Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 15, No. 1, pp. 1 – 22, 2019

APPLYING A TYPE OF SOC-FUNCTIONS TO SOLVE A SYSTEM OF EQUALITIES AND INEQUALITIES UNDER THE ORDER INDUCED BY SECOND-ORDER CONE

Xin-He Miao*, Nuo Qi, B. Saheya† and Jein-Shan Chen‡

Abstract: In this paper, we introduce a special type of SOC-functions which is a vector-valued function associated with second-order cone. By using it, we construct a type of smoothing functions which converges to the projection function onto second-order cone. Then, we reformulate the system of equalities and inequalities under the order induced by second-order cone as a system of parameterized smooth equations. Accordingly, we propose a smoothing-type Newton algorithm to solve the reformulation, and show that the proposed algorithm is globally convergent and locally quadratically convergent under suitable assumptions. Preliminary numerical results demonstrate that the approach is effective. Numerical comparison based on various smoothing functions is reported as well.

Key words: *system of equalities and inequalities, second-order cone, SOC-function, smoothing algorithm, global convergence*

Mathematics Subject Classification: *65K10, 93B40, 26D07*

1 Introduction

The second-order cone (SOC for short and denoted by $\mathcal{K}^n$) in $\mathbb{R}^n$ $(n \geq 1)$, also called the Lorentz cone, is defined as

$$\mathcal{K}^n = \left\{(x_1, x_2) \in \mathbb{R} \times \mathbb{R}^{n-1} \,|\, \|x_2\| \leq x_1\right\},$$

where $\|\cdot\|$ denotes the Euclidean norm. By the definition of $\mathcal{K}^n$, if $n = 1$, $\mathcal{K}^1$ is the set of nonnegative reals $\mathbb{R}_+$. Moreover, we know that a general second-order cone $\mathcal{K}$ is the Cartesian product of SOCs, i.e.,

$$\mathcal{K} := \mathcal{K}^{n_1} \times \mathcal{K}^{n_2} \times \cdots \times \mathcal{K}^{n_r}.$$

Since all the analysis can be carried over to the setting of Cartesian product, we only focus on the single second-order cone $\mathcal{K}^n$ for simplicity. It is well known that the second-order cone $\mathcal{K}^n$ is a symmetric cone. During the past decade, optimization problems involved SOC constraints and their corresponding solutions methods have been studied extensively,

*The author's work is supported by National Natural Science Foundation of China (No. 11471241).
†The author's work is supported by National Natural Science Foundation of China (No. 11402127,11401326).
‡Corresponding author. The author's work is supported by Ministry of Science and Technology, Taiwan

see [1, 5, 8, 13, 19, 20, 24, 29–31, 34, 35] and references therein.

There is a spectral decomposition with respect to second-order cone $\mathcal{K}^n$ in $\mathbb{R}^n$, which plays a very important role in the study of second-order cone optimization problems. For any vector $x = (x_1, x_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$, the spectral decomposition (or spectral factorization) with respect to $\mathcal{K}^n$ is given by

$$x = \lambda_1(x)u_x^{(1)} + \lambda_2(x)u_x^{(2)}, \tag{1.1}$$

where $\lambda_1(x)$, $\lambda_2(x)$ and $u_x^{(1)}$, $u_x^{(2)}$ are called the spectral values and the spectral vectors of x, respectively, with their corresponding formulas as bellow:

$$\lambda_i(x) = x_1 + (-1)^i \|x_2\|, \quad i = 1, 2, \tag{1.2}$$

$$u_x^{(i)} = \begin{cases} \frac{1}{2}\begin{bmatrix} 1 \\ (-1)^i \frac{x_2}{\|x_2\|} \end{bmatrix}, & \text{if } x_2 \neq 0, \\ \frac{1}{2}\begin{bmatrix} 1 \\ (-1)^i w \end{bmatrix}, & \text{if } x_2 = 0, \end{cases} \tag{1.3}$$

for $i = 1, 2$ with w being any vector in $\mathbb{R}^{n-1}$ satisfying $\|w\| = 1$. Moreover, $\left\{u_x^{(1)}, u_x^{(2)}\right\}$ is called a Jordan frame satisfying the following properties:

$$u_x^{(1)} + u_x^{(2)} = e,\ \left\langle u_x^{(1)}, u_x^{(2)} \right\rangle = 0,\ u_x^{(1)} \circ u_x^{(2)} = 0 \ \text{ and } \ u_x^{(i)} \circ u_x^{(i)} = u_x^{(i)}\ (i = 1, 2),$$

where $e = (1, 0, \cdots, 0)^T \in \mathbb{R}^n$ is the unit element and Jordan product $x \circ y$ is defined by $x \circ y := (\langle x, y \rangle, x_1 y_2 + y_1 x_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$ for any $x = (x_1, x_2), y = (y_1, y_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$. For more details about Jordan product, please refer to [11].

In [5, 6], for any real-valued function $f : \mathbb{R} \to \mathbb{R}$ and $x = (x_1, x_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$, based on the spectral factorization of x with respect to $\mathcal{K}^n$, a type of vector-valued function associated with $\mathcal{K}^n$ (also called SOC-function) is introduced. More specifically, if we apply f to the spectral values of x in (1.1), then we obtain the function $f^{\rm soc} : \mathbb{R}^n \to \mathbb{R}^n$ given by

$$f^{\rm soc}(x) = f(\lambda_1(x))u_x^{(1)} + f(\lambda_2(x))u_x^{(2)}. \tag{1.4}$$

From the expression (1.4), it is clear that the SOC-function $f^{\rm soc}$ is unambiguous whether $x_2 = 0$ or $x_2 \neq 0$. Further properties regarding $f^{\rm soc}$ were discussed in [3–5, 7, 17, 32]. It is also known that such SOC-functions $f^{\rm soc}$ associated with second-order cone play a crucial role in the theory and numerical algorithm for second-order cone programming, see [1, 5, 8, 13, 19, 20, 24, 29–31, 34, 35] again.

In this paper, in light of the definition of $f^{\rm soc}$, we define another type of SOC-function Φ_μ (see Section 2 for details). In particular, using the SOC-function Φ_μ, we will solve the following system of equalities and inequalities under the order induced by the second-order cone:

$$\begin{cases} f_I(x) \preceq_{\mathcal{K}^m} 0, \\ f_E(x) = 0, \end{cases} \tag{1.5}$$

where $f_I(x) = (f_1(x), \cdots, f_m(x))^T$, $f_E(x) = (f_{m+1}(x), \cdots, f_n(x))^T$, and "$x \preceq_{\mathcal{K}^m} 0$" means "$-x \in \mathcal{K}^m$". Likewise, $x \succeq_{\mathcal{K}^m} 0$ means $x \in \mathcal{K}^m$ and $x \succ_{\mathcal{K}^m} 0$ means $x \in {\rm int}(\mathcal{K}^m)$ whereas

