu B$, B).

4. Splitting methods for the case of positive definite *A*

In this section, we propose two splitting methods, related to Algorithms A1 and A2, which work under the assumptions that A is (symmetric or nonsymmetric) PD and B is SPD. As in the ND case, if A is not PD, there exists a $\mu > 0$ such that $A + \mu B$ is PD and a solution of EiCP can be found by solving its equivalent shifted EiCP($A + \mu B$, B). So, the first hypothesis on A is not restrictive at least in theory. However, both the new algorithms of this section require B to be symmetric while the symmetry of B is not necessary for the Algorithm A1.

4.1. Algorithm B1

We define next the first method, called Algorithm B1. We assume that the matrix D of the splitting $A = C - D$ is symmetric positive semi-definite (SPSD).

Comparing Algorithms A1 and B1, we note that the only difference is that in the Linear Complementarity Problem the matrix $\lambda_k B$ has been moved from the right-hand side q^k (which became r^k in Algorithm B1) to the matrix argument, which was D in Algorithms A1 and B2 and became $E_k = \lambda_k B + D$ in Algorithm B1. Since A is PD and B is SPD, $\lambda_k > 0$ for all k. Moreover, since D is SPSD, then E_k is an SPD matrix for all k. When looking at the more explicit relations describing the Iterative Step, namely (12)–(15), this difference translates into substituting x^{k+1} for x^k in two instances. Indeed, for Algorithm B1 we have, instead of (12)–(15),

$$\lambda_k B x^{k+1} - C x^k + D x^{k+1} \geq 0, \tag{54}$$

$$x^k \geq 0, \tag{55}$$

Algorithm B1

▹ **Initialization Step**

(i) Choose an SPSD matrix D and define $C = A + D$.

(ii) Choose a positive tolerance ϵ and $0 \neq x^0 \in \mathbb{R}^n_+$.

▹ **Iterative Step**

(i) Given $x^k \in \mathbb{R}^n$, define

$$\lambda_k = \frac{(x^k)^t A x^k}{(x^k)^t B x^k}, \tag{53}$$

$$r^k = -Cx^k.$$

(ii) Let x^{k+1} as the unique solution of LCP$(\lambda_k B + D, r^k)$.

(iii) Terminate when $\|x^{k+1} - x^k\| < \epsilon$.

$$(x^{k+1})^t(\lambda_k B x^{k+1} - Cx^k + Dx^{k+1}) = 0, \tag{56}$$

$$(x^k)^t(\lambda_k B x^k - Cx^k + Dx^k) = 0, \tag{57}$$

and the only changes with respect to Algorithm A1 occur in the first terms of the right-hand side of (54) and (56). Note that as for the Algorithm A1, LCP (54)–(57) has a unique solution for each k.

4.2. Algorithm B2

Next we define Algorithm B2, similar to Algorithm A2, which demands that A and B are SPD matrices.

Algorithm B2

▹ **Initialization Step**

Same as the Initialization Step of Algorithm B1 in Section 4.1.

▹ **Iterative Step**

(i) Given $x^k \in \mathbb{R}^n$, define

$$\lambda_k = \frac{(x^k)^t A x^k}{(x^k)^t B x^k}, \tag{58}$$

$$r^k = -Cx^k.$$

(ii) Define y^k as the unique solution of LCP$(\lambda_k B + D, r^k)$,

$$d^k = y^k - x^k, \tag{59}$$

$$x^{k+1} = x^k + \alpha_k d^k, \tag{60}$$

with $\alpha_k \in (0, 1]$ obtained through a line search for maximizing φ, thus guaranteeing that $\varphi(x^k) < \varphi(x^{k+1})$.

(iii) Terminate when $\|x^{k+1} - x^k\| < \epsilon$.

Note that Algorithm B2 consists of adding to Algorithm B1 a line search between the current iterate and the solution of the Linear Complementary Problem, precisely mirroring the line search added to Algorithm A1 in order to obtain Algorithm A2.

We have obtained so far quite limited convergence results for Algorithm B1, while our results for Algorithm B2 are virtually equivalent to those established in Proposition 3.3 for Algorithm A2. We start with two elementary results on Algorithm B1.

Proposition 4.1: *Let* $\{(x^k, \lambda_k)\}$ *be the sequence generated by Algorithm* B1.

(i) $x^k \neq 0$ *for all* $k \geq 0$.
(ii) $\lambda_k > 0$ *for all* $k \geq 0$.
(iii) *Algorithm* B1 *is well defined.*

Proof: Similar to the proof of Proposition 2.1. We proceed by induction. Assuming inductively that $x^k \neq 0, \lambda_k > 0$, we must prove that x^{k+1} is well defined and nonnull, and that λ_{k+1} is also well defined and positive. Since in this section, both A and B are assumed to be PD, it follows from (58) that $\lambda_k > 0$. As D is SPSD, then $\lambda_k B + D$ is PD. It follows that LCP$(\lambda_k B + D, r^k)$ has a unique solution, i.e. x^{k+1} is well defined. We claim that $x^{k+1} \neq 0$. Otherwise, we get from (54)–(55) that $-(x^k)Cx^k \geq 0$, and then from (57),

$$0 = (x^k)^t(\lambda_k B + D)x^k - (x^k)^t Cx^k \geq (x^k)^t(\lambda_k B + D)x^k. \tag{61}$$

Since $\lambda_k B + D$ is PD, (57) contradicts the inductive hypothesis that $x^k \neq 0$. We have proved that $x^{k+1} \neq 0$. Then, it follows from (58) that λ_{k+1} is well defined and positive. ■

Proposition 4.2: *Let* $\{(x^k, \lambda_k)\}$ *be the sequence generated by Algorithm* B1. *If* $\{x^k\}$ *converges to some* $x^* \neq 0$, *then* $\{\lambda_k\}$ *converges to* $\lambda^* = \varphi(x^*)$, *with* φ *as in* (26), *and* (x^*, λ^*) *solves EiCP*(A, B).

Proof: It suffices to invoke the continuity of φ and take limits with $k \to \infty$ in (54)–(56), obtaining that the pair (x^*, λ^*) satisfies (8)–(10). ■

The remainder of the analysis holds for Algorithm B2 but not for B1. We will point out later on the critical point which precludes the extension of the following results to Algorithm B1. Define $E_k = \lambda_k D + B$, $\langle u, v\rangle_{E_k} = u^t E_k v$, $\|u\|_{E_k} = \sqrt{\langle u, u\rangle_{E_k}}$. This notation makes sense because, under the hypotheses of this subsection, B is SPD, and D is SPSD, so that $\lambda_k B + D$ is also SPD. We start by showing the Algorithm B2 is also well defined.

Proposition 4.3: *Algorithm* B2 *is well defined, and* $0 \neq x^k \geq 0$ *for all k.*

Proof: Again, we proceed by induction. Using the same argument as in the proof of Proposition 4.2 with y^k instead of x^{k+1}, we obtain that y^k is well defined, nonnegative and nonnull. Since $0 \neq x^k \geq 0$ by inductive assumption, and x^{k+1} belongs to the segment between x^k and y^k, we conclude that x^{k+1} is also well defined, nonnegative and nonnull, so that the only remaining task consists of proving that d^k is an ascent direction for φ at x^k, i.e. that $\nabla\varphi(x^k)^t d^k > 0$ for all k. Recall that, in view of (37), $\nabla\varphi(x^k) = \sigma_k[A - \lambda_k B]x^k$, with $\sigma_k > 0$.

