SIAM J. SCI. COMPUT.
Vol. 37, No. 4, pp. A2046–A2075

A KRYLOV SUBSPACE METHOD FOR LARGE-SCALE SECOND-ORDER CONE LINEAR COMPLEMENTARITY PROBLEM*

LEI-HONG ZHANG†, WEI HONG YANG‡, CHUNGEN SHEN§, AND REN-CANG LI¶

Abstract. In this paper, we first show that the second-order cone linear complementarity problem (SOCLCP) can be solved by finding a positive zero $s_* \in \mathbb{R}$ of a particular rational function $h(s)$, and we then propose a Krylov subspace method to reduce $h(s)$ to $h_\ell(s)$ as in the model reduction. The zero s_* of $h(s)$ can be accurately approximated by that of $h_\ell(s) = 0$, which itself can be cast as a small eigenvalue problem. The new method is made possible by a complete characterization of the curve of $h(s)$, and it has several advantages over the bisection-Newton (BN) iteration recently proposed by [L.-H. Zhang and W. H. Yang, *Math. Comp.*, 83 (2013), pp. 1701–1720] and shown to be very efficient for small- to medium-size problems. The method is tested and compared against the BN iteration and two other state-of-the-art packages: SDPT3 and SeDuMi. Our numerical results show that the method is very efficient for both small to medium dense problems and large-scale ones.

Key words. second-order cone, linear complementarity problem, SOCLCP, globally uniquely solvable property, GUS, Krylov subspace, model reduction, linear complementarity problem via Arnoldi process, LCPvA

AMS subject classifications. 90C33, 65K05, 65F99, 65F15, 65F30, 65P99

DOI. 10.1137/140995064

1. Introduction.

For a given matrix-vector pair $(M, \boldsymbol{q}) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n$, in this paper, we will consider computational methods for the *second-order cone linear complementarity problem* (SOCLCP):

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

where $\mathbb{K}^n$ is the *second-order cone*, also known as the *Lorentz cone* or the *ice cream cone*, defined by

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

*Submitted to the journal's Methods and Algorithms for Scientific Computing section November 10, 2014; accepted for publication (in revised form) May 28, 2015; published electronically August 13, 2015.

http://www.siam.org/journals/sisc/37-4/99506.html

†School of Mathematics, Shanghai Key Laboratory of Financial Information Technology, Shanghai University of Finance and Economics, 777 Guoding Road, Shanghai 200433, China (longzlh@gmail.com). This author's work was supported in part by NSFC grant 11371102 and the Basic Academic Discipline Program, the 11th five year plan of 211 Project for Shanghai University of Finance and Economics.

‡School of Mathematical Sciences, Fudan University, Shanghai 200433, China (whyang@fudan.edu.cn). This author's work was supported by NSFC grant 11371102 and NSFC Key Project 91330201.

§School of Statistics and Mathematics, Shanghai Finance University, Shanghai 201209, China (shenchungen@gmail.com). This author's work was supported in part by NSFC grants 11101281 and 11271259 and Innovation Program of Shanghai Municipal Education Commission grant 12YZ172.

¶Department of Mathematics, University of Texas at Arlington, Arlington, TX 76019-0408 (rcli@uta.edu). This author's work was supported in part by NSF grants DMS-1115834 and DMS-1317330, by a research gift grant from Intel Corporation, and by NSFC grant 11428104.

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$(M, \mathbb{K}^n, \boldsymbol{q})$. SOCLCP (1.1) can arise from solving the type of conic optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{K}^n} \frac{1}{2}\boldsymbol{x}^\top M\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} \tag{1.2}$$

through its KKT condition. Conventionally, conic optimization is concerned with linear objective functions [3], while in (1.2), the objective function is quadratic. Conic optimization has been a thriving research area within the optimization community.

SOLCLP (1.1) is also a special case of the linear complementarity problem over a Cartesian product of *multiple second-order cones* $\mathbb{K}_{\times m} := \mathbb{K}^{n_1} \times \mathbb{K}^{n_2} \times \cdots \times \mathbb{K}^{n_m}$:

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

We refer to, e.g., [4, 11, 13, 18, 25, 26, 29, 36, 45] for various applications as well as for the theoretical and numerical studies. As it is a more general problem, many theoretical properties that we will discuss, develop, and exploit for SOLCLP (1.1) are no longer admitted by (1.3). These properties are crucial for us to design our efficient algorithms for (1.1), however.

Any efficient approach for SOCLCP (1.1) not only is of value in its own right but also shed lights on solving various other second-order cone related programming problems to be discussed later in section 7, including the more general linear complementarity problem (1.3). In fact, the matrix splitting method proposed in [60] for (1.3) is built upon the bisection-Newton (BN) iteration [59] for (1.1), whose efficiency alone basically determines the efficiency of the matrix splitting method there. But the splitting method is structured in such a way that the BN iteration is easily replaceable by any efficient solver for (1.1) to yield another splitting method. Our algorithm for (1.1) in this paper represents such a solver.

The BN iteration is a recent development based on the basic algebraic-geometric properties for SOCLCP (1.1) with M having the so-called globally uniquely solvable (GUS) property[1] [24, 25, 26, 58] (see Definition 2.3 below); BN was employed and embedded successfully into the matrix-splitting framework in [60] to solve the linear complementary problem (1.3). The BN iteration solves SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ from an entirely different perspective from other methods in the literature (e.g., [10, 11, 13, 18, 29, 33, 35, 44, 53, 55, 56, 57]), and it is quite efficient for small- to medium-scale problems [59]. However, for large-scale problems, there are two bottlenecks: (1) it needs to solve the unique positive eigenvalue of $M^\top J_n$ and the associated eigenvector accurately, and (2) repeatedly it has to solve many linear systems with coefficient matrices $M - sJ_n$, where s is some varying shift and

$$J_n := \mathrm{diag}(1, -1, \ldots, -1) \in \mathbb{R}^{n\times n}. \tag{1.4}$$

In this paper, we will propose an efficient algorithm for SOCLCP (1.1). Our numerical tests show that it is competitive with the BN iteration for small- to medium-scale problems, but more importantly it is suitable for large-scale problems.

Our basic idea is as follows. First, we will show that SOCLCP (1.1) with M having the GUS property can be solved by finding a particular positive zero s_* of a rational function $h(s)$. We will show that this function $h(s)$ enjoys many nice geometry properties on $s \in (0, +\infty)$ and indeed it can only admit one or two zeros on

[1] A complete algebraic-geometric characterization of this property will be detailed in Theorem 2.4.

$(0, +\infty)$. Based on the rich techniques in the model reduction, we then propose a Krylov subspace method to approximate $h(s)$ by a Padé-type approximation $h_\ell(s)$. With the aid of the geometry properties of $h(s)$, the positive zero s_* of $h(s)$ can be efficiently computed by solving $h_\ell(s) = 0$ through a small eigenvalue problem. Our new method does not rely upon the spectral information of $M^\top J_n$ and is suitable for large-scale problems. We tested our method and compared it with the BN iteration and two available MATLAB software packages, SDPT3 [55, 56, 57] and SeDuMi [53], for small to medium dense problems as well as for large-scale sparse problems. Preliminary numerical experiments show that our method is rather efficient in either case in terms of accuracy and speed.

Both SOCLCPs (1.1) and (1.3) are similar to the classical linear complementarity problem over the cone $\mathbb{R}^n_+ = \{\boldsymbol{x} = [x_i] \in \mathbb{R}^n \,:\, x_i \geq 0, i = 1, 2, \ldots, n\}$,

$$\boldsymbol{x} \in \mathbb{R}^n_+, \quad \boldsymbol{q} + M\boldsymbol{x} \in \mathbb{R}^n_+, \quad \boldsymbol{x}^\top(\boldsymbol{q} + M\boldsymbol{x}) = 0,$$

which arises from many areas and has a wealth of development in both theory and implementation (see, e.g., [9, 12, 15, 31, 34, 42]).

The rest of this article is organized as follows. In section 2, we present some basic results for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$, primarily with the GUS property, and then we give a brief description of the BN iteration [59] in section 3. We take a new look at SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ from the view point of finding the positive zero s_* of $h(s)$ in section 4, which is followed by a detailed theoretical analysis of its geometry property in section 5. Section 6 discusses finding the zero $s_* > 0$ of $h(s)$: the Padé-type approximation $h_\ell(s)$ of $h(s)$ via Krylov subspace techniques is first introduced, and then the strategy for obtaining an approximation of s_* is presented. We outline several potential applications of any efficient solver for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ to solve other important cone-related programming problems in section 7. Our numerical tests and comparison with the BN iteration, SDPT3, and SeDuMi are reported in section 8. Final concluding remarks are made in section 9.

Notation. Throughout this paper, all vectors are column vectors and are typeset in bold lowercase letters. For $A \in \mathbb{R}^{n\times m}$ (the set of all $m \times n$ real matrices), $A^\top$ denotes its transpose, and $\mathcal{R}(A)$ and $\mathcal{N}(A)$ represent the range and kernel of A, respectively. Thus $\mathcal{R}(A)^\perp = \mathcal{N}(A^\top)$, 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 ith 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. Notation $(i : j)$ 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. For a matrix already with a subscript, e.g., Y_r, we write $Y_{r;(k:\ell,i:j)}$ for $(Y_r)_{(k:\ell,i:j)}$. Notation $\|\cdot\|_2$ is either the matrix spectral norm or the Euclidean vector norm, depending on its arguments:

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

The ℓth Krylov subspace generated by $A \in \mathbb{R}^{n\times n}$ on $\boldsymbol{x} \in \mathbb{R}^n$ is defined as

$$\mathcal{K}_\ell(A, \boldsymbol{x}) = \operatorname{span}(\boldsymbol{x}, A\boldsymbol{x}, \ldots, A^{\ell-1}\boldsymbol{x}).$$

2. Basic Properties of SOCLCP. In this section, we review several basic results for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$. These results form the theoretical basis of our development later. For ease of presentation, we define

$$M_s := M - sJ_n, \quad \mathbb{K}_s := M_s\mathbb{K}^n = \{M_s\boldsymbol{x} : \boldsymbol{x} \in \mathbb{K}^n\}. \tag{2.1}$$

Evidently, $\mathbb{K}_0 = M\mathbb{K}^n$. Since $\mathbb{K}_s$ approaches $-\mathbb{K}^n$ as $s \to +\infty$ [58, Lemma 9], we define

$$\mathbb{K}_\infty := -\mathbb{K}^n.$$

Finally, $\partial(\mathbb{K}^n)$ and $\mathrm{int}(\mathbb{K}^n)$ stand for the boundary and the interior of $\mathbb{K}^n$, respectively.

LEMMA 2.1 (see [58]). *The following statements hold:*

(i) *For nonzero vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{K}^n$, $\boldsymbol{x}^\top\boldsymbol{y} = 0$ 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 $\boldsymbol{x}^\top\boldsymbol{y} > 0$ for all nonzero vector $\boldsymbol{y} \in \mathbb{K}^n$ if and only if $\boldsymbol{x} \in \mathrm{int}(\mathbb{K}^n)$.*

The next theorem characterizes three mutually exclusive cases for SOCLCP $(M, \mathbb{K}^n, \boldsymbol{q})$. We point out that if M is nonsingular, (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 2.2 (see [59, 60]). *There are three mutually exclusive cases for the solution $\boldsymbol{x} \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$, namely,*

(C1) *$\boldsymbol{q} \in \mathbb{K}^n$ (which implies that $\boldsymbol{x} = \boldsymbol{0}$ is the solution);*

(C2) *$\mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q}) \supseteq \{\boldsymbol{x} \in \mathbb{K}^n : M\boldsymbol{x} + \boldsymbol{q} = \boldsymbol{0}\} \neq \varnothing$;*

(C3) *there exists $s_* > 0$ such that $M\boldsymbol{x} + \boldsymbol{q} = s_*J_n\boldsymbol{x} \in \partial(\mathbb{K}^n)$.*

Next we introduce the so-called GUS property on M, previously discussed in [24, 25, 26, 58].

DEFINITION 2.3. *$M \in \mathbb{R}^{n\times n}$ is said to have the GUS property if the solution $\boldsymbol{x}$ of* SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ *is unique for any given $\boldsymbol{q} \in \mathbb{R}^n$.*

The next theorem provides an algebraic-geometric characterization of such a property. The reader is referred to [58, 59, 60] for proofs and more.

THEOREM 2.4. *$M \in \mathbb{R}^{n\times n}$ has the GUS property if and only if the following three statements hold:*[2]

(i) *MJ_n has exactly one positive eigenvalue τ in $[0, +\infty)$. Moreover,*

$$\mathrm{rank}(MJ_n - \tau I_n) = \mathrm{rank}(M_\tau J_n) = n - 1 \tag{2.2}$$

and MJ_n has an eigenvector $\boldsymbol{w} \in \mathrm{int}(\mathbb{K}^n)$ associated with τ;

(ii) *$M^\top J_n$ has an eigenvector $\boldsymbol{v} \in \mathrm{int}(\mathbb{K}^n)$ associated with τ, and $\mathcal{R}(M_\tau) = (J_n\boldsymbol{v})^\perp$;*

(iii) *for all $\boldsymbol{a} \in \partial(\mathbb{K}^n)$, $\boldsymbol{a}^\top M\boldsymbol{a} \geq 0$ and $\boldsymbol{a}^\top M^{-1}\boldsymbol{a} \geq 0$.*

When M has the GUS property, the following statements hold:

(a) *For $0 < s < \tau < t$, $\mathbb{K}_0\backslash\{\boldsymbol{0}\} \subset \mathrm{int}(\mathbb{K}_s)$, and $\mathbb{K}_\infty\backslash\{\boldsymbol{0}\} \subset \mathrm{int}(\mathbb{K}_t)$. Moreover, $\mathbb{K}_s$ and $\mathbb{K}_t$ lie in the two opposite sides of $\mathcal{R}(M_\tau)$, which is a hyperplane in $\mathbb{R}^n$.*

(b) *$\mathbb{K}_t\backslash\{\boldsymbol{0}\} \subset \mathrm{int}(\mathbb{K}_s)$ for $0 < t < s < \tau$ and for $\tau < s < t$.*

[2] We note that MJ_n has the same eigenvalues as $(MJ_n)^\top = J_nM^\top$, which has the same eigenvalues as $J_n^{-1}(J_nM^\top)J_n = M^\top J_n$.

(c) *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*

$$s_* \begin{cases} = \tau & \text{if } (-\boldsymbol{q})^\top J_n \boldsymbol{v} = 0 \ \text{ or equivalently, } \ \boldsymbol{q} \in \mathcal{R}(M_\tau), \\ < \tau & \text{if } (-\boldsymbol{q})^\top J_n \boldsymbol{v} > 0, \\ > \tau & \text{if } (-\boldsymbol{q})^\top J_n \boldsymbol{v} < 0. \end{cases}$$

We remark that Theorem 2.4(i) does not say anything about the algebraic multiplicity of τ, i.e., it does not exclude the case in which τ is a multiple eigenvalue algebraically. However, if τ indeed is an algebraically multiple eigenvalue, the corresponding eigenvector of MJ_n is unique, modulo a nonzero scalar factor, because of (2.2).

Theorem 2.4(i) implies that 0 cannot be an eigenvalue of MJ_n and thus M must be nonsingular if M has the GUS property. Also, an immediate consequence of this theorem is that *a symmetric positive definite* M *has the GUS property.* In fact, more can be said:

1. any positive definite (not necessarily symmetric) M, in the sense that $M+M^\top$ is symmetric positive definite, has the GUS property [26, Theorem 17], and
2. a symmetric M has the GUS property if and only if it is positive definite (implied by [26, Theorem 21]).

In the case when M is symmetric positive definite, $M - \lambda J_n$ is the so-called positive definite pencil [39, Definition 1.1] since $M - 0 \cdot J_n = M$ is symmetric positive definite. It follows from a more general result there that $M - \lambda J_n$ is diagonalizable and has one simple positive eigenvalue and $n-1$ negative eigenvalues [39, Lemma 3.8]. So in this case, τ in Theorem 2.4(i) is a simple eigenvalue.