$\mathrm{int}(\mathcal{K}^m)$ denotes the interior of $\mathcal{K}^m$. Throughout this paper, we assume that f_i is continuously differentiable for any $i \in \{1, 2, ..., n\}$. We also define

$$f(x) := \left[\begin{array}{c} f_I(x) \\ f_E(x) \end{array} \right]$$

and hence f is continuously differentiable. When $\mathcal{K}^m = \mathbb{R}^m_+$, the system (1.5) reduces to the standard system of equalities and inequalities over $\mathbb{R}^m$. The corresponding standard system (1.5) has been studied extensively due to its various applications, and there are many methods for solving such problems, see [10, 27, 28, 33, 37]. For the setting of second-order cone, we know that the KKT conditions of the second-order cone constrained optimization problems can be expressed in form of (1.5), i.e., the system of equalities and inequalities under the order induced by second-order cones. For example, for the following second-order cone optimization problem:

$$\begin{array}{ll} \min & h(x) \\ \text{s.t.} & -g(x) \in \mathcal{K}^m, \end{array}$$

the KKT conditions of this problem is as follows

$$\begin{array}{rcl} \nabla h(x) + \nabla g(x)\lambda & = & 0, \\ \lambda^T g(x) & = & 0, \\ -\lambda & \preceq_{\mathcal{K}^m} & 0, \\ g(x) & \preceq_{\mathcal{K}^m} & 0, \end{array}$$

where $\nabla g(x)$ denotes the gradient matrix of g. Now, by denoting

$$f_I(x, \lambda) := \left[\begin{array}{c} -\lambda \\ g(x) \end{array} \right] \quad \text{and} \quad f_E(x, \lambda) := \left[\begin{array}{c} \nabla h(x) + \nabla g(x)\lambda \\ \lambda^T g(x) \end{array} \right],$$

it is clear to see that the KKT conditions of the second-order cone optimization problem is in form of (1.5). From this view, the investigation of the system (1.5) provides a theoretical way for solving second-order cone optimization problems. Hence, the study of the system (1.5) is important and that is the main motivation for this paper.

So far, there are many kinds of numerical methods for solving the second-order cone optimization problems. Among which, there is a class of popular numerical method, the so-called smoothing-type algorithms. This kind of algorithm has also been a powerful tool for solving many other optimization problems, including symmetric cone complementarity problems [15, 16, 20–22], symmetric cone linear programming [23, 26], the system of inequalities under the order induced by symmetric cone [18, 25, 38], and so on. From these recent studies, most of the existing smoothing-type algorithms were designed on the basis of a monotone line search. In order to achieve better computational results, the nonmonotone line search technique is sometimes adopted in the numerical implementations of smoothing-type algorithms [15,36,37]. The main reason is that the nonmonotone line search scheme can improve the likelihood of finding a global optimal solution and convergence speed in cases where the function involved is highly nonconvex or has a valley in a small neighborhood of some point. In view of this, in this paper we also develop a nonmonotone smoothing-type algorithm for solving the system of equalities and inequalities under the order induced by second-order cones.

The remaining parts of this paper are organized as follows. In Section 2, some background concepts and preliminary results about the second-order cone are given. In Section 3, we reformulate (1.5) as a system of smoothing equations in which Φ_μ is employed. In Section 4, we propose a nonmonotone smoothing-type algorithm for solving (1.5), and show that the algorithm is well defined. Moreover, we also discuss the global convergence and locally quadratic convergence of the proposed algorithm. The preliminary numerical results are reported to demonstrate that the proposed algorithm is effective in Section 5. Some numerical comparison in light of performance profiles is presented which indicates the difference of numerical performance when various smoothing functions are used.

$\boxed{2}$ Preliminaries

In this section, we briefly review some basic properties about the second-order cone and the vector-valued functions with respect to SOC, which will be extensively used in subsequent analysis. More details about the second-order cone and the vector-valued functions can be found in [3–5,13,14,17].

First, we review the projection of $x \in \mathbb{R}^n$ onto the second-order cone $\mathcal{K}^n \subset \mathbb{R}^n$. For the second-order cone $\mathcal{K}^n$, let $(\mathcal{K}^n)^*$ denote its dual cone. Then, $(\mathcal{K}^n)^*$ is given by

$$(\mathcal{K}^n)^* := \left\{ y = (y_1, y_2) \in \mathbb{R} \times \mathbb{R}^{n-1} \,|\, \langle x, y \rangle \geq 0, \forall x \in \mathcal{K}^n \right\}.$$

Moreover, it is well known that the second-order cone $\mathcal{K}^n$ is a self-dual cone, i.e., $(\mathcal{K}^n)^* = \mathcal{K}^n$. Let x_+ denote the projection of $x \in \mathbb{R}^n$ onto the second-order cone $\mathcal{K}^n$, and x_- denote the projection of $-x$ onto the dual cone $(\mathcal{K}^n)^*$. With these notations, for any $x \in \mathbb{R}^n$, it is not hard to verify that $x = x_+ - x_-$. In particular, due to the special structure of $\mathcal{K}^n$, the explicit formula of the projection of $x \in \mathbb{R}^n$ onto $\mathcal{K}^n$ is obtained in [14] as below:

$$x_+ = \begin{cases} x & \text{if } x \in \mathcal{K}^n, \\ 0 & \text{if } x \in -(\mathcal{K}^n)^* = -\mathcal{K}^n, \\ u & \text{otherwise,} \end{cases} \tag{2.1}$$

where

$$u = \left[\begin{array}{c} \dfrac{x_1 + \|x_2\|}{2} \\ \left(\dfrac{x_1 + \|x_2\|}{2} \right) \dfrac{x_2}{\|x_2\|} \end{array} \right].$$

In fact, according to the spectral decomposition of x, the expression of the projection x_+ onto $\mathcal{K}^n$ can be alternatively expressed as (see [13, Prop. 3.3(b)])

$$x_+ = \left((\lambda_1(x))_+ u_x^{(1)} + ((\lambda_2(x))_+ u_x^{(2)},\right.$$

where $(\alpha)_+ = \max\{0, \alpha\}$ for any $\alpha \in \mathbb{R}$.

From the definition (1.4) of the vector-valued function associated with $\mathcal{K}^n$, we know that the projection x_+ onto $\mathcal{K}^n$ is a vector-valued function. Moreover, it is known that the projection x_+ and $(\alpha)_+$ for any $\alpha \in \mathbb{R}$ have many the same properties, such as the continuity, the directional differentiability and semismooth and so on. Indeed, these properties are established for general vector-valued functions associated with SOC. Among which, Chen, Chen and Tseng [5] have obtained that many properties of f^{soc} are inherited from the function f, which is presented in the following proposition.

Proposition 2.1. *Suppose that $x = (x_1, x_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$ has the spectral decomposition given as in* (1.1)-(1.3)*. For any the function $f : \mathbb{R} \to \mathbb{R}$ and the vector-valued function f^{soc} defined by* (1.4)*, the following hold.*

(a) *f^{soc} is continuous at $x \in \mathbb{R}^n$ with spectral values $\lambda_1(x)$, $\lambda_2(x) \Longleftrightarrow f$ is continuous at $\lambda_1(x)$, $\lambda_2(x)$;*

(b) *f^{soc} is directionally differentiable at $x \in \mathbb{R}^n$ with spectral values $\lambda_1(x)$, $\lambda_2(x) \Longleftrightarrow f$ is directionally differentiable at $\lambda_1(x)$, $\lambda_2(x)$;*

(c) *f^{soc} is differentiable at $x \in \mathbb{R}^n$ with spectral values $\lambda_1(x)$, $\lambda_2(x) \Longleftrightarrow f$ is differentiable at $\lambda_1(x)$, $\lambda_2(x)$;*

(d) *f^{soc} is strictly continuous at $x \in \mathbb{R}^n$ with spectral values $\lambda_1(x)$, $\lambda_2(x) \Longleftrightarrow f$ is strictly continuous at $\lambda_1(x)$, $\lambda_2(x)$;*

(e) *f^{soc} is semismooth at $x \in \mathbb{R}^n$ with spectral values $\lambda_1(x)$, $\lambda_2(x) \Longleftrightarrow f$ is semismooth at $\lambda_1(x)$, $\lambda_2(x)$;*

(f) *f^{soc} is continuously differentiable at $x \in \mathbb{R}^n$ with spectral values $\lambda_1(x)$, $\lambda_2(x) \Longleftrightarrow f$ is continuously differentiable at $\lambda_1(x)$, $\lambda_2(x)$.*