Now, it is clear from the definition of Algorithm B2 that (54)–(57) hold with y^k substituting for x^{k+1}. Since $x^k \geq 0$ for all k as established above, multiplying x^k by (54) and subtracting (57) with y^k instead of x^{k+1}, we get

$$\begin{aligned}
0 \leq (x^k - y^k)^t \left[\lambda_k B y^k - C x^k + D y^k\right] &= (x^k - y^k)^t \left[\lambda_k B y^k - C x^k + D x^k + D(y^k - x^k)\right] \\
&= (x^k - y^k)^t \left[\lambda_k B y^k - A x^k + D(y^k - x^k)\right] \\
&= (x^k - y^k)^t \left[\lambda_k B x^k - A x^k + (\lambda_k B + D)(y^k - x^k)\right] \\
&= (x^k - y^k)^t \left[\lambda_k B x^k - A x^k\right] - \left\|y^k - x^k\right\|_{E_k}^2 = \frac{1}{\sigma_k}(d^k)^t \nabla\varphi(x^k) - \left\|d^k\right\|_{E_k}^2,
\end{aligned}$$

so that

$$0 < \sigma_k \left\|d^k\right\|_{E_k}^2 \leq \nabla\varphi(x^k)^t d^k,$$

and d^k is an ascent direction for φ at x^k. Hence, performing a line search starting with $\alpha = 1$, we will find a value of α_k such that $x^{k+1} = x^k + \alpha_k d^k$ satisfies $\varphi(x^k) < \varphi(x^{k+1})$, completing the induction step and the proof. ■

We continue the convergence analysis of Algorithm B2 with some intermediate results.

Proposition 4.4: *Let* $\{(x^k, \lambda_k)\}$ *be the sequence generated by Algorithm* B2. *Then*

(i) $\langle x, y^k\rangle_{E_k} \geq \left\|x^k\right\|_{E_k}^2$ *for all* k.
(ii) $\left\|x^k\right\|_{E_k}^2 + \left\|y^k - x^k\right\|_{E^k}^2 \leq \left\|y^k\right\|_{E_k}^2$ *for all* k.
(iii) $\left\|x^k\right\|_{E_k} \leq \left\|x^{k+1}\right\|_{E_k}$ *for all* k.
(iv) *The sequence* λ_k *is non-decreasing and convergent.*

Proof: (i) Since $x^k \geq 0$ for all k, multiplying x^k by (54) we get $0 \leq (x^k)^t(\lambda_k B y^k - C x^k + D y^k) = (x^k)^t[(\lambda_k B + D)y^k - C x^k]$, implying that

$$\langle y^k, x^k\rangle_{E_k} = (x^k)^t E_k y^k \geq (x^k)^t C x^k = (x^k)^t(\lambda_k B + D)x^k = \left\|x^k\right\|_{E_k}^2,$$

using (57) in the second equality.

(ii) Follows easily from item (ii).

(iii) By (ii), $\left\|x^k\right\|_{E_K} \leq \left\|y^k\right\|_{E_k}$ for all k. Since x^{k+1} belongs to the segment between x^k and y^k, the result follows from the convexity of $\|\cdot\|_{E_k}$.

(iv) Since $\lambda_k = \varphi(x^k)$, we conclude from the Iterative Step of Algorithm B2 (in particular, from the use of the line search in the computation of x^{k+1}), that λ_k is non-decreasing. Since $\{\lambda_k\}$ is bounded above, with the same argument as in the proof of Proposition 2.3(i), we obtain that the sequence is convergent. ■

Let $\lambda^* = \lim_{k\to\infty} \lambda_k$, which exists by Proposition 4.4(iii), and $E^* = \lambda^* B + D$.

Proposition 4.5: *Let* $\{(x^k, \lambda_k)\}$ *be the sequence generated by Algorithm* B2. *Then* $0 < \|x^0\|_{E_0} \leq \|x^k\|_{E^*}$ *for all k, and* $\{x^k\}$ *is bounded away from 0.*

Proof: Note first that, since $0 < \lambda_k \leq \lambda_{k+1} \leq \lambda^*$, then it follows from the definition of E_k that $\|x\|_{E_k} \leq \|x\|_{E_{k+1}} \leq \|x\|_{E^*}$ for all $x \in \mathbb{R}^n$ and all k. Now, invoking Proposition 4.4(iii),

$$\|x^0\|_{E_0} \leq \|x^1\|_{E_0} \leq \|x^1\|_{E_1} \leq \cdots \|x^{k-1}\|_{E_{k-1}} \leq \|x^k\|_{E_{k-1}} \leq \|x^k\|_{E_k} \leq \|x^k\|_{E^*},$$

and the conclusion follows. ■

We comment now on the obstacles in the way of establishing better convergence results for Algorithm B1, under the assumption of boundedness of $\{x^k\}$. The essential ingredient for proving that cluster points of the sequence $\{x^k\}$ generated by Algorithm B1 are solutions of EiCP(A,B) would be a proof of the fact that the difference of consecutive iterates goes to zero, in which case the limit of a convergent subsequence $\{x^j_k\}$ would also be the limit of $\{x^{j_k+1}\}$ and the optimality would result from taking limits in (54), (55) and (57) along the subsequence. In the case of Algorithm A1, the fact that the difference between consecutive iterates eventually vanishes follows from (22), which comes from summing (18). For Algorithm B1, we have, instead of (18), $\|x^{k+1} - x^k\|^2_{E_k} \leq \|x^{k+1}\|^2_{E_k} - \|x^k\|^2_{E_k}$, which can be proved as in Proposition 4.4(ii). The difficulty lies in the fact that when summing this inequality, say with k between 1 and ℓ, the sum in the right-hand side is not telescopic any more, and hence we do not have cancellations. This is due to the fact that the norm changes with k in each term of the summation, because the same happens with the matrix in each LCP subproblem: for Algorithm A1 it is always the same matrix, namely D, while for Algorithm B1 we use E_k in iteration k. Nevertheless, it follows easily from summing the inequality above that

$$\sum_{k=1}^{\ell} \|x^{k+1} - x^k\|^2_D \leq \hat{\gamma} + \sum_{k=1}^{\ell} (\lambda_{k-1} - \lambda_k) \|x^k\|^2_B, \tag{62}$$

where $\hat{\gamma}$ is an upper bound for $\|x^k\|^2_{E_k}$, which exists because $\{\lambda_k\}$ is bounded and $\{x^k\}$ is assumed to be bounded. In fact, if the sequence $\{\lambda_k\}$ generated by Algorithm B1 is either non-decreasing or non-increasing, it is easy to get a constant upper bound for the left-hand side of (62), entailing that the difference between consecutive iterates of $\{x^k\}$ eventually vanishes, from which the optimality of the cluster points of the sequence is an easy consequence. Indeed, if $\{\lambda_k\}$ is non-decreasing, then the summation in the right-hand side of (62) is non-positive and the left-hand side is bounded by $\hat{\gamma}$; if $\{\lambda_k\}$ is non-increasing, then the left-hand side is bounded by $\hat{\gamma} + \lambda_0 \bar{\gamma}$, where $\bar{\gamma}$ is an upper bound for $\|x^k\|^2_B$. Since it seems very unlikely that the sequence $\{\lambda_k\}$ be non-decreasing (unless we force it to be so, e.g. through a line search, as done in Algorithm B2), and even less likely that it be non-increasing (because $x^{k+1} - x^k$ is an ascent direction for φ), we refrained from pursuing this proofline in detail.

We establish next the main convergence result for Algorithm B2.

Proposition 4.6: *Let* $\{(x^k, \lambda_k)\}$ *be the sequence generated by Algorithm* B2. *If the sequence* $\{x^k\}$ *is bounded and* $(\bar{x}, \bar{\lambda})$ *is a cluster point of* $\{(x^k, \lambda_k)\}$, *then the pair* $(\bar{x}, \bar{\lambda})$ *solves EiCP*(A, B) *and the sequence* $\{\lambda_k\}$ *converges to some complementary eigenvalue* $\bar{\lambda}$.