Although all positive definite (not necessarily symmetric) matrices have the GUS property as a consequence of Theorem 2.4, in general it is not easy to verify whether a given matrix M has the property. This makes the GUS property hard to verify, or even restrictive. But in section 7, we will demonstrate that $\mathrm{SOCLCP}(M, \mathbb{K}^n, \boldsymbol{q})$ with positive definite M arises from solving several important cone-related programming problems, including the case of multiple second-order cones (1.3). This means that SOCLCPs with M being positive definite are already significant enough to warrant a thorough investigation, as we will make in this paper.

By Theorem 2.2, we know that if $\boldsymbol{x} \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$ is not in the cases (C1) and (C2), then it must be in the case (C3), i.e., there exists $s_* > 0$ such that $M\boldsymbol{x} + \boldsymbol{q} = s_* J_n \boldsymbol{x}$. If also M has the GUS property, then $\boldsymbol{x}$ is unique and thus s_* is unique, too. According to Theorem 2.4(c), we can locate s_* with the information of the eigenpair $(\tau, \boldsymbol{v})$ of $M^\top J_n$. This is exactly what the BN iteration was designed to do [59].

Throughout the rest of this paper, our target is $\mathrm{SOCLCP}(M, \mathbb{K}^n, \boldsymbol{q})$ with M having the GUS property, and all assignments to τ, s_*, $\boldsymbol{q}$, $\boldsymbol{v}$, $\boldsymbol{w}$ in Theorem 2.4 will be kept.

3. The BN iteration. The BN iterative method [59] is based on Theorem 2.2, where the cases (C1) and (C2) are rather trivial. It is the case (C3) that is conceivably more common and needs most work. The BN iteration combines the bisection and the Newton method to generate a sequence $\{s_k\}$ that converges to s_* by investigating whether

$$\boldsymbol{x}(s_k) := -(M - s_k J_n)^{-1} \boldsymbol{q}$$

is in $\mathbb{K}^n$. We refer to [59, Algorithm 3] and [60, Algorithm 2] for more detail.

The iteration needs to know the eigenpair $(\tau, \boldsymbol{v})$ of $M^\top J_n$ associated with the unique positive eigenvalue τ of $M^\top J_n$. Therefore, the main computational burden is to find the eigenpair $(\tau, \boldsymbol{v})$ and to compute $\boldsymbol{x}(s_k)$ in each step. The latter has a cost of $O(n^3)$ flops in general but can be much less for certain special structured M. For example, when M is in the upper Hessenberg form, it costs only $O(n^2)$ to compute $\boldsymbol{x}(s_k)$. In general we can always reduce M to an upper Hessenberg matrix through an orthogonal transformation (more detail in subsection 6.5). This reduction of M to an upper Hessenberg matrix itself still costs $O(n^3)$ flops, but it needs to be done only once.

For the BN iteration with a preprocessing, transforming M to $Q^\top MQ$ is rather efficient for small- to medium-scale problems. However, for large-scale problems, such a transformation is too costly because it costs $O(n^3)$ flops. Thus there are two computational bottlenecks in this approach, namely,

1. the need for the eigenpair $(\tau, \boldsymbol{v})$ of $M^\top J_n$, and
2. the transformation of M to $Q^\top MQ$ which in general cannot exploit, e.g., much of the sparsity structure of M for a large-scale SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$.

In this paper, we will attempt to circumvent both computational bottlenecks by developing numerical methods that are efficient for large-scale SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ for which either M is sparse or the matrix-vector product with M is fast (at a cost much less than $O(n^2)$ flops that is needed usually for a dense M). Our new method still focuses on the case (C3).

4. Transform SOCLCP into a zero-finding problem. Our new method relies on the following simple observation.

THEOREM 4.1. *Suppose $s \in \mathbb{R}$ is not an eigenvalue of MJ_n. Let*

$$\boldsymbol{x}(s) \equiv \begin{bmatrix} x_1(s) \\ \boldsymbol{x}_2(s) \end{bmatrix} := -(M - sJ_n)^{-1}\boldsymbol{q}, \tag{4.1}$$

where J_n is given by (1.4). *Then $\boldsymbol{x}(s) \in \partial(\mathbb{K}^n)$ if and only if $x_1(s) > 0$ and $h(s) = 0$, where*

$$h(s) := \boldsymbol{x}(s)^\top J_n \boldsymbol{x}(s) = \boldsymbol{q}^\top (M - sJ_n)^{-\top} J_n (M - sJ_n)^{-1}\boldsymbol{q}. \tag{4.2}$$

Proof. The proof is evident by the definition of $\mathbb{K}^n$. ∎

According to Theorem 4.1, we know that finding s_* for the case (C3) in Theorem 2.2 is equivalent to finding the zero s_* of $h(s)$. Notice that

$$h(s) = \boldsymbol{q}^\top [s^2 J_n - (M + M^\top)s + MJ_nM^\top]^{-1}\boldsymbol{q} \tag{4.3}$$

is a rational function that involves the inverse of an $n \times n$ quadratic matrix-valued function. Such a function has the same form as the so-called second-order transfer functions that are sought to be reduced in, e.g., [5, 6, 14, 17, 37, 51, 54], by the name of the *second-order model reduction*. The basic idea, in the context of $h(s)$ here, is to approximate $h(s)$ by a Padé-type approximation $h_\ell(s)$ having the same form but involving the inverse of a much smaller, say, $\ell \times \ell$, quadratic matrix-valued function. Usually $\ell \ll n$, say, $\ell = 10$ up to 50 for example. Some of the *poles* (i.e., values of s at which $h_\ell(s)$ becomes ∞) and zeros of $h_\ell(s)$ are often good approximations to some of those of $h(s)$, respectively. Note from (4.1) and (4.2) that generically, the unique positive eigenvalue τ of MJ_n (which is also the one of $M^\top J_n$) is a pole of $h(s)$, and thus conceivably one of the poles of $h_\ell(s)$ should provide a good approximation of τ (see the details in sections 5 and 6 below).

Thanks to powerful techniques developed in the model reduction, the (rational) Krylov subspace techniques can be employed to construct $h_\ell(s)$ at the price of only solving certain linear systems involving M. This makes it possible for us to numerically solve large-scale SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ because the involved linear systems can be solved iteratively through exploiting the sparsity of M or fast matrix-vector multiplications by M. Furthermore, this approach does not require the eigenpair $(\tau, \boldsymbol{v})$ of $M^\top J_n$, and it even can produce accurate approximations to τ by the poles of $h_\ell(s)$ (see Theorem 5.2 below). We shall discuss these issues in detail in section 6.

In Theorem 2.4(c), the case $s_* \neq \tau$ is generic and should occur more frequently than otherwise, and thus the case should be considered more carefully. On the other hand, Zhang and Yang [59] explained how to compute the solution $\boldsymbol{x} \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$ by a direct method [59, Algorithm 2] for the special case $s_* = \tau$. Therefore we will not invest much effort in the special case from now on.

5. Analyze the curve of *h(s)*. In this section, we shall present a theoretical analysis of the curve of $h(s)$ defined in (4.2). By relying upon the special structure of SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$, our analysis reveals some basic geometry properties of $h(s)$ both for the generic case $s_* \neq \tau$ and for the special case $s_* = \tau$. These findings are helpful in guiding us to compute the wanted zero and pole of $h(s)$.

5.1. The general case $s_* \neq \tau$. We first analyze the curve $h(s)$ of (4.2) for the generic case: $s_* \neq \tau$. According to Theorem 2.4(c), $s_* \neq \tau$ *if and only if* $\boldsymbol{v}^\top J_n \boldsymbol{q} \neq 0$ *or, equivalently,* $\boldsymbol{q} \notin \mathcal{R}(M_\tau)$.

As a convention, in what follows, we will use "$h(s) = -$" to mean $h(s) < 0$ and likewise "$h(s) = +$" to mean $h(s) > 0$.

LEMMA 5.1. *Suppose that M has the GUS property and $\boldsymbol{q} \notin (-M\mathbb{K}^n) \cup \mathbb{K}^n$. We have the following statements:*

(a) *If $s_* \in (0, \tau)$, then*

$$h(s) = \begin{cases} - & \text{for } s \in (0, s_*), \\ 0 & \text{for } s = s_*, \\ + & \text{for } s \in (s_*, \tau). \end{cases}$$

(b) *If $s_* \in (\tau, \infty)$, then*

$$h(s) = \begin{cases} + & \text{for } s \in (\tau, s_*), \\ 0 & \text{for } s = s_*, \\ - & \text{for } s \in (s_*, \infty). \end{cases}$$

Proof. First, we know that under the conditions, the solution $\boldsymbol{x}(s_*) \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$ is on the boundary $\partial(\mathbb{K}^n)$ of $\mathbb{K}^n$, where $\boldsymbol{x}(s)$ is defined by (4.1).

Consider (a), where $s \in (0, s_*)$. We claim that $\boldsymbol{x}(s) = -M_s^{-1}\boldsymbol{q} \notin \mathbb{K}^n \cup (-\mathbb{K}^n)$, which immediately implies $h(s) \equiv [x_1(s)]^2 - \|\boldsymbol{x}_2(s)\|_2^2 < 0$. Otherwise, if $\boldsymbol{x}(s) \in \mathbb{K}^n$, by Theorem 2.4(b), we have

$$-\boldsymbol{q} = M_s \boldsymbol{x}(s) \in \mathbb{K}_s \backslash \{\mathbf{0}\} \subset \mathrm{int}(\mathbb{K}_{s_*}),$$

implying $-\boldsymbol{q} = M_{s_*}\boldsymbol{y}$ for some $\boldsymbol{y} \in \mathrm{int}(\mathbb{K}^n)$. Thus $\boldsymbol{x}(s_*) = -M_{s_*}^{-1}\boldsymbol{q} = \boldsymbol{y} \in \mathrm{int}(\mathbb{K}^n)$, contradicting $\boldsymbol{x}(s_*) \in \partial(\mathbb{K}^n)$. Similarly,

$$\boldsymbol{x}(s) \in -\mathbb{K}^n \;\Rightarrow\; \boldsymbol{q} = M_s(-\boldsymbol{x}(s)) \in \mathbb{K}_s \backslash \{\mathbf{0}\} \subset \mathrm{int}(\mathbb{K}_{s_*}) \;\Rightarrow\; -\boldsymbol{x}(s_*) \in \mathrm{int}(\mathbb{K}^n),$$

TABLE 1
The sign of $h(s)$ on $(0,\infty)$ in terms of location of $\boldsymbol{q}$.

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

contradicting again $\boldsymbol{x}(s_*) \in \partial(\mathbb{K}^n)$. Now, for $s_* < s < \tau$, from Theorem 2.4(b), we have

$$-\boldsymbol{q} = M_{s_*}\boldsymbol{x}(s_*) \in \mathbb{K}_{s_*}\backslash\{\mathbf{0}\} \subset \mathrm{int}(\mathbb{K}_s) \;\Rightarrow\; \boldsymbol{x}(s) \in \mathrm{int}(\mathbb{K}_s) \;\Rightarrow\; h(s) > 0.$$

Case (b) can be proved similarly. For $s > s_*$, we claim that $\boldsymbol{x}(s) = -M_s^{-1}\boldsymbol{q} \notin \mathbb{K}^n \cup (-\mathbb{K}^n)$, which directly implies $h(s) < 0$. Otherwise, by Theorem 2.4(b), we have

$$\boldsymbol{x}(s) \in \mathbb{K}^n \;\Rightarrow\; -\boldsymbol{q} = M_s\boldsymbol{x}(s) \in \mathbb{K}_s\backslash\{\mathbf{0}\} \subset \mathrm{int}(\mathbb{K}_{s_*}) \;\Rightarrow\; \boldsymbol{x}(s_*) \in \mathrm{int}(\mathbb{K}^n),$$

contradicting $\boldsymbol{x}(s_*) \in \partial(\mathbb{K}^n)$. Similarly,

$$\boldsymbol{x}(s) \in -\mathbb{K}^n \;\Rightarrow\; \boldsymbol{q} = M_s(-\boldsymbol{x}(s)) \in \mathbb{K}_s\backslash\{\mathbf{0}\} \subset \mathrm{int}(\mathbb{K}_{s_*}) \;\Rightarrow\; -\boldsymbol{x}(s_*) \in \mathrm{int}(\mathbb{K}^n),$$

contradicting again $\boldsymbol{x}(s_*) \in \partial(\mathbb{K}^n)$. Now, for $\tau < s < s_*$, from Theorem 2.4(b), we have

$$-\boldsymbol{q} = M_{s_*}\boldsymbol{x}(s_*) \in \mathbb{K}_{s_*}\backslash\{\mathbf{0}\} \subset \mathrm{int}(\mathbb{K}_s) \;\Rightarrow\; \boldsymbol{x}(s) \in \mathrm{int}(\mathbb{K}_s) \;\Rightarrow\; h(s) > 0.$$

The proof is complete. □

We now present our main theorem in this subsection. It gives a completed picture of $h(s)$ on $(0,\infty)$ as detailed in Table 1 for the current case. 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, $\boldsymbol{q} \notin \mathcal{R}(M_\tau)$ 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 5.2. *Suppose M has the GUS property. If $\boldsymbol{q} \notin \mathcal{R}(M_\tau)$, then the following hold:*

(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_*.*

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

Proof. (1) We first remark that the five cases of $\boldsymbol{q}$ presented in Table 1 are mutually exclusive and also complete for $\boldsymbol{q} \in \mathbb{R}^n \backslash \mathcal{R}(M_\tau)$. The first two cases are (C1) and (C2) in Theorem 2.2.

We consider case 3. Note that $\boldsymbol{q} \in (-\mathbb{K}^n)\backslash(-M\mathbb{K}^n)$ implies $\boldsymbol{q}^\top J_n \boldsymbol{v} < 0$ and by Theorem 2.4(c), we know that $0 < s_* < \tau$. Thus, using Lemma 5.1, we only need to show $h(s) > 0$ for $s > \tau$, which is true because $-\mathbb{K}^n \backslash \{\mathbf{0}\} \subseteq \mathrm{int}(\mathbb{K}_s)$ by Theorem 2.4(a), and thus

$$\boldsymbol{q} \in (-\mathbb{K}^n)\backslash(-M\mathbb{K}^n) \subseteq \mathrm{int}(\mathbb{K}_s) \;\Rightarrow\; -\boldsymbol{x}(s) \in \mathrm{int}(\mathbb{K}^n) \;\Rightarrow\; h(s) > 0.$$

For case 4, we note that for $s \in (0, \tau)$, it follows from Theorem 2.4(a), (b) that

$$\boldsymbol{q} \in M\mathbb{K}^n \backslash \mathbb{K}^n = \mathbb{K}_0 \backslash \mathbb{K}^n \subset \mathrm{int}(\mathbb{K}_s) \;\Rightarrow\; -\boldsymbol{x}(s) \in \mathrm{int}(\mathbb{K}^n) \;\Rightarrow\; h(s) > 0,$$

which also implies that $s_* > \tau$. Therefore, combining it with Lemma 5.1 yields the conclusion.

For the last case, note that $\boldsymbol{q} \notin (-M\mathbb{K}^n) \cup M\mathbb{K}^n \cup \mathbb{K}^n \cup (-\mathbb{K}^n) \cup \mathcal{R}(M_\tau)$ implies both $\boldsymbol{q} \notin (-M\mathbb{K}^n) \cup \mathbb{K}^n$ and $-\boldsymbol{q} \notin (-M\mathbb{K}^n) \cup \mathbb{K}^n$. We now prove the result for $\boldsymbol{v}^\top J_n \boldsymbol{q} < 0$ and $\boldsymbol{v}^\top J_n \boldsymbol{q} > 0$.

