Journal of Scientific Computing (2019) 79:1608–1629
https://doi.org/10.1007/s10915-019-00907-4

An Efficient Numerical Method for the Symmetric Positive Definite Second-Order Cone Linear Complementarity Problem

Xiang Wang[1] · Xing Li[2] · Lei-Hong Zhang[3] · Ren-Cang Li[4]

Received: 10 April 2018 / Revised: 6 November 2018 / Accepted: 7 January 2019 /
Published online: 1 February 2019

Abstract
An efficient numerical method for solving a symmetric positive definite second-order cone linear complementarity problem (SOCLCP) is proposed. The method is shown to be more efficient than recently developed iterative methods for small-to-medium sized and dense SOCLCP. Therefore it can serve as an excellent core computational engine in solutions of large scale symmetric positive definite SOCLCP solved by subspace projection methods, solutions of general SOCLCP and the quadratic programming over a Cartesian product of multiple second-order cones, in which small-to-medium sized SOCLCPs have to be solved repeatedly, efficiently, and robustly.

Keywords Second-order cone · Linear complementarity problem · SOCLCP · Globally uniquely solvable property · GUS

Mathematics Subject Classification 90C33 · 65K05 · 65F99 · 65F15 · 65F30 · 65P99

✉ Ren-Cang Li
rcli@uta.edu

Xiang Wang
wangxiang49@ncu.edu.cn

Xing Li
lxkate@163.com

Lei-Hong Zhang
longzlh@163.com

[1] Department of Mathematics, Nanchang University, 999 Xuefu Road, Nanchang 330031, China

[2] School of Mathematics, Shanghai University of Finance and Economics, 777 Guoding Road, Shanghai 200433, China

[3] School of Mathematical Sciences, Soochow University, Suzhou 215006, Jiangsu, China

[4] Department of Mathematics, University of Texas at Arlington, P.O. Box 19408, Arlington, TX 76019-0408, USA

1 Introduction

Given a symmetric positive definite matrix $M \in \mathbb{R}^{n\times n}$, we are interested in solving the following *second-order cone linear complementarity problem* (SOCLCP):

$$\boldsymbol{x} \in \mathbb{K}^n, \quad \boldsymbol{q} + M\boldsymbol{x} \in \mathbb{K}^n, \quad \boldsymbol{x}^{\mathrm{T}}(\boldsymbol{q} + M\boldsymbol{x}) = 0, \tag{1.1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$, and $\mathbb{K}^n$ is the *second-order cone* (SOC):

$$\mathbb{K}^n := \left\{ \left[x_1, \boldsymbol{x}_2^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R} \times \mathbb{R}^{n-1} \,:\, \|\boldsymbol{x}_2\|_2 \le x_1 \right\}.$$

One source of (1.1) is the KKT system of the following conic optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{K}^n} \frac{1}{2}\boldsymbol{x}^{\mathrm{T}} M\boldsymbol{x} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{x},$$

where M is symmetric positive definite.

Since M is assumed symmetric positive definite, SOCLCP (1.1) admits the *globally uniquely solvable* (GUS) property, i.e., it has a unique solution for any given $\boldsymbol{q} \in \mathbb{R}^n$. The GUS property for SOCLCP has been studied in, e.g., [13–15,32]. In particular, Yang and Yuan [32] provided an algebraic characterization of the GUS property. Based on the characterization, several efficient iterative methods [36–38] have been developed. Other numerical methods, related theory and applications for SOCLCP, can be found in, e.g., [3,7,8,11,16,20,26,33,34].

Besides the GUS property, the symmetric positive definiteness of M allows us to use the efficient state-of-the-art eigendecomposition subroutines in LAPACK [2] to decompose M or the matrix pencil $M - \lambda J_n$ to develop more efficient methods for (1.1), especially for modest n, than what we have in the literature, where

$$J_n := \mathrm{diag}(1, -1, \ldots, -1) \in \mathbb{R}^{n\times n}. \tag{1.2}$$

Our main goal of this paper is to develop an efficient numerical method for solving SOCLCP (1.1) with a symmetric positive definite M along this line. The method starts by computing the eigendecomposition of $M - \lambda J_n$ to turn SOCLCP (1.1) into a zero-finding problem for a rational function and then a very fast zero-finder based on a simple rational approximation is devised for the transformed problem. The proposed method is fast and robust for small-to-medium sized SOCLCP and, though iterative in nature, can be regarded as a direct method, just as nowadays the QR algorithm for the eigenvalue problem is widely treated as a direct method due to the fact that it is such a robust iterative algorithm.

Conceivably, there are several scenarios the new method can be called upon to play critical roles: (1) when a symmetric positive definite SOCLCP (1.1) itself is of modest size, (2) during intermediate steps of solving a large scale SOCLCP by projections, numerous SOCLCP of modest sizes have to be efficiently solved [38, Theorem 6.7], (3) any general SOCLCP

$$\boldsymbol{x} \in \mathbb{K}_{\times m}, \quad \boldsymbol{q} + M\boldsymbol{x} \in \mathbb{K}_{\times m}, \quad \boldsymbol{x}^{\mathrm{T}}(\boldsymbol{q} + M\boldsymbol{x}) = 0, \tag{1.3}$$

over a Cartesian product of multiple second-order cones $\mathbb{K}_{\times m} := \mathbb{K}^{n_1} \times \mathbb{K}^{n_2} \times \cdots \times \mathbb{K}^{n_m}$ by the matrix splitting method [37, Algorithm 1] in which a sequence of small size SOCLCPs have to be solved during each iterative step, and (4) the quadratic programming over $\mathbb{K}_{\times m}$ (QP-SOC) (i.e., minimization of a quadratic $\frac{1}{2}\boldsymbol{x}^{\mathrm{T}} M\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}$ subject to $A\boldsymbol{x} = \boldsymbol{b}$, $\boldsymbol{x} \in \mathbb{K}_{\times m}$) by the Augmented Lagrangian Method (ALM) [18,27] in which a particular SOCLCP (1.3) needs to be solved in each iteration.

The rest of this article is organized as follows. In Sect. 2, we first present basic properties of SOCLCP, and in Sect. 3, we apply the eigendecomposition of the symmetric positive

definite pencil $M - \lambda J_n$ to transform SOCLCP into a zero-finding problem. Basing on the new reformulation, we next describe the solution for the critical case in Sect. 4. Theoretical results and detailed algorithmic procedure for the generic case are detailed in Sect. 5. To demonstrate potential usages of our proposed algorithm, in Sect. 6, we provide two applications of our algorithm, namely the general SOCLCP (1.3), and QP-SOC. Numerical results of our new algorithm as well as its performance in these two applications are reported in Sect. 7. In particular, we conduct our numerical testing by comparing the efficiency of the proposed algorithm with CVX (a modeling system for constructing and solving various convex programming) and two other recent algorithms: the bisection-Newton method (BN) [36] and the linear complementarity problem via the Arnold process (LCPvA) [38] for (1.1). Final remarks are drawn in Sect. 8.

Notation: Throughout this paper, all vectors are column vectors and are typeset in bold lower case letters. For $A \in \mathbb{R}^{m\times n}$ (the set of all $m \times n$ real matrices), A^{T} denotes its transpose, and $\mathcal{R}(A)$ and $\mathcal{N}(A)$ represent the range and kernel of A, respectively. Thus $\mathcal{R}(A)^{\perp} = \mathcal{N}(A^{\mathrm{T}})$, where $\mathcal{R}(A)^{\perp}$ denotes the orthogonal complement of $\mathcal{R}(A)$. As usual, the identity matrix in $\mathbb{R}^{n\times n}$ will be denoted by $I_n \equiv [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n]$, where $\boldsymbol{e}_i$ is its i column. We shall also adopt MATLAB-like convention to access the entries of vectors and matrices. For example, $\boldsymbol{x}_{(i)}$ is the ith element of $\boldsymbol{x}$ and $A_{(i,j)}$ is the (i, j)th entry of A, where $(i : j)$ stands for the set of integers from i to j inclusive, and $A_{(k:\ell,i:j)}$ is the submatrix of A that consists of intersections from row k to row ℓ and column i to column j. Notation $\|\cdot\|_2$ is either the matrix spectral norm or the Euclidean vector norm, depending on its arguments:

$$\|\boldsymbol{x}\|_2 := \sqrt{\sum_i |\boldsymbol{x}_{(i)}|^2}, \quad \|A\|_2 := \max_{\boldsymbol{x}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}.$$

2 Basic Properties of SOCLCP

To simplify our presentation, we will denote (1.1) by SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ whose set of solutions will be denoted by SOL$(M, \mathbb{K}^n, \boldsymbol{q})$. In this section, we review several basic results for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$. Denote by $\partial(\mathbb{K}^n)$ and int$(\mathbb{K}^n)$ the boundary and the interior of $\mathbb{K}^n$, respectively.

Lemma 2.1 ([32]) *The following statements hold:*

(i) *For nonzero vectors* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{K}^n$, $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = 0$ *if and only if* $\boldsymbol{x} \in \partial(\mathbb{K}^n)$, $\boldsymbol{y} \in \partial(\mathbb{K}^n)$ *and* $\boldsymbol{y} = sJ_n\boldsymbol{x}$ *for some* $s > 0$.
(ii) *Let* $\boldsymbol{x} \in \mathbb{K}^n$. *Then* $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} > 0$ *for all nonzero vectors* $\boldsymbol{y} \in \mathbb{K}^n$ *if and only if* $\boldsymbol{x} \in \mathrm{int}(\mathbb{K}^n)$.

The next theorem characterizes the three mutually exclusive cases for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$. We point out (C2) can be equivalently restated as $-M^{-1}\boldsymbol{q} \in \mathbb{K}^n$ (which implies that $\boldsymbol{x} = -M^{-1}\boldsymbol{q}$ is the solution). In (C3), $M - s_*J_n$ may or may not be singular.