***Proof*:** The proof is similar to the one of Proposition 3.3 for Algorithm A2. It is easy to conclude from the definition of Algorithm B2 and Proposition 4.3 that it is an instance of the descent direction method applied to minimizing φ on the nonnegative orthant. Again, if we prove that there exist some $\eta > 0$ such that $-\nabla\varphi(x^k)^t d^k \leq -\eta \left\| d^k \right\|^2$ for all k, then all cluster points of $\{z^k\}$ are stationary point for such optimization problem, and the value of η given in the proof of Proposition 3.3, namely $\eta = 2\mu/\nu\rho^2$, does the job, also in this case. By Proposition 4.5, $\{x^k\}$ is bounded away from 0, and hence all its cluster points are nonnull. Again, if $\bar{x}$ is a nonnull stationary point for the problem of maximizing φ over the nonnegative orthant, then the pair $(\bar{x}, \varphi(\bar{x}))$ is a solution of EiCP(*A*,*B*). By Proposition 4.4(iv), $\bar{\lambda}$ is the limit of the sequence λ_k. Since $\lambda(k) = \varphi(x^k)$, we conclude that $\bar{\lambda} = \varphi(\bar{x})$, completing the proof. ■

Remark 4.7: In practice, D should be chosen as an SPSD matrix such that $E_k = \lambda_k B + D$ is PD for each k. Since A is (symmetric or nonsymmetric) PD and B is SPD, then $D = 0$ is an obvious choice.

5. Numerical experiments

In this section, we describe how we set up the numerical experiments, that is how we chose the matrices A and B for each set of problems and we present important details for the implementation of the proposed algorithms. Finally, we discuss their numerical performance for computing complementary eigenvalues. The splitting methods were implemented in MATLAB environment [18] (version 8.6, R2015b) and performed on an Intel Core i7 clocked at 2.40 GHz.

5.1. Test problems

As discussed before, an EiCP can always be reduced to an equivalent EiCP with a PD or ND matrix A by shifting. We have taken into consideration this property by generating only test problems where A belongs to one of these two classes of matrices. Furthermore, for all the algorithms to be applied we also require the matrix B to be SPD. For the set of Test Problems 1 and 2, we set A as a nonsymmetric ND matrix of the form

$$A = G + \mu I, \tag{63}$$

with G being a randomly generated matrix with elements uniformly distributed in the interval $[1, 10]$ and $\mu < 0$ chosen such that A is ND. The matrix B was taken as the identity matrix in Test Problems 1, while in Test Problems 2, B is a symmetric strictly diagonally dominant matrix with positive diagonal elements of the following form:

$$b_{i,i} = 10, \quad i = 1, \ldots, n, \tag{64a}$$

$$b_{i,j} = -1, \quad j = i+1, \ldots, i+4, \quad i = 1, \ldots, n, \tag{64b}$$

$$b_{i,j} = -1, \quad j = i-1, \ldots, i-4, \quad i = 1, \ldots, n. \tag{64c}$$

Hence, B is an SPD matrix [5]. For the above test problems, we denote each instance by RAND(n), where n represents the order of the matrices A and B (we considered $n = 10, 20, 30, 40, 50, 100, 250, 500, 750, 1000$).

Test Problems 3 consider B as the identity matrix and $A = -H$, where H is an SPD matrix from the Harwell–Boeing collection [11]. Then A is SND. Each problem is denoted by the name used in the collection and the order of each matrix A (and also B) is included in brackets after the corresponding notation.

In Test Problems 4 and 5, the matrix A is a nonsymmetric PD matrix of the form (63) with $\mu > 0$. As before, these two sets of test problems differ on the matrix B, which is the identity in Test Problems 4 and has the form (64) in the second case. Finally, in Test Problems 6, we considered B as the identity matrix and $A = H$, where H is an SPD matrix from the Harwell–Boeing collection [11].

Note that for all the instances of Test Problems 1, 2, 4 and 5, the matrix A is dense, as all the elements are nonzero. The 1-norm condition number estimate of A and B is indicated in the following tables in the columns titled $cond_A$ and $cond_B$, respectively. The matrices of Test Problems 3 and 6 are sparse, and the number of nonzero elements of these matrices is given in the column titled Nnzeros.

5.2. Implementation details

In each iteration of the splitting algorithms presented in this paper, an LCP(M,q) is solved, with $M = D$ for Algorithms A1 and A2 and $M = \lambda_k B + D$ for Algorithms B1 and B2, where $\lambda_k > 0$ is the Rayleigh Quotient estimation of a complementary eigenvalue computed at iteration k. It is well known that if M is PD, then LCP(M,q) has a unique solution for each vector q [5]. Furthermore, for this class of matrices LCP can be solved very efficiently by the so-called Block Principal Pivoting (BPP) algorithm [13]. Our splittings suggested in Remark 3.4 for Algorithms A1 and A2 and in Remark 4.7 for Algorithms B1 and B2 lead to an LCP with a PD matrix in each iteration. Furthermore, for Algorithms A2 and B2 we use the exact line search described in [4].

The initial point for the splitting algorithms is another important issue. As in [4], in our experiments, we use a canonical vector e^s, which is chosen by a preprocessing technique based on the following property: $(\lambda, x) = (a_{ii}/b_{ii}, e^i)$ is a solution of EiCP if and only if $r_i = \min\{a_{ii}b_{ji} - a_{ji}b_{ii} : j = 1, \ldots, n\} \geq 0$. The preprocessing technique investigates whether the above property is true for some $i = 1, \ldots, n$. In the positive case, a solution of EiCP is at hand. Otherwise, $r_i < 0$, $\forall i = 1, \ldots, n$, and the canonical vector e^s is chosen so that $s = argmax\{r_i : i = 1, \ldots, n\}$.

In order to analyse the efficiency of the splitting algorithms, we solved nonsymmetric EiCPs presented in Section 5.1, by using the semi-smooth Newton algorithm presented in [1] and the hybrid method proposed in [9]. A semi-smooth Newton algorithm has been introduced in [1] and solves EiCP by exploiting its formulation as a system of semi-smooth equations. The algorithm employs generalized Jacobians in each iteration and possesses fast local convergence under mild hypotheses. However, global convergence to a solution of EiCP cannot be guaranteed and the algorithm may fail for some instances. The hybrid algorithm possesses global convergence to a solution of EiCP and is known to be

quite efficient for the solution of small- and medium-scale EiCPs. The algorithm combines the semi-smooth Newton algorithm with a global convergent enumerative method that is designed for finding a global minimum with a zero optimal value of an NLP formulation of the EiCP. This hybrid algorithm may start with the semi-smooth Newton method if we believe that a good initial point is at hand. In general, the hybrid method starts by applying the enumerative method. During this phase, when the current feasible solution of NLP is sufficiently close to a possible global minimum of NLP, i.e. when the value of the objective function of NLP is sufficiently small, then the algorithm switches to the semi-smooth Newton algorithm by using as initial point the current solution provided by the enumerative method. Now, either the semi-smooth Newton algorithm is able to find a solution of EiCP or the enumerative algorithm continues at the current node. The whole procedure is repeated and is able to find a solution of EiCP by one of the semi-smooth or enumerative methods. The experiments of solving Test Problems 4 and 5 by these methods are shown in Section 5.5.

The EiCPs solved in Test Problems 3 and 6 are symmetric, that is, both A and B are symmetric matrices. So, these EiCPs can be solved by computing a stationary point of SQFP (5)–(7). The spectral projected-gradient (SPG) and the spectral block active set algorithm (SBAS) discussed in [4,15], respectively, are particularly recommended to deal with the symmetric EiCP. The results in [4] are used as a comparison with the algorithms presented in this paper.

All the algorithms are implemented in MATLAB and the IPOPT (Interior Point OPTimizer) solver [28] has been used to find a (local) solution to the NLP problems at each node of the enumerative method.