(a) $\boldsymbol{v}^\top J_n \boldsymbol{q} < 0$. By Theorem 2.4(c), we know $0 < s_* < \tau$. So $s_{*;1} = s_*$. On the other hand, consider the problem SOCLCP$(M, \mathbb{K}^n, -\boldsymbol{q})$. Since $-\boldsymbol{q} \notin (-M\mathbb{K}^n) \cup \mathbb{K}^n$, we know also from Theorem 2.4(c) that there is another $s_{*;2} > 0$ such that

$$-(M - s_{*;2} J_n)^{-1}(-\boldsymbol{q}) = -\boldsymbol{x}(s_{*;2}) \in \mathrm{SOL}(M, \mathbb{K}^n, -\boldsymbol{q}).$$

Moreover, because $\boldsymbol{v}^\top J_n(-\boldsymbol{q}) > 0$, by applying Theorem 2.4(c) to the problem SOCLCP$(M, \mathbb{K}^n, -\boldsymbol{q})$, we conclude that $s_{*;2} > \tau$. Consequently, based on Lemma 5.1, the assertion follows.

(b) $\boldsymbol{v}^\top J_n \boldsymbol{q} > 0$. The proof for this situation is similar to the first situation (a), except that s_* now is $s_{*;2}$. The details are omitted.

(2) Premultiplying $(M - sJ_n)\boldsymbol{x}(s) = -\boldsymbol{q}$ by $\boldsymbol{v}^\top J_n$ on both sides and using $(J_n \boldsymbol{v})^\top M = \tau \boldsymbol{v}^\top$, we know that for any $0 < s \neq \tau$,

$$(\tau - s)\boldsymbol{v}^\top \boldsymbol{x}(s) = -\boldsymbol{v}^\top J_n \boldsymbol{q} \neq 0 \tag{5.1}$$

and hence

$$\lim_{s \to \tau} |\boldsymbol{v}^\top \boldsymbol{x}(s)| = +\infty \quad \text{and} \quad \lim_{s \to \tau} \|\boldsymbol{x}(s)\|_2 = +\infty.$$

Let $\boldsymbol{z}$ be *any* accumulation point of $\frac{\boldsymbol{x}(s)}{\|\boldsymbol{x}(s)\|_2}$ as $s \to \tau$, i.e., there exists a sequence $s_1, s_2, \ldots$ such that $s_i \to \tau$ and $\frac{\boldsymbol{x}(s_i)}{\|\boldsymbol{x}(s_i)\|_2} \to \boldsymbol{z}$ as $i \to +\infty$. Dividing $(M - sJ_n)\boldsymbol{x}(s) = -\boldsymbol{q}$ by $\|\boldsymbol{x}(s)\|_2$, we have for $0 < s \neq \tau$

$$(MJ_n - sI_n) J_n \frac{\boldsymbol{x}(s)}{\|\boldsymbol{x}(s)\|_2} = -\frac{\boldsymbol{q}}{\|\boldsymbol{x}(s)\|_2}. \tag{5.2}$$

Putting $s = s_i$ in (5.2) and letting $i \to +\infty$, we have $(MJ_n - \tau I_n) J_n \boldsymbol{z} = \mathbf{0}$. Since $\|J_n \boldsymbol{z}\|_2 = 1$, by Theorem 2.4(i),

$$\text{either} \quad J_n \boldsymbol{z} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2} \quad \text{or} \quad J_n \boldsymbol{z} = -\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}.$$

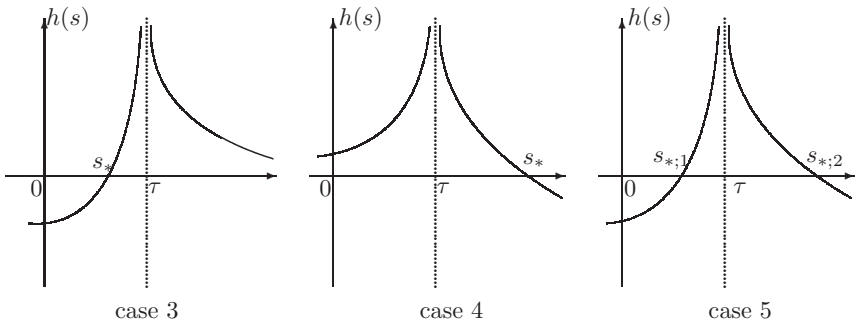

FIG. 1. $h(s)$ *corresponding to cases* 3, 4, *and* 5 *in Table* 1.

In other words, the only possible accumulation points of $\frac{\boldsymbol{x}(s)}{\|\boldsymbol{x}(s)\|_2}$ are $\boldsymbol{z} = \pm J_n \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$. This fact together with $\lim_{s\to\tau} \|\boldsymbol{x}(s)\|_2 = +\infty$ is sufficient for us to conclude $\lim_{s\to\tau} h(s) = +\infty$. In fact, if the opposite were true, then there would be a sequence $s_1, s_2, \ldots$ converging to τ such that $\{h(s_i)\}$ is bounded. We can assume without loss of generality that $\lim_{i\to+\infty} h(s_i) = \varphi < +\infty$. Note that we can choose a subsequence $s_{i_1}, s_{i_2}, \ldots$ such that

$$\lim_{j\to+\infty} \frac{\boldsymbol{x}(s_{i_j})}{\|\boldsymbol{x}(s_{i_j})\|_2} = J_n \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2} \quad \text{or} \quad \lim_{j\to+\infty} \frac{\boldsymbol{x}(s_{i_j})}{\|\boldsymbol{x}(s_{i_j})\|_2} = -J_n \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}.$$

But in either case,

$$\varphi = \lim_{j\to+\infty} h(s_{i_j}) = \lim_{j\to+\infty} \frac{\boldsymbol{x}(s_{i_j})^\top J_n \boldsymbol{x}(s_{i_j})}{\|\boldsymbol{x}(s_{i_j})\|_2^2} \|\boldsymbol{x}(s_{i_j})\|_2^2 = \frac{\boldsymbol{w}^\top J_n \boldsymbol{w}}{\|\boldsymbol{w}\|_2^2} \lim_{j\to+\infty} \|\boldsymbol{x}(s_{i_j})\|_2^2 = +\infty,$$

a contradiction, where we have used the fact that $\boldsymbol{w} \in \mathrm{int}(\mathbb{K}^n)$ and thus $\boldsymbol{w}^\top J_n \boldsymbol{w} > 0$. This completes the proof. □

Remark 5.3. Except for the trivial cases, cases 1 and 2, Theorem 5.2 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 $\boldsymbol{q}$. These one-to-one correspondences are important for correctly identifying the right approximation of s_* via the reduced $h_\ell(s)$ (see section 6.1). Before reducing $h(s)$ to $h_\ell(s)$, 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, or 5. Now the location of $\boldsymbol{q}$ and, thereby, the sign pattern of $h(s)$ or, equivalently, the relationship between τ and s_*, are then clear.

5.2. The special case $s_* = \tau$. We now analyze the curve $h(s)$ for the special case $s_* = \tau$. The next theorem implies that $h(s) < 0$ for all $0 < s \neq \tau$.

THEOREM 5.4. *Suppose* M *has the GUS property and* $\boldsymbol{q} \in \mathcal{R}(M_\tau)$. *Then* $h(s) < 0$ *for all* $0 < s \neq \tau$.

Proof. We show that $\boldsymbol{x}(s) \notin \mathbb{K}^n \cup (-\mathbb{K}^n)$ and therefore $h(s) < 0$ for $0 < s \neq \tau$. Suppose on the contrary that $\boldsymbol{x}(s) \in \mathbb{K}^n \cup (-\mathbb{K}^n)$. Then either $\boldsymbol{x}(s) \in \mathbb{K}^n \Rightarrow \boldsymbol{q} \in \mathbb{K}_s$, which contradicts $\boldsymbol{q} \in \mathcal{R}(M_\tau)$ because $\mathbb{K}_s \cap \mathcal{R}(M_\tau) = \varnothing$, or $\boldsymbol{x}(s) \in -\mathbb{K}^n \Rightarrow \boldsymbol{q} \in -\mathbb{K}_s$, which contradicts $\boldsymbol{q} \in \mathcal{R}(M_\tau)$ because $(-\mathbb{K}_s) \cap \mathcal{R}(M_\tau) = \varnothing$. This establishes our assertion. □

Recall our discussion immediately after Theorem 2.4 that we cannot exclude the possibility that algebraically τ is a multiple eigenvalue of $M^\top J_n$. But if τ is indeed

a *simple* eigenvalue of $M^\top J_n$, the following theorem says that τ is no longer a pole of $h(s)$, i.e., it is a removable singularity. In such a case, by Theorem 5.4, we know $\lim_{s\to\tau} h(s) \le 0$ with a possibility of being an equality.

THEOREM 5.5. *Suppose M has the GUS property. If $\boldsymbol{q} \in \mathcal{R}(M_\tau)$ (and thus $s_* = \tau$) and if τ is a simple eigenvalue of $M^\top J_n$, then $h(s)$ is analytic in $(0, +\infty)$. In particular, if M is symmetric positive definite, then $h(s)$ is analytic in $(0, +\infty)$.*

Proof. It suffices to prove $h(s)$ is analytic at $s = \tau$. To this end, we first note from Theorem 2.4 that when M has the GUS property, we have

$$M^\top J_n \boldsymbol{v} = \tau \boldsymbol{v} \;\Rightarrow\; \boldsymbol{v}^\top J_n M = \tau \boldsymbol{v}^\top.$$

Therefore, if τ is a *simple* eigenvalue of $M^\top J_n$, by the Jordan canonical decomposition of a matrix, there exists a nonsingular matrix $X \in \mathbb{R}^{n\times n}$ such that

$$X\, J_n M = \begin{bmatrix} \tau & \mathbf{0}^\top \\ \mathbf{0} & B \end{bmatrix} X \quad \text{with} \quad X_{(1,:)} = \boldsymbol{v}^\top,$$

i.e., the first row of X is $\boldsymbol{v}^\top$. Thus,

$$\begin{aligned} \boldsymbol{x}(s) &= -(M - sJ_n)^{-1}\boldsymbol{q} \\ &= -(J_n M - sI_n)^{-1} J_n \boldsymbol{q} \\ &= -X^{-1} \begin{bmatrix} \frac{1}{\tau - s} & \mathbf{0}^\top \\ \mathbf{0} & (B - sI_{n-1})^{-1} \end{bmatrix} X J_n \boldsymbol{q} \\ &= -X^{-1} \begin{bmatrix} \frac{1}{\tau - s} & \mathbf{0}^\top \\ \mathbf{0} & (B - sI_{n-1})^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ \boldsymbol{z}_2 \end{bmatrix} \\ &= -X^{-1} \begin{bmatrix} 0 \\ (B - sI_{n-1})^{-1}\boldsymbol{z}_2 \end{bmatrix}, \end{aligned}$$

where we have used $XJ_n\boldsymbol{q} = \begin{bmatrix} \boldsymbol{v}^\top J_n \boldsymbol{q} \\ X_{(2:n,:)} J_n \boldsymbol{q} \end{bmatrix}$ and $\boldsymbol{v}^\top J_n \boldsymbol{q} = 0$ by Theorem 2.4(c), and set $\boldsymbol{z}_2 := X_{(2:n,:)} J_n \boldsymbol{q}$. Therefore,

$$\begin{aligned} h(s) &= \boldsymbol{x}(s)^\top J_n \boldsymbol{x}(s) \\ &= \left[0, \boldsymbol{z}_2^\top (B - sI_{n-1})^{-\top}\right] X^{-\top} J_n X^{-1} \begin{bmatrix} 0 \\ (B - sI_{n-1})^{-1}\boldsymbol{z}_2 \end{bmatrix} \\ &= \boldsymbol{z}_2^\top (B - sI_{n-1})^{-\top} C (B - sI_{n-1})^{-1} \boldsymbol{z}_2, \end{aligned} \tag{5.3}$$

where $C = (X^{-\top} J_n X^{-1})_{(2:n,2:n)} \in \mathbb{R}^{(n-1)\times(n-1)}$. Since every eigenvalue of B is also an eigenvalue of $M^\top J_n$ and τ is assumed a simple eigenvalue of $M^\top J_n$, we conclude that τ cannot be an eigenvalue of B, i.e., $B - \tau I_{n-1}$ is nonsingular, and thus τ is not a pole of $h(s)$. ∎

Remark 5.6. Theorem 5.5 reveals the similarity between SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ and the trust-region subproblem [43, chapter 4]:

$$\min_{\boldsymbol{x}^\top \boldsymbol{x} \le \delta} f(\boldsymbol{x}) \quad \text{with} \quad f(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^\top M \boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x}. \tag{5.4}$$

Indeed, as we have pointed out in (1.2) that when M is symmetric, SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ in (1.1) is just the KKT condition of the conic programming

$$\min_{\boldsymbol{x}^\top J_n \boldsymbol{x} \ge 0,\, x_1 \ge 0} f(\boldsymbol{x}) \quad \text{with} \quad f(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^\top M \boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x}, \tag{5.5}$$

and s in $h(s)$ is the Lagrangian multiplier. By comparing (5.4) to (5.5), we note both have the same objective function but different constraints. Moreover, when both solutions are on the boundaries, then for (5.4), a real-valued rational function $\|p(\lambda)\|_2$ of the Lagrangian multiplier λ, as the counterpart of $h(s)$ here, can also be defined [43, (4.37)]. It is well known that in the general case, $-\lambda_{\min}(M)$ (the negative of the smallest eigenvalue of M) is a pole of $\|p(\lambda)\|_2$, but, like the special case $s = s_*$ in SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$, the Lagrangian multiplier associated with the optimal solution is $\lambda_* = -\lambda_{\min}(M)$ and λ_* is not a pole in the "hard case" [43].

Remark 5.7. Theorems 5.2, 5.4, and 5.5 are important because they together provide us with ways to detect the special case $\boldsymbol{q} \in \mathcal{R}(M_\tau)$. In particular, Theorems 5.2 and 5.4 simply suggest that $\boldsymbol{q} \in \mathcal{R}(M_\tau)$ if $h(s) < 0$ for $0 < s \neq \tau$, so one practical strategy is to check if $h(s) > 0$ for s near τ (see Figure 1).

6. Compute the positive zero s_* of $h(s)$.

6.1. Reduce $h(s)$ via the Arnoldi process. There are several approaches (see, e.g., [5, 51, 54]) to reduce the "transfer function" $h(s)$ in (4.2) which can be related to a time-invariant dynamical system. In particular, in the second-order form as in (4.3), there are two existing treatments:

1. linearizing $h(s)$ and reducing it via the bi-orthogonal Lanczos process by Gallivan, Grimme, and Van Dooren [20] and by Feldman and Freund [17], and
2. reducing $h(s)$ *via* SOAR (second-order Arnoldi) by Bai and Su [5, 6].

Note our "transfer function" $h(s)$ bears the factor form (4.2), but the reduced $h_\ell(s)$ by either treatment above does not preserve the factor form. In addition, *serious breakdowns* may occur.

We opt to design a special reduction procedure, similar to [38], which respects the factor form in (4.2) and uses only the classical Arnoldi process. We will also show that our reduced $h_\ell(s)$ corresponds to another SOCLCP in the form of (1.1) of much smaller scale. Moreover, the Taylor series of $h_\ell(s)$ agrees with that of $h(s)$ in their first ℓ terms at the expansion point s_0. More detailed discussions on the advantages of our factor-form-preserving reduction can be found in Remarks 6.2 and 6.8 below.