Theorem 2.1 ([36,37]) *There are three mutually exclusive cases for the solution* $\boldsymbol{x} \in$ SOL$(M, \mathbb{K}^n, \boldsymbol{q})$, *namely:*

(C1) $\boldsymbol{q} \in \mathbb{K}^n$ *(which implies that* $\boldsymbol{x} = \boldsymbol{0}$ *is the solution);*
(C2) SOL$(M, \mathbb{K}^n, \boldsymbol{q}) \supseteq \{\boldsymbol{x} \in \mathbb{K}^n : M\boldsymbol{x} + \boldsymbol{q} = \boldsymbol{0}\} \neq \varnothing$;
(C3) *there exists* $s_* > 0$ *such that* $M\boldsymbol{x} + \boldsymbol{q} = s_*J_n\boldsymbol{x} \in \partial(\mathbb{K}^n)$.

Recalling that M is assumed symmetric positive definite, we then have the following eigendecomposition.

Lemma 2.2 *There is a nonsingular matrix $V \in \mathbb{R}^{n\times n}$ such that*

$$V^{\mathrm{T}} M V = \Omega \equiv \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_n), \quad V^{\mathrm{T}} J_n V = J_n, \tag{2.1}$$

where $0 < \omega_1$ and $0 < \omega_2 \le \cdots \le \omega_n$.

Proof Since M is assumed symmetric positive definite, it has a Cholesky decomposition $M = R^{\mathrm{T}} R$. Now notice that $R^{-\mathrm{T}} J_n R^{-1} \in \mathbb{R}^{n\times n}$ is symmetric and let its eigenvalues be $\{\mu_i\}_{i=1}^n$. Because $R^{-\mathrm{T}} J_n R^{-1} \in \mathbb{R}^{n\times n}$ has the same inertia as J_n, these eigenvalues can be ordered in such a way that

$$\mu_n \le \cdots \le \mu_2 < 0 < \mu_1.$$

$R^{-\mathrm{T}} J_n R^{-1}$ has an eigendecomposition

$$R^{-\mathrm{T}} J_n R^{-1} = U D U^{\mathrm{T}}, \quad D = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_n),$$

where U is an orthogonal matrix. Set $\omega_i = 1/|\mu_i|$ for $1 \le i \le n$. We have

$$R^{-\mathrm{T}} J_n R^{-1} = U D U^{\mathrm{T}} = U J_n \Omega^{-1} U^{\mathrm{T}} = U \Omega^{-1/2} J_n \Omega^{-1/2} U^{\mathrm{T}}.$$

Finally set $V = R^{-1} U \Omega^{1/2}$ to conclude the proof. □

The equations in (2.1) give an eigendecomposition of the matrix pencil $M - \lambda J_n$. In particular, it has one positive eigenvalue ω_1 and $n-1$ negative eigenvalues $-\omega_i$:

$$-\omega_n \le \cdots \le -\omega_2 < 0 < \omega_1,$$

and the columns of V are the corresponding eigenvectors. In fact, it can be verified that

$$M V J_n = J_n V \Omega.$$

We also have

$$M J_n (J_n V) = M V = V^{-\mathrm{T}} V^{\mathrm{T}} M V = V^{-\mathrm{T}} \Omega = J_n V J_n \Omega = (J_n V) J_n \Omega,$$

where we have used $V^{-\mathrm{T}} = J_n V J_n$ by the second equation in (2.1). This gives an eigendecomposition of $M J_n$. In particular, $M J_n$ has the same eigenvalues as $M - \lambda J_n$ with corresponding eigenvectors given by the columns of $J_n V$.

The next theorem provides an algebraic-geometric characterization of the solution set $\mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$. The reader is referred to [32,36,37] for proofs and more.

Theorem 2.2 *If $\boldsymbol{x} \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$ is not in the cases* (C1) *and* (C2)*, i.e., $\boldsymbol{q} \notin (-M\mathbb{K}^n) \cup \mathbb{K}^n$, then there exists a unique $s_* > 0$ such that $M\boldsymbol{x} + \boldsymbol{q} = s_* J_n \boldsymbol{x}$ and*

$$s_* \begin{cases} = \omega_1, & \text{if } (-\boldsymbol{q})^{\mathrm{T}} \boldsymbol{v} = 0 \ \text{ or equivalently, } \ \boldsymbol{q} \in \mathcal{R}(M - \omega_1 J_n), \\ < \omega_1, & \text{if } (-\boldsymbol{q})^{\mathrm{T}} \boldsymbol{v} > 0, \\ > \omega_1, & \text{if } (-\boldsymbol{q})^{\mathrm{T}} \boldsymbol{v} < 0, \end{cases}$$

where $\boldsymbol{v}$ is the first column of V, the eigenvector of $M - \lambda J_n$ associated with ω_1: $M\boldsymbol{v} = \omega_1 J_n \boldsymbol{v}$, and $\mathcal{R}(M - \omega_1 J_n) = \boldsymbol{v}^{\perp} := \{\boldsymbol{u} \in \mathbb{R}^n \ : \ \boldsymbol{u} \perp \boldsymbol{v}\}$.

3 Transform SOCLCP into a Zero-Finding Problem

Our new method relies on the following simple observation.

Theorem 3.1 ([38]) *Suppose $s \in \mathbb{R}$ is not an eigenvalue of $M - \lambda J_n$. Let*

$$\boldsymbol{x}(s) \equiv \begin{bmatrix} x_1(s) \\ \boldsymbol{x}_2(s) \end{bmatrix} := -(M - sJ_n)^{-1}\boldsymbol{q},$$

where J_n is given in (1.2). Then $\boldsymbol{x}(s) \in \partial(\mathbb{K}^n)$ if and only if $x_1(s) > 0$ and $h(s) = 0$, where

$$h(s) := \boldsymbol{x}(s)^{\mathrm{T}} J_n \boldsymbol{x}(s) = \boldsymbol{q}^{\mathrm{T}}(M - sJ_n)^{-\mathrm{T}} J_n (M - sJ_n)^{-1}\boldsymbol{q}.$$

Using the decompositions in (2.1), we have

$$\boldsymbol{x}(s) = -V(\Omega - sJ_n)^{-1}V^{\mathrm{T}}\boldsymbol{q},$$

and

$$\begin{aligned} h(s) &= \boldsymbol{q}^{\mathrm{T}} V(\Omega - sJ_n)^{-1} J_n (\Omega - sJ_n)^{-1} V^{\mathrm{T}}\boldsymbol{q} \\ &= \frac{\xi_1^2}{(\omega_1 - s)^2} - \sum_{i=2}^{n} \frac{\xi_i^2}{(\omega_i + s)^2} \\ &= \frac{\xi_1^2}{(s - \omega_1)^2} - \sum_{i=2}^{n} \frac{\xi_i^2}{(s + \omega_i)^2}, \end{aligned} \tag{3.1}$$

where ξ_i for $1 \le i \le n$ are the entries of $V^T\boldsymbol{q}$, i.e.,

$$V^{\mathrm{T}}\boldsymbol{q} = [\xi_1, \ldots, \xi_n]^{\mathrm{T}}. \tag{3.2}$$

According to Theorem 3.1, we know that finding s_* for the case (C3) in Theorem 2.1 is equivalent to finding the positive zero s_* of $h(s)$.

Lemma 3.1 *Suppose $\xi_1 \neq 0$ and not all ξ_i for $2 \le i \le n$ are 0.*

1. *If $h(0) \ge 0$, i.e., $\boldsymbol{q}^{\mathrm{T}} M^{-\mathrm{T}} J_n M^{-1}\boldsymbol{q} \ge 0$, and if $(-M^{-1}\boldsymbol{q})_{(1)} \ge 0$, then $\boldsymbol{x} = -M^{-1}\boldsymbol{q}$ is in $\mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$.*
2. *If $h(0) < 0$, then $h(s)$ has one and only one zero in $(0, \omega_1)$; if $h(0) > 0$, then $h(s) > 0$ for all $s \in [0, \omega_1)$.*
3. *If $\boldsymbol{q}^{\mathrm{T}} J_n \boldsymbol{q} \ge 0$, then $h(s) > 0$ for all $s \in (\omega_1, \infty)$; if $\boldsymbol{q}^{\mathrm{T}} J_n \boldsymbol{q} < 0$, then $h(s)$ has one and only one zero in (ω_1, ∞).*

Proof For item 1, by assumption, $\boldsymbol{x} = -M^{-1}\boldsymbol{q} \in \mathbb{K}^n$ and at the same time $M\boldsymbol{x} + \boldsymbol{q} = 0$.

For $-\omega_2 < s < \omega_1$, we have

$$h'(s) = -2\frac{\xi_1^2}{(s - \omega_1)^3} + 2\sum_{i=2}^{n} \frac{\xi_i^2}{(s + \omega_i)^3} > 0.$$

Now item 2 is a direct consequence of this.

Consider now $s > \omega_1$ and let $g(s) = (s - \omega_1)^2 h(s)$. We have

$$g(s) = \xi_1^2 - \sum_{i=2}^{n} \frac{\xi_i^2 (s - \omega_1)^2}{(s - \omega_1 + \omega_i + \omega_1)^2} = \xi_1^2 - \sum_{i=2}^{n} \frac{\xi_i^2}{\left(1 + \frac{\omega_i + \omega_1}{s - \omega_1}\right)^2}$$

is monotonically decreasing and $g(\omega_1) = \xi_1^2 > 0$. The functions $h(s)$ and $g(s)$ has the same sign in (ω_1, ∞). It can be verified that for $s \in (\omega_1, \infty)$

$$g(s) > \xi_1^2 - \sum_{i=2} \xi_2^2 = \lim_{s\to\infty} g(s).$$

It suffices to show that $\boldsymbol{q}^{\mathrm{T}} J_n \boldsymbol{q} = \xi_1^2 - \sum_{i=2} \xi_2^2$. To this end, we have

$$\xi_1^2 - \sum_{i=2} \xi_2^2 = (V^{\mathrm{T}}\boldsymbol{q})^{\mathrm{T}} J_n (V^{\mathrm{T}}\boldsymbol{q}) = \boldsymbol{q}^{\mathrm{T}} V J_n V^{\mathrm{T}} \boldsymbol{q}.$$