5.3. Performance for an ND matrix A

In this section, we report the performance of Algorithms A1 and A2 for solving test problems with $A \in$ ND. Algorithm B1 can be applied to Test Problems 1, 2 and 3 and Algorithm B2 to Test Problems 3 by solving the shifted EiCP($A + \mu B, B$) with $\mu > 0$ such that $A + \mu B$ is a PD matrix. However, the numerical results of our experiments showed that both the algorithms Bi perform much worse for solving the shifted EiCP than the algorithms Ai for solving the original EiCP. So, we do not report the results of these experiments. In the following tables, we show the results obtained by solving the test problems with choices for the matrix D and the initial point discussed in Section 5.2. In these tables, we include the value of the computed complementary eigenvalue, the number of iterations required by the selected algorithm, and CPU time in seconds. The columns titled as 'w_{BPP}', 'b_{BPP}', and 'av_{BPP}' indicate the number of iterations for BPP algorithm for the worst and the best performance, and the average number of iterations, respectively. Furthermore, the column titled 'comp' shows the value of $x^t w$ at the solution, while the column titled 'dualfeas' reports the value of $\min\{w_i : i = 1, \ldots, n\}$. Note that these two values give an idea of the accuracy of the solution computed by the algorithms. In fact, the smaller these positive values are the more accurate the computed solutions are.

The value of tolerance to terminate the algorithms is 10^{-6} for all test problems and the maximum number of iterations is set as 300 for Test Problems 1 and 2 and as 5000 for Test Problems 3.

Table 1. Performance of Algorithm A1 for solving Test Problems 1 and 2.

Problem	$cond_A$	$cond_B$	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
						$B = I$				
RAND(10)	1.22e+01	1	−10.630146	12	1.10161e−01	3	2	2.91	−4.26707e−06	−2.58335e−06
RAND(20)	2.83e+01	1	−10.658875	11	1.22833e−02	3	2	2.90	5.54662e−06	−9.15831e−07
RAND(30)	4.06e+01	1	−10.846811	10	1.11439e−02	3	3	3.00	1.15138e−05	−1.93478e−06
RAND(40)	5.28e+01	1	−10.365389	9	8.23782e−03	3	2	2.88	3.28932e−05	−4.96388e−06
RAND(50)	6.43e+01	1	−10.691757	9	9.55812e−03	3	3	3.00	4.52737e−05	−3.06775e−06
RAND(100)	1.25e+02	1	−10.577362	9	1.52249e−02	3	3	3.00	2.44659e−06	−4.46459e−07
RAND(250)	2.90e+02	1	−10.612081	8	4.79972e−02	4	3	3.71	2.63120e−04	−1.50714e−06
RAND(500)	5.68e+02	1	−10.631220	9	2.15858e−01	4	3	3.75	−1.31414e−04	−1.00978e−06
RAND(750)	8.35e+02	1	−10.622299	9	5.52857e−01	4	3	3.88	−2.40220e−04	−1.70783e−06
RAND(1000)	1.10e+03	1	−10.607556	9	9.96421e−01	4	3	3.75	−3.19461e−04	−1.70220e−06
						B defined in (64)				
RAND(10)	1.22e+01	5.58e+00	−2.594865	26	1.23734e−01	5	3	3.12	2.70677e−06	−9.00452e−06
RAND(20)	2.83e+01	7.96e+00	−3.338590	18	1.85861e−02	5	3	3.18	−3.90017e−06	−1.29536e−05
RAND(30)	4.06e+01	8.70e+00	−3.990632	16	1.68964e−02	6	3	3.27	9.21808e−06	−1.24422e−05
RAND(40)	5.28e+01	8.91e+00	−4.123274	15	1.68537e−02	4	3	3.14	8.11490e−06	−9.07478e−06
RAND(50)	6.43e+01	8.97e+00	−4.423012	14	1.55113e−02	3	3	3.00	1.35331e−05	−2.09608e−05
RAND(100)	1.25e+02	8.99e+00	−4.796597	11	1.83945e−02	5	3	3.20	9.55654e−06	−4.69789e−05
RAND(250)	2.90e+02	8.99e+00	−5.104439	10	5.26103e−02	4	3	3.78	−3.31713e−06	−2.32356e−05
RAND(500)	5.68e+02	8.99e+00	−5.211826	9	1.79398e−01	4	3	3.75	2.24735e−04	−1.21403e−05
RAND(750)	8.35e+02	8.99e+00	−5.241951	9	4.74475e−01	4	3	3.75	2.99049e−04	−1.94029e−05
RAND(1000)	1.10e+03	8.99e+00	−5.251345	9	9.32925e−01	4	3	3.75	2.84014e−04	−4.14814e−06

In Test Problems 1 and 2, the matrix A is ND, but not symmetric and Algorithm A2 cannot be used for solving these test problems. As stated above, we chose $D = -1/2(A + A^t)$. So, the resulting EiCP(A, B) were solved by Algorithm A1. Table 1 includes the numerical results of all the experiments for solving Test Problems 1 and 2. They show that the splitting Algorithm A1 was very efficient for solving all those problems with nonsymmetric ND matrices. In fact, the number of iterations and CPU time are always quite small and the accuracies of the computed solutions (measured by the quantities comp and dualfeas) are quite good. Furthermore, the BPP algorithm was quite efficient for solving the required LCPs as the number of iterations for this algorithm is at maximum 6. Note that no canonical vector is a solution of these EiCPs as the preprocessing technique mentioned before was not able to find a solution of the EiCP.

In Tables 2 and 3, we report the performance of Algorithms A1 and A2 for solving Test Problems 3. Note that, also in this case, all the instances were solved within the allowed number of iterations and stopping tolerance. It is important to add that Algorithms A1 and A2 work particularly well for the largest EiCP instances with very sparse matrices. This is noticed by the reduced number of iterations required by these algorithms for the last three EiCPs. Finally, the use of the line search technique seems to have no impact on the efficiency of the splitting algorithm as Algorithms A1 and A2 perform in a very similar way.

5.4. Performance for a PD matrix A

In this section, we report the performance of the proposed splitting algorithms for dealing with Test Problems 4, 5 and 6, in which A is a symmetric or nonsymmetric PD matrix. We show the performance of Algorithms B1 and B2 for solving these EiCPs. As already stated, we noted that Algorithms A1 and A2 perform better when the matrix A is ND. For this

Table 2. Performance of Algorithm A1 for solving Test Problems 3.

Problem	$cond_A$	Nnzeros	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
BCSSTK02(66)	1.30e+04	4340	−6.15318e+00	72	1.77752e−01	10	2	2.24	−6.45819e−10	−1.21102e−06
BCSSTK04(132)	5.60e+06	3420	−6.62140e+00	201	2.12439e−01	11	2	2.10	−1.00844e−09	−1.77936e−06
BCSSTK05(153)	3.50e+04	2421	−6.19116e+02	29	4.32233e−02	14	2	2.79	1.26639e−09	−1.26523e−05
BCSSTK10(1086)	1.30e+06	20400	−8.54132e+01	345	1.12597e+00	38	2	2.28	−5.57161e−08	−1.69793e−05
BCSSTK27(1224)	7.70e+04	56126	−2.09016e+02	403	2.10972e+00	35	2	2.35	−4.09381e−10	−1.33348e−04
s1rmq4m1(5489)	3.21e+06	239433	−3.85009e−01	14	2.32216e+00	48	2	6.92	−2.33476e−10	−3.39730e−07
s1rmt3m1(5489)	5.38e+06	198723	−3.85016e−01	12	1.97820e+00	60	2	8.82	9.53655e−09	−1.68196e−08
s2rmq4m1(5489)	1.15e+08	238813	−3.88911e−04	10	1.92511e+00	52	2	9.33	−1.59936e−10	−8.72999e−07

Table 3. Performance of Algorithm A2 for solving Test Problems 3.