Before we present our reduction procedure, we first discuss a commonly used shifting technique. Suppose[3] $0 < s_0 \in \mathbb{R}$ is an arbitrary but otherwise fixed expansion point. Let

$$A_{s_0} = M_{s_0}^{-1} J_n, \quad \boldsymbol{b}_{s_0} = -M_{s_0}^{-1}\boldsymbol{q}, \tag{6.1}$$

where M_{s_0} is defined by (2.1). Write $s = s_0 + \sigma$. We have[4]

$$\begin{aligned} \boldsymbol{x}(s) &= -(M - sJ_n)^{-1}\boldsymbol{q} \\ &= -(M_{s_0} - \sigma J_n)^{-1}\boldsymbol{q} \\ &= -(I_n - \sigma M_{s_0}^{-1} J_n)^{-1}(M_{s_0}^{-1}\boldsymbol{q}) \\ &= \quad (I_n - \sigma A_{s_0})^{-1}\boldsymbol{b}_{s_0}, \end{aligned}$$

[3] As far as the Arnoldi process in Algorithm 1 and the approximation property in Theorem 6.1 are concerned, it is not necessary to have $s_0 \in \mathbb{R}$, not to mention $s_0 > 0$. We are making this assumption here mainly because only such an s_0 interests us because it is intended to approximate s_* in the end.

[4] Because of $s = s_0 + \sigma$, a function in s is a function in σ, too, and vice versa. For this reason, we conveniently write $h(s)$, when regarded as a function in σ, as $g(\sigma)$. Similarly for their reduced ones, we have notation $h_\ell(s)$ and $g_\ell(\sigma)$.

ALGORITHM 1. THE ARNOLDI PROCESS.

Given s_0 which is neither a pole nor a zero of $h(s)$, this procedure computes an orthonormal basis matrix of $\mathcal{K}_\ell(A_{s_0}, \boldsymbol{b}_{s_0})$, where A_{s_0} and $\boldsymbol{b}_{s_0}$ are defined by (6.1).

1: solve $M_{s_0}\boldsymbol{b}_{s_0} = -\boldsymbol{q}$ for $\boldsymbol{b}_{s_0}$;
2: $\boldsymbol{y}_1 = \boldsymbol{b}_{s_0}/\|\boldsymbol{b}_{s_0}\|_2$; $Y_1 = [\boldsymbol{y}_1]$;
3: **for** $j = 1, 2, \ldots, \ell$ **do**
4: solve $M_{s_0}\boldsymbol{w} = J_n\boldsymbol{y}_j$ for $\boldsymbol{w}$;
5: $H_{(1:j,j)} = Y_j^\top \boldsymbol{w}$, $\boldsymbol{w} = \boldsymbol{w} - Y_j H_{(1:j,j)}$;
6: $\beta = \|\boldsymbol{w}\|_2$;
7: **if** $\beta = 0$ **then**
8: **BREAK**;
9: **else**
10: $H_{(j+1,j)} = \|\boldsymbol{w}\|_2$, $\boldsymbol{y}_{j+1} = \boldsymbol{w}/\beta$, $Y_{j+1} = [Y_j, \boldsymbol{y}_{j+1}]$;
11: **end if**
12: **end for**
13: **return** $Y_\ell = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\ell]$, an orthonormal basis matrix of $\mathcal{K}_\ell(A_{s_0}, \boldsymbol{b}_{s_0})$, and H.

$$\begin{aligned} h(s) &= \boldsymbol{x}(s)^\top J_n \boldsymbol{x}(s) \\ &= \boldsymbol{b}_{s_0}^\top (I_n - \sigma A_{s_0})^{-\top} J_n (I_n - \sigma A_{s_0})^{-1} \boldsymbol{b}_{s_0} \\ &=: g(\sigma). \end{aligned}$$

Suppose s_0 is neither a pole nor a zero of $h(s)$ (in other words, $h(s_0) \neq 0$ and is finite). This implies $h(s_0) = \boldsymbol{b}_{s_0}^\top J_n \boldsymbol{b}_{s_0} \neq 0$. Our reduction process begins with the Arnoldi process as in Algorithm 1 to generate the Krylov subspace $\mathcal{K}_\ell(A_{s_0}, \boldsymbol{b}_{s_0})$.

It can be seen that if the process does not break down at line 8 till $j = \ell$, then we have in exact arithmetic

$$A_{s_0} Y_\ell = Y_\ell H_\ell + H_{(\ell+1,\ell)} \boldsymbol{y}_{\ell+1} \boldsymbol{e}_\ell^\top \quad \text{and} \quad Y_\ell^\top A_{s_0} Y_\ell = H_\ell, \tag{6.2}$$

where $Y_\ell = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\ell]$, $H_\ell := H_{(1:\ell,1:\ell)} \in \mathbb{R}^{\ell \times \ell}$ is an upper Hessenberg matrix. Now, our reduced $h_\ell(s)$ is given by

$$h_\ell(s) = \|\boldsymbol{b}_{s_0}\|_2^2 \boldsymbol{e}_1^\top (I_\ell - \sigma H_\ell)^{-\top} Y_\ell^\top J_n Y_\ell (I_\ell - \sigma H_\ell)^{-1} \boldsymbol{e}_1 =: g_\ell(\sigma). \tag{6.3}$$

The following theorem implies that the Taylor series of $h_\ell(s)$ at s_0 matches that of $h(s)$ also at s_0 in their first ℓ terms. This moment matching property is also a generalization of the one for the Lanczos algorithm of [7]. Theorem 6.1 can be proved in a similar way to that in [38] and the reader is referred to [61] for detail.

THEOREM 6.1. *Suppose Algorithm* 1 *runs to its completion without breakdown to produce* H_ℓ. *Let* $h(s)$ *be defined by* (4.2) *and* $h_\ell(s)$ *by* (6.3). *Then*

$$h(s) = h_\ell(s) + O(|s - s_0|^\ell). \tag{6.4}$$

Remark 6.2. By comparing our factor-form-preserving reduction procedure with the two existing treatments, i.e., linearization+reduction via the bi-orthogonalization Lanczos process or reduction via SOAR, the computational cost of ours is lower in order to obtain the same order reduction. In particular, it can be seen that only one linear system of size n is needed for each order of accuracy achieved in our factor-form-preserving reduction procedure. For the storage, since both our reduction

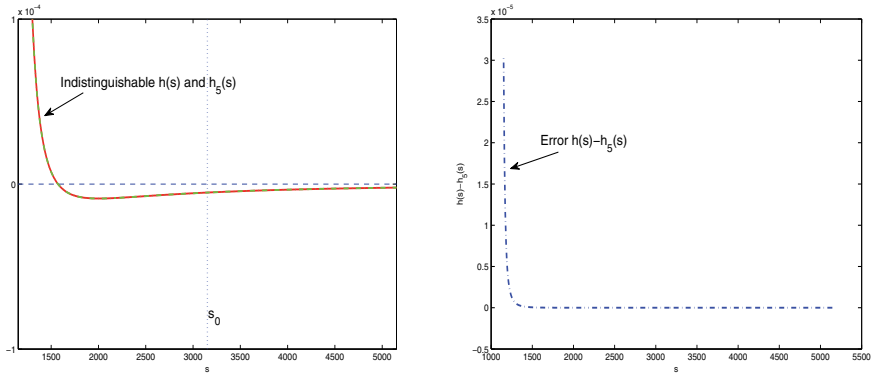

FIG. 2. *Example* 6.3*:* $h(s)$ versus $h_\ell(s)$. Left: *indistinguishable* $h(s)$ *and* $h_5(s)$. Right: *error* $h(s) - h_5(s)$.

procedure and SOAR are based on the Arnoldi process, the storage requirements are almost the same. The linearization+reduction via the bi-orthogonalization Lanczos process is based on the three-term recurrence, and the storage is smaller. Serious breakdowns may occur in the two existing methods, and that makes them less stable or more involved if the look-ahead strategy [47] is used as a cure.[5] On the contrary, a breakdown (i.e., $\beta = 0$ occurs at line 7) in the Arnoldi process in Algorithm 1 is always welcome, because it then finds an invariant subspace and thus makes $h(s) \equiv h_\ell(s)$.

Example 6.3. As an illustration, we test this reduction on $M \in \mathbb{R}^{66\times 66}$ of BCSSTK02 from the matrix market.[6] This M is symmetric positive definite. Although it is small and dense, the resulting reduction behavior is rather representative. We choose $\boldsymbol{q} = [1, \ldots, 1]^\top \in \mathbb{R}^{66}$. Applying Algorithm 1 with $s_0 = \frac{\|M\|_1}{10} \approx 3.15 \times 10^3$ for ℓ steps, we plot the original function $h(s)$ and the reduced one $h_\ell(s)$. In Figure 2, $h(s)$ versus $h_5(s)$ (left) and the difference $h(s) - h_5(s)$ (right) are plotted. Both curves in the left plot are visually indistinguishable. The right plot shows that the error $h(s) - h_5(s)$ is in the order of $O(10^{-5})$, a remarkable accuracy of approximations with just $\ell = 5$.

6.2. Approximations for poles and zeros. We have seen in Example 6.3 that the reduced $h_\ell(s)$ matches $h(s)$ for a long range of s as often it is in model order reduction (see, e.g., [5, 17, 27, 51]). Therefore, we can approximate s_* by some zero of $h_\ell(s)$. Denoting $T_\ell = (Y_\ell^\top J_n Y_\ell)^{-1} \in \mathbb{R}^{\ell\times\ell}$, we know that

$$h_\ell(s) = g_\ell(\sigma) = \|\boldsymbol{b}_{s_0}\|_2^2 \boldsymbol{e}_1^\top \left[\sigma^2 H_\ell T_\ell H_\ell^\top - \sigma(T_\ell H_\ell^\top + H_\ell T_\ell) + T_\ell\right]^{-1} \boldsymbol{e}_1. \tag{6.5}$$

Let $K_\ell = H_\ell T_\ell H_\ell^\top$ and $L_\ell = T_\ell H_\ell^\top + H_\ell T_\ell$, and partition

$$Q(\sigma) := \sigma^2 K_\ell - \sigma L_\ell + T_\ell = \begin{array}{c} \\ 1 \\ \ell-1 \end{array} \begin{array}{c} \begin{array}{cc} 1 & \ell-1 \end{array} \\ \begin{bmatrix} \alpha & \boldsymbol{a}^\top \\ \boldsymbol{a} & \widehat{Q} \end{bmatrix} \end{array}.$$

[5] We have done preliminary numerical experiments on the bi-orthogonalization Lanczos process with look-ahead strategy [47], but the numerical results show the approach is not better and yet is less stable than our factor-form-preserving reduction procedure.

[6] http://math.nist.gov/MatrixMarket/.

It can be seen from the expression of $h_\ell(s)$ in (6.5) that $h_\ell(s)$ can be expressed as

$$h_\ell(s) = g_\ell(\sigma) = \frac{\|\boldsymbol{b}_{s_0}\|_2^2}{\alpha - \boldsymbol{a}^\top \widehat{Q}^{-1}\boldsymbol{a}}$$

from which we know that the zeros of $h_\ell(s)$ correspond to some of the eigenvalues of the quadratic eigenvalue problem (QEP):

$$\widehat{Q}(\sigma)\hat{\boldsymbol{z}} := (\sigma^2 \widehat{K}_\ell - \sigma \widehat{L}_\ell + \widehat{T}_\ell)\hat{\boldsymbol{z}} = 0, \tag{6.6}$$

where $\widehat{K}_\ell = K_{\ell;(2:\ell,2:\ell)}$, $\widehat{L}_\ell = L_{\ell;(2:\ell,2:\ell)}$, and $\widehat{T}_\ell = T_{\ell;(2:\ell,2:\ell)}$. Depending on $\boldsymbol{a}$, some eigenvalues of this QEP may not be zeros of $h_\ell(s)$, but generically, the zeros of $h_\ell(s)$ and the eigenvalues of QEP (6.6) are the same. Basing on Theorems 4.1 and 2.4, we choose one eigenvalue μ satisfying the three conditions

$$\begin{cases} s_1 = \mathrm{Re}(\mu) + s_0 > 0, \\ |\textsc{Im}\,(\mu)| \le 10^{-6}, \\ \text{the first entry } x_1(s_1) \text{ of } \boldsymbol{x}(s_1) = -(M - s_1 J_n)^{-1}\boldsymbol{q} \text{ is positive} \end{cases} \tag{6.7}$$

and use $s_1 = \mathrm{Re}(\mu) + s_0$ as the approximation of s_*, where $\textsc{Re}\,(\mu)$ and $\textsc{Im}\,(\mu)$ stand for the real part and the imaginary part of μ, respectively. According to Theorem 5.2, we know that for a sufficiently good approximation $h_\ell(s)$, likely there is only one s_1 that satisfies these conditions. Furthermore, since $\widehat{Q}(\sigma)$ is in general small in size, the cost solving the QEP $\widehat{Q}(\sigma)\hat{\boldsymbol{z}} = 0$ can be regarded as of $O(1)$.

The poles of $h(s) = (J_n\boldsymbol{q})^\top (MJ_n - sI_n)^{-\top}(MJ_n - sI_n)^{-1}\boldsymbol{q}$ correspond to some of the eigenvalues of MJ_n. Depending on $\boldsymbol{q}$, some eigenvalues of MJ_n may give removable poles of $h(s)$, but generically, the poles and the eigenvalues are the same. By Theorem 2.4, we know that if M has the GUS property, τ is the unique positive pole of $h(s)$ (see Theorem 5.2 for more detailed discussions on the pole τ). This observation suggests that we can obtain an approximation of τ by finding the positive pole(s) of $h_\ell(s)$, if any. From (6.3), the poles τ_1 of $h_\ell(s)$ relate to the eigenvalues ν of H_ℓ by $\tau_1 = s_0 + 1/\nu$. Note that H_ℓ might have complex eigenvalues, and as a practical strategy, we choose

$$\eta = \max\left\{\textsc{Re}\,(\mu) \,:\, 1/\mu \in \mathrm{eig}(H_\ell),\ |\textsc{Im}\,(\mu)| \le 10^{-6}\right\} \tag{6.8}$$

and use $\tau_1 = \eta + s_0$ as an approximation of τ.

To demonstrate the effectiveness of the above-mentioned ways for approximating the poles and zeros of $h(s)$, we now revisit Example 6.3 (more numerical results will be reported in section 8).

Example 6.4. Continuing with Example 6.3, Figure 3 compares $h_5(s)$ with $h(s)$ over an even larger range of s to the left of s_0 to include a pole of $h_5(s)$ with $h(s)$. Now, by computing the eigenvalues of $Q(\sigma)$ and $\widehat{Q}(\sigma)$, we find the approximation $\tau_1 \approx 1.0992 \times 10^3$ of τ and the approximation $s_1 \approx 1.5721 \times 10^3$ of s_* with $h(s_1) \approx 2.82 \times 10^{-8}$. Interestingly, Figure 3 shows another zero around $\hat{s}_1 = 8.3912 \times 10^2$ of $h(s)$. This zero is not the solution for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ since the first element of $\boldsymbol{x}(\hat{s}_1) = -(M - s_1 J_n)^{-1}\boldsymbol{q}$ is negative, i.e., $\boldsymbol{x}(\hat{s}_1) \notin \partial(\mathbb{K}^n)$. Indeed, this is an example for case 5 in Theorem 5.2: 8.3912×10^2 and 1.5721×10^3 are the approximations to $s_{*;1}$ and $s_{*;2}$, respectively, but $s_* = s_{*;2}$ is the desired one. So this example falls into case 5 there.

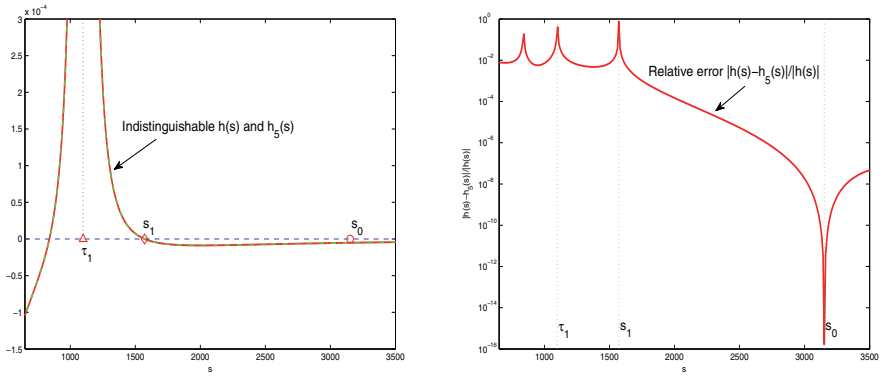

FIG. 3. *Example* 6.4. Left: *indistinguishable* $h(s)$ *and* $h_5(s)$ *(with limited y-axis range so that the useful details can be recognized), the expansion point* s_0*, and the approximate zero* s_1 *and pole* τ_1 *by* $h_5(s)$. Right: *small relative error* $|h(s)-h_5(s)|/|h_5(s)|$ *away from zeros and poles over a wide range of* s.