It remains to show $V J_n V^{\mathrm{T}} = J_n$. Since $V^{\mathrm{T}} J_n V = J_n$, we have $V^{\mathrm{T}} = J_n V^{-1} J_n$ and thus

$$V J_n V^{\mathrm{T}} = V J_n (J_n V^{-1} J_n) = J_n,$$

as expected. □

Lemma 3.2 *If $\boldsymbol{q}^{\mathrm{T}} M^{-\mathrm{T}} J_n M^{-1} \boldsymbol{q} \geq 0$ and $\boldsymbol{q}^{\mathrm{T}} J_n \boldsymbol{q} \geq 0$, then either $\boldsymbol{q} \in \mathbb{K}^n$ or $-M^{-1}\boldsymbol{q} \in \mathbb{K}^n$.*

Proof Assume to the contrary that both $\boldsymbol{q} \notin \mathbb{K}^n$ and $-M^{-1}\boldsymbol{q} \notin \mathbb{K}^n$, i.e., both $\boldsymbol{q}_{(1)} < 0$ and $\boldsymbol{x}_{(1)} > 0$, where $\boldsymbol{x} = M^{-1}\boldsymbol{q}$. This, together with the conditions that $\boldsymbol{q}^{\mathrm{T}} M^{-\mathrm{T}} J_n M^{-1} \boldsymbol{q} \geq 0$ and $\boldsymbol{q}^{\mathrm{T}} J_n \boldsymbol{q} \geq 0$, imply

$$\boldsymbol{x}_{(1)} \geq \sqrt{\sum_{i=2}^{n} |x_{(i)}|^2}, \quad -\boldsymbol{q}_{(1)} \geq \sqrt{\sum_{i=2}^{n} |q_{(i)}|^2}.$$

Thus

$$\boldsymbol{q}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{q}_{(1)}\boldsymbol{x}_{(1)} + \sum_{i=2}^{n} \boldsymbol{q}_{(i)}\boldsymbol{x}_{(i)} \leq \boldsymbol{q}_{(1)}\boldsymbol{x}_{(1)} + \sqrt{\sum_{i=2}^{n} |x_{(i)}|^2} \sqrt{\sum_{i=2}^{n} |q_{(i)}|^2} \leq 0.$$

On the other hand, since M is symmetric positive definite, so is M^{-1} and thus $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{q}^{\mathrm{T}} M^{-1} \boldsymbol{q} > 0$, a contradiction. □

4 Special Case $\xi_1 = 0$

In this and the next section, we shall devise numerical schemes to efficiently solve $h(s) = 0$ with $h(s)$ given by (3.1).

The case $\xi_1 = 0$ corresponds to the case when $s_* = \omega_1$ in [38]. Now $h(s) < 0$ for all $s > 0$, and straightforwardly solving $h(s) = 0$ for some $s_* > 0$ is not going to work. But when $s_* = \omega_1$, $M\boldsymbol{x} + \boldsymbol{q} = s_* J_n \boldsymbol{x}$ gives, upon using (2.1),

$$(\Omega - \omega_1 J_n) V^{-1} \boldsymbol{x} = -V^{\mathrm{T}} \boldsymbol{q}. \tag{4.1}$$

Although the equation (4.1) is consistent, it only determines uniquely the last $n-1$ entries of $V^{-1}\boldsymbol{x}$ with $\beta := (V^{-1}\boldsymbol{x})_{(1)}$ completely free. In fact, we will have $V^{-1}\boldsymbol{x} = -(\Omega - \omega_1 J_n)^{\dagger} V^{\mathrm{T}} \boldsymbol{q} + \beta \boldsymbol{e}_1$, giving

$$\boldsymbol{x} = V[-(\Omega - \omega_1 J_n)^{\dagger} V^{\mathrm{T}} \boldsymbol{q} + \beta \boldsymbol{e}_1], \tag{4.2a}$$

Table 1 The sign of $h(s)$ on $(0, \infty)$ in terms of location of $\boldsymbol{q}$ [38]

Cases	Where is $\boldsymbol{q}$?	Solution x_* or $h(s)$
1	$\boldsymbol{q} \in \mathbb{K}^n$	$\boldsymbol{0} = \boldsymbol{x}_* \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$
2	$\boldsymbol{q} \in -M\mathbb{K}^n$	$-M^{-1}\boldsymbol{q} = \boldsymbol{x}_* \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$
3	$\boldsymbol{q} \in (-\mathbb{K}^n)\backslash(-M\mathbb{K}^n)$, $\boldsymbol{q} \notin \mathcal{R}(M - \omega_1 J_n)$	$h(s) = \begin{cases} - & \text{for } s \in (0, s_*), \\ 0 & \text{for } s = s_*, \\ + & \text{for } s \in (s_*, \omega_1), \\ + & \text{for } s \in (\omega_1, \infty). \end{cases}$
4	$\boldsymbol{q} \in M\mathbb{K}^n\backslash\mathbb{K}^n$, $\boldsymbol{q} \notin \mathcal{R}(M - \omega_1 J_n)$	$h(s) = \begin{cases} + & \text{for } s \in (0, \omega_1), \\ + & \text{for } s \in (\omega_1, s_*), \\ 0 & \text{for } s = s_*, \\ - & \text{for } s \in (s_*, \infty). \end{cases}$
5	$\boldsymbol{q} \notin (-M\mathbb{K}^n) \cup M\mathbb{K}^n \cup \mathbb{K}^n \cup (-\mathbb{K}^n)$, $\boldsymbol{q} \notin \mathcal{R}(M - \omega_1 J_n)$	$h(s) = \begin{cases} - & \text{for } s \in (0, s_{*;1}), \\ 0 & \text{for } s = s_{*;1}, \\ + & \text{for } s \in (s_{*;1}, \omega_1), \\ + & \text{for } s \in (\omega_1, s_{*;2}), \\ 0 & \text{for } s = s_{*;2}, \\ - & \text{for } s \in (s_{*;2}, \infty). \end{cases}$

where $(\Omega - \omega_1 J_n)^\dagger$ is the Moore-Penrose inverse of $\Omega - \omega_1 J_n$ [28, p.102]. We have to determine β to make $\boldsymbol{x} \in \partial\mathbb{K}^n$. To this end, we notice, using (2.1) and (3.2),

$$\boldsymbol{x}^{\mathrm{T}} J_n \boldsymbol{x} = -\sum_{i=2}^{n} \frac{\xi_i^2}{(\omega_i - \omega_1)^2} + \beta^2.$$

To make $\boldsymbol{x} \in \partial\mathbb{K}^n$, we need

$$\beta = \pm \sqrt{\sum_{i=2}^{n} \frac{\xi_i^2}{(\omega_i - \omega_1)^2}} \tag{4.2b}$$

such that the first component of $\boldsymbol{x}$ is positive.

5 General Case $\xi_1 \neq 0$

This corresponds to the case when $s_* \neq \omega_1$ in [38]. According to Theorem 2.2(c), $s_* \neq \omega_1$ *if and only if* $\boldsymbol{v}^{\mathrm{T}} J_n \boldsymbol{q} \neq 0$ *or, equivalently,* $\boldsymbol{q} \notin \mathcal{R}(M - \omega_1 J_n)$.

The following theorem due to [38] gives a complete picture of $h(s)$ on $(0, \infty)$ as detailed in Table 1 for the current case, where, and in what follows, "$h(s) = -$" means $h(s) < 0$ and likewise "$h(s) = +$" mean $h(s) > 0$. In particular, it shows that $h(s)$ has at least one but at most two positive zeros in $(0, \infty)$. For cases 3 and 4 in the table, $\boldsymbol{q} \notin \mathcal{R}(M - \omega_1 J_n)$ is redundantly added for clarity since it is in fact implied by $\boldsymbol{q} \in (-\mathbb{K}^n)\backslash(-M\mathbb{K}^n)$ for case 3 and by $\boldsymbol{q} \in M\mathbb{K}^n\backslash\mathbb{K}^n$ for case 4.

Theorem 5.1 ([38]) *If* $\boldsymbol{q} \notin \mathcal{R}(M - \omega_1 J_n)$ *(and thus* $s_* \neq \omega_1$*), then*

(1) *the sign of* $h(s)$ *for* $s \in (0, +\infty)$ *can be characterized by cases* 3, 4, *and* 5 *in Table 1. In cases* 3 *and* 4, $h(s)$ *has one positive root* s_**, and in case* 5*, it has two positive roots* $s_{*;1}$ *and* $s_{*;2}$ *one of which is* s_**, and*

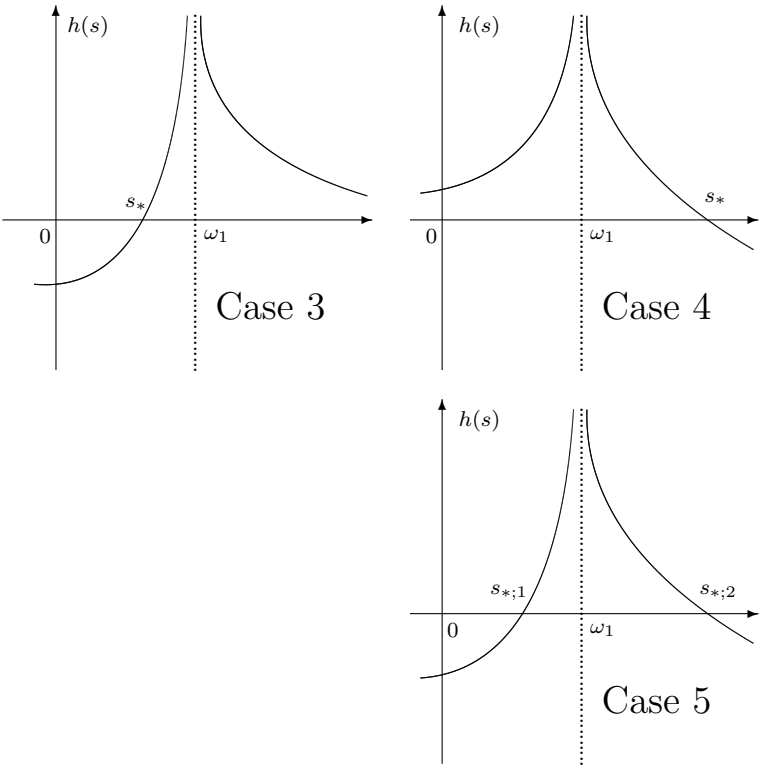

Fig. 1 $h(s)$ corresponding to cases 3, 4, and 5 in Table 1 [38]

(2) $\lim_{s\to\omega_1} h(s) = +\infty$, *which combining with Table 1, implies that* ω_1 *is the unique pole of* $h(s)$ *in* $[0, +\infty)$.