Problem	$cond_A$	Nnzeros	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
BCSSTK02(66)	1.30e+04	4340	−6.15318e+00	72	7.80361e−02	10	2	2.24	−6.45819e−10	−1.21102e−06
BCSSTK04(132)	5.60e+06	3432	−6.62140e+00	201	2.12336e−01	11	2	2.10	−2.29675e−10	−1.77936e−06
BCSSTK05(153)	3.53e+04	2421	−6.19116e+02	29	4.00051e−02	14	2	2.79	−5.01549e−09	−1.26524e−05
BCSSTK10(1086)	1.31e+06	20400	−8.54132e+01	345	1.29363e+00	38	2	2.28	−2.38315e−07	−1.69793e−05
BCSSTK27(1224)	7.70e+04	56126	−2.09016e+02	403	2.37267e+00	35	2	2.35	−4.09417e−10	−1.33348e−04
s1rmq4m1(5489)	3.21e+06	239433	−3.85009e−01	14	2.48711e+00	48	2	6.92	−9.59145e−10	−3.39730e−07
s1rmt3m1(5489)	5.38e+06	198723	−3.85016e−01	12	2.08252e+00	60	2	8.82	8.21915e−09	−1.68204e−08
s2rmq4m1(5489)	3.22e+08	238813	−3.88911e−04	10	2.01258e+00	52	2	9.33	1.63350e−11	−8.72999e−07

reason, we present the solution of the same test problems by shifting the original matrix A as $A + \mu B$, with $\mu < 0$ and chosen such that $A + \mu B$ is ND.

As before, the value of tolerance to terminate the algorithms is 10^{-6} for all the test problems and the maximum number of iterations is set as 300 in the first two test problems, and as 5000 in Test Problems 6.

The numerical results of solving Test Problems 4 and 5 reported in Table 4 show that Algorithm B1 perform in a way similar to Algorithm A1 for solving the similar Test Problems 1 and 2. BPP algorithm is even more efficient for dealing with the LCP required in each iteration. Since the matrices of these LCPs are strictly diagonally dominant with positive diagonal elements, these experiments confirm the great efficiency of BPP algorithm to solve an LCP with such a class of matrices [26].

We shifted the matrix A in Test Problems 4 in order to obtain an ND matrix and we solved the resulting EiCP$(A + \mu B, B)$ with $\mu < 0$ by using Algorithm A1. The results, displayed in Table 5, show that Algorithm A1 efficiently solved all the instances. Although the number of required iterations is bigger than that required by Algorithm B1, the computational time is still very small so as the values in the last two columns.

In Tables 6 and 7, we present the results of Algorithms B1 and B2 for solving Test Problems 6, which are the same problems solved in [4]. As in [4], all the problems were scaled according to the procedure described in [14]. Note that the splitting algorithms were able to solve all the problems. As before, it seems that the use of the line search has no impact on the efficiency of this variant of the splitting algorithm, as Algorithms B1 and B2 perform in a very similar way. The same test problems were solved by Algorithms A1 and A2 by considering a shifted SND matrix. The corresponding results are showed in Tables 8 and 9 and show that Algorithms A1 and A2 solved very efficiently all these problems.

As for the EiCP with symmetric ND matrices, Algorithm A1 (and A2) seem to perform very well for large-scale symmetric EiCPs when the SPD matrix A is large and sparse. In our

Table 4. Performance of Algorithm B1 for solving Test Problems 4 and 5.

Problem	$cond_A$	$cond_B$	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
						$B = I$				
RAND(10)	9.42e+01	1	65.628911	12	6.86685e−03	2	2	2.00	−9.43720e−06	−1.00526e−05
RAND(20)	2.09e+03	1	123.771740	12	7.95694e−03	2	2	2.00	−1.72867e−06	−9.63468e−06
RAND(30)	1.03e+03	1	185.930994	11	6.73599e−03	2	2	2.00	−6.09484e−06	−6.89856e−06
RAND(40)	6.88e+03	1	241.820799	11	6.79372e−03	2	2	2.00	−2.31446e−06	−2.02950e−06
RAND(50)	2.57e+03	1	306.894486	15	9.83093e−03	2	2	2.00	7.63928e−07	−3.94007e−06
RAND(100)	8.34e+03	1	584.491662	10	5.55566e−01	2	2	2.00	8.97367e−07	−2.38216e−06
RAND(250)	1.17e+04	1	1429.103943	9	4.60674e−02	2	2	2.00	−5.53884e−07	−1.47538e−06
RAND(500)	2.55e+05	1	2827.577620	8	1.38157e−01	2	2	2.00	3.61825e−06	−5.63005e−06
RAND(750)	8.44e+04	1	4229.275068	8	2.89069e−01	2	2	2.00	1.01651e−06	−2.50049e−06
RAND(1000)	1.82e+05	1	5617.498456	8	4.88537e−01	2	2	2.00	−8.21974e−07	−1.32529e−06
						B given by (64)				
RAND(10)	9.42e+01	5.58e+00	15.447736	8	8.32302e−03	2	2	2.00	−1.52406e−06	−1.11256e−06
RAND(20)	2.09e+03	7.96e+00	41.816969	8	4.55666e−03	2	2	2.00	−4.21824e−07	−1.92430e−06
RAND(30)	1.03e+03	8.70e+00	71.823927	8	5.55609e−03	2	2	2.00	−3.86187e−07	−1.11698e−06
RAND(40)	6.88e+03	8.91e+00	99.533528	8	5.72159e−03	2	2	2.00	6.34172e−06	−9.00441e−07
RAND(50)	2.57e+03	8.97e+00	130.372316	7	4.57248e−03	2	2	2.00	−3.94954e−05	−6.31265e−06
RAND(100)	8.34e+03	8.99e+00	268.140323	8	7.13308e−01	2	2	2.00	1.50083e−06	−3.39707e−07
RAND(250)	1.17e+04	8.99e+00	685.167495	7	8.80536e−02	2	2	2.00	3.10199e−05	−4.61664e−06
RAND(500)	2.55e+05	8.99e+00	1378.802331	7	3.43613e−01	2	2	2.00	−4.63677e−06	−1.13262e−06
RAND(750)	8.44e+04	8.99e+00	2074.489158	7	6.58229e−01	2	2	2.00	−8.81876e−07	−6.23184e−07
RAND(1000)	1.82e+05	8.99e+00	2764.906847	7	1.15825e+00	2	2	2.00	−4.21272e−07	−3.26782e−07

Table 5. Performance of Algorithm A1 for solving Test Problems 4 and 5, with a shifted A matrix.

Problem	$cond_A$	$cond_B$	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
						$B = I$				
RAND(10)	8.89e+00	1	−15.099369	14	1.18758e−01	3	2	2.08	−2.14945e−06	−2.21869e−06
RAND(20)	1.62e+01	1	−18.138062	11	7.80958e−03	3	2	2.10	2.83959e−06	−2.60456e−06
RAND(30)	9.90e+00	1	−38.071466	12	7.57728e−03	3	2	2.09	2.15033e−06	−6.29441e−06
RAND(40)	1.36e+01	1	−45.174376	12	7.21936e−03	3	2	2.09	−3.33915e−06	−3.76009e−06
RAND(50)	1.17e+01	1	−58.785153	12	7.91940e−03	3	2	2.09	−6.29173e−06	−6.43847e−06
RAND(100)	1.79e+01	1	−78.549972	11	9.71219e−03	3	2	2.10	2.63267e−06	−1.79271e−06
RAND(250)	2.45e+01	1	−126.386143	10	1.74558e−02	3	2	2.11	1.46856e−06	−9.41413e−07
RAND(500)	3.47e+01	1	−177.863553	9	5.01653e−02	3	2	2.13	−2.53723e−05	−1.48843e−06
RAND(750)	3.54e+01	1	−255.355521	9	1.17962e−01	3	2	2.13	−2.42062e−05	−1.30335e−06
RAND(1000)	4.76e+01	1	−260.370783	9	2.40239e−01	3	2	2.13	3.04112e−06	−2.21559e−07
						B given by (64)				
RAND(10)	8.89e+00	5.58e+00	−3.617326	41	1.85492e−01	4	3	3.13	5.48184e−06	−1.31431e−05
RAND(20)	1.62e+01	7.96e+00	−5.536750	29	3.17392e−02	4	3	3.11	−4.46606e−06	−2.38530e−05
RAND(30)	9.90e+00	8.70e+00	−13.044711	87	2.30121e−01	5	3	3.95	1.27788e−05	−3.13080e−05
RAND(40)	1.36e+01	8.91e+00	−17.070718	72	6.23715e−02	5	3	3.14	−1.73384e−06	−7.02762e−05
RAND(50)	1.17e+01	8.97e+00	−23.000144	123	1.64172e−01	5	3	3.99	1.35113e−05	−6.93900e−05
RAND(100)	1.79e+01	8.99e+00	−34.786430	43	6.18236e−02	4	3	3.05	1.59829e−05	−9.58710e−05
RAND(250)	2.45e+01	8.99e+00	−60.428459	25	5.50755e−02	3	2	2.25	−2.55110e−06	−1.63434e−04
RAND(500)	3.47e+01	8.99e+00	−87.009985	19	1.31842e−01	3	2	2.22	−1.50078e−05	−1.44350e−04
RAND(750)	3.54e+01	8.99e+00	−125.838980	18	3.04071e−01	3	2	2.24	−3.10385e−05	−3.39770e−04
RAND(1000)	4.76e+01	8.99e+00	−128.800807	15	4.83246e−01	3	2	2.21	−1.07187e−05	−4.75847e−04