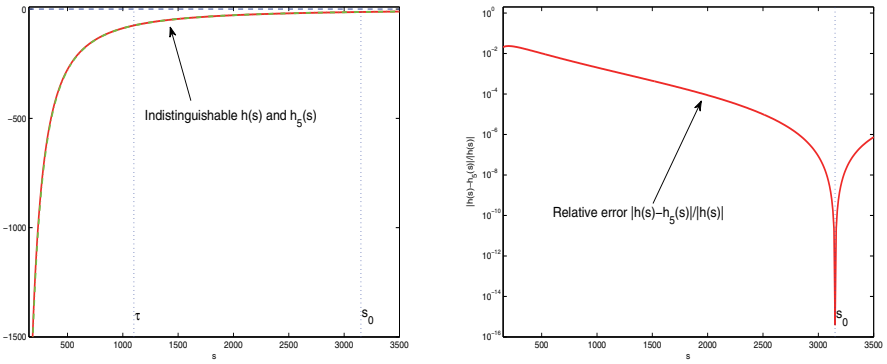

FIG. 4. *Example* 6.5. *The special* $s_* = \tau$ *which is no longer a pole.* Left: *indistinguishable* $h(s)$ *and* $h_5(s)$. Right: *relative error* $|h(s) - h_5(s)|/|h_5(s)|$.

Next we look at an example for the special case $s_* = \tau$. Theorem 5.4 claims that $h(s) < 0$ for $0 < s \neq \tau$ and Theorem 5.5 says τ is no longer a pole if τ is a simple eigenvalue of MJ_n.

Example 6.5. We again use the matrix $M \in \mathbb{R}^{66\times 66}$ of BCSSTK02 as an illustration. To construct the special case, we first compute the largest eigenvalue $\tau \approx 1.0996 \times 10^3$ of MJ_n, and then generate the solution

$$\boldsymbol{x} = [\sqrt{n-1}, 1, \cdots, 1]^\top \in \mathbb{R}^{66},$$

and finally set $\boldsymbol{q} = -(M - \tau J_n)\boldsymbol{x}$. The function $h(s)$ and the reduced $h_5(s)$ associated with the example are plotted in Figure 4, which shows that indeed $h(s) < 0$ in the plotted range of $s > 0$ and τ is not a pole any more, and also $h_5(s)$ has no poles on $(0, \infty)$. This would be the case for which [59, Algorithm 2] is triggered to solve this SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$.

6.3. The reduced SOCLCP. In this subsection, we will establish an interesting connection of the factor-form-preserving reduced $h_\ell(s)$ to an SOCLCP in the form of (1.1) of a much smaller size. To this end, we first establish the following lemma.

LEMMA 6.6. *Let $Y_\ell = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\ell] \in \mathbb{R}^{n\times\ell}$ $(1 \le \ell < n)$ be the orthonormal basis matrix by Algorithm* 1*, i.e., $Y_\ell^\top Y_\ell = I_\ell$. Then $Y_\ell^\top J_n Y_\ell$ has an eigenvalue $\varrho := 2\|Y_{\ell;(1,:)}\|_2^2 - 1 \in [-1, 1]$, and all other eigenvalues are -1.*

Proof. Note $J_n = 2\boldsymbol{e}_1\boldsymbol{e}_1^\top - I_n$ and $Y_\ell^\top Y_\ell = I_\ell$. We have

$$Y_\ell^\top J_n Y_\ell = 2Y_\ell^\top \boldsymbol{e}_1\boldsymbol{e}_1^\top J_n Y_\ell - I_\ell = 2Y_{\ell;(1,:)}^\top Y_{\ell;(1,:)} - I_\ell,$$

where $Y_{\ell;(1,:)}$ is the first row of Y_ℓ. Hence, $Y_\ell^\top J_n Y_\ell$ has an eigenvalue $\varrho = 2\|Y_{\ell;(1,:)}\|_2^2 - 1$, and all other eigenvalues are -1. Due to $1 \le \ell < n$, we have that $0 \le \|Y_{\ell;(1,:)}\|_2^2 \le 1$ and thus $-1 \le \varrho \le 1$. ∎

Since Y_ℓ has orthogonal columns, $\varrho > 0$ for sufficiently large ℓ. Whenever $\varrho > 0$, the matrix $Y_\ell^\top J_n Y_\ell$ admits the following decomposition:

$$\begin{aligned} Y_\ell^\top J_n Y_\ell &= P_\ell \operatorname{diag}(\varrho, -I_{\ell-1}) P_\ell^\top \qquad \text{(eigendecomposition)} \\ &= P_\ell \operatorname{diag}(\sqrt{\varrho}, I_{\ell-1}) J_\ell \operatorname{diag}(\sqrt{\varrho}, I_{\ell-1}) P_\ell^\top, \end{aligned} \tag{6.9}$$

where $P_\ell \in \mathbb{R}^{\ell\times\ell}$ is orthogonal, i.e., $P_\ell^\top P_\ell = I_\ell$. Substituting (6.9) into $h_\ell(s)$ yields

$$\begin{aligned} h_\ell(s) &= \|\boldsymbol{b}_{s_0}\|_2^2 \boldsymbol{e}_1^\top (\widehat{P}_\ell - \sigma H_\ell \widehat{P}_\ell)^{-\top} J_\ell (\widehat{P}_\ell - \sigma H_\ell \widehat{P}_\ell)^{-1} \boldsymbol{e}_1 \\ &= \|\boldsymbol{b}_{s_0}\|_2^2 \boldsymbol{q}_\ell^\top (M_\ell - \sigma J_\ell)^{-\top} J_\ell (M_\ell - \sigma J_\ell)^{-1} \boldsymbol{q}_\ell, \end{aligned} \tag{6.10}$$

which corresponds to SOCLCP$(M_\ell, \mathbb{K}^\ell, \boldsymbol{q}_\ell)$, where $\widehat{P}_\ell = P_\ell \operatorname{diag}(1/\sqrt{\varrho}, I_{\ell-1})$, and

$$M_\ell = J_\ell \widehat{P}_\ell^{-1} H_\ell^{-1} \widehat{P}_\ell \quad \text{and} \quad \boldsymbol{q}_\ell = J_\ell \widehat{P}_\ell^{-1} H_\ell^{-1} \boldsymbol{e}_1. \tag{6.11}$$

THEOREM 6.7. *In Lemma* 6.6*, if $\varrho > 0$, then h_ℓ corresponds to* SOCLCP$(M_\ell, \mathbb{K}^\ell, \boldsymbol{q}_\ell)$ *with M_ℓ and $\boldsymbol{q}_\ell$ given by* (6.11).

Remark 6.8. As was explained, by connecting the function $h_\ell(s)$ with an SOCLCP in the form of (1.1), we know from (6.10) that solving $h_\ell(s) = g_\ell(\sigma) = 0$ is equivalent to solving a much smaller SOCLCP$(M_\ell, \mathbb{K}^\ell, \boldsymbol{q}_\ell)$. Therefore, if we have an efficient algorithm for solving small- to medium-size SOCLCPs in the form of (1.1), $h_\ell(s) = 0$ can also be tackled efficiently. From this point of view, *our factor-form-preserving reduction procedure can be viewed as a projection technique that projects the original large-scale* SOCLCP (1.1) *to a smaller* SOCLCP (1.1) *with the property that the Taylor series of the corresponding functions $h(s)$ and $h_\ell(s)$ at the expansion point match till the ℓth-order term.*

6.4. Algorithmic framework: LCPvA. Our foregoing discussions naturally lead to our main algorithmic framework to solve SOCLCP (1.1) by Krylov subspace reduction via the Arnoldi process. It is outlined in Algorithm 2. However, in order to make the algorithm more efficient and stable, there are still some issues to be addressed. First, it is noted that at line 8 in Algorithm 2, the iterate $\boldsymbol{x}(s_{k+1})$ is the solution of a linear system, and a certain iterative solver will have to be employed for extremely large-scale problems. A question as to how accurately the linear system should be solved naturally arises, in order to make sure the stopping criteria at line 9 are satisfiable at convergence.

Suppose that we have an approximation $\hat{s} \approx s_*$ such that the relative error

$$\frac{\|\boldsymbol{x}(\hat{s}) - \boldsymbol{x}(s_*)\|_2}{\|\boldsymbol{x}(\hat{s})\|_2} \le \epsilon_1. \tag{6.12}$$

ALGORITHM 2. LCPvA: LINEAR COMPLEMENTARITY PROBLEM VIA THE ARNOLD PROCESS FOR SOCLCP($M, \mathbb{K}^n, \boldsymbol{q}$).

input: $M, \boldsymbol{q}$, and $k_{\max}$ (maximum number of outer iterations allowed);
output: an approximation to $\boldsymbol{x} \in \mathrm{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$.

if $\boldsymbol{q} \in \mathbb{K}^n$ **then** $\boldsymbol{x} = \boldsymbol{0}$ and **return**;
if $-M^{-1}\boldsymbol{q} \in \mathbb{K}^n$ **then** $\boldsymbol{x} = -M^{-1}\boldsymbol{q}$ and **return**;
select s_0 and set $k = -1$;
repeat
  $k = k + 1$;
  apply Algorithm 1 (with full reorthogonalization) to generate H_ℓ and Y_ℓ associated with $\mathcal{K}_\ell(A_{s_k}, \boldsymbol{b}_{s_k})$;
  solve QEP (6.6) for the approximation s_{k+1} of s_* according to (6.7);
  solve $(M - s_{k+1}J_n)\boldsymbol{x}(s_{k+1}) = -\boldsymbol{q}$ for $\boldsymbol{x}(s_{k+1})$;
  if $|h(s_{k+1})| \le \epsilon \|\boldsymbol{x}(s_{k+1})\|_2^2$ and $x_1(s_{k+1}) > 0$ **then**
    return $\boldsymbol{x}(s_{k+1})$ as an approximate solution of SOCLCP($M, \mathbb{K}^n, \boldsymbol{w}$), and LCPvA **stops**;
  end if
until $k \ge k_{\max}$;
if $k \ge k_{\max}$ and there is no positive zero of $h_\ell(s)$ **then**
  call [59, Algorithm 2] for the special case $s_* = \tau$;
else
  LCPvA fails to find an approximate solution.
end if

Technically, $\|\boldsymbol{x}(\hat{s})\|_2$ in the denominator in (6.12) should be replaced by $\|\boldsymbol{x}(s_*)\|_2$. But it is unknown and thus numerically impractical to use. Since at convergence $\|\boldsymbol{x}(\hat{s})\|_2$ and $\|\boldsymbol{x}(s_*)\|_2$ are very much the same for the purpose of monitoring errors, which one of $\|\boldsymbol{x}(\hat{s})\|_2$ and $\|\boldsymbol{x}(s_*)\|_2$ gets used does not matter much. In practice, $\boldsymbol{x}(\hat{s})$ has to be computed by solving $M_{\hat{s}}\boldsymbol{x}(\hat{s}) = -\boldsymbol{q}$, where $M_{\hat{s}} = M - \hat{s}J_n$. Suppose $\hat{\boldsymbol{x}}$ is a computed approximation to $\boldsymbol{x}(\hat{s})$, and let $\boldsymbol{r}$ be the associated residual vector

$$\boldsymbol{r} = M_{\hat{s}}\hat{\boldsymbol{x}} + \boldsymbol{q}.$$

In other words, $\hat{\boldsymbol{x}}$ is the exact solution of $M_{\hat{s}}\boldsymbol{x}(\hat{s}) = -\boldsymbol{q} + \boldsymbol{r}$. By the standard perturbation theory for linear system [16], we get

$$\frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}(\hat{s})\|_2}{\|\boldsymbol{x}(\hat{s})\|_2} \le \kappa(M_{\hat{s}})\epsilon_2, \tag{6.13}$$

where $\kappa(M_{\hat{s}}) = \|M_{\hat{s}}\|_2\|M_{\hat{s}}^{-1}\|_2$ is the spectral condition number of $M_{\hat{s}}$, and $\epsilon_2 = \frac{\|\boldsymbol{r}\|_2}{\|\boldsymbol{q}\|_2}$. Hence

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}(s_*)\|_2 \le \|\hat{\boldsymbol{x}} - \boldsymbol{x}(\hat{s})\|_2 + \|\boldsymbol{x}(\hat{s}) - \boldsymbol{x}(s_*)\|_2 \le \Big[\kappa(M_{\hat{s}})\epsilon_2 + \epsilon_1\Big]\|\boldsymbol{x}(\hat{s})\|_2.$$

Using $h(s_*) = 0$, we have

$$\hat{\boldsymbol{x}}^\top J_n\hat{\boldsymbol{x}} = h(s_*) + 2[\hat{\boldsymbol{x}} - \boldsymbol{x}(s_*)]^\top J_n\boldsymbol{x}(s_*) + O(\epsilon_{\max}^2) = 2[\hat{\boldsymbol{x}} - \boldsymbol{x}(s_*)]^\top J_n\boldsymbol{x}(s_*) + O(\epsilon_{\max}^2),$$

where $\epsilon_{\max} = \max\{\epsilon_1, \epsilon_2\}$. Therefore

$$|\hat{\boldsymbol{x}}^\top J_n\hat{\boldsymbol{x}}| \le 2\Big[\kappa(M_{\hat{s}})\epsilon_2 + \epsilon_1\Big]\|\boldsymbol{x}(\hat{s})\|_2^2 + O(\epsilon_{\max}^2). \tag{6.14}$$

Now we are ready to make practical implementation suggestions on lines 8 and 9. Since (6.12) is about the limit of the approximation in theory, we may let ϵ_1 be as tiny as necessary for our analysis. But in view of (6.14), we assume $\epsilon_1 \ll \kappa(M_{\hat{s}})\epsilon_2$. Further, to assume ϵ_2 is sufficiently tiny such that $\|\hat{\boldsymbol{x}}\|_2 \approx \|\boldsymbol{x}(\hat{s})\|_2$, we see practically that (6.14) becomes

$$|\hat{\boldsymbol{x}}^\top J_n \hat{\boldsymbol{x}}| \le 2\kappa(M_{\hat{s}})\epsilon_2 \|\hat{\boldsymbol{x}}\|_2^2 + O(\epsilon_2^2). \tag{6.15}$$

Together, both (6.13) and (6.15) suggest that for the test at line 9 to be satisfiable at convergence, the linear system at line 8 should be solved to the level of accuracy such that

$$\epsilon_2 = \frac{\|\boldsymbol{r}\|_2}{\|\boldsymbol{q}\|_2} \le \frac{\epsilon}{2\kappa(M_{s_{k+1}})}. \tag{6.16}$$

Unfortunately, $\kappa(M_{s_{k+1}})$ is an unknown quantity whose accurate estimation is nontrivial and costly [16]. The good thing is that there is no need to do an accurate estimation. Usually a very rough one is good enough. For Algorithm 2, we suggest using the maximum of all $\|\boldsymbol{q}\|_2/\|\hat{\boldsymbol{x}}\|_2$ available so far in the loop as a rough estimate of $\|M_{s_{k+1}}^{-1}\|_2$ in (6.16), where $\hat{\boldsymbol{x}}$ denotes any one of the already computed approximation solutions at line 8.