Note that the projection function x_+ onto $\mathcal{K}^n$ is not a smoothing function on the whole space $\mathbb{R}^n$. From Proposition 2.1, we can make some smoothing functions for the projection x_+ onto $\mathcal{K}^n$ if we smooth the functions $f(\lambda_i(x))$ for $i = 1, 2$. More specifically, we consider a family of smoothing functions $\phi(\mu, \cdot) : \mathbb{R} \to \mathbb{R}$ with respect to $(\alpha)_+$ satisfying

$$\lim_{\mu \downarrow 0} \phi(\mu, \alpha) = (\alpha)_+ \quad \text{and} \quad 0 \leq \frac{\partial \phi}{\partial \alpha}(\mu, \alpha) \leq 1. \tag{2.2}$$

for all $\alpha \in \mathbb{R}$. Are there functions satisfying the above conditions? Yes, there are many. We illustrate three of them here:

$$\begin{aligned}
\phi_1(\mu, \alpha) &= \frac{\sqrt{\alpha^2 + 4\mu^2} + \alpha}{2}, \qquad (\mu > 0) \\
\phi_2(\mu, \alpha) &= \mu \ln(e^{\frac{\alpha}{\mu}} + 1), \qquad (\mu > 0) \\
\phi_3(\mu, \alpha) &= \begin{cases} \alpha, & \text{if } \alpha \geq \mu, \\ \frac{(\alpha+\mu)^2}{4\mu}, & \text{if } -\mu < \alpha < \mu, \\ 0, & \text{if } \alpha \leq -\mu. \end{cases} \qquad (\mu > 0)
\end{aligned}$$

In fact, the functions ϕ_1 and ϕ_2 were considered in [13, 17], while the function ϕ_3 was employed in [18, 37]. In addition, as for the function ϕ_3, there is a more general function $\phi_p(\mu, \cdot) : \mathbb{R} \to \mathbb{R}$ given by

$$\phi_p(\mu, \alpha) = \begin{cases} \alpha & \text{if} \quad \alpha \geq \frac{\mu}{p-1}, \\ \frac{\mu}{p-1} \left[\frac{(p-1)(\alpha+\mu)}{p\mu} \right]^p & \text{if} \quad -\mu < \alpha < \frac{\mu}{p-1}, \\ 0 & \text{if} \quad \alpha \leq -\mu, \end{cases}$$

where $\mu > 0$ and $p \geq 2$. This function ϕ_p is recently studied in [9] and it is not hard to verify that ϕ_p also satisfies the above conditions (2.2). All the functions ϕ_1, ϕ_2 and ϕ_3 will play the role of smoothing functions as $f(\lambda_i(x))$ in (1.4). In other words, based on these smoothing functions, we define a type of SOC-functions $\Phi_\mu(\cdot)$ on $\mathbb{R}^n$ associated with $\mathcal{K}^n$ ($n \geq 1$) as

$$\Phi_\mu(x) := \phi(\mu, \lambda_1(x))u_x^{(1)} + \phi(\mu, \lambda_2(x))u_x^{(2)} \quad \forall x = (x_1, x_2) \in \mathbb{R} \times \mathbb{R}^{n-1}, \tag{2.3}$$

where $\lambda_1(x)$, $\lambda_2(x)$ are given by (1.2) and $u_x^{(1)}$, $u_x^{(2)}$ are given by (1.3). In light of the properties of $\phi(\mu, \alpha)$, we show as below that the SOC-function $\Phi_\mu(x)$ becomes the smoothing function for the projection function x_+ onto $\mathcal{K}^n$.

We depict the graphs of $\phi_i(\mu, \alpha)$ for i = 1, 2, 3, in Figure 1. From Figure 1, we see that ϕ_3 is the one which best approximates the function $(\alpha)_+$ under the sense that it is closest to $(\alpha)_+$ among all $\phi_i(\mu, \alpha)$ for i = 1, 2, 3.

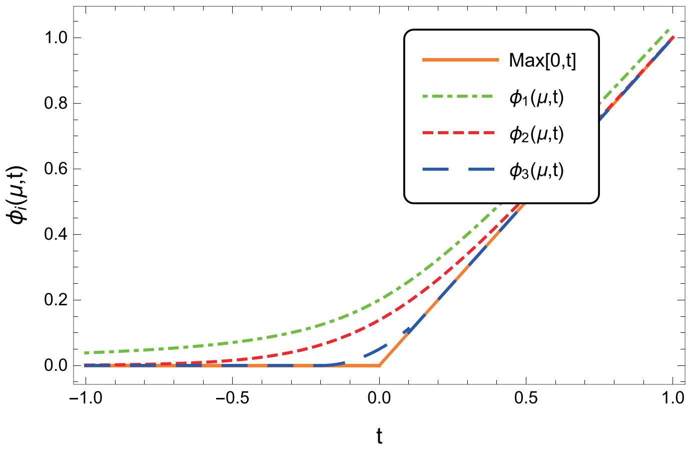

Figure 1: Graphs of $\max(0, t)$ and all three $\phi_i(\mu, t)$ with $\mu = 0.2$.

Proposition 2.2. *Suppose that $x = (x_1, x_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$ has the spectral decomposition given as in* (1.1)-(1.3)*, and that $\phi(\mu, \cdot)$ with $\mu > 0$ is continuously differentiable function satisfying* (2.2)*. Then, the following hold.*

(a) *The function $\Phi_\mu(x) : \mathbb{R}^n \to \mathbb{R}^n$ defined as in* (2.3) *is continuously differentiable. Moreover, its Jacobian matrix at x is described as*

$$\frac{\partial \Phi_\mu(x)}{\partial x} = \begin{cases} \frac{\partial \phi}{\partial \lambda}(\mu, x_1)I & \text{if} \quad x_2 = 0, \\ \begin{bmatrix} b & c{x_2}^T/\|x_2\| \\ cx_2/\|x_2\| & aI + (b-a)x_2{x_2}^T/\|x_2\|^2 \end{bmatrix} & \text{if} \quad x_2 \neq 0, \end{cases} \tag{2.4}$$

where

$$\begin{aligned} a &= \frac{\phi(\mu,\lambda_2(x)) - \phi(\mu,\lambda_1(x))}{\lambda_2(x) - \lambda_1(x)}, \\ b &= \frac{1}{2}\left(\frac{\partial \phi}{\partial \lambda_2}(\mu, \lambda_2(x)) + \frac{\partial \phi}{\partial \lambda_1}(\mu, \lambda_1(x))\right), \\ c &= \frac{1}{2}\left(\frac{\partial \phi}{\partial \lambda_2}(\mu, \lambda_2(x)) - \frac{\partial \phi}{\partial \lambda_1}(\mu, \lambda_1(x))\right); \end{aligned} \tag{2.5}$$

(b) *Both $\frac{\partial \Phi_\mu(x)}{\partial x}$ and $I - \frac{\partial \Phi_\mu(x)}{\partial x}$ are positive semi-definite matrices;*

(c) $\lim_{\mu \to 0} \Phi_\mu(x) = x_+ = (\lambda_1(x))_+ u_x^{(1)} + (\lambda_2(x))_+ u_x^{(2)}$ *for $i = 1, 2$.*

Proof. (a) From the expression (2.3) and the assumption of $\phi(\mu,\cdot)$ being continuously differentiable, it is easy to verify that the function Φ_μ is continuously differentiable. The Jacobian matrix (2.4) of $\Phi_\mu(x)$ can be obtained by adopting the same arguments as in [13, Proposition 5.2]. Hence, we omit the details here.