Except for the trivial cases: cases 1 and 2, Theorem 5.1 reveals a complete picture of $h(s)$, which is illustrated in Figure 1. We point out that the three patterns of $h(s)$ in Figure 1 are one-to-one corresponding to the three locations of $\boldsymbol{q}$. In actual computations, we will first check if it is in case 1 or case 2. If neither case occurs, then it must fall into one of the cases 3, 4, and 5. Now the location of $\boldsymbol{q}$ and, thereby, the sign pattern of $h(s)$ or, equivalently, the relationship between ω_1 and s_*, are then determined.

With the above discussions, we can outline the framework of our algorithm in Algorithm 5.1. For Algorithm 5.1 to be workable and efficient, the remaining task is to solve $h(s) = 0$ for its solution $s_* > 0$ for the three cases as described in Table 1 and Figure 1, where $h(s)$ is, as given by (3.1),

$$h(s) = \frac{\xi_1^2}{(s-\omega_1)^2} - \sum_{i=2}^{n} \frac{\xi_i^2}{(s+\omega_i)^2}. \tag{3.1}$$

We have two practical ways for this task. One is the Newton method, the standard method to solve any nonlinear equation such as $h(s) = 0$. The other one is specifically tailored towards $h(s) = 0$ by taking the special form of $h(s)$ into full account. The design of this latter method draws inspiration from the past work on solving secular equations arising from

Algorithm 5.1 Linear Complementarity Problem

Input: M (symmetric positive definite), $\boldsymbol{q}$;
Output: an approximation to $\boldsymbol{x} \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$.

1: **if** $\boldsymbol{q} \in \mathbb{K}^n$ **then** $\boldsymbol{x} = \boldsymbol{0}$ and **return**;
2: **if** $-M^{-1}\boldsymbol{q} \in \mathbb{K}^n$ **then** $\boldsymbol{x} = -M^{-1}\boldsymbol{q}$ and **return**;
3: compute the decompositions (2.1);
4: compute the vector $\boldsymbol{w} = V^{\mathrm{T}}\boldsymbol{q}$ as in (3.2);
5: **if** $\xi_1 = 0$ **then**
6: compute $\boldsymbol{x}$ by (4.2);
7: **else**
8: **if** $h(0) < 0$ **then**
9: find the zero s_* of $h(s)$ in $(0, \omega_1)$ and check if $\boldsymbol{x} = -V(\Omega - s_* J_n)^{-1}\boldsymbol{w} \in \partial(\mathbb{K}^n)$; if it is, **return**;
10: **end if**
11: $h(s)$ must have one and only one zero s_* in (ω_1, ∞); find the zero s_* and compute $\boldsymbol{x} = -V(\Omega - s_* J_n)^{-1}\boldsymbol{w}$ and **return**.
12: **end if**

the divide-and-conquer method for the symmetric eigenvalue problem [6,9,21]. As expected, the second method performs much better later in our numerical experiments. But before we explain both methods, we will discuss an effective initial guess strategy, similar to what was done in [21].

5.1 Initial Guess Strategy

We have two cases to worry about: either $s_* \in (0, \omega_1)$ or $s_* > \omega_1$. For convenience, write

$$f(s) = \sum_{i=2}^{n} \frac{\xi_i^2}{(s+\omega_i)^2}, \quad h(s) = \frac{\xi_1^2}{(s-\omega_1)^2} - f(s).$$

Case $s_* \in (0, \omega_1)$. We first evaluate $f(\omega_1/2)$ and then $h(\omega_1/2)$, as a by-product, which will become useful later. Then we solve

$$\frac{\xi_1^2}{(s-\omega_1)^2} - f(\omega_1/2) = 0$$

for its smaller root

$$s_0 = \omega_1 - |\xi_1| \Big[f(\omega_1/2) \Big]^{-1/2} = \omega_1 - |\xi_1| \left(\sum_{i=2}^{n} \frac{\xi_i^2}{(\omega_i + \omega_1/2)^2} \right)^{-1/2} \tag{5.1}$$

and use the root as an initial guess. We claim that $s_0 \in (0, \omega_1)$. To see this, we note that evidently $s_0 < \omega_1$. Because $s_* \in (0, \omega_1)$, we have

$$h(0) = \frac{\xi_1^2}{\omega_1^2} - \sum_{i=2}^{n} \frac{\xi_i^2}{\omega_i^2} < 0 \quad \Rightarrow \quad \left(\frac{\xi_1^2}{\omega_1^2} \right)^{-1/2} > \left(\sum_{i=2}^{n} \frac{\xi_i^2}{\omega_i^2} \right)^{-1/2}.$$

Now

$$s_0 > \omega_1 - |\xi_1| \left(\sum_{i=2}^{n} \frac{\xi_i^2}{(\omega_i)^2} \right)^{-1/2} = |\xi_1| \left[\left(\frac{\xi_1^2}{\omega_1^2} \right)^{-1/2} - \left(\sum_{i=2}^{n} \frac{\xi_i^2}{\omega_i^2} \right)^{-1/2} \right] > 0,$$

as expected. In fact, more can be said as in the next theorem. We note that $s_* \in (0, \omega_1/2)$ if $h(\omega_1/2) > 0$, and $s_* \in (\omega_1/2, \omega_1)$ if $h(\omega_1/2) < 0$. In this regard, Theorem 5.2 provides a theoretical justification as to why s_0 in (5.1) is a good initial guess.

Theorem 5.2 *Suppose $s_* \in (0, \omega_1)$ and let s_0 be given by (5.1). If $h(\omega_1/2) > 0$, then $s_0 \in (0, \omega_1/2)$; if $h(\omega_1/2) < 0$, then $s_0 \in (\omega_1/2, \omega_1)$.*

Proof We already know that $s_0 \in (0, \omega_1)$. Write $f_0 = f(\omega_1/2)$. By (5.1), we have

$$s_0 - \frac{\omega_1}{2} = \frac{\omega_1}{2} - |\xi_1| f_0^{-1/2} = |\xi_1| \left[\left(\frac{\xi_1^2}{(\omega_1/2)^2} \right)^{-1/2} - f_0^{-1/2} \right]. \tag{5.2}$$

If $h(\omega_1/2) > 0$, i.e., $\frac{\xi_1^2}{(\omega_1/2)^2} - f_0 > 0$, then $s_0 - \omega_1/2 < 0$ by (5.2). On the other hand, if $h(\omega_1/2) < 0$, i.e., $\frac{\xi_1^2}{(\omega_1/2)^2} - f_0 < 0$, then $s_0 - \omega_1/2 > 0$ by (5.2). □

A word of caution is warranted here in using the formula (5.1) in the case where s_* is very close to ω_1. First, there is a danger, i.e., possible catastrophic cancellation, in computing s_0 formulated as in (5.1). Second, in the extreme case, s_* and ω_1 may be the same in the working floating point environment. Therefore, for numerical stability, instead of computing s_*, we should compute the difference between ω_1 and s_*, say $\omega_1 - s_*$, and, accordingly, we compute

$$\omega_1 - s_0 = |\xi_1| \left(\sum_{i=2}^{n} \frac{\xi_i^2}{(\omega_i + \omega_1/2)^2} \right)^{-1/2}$$

as an initial guess to $\omega_1 - s_*$. This idea of computing the difference shares the same reasoning as in the practice of solving secular equations [21] we mentioned moments ago. As a guideline, we propose to compute s_* when $h(\omega_1/2) > 0$ for which case $s_* \in (0, \omega_1/2)$ or $\omega_1 - s_*$ when $h(\omega_1/2) < 0$ for which case $s_* \in (\omega_1/2, \omega_1)$ and thus $\omega_1 - s_* \in (0, \omega_1/2)$.

Case $s_* > \omega_1$. We solve

$$\frac{\xi_1^2}{(s - \omega_1)^2} - f(\omega_1) = 0$$

for its larger root

$$s_0 = \omega_1 + |\xi_1| \Big[f(\omega_1) \Big]^{-1/2} = \omega_1 + |\xi_1| \left(\sum_{i=2}^{n} \frac{\xi_i^2}{(\omega_1 + \omega_i)^2} \right)^{-1/2}$$

and use the root as an initial guess. Evidently $s_0 > \omega_1$.

5.2 Newton's Iteration

As the position of zero s_* and poles of $h(s)$ are now clear, we can use the generic Newton iteration

$$s_{k+1} = s_k - \frac{h(s_k)}{h'(s_k)} \tag{5.3}$$

to find s_*. Unfortunately, we don't know *a priori* if $s_{k+1} \in (0, \omega_1)$ in the case of $s_* \in (0, \omega_1)$ or if $s_{k+1} > \omega_1$ in the case of $s_* > \omega_1$. This could cause problems numerically, but it can be easily overcame by a bisection safeguard strategy, using lower and upper bounds of s_* that are iteratively refined during the iteration process. The basic idea is as follows. Initially

during the computations of the initial guess in Sect. 5.1, we determine an interval (α, β) that contains s_* as tightly as we can get at the moment. Consider now $\tau = s_k \in (\alpha, \beta)$. We first update the interval according to

$$\begin{aligned}&\textbf{case } s_* \in (0, \omega_1): \quad \beta \leftarrow \tau \text{ if } h(\tau) > 0; \alpha \leftarrow \tau \text{ if } h(\tau) < 0;\\ &\textbf{case } s_* > \omega_1: \quad \beta \leftarrow \tau \text{ if } h(\tau) < 0; \alpha \leftarrow \tau \text{ if } h(\tau) > 0.\end{aligned}$$

After s_{k+1} in (5.3) is computed, we take it as the next approximation if $s_{k+1} \in (\alpha, \beta)$ or $(\alpha + \beta)/2$ otherwise. Repeat the process until $|h(s_k)| \le \epsilon_h$, a prescribed tolerance, which is set to be 10^{-12} in our numerical testing.