Other issues for improving the efficiency and stability of Algorithm 2 are discussed in the next two subsections: in subsection 6.5, we shall suggest a strategy to improve the efficiency for small- to medium-size dense problems, and in subsection 6.6, we will discuss the stability issue for large-scale SOCLCP.

6.5. Transform dense SOCLCP to upper Hessenberg SOCLCP. The matrix M in this subsection is assumed to be of small to medium sizes and is dense. We note that the main computationally heavy part of Algorithm 1 lies at its line 4:

$$\text{solve } M_{s_0}\boldsymbol{w} = J_n \boldsymbol{y}_j \text{ for } \boldsymbol{w}. \tag{6.17}$$

These are linear systems like $M_{s_0}\boldsymbol{t} = \mathbf{r}$ and usually cost $O(n^3)$ flops for every first use of a new expansion point s_0 and $O(n^2)$ flops afterward. Since s_0 will have to be updated as it has to move toward s_*, linear systems $M_{s_0}\boldsymbol{t} = \mathbf{r}$ will have to be solved for a few different s_0, and thus they could be computationally expensive.

Previously, we mentioned that if M is in the upper/lower Hessenberg form, the upper/lower triangular form included, then (6.17) takes only $O(n^2)$ flops to compute, independently of s_0. This is good.

But what happens when M is not in the upper/lower Hessenberg form? When the size of M is modest, say, up to a few thousands, and M is also dense, we can preprocess SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ to transform it into one for which M is in the upper Hessenberg form. Here is the detail. Find $Q = \text{diag}(1, Q_0)$ with orthogonal $Q_0 \in \mathbb{R}^{(n-1)\times(n-1)}$ such that $Q^\top MQ$ is upper Hessenberg. Such a transformation from M to $Q^\top MQ$ is not new and in fact it is the first step by the name of *Hessenberg reduction* in solving a dense eigenvalue problem by the QR algorithm [16]. Important for us here is that it does not change the second-order cone $\mathbb{K}^n$: $Q\mathbb{K}^n = \mathbb{K}^n$ and at the same time $Q^\top J_n Q = J_n$. Therefore, we can solve SOCLCP$(Q^\top MQ, \mathbb{K}^n, Q^\top \boldsymbol{q})$ instead: any $\boldsymbol{z} \in \text{SOL}(Q^\top MQ, \mathbb{K}^n, Q^\top \boldsymbol{q})$ gives $\boldsymbol{x} = Q\boldsymbol{z} \in \text{SOL}(M, \mathbb{K}^n, \boldsymbol{q})$ and vice versa.

It should be pointed out that the Hessenberg reduction itself, although only needed to be done once, still costs $O(n^3)$ flops, which can be too expensive to be practical for a very-large-scale problem. Nevertheless, for large-scale and sparse M,

the involved linear systems can be solved iteratively through exploiting the sparsity of M or fast matrix-vector multiplications by M (at a cost usually much less than $O(n^2)$ flops that is needed for a dense M).

Comparing with BN in [59], the new approach has the following noticeable advantage: it does not need to compute the eigenpair $(\tau, \boldsymbol{v})$ of $M^\top J_n$, while the computational complexity is almost the same as that of BN.

6.6. Reorthogonalization to prevent loss of orthogonality. A potential problem in the Arnoldi process is loss of orthogonality. There are extensive discussions on this issue, for example, in [16, Chapter 7] and [46] and the references therein. It is argued that the loss of orthogonality does not cause the algorithm to behave completely unpredictably. Indeed, loss of orthogonality could result in multiple copies of the same eigenvalues. There are also several cures for such loss of orthogonality, and the Arnoldi algorithm with full reorthogonalization is one. Another alternative is selective reorthogonalization [46] by orthogonalizing any new Arnoldi vector against already converged Ritz vectors. Our choice is simply to do full reorthogonalization. Our goal is to find the zero s_* of $h(s)$ by iteratively (the outer loop) choosing new and better shift s_0 obtained from the eigenvalues of the QEP for $\widehat{Q}(\sigma)$ associated with the previous Arnoldi process; from this point of view, we do not need Krylov subspaces of very large dimension in each outer loop, and therefore cost for doing full reorthogonalization is usually manageable.

7. Applications. We remarked in Remark 5.6 that when M is symmetric positive definite, SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ is equivalent to the conic programming (1.2)

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

in the sense that SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ is the KKT condition of (1.2) and the KKT point is unique. This observation opens the door to use of our algorithm LCPvA as building blocks to solve an important and more general programming in optimization—*quadratic programming over a second-order cone* (QP-SOC):

$$\min_{A\boldsymbol{x}=\boldsymbol{b},\ \boldsymbol{x}\in\mathbb{K}^n} \frac{1}{2}\boldsymbol{x}^\top W\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x}. \tag{7.1}$$

The traditionally well-known second-order cone programming [2, 22, 40] in which $W = 0$ is a special case; see also [28] and the webpage of CVXOPT.[7] Two other practical examples of QP-SOC are as follows:

(a) *Lorentz-copositivity* (see [19, Proposition 4.1], [41, Definition 2.1], [1, 32, 52]): A matrix $W \in \mathbb{R}^{n\times n}$ is said to be *Lorentz-copositive* if $\boldsymbol{x}^\top W\boldsymbol{x} \geq 0$ for $\boldsymbol{x} \in \mathbb{K}^n$. The Lorentz-copositivity is one kind of extension of the classical and well-known *copositivity* [32, Definition 1.1]. It can be seen that testing whether W is Lorentz-copositive is equivalent to checking whether

$$\left\{ \min_{\boldsymbol{e}_1^\top \boldsymbol{x}=1,\ \boldsymbol{x}\in\mathbb{K}^n} \frac{1}{2}\boldsymbol{x}^\top W\boldsymbol{x} \right\} \geq 0.$$

(b) *The trust-region subproblem (TRS)*: For simplicity, we consider

$$\min_{\|\boldsymbol{y}\|_2\leq\delta} \frac{1}{2}\boldsymbol{y}^\top H\boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{p}. \tag{7.2}$$

[7] http://cvxopt.org/userguide/coneprog.html.

By introducing $\boldsymbol{x} = \begin{bmatrix} \beta \\ \boldsymbol{y} \end{bmatrix}$ and $W = \begin{bmatrix} \alpha & \boldsymbol{p}^\top \\ \boldsymbol{p} & H \end{bmatrix}$ for any given α, we see that (7.2) is equivalent to

$$\min_{\boldsymbol{e}_1^\top \boldsymbol{x} = \delta,\ \boldsymbol{x} \in \mathbb{K}^n} \frac{1}{2} \boldsymbol{x}^\top W \boldsymbol{x}. \tag{7.3}$$

This reformulation (7.3) of TRS is the so-called parameterized eigenvalue problem in [49, 50], and the LSTRS [49, 50] method for TRS (7.2) is proposed based on this equivalent formulation.

Now we outline how to use our LCPvA as building blocks to solve QP-SOC (7.1). To this end, we introduce the well-known augmented Lagrangian methods (ALM) proposed by Hestenes [30] and Powell [48] (see also [8, chapter 2] and [43, chapter 17.3]) in optimization. It starts by the so-called augmented Lagrangian function:

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

where $\mu > 0$ is the penalty parameter. One way of using ALM to solve (7.1) is the following (e.g., [43, Framework 17.3] and [8, Chapter 2]): given initial guess $\mu_0 > 0$, and λ_0,

1: **repeat**
2: find $\boldsymbol{x}_k = \arg\min_{\boldsymbol{x} \in \mathbb{K}^n} \mathscr{L}_A(\boldsymbol{x}, \lambda_k; \mu_k)$;
3: $\lambda_{k+1} = \lambda_k - \mu_k (A\boldsymbol{x}_k - \boldsymbol{b})$;
4: choose new penalty parameter $\mu_{k+1} \geq \mu_k$;
5: **until** convergence

There are some fairly well-established convergence analyses for it, e.g., in [8, Propositions 2.1–2.3] and [43, Chapter 17.3]. The key in this framework is the computation of

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x} \in \mathbb{K}^n} \Big\{ \frac{1}{2} \boldsymbol{x}^\top \underbrace{(W + \mu_k A^\top A)}_{=:M} \boldsymbol{x} + \boldsymbol{x}^\top \underbrace{(\boldsymbol{c} - A^\top \lambda_k - A^\top \boldsymbol{b})}_{=:\boldsymbol{q}} \Big\},$$

which takes the form of (1.2), leading to SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ that can be solved by LCPvA efficiently provided M has the GUS property. We argue that this is often the case. In fact, for a convex QP-SOC (i.e., W is positive definite), $M = W + \mu_k A^\top A$ is also positive definite and thus has the GUS property. In general, because $\mu_k A^\top A$ is positive semidefinite for $\mu_k > 0$ and μ_k increases with k, even for a nonconvex QP-SOC, after many iterations M may be positive definite and thus also has the GUS property.

Finally, the linear complementarity problem (1.3) over multiple second-order cones is also the KKT system of the minimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{K}_{\times m}} \frac{1}{2} \boldsymbol{x}^\top M \boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x}. \tag{7.4}$$

The latter (7.4) is the so-called second-order conic-constrained quadratic programming in [21, (1a)–(1b) and (2a)–(2b)]. In the framework of the matrix-splitting method [60], each iteration for (1.3) can be broken down into solving m single cone SOCLCP that can be solved by LCPvA.

TABLE 2
Parameters in LCPvA.

Parameter	Value	Description
ε	$10^{-7} \sim 10^{-4}$	Stopping criterion at line 9 in Algorithm 2
s_0	$\frac{\|\|M\|\|_1}{5}$	Initial shift in Algorithm 2
$k_{\max}$	≥ 3	Maximal number of iterations for the outer loop of Algorithm 2
$\text{iter}_{\text{Arnoldi}}$	$\mathbb{Z}^{k_{\max}}$	$\text{iter}_{\text{Arnoldi}}(i)$ is the number of Arnoldi steps in the ith outer loop of Algorithm 2

8. Numerical experiments. We coded Algorithm 2 (LCPvA) in MATLAB 7.13.0 (R2011b) and tested it against other efficient algorithms on an iMac ME086CH/A with Intel Core i5 at 2.7 GHz and 8 GB memory. As we shall see, preliminary results demonstrate the high efficiency of LCPvA for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$.

We divide our experiments into two parts: (1) numerical tests for (medium-sized) dense M and (2) for large-scale sparse M. In our current implementation, for dense M, we preprocess the associated SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ as we discussed in subsection 6.5 to make sure all subsequent linear systems are solved in $O(n^2)$ flops. But for sparse M, we still use MATLAB's sparse LU decomposition to solve the linear system at line 8 of Algorithm 2 as well as the one at line 4 of Algorithm 1 in which one LU should be computed before the **for**-loop and subsequently we just use triangular system solving. This is because an LU decomposition costs potentially much more than two triangular system solvings. The use of the LU decomposition in the sparse case limits the size of M that we can test on. In the future, we will explore solving these linear systems iteratively to allow even larger M.

All tested M are symmetric and positive definite (implying M has the GUS property) and their condition numbers are varied from 10 to 10^5. As a comparison, we also report the numerical results from three other methods: the BN iteration [59] and two popular MATLAB software packages: SDPT3 [55, 56, 57] (version 4)[8] and SeDuMi [53] (version 1.3). We point out that both SDPT3 and SeDuMi are designed for general semidefinite quadratic linear programming, not particularly targeting SOCLCP in the form of (1.1), but when M is symmetric and positive definite, solving SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ is equivalent to solving the conic programming (1.2). In fact, if M is decomposed into or already takes the form

$$M = \widetilde{M}^\top \widetilde{M}, \tag{8.1}$$

then according to [56, section 4.6] and [53], (1.2) can be converted equivalently to the standard programming problems that can be solved by SDPT3 and SeDuMi (see [56, section 4.6], [53], and also [61, Appendix A]).

We run SDPT3 with its default setups and run the BN iteration with the parameters used in the numerical testing in [59]; for SeDuMi, we set pars.eps $= 10^{-10}$ as the desired accuracy. As for our method LCPvA, the involved parameters and their recommended ranges are summarized in Table 2. The use of the particular s_0 as in the table is drawn from our own experience on numerous tests.

[8]The MATLAB package of SDPT3 is available at www.math.nus.edu.sg/∼mattohkc/sdpt3.html and that of SeDuMi is available at sedumi.ie.lehigh.edu/downloads.

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 three relative errors

$$\chi_{\mathrm{rel}_1} = \frac{\max\{\|\boldsymbol{x}_{(2:n)}\|_2 - x_1, 0\}}{\|\boldsymbol{x}\|_2}, \tag{8.2a}$$

$$\chi_{\mathrm{rel}_2} = \frac{\max\{\|\boldsymbol{g}_{(2:n)}\|_2 - g_1, 0\}}{\|M\|_1\|\boldsymbol{x}\|_2 + \|\boldsymbol{q}\|_2}, \tag{8.2b}$$

$$\chi_{\mathrm{rel}_3} = \frac{|\boldsymbol{x}^\top \boldsymbol{g}|}{\|\boldsymbol{x}\|_2(\|M\|_1\|\boldsymbol{x}\|_2 + \|\boldsymbol{q}\|_2)} \tag{8.2c}$$

and the total relative error

$$\chi_{\mathrm{rel}} = \chi_{\mathrm{rel}_1} + \chi_{\mathrm{rel}_2} + \chi_{\mathrm{rel}_3}, \tag{8.3}$$

where $\boldsymbol{g} = M\boldsymbol{x} + \boldsymbol{q}$. Our rationale in deciding the normalizing factors in (8.2) is the following. Let $\mathrm{fl}(\cdot)$ denote the computed result of an expression and $\mathfrak{u}$ be the machine's unit roundoff. Then for exact $\boldsymbol{x}$ and $\boldsymbol{q}$, we have $\mathrm{fl}(\boldsymbol{g}) = M\boldsymbol{x}+\boldsymbol{q}+O(\mathfrak{u}\,[\|M\|_1\|\boldsymbol{x}\|_2+\|\boldsymbol{q}\|_2])$ and thus

$$\begin{aligned}
\mathrm{fl}(\|\boldsymbol{x}_{(2:n)}\|_2 - x_1) &= \|\boldsymbol{x}_{(2:n)}\|_2 - x_1 + O(\mathfrak{u}\,\|\boldsymbol{x}\|_2),\\
\mathrm{fl}(\|\boldsymbol{g}_{(2:n)}\|_2 - g_1) &= \|\boldsymbol{g}_{(2:n)}\|_2 - g_1 + O(\mathfrak{u}\,[\|M\|_1\|\boldsymbol{x}\|_2 + \|\boldsymbol{q}\|_2]),\\
\mathrm{fl}(\boldsymbol{x}^\top\boldsymbol{g}) &= \boldsymbol{x}^\top\boldsymbol{g} + O(\mathfrak{u}\,\|\boldsymbol{x}\|_2[\|M\|_1\|\boldsymbol{x}\|_2 + \|\boldsymbol{q}\|_2]).
\end{aligned}$$

8.1. Numerical tests for dense M. This subsection is dedicated to numerical tests for medium-scale dense problems. Following [60], in this set of tests, we generate pairs $(M = \widetilde{M}^\top\widetilde{M}, \boldsymbol{q})$, where entries of $\boldsymbol{q}$ are uniformly distributed in $[-1, 1]$ and

$$\widetilde{M} = \mathrm{diag}\left(1, \sqrt{1+\delta}, \sqrt{1+2\delta}, \ldots, \sqrt{1+(n-1)\delta}\right) Q,$$

where $Q = \texttt{orth(randn(n,n))}$ and $\delta = \frac{\texttt{cond}}{n}$. The way we construct M makes it convenient to

1. control the condition number of M, which is $(1 - \frac{1}{n})\texttt{cond} + 1 \approx \texttt{cond}$, and
2. easily formulate equivalent problems that SDPT3 and SeDuMi apply, as both of them work on $\widetilde{M}$, not $M = \widetilde{M}^\top\widetilde{M}$.