(b) First, we prove that the matrix $\frac{\partial \Phi_\mu(x)}{\partial x}$ is positive semi-definite. For the case of $x_2 = 0$, we know that $\frac{\partial \Phi_\mu(x)}{\partial x} = \frac{\partial \phi}{\partial \lambda}(\mu, x_1)I$. Then, from $0 \leq \frac{\partial \phi}{\partial \alpha}(\mu,\alpha) \leq 1$, it is clear to see that the matrix $\frac{\partial \Phi_\mu(x)}{\partial x}$ is positive semi-definite. For the case of $x_2 \neq 0$, from $\frac{\partial \phi}{\partial \alpha}(\mu,\alpha) \geq 0$ and (2.5), we have $b \geq 0$. In order to prove that the matrix $\frac{\partial \Phi_\mu(x)}{\partial x}$ is positive semi-definite, we only need to verify that the Schur Complement of b with respect to $\frac{\partial \Phi_\mu(x)}{\partial x}$ is positive semi-definite. Note that the Schur Complement of b has the form of

$$aI + (b-a)\frac{x_2x_2^T}{\|x_2\|^2} - \frac{c^2}{b}\frac{x_2x_2^T}{\|x_2\|^2} = a\left(I - \frac{x_2x_2^T}{\|x_2\|^2}\right) + \frac{b^2-c^2}{b}\frac{x_2x_2^T}{\|x_2\|^2}.$$

Since $\frac{\partial \phi}{\partial \alpha}(\mu,\alpha) \geq 0$, we obtain that the function $\phi(\mu,\alpha)$ with respect to α is increasing, which leads to $a \geq 0$. Besides, from (2.5), we observe that

$$b^2 - c^2 = \frac{\partial \phi}{\partial \lambda_2}(\mu, \lambda_2(x))\frac{\partial \phi}{\partial \lambda_1}(\mu, \lambda_1(x)) \geq 0.$$

With this, it follows that the Schur Complement of b with respect to $\frac{\partial \Phi_\mu(x)}{\partial x}$ is a linear non-negative combination of the matrices $\frac{x_2x_2^T}{\|x_2\|^2}$ and $I - \frac{x_2x_2^T}{\|x_2\|^2}$. Thus, we show that the Schur Complement of b is positive semi-definite, which says the matrix $\frac{\partial \Phi_\mu(x)}{\partial x}$ is positive semi-definite.

Combining with $\frac{\partial \phi}{\partial \alpha}(\mu,\alpha) \leq 1$ and following similar arguments as above, we can also argue that the matrix $I - \frac{\partial \Phi_\mu(x)}{\partial x}$ is also positive semi-definite.

(c) By the definition of the function $\Phi_\mu(x)$, it can be verified directly. □

We point out that the definition of (2.3) includes the similar way to define smoothing functions in [13, Section 4] as a special case, and hence [13, Prop. 4.1] is covered by Proposition 2.2. Indeed, Proposition 2.2 can be also verified by geometric views. More specifically, from Figures 2, 3 and 4, we see that when $\mu \downarrow 0$, ϕ_i is getting closer to $(\alpha)_+$, which verifies Proposition 2.2(c).

$\boxed{3}$ Applying Φ_μ to solve the system (1.5)

In this section, in light of the smoothing vector-valued function Φ_μ, we reformulate (1.5) as a system of smoothing equations. To this end, we need a partial order induced by SOC. More specifically, for any $x \in \mathbb{R}^n$, using the definition of the partial order "$\preceq_{\mathcal{K}^m}$" and the projection function x_+ in (2.1), we have

$$f_I(x) \preceq_{\mathcal{K}^m} 0 \iff -f_I(x) \in \mathcal{K}^m \iff f_I(x) \in -\mathcal{K}^m \iff (f_I(x))_+ = 0.$$

Hence, the system (1.5) is equivalent to the following system of equations:

$$\left\{ \begin{array}{rcl} (f_I(x))_+ & = & 0, \\ f_E(x) & = & 0. \end{array} \right. \tag{3.1}$$

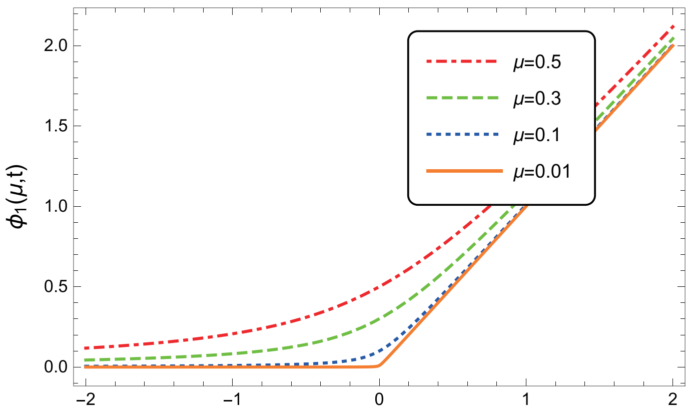

Figure 2: Graphs of $\phi_1(\mu, \alpha)$ with $\mu = 0.01, 0.1, 0.3, 0.5$.

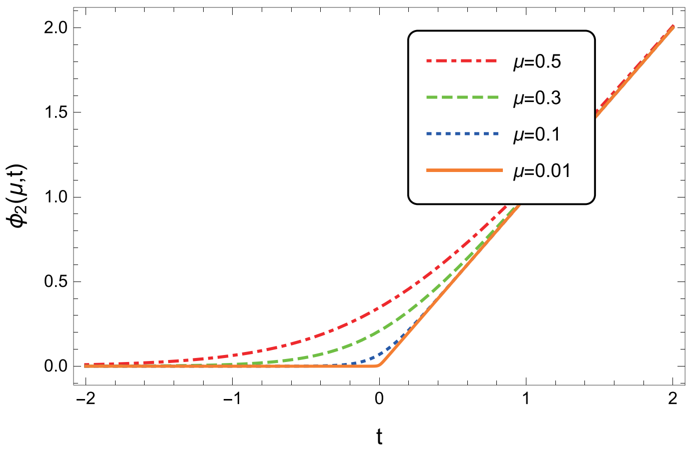

Figure 3: Graphs of $\phi_2(\mu, \alpha)$ with $\mu = 0.01, 0.1, 0.3, 0.5$.

Note that the function $(f_I(\cdot))_+$ in the above equation (3.1) is nonsmooth. Therefore, the smoothing-type Newton methods cannot be directly applied to solve the equation (3.1). To conquer this, we employ the smoothing function $\Phi_\mu(\cdot)$ defined in (2.3), and define the following function:

$$F(\mu, x, y) := \begin{bmatrix} f_I(x) - y \\ f_E(x) \\ \Phi_\mu(y) \end{bmatrix}.$$

From Proposition 2.2(c), it follows that

$$\begin{aligned} & F(\mu, x, y) = 0 \quad \text{and} \quad \mu = 0 \\ \Longleftrightarrow \quad & y = f_I(x),\ f_E(x) = 0,\ \Phi_\mu(y) = 0 \ \text{ and } \ \mu = 0 \\ \Longleftrightarrow \quad & y = f_I(x),\ f_E(x) = 0 \ \text{ and } \ y_+ = 0 \\ \Longleftrightarrow \quad & (f_I(x))_+ = 0,\ f_E(x) = 0 \\ \Longleftrightarrow \quad & f_I(x) \preceq_{\mathcal{K}^m} 0,\ f_E(x) = 0. \end{aligned}$$

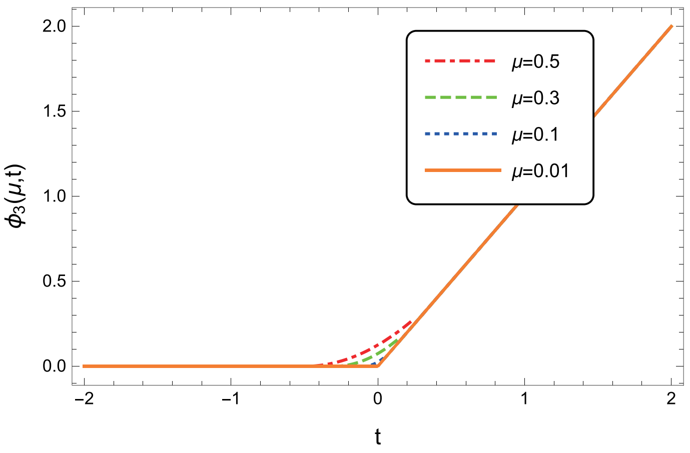

Figure 4: Graphs of $\phi_3(\mu,\alpha)$ with $\mu = 0.01, 0.1, 0.3, 0.5$.