5.3 Method by Simple Rational Approximation

A more natural way for solving $h(s) = 0$ is through approximate $h(s)$ by rational approximations since $h(s)$ is rational. To simplify notation, we write $\tau = s_k \approx s_*$ and $\tau_1 = s_{k+1}$. The Newton iteration is the result of approximating $h(s)$ by its tangent line at $s = \tau$. But given the special form, we propose to approximate $h(s)$ by

$$g(s) = \frac{a}{(s - \omega_1)^2} - \frac{b}{(s + \omega_2)^2} \tag{5.4}$$

satisfying $g(\tau) = h(\tau)$ and $g'(\tau) = h'(\tau)$. One particular argument for using such a rational function is that $g(s)$ has poles at $-\omega_2$ and ω_1, arguably the most important ones of $h(s)$ for the part of $h(s)$ we are interested in. Note also $g(s)$ and $h(s)$ match at the infinities.

We have to compute a and b. For that, we resort to $g(\tau) = h(\tau)$ and $g'(\tau) = h'(\tau)$:

$$\frac{a}{(\tau - \omega_1)^2} - \frac{b}{(\tau + \omega_2)^2} = h(\tau), \quad \frac{-2a}{(\tau - \omega_1)^3} - \frac{-2b}{(\tau + \omega_2)^3} = h'(\tau),$$

yielding

$$\begin{aligned}a &= -\frac{(\tau - \omega_1)^3(\tau + \omega_2)}{\omega_1 + \omega_2}\left(\frac{h(\tau)}{\tau + \omega_2} + \frac{h'(\tau)}{2}\right),\\ b &= -\frac{(\tau - \omega_1)(\tau + \omega_2)^3}{\omega_1 + \omega_2}\left(\frac{h(\tau)}{\tau - \omega_1} + \frac{h'(\tau)}{2}\right).\end{aligned} \tag{5.5}$$

Once a and b are computed, we solve $g(s) = 0$ for its root in $(0, \omega_1)$ or to the right of ω_1, depending whether $s_* \in (0, \omega_1)$ or $s_* > \omega_1$, as the next approximation to s_*.

The question is whether $g(s) = 0$ has the desired root. The answer depends on the signs of a and b. Ideally, we would expect both $a > 0$ and $b > 0$. Plugging in $h(\tau)$ and $h'(\tau)$, we find

$$\begin{aligned}a &= \frac{(\omega_1 - \tau)^3}{\omega_1 + \omega_2}\left[\frac{(\omega_1 + \omega_2)\xi_1^2}{(\omega_1 - \tau)^3} - \sum_{i=2}^{n}\frac{(\omega_i - \omega_2)\xi_i^2}{(\tau + \omega_2)^3}\right],\\ b &= \frac{(\tau + \omega_2)^3}{\omega_1 + \omega_2}\sum_{i=2}^{n}\frac{(\omega_1 + \omega_i)\xi_i^2}{(\tau + \omega_2)^3} > 0.\end{aligned} \tag{5.6}$$

So $b > 0$ for $\tau \in (0, \infty)$ and $a > 0$ for $\tau \in (\omega_1, \infty)$, but it is not clear if $a > 0$ for $\tau \in (0, \omega_1)$ as well because within the big bracket in the right-hand side of (5.6) it is the difference between two positive numbers when $\tau \in (0, \omega_1)$ (unless $n = 2$). In the case when $h(\tau) > 0$, we do have $a > 0$ by (5.5), because

$$h'(\tau) = -\frac{\xi_1^2}{(\tau-\omega_1)^3} + \sum_{i=2}^{n} \frac{2\xi_i^2}{(\tau+\omega_i)^3} > 0 \quad \text{for } \tau \in (0, \omega_1).$$

Obviously, possibly $a < 0$ is disappointing theoretically, but numerically it may not matter much and conceivably happens extremely rarely. When it does happen, we can employ the same safeguard strategy as explained for the Newton iteration. In what follows, we present a formula for the next approximation $\tau_1 \approx s_*$, assuming $a > 0$, and then comment on what we will do if $a < 0$, among other undesirable situations.

Assume $a > 0$ (note $b > 0$ always). In case of $s_* \in (0, \omega_1)$, we seek the smaller root τ_1 of $g(s) = 0$. Thus we have

$$\frac{\omega_1 - \tau_1}{\tau_1 + \omega_2} = \sqrt{\frac{a}{b}}. \tag{5.7}$$

It is usually a good idea in an iterative algorithm to compute the difference between the current approximation and the next one because often the difference is smaller in magnitude than the approximation and consequently adding the difference to the approximation yields the next one that is more accurate. Taking this into consideration, we let $\delta = \tau_1 - \tau$ and rewrite (5.7) as

$$\frac{\delta + (\tau - \omega_1)}{\delta + (\tau + \omega_2)} = -\sqrt{\frac{a}{b}},$$

yielding

$$\tau_1 = \tau + \delta \quad \text{with} \quad \delta = \frac{-(\tau-\omega_1) - (\tau+\omega_2)\sqrt{a/b}}{1 + \sqrt{a/b}}. \tag{5.8}$$

In case of $s_* > \omega_1$, we seek the bigger root τ_1 of $g(s) = 0$. Thus we have

$$\frac{\tau_1 - \omega_1}{\tau_1 + \omega_2} = \sqrt{\frac{a}{b}} \quad \Rightarrow \quad \frac{\delta + (\tau - \omega_1)}{\delta + (\tau + \omega_2)} = \sqrt{\frac{a}{b}},$$

yielding

$$\tau_1 = \tau + \delta \quad \text{with} \quad \delta = \frac{-(\tau-\omega_1) + (\tau+\omega_2)\sqrt{a/b}}{1 - \sqrt{a/b}}. \tag{5.9}$$

Now that we have the formulas for τ_1 in (5.8) and (5.9), what we don't know *a priori* if $\omega_1 - \tau_1 \in (0, \omega_1)$ in the case of (5.8), or if $\tau_1 - \omega_1 > 0$ in the case of (5.9). This again could cause problems numerically, but it can be easily overcame by the same safeguard strategy as explained for the Newton iteration.

Part of the reason why we pick $g(s)$ as in (5.4) is because then $g(s) = 0$ is easy to solve, while it keeps two most important poles of $h(s)$ with respect to its zeros in $(0, \omega_1)$ and/or (ω_1, ∞). There are other possible forms equally effective for approximating $h(s)$ for the purpose of computing the zeros, e.g.,

$$\frac{\xi_1^2}{(s-\omega_1)^2} - \frac{b}{(s+\omega_2)^2} + c, \quad \frac{a}{(s-\omega_1)^2} - \frac{b}{(s+\omega_2)^2} + c.$$

The last one can even be made to match $h(s)$ up to the second derivative at a point. The only downside is that we have to use another iterative method, e.g., the Newton method, to compute the zeros of these approximating rational functions.

6 Applications of Algorithm 5.1

We now provide two applications where our Algorithm 5.1 for (1.1) can be extended or incorporated into a larger framework to solve more general problems. The first application of SOCLCP (1.1) is just an extension of the single second-order cone $\mathbb{K}^n$ to a Cartesian product of multiple second-order cones, and our second application is to solve the quadratic programming over the second-order cones (QP-SOC).

6.1 SOCLCP over Multiple Second-Order Cones

Within the matrix-splitting framework [17,24,38], we can apply our Algorithm 5.1 to solve the general SOCLCP (1.3) over $\mathbb{K}_{\times m} := \mathbb{K}^{n_1} \times \mathbb{K}^{n_2} \times \cdots \times \mathbb{K}^{n_m}$ where $\sum_{i=1}^m n_i = n$. The matrix-splitting method for solving (1.3) starts by splitting M into $M = B + C$ so that $B - C$ is positive definite[1] (but not necessarily symmetric) in the sense that $\boldsymbol{x}^{\mathrm{T}}(B - C)\boldsymbol{x} > 0$ for any $0 \neq \boldsymbol{x} \in \mathbb{R}^n$. Such a splitting (B, C) of M is said *regular* [24]. The convergence of the matrix-splitting method by a regular splitting is guaranteed [37, section 3]. To demonstrate an application of Algorithm 5.1 for solving (1.3) by the matrix-splitting method, we let

$$M = \begin{array}{c} \begin{matrix} \scriptstyle n_1 & \scriptstyle n_2 & \scriptstyle \cdots & \scriptstyle n_m \end{matrix} \\ \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1m} \\ M_{21} & M_{22} & \cdots & M_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{bmatrix} \end{array} \begin{array}{l} \scriptstyle n_1 \\ \scriptstyle n_2 \\ \scriptstyle \vdots \\ \scriptstyle n_m \end{array}, \; B = \begin{bmatrix} M_{11} & & & \\ M_{21} & M_{22} & & \\ \vdots & \vdots & \ddots & \\ M_{m1} & M_{m2} & \cdots & M_{mm} \end{bmatrix}, \tag{6.1}$$

and then $C = M - B$. They are partitioned according to $\mathbb{K}_{\times m}$. It can be verified that

$$(B - C) + (B - C)^{\mathrm{T}} = 2\,\mathrm{diag}(M_{11}, M_{22}, \ldots, M_{mm}),$$

which is positive definite and thus the splitting (6.1) is regular. We call the matrix-splitting method [37, Algorithm 1] with this splitting the *block SOR method* (BSOR).