To evaluate the performance of the methods, for each n, we vary `cond` from 10 to 10^5 and set $\epsilon = 10^{-7}$ as the stopping criteria at line 9 of Algorithm 2, and we choose $k_{\max} = 3$ and $\mathrm{iter}_{\mathsf{Arnoldi}} = [30, 20, 10]$, meaning that the calls to Algorithm 1 at line 6 of Algorithm 2 are with $\ell = 30$, 20, 10 as $k = 0, 1, 2$, respectively.

For every given pair (n, δ), we generate five random SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ together with randomly generated initial points $\boldsymbol{x}_0$ for BN, SDPT3 and SeDuMi. Feeding these problems into the four methods,[9] BN, SDPT3, SeDuMi, and LCPvA, in Table 3 we report average numbers of iterations (# iter), CPU times (measured by the MATLAB function cputime) and relative errors χ_{rel} defined by (8.2) for each method, over the five random problems for each pair (n, δ). In particular, we mention that # iter for LCPvA is the number of total Arnoldi steps in all calls to Algorithm 1 for each run of LCPvA, and the number of the iterations for BN is listed as $\mathrm{iter}_{\mathrm{bisection}}/\mathrm{iter}_{\mathrm{Newton}}$, where $\mathrm{iter}_{\mathrm{bisection}}$ is the number of the bisection steps and $\mathrm{iter}_{\mathrm{Newton}}$ is that of the Newton steps.

[9] For SDPT3 and SeDuMi, the inputs are $\widetilde{M}$, $\boldsymbol{q}$, and $\boldsymbol{x}_0$.

TABLE 3
Average numbers of iterations, CPU times, and relative errors for dense M.

	n	1000			3000			5000		
	cond	10	10^3	10^5	10	10^3	10^5	10	10^3	10^5
# iter	BN	10.0/2.8	16.4/1.8	23.2/1.2	16.4/2.2	5.8/5.4	8.6/6.2	9.8/3.6	17.2/2.0	23.5/1.2
	SDPT3	18.0	16.0	15.2	16.0	16.0	18.0	16.0	18.0	20.2
	SeDuMi	13.8	12.8	15.8	13.8	13.8	17.2	15.0	13.6	16.4
	LCPvA	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0
CPU(s)	BN	2.1	2.0	2.2	33.2	33.5	33.1	127.9	128.5	131.1
	SDPT3	43.2	38.8	41.5	979.6	1014.7	1135.3	5144.9	5751.8	6039.5
	SeDuMi	11.9	10.8	15.3	312.1	287.5	360.4	1521.5	1349.0	1646.8
	LCPvA	1.8	1.8	1.7	35.5	35.4	35.3	161.0	160.4	161.5
χ_{rel_1}	BN	$6.0e{-17}$	$4.2e{-16}$	0	$6.5e{-17}$	$1.7e{-16}$	0	$5.9e{-16}$	$3.0e{-16}$	0
	SDPT3	0	0	0	0	0	0	0	0	0
	SeDuMi	0	0	0	0	0	0	0	0	0
	LCPvA	$1.8e{-14}$	$2.7e{-12}$	$1.2e{-12}$	$9.2e{-14}$	$2.4e{-10}$	$6.3e{-12}$	$2.4e{-11}$	$6.8e{-12}$	0
χ_{rel_2}	BN	$1.5e{-18}$	$1.1e{-17}$	0	$2.0e{-17}$	$4.1e{-17}$	0	$3.2e{-17}$	$1.8e{-17}$	0
	SDPT3	$4.1e{-07}$	$3.4e{-07}$	0	0	$5.7e{-07}$	$2.7e{-06}$	0	$4.4e{-07}$	$3.3e{-06}$
	SeDuMi	$2.4e{-06}$	0	0	0	0	0	$3.7e{-06}$	0	0
	LCPvA	$1.0e{-15}$	$1.2e{-13}$	$6.0e{-14}$	$3.2e{-15}$	$6.9e{-12}$	$1.6e{-13}$	$7.0e{-13}$	$1.5e{-13}$	0
χ_{rel_3}	BN	$5.0e{-15}$	$1.4e{-13}$	$1.1e{-13}$	$3.0e{-16}$	$2.2e{-15}$	$7.3e{-13}$	$7.6e{-16}$	$1.8e{-15}$	$7.4e{-12}$
	SDPT3	$2.9e{-07}$	$2.4e{-07}$	$6.1e{-06}$	$9.8e{-06}$	$4.0e{-07}$	$1.9e{-06}$	$1.8e{-05}$	$3.1e{-07}$	$2.4e{-06}$
	SeDuMi	$2.4e{-06}$	$1.9e{-06}$	$1.3e{-06}$	$2.7e{-06}$	$1.2e{-06}$	$4.5e{-06}$	$2.6e{-06}$	$1.7e{-06}$	$1.0e{-05}$
	LCPvA	$2.3e{-15}$	$2.1e{-12}$	$7.5e{-08}$	$2.2e{-14}$	$9.8e{-12}$	$1.1e{-08}$	$1.0e{-12}$	$3.3e{-13}$	$2.6e{-10}$
χ_{rel}	BN	$5.1e{-15}$	$1.4e{-13}$	$1.1e{-13}$	$3.9e{-16}$	$2.4e{-15}$	$7.3e{-13}$	$1.3e{-15}$	$2.1e{-15}$	$7.4e{-12}$
	SDPT3	$7.1e{-07}$	$5.9e{-07}$	$6.1e{-06}$	$9.9e{-06}$	$9.7e{-07}$	$4.6e{-06}$	$1.8e{-05}$	$7.5e{-07}$	$5.7e{-06}$
	SeDuMi	$4.9e{-06}$	$1.9e{-06}$	$1.3e{-06}$	$2.8e{-06}$	$1.2e{-06}$	$4.5e{-06}$	$6.4e{-06}$	$1.7e{-06}$	$1.0e{-05}$
	LCPvA	$2.2e{-14}$	$4.9e{-12}$	$7.5e{-08}$	$1.2e{-13}$	$2.6e{-10}$	$1.1e{-08}$	$2.6e{-11}$	$7.3e{-12}$	$2.6e{-10}$

In applying BN and LCPvA, we first perform upper Hessenberg reduction on M as explained in subsection 6.5.

Table 3 clearly shows the effectiveness of LCPvA for dense M in considering both speed and relative accuracy. It is also observed that BN comes out the best for $n = 3000$ and bigger, but LCPvA is still comparable. In any case, LCPvA is far more efficient than SDPT3 and SeDuMi.

8.2. Numerical tests for sparse M. For sparse problems, we can afford to treat M that are larger in size than those in the previous subsection because they are sparse.

Our investigation is carried out on two testing sets. The test matrices M in the first set are randomly generated, while the matrices in the second are from the Matrix Market.[10] The same stopping criteria as in the dense problems are used for SDPT3 and SeDuMi, while for LCPvA, we relax the stopping criterion ϵ to $\epsilon = 10^{-5}$ because with it, LCPvA already yields solutions with comparable relative errors to or much smaller relative errors than what SDPT3 and SeDuMi give. For the eigenpair $(\tau, \boldsymbol{v})$ that BN needs, we use the MATLAB function `eigs` with the default options. This alone makes BN much less competitive on the first random testing set. Due to the destruction to the sparsity of M, we do not transform M to an upper Hessenberg matrix as we have done in the dense case.

[10] http://math.nist.gov/MatrixMarket/index.html.

TABLE 4
Numerical results for sparse problems in the first testing set.

		kind=1			kind=2		
	$(\frac{1}{\texttt{rc}})^2$ ($\approx$ cond)	10^2	10^4	10^5	10^2	10^4	10^5
	$d_{\widetilde{M}}$	$3.2e-04$	$5.0e-04$	$4.1e-04$	$5.1e-04$	$6.5e-04$	$5.5e-04$
	d_M	$7.6e-04$	$9.1e-04$	$7.4e-04$	$2.9e-03$	$3.1e-03$	$2.6e-03$
# iter	BN	7.4/5.4	6.2/6.6	9.0/10.0	13.4/2.6	5.8/5.4	8.6/6.2
	SDPT3	21.2	21.0	20.0	16.0	20.0	20.6
	SeDuMi	17.4	18.6	18.0	14.4	17.8	18.4
	LCPvA	30.0	42.0	30.0	30.0	52.0	46.0
CPU(s)	BN	225.9	278.9	420.6	269.7	301.3	371.3
	SDPT3	1.5	1.1	1.8	52.6	64.8	67.2
	SeDuMi	2.9	3.3	3.7	824.7	1002.9	1067.1
	LCPvA	1.3	2.5	1.5	374.2	528.3	716.2
χ_{rel_1}	BN	$1.4e-15$	$\mathbf{5.6e-01}$	$\mathbf{1.4e-00}$	$5.8e-16$	$9.1e-16$	$\mathbf{2.8e-01}$
	SDPT3	0	0	0	0	0	0
	SeDuMi	0	0	0	0	0	0
	LCPvA	$4.2e-13$	$1.4e-14$	0	0	$1.6e-11$	$2.1e-12$
χ_{rel_2}	BN	$4.9e-16$	$\mathbf{1.1e-02}$	$\mathbf{5.9e-01}$	$1.6e-17$	$1.1e-18$	$3.9e-06$
	SDPT3	$2.7e-05$	$5.8e-04$	$6.6e-05$	$2.7e-06$	$1.0e-05$	$7.3e-06$
	SeDuMi	0	$3.4e-07$	0	$3.8e-07$	$2.1e-06$	$1.2e-06$
	LCPvA	$1.7e-13$	$8.6e-06$	0	0	$8.9e-15$	$2.4e-13$
χ_{rel_3}	BN	$6.0e-16$	$\mathbf{1.5e-03}$	$6.7e-16$	$3.1e-16$	$1.3e-18$	$1.1e-15$
	SDPT3	$2.0e-05$	$4.1e-04$	$4.7e-05$	$1.9e-06$	$7.4e-06$	$5.1e-06$
	SeDuMi	$1.1e-06$	$3.2e-06$	$2.0e-06$	$1.0e-06$	$2.8e-06$	$1.1e-06$
	LCPvA	$1.7e-13$	$9.9e-06$	$9.7e-10$	$2.4e-10$	$2.5e-10$	$4.7e-06$
χ_{rel}	BN	$2.5e-15$	$\mathbf{5.8e-01}$	$\mathbf{2.0e-00}$	$9.0e-16$	$9.2e-16$	$\mathbf{2.8e-01}$
	SDPT3	$4.7e-05$	$9.9e-04$	$1.1e-04$	$4.6e-06$	$1.7e-05$	$1.2e-05$
	SeDuMi	$1.1e-06$	$3.6e-06$	$2.0e-06$	$1.4e-06$	$4.9e-06$	$2.3e-06$
	LCPvA	$5.4e-12$	$1.8e-05$	$9.7e-10$	$2.4e-10$	$2.7e-10$	$4.7e-06$

In our first testing set, we choose to first generate the sparse $\widetilde{M}$ with a prescribed sparsity and condition number, and then form $M = \widetilde{M}^\top \widetilde{M}$. It is clear that the number of nonzero entries in M generated from this procedure is usually much larger than that of $\widetilde{M}$. Limited by our current hardware environment, we choose $n = 10,000$ and generate $(M = \widetilde{M}^\top \widetilde{M}, \boldsymbol{q})$, where $\boldsymbol{q}$ is randomly generated with elements uniformly distributed in the interval $[-1, 1]$ and $\widetilde{M}$ is generated by the MATLAB function

$$\widetilde{M} = \texttt{sprandsym(n,density,rc,kind)}.$$

It returns an n-by-n symmetric and positive definite random matrix with approximately the prescribed `density` and an approximate condition number $\frac{1}{\texttt{rc}}$, implying that the condition number `cond` of M is approximately $\left(\frac{1}{\texttt{rc}}\right)^2$. The value of the input `kind` can be either 1 or 2, where the former means that $\widetilde{M}$ is generated by random Jacobi rotations on a positive definite diagonal matrix, while the latter indicates that $\widetilde{M}$ is a shifted sum of outer products. Both cases are tested with `density` $= 0.0005$ and different `rc` to make `cond` vary from 10^2 to 10^5.

For each prescribed pair (`rc`,`kind`), we generate five triples $(M = \widetilde{M}^\top \widetilde{M}, \boldsymbol{q}, \boldsymbol{x}_0)$, where $\boldsymbol{x}_0$ is the initial point for the three algorithms other than LCPvA, and record the numerical results by the four methods in Table 4. Also reported are the average sparse densities d_M and $d_{\widetilde{M}}$ of M and $\widetilde{M}$ for each pair (`rc`,`kind`), respectively.

We have several observations from Table 4:

(1) The numerical results show that each algorithm performs quite differently for the cases `kind`=1 and `kind`=2.

TABLE 5
Numerical results for sparse problems in the second testing set.

	M	S1RMQ4M1	S1RMT3M1	S2RMQ4M1	S2RMT3M1	S3RMQ4M1	S3RMT3M1	S3RMT3M3	BCSSTK17	BCSSTK18
	n	5489	5489	5489	5489	5489	5489	5357	10974	11948
	cond	3.2e+06	5.4e+06	1.2e+08	5.8e+08	1.3e+10	1.3e+10	2.6e+10	6.5e+01	6.5e+01
	2-norm	6.9e+05	9.7e+05	6.9e+04	9.7e+04	6.9e+03	9.7e+03	9.6e+03	1.3e+10	4.3e+10
	d_M	8.7e−03	7.2e−03	8.7e−03	7.2e−03	8.7e−03	7.2e−03	7.2e−03	3.6e−03	1.0e−03
	$d_{\widetilde{M}}$	3.3e−02	3.2e−02	3.3e−02	3.3e−02	3.3e−02	3.3e−02	7.7e−02	1.3e−02	2.0e−02
# iter	BN	25/1	25/1	21/1	22/2	18/2	18/2	18/2	1/10	—
	SDPT3	15	16	14	13	12	18	14	18	32
	SeDuMi	20	21	17	18	16	17	17	14	20
	LCPvA	30	30	30	30	30	30	30	20	30
CPU(s)	BN	16.6	13.3	14.8	20.6	21.6	20.4	18.4	588.3	—
	SDPT3	11.4	12.1	10.7	10.1	9.2	13.7	247.4	25.9	199.3
	SeDuMi	36.7	34.9	22.7	28.6	16.1	18.8	102.4	25.5	151.1
	LCPvA	9.7	5.8	9.6	6.0	9.1	6.2	5.2	6.3	6.7
χ_{rel_1}	BN	0	0	0	0	0	0	0	**1.4e−00**	—
	SDPT3	0	0	0	0	0	0	0	0	0
	SeDuMi	0	0	0	0	0	0	0	0	0
	LCPvA	0	0	2.4e−09	0	5.9e−10	0	1.2e−09	8.9e−14	**1.0e−00**
χ_{rel_2}	BN	0	0	0	0	0	0	0	6.6e−11	—
	SDPT3	1.7e−05	8.8e−06	7.7e−07	1.7e−05	6.5e−06	0	0	0	0
	SeDuMi	0	0	2.1e−08	1.4e−07	2.0e−06	9.1e−07	9.3e−07	0	0
	LCPvA	0	0	1.3e−10	0	3.5e−11	0	5.9e−11	4.2e−24	7.6e−23
χ_{rel_3}	BN	1.1e−13	2.4e−13	1.8e−12	7.5e−17	1.4e−16	2.6e−17	1.9e−17	2.3e−25	—
	SDPT3	1.2e−05	6.2e−06	5.5e−07	1.2e−05	4.6e−06	1.2e−08	4.0e−06	4.6e−07	8.1e−08
	SeDuMi	3.9e−07	4.3e−07	1.5e−08	9.8e−08	1.4e−06	6.5e−07	6.6e−07	1.4e−07	2.6e−08
	LCPvA	3.0e−09	2.7e−05	1.9e−10	1.1e−10	4.9e−11	1.8e−09	8.3e−11	6.6e−24	1.0e−27
χ_{rel}	BN	1.1e−13	2.4e−13	1.8e−12	7.5e−17	1.4e−16	2.6e−17	1.9e−17	**1.4e−00**	—
	SDPT3	2.8e−05	1.5e−05	1.3e−06	2.9e−05	1.1e−05	1.2e−08	4.0e−06	4.6e−07	8.1e−08
	SeDuMi	3.9e−07	4.3e−07	3.6e−08	2.4e−07	3.5e−06	1.6e−06	1.6e−06	1.4e−07	2.6e−08
	LCPvA	3.0e−09	2.7e−05	2.8e−09	1.1e−10	6.8e−10	1.8e−09	1.3e−09	8.9e−14	**1.0e−00**

(2) For `kind`=1, it turns out that LCPvA has the best numerical performance in terms of speed and accuracy. Note that there are almost twice as many nonzeros in M as in $\widetilde{M}$. It is also observed that BN performs the worst due to the requirement in finding an accurate eigenpair $(\tau, \boldsymbol{v})$.