In other words, as long as the system $F(\mu,x,y)=0$ and $\mu=0$ is solved, the corresponding x is a solution to the original system (1.5). In view of Proposition 2.2(a), we can obtain the solution to the system (1.5) by applying smoothing-type Newton method for solving $F(\mu,x,y)=0$ and setting $\mu\downarrow 0$ at the same time. To do this, for any $z=(\mu,x,y)\in \mathbb{R}_{++}\times \mathbb{R}^n\times\mathbb{R}^m$, we further define a continuously differentiable function $H:\mathbb{R}_{++}\times\mathbb{R}^n\times\mathbb{R}^m\to \mathbb{R}_{++}\times\mathbb{R}^n\times\mathbb{R}^m$ as follows:

$$H(z) := \begin{bmatrix} \mu \\ f_I(x)-y+\mu x_I \\ f_E(x)+\mu x_E \\ \Phi_\mu(y)+\mu y \end{bmatrix}, \tag{3.2}$$

where $x_I := (x_1,x_2,...,x_m)^T\in\mathbb{R}^m$, $x_E := (x_{m+1},...,x_n)^T\in\mathbb{R}^{n-m}$, $x := (x_I^T, x_E^T)^T\in\mathbb{R}^n$ and $y\in\mathbb{R}^m$. Then, it is clear to see that when $H(z)=0$, we have $\mu=0$ and x is a solution to the system (1.5). Now, we let $H'(z)$ denote the Jacobian matrix of the function H at z, then for any $z\in\mathbb{R}_{++}\times\mathbb{R}^n\times\mathbb{R}^m$, we obtain that

$$H'(z) = \begin{bmatrix} 1 & 0_n & 0_m \\ x_I & f_I'+\mu U & -I_m \\ x_E & f_E'+\mu V & 0_{(n-m)\times m} \\ \frac{\partial \Phi_\mu(y)}{\partial\mu}+y & 0_{m\times n} & \frac{\partial\Phi_\mu(y)}{\partial y}+\mu I_m \end{bmatrix}, \tag{3.3}$$

where $U := \begin{bmatrix} I_m & 0_{m\times(n-m)}\end{bmatrix}$, $V := \begin{bmatrix} 0_{(n-m)\times m} & I_{n-m}\end{bmatrix}$, 0_l denotes l dimensional zero vector, and $0_{l\times q}$ denotes $l\times q$ zero matrix for any positive integer l and q. In summary, we will apply smoothing-type Newton method to solve the smoothed equation $H(z)=0$ at each iteration and make $\mu>0$ as well as $H(z)\to 0$ to find a solution of the system (1.5).

$\boxed{4}$ A smoothing-type Newton algorithm

Now, we consider a smoothing-type Newton algorithm with a nonmonotone line search, and show that the algorithm is well defined. For convenience, we denote the merit function Ψ as $\Psi(z):=\|H(z)\|^2$ for any $z\in\mathbb{R}_{++}\times\mathbb{R}^n\times\mathbb{R}^m$.

Algorithm 4.1. (A smoothing-type Newton Algorithm)

Step 0 Choose $\gamma \in (0,1)$, $\xi \in (0, \frac{1}{2})$. Take $\eta > 0$, $\sigma \in (0,1)$ such that $\sigma\eta < 1$. Let $\mu_0 = \eta$ and $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$ be an arbitrary vector. Set $z^0 = (\mu_0, x^0, y^0)$, $e^0 := (1, 0, ..., 0) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m$, $G_0 := \|H(z^0)\|^2 = \Psi(z^0)$ and $S_0 := 1$. Choose $\beta_{\min}$ and $\beta_{\max}$ such that $0 \leq \beta_{\min} \leq \beta_{\max} < 1$. Set $\tau(z^0) := \sigma \min\{1, \Psi(z^0)\}$ and $k := 0$.

Step 1 If $\|H(z^k)\| = 0$, stop. Otherwise, go to Step 2.

Step 2 Compute $\Delta z^k := (\Delta\mu_k, \Delta x^k, \Delta y^k) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m$ by

$$H'(z^k)\Delta z^k = -H(z^k) + \eta\tau(z^k)e^0. \tag{4.1}$$

Step 3 Let α_k be the maximum of the values $1, \gamma, \gamma^2, ...$ such that

$$\Psi(z^k + \alpha_k \Delta z^k) \leq [1 - 2\xi(1 - \sigma\eta)\alpha_k]\, G_k. \tag{4.2}$$

Step 4 Set $z^{k+1} := z^k + \alpha_k \Delta z^k$. If $\|H(z^{k+1})\| = 0$, stop. Otherwise, go to Step 5.

Step 5 Choose $\beta_k \in [\beta_{min}, \beta_{max}]$. Set

$$\begin{array}{rcl} S_{k+1} & := & \beta_k S_k + 1, \\ \tau(z^{k+1}) & := & \min\left\{\sigma, \sigma\Psi(z^{k+1}), \tau(z^k)\right\}, \\ G_{k+1} & := & \left(\beta_k S_k G_k + \Psi(z^{k+1})\right)/S_{k+1}, \end{array} \tag{4.3}$$

and set $k := k + 1$. Go to Step 2.

The nonmonotone line search technique in Algorithm 4.1 was introduced in [36]. From the first and third equations in (4.3), we know that G_{k+1} is a convex combination of G_k and $\Psi(z^{k+1})$. In fact, G_k is expressed as a convex combination of $\Psi(z^0), \Psi(z^1), ..., \Psi(z^k)$. Moreover, the main role of β_k is to control the degree of non-monotonicity. If $\beta_k = 0$ for every k, then the corresponding line search is the usual monotone Armijo line search.

Proposition 4.2. *Suppose that the sequences $\{z^k\}$, $\{\mu_k\}$, $\{G_k\}$, $\{\Psi(z^k)\}$ and $\{\tau(z^k)\}$ are generated by Algorithm 4.1. Then, the following hold.*

(a) *The sequence $\{G_k\}$ is monotonically decreasing and $\Psi(z^k) \leq G_k$ for all $k \in \mathbb{N}$;*

(b) *The sequence $\{\tau(z^k)\}$ is monotonically decreasing;*

(c) *$\eta\tau(z^k) \leq \mu_k$ for all $k \in \mathbb{N}$;*

(d) *The sequence $\{\mu_k\}$ is monotonically decreasing and $\mu_k > 0$ for all $k \in \mathbb{N}$.*

Proof. The proof is similar to Remark 3.1 in [37], we omit the details. □

Next, we show that Algorithm 4.1 is well-defined and establish its local quadratic convergence. For simplicity, we denote the Jacobian matrix of the function f by

$$f'(x) := \left[\begin{array}{c} f'_I(x) \\ f'_E(x) \end{array} \right]$$

and use the following assumption.

Assumption 4.1. $f'(x) + \mu I_n$ is invertible for any $x \in \mathbb{R}^n$ and $\mu \in \mathbb{R}_{++}$.

We point our that the Assumption 4.1 is only a mild condition and there are many functions satisfying the assumption. For example, if f is a monotone function, then $f'(x)$ is a positive semi-definite matrix for any $x \in \mathbb{R}^n$. Thus, Assumption 4.1 is satisfied.