Partition $\boldsymbol{x} = [\boldsymbol{x}_1^{\mathrm{T}}, \ldots, \boldsymbol{x}_m^{\mathrm{T}}]^{\mathrm{T}}$ with $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$. It can be seen that

$$\boldsymbol{x} \in \mathbb{K}_{\times m},\ \boldsymbol{y} \in \mathbb{K}_{\times m} \text{ and } \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = 0 \Longleftrightarrow \boldsymbol{x}_i \in \mathbb{K}^{n_i},\ \boldsymbol{y}_i \in \mathbb{K}^{n_i} \text{ and } \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{y}_i = 0$$

for $i = 1, 2, \ldots, m$. Let $\boldsymbol{x}^{(\ell)}$ be the ℓth iterative approximation by the matrix-splitting method [37, Algorithm 1]. The next iterative approximation $\boldsymbol{x}^{(\ell+1)} \in \mathbb{K}_{\times m}$ is obtained via solving sequentially the following subproblems:

$$\boldsymbol{x}_i \in \mathbb{K}^{n_i},\ M_{ii}\boldsymbol{x}_i + \mathbf{t}_i^{(\ell)} \in \mathbb{K}^{n_i},\ \boldsymbol{x}_i^{\mathrm{T}}(M_{ii}\boldsymbol{x}_i + \boldsymbol{t}_i^{(\ell)}) = 0 \tag{6.2}$$

for $i = 1, 2, \ldots, m$, where

$$\boldsymbol{t}_i^{(\ell)} := \begin{cases} \boldsymbol{q}_1^{(\ell)}, & \text{if } i = 1, \\ \sum_{j=1}^{i-1} M_{ij}\boldsymbol{x}_j^{(\ell+1)} + \boldsymbol{q}_i^{(\ell)}, & \text{if } i > 1, \end{cases}$$

and $\boldsymbol{q}^{(\ell)} := \boldsymbol{q} + C\boldsymbol{x}^{(\ell)} = [(\boldsymbol{q}_1^{(\ell)})^{\mathrm{T}}, \ldots, (\boldsymbol{q}_m^{(\ell)})^{\mathrm{T}}]^{\mathrm{T}}$. Using Algorithm 5.1 for the subproblem (6.2), in Sect. 7, we will present numerical experiments of BSOR for (1.3).

[1] Equivalently, $(B - C) + (B - C)^{\mathrm{T}}$ is symmetric positive definite.

6.2 QP-SOC

We next consider the following convex quadratic programming over $\mathbb{K}_{\times m}$:

$$\min_{A\boldsymbol{x}=\boldsymbol{b},\ \boldsymbol{x}\in\mathbb{K}_{\times m}} \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}W\boldsymbol{x}+\boldsymbol{x}^{\mathrm{T}}\boldsymbol{c} \tag{6.3}$$

where $A\in\mathbb{R}^{k\times n}$, $\boldsymbol{b}\in\mathbb{R}^k$ and $W\in\mathbb{R}^{n\times n}$ is symmetric positive definite. (6.3) is a generalization of the usual second-order cone programming which corresponds to $W=0$ and has been widely-used in many real-world applications (see e.g., [1,12,22]). The well-known augmented Lagrangian method (ALM) proposed by Hestenes [18] and Powell [27] (see also [4, Chapter 2] and [25, Chapter 17.3]) has been applied in [38] to solve (6.3). We now outline the method below.

Define the *augmented Lagrangian function*

$$\begin{aligned}\mathscr{L}(\boldsymbol{x};\lambda,\mu) &:= \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}W\boldsymbol{x}+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}-\lambda^{\mathrm{T}}(A\boldsymbol{x}-\boldsymbol{b})+\frac{\mu}{2}\|A\boldsymbol{x}-\boldsymbol{b}\|_2^2\\ &= \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}(W+\mu A^{\mathrm{T}}A)\boldsymbol{x}+\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{c}-A^{\mathrm{T}}\lambda-\mu A^{\mathrm{T}}\boldsymbol{b})+\frac{\mu}{2}\|\boldsymbol{b}\|_2^2+\boldsymbol{b}^{\mathrm{T}}\lambda,\end{aligned}$$

where $\mu>0$ is a penalty parameter. Then, given $\mu_0>0$ and λ_0, the basic steps of ALM can be summarized as in Algorithm 6.1 (see e.g., [25, Framework 17.3] and [4, Chapter 2]).

The convergence analysis for ALM has been well-established and the reader is referred to, e.g., [4, Poposition 2.1-2.3] and [25, Chapter 17.3] for more details. It can be seen that the main computational task in this framework is to compute $\boldsymbol{x}^{(\ell)}$ at Line 2, i.e.,

$$\boldsymbol{x}^{(\ell)}=\arg\min_{\boldsymbol{x}\in\mathbb{K}_{\times m}}\left\{\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}(W+\mu_\ell A^{\mathrm{T}}A)\boldsymbol{x}+\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{c}-A^{\mathrm{T}}\lambda_\ell-\mu_\ell A^{\mathrm{T}}\boldsymbol{b})\right\}, \tag{6.4}$$

which, by the convexity, has a unique solution (see e.g., [37]). It in fact gives rise to a SOCLCP (1.3) with $M=W+\mu_\ell A^{\mathrm{T}}A$ and $\boldsymbol{q}=\boldsymbol{c}-A^{\mathrm{T}}\lambda_\ell-\mu_\ell A^{\mathrm{T}}\boldsymbol{b}$. In our numerical testing in Sect. 7, we will focus on a particular QP-SOC which arises from testing the Lorentz-copositivity of a given matrix W. Specifically, W is Lorentz-copositive (see [19] and also [23, Definition 2.1]) on $\mathbb{K}_{\times m}$ if and only if $\boldsymbol{x}^{\mathrm{T}}W\boldsymbol{x}\geq 0$ for all $\boldsymbol{x}^{\mathrm{T}}=[\boldsymbol{x}_1^{\mathrm{T}},\ldots,\boldsymbol{x}_m^{\mathrm{T}}]^{\mathrm{T}}\in\mathbb{K}_{\times m}$ with $\mathbf{0}\neq\boldsymbol{x}_i\in\mathbb{K}^{n_i}$. Numerical methods based on Krylov subspaces have been developed in [35] for particularly $\mathbb{K}_{\times m}=\mathbb{K}^n$, i.e., $m=1$. In Sect. 7, we will use ALM with our Algorithm 5.1 embedded for subproblem (6.4) to solve the QP-SOC:

$$\min_{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}=1,\ \boldsymbol{x}\in\mathbb{K}_{\times m}} \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}W\boldsymbol{x} \tag{6.5}$$

where $\boldsymbol{a}^{\mathrm{T}}=[\underbrace{1,0,\ldots,0}_{n_1},\underbrace{1,0,\ldots,0}_{n_2},\ldots,\underbrace{1,0,\ldots,0}_{n_m}]$. Equation $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}=1$ is a type of normalization constraint [10] for $\boldsymbol{x}$.

7 Numerical Results

In this section, we will carry out numerical tests of Algorithm 5.1 for SOCLCP (1.1), and also on its applications to general SOCLCP (1.3) as well as QP-SOC (6.3). We conduct these numerical experiments in MATLAB R2017a with 1.8GHz Intel Core i7, 8GB memory and

Algorithm 6.1 The Augmented Lagrangian Method (ALM) for (6.3)

Input: $W \in \mathbb{R}^{n\times n}$ (symmetric positive definite), $A \in \mathbb{R}^{k\times n}, \boldsymbol{b} \in \mathbb{R}^k, \boldsymbol{c} \in \mathbb{R}^n$;
Output: an approximate minimizer to (6.3).

1: **for** $\ell = 1, \ldots,$ until convergence **do**
2: find $\boldsymbol{x}^{(\ell)} = \arg\min_{\boldsymbol{x}\in\mathbb{K}^n} \mathscr{L}(\boldsymbol{x}; \lambda_\ell, \mu_\ell)$;
3: $\lambda_{\ell+1} = \lambda_\ell - \mu_\ell(A\boldsymbol{x}^{(\ell)} - \boldsymbol{b})$;
4: choose new penalty parameter $\mu_{\ell+1} = \nu\mu_\ell$, where $\nu > 1$ is a pre-selected factor;
5: **end for**
6: **return** the last $\boldsymbol{x}^{(\ell)}$ as an approximate minimizer to (6.3).

the 64bit Windows 10 system. The double precision arithmetic is used. For the efficient implementation of Algorithm 5.1, we employ the mex version[2] of LAPACK's subroutine `dsyevd`[3] [2] in MATLAB for computing all eigenvalues and eigenvectors of the real symmetric matrix M at Line 3 of Algorithm 5.1.

In our implementation, for a given vector $\boldsymbol{x}$, numerically we regard $\boldsymbol{x} \in \mathbb{K}^n$ if

$$\boldsymbol{x}_{(1)} - \|\boldsymbol{x}_{(2:n)}\|_2 \geq 10^{-6}\|\boldsymbol{x}\|_2,$$

and $\boldsymbol{x} \in \partial(\mathbb{K}^n)$ if

$$\boldsymbol{x}_{(1)} > 0 \text{ and } |\boldsymbol{x}_{(1)} - \|\boldsymbol{x}_{(2:n)}\|_2| \leq 10^{-6}\|\boldsymbol{x}\|_2.$$

Finally, in order to measure the accuracy of a computed solution $\boldsymbol{x} \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$, we define the following three relative errors

$$\chi_{\mathrm{rel}_1} = \frac{\max\{\|\boldsymbol{x}_{(2:n)}\|_2 - x_1, 0\}}{\|\boldsymbol{x}\|_2}, \tag{7.1a}$$

$$\chi_{\mathrm{rel}_2} = \frac{\max\{\|\boldsymbol{g}_{(2:n)}\|_2 - g_1, 0\}}{\|M\|_1\|\boldsymbol{x}\|_2 + \|\boldsymbol{q}\|_2}, \tag{7.1b}$$

$$\chi_{\mathrm{rel}_3} = \frac{|\boldsymbol{x}^{\mathrm{T}}\boldsymbol{g}|}{\|\boldsymbol{x}\|_2(\|M\|_1\|\boldsymbol{x}\|_2 + \|\boldsymbol{q}\|_2)}, \tag{7.1c}$$

and the total relative error

$$\chi_{\mathrm{rel}} = \chi_{\mathrm{rel}_1} + \chi_{\mathrm{rel}_2} + \chi_{\mathrm{rel}_3}, \tag{7.2}$$

where $\boldsymbol{g} = M\boldsymbol{x} + \boldsymbol{q}$. Our rationale in deciding the normalizing factors in (7.1) is fully explained in [38, section 8].

7.1 Numerical Results for SOCLCP (1.1) over $\mathbb{K}^n$

In this subsection, we first present numerical results to illustrate the efficiency of Algorithm 5.1 for solving SOCLCP (1.1). We use matrices M generated randomly by $M =$ `randn(n)`, $M = M^{\mathrm{T}}M$ and $\boldsymbol{q} =$ `randn(n,1)` $\in \mathbb{R}^n$, with the sizes varying from $n = 500$ to 5000. We tested the following five methods:

[2] `mexeig` is available at: www.math.nus.edu.sg/~matsundf/.