(3) For `kind`=2, SDPT3 is the best in time, but its solutions are often less accurate than the ones by LCPvA.

Our second set of testing matrices M are from the Matrix Market found through searching all real sparse symmetric, positive definite, and nondiagonal matrices with $5000 \le n \le 90000$. We found nine matrices, namely, S1RMQ4M1, S1RMT3M1, S2RMQ4M1, S2RMT3M1, S3RMQ4M1, S3RMT3M1, S3RMT3M3, BCSSTK17, and BCSSTK18, whose basic properties are listed at the top of Table 5. The corresponding $\widetilde{M}$ for SDPT3 and SeDuMi is computed by $\widetilde{M} = \texttt{chol}(M)$ (the CPU time for this is not counted in), and we also report d_M and $d_{\widetilde{M}}$, the sparse densities of M and $\widetilde{M}$, respectively. To form a specific SOCLCP (1.1), we take $\boldsymbol{q} = [1, 1, \ldots, 1]^\top \in \mathbb{R}^n$ and generate a random $\boldsymbol{x}_0$ as the starting vector for BN, SDPT3 and SeDuMi.

Table 5 summarizes the numerical results for the second test set, where "—" for BN means that it fails in `eigs` for solving the eigenpair $(\tau, \boldsymbol{v})$. We observe that LCPvA has a very good performance in all the cases except for BCSSTK18. The failure is due to that the choice $s_0 \approx 1.0252 \times 10^{10}$ by Table 2 is too far from the true $s_* \approx 3.8768 \times 10^3$; as a result, $h_\ell(s)$ provides a very poor approximation of $h(s)$ near s_* and the next shift satisfying (6.7) cannot be found. We experimented with using

different initial shifts and were able to make LCPvA run successfully with $s_0 = \|\boldsymbol{q}\|_2/5$ to give $\chi_{\text{rel}_1} = \chi_{\text{rel}_2} = 0$ and $\chi_{\text{rel}_3} = 1.9 \times 10^{-11}$ in 6.15 seconds. Although the s_0 given in Table 2 had been working well until BCSSTK18, this suggests that there are cases for which more effective s_0 is needed for LCPvA to succeed. We shall investigate this issue in our future study.

Finally, it is interesting to note that for BCSSTK17, LCPvA uses only 20 Arnoldi steps, which implies that a friendly breakdown occurred and, thereby, results in $h_{20}(s) \equiv h(s)$; consequently, s_* is an exact zero of $h_{20}(s)$. This is partially revealed by the tiny relative error $\chi_{\text{rel}} = 8.9 \times 10^{-14}$.

9. Concluding remarks and future work. In this paper, we have proposed a Krylov subspace method for the linear complementarity problem over a single second-order cone. The method is in the spirit of projection and mimics the well-known Rayleigh–Ritz procedure [16, 23] for the large-scale eigenvalue problems. It is known that the Rayleigh–Ritz procedure first seeks a good approximate subspace to an invariant subspace and then projects the eigenvalue problem to a much smaller one, which is then solved to give approximate eigenpairs. The Krylov subspace technique is the key in the first phase. When it comes to SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$, it is not that evident what would constitute a good approximate subspace that could be used as a projection subspace to yield a much smaller size SOCLCP (1.1). In this paper, through the special rational function $h(s)$ and techniques in the model reduction, we have successfully defined good approximate subspaces upon which the original large-scale SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ is projected to yield problems of much smaller scale. Their accuracy in approximating the original problem can be controlled by the number ℓ of Arnoldi steps and measured by $|h(s) - h_\ell(s)|$, which is of order $O(s^\ell)$. Based on this idea, we presented an algorithmic framework, LCPvA (linear complementarity problem via Arnold), in Algorithm 2, which was tested against existing sophisticated solvers. Our preliminary numerical results demonstrate its efficiency.

Detailed theoretical analysis of the curve of $h(s)$ is also presented. It is of interest both in theory and in computation. The success of our proposed method here in large part is due to our complete knowledge of how $h(s)$ behaves. That knowledge makes it possible for us to select the right one among all zeros of the reduced $h_\ell(s)$ to approximate the desired one of $h(s)$.

We have identified two relevant problems that warrant further investigation. One is to iteratively solve the linear systems arising in the Arnoldi process in Algorithm 1 for extremely large-scale SOCLCP; how accurate the solutions to the related linear systems should be so that the resulted approximate solution of SOCLCP is within the given tolerance is a practical issue. Our current implementation uses MATLAB's sparse LU for simplicity. The other is to find better initial s_0. So far the one given in Table 2 works well for us, except for one example in BCSSTK18, for which we found that a better initial s_0 is needed for convergence.

We outlined in section 7 several potential uses of any efficient solver for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ in solving practically important cone-related programming problems. LCPvA can certainly be used as building blocks to solve these problems.

Our major assumption for SOCLCP$(M, \mathbb{K}^n, \boldsymbol{q})$ is that M has the GUS property upon which all our developments are built. What happens when M does not have the GUS property? This will be yet another problem that warrants our attention in the future.

Acknowledgment. We would like to thank the two anonymous referees for their helpful suggestions and comments, which have improved the presentation of this paper significantly and also exposed other potential future research topics.

REFERENCES

[1] S. Adly and H. Rammal, *A new method for solving second-order cone eigenvalue complementarity problems*, J. Optim. Theory Appl., (2014), DOI: 10.1007/s10957-014-0645-0.

[2] F. Alizadeh and D. Goldfarb, *Second-order cone program.*, Math. Program., 95 (2003), pp. 3–51.

[3] M. F. Anjos and J. B. Lasserre, eds., *Handbook on Semidefinite, Conic and Polynomial Optimization*, Internat. Ser. Oper. Res. Management Sci. 166, Springer, Berlin, 2012.

[4] A. Auslender, *Variational inequalities over the cone of semidefinite positive symmetric matrices and over the Lorentz cone*, Optim. Methods Softw., 18 (2003), pp. 359–376.

[5] Z. Bai and Y. Su, *Dimension reduction of large-scale second-order dynamical systems via a second-order Arnoldi method*, SIAM J. Sci. Comput., 25 (2005), pp. 1692–1709.

[6] Z. Bai and Y. Su, *SOAR: A second-order Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 640–659.

[7] Z. Bai and Q. Ye, *Error estimation of the Padé approximation of transfer functions via the Lanczos process*, Electron. Trans. Numer. Anal., 7 (1998), pp. 1–17.

[8] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, 1982.

[9] J. Burke and S. Xu, *The global linear convergence of a noninterior path-following algorithm for linear complementarity problems*, Math. Oper. Res., 23 (1998), pp. 719–734.

[10] J.-S. Chen and S. H. Pan, *A descent method for a reformulation of the second-order cone complementarity problem*, J. Comput. Appl. Math., 213 (2008), pp. 547–558.

[11] J.-S. Chen and P. Tseng, *An unconstrained smooth minimization reformulation of the second-order cone complementarity problem*, Math. Program., 104 (2005), pp. 293–327.

[12] X. Chen, L. Qi, and D. F. Sun, *Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities*, Math. Comp., 67 (1998), pp. 519–540.

[13] X. D. Chen, D. F. Sun, and J. Sun, *Complementarity functions and numerical experiments on some smoothing Newton methods for second-order-cone complementarity problems*, Comput. Optim. Appl., 25 (2003), pp. 39–56.

[14] E. Chiprout and M. Nakhla, *Asymptotic Waveform Evaluation and Moment Matching for Interconnect Analysis*, Kluwer Academic Publishers, Norwell, MA, 1994.

[15] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementarity Problem*, in Computer Science and Scientific Computing, Academic Press, New York, 1992.

[16] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[17] P. Feldman and R. W. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. Computer-Aided Design, 14 (1995), pp. 639–649.

[18] M. Fukushima, Z.-Q. Luo, and P. Tseng, *Smoothing functions for second-order-cone complementarity problems*, SIAM J. Optim., 12 (2002), pp. 436–460.

[19] P. Gajardo and A. Seeger, *Solving inverse cone-constrained eigenvalue problems*, Numer. Math., 123 (2013), pp. 309–331.

[20] K. Gallivan, E. Grimme, and P. Van Dooren, *Asymptotic waveform evaluation via a Lanczos method*, Appl. Math. Lett., 7 (1994), pp. 75–80.

[21] N. Goldberg and S. Leyffer, *Active Set Method for Second-Order Conic-Constrained Quadratic Programming*, Tech. report ANL/MCS-P5085-0214, Mathematics and Computer Science Division, Argonne National Laboratory, 2014; available online from https://wiki.mcs.anl.gov/leyffer/images/e/e2/ConicProjGrad.pdf.

[22] D. Goldfarb and K. Scheinberg, *Product-form Cholesky factorization in interior point methods for second-order cone program.*, Math. Program., 103 (2005), pp. 153–179.

[23] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[24] M. S. Gowda and R. Sznajder, *Automorphism invariance of P- and GUS-properties of linear transformations on Euclidean Jordan algebras*, Math. Oper. Res., 31 (2006), 109123.

[25] M. S. Gowda and R. Sznajder, *Some global uniqueness and solvability results for linear complementarity problems over symmetric cones*, SIAM J. Optim., 18 (2007), pp. 461–481.

[26] M. S. Gowda, R. Sznajder, and J. Tao, *Some P-properties for linear transformations on Euclidean Jordan algebras*, Linear Algebra Appl., 393 (2004), pp. 203–232.

[27] E. J. Grimme, *Krylov Projection Methods for Model Reduction*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1997.

[28] X.-L. Guo, Z.-B. Deng, S.-C. Fang, Z.-B. Wang, and W.-X. Xing, *Quadratic optimization over a second-order cone with linear equality constraints*, J. Oper. Res. Soc. China, 2 (2014), pp. 17–38.
[29] S. Hayashi, N. Yamashita, and M. Fukushima, *A combined smoothing and regularization method for monotone second-order cone complementarity problems*, SIAM J. Optim., 15 (2005), pp. 593–615.
[30] M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.
[31] T. Heyn, M. Anitescu, A. Tasora, and D. Negrut, *Using Krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation*, Internat. J. Numer. Methods Eng., 95 (2013), pp. 541–561.
[32] J.-B. Hiriart-Urruty and A. Seeger, *A variational approach to copositive matrices*, SIAM Rev., 52 (2010), pp. 593–629.
[33] Z. H. Huang and T. Ni, *Smoothing algorithms for complementarity problems over symmetric cones*, Comput. Optim. Appl., 45 (2010), pp. 557–579.
[34] C. Kanzow, *Some noninterior continuation methods for linear complementarity problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 851–868.
[35] M. Kojima, S. Shindoh, and S. Hara, *Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices*, SIAM J. Optim., 7 (1997), pp. 86–125.
[36] L. C. Kong, J. Sun, and N. H. Xiu, *A regularized smoothing Newton method for symmetric cone complementarity problems*, SIAM J. Optim., 19 (2008), pp. 1028–1047.
[37] R.-C. Li and Z. Bai, *Structure-preserving model reduction using a Krylov subspace projection formulation*, Commun. Math. Sci., 3 (2005), pp. 179–199.
[38] R.-C. Li and Q. Ye, *Simultaneous similarity reductions for a pair of matrices to condensed forms*, Commun. Math. Stat., 2 (2014), pp. 139–153.
[39] X. Liang, R.-C. Li, and Z. Bai, *Trace minimization principles for positive semi-definite pencils*, Linear Algebra Appl., 438 (2013), pp. 3085–3106.
[40] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, *Application of second-order cone programming*, Linear Algebra Appl., 284 (1998), pp. 193–228.
[41] R. Loewy, *Positive operators on the n-dimensional ice cream cone*, J. Math. Anal. Appl., 49 (1975), pp. 375–392.
[42] J. L. Morales, J. Nocedal, and M. Smelyanskiy, *An algorithm for the fast solution of symmetric linear complementarity problems*, Numer. Math., 111 (2008), pp. 251–266.
[43] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., Springer, New York, 2006.
[44] S. Pan and J.-S. Chen, *A damped Gauss-Newton method for the second-order cone complementarity problem*, Appl. Math. Optim., 59 (2009), pp. 293–318.
[45] J.-S. Pang, D. F. Sun, and J. Sun, *Semismooth homeomorphisms and strong stability of semidefinite and Lorentz complementarity problems*, Math. Oper. Res., 28 (2003), pp. 39–63.
[46] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.
[47] B. N. Parlett, D. R. Taylor, and Z. A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.
[48] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969. pp. 283–298.
[49] M. Rojas, S. A. Santos, and D. C. Sorensen, *A new matrix-free algorithm for the large-scale trust-region subproblem*, SIAM J. Optim., 11 (2000), pp. 611–646.
[50] M. Rojas, S. A. Santos, and D. C. Sorensen, *Algorithm* 873*: LSTRS: MATLAB software for large-scale trust-region subproblems and regularization*, ACM Trans. Math. Software, 34 (2008).
[51] W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, eds., *Model Order Reduction: Theory, Research Aspects and Applications*, Math. Ind. 13, Springer, New York, 2008.
[52] A. Seeger and M. Torki, *On eigenvalues induced by a cone constraint*, Linear Algebra Appl., 372 (2003), pp. 181–206.
[53] J. F. Sturm, *Using SeDuMi* 1.02*, a MATLAB toolbox for optimization over symmetric cones*, Optim. Methods Softw., 11 (1999), pp. 625–653.
[54] T.-J. Su and J. R. R. Craig, *Model reduction and control of flexible structures using Krylov vectors*, J. Guidance Control Dynamics, 14 (1991), pp. 260–267.
[55] K. C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT*3*—A MATLAB software package for semidefinite programming*, Optim. Methods Softw., 11 (1999), pp. 545–581.

[56] K. C. Toh, M. J. Todd, and R. H. Tütüncü, *On the Implementation and Usage of SDPT3—A MATLAB Software Package for Semidefinite-Quadratic-Linear Programming, Version* 4.0, Tech. report, Department of Mathematics, National University of Singapore, 2010.

[57] R. H. Tütüncü, K. C. Toh, and M. J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Math. Program., 95 (2003), pp. 189–217.

[58] W. H. Yang and X. M. Yuan, *The GUS-property of second-order cone linear complementarity problems*, Math. Program., 141 (2013), pp. 295–317.

[59] L.-H. Zhang and W. H. Yang, *An efficient algorithm for second-order cone linear complementarity problems*, Math. Comp., 83 (2013), pp. 1701–1726.

[60] L.-H. Zhang and W. H. Yang, *An efficient matrix splitting method for the second-order cone complementarity problem*, SIAM J. Optim., 24 (2014), pp. 1178–1205.

[61] L.-H. Zhang, W. H. Yang, C. Shen, and R.-C. Li, *A Krylov Subspace Method for Large Scale Second Order Cone Linear Complementarity Problem*, Tech. report 2014-19, Department of Mathematics, University of Texas at Arlington, 2014; available online from http://www.uta.edu/math/preprint.