Theorem 4.3. *Suppose that f is a continuously differentiable function and Assumption 4.1 is satisfied. Then, Algorithm 4.1 is well-defined.*

Proof. In order to show that Algorithm 4.1 is well-defined, we need to prove that Newton equation (4.1) is solvable, and the line search (4.2) is well-defined.

First, we prove that Newton equation (4.1) is solvable. By the expression of Jacobian matrix $H'(z)$ in (3.3), we see that the determinant $\det(H'(z))$ of $H'(z)$ satisfies

$$\det(H'(z)) = \det\left(f'(x) + \mu I_n\right) \cdot \det\left(\frac{\partial \Phi_\mu(y)}{\partial y} + \mu I_m\right)$$

for any $z \in \mathbb{R}_{++} \times \mathbb{R}^n \times \mathbb{R}^m$. Moreover, from Proposition 2.2(b), we know that $\frac{\partial \Phi_\mu(y)}{\partial y}$ is positive semi-definite for $\mu \in \mathbb{R}_{++}$. Hence, combing this with Assumption 4.1, we obtain that $H'(z)$ is nonsingular for any $z \in \mathbb{R}_{++} \times \mathbb{R}^n \times \mathbb{R}^m$ with $\mu > 0$. Applying Proposition 4.2(d), it follows that Newton equation (4.1) is solvable.

Secondly, we prove that the line search (4.2) is well-defined. For notational convenience, we denote

$$w_k(\alpha) := \Psi\left(z^k + \alpha \Delta z^k\right) - \Psi\left(z^k\right) - \alpha \Psi'\left(z^k\right) \Delta z^k.$$

From Newton equation (4.1) and the definition of Ψ, we have

$$\begin{aligned} \Psi\left(z^k + \alpha \Delta z^k\right) &= w_k(\alpha) + \Psi\left(z^k\right) + \alpha \Psi'\left(z^k\right) \Delta z^k \\ &= w_k(\alpha) + \Psi\left(z^k\right) + 2\alpha H\left(z^k\right)^T \left(-H(z^k) + \eta\tau(z^k) e^0\right) \\ &\leq w_k(\alpha) + (1-2\alpha)\Psi\left(z^k\right) + 2\alpha\eta\tau(z^k) \left\|H(z^k)\right\|. \end{aligned}$$

If $\Psi(z^k) \leq 1$, then we have $\|H(z^k)\| \leq 1$. Hence, it follows that

$$\tau(z^k)\|H(z^k)\| \leq \sigma\Psi(z^k)\|H(z^k)\| \leq \sigma\Psi(z^k).$$

If $\Psi(z^k) > 1$, then we see that $\Psi(z^k) = \|H(z^k)\|^2 \geq \|H(z^k)\|$, which yields

$$\tau(z^k)\|H(z^k)\| \leq \sigma\|H(z^k)\| \leq \sigma\Psi(z^k).$$

Thus, from all the above, we obtain that

$$\begin{aligned} \Psi\left(z^k + \alpha \Delta z^k\right) &\leq w_k(\alpha) + (1-2\alpha)\Psi(z^k) + 2\alpha\eta\sigma\Psi(z^k) \\ &= w_k(\alpha) + \left[1 - 2(1-\sigma\eta)\alpha\right]\Psi(z^k) \qquad (4.4) \\ &\leq w_k(\alpha) + \left[1 - 2(1-\sigma\eta)\alpha\right] G_k. \end{aligned}$$

Since the function H is continuous and differentiable for any $z \in \mathbb{R}_{++} \times \mathbb{R}^n \times \mathbb{R}^m$, we have $w_k(\alpha) = o(\alpha)$ for all $k \in \mathbb{N}$. Combining with (4.4), this indicates that the line search (4.2) is well-defined. □

Theorem 4.4. *Suppose that f is a continuously differentiable function and Assumption 4.1 is satisfied. Then the sequence $\{z^k\}$ generated by Algorithm 4.1 is bounded; and any accumulation point of the sequence $\{x^k\}$ is a solution of the system (1.5).*

Proof. The proof is similar to [37, Theorem 4.1] and we omit it. □

In Theorem 4.4, we give the global convergence of Algorithm 4.1. Now, we analyze the convergence rate for Algorithm 4.1. We start with introducing the following concepts. A locally Lipschitz function $F : \mathbb{R}^n \to \mathbb{R}^m$ is said to be semismooth (or strongly semismooth) at $x \in \mathbb{R}^n$ if F is directionally differentiable at x and

$$F(x+h) - F(h) - Vh = o(\|h\|) \quad (or = O(\|h\|^2))$$

holds for any $V \in \partial F(x+h)$, where $\partial F(x)$ is the generalized Jacobian matrix of the function F at $x \in \mathbb{R}^n$ in the sense of Clarke [2]. There are many functions being semismooth, such as convex functions, smooth functions, piecewise linear functions and so on. In addition, it is known that the composition of semismooth functions is still a semismooth function, and the composition of strongly semismooth functions is still a strongly semismooth function [12]. From Proposition 2.2 (a), we know that $\Phi_\mu(x)$ defined by (2.3) is smooth on $\mathbb{R}^n$.

With the definition (3.2) of H, mimicking the arguments as in [37, Theorem 5.1], we have the local quadratic convergence of Algorithm 4.1.

Theorem 4.5. *Suppose that the conditions given in Theorem 4.4 are satisfied, and $z^* = (\mu_*, x^*, y^*)$ is an accumulation point of sequence $\{z^k\}$ which is generated by Algorithm 4.1.*

(a) *If all $V \in \partial H(z^*)$ are nonsingular, then the sequence $\{z^k\}$ converges to z^*, and*

$$\|z^{k+1} - z^k\| = o(\|z^k - z^*\|), \quad \mu_{k+1} = o(\mu_k);$$

(b) *If the functions f and Φ_μ satisfy that f' and Φ_μ' are Lipschitz continuous on $\mathbb{R}^n$, then $\|z^{k+1} - z^k\| = O(\|z^k - z^*\|)^2$ and $\mu_{k+1} = O(\mu_k^2)$.*

$\boxed{5}$ Numerical experiments

In this section, we present some numerical examples to demonstrate the efficiency of Algorithm 4.1 for solving the system (1.5). In our tests, all experiments are done on a PC with CPU of 1.9 GHz and RAM of 8.0 GB, and all the program codes are written in MATLAB and run in MATLAB environment. We point out that if there are no n numbers in $I \cup E$, we can adopt a similar way to those given in [37], then the system (1.5) can be transformed as a new problem and we can solve the new problem using Algorithm 4.1. By this approach, a solution of the original problem can be found.

Throughout the following experiments, we employ three functions ϕ_1, ϕ_2 and ϕ_3 along with the proposed algorithm to implement each example. Note that, for the function ϕ_1, its corresponding SOC-function Φ_μ can be alternatively expressed as

$$\widetilde{\Phi}_\mu(x) = \frac{x + \sqrt{x^2 + 4\mu^2 e}}{2} \quad \text{with} \quad e = (1, 0, \cdots, 0)^T \in \mathcal{K}^n.$$

This form is simpler than the $\Phi_\mu(x)$ induced from (2.3). Hence, we adopt it in our implementation. Moreover, the parameters used in the algorithm are chosen as follows:

$$\gamma = 0.3, \quad \xi = 10^{-4}, \quad \eta = 1.0, \quad \beta_0 = 0.01, \quad \mu_0 = 1.0, \quad S_0 = 1.0,$$

and the parameters c and σ are chosen according to the ones listed in Table 1 and Table 4. In the implementation, the stopping rule is $\|H(z)\| \leq 10^{-6}$, the step length $\nu \leq 10^{-6}$, or the number of iteration is over 500; and the starting points are randomly generated from the interval $[-1, 1]$.