next experiment, we studied the proposed algorithms for solving large-scale nonsymmetric EiCP when A is a dense PD matrix. To do this, we considered large instances of Test Problems 5, in particular we set $n = [5000, 6000, 7000, 8000, 9000, 10000]$ and we generated the matrices A and B as explained in Section 5.1. We solved these problems by Algorithm A1, by shifting the matrix A, and by Algorithm B1. The numerical results of these experiments

Table 6. Performance of Algorithm B1 for solving Test Problems 6.

Problem	$cond_A$	Nnzeros	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
BCSSTK02(66)	1.30e+04	4340	1.02480e+00	79	5.85288e−02	2	2	2.00	−1.83868e−12	−1.61444e−08
BCSSTK04(132)	5.60e+06	3432	1.19001e+00	593	5.31760e−01	2	2	2.00	−2.29388e−12	−4.96939e−07
BCSSTK05(153)	3.53e+04	2421	8.57875e−01	33	3.18356e−02	2	2	2.00	−1.25706e−12	−4.97074e−11
BCSSTK10(1086)	1.31e+06	20400	4.70079e−02	3520	1.16391e+02	2	2	2.00	−9.38009e−14	−1.42185e−08
BCSSTK27(1224)	7.70e+04	56126	1.06951e+00	132	5.97071e+00	2	2	2.00	−1.92779e−12	−7.41130e−07
s1rmq4m1(5489)	3.21e+06	239433	1.31175e+00	2571	3.76096e+03	2	2	2.00	−2.60604e−12	−3.30718e−07
s1rmt3m1(5489)	5.38e+06	198723	1.22843e+00	4565	6.58323e+03	2	2	2.00	−2.45817e−12	−1.00903e−07
s2rmq4m1(5489)	3.22e+08	238813	1.30810e+00	3053	4.35372e+03	2	2	2.00	−2.64365e−12	−3.29944e−07

Table 7. Performance of Algorithm B2 for solving Test Problems 6.

Problem	$cond_A$	Nnzeros	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
BCSSTK02(66)	1.30e+04	4340	1.02480e+00	79	5.73456e−02	2	2	2.00	−1.83868e−12	−1.61444e−08
BCSSTK04(132)	5.60e+06	3432	1.19001e+00	593	4.23137e−01	2	2	2.00	−2.29388e−12	−4.96939e−07
BCSSTK05(153)	3.53e+04	2421	8.57875e−01	33	2.36030e−02	2	2	2.00	−1.25706e−12	−4.97074e−11
BCSSTK10(1086)	1.31e+06	20400	4.70079e−02	3520	5.02517e+01	2	2	2.00	−9.38492e−14	−1.42185e−08
BCSSTK27(1224)	7.70e+04	56126	1.06951e+00	132	2.84235e+00	2	2	2.00	−1.92779e−12	−7.41130e−07
s1rmq4m1(5489)	3.21e+06	239433	1.31175e+00	2571	8.34752e+02	2	2	2.00	−2.61089e−12	−3.30718e−07
s1rmt3m1(5489)	5.38e+06	198723	1.22843e+00	4565	1.58702e+03	2	2	2.00	−2.45817e−12	−1.00903e−07
s2rmq4m1(5489)	3.22e+08	238813	1.30810e+00	3053	9.99547e+02	2	2	2.00	−2.64365e−12	−3.29944e−07

Table 8. Performance of Algorithm A1 for solving Test Problems 6, with a shifted matrix A.

Problem	$cond_A$	Nnzeros	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
BCSSTK02(66)	2.48e+00	4340	−3.725258	2	9.27609e−02	3	3	3.00	−1.31073e−08	−7.68075e−05
BCSSTK04(132)	1.89e+00	3432	−3.350679	2	4.27694e−03	3	3	3.00	−4.36045e−09	−7.05687e−05
BCSSTK05(153)	2.78e+00	2421	−2.943277	69	7.14022e−02	4	2	2.13	−6.16498e−06	−2.50742e−05
BCSSTK10(1086)	4.02e+00	20400	−4.002941	2	1.06979e−02	3	3	3.00	−9.33448e−07	−1.22438e−03
BCSSTK27(1224)	4.08e+00	56126	−4.078541	2	1.55765e−02	4	4	4.00	−5.36495e−07	−8.42754e−04
s1rmq4m1(5489)	3.91e+00	39433	−4.011281	2	1.99958e−01	4	4	4.00	−3.51905e−07	−5.34151e−04
s1rmt3m1(5489)	3.82e+00	98723	−4.014641	2	1.94493e−01	4	4	4.00	−2.09611e−07	−4.08494e−04
s2rmq4m1(5489)	3.89e+00	38813	−4.004969	2	1.88591e−01	3	3	3.00	−2.93536e−09	−4.87054e−05

Table 9. Performance of Algorithm A2 for solving Test Problems 6, with a shifted matrix A.

Problem	$cond_A$	Nnzeros	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
BCSSTK02(66)	2.48e+00	4340	−3.725258	2	1.64026e−02	3	3	3.00	−1.31073e−08	−7.68075e−05
BCSSTK04(132)	1.89e+00	3432	−3.350679	2	2.50113e−03	3	3	3.00	−4.36045e−09	−7.05687e−05
BCSSTK05(153)	2.78e+00	2421	−2.943277	69	5.68438e−02	4	2	2.13	−6.16498e−06	−2.50742e−05
BCSSTK10(1086)	4.02e+00	20400	−4.002941	2	6.86064e−03	3	3	3.00	−9.33448e−07	−1.22438e−03
BCSSTK27(1224)	4.08e+00	56126	−4.078541	2	9.58222e−03	4	4	4.00	−5.36495e−07	−8.42754e−04
s1rmq4m1(5489)	3.91e+00	39433	−4.011281	2	1.37201e−01	4	4	4.00	−3.51905e−07	−5.34151e−04
s1rmt3m1(5489)	3.82e+00	98723	−4.014641	2	1.35653e−01	4	4	4.00	−2.09611e−07	−4.08494e−04
s2rmq4m1(5489)	3.89e+00	38813	−4.004969	2	1.33916e−01	3	3	3.00	−2.93536e−09	−4.87054e−05

included in Tables 10 and 11 confirm the efficiency of the proposed algorithms, as all the instances were accurately solved in a small number of iterations and computational time.