[3] LAPACK's subroutine `dsyevd` computes all eigenvalues, and optionally, eigenvectors of a real symmetric matrix. It uses the divide-and-conquer algorithm when eigenvectors are desired. Through its MATLAB interface via `mexeig`, it computes the eigendecomposition of a symmetric matrix much faster than MATLAB's `eig`.

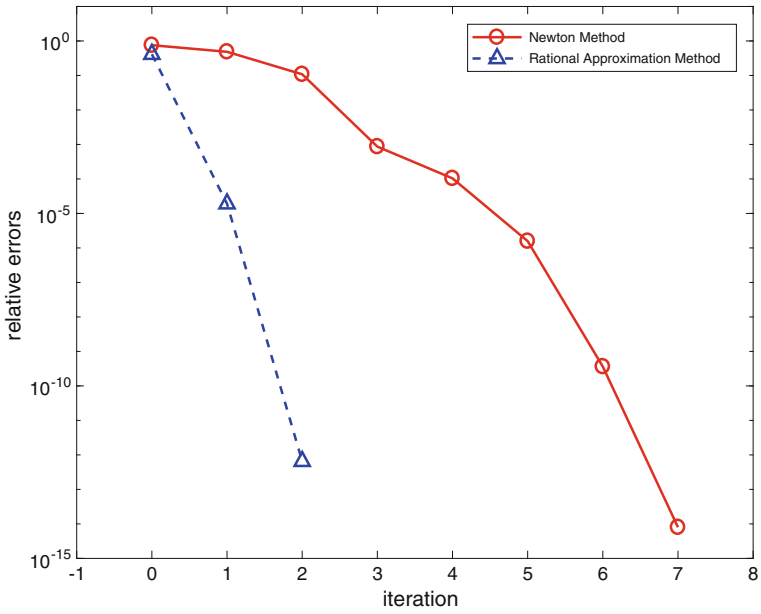

Fig. 2 $\chi_{\rm rel}$ *vs.* iteration step for a test problem $h(s) = 0$ with $n = 1000$

- `PsdLcpN`: Algorithm 5.1 with Newton's iteration for solving $h(s) = 0$;
- `PsdLcp`: Algorithm 5.1 with the new zero-finder in Sect. 5.3 for solving $h(s) = 0$;
- `BN` [36]: the bisection-Newton method with the upper Hessenberg reduction preprocessing;
- `LCPvA` [38]: linear complementarity problem via the Arnold process;
- `CVX` (version 2.1)[4]: a modeling system for constructing and solving various convex programmings, including linear/quadratic programmings, second-order cone programmings, and semidefinite programmings. It can handle both the general SOCLCP (1.3) as well as QP-SOC (6.3) by solver SDPT3 [30,31] or Sedumi [29].

For the last three methods, their default settings are used, while for `PsdLcpN` and `PsdLcp`, we terminate the iterations if $|h(s_k)| \le \epsilon_h = 10^{-12}$. It is observed that `PsdLcp` uses much fewer numbers of iterations for finding the zero of $h(s)$ than `PsdLcpN`, demonstrating the power of the simple rational approximation (5.4). As a typical example, in Figure 2, we plot the relative errors $\chi_{\rm rel}$ during an iterative process for solving $h(s) = 0$ (with $n = 1000$). It clearly demonstrates the fast convergence of `PsdLcp` over `PsdLcpN`.

In Table 2, we report numerical results averaged over 10 random tests for the four methods other than `PsdLcpN`. It shows that in terms of CPU time and the relative error, `PsdLcp` is more efficient than the other three methods when the dimension is modest. In the next subsection, we will perform numerical experiments on general SOCLCP (1.3), where `PsdLcp` serves as a key computational kernel for the efficiency of BSOR with the splitting (6.1).

[4] http://cvxr.com/cvx.

Table 2 A comparison of the algorithms for SOCLCP (1.1)

n	Cond(M)		PsdLcp	BN	LCPvA	CVX
500	1.3×10^8	CPU(s)	0.04	0.06	0.08	1.99
		χ_{rel}	1.1×10^{-9}	4.7×10^{-13}	6.8×10^{-13}	3.3×10^{-7}
		# iter	2.2	–	–	16.3
1000	4.2×10^7	CPU(s)	0.20	0.28	0.33	12.01
		χ_{rel}	5.4×10^{-11}	1.3×10^{-11}	1.7×10^{-12}	1.9×10^{-7}
		# iter	2.4	–	–	15.3
2000	4.3×10^9	CPU(s)	1.57	2.70	4.47	106.29
		χ_{rel}	9.4×10^{-11}	2.4×10^{-11}	1.8×10^{-10}	4.1×10^{-7}
		# iter	2.0	–	–	16.0
3000	2.6×10^8	CPU(s)	5.74	8.91	7.31	366.67
		χ_{rel}	6.6×10^{-12}	8.0×10^{-12}	4.3×10^{-11}	2.6×10^{-7}
		# iter	2.4	–	–	16.2
4000	2.3×10^{10}	CPU(s)	13.71	20.37	16.84	883.54
		χ_{rel}	1.9×10^{-9}	2.4×10^{-11}	2.5×10^{-10}	1.9×10^{-5}
		# iter	1.9	–	–	16.4
5000	3.8×10^9	CPU(s)	26.40	38.92	28.60	1811.53
		χ_{rel}	4.3×10^{-11}	2.2×10^{-12}	3.8×10^{-11}	1.6×10^{-6}
		# iter	7.9	–	–	17.2

7.2 Numerical Results for SOCLCP over $\mathbb{K}_{\times m}$

Within the framework of BSOR described in Sect. 6.1, we incorporate a particular solver (i.e., `BN`, `LCPvA`, or `PsdLcp`) for subproblem (6.2), and denote the resulting algorithms as `BSOR_BN`, `BSOR_LCPvA` or `BSOR_PsdLcp`, respectively. For the sake of consistency, we terminate iterations in all three methods whenever the corresponding total relative error χ_{rel} associated with (1.3) is less than or equal to 10^{-7}. In Table 3, we report the average numerical results for several different $\mathbb{K}_{\times m}$ with randomly generated symmetric positive matrices M in Sect. 7.1 over 10 random test cases. The numerical results for these SOCLCP (1.3) demonstrate that the workhorse `PsdLcp` for solving subproblem (6.2) improves the performance of the matrix-splitting method significantly, especially when the number m of the Cartesian product of second-order cones gets large. It also indicates that `BSOR_PsdLcp` is far more efficient than `CVX` for these dense random test problems.

7.3 Numerical Results for Lorentz Copositivity (6.5)

As the last numerical demonstration of our proposed method, we will test the particular model (6.5) of QP-SOC (6.3), which is related to the Lorentz copositivity [35] on a general second-order cone $\mathbb{K}_{\times m}$. After solving (6.5), we can claim that W is Lorentz copositive if the minimum is nonnegative. We implement ALM (Algorithm 6.1) by applying a specific solver (`BSOR_PsdLcp`, `BSOR_BN`, `BSOR_LCPvA` or `CVX`) at Line 2. With initially $\lambda_0 = 0.1$, $\mu_0 = 10$ and setting $\nu = 1.5$, ALM iterations are terminated whenever $k > 50$ or $\epsilon = \chi_{\text{rel},k} + \|A\boldsymbol{x}_k - \boldsymbol{b}\|_1 \leq 10^{-6}$, where $\chi_{\text{rel},k}$ represents the total relative error (7.2) of the

Table 3 A comparison of the algorithms for SOCLCP (1.3)

	n	2000			4000			8000		
	m	5	10	100	5	10	100	50	100	200
CPU(s)	BSOR_PsdLcp	3.98	1.82	1.94	17.90	9.74	3.98	10.45	9.74	13.13
	BSOR_BN	5.63	2.74	6.48	26.00	14.47	9.67	13.88	14.71	22.90
	BSOR_LCPvA	7.67	3.47	3.91	30.54	17.57	9.45	18.00	22.03	23.58
	CVX	95.99	101.19	109.80	905.51	823.00	892.89	7489.22	7762.40	8058.64
# iter	BSOR_PsdLcp	31.2	34.6	39.7	29.4	31.7	38.8	36.7	38.1	39.1
	BSOR_BN	30.6	33.6	41.7	29.8	32.9	38.1	35.4	36.9	38.3
	BSOR_LCPvA	30.3	33.5	39.9	28.9	31.6	37.7	35.8	36.9	37.8
	CVX	15.1	16.0	16.1	17.6	15.9	16.7	18.1	18.7	19.1
χrel	BSOR_PsdLcp	2.1×10^{-8}	1.6×10^{-8}	5.5×10^{-9}	2.1×10^{-8}	1.5×10^{-8}	5.9×10^{-9}	7.9×10^{-9}	5.6×10^{-9}	3.9×10^{-9}
	BSOR_BN	1.9×10^{-8}	1.5×10^{-8}	5.7×10^{-9}	1.9×10^{-8}	1.5×10^{-8}	5.7×10^{-9}	7.7×10^{-9}	5.8×10^{-9}	4.2×10^{-9}
	BSOR_LCPvA	2.0×10^{-8}	1.6×10^{-8}	5.6×10^{-9}	2.0×10^{-8}	1.5×10^{-8}	5.5×10^{-9}	7.4×10^{-9}	6.1×10^{-9}	4.0×10^{-9}
	CVX	7.1×10^{-7}	7.7×10^{-7}	3.3×10^{-6}	3.0×10^{-6}	5.7×10^{-8}	3.1×10^{-7}	4.3×10^{-7}	2.9×10^{-8}	7.8×10^{-8}

Table 4 A comparison of the algorithms for the Lorentz copositivity (6.5)

	m	50			200			500		
	solver	BSOR_PsdLcp	BSOR_BN	CVX	BSOR_PsdLcp	BSOR_BN	CVX	BSOR_PsdLcp	BSOR_BN	CVX
bcsstk17	CPU(s)	3.74	3.98	28.02	4.28	4.61	33.74	4.94	5.67	53.03
	# iter	2	2	19	2	2	21	2	2	23
	ϵ	5.1×10^{-11}	5.1×10^{-11}	1.1×10^{-8}	5.1×10^{-11}	5.1×10^{-11}	1.4×10^{-8}	5.1×10^{-11}	5.1×10^{-11}	1.1×10^{-8}
bcsstk18	CPU(s)	4.90	4.76	29.71	7.57	6.27	30.66	6.19	7.09	40.67
	# iter	2	2	20	2	2	19	2	2	20
	ϵ	8.7×10^{-9}	8.7×10^{-9}	9.6×10^{-9}	9.3×10^{-9}	9.3×10^{-9}	1.4×10^{-8}	1.9×10^{-8}	1.9×10^{-8}	1.3×10^{-8}
bcsstm25	CPU(s)	14.56	9.51	6.78	10.72	14.63	7.07	11.31	21.94	11.10
	# iter	2	2	18	2	2	19	2	2	19
	ϵ	2.3×10^{-7}	2.3×10^{-7}	1.3×10^{-8}	2.3×10^{-7}	2.3×10^{-7}	1.3×10^{-8}	2.3×10^{-7}	2.3×10^{-7}	1.0×10^{-8}

SOCLCP associated with (6.4) at the kth iteration. To examine the accuracy of computed solutions by CVX, we report ϵ = cvx_slvtol of CVX, which is the relative duality gap from SDPT3 [30].