Now, we present the test examples. We first consider two examples in which the system (1.5) only includes inequalities, i.e., $m = n$. Note that a similar way to construct the two examples was given in [25].

Example 5.1. Consider the system (1.5) with inequalities only, where $f(x) := Mx+q \preceq_{\mathcal{K}^n} 0$ and $\mathcal{K}^n := \mathcal{K}^{n_1} \times \cdots \times \mathcal{K}^{n_r}$. Here M is generated by $M = BB^T$ with $B \in \mathbb{R}^{n\times n}$ being a matrix whose every component is randomly chosen from the interval $[0, 1]$ and $q \in \mathbb{R}^n$ being a vector whose every component is 1.

For Example 5.1, the tested problems are generated with sizes $n = 500, 1000, ..., 4500$ and each $n_i = 10$. The random problems of each size are generated 10 times. Besides using the three functions along with Algorithm 4.1 for solving Example 5.1, we have also tested it by using the smoothing-type algorithm with the monotone line search which was introduced in [25] (for this case, we choose the function ϕ_1). Table 1 shows the numerical results where

"fun"	denotes the three functions,
"suc"	denotes the number that Algorithm 4.1 successfully solves every generated problem,
"iter"	denotes the average iteration numbers,
"cpu"	denotes the average CPU time in seconds,
"res"	denotes the average residual norm $\|H(z)\|$ for 9 test problems.

The initial points are also randomly generated. In light of "iter" and "cpu" in Table 1, we can conclude that

$$\phi_3(\mu, \alpha) > \phi_1(\mu, \alpha) > \phi_2(\mu, \alpha)$$

where ">" means "better performance". In Table 2, we compare Algorithm 4.1 with non-monotone line search and the smoothing-type algorithm with monotone line search studied in [25]. Although the number that Algorithm 4.1 successfully solves every generated problem is less than the one by the smoothing-type algorithm with monotone line search as aforementioned in overall, the performance based on cpu time and iterations of our proposed algorithm outperforms better than the other. This indicates that Algorithm 4.1 has some advantages over the one with the monotone line search in [25].

Another way to compare the performance of function $\phi_i(\mu, \alpha), i = 1, 2, 3$, is via the so-called "performance profile", which is introduced in [39]. In this means, we regard Algorithm 4.1 corresponding to a smoothing function $\phi_i(\mu, \alpha), i = 1, 2, 3$ as a solver, and assume that there are n_s solvers and n_p test problems from the test set $\mathcal{P}$ which is generated randomly. We are interested in using the iteration number as performance measure for Algorithm 4.1 with different $\phi_i(\mu, \alpha)$. For each problem p and solver s, let

$$f_{p,s} = \text{iteration number required to solve problem } p \text{ by solver } s.$$

n	fun	suc	iter	cpu	res
500	ϕ_1	10	5.000	0.251	8.864e-09
500	ϕ_2	10	7.800	1.496	2.600e-07
500	ϕ_3	10	3.500	0.707	3.762e-07
1000	ϕ_1	10	5.000	0.632	2.165e-08
1000	ϕ_2	10	7.200	5.240	8.657e-08
1000	ϕ_3	10	3.400	3.093	4.853e-07
1500	ϕ_1	9	5.000	1.224	1.537e-07
1500	ϕ_2	9	8.111	13.232	3.124e-07
1500	ϕ_3	9	4.222	8.781	2.706e-07
2000	ϕ_1	10	5.000	2.145	1.599e-07
2000	ϕ_2	10	7.700	24.130	2.234e-07
2000	ϕ_3	10	4.200	16.925	1.923e-07
2500	ϕ_1	9	5.000	3.519	3.897e-08
2500	ϕ_2	9	6.889	34.849	2.016e-07
2500	ϕ_3	9	4.000	27.870	1.479e-07
3000	ϕ_1	10	5.000	5.161	9.769e-08
3000	ϕ_2	10	8.300	69.723	1.714e-07
3000	ϕ_3	10	4.100	45.891	1.608e-07
3500	ϕ_1	7	5.000	7.415	2.226e-07
3500	ϕ_2	7	7.857	102.272	4.037e-07
3500	ϕ_3	7	4.429	75.068	2.334e-07
4000	ϕ_1	9	5.000	9.974	5.795e-08
4000	ϕ_2	9	6.444	106.850	3.132e-07
4000	ϕ_3	9	4.000	98.983	7.743e-08
4500	ϕ_1	8	5.000	13.075	2.374e-07
4500	ϕ_2	8	10.250	240.602	3.115e-07
4500	ϕ_3	8	4.250	147.863	3.070e-07

Table 1: Average performance of Algorithm4.1 for Example 5.1 ($c = 0.01, \sigma = 10^{-5}$)

We employ the performance ratio

$$r_{p,s} := \frac{f_{p,s}}{\min\{f_{p,s} : s \in \mathcal{S}\}},$$

where $\mathcal{S}$ is the four solvers set. We assume that a parameter $r_{p,s} \leq r_M$ for all p, s are chosen, and $r_{p,s} = r_M$ if and only if solver s does not solve problem p. In order to obtain an overall assessment for each solver, we define

$$\rho_s(\tau) := \frac{1}{n_p}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\},$$

which is called the performance profile of the number of iteration for solver s. Then, $\rho_s(\tau)$ is the probability for solver $s \in \mathcal{S}$ that a performance ratio $f_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio.

We then need to test the three functions for Example 5.1. In particular, the random problems of each size are generated 50 times. In order to obtain an overall assessment for the

Non-monotone					Monotone				
n	suc	iter	cpu	res	n	suc	iter	cpu	res
500	10	5.000	0.251	8.864e-09	500	10	5.500	0.289	4.905e-07
1000	10	5.000	0.632	2.165e-08	1000	10	5.500	0.616	7.184e-08
1500	9	5.000	1.224	1.537e-07	1500	9	6.000	1.466	4.654e-09
2000	10	5.000	2.145	1.599e-07	2000	10	6.500	2.866	3.151e-08
2500	9	5.000	3.519	3.897e-08	2500	10	6.000	4.477	4.320e-08
3000	10	5.000	5.161	9.769e-08	3000	10	6.500	7.348	1.743e-07
3500	7	5.000	7.415	2.226e-07	3500	10	8.000	11.957	5.674e-07
4000	9	5.000	9.974	5.795e-08	4000	10	7.000	14.875	2.166e-08
4500	8	5.000	13.075	2.374e-07	4500	10	7.000	19.204	2.433e-08

Table 2: Comparisons of non-monotone Algorithm 4.1 and monotone Algorithm in [25] for Example 5.1

three functions, we are interested in using the number of iterations as a performance measure for Algorithm 4.1 with $\phi_1(\mu,\alpha)$, $\phi_2(\mu,\alpha)$, and $\phi_3(\mu,\alpha)$, respectively. The performance plot based on iteration number is presented in Figure 5. From this figure, we also see that $\phi_3(\mu,\alpha)$ working with Algorithm 4.1 has the best numerical performance, followed by $\phi_4(\mu,\alpha)$. In other words, in view of "iteration numbers", there has

$$\phi_3(\mu,\alpha) > \phi_1(\mu,\alpha) > \phi_2(\mu,\alpha)$$

where ">" means "better performance".

We are also interested in using the computing time as performance measure for Algorithm 4.1 with different $\phi_i(\mu,\alpha), i = 1,2,3$. The performance plot based on computing time is presented in Figure 6. From this figure, we can also see the function $\phi_3(\mu,t)$ has best performance. In other words, in view of "computing time", there has

$$\phi_3(\mu,\alpha) > \phi_1(\mu,\alpha) > \phi_2(\mu,\alpha)$$

where ">" means "better performance".