The numerical experiments reported in this section lead to our recommendation of using Algorithm A1 for the solution of an EiCP with a symmetric or nonsymmetric ND matrix A and $D = -1/2(A + A^t)$ ($D = -A$ if A is SND). If A is SND, then Algorithm A2 is a valid alternative to Algorithm A1. If A is not ND, then a solution of EiCP can still be computed efficiently by Algorithms A1 and A2 (in the symmetric case) applied to a shifted

Table 10. Performance of Algorithm A1 for solving large instances of Test Problems 5, with a shifted matrix A.

Problem	$cond_A$	$cond_B$	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
RAND(5000)	8.95e+01	8.99e+00	−321.644427	12	1.50643e+01	3	2	2.27	5.75963e−05	−2.42853e−04
RAND(6000)	8.99e+01	8.99e+00	−380.387984	12	2.46081e+01	3	2	2.27	5.99037e−05	−2.62447e−04
RAND(7000)	1.04e+02	8.99e+00	−380.780731	11	3.37807e+01	3	2	2.30	1.82876e−04	−6.02156e−04
RAND(8000)	1.05e+02	8.99e+00	−426.238660	11	4.89954e+01	3	2	2.30	1.80370e−04	−6.03307e−04
RAND(9000)	1.12e+02	8.99e+00	−459.240911	11	6.94255e+01	3	2	2.30	1.62765e−04	−4.79112e−04
RAND(10000)	1.03e+01	8.99e+00	−558.102886	11	8.36563e+01	3	2	2.30	2.59913e−04	−1.20086e−03

Table 11. Performance of Algorithm B1 for solving large instances of Test Problems 5.

Problem	$cond_A$	$cond_B$	λ	iter	CPU	w_{BPP}	b_{BPP}	av_{BPP}	comp	dualfeas
RAND(5000)	2.74e+06	8.99e+00	13800.782159	6	1.11416e+01	2	2	2.00	−3.60803e−06	−3.57341e−06
RAND(6000)	1.83e+06	8.99e+00	16557.773299	6	1.62209e+01	2	2	2.00	3.47488e−06	−3.45744e−06
RAND(7000)	1.81e+06	8.99e+00	19312.184161	6	2.41998e+01	2	2	2.00	9.11727e−06	−2.48884e−06
RAND(8000)	2.00e+06	8.99e+00	22066.676993	6	3.53896e+01	2	2	2.00	5.89925e−06	−2.12796e−06
RAND(9000)	2.79e+06	8.99e+00	24819.123060	6	5.18718e+01	2	2	2.00	−1.05008e−05	−1.98170e−06
RAND(10000)	2.60e+07	8.99e+00	27577.374734	6	6.87384e+01	2	2	2.00	−2.99658e−09	−1.56846e−06

Table 12. Performance of Algorithms SPG and SBAS for solving Test Problems 6.

Solver	Problem	λ	iter
SPG	BCSSTK02(66)	1.04140e+00	48
SBAS		1.02480e+00	62
SPG	BCSSTK04(132)	1.19001e+00	91
SBAS		1.19001e+00	97
SPG	BCSSTK05(153)	1.02522e+00	947
SBAS		8.57875e−01	20
SPG	BCSSTK10(1086)	1.45692e+00	5000*
SBAS		4.70079e−02	454
SPG	BCSSTK27(1224)	1.31341e+00	5000*
SBAS		1.30785e+00	192
SPG	s1rmq4m1(5489)	1.31175e+00	440
SBAS		1.31175e+00	403
SPG	s1rmt3m1(5489)	1.22843e+00	386
SBAS		1.22843e+00	1077
SPG	s2rmq4m1(5489)	1.30810e+00	5000*
SBAS		1.30810e+00	347

EiCP($A + \mu B$, B) with $\mu < 0$ such that $A + \mu B$ is ND. Algorithms B1 and B2 can also be very efficient to deal with an EiCP with a PD matrix A but do not seem so robust as their corresponding versions A1 and A2.

5.5. Comparison with other algorithms

In this section, we include numerical results on the solutions of the test problems with a PD matrix by some algorithms that are considered to be the best for solving the symmetric and nonsymmetric EiCPs. Note that we only consider Test Problems 4, 5 and 6, as these are the ones whose numerical results for all the four versions of the splitting method have been reported before in this paper and in [4] (symmetric instances).

For the symmetric EiCP, we include in Table 12 the results reported in [4] for solving Test Problems 6 by the spectral block active set (SBAS) algorithm and spectral-projected

Table 13. Performance of the semi-smooth Newton method for solving Test Problems 4 and 5.

Problem	λ	Iter	CPU	comp	dualfeas
			$B = I$		
RAND(10)	66.805433	11	1.81041e−02	−2.99923e−08	−3.01356e−07
RAND(20)	128.390806	14	3.65155e−03	−5.56204e−10	−5.63503e−09
RAND(30)	178.657680	10	2.26015e−03	−2.42309e−08	−1.08491e−07
RAND(40)	242.214753	7	2.84879e−03	−1.31110e−09	−4.28940e−09
RAND(50)	304.231260	10	1.53517e−02	−1.21935e−09	−6.09147e−08
RAND(100)	585.038741	14	2.96391e−02	−1.76570e−10	−1.24657e−09
RAND(250)	1431.727973	11	1.25984e−01	−1.73999e−12	−4.41910e−10
RAND(500)	2829.469606	9	5.53335e−01	−3.09304e−15	−1.97238e−14
RAND(750)	4222.157448	18	2.83971e+00	−1.73143e−11	−1.54318e−10
RAND(1000)	5613.654481	14	4.54821e+00	−4.70814e−10	−4.72324e−07
			B given by (64)		
RAND(10)	15.641489	9	1.62196e−02	−1.15809e−08	−3.27750e−08
RAND(20)	43.998835	41	6.05905e−03	−6.72194e−09	−2.15176e−08
RAND(30)	68.965288	41	7.52118e−03	−1.84191e−08	−1.18727e−07
RAND(40)	99.820525	14	2.52377e−02	−3.80668e−09	−9.71869e−09
RAND(50)	129.700395	20	1.19147e−02	−5.77790e−10	−2.53174e−09
RAND(100)	268.709076	20	4.32256e−02	−3.09453e−10	−1.20887e−09
RAND(250)	686.576729	12	1.17462e−01	−3.14756e−15	−2.03466e−14
RAND(500)	1379.758459	17	1.02307e+00	−1.60295e−11	−2.30102e−10
RAND(750)	2071.347344	21	3.67975e+00	−3.94160e−11	−1.53085e−10
RAND(1000)	2763.086773	36	1.18872e+01	−3.61007e−10	−3.09248e−09

gradient (SPG) method [4]. An * is written in the table when the algorithm was not able to terminate in a number of iterations smaller than the maximum limit allowed. The initial point for both algorithms was chosen by the preprocessing technique that was also used for the splitting methods. A comparison with the performance of the splitting algorithms for solving the same test problems (see Tables 8 and 9) leads to the conclusion that the Algorithms A1 and A2 seem to be more efficient than the SBAS algorithm, which is the best of the two projected-gradient methods.

For the nonsymmetric EiCP, we report the experiments of solving Test Problems 4 and 5 by the semi-smooth Newton method [1] and the hybrid method [8]. For the semi-smooth Newton method, we set a maximum number of iteration as 100 and a stopping tolerance set as 10^{-6}. In Table 13, we report the value of the computed complementary eigenvalue, the required number of iterations, and the CPU time in seconds. The columns titled 'comp' and 'dualfeas' have the usual definition. The initial point for this method is given by $(\bar{x}, \bar{\lambda}, \bar{w})$, where $\bar{x}$ is computed by the preprocessing technique that was employed for the splitting methods, $\bar{\lambda} = (\bar{x}^t A\bar{x})/(\bar{x}^t B\bar{x})$ and $w = (\bar{\lambda}B - A)\bar{x}$. The method was able to successfully solve all the instances of Test Problems 4 and 5. Since for these nonsymmetric Test Problems 4 and 5, the semi-smooth Newton method is successful with a special given initial point, then hybrid algorithm first applies this last method and terminates with the solution of EiCP computed by this algorithm. The numerical results reported in Table 13 indicate that the splitting methods A1 and B1 are competitive (and even more efficient for some instances) with the semi-smooth method in terms of iterations, CPU time and accuracy of the computed solution. We also solved large instances of Test Problems 5 by using the hybrid algorithm starting with the semi-smooth Newton method. The corresponding results in Table 14 show that the semi-smooth method used alone or within the hybrid method fails to find a solution for two instances, which, instead, were successfully solved

Table 14. Performance of the hybrid algorithm for solving large instances of Test Problems 5.