The testing matrices are bcsstk17, bcsstk18, and bcsstm25 from the Harwell-Boeing collection, whose dimensions are $n = 10974,\ 11948$ and 15439, respectively. They are all symmetric positive definite and the ones with dimension larger than 10000 among those that are tested in [5]. To construct $\mathbb{K}_{\times m}$, we set $n_i = \lfloor n/m \rfloor$ (i.e., the largest integer that is no larger than n/m) for $1 \le i \le m-1$ and $n_m = n - \sum_{i=1}^{m-1} n_i$. The numerical results are reported in Table 4, where results by BSOR_LCPvA are not presented due to its failure in some cases. It is observed that BSOR_PsdLcp generally performs better for matrices bcsstk17, bcsstk18. For bcsstm25, CVX converges fastest but BSOR_PsdLcp becomes more competitive as m grows (at the same time, the size of the subproblem gets smaller).

8 Concluding Remarks

In this paper, we have focused on a class of SOCLCP with the GUS property. By relying on the modern powerful eigendecomposition of $M - \lambda J_n$ and the intrinsic properties of SOCLCP, we have developed a new algorithm for solving the symmetric positive definite SOCLCP. Because our iterative solution for solving $h(s) = 0$ by the simple rational approximation is so fast and guaranteed to have high quality, the new algorithm can be safely regarded as a direct method rather than an iterative one, just as nowadays the QR algorithm for the eigenvalue problem, though iterative in nature, is regarded as a direct method for all practical purposes.

The new method in general has better performance than the recent sophisticated iterative solvers for modest size problems, and thus can serve as core computational engines for iterative methods for general SOCLCP (1.3) as well as QP-SOC (6.3), whose numerical performance rely heavily on the efficiency and robustness of solving many small-to-medium sized SOCLCP (1.1).

Acknowledgements The authors are grateful to two anonymous referees for their helpful comments and suggestions that improve the presentation. Wang is supported in part by the National Natural Science Foundation of China: NSFC-11461046, NSF of Jiangxi Province: 20161ACB21005 and 20181ACB20001, Zhang is supported in part by the National Natural Science Foundation of China: NSFC-11671246 and NSFC-91730303, and R.-C. Li is supported in part by NSF Grants: CCF-1527104 and DMS-1719620.

References

1. Alizadeh, F., Goldfarb, D.: Second-order cone programming. Math. Progr. **95**, 3–51 (2003)
2. Anderson, E., Bai, Z., Bischof, C., Demmel, J.W., Dongarra, J., Croz, J.D., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., Sorensen, D.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
3. Auslender, A.: Variational inequalities over the cone of semidefinite positive symmetric matrices and over the Lorentz cone. Opt. Methods Soft. **18**, 359–376 (2003)
4. Bertsekas, D.P.: Constrained Optimization and Lagrange Multiplier Methods. Academic Press, Cambridge (1982)
5. Brás, C.P., Fischerb, A., Júdice, J.J., Schönefeldb, K., Seifert, S.: A block active set algorithm with spectral choice line search for the symmetric eigenvalue complementarity problem. Appl. Math. Comput. **294**, 36–48 (2017)
6. Bunch, J.R., Nielsen, C.P., Sorensen, D.C.: Rank-one modification of the symmetric eigenproblem. Numer. Math. **31**, 31–48 (1978)

7. Chen, J.S., Tseng, P.: An unconstrained smooth minimization reformulation of the second-order cone complementarity problem. Math. Progr. **104**, 293–327 (2005)
8. Chen, X.D., Sun, D.F., Sun, J.: Complementarity functions and numerical experiments on some smoothing Newton methods for second-order-cone complementarity problems. Comput. Optim. Appl. **25**, 39–56 (2003)
9. Cuppen, J.J.M.: A divide and conquer method for the symmetric eigenproblem. Numer. Math. **36**, 177–195 (1981)
10. Fernandes, L.M., Fukushima, M., Júdice, J.J., Sherali, H.D.: The second-order cone eigenvalue complementarity problem. Opt. Methods Soft. **31**, 24–52 (2016)
11. Fukushima, M., Luo, Z.Q., Tseng, P.: Smoothing functions for second-order-cone complementarity problems. SIAM J. Optim. **12**, 436–460 (2002)
12. Goldfarb, D., Scheinberg, K.: Product-form cholesky factorization in interior point methods for second-order cone programming. Math. Progr. **103**, 153–179 (2005)
13. Gowda, M.S., Sznajder, R.: Automorphism invariance of P- and GUS-properties of linear transformations on Euclidean Jordan algebras. Math. Oper. Res. **31**, 109–123 (2006)
14. Gowda, M.S., Sznajder, R.: Some global uniqueness and solvability results for linear complementarity problems over symmetric cones. SIAM J. Optim. **18**, 461–481 (2007)
15. Gowda, M.S., Sznajder, R., Tao, J.: Some P-properties for linear transformations on Euclidean Jordan algebras. Linear Algebra Appl. **393**, 203–232 (2004)
16. Hayashi, S., Yamashita, N., Fukushima, M.: A combined smoothing and regularization method for monotone second-order cone complementarity problems. SIAM J. Optim. **15**, 593–615 (2005)
17. Hayashi, S., Yamashita, N., Fukushima, M.: A matrix-splitting method for symmetric affine second-order cone complementarity problems. J. Comput. Appl. Math. **175**, 335–353 (2005)
18. Hestenes, M.R.: Multiplier and gradient methods. J. Optim. Theory Appl. **4**, 303–320 (1969)
19. Hiriart-Urruty, J.B., Seeger, A.: A variational approach to copositive matrices. SIAM Rev. **52**, 593–629 (2010)
20. Kong, L.C., Sun, J., Xiu, N.H.: A regularized smoothing Newton method for symmetric cone complementarity problems. SIAM J. Optim. **19**, 1028–1047 (2008)
21. Li, R.-C.: Solving secular equations stably and efficiently. Technical Report UCB//CSD-94-851, Computer Science Division, Department of EECS, University of California at Berkeley (1993)
22. Lobo, M.S., Vandenberghe, L., Boyd, S., Lebret, H.: Application of second-order cone programming. Linear Algebra Appl. **284**(2), 193–228 (1998)
23. Loewy, R.: Positive operators on the n-dimensional ice cream cone. J. Math. Anal. Appl. **49**(2), 375–392 (1975)
24. Luo, Z.Q., Tseng, P.: On the linear convergence of descent methods for convex essentially smooth minimization. SIAM J. Control Optim. **30**(2), 408–425 (1992)
25. Nocedal, J., Wright, S.: Numerical Optimization, 2nd edn. Springer, New York (2006)
26. Pang, J.S., Sun, D.F., Sun, J.: Semismooth homeomorphisms and strong stability of semidefinite and Lorentz complementarity problems. Math. Oper. Res. **28**, 39–63 (2003)
27. Powell, M.J.D.: A method for nonlinear constraints in minimization problems. In: Fletcher, R. (ed.) Optimization, pp. 283–298. Academic Press, New York (1969)
28. Stewart, G.W., Sun, J.G.: Matrix Perturbation Theory. Academic Press, Boston (1990)
29. Sturm, J.F.: Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Opt. Methods Soft. **11**, 625–653 (1999)
30. Toh, K.C., Todd, M.J., Tütüncü, R.H.: SDPT3-a MATLAB software package for semidefinite programming. Opt. Methods Soft. **11**, 545–581 (1999)
31. Tütüncü, R.H., Toh, K.C., Todd, M.J.: Solving semidefinite-quadratic-linear programs using SDPT3. Math. Progr. **95**, 189–217 (2003)
32. Yang, W.H., Yuan, X.M.: The GUS-property of second-order cone linear complementarity problems. Math. Progr. **141**, 295–317 (2013)
33. Yang, W.H., Zhang, L.-H., Shen, C.: Solution analysis for the pseudomonotone second-order cone linear complementarity problem. Optimization **65**(9), 1703–1715 (2016)
34. Yang, W.H., Zhang, L.-H., Shen, C.: On the range of the pseudomonotone second-order cone linear complementarity problem. J. Optim. Theory Appl. **173**(2), 504–522 (2017)
35. Zhang, L.-H., Shen, C., Yang, W.H., Júdice, J.J.: A Lanczos method for large-scale extreme Lorentz eigenvalue problems. SIAM J. Matrix Anal. Appl. **39**(2), 611–631 (2018)
36. Zhang, L.-H., Yang, W.H.: An efficient algorithm for second-order cone linear complementarity problems. Math. Comput. **83**, 1701–1726 (2013)
37. Zhang, L.-H., Yang, W.H.: An efficient matrix splitting method for the second-order cone complementarity problem. SIAM J. Optim. **24**(3), 1178–1205 (2014)

38. Zhang, L.-H., Yang, W.H., Shen, C., Li, R.-C.: A Krylov subspace method for large scale second order cone linear complementarity problem. SIAM J. Sci. Comput. **37**(4), A2046–A2075 (2015)