Problem	λ	Iter	CPU	comp	dualfeas
RAND(5000)	*				
RAND(6000)	16557.773299	14	5.57236e+02	−2.33955e−11	−5.68051e−11
RAND(7000)	19312.184161	18	1.12480e+03	−2.60797e−12	−1.61638e−11
RAND(8000)	22066.676993	19	1.76105e+03	−4.31110e−11	−3.83147e−10
RAND(9000)	24819.123060	25	3.37230e+03	−2.43958e−12	−3.38710e−10
RAND(10000)	*				

by Algorithms A1 and B1. Moreover, the computational time for the successful instances is much bigger than that required by the splitting methods proposed in this paper.

6. Conclusion

In this paper, we have studied splitting methods for solving the Eigenvalue Complementarity Problem EiCP(A,B). We proposed four variants of splitting methods, called A1, A2, B1 and B2, which are implemented depending on the properties of matrices A and B. Convergence analysis for each one of the four versions of the splitting method is presented. Simple choices for the splitting matrices are introduced that seem to work well in practice. Furthermore, the most efficient variants of the splitting method are shown to be competitive with the best state-of-the-art algorithms for the solution of symmetric and nonsymmetric EiCPs. Sufficient conditions that guarantee global convergence for the splitting algorithms should be investigated in the future. Furthermore, other choices for the splitting matrices are expected to be done particularly for the solution or structured EiCPs that appear in applications. Finally, it is interesting in the future to extend the splitting method to deal with the so-called Quadratic Eigenvalue Complementarity Problem (QEiCP) [12] and eigenvalue complementarity problems EiCP and QEiCP on other convex cones [3].

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The work of Alfredo N. Iusem was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico CNPq grant no. 301280/86. In the scope of R&D Unit 50008, Joaquim J. Júdice was financed by the applicable financial framework (FCT/MEC through national funds and when applicable co-funded by FEDER PT2020 partnership agreement).

ORCID

Valentina Sessa http://orcid.org/0000-0002-0083-2515

References

[1] S. Adly and A. Seeger, *A nonsmooth algorithm for cone constrained eigenvalue problems*, Comput. Optim. Appl. 49 (2011), pp. 299–318.

[2] C. Brás, M. Fukushima, J. Júdice, and S. Rosa, *Variational inequality formulation for the asymmetric eigenvalue complementarity problem and its solution by means of gap functions*, Pac. J. Optim. 8 (2012), pp. 197–215.
[3] C. Brás, M. Fukushima, A. Iusem, and J. Júdice, *On the quadratic eigenvalue complementarity problem over a general convex cone*, Appl. Math. Comput. 271 (2015), pp. 594–608.
[4] C. Brás, A. Fischer, J. Júdice, K. Schönefeld, and S. Seifert, *A block active set algorithm with spectral choice line search for the symmetric eigenvalue complementarity problem*, Appl. Math. Comput. 294 (2017), pp. 36–48.
[5] R.W. Cottle, J.S. Pang, and R.S. Stone, *The Linear Complementarity Problem*, Academic Press, New York, 1992.
[6] A.R. De Pierro and A.N. Iusem, *Convergence properties of iterative methods for symmetric positive semidefinite linear complementarity problems*, Math. Oper. Res. 18 (1993), pp. 317–333.
[7] F. Facchinei and J.S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Springer, Berlin, 2003.
[8] L.M. Fernandes, J. Júdice, H. Sherali, and M.A. Forjaz, *On an enumerative algorithm for solving eigenvalue complementarity problems*, Comput Optim Appl 59 (2014), pp. 113–134.
[9] L.M. Fernandes, J. Júdice, H. Sherali, and M Fukushima, *On the computation of all eigenvalues for the eigenvalue complementarity problem*, J. Global Optim. 59 (2014), pp. 307–326.
[10] G.H. Golub and C.F. Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, 1996.
[11] http://math.nist.gov/MatrixMarket/index.html
[12] A. Iusem, J. Júdice, V. Sessa, and H. Sherali, *On the numerical solution of the quadratic eigenvalue complementarity problem*, Numer. Algorithms 72 (2016), pp. 721–747.
[13] J. Júdice and F. Pires, *A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems*, Comput. Oper. Res. 21 (1994), pp. 587–596.
[14] J. Júdice, H. Sherali, and I. Ribeiro, *The eigenvalue complementarity problem*, Comput. Optim. Appl. 37 (2007), pp. 139–156.
[15] J. Júdice, M. Raydan, S. Rosa, and S. Santos, *On the solution of the symmetric complementarity problem by the spectral projected gradient method*, Numer. Algorithms 44 (2008), pp. 391–407.
[16] J. Júdice, H. Sherali, I. Ribeiro, and S. Rosa, *On the asymmetric eigenvalue complementarity problem*, Optim. Methods Softw. 24 (2009), pp. 549–568.
[17] H. Le Thi, M. Moeini, T. Pham Dinh, and J. Júdice, *A DC programming approach for solving the symmetric eigenvalue complementarity problem*, Comput. Optim. Appl. 5 (2012), pp. 1097–1117.
[18] MATLAB, version 8.0.0.783 (R2012b). The MathWorks Inc., Natick, MS, 2012.
[19] Y.S. Niu, T. Pham Dinh, H. Le Thi, and J. Júdice, *Efficient DC programming approaches for the asymmetric eigenvalue complementarity problem*, Optim. Methods Softw. 28 (2013), pp. 812–829.
[20] A.M. Ostrowsky, *Solution of Equations and Systems of Equations*, Academic Press, New York, 1996.
[21] A. Pinto da Costa and A. Seeger, *Cone constrained eigenvalue problems, theory and algorithms*, Comput. Optim. Appl. 45 (2010), pp. 25–57.
[22] A. Pinto da Costa, J. Martins, I. Figueiredo, and J. Júdice, *The directional instability problem in systems with frictional contacts*, Comput. Methods Appl. Mech. Eng. 193 (2004), pp. 357–384.
[23] M. Queiroz, J. Júdice, and C. Humes, *The symmetric eigenvalue complementarity problem*, Math. Comput. 73 (2003), pp. 1849–1864.
[24] A. Seeger, *Eigenvalue analysis of equilibrium processes defined by linear complementarity conditions*, Linear. Algebra Appl. 294 (1999), pp. 1–14.
[25] A. Seeger and M. Torki, *On eigenvalues induced by a cone constraint*, Linear Algebra Appl. 372 (2003), pp. 181–206.
[26] S. Takriti and K. Murty, *On the convergence of the block principal pivoting algorithm for the LCP*, Eur. J. Oper. Res. 102 (1997), pp. 657–666.
[27] R.S. Varga, *Matrix Iterative Analysis*, Prentice Hall, New Jersey, 1962.

[28] A. Wächter and L.T. Biegler, *On the implementation of a primal–dual interior point filter line search algorithm for large-scale nonlinear programming*, Math. Program. 106 (2006), pp. 25–57.
[29] Y. Zhou and M. Gowda, *On the finiteness of the cone spectrum of certain linear transformations on euclidean Jordan algebras*, Linear Algebra Appl. 431 (2009), pp. 772–782.