In summary, for the Example 5.1, no matter the number of iterations or the computing time is taken into account, the function $\phi_3(\mu,\alpha)$ is the best choice for the Algorithm 4.1.

Example 5.2. Consider the system (1.5) with inequalities only, where $x \in \mathbb{R}^5$, $\mathcal{K}^5 = \mathcal{K}^3 \times \mathcal{K}^2$ and

$$f(x) := \left[\begin{array}{c} 24(2x_1 - x_2)^3 + exp(x_1 + x_3) - 4x_4 + x_5 \\ -12(2x_1 - x_2)^3 + 3(3x_2 + 5x_3)/\sqrt{1 + (3x_2 + 5x_3)^2} - 6x_4 - 7x_5 \\ -exp(x_1 - x_3) + 5(3x_2 + 5x_3)/\sqrt{1 + (3x_2 + 5x_3)^2} - 3x_4 + 5x_5 \\ 4x_1 + 6x_2 + 3x_3 - 1 \\ -x_1 + 7x_2 - 5x_3 + 2 \end{array} \right] \preceq_{\mathcal{K}^5} 0.$$

This problem is taken from [17].

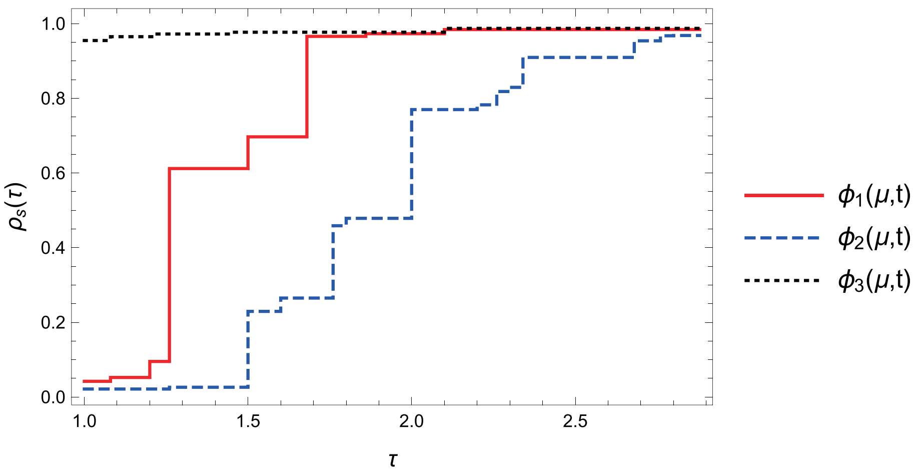

Figure 5: Performance profile of iteration numbers for Example 5.1.

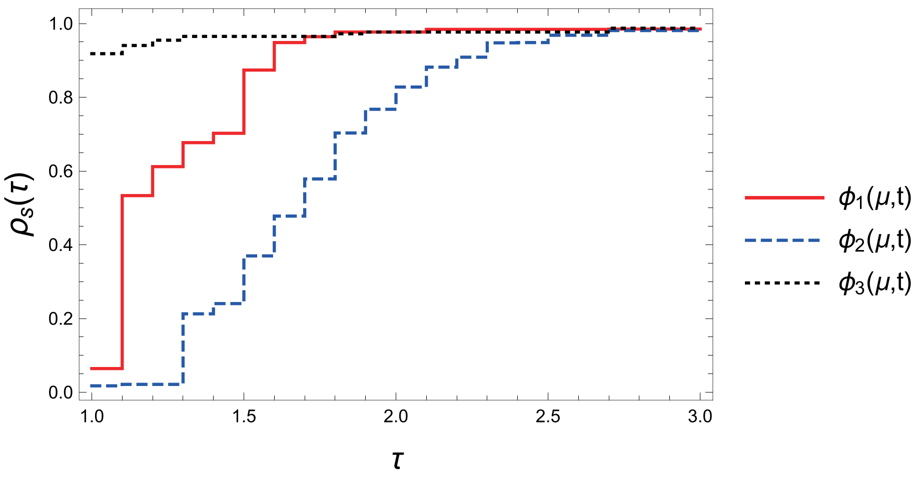

Figure 6: Performance profile of computing time for Example 5.1.

Example 5.2 is tested 20 times for 20 random starting points. Similar to the case of Example 5.1, besides using Algorithm 4.1 to test Example 5.2, we have also tested it using the monotone smoothing-type algorithm in [25]. From Table 3, we see that there is no big difference regarding performance between these two algorithms for Example 5.2.

Moreover, Figure 7 shows the performance profile of iteration number in Algorithm 4.1 for Example 5.2 on 100 test problems with random starting points. The three solvers correspond to Algorithm 4.1 with $\phi_1(\mu,\alpha)$, $\phi_2(\mu,\alpha)$, and $\phi_3(\mu,\alpha)$, respectively. From this figure, we see that $\phi_3(\mu,\alpha)$ working with Algorithm 4.1 has the best numerical performance. followed by $\phi_2(\mu,t)$. In summary, from the viewpoint of "iteration numbers", we conclude that

$$\phi_3(\mu,\alpha) > \phi_2(\mu,\alpha) > \phi_1(\mu,\alpha),$$

where ">" means "better performance".

Non-monotone				Monotone			
suc	iter	cpu	res	suc	iter	cpu	res
20	13.500	0.002	5.835e-08	20	8.750	0.005	1.2510e-07

Table 3: Comparisons of non-monotone Algorithm 4.1 and monotone Algorithm in [25] for Example 5.2

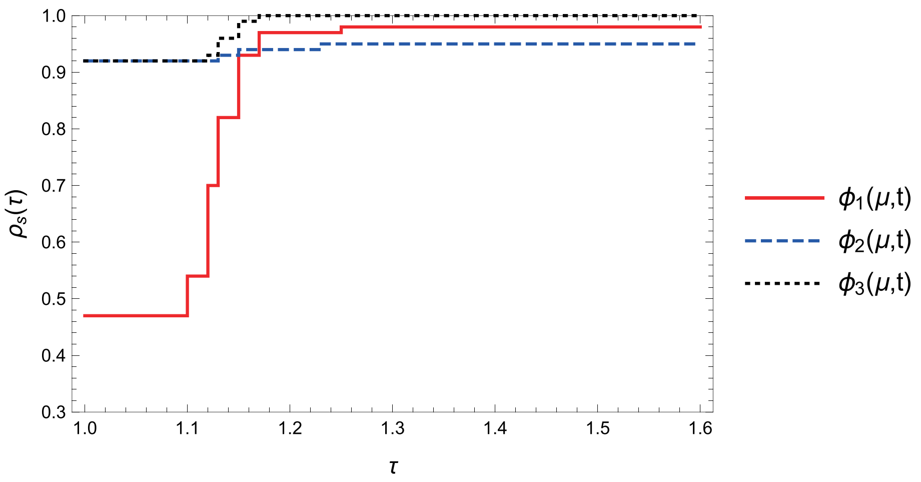

Figure 7: Performance profile of iteration number for Example 5.2.

Example 5.3. Consider the system of equalities and inequalities (1.5), where

$$f(x) := \left(f_I(x)^T, f_E(x)^T\right)^T \quad x \in \mathbb{R}^6,$$

with

$$\begin{aligned} f_I(x) &= \begin{bmatrix} -x_1^4 \\ 3x_2^3 + 2x_2 - x_3 - 5x_3^2 \\ -4x_2^2 - 7x_3 + 10x_3^3 \\ -x_4^3 - x_5 \\ x_5 + x_6 \end{bmatrix} \preceq_{\mathcal{K}^5 = \mathcal{K}^3 \times \mathcal{K}^2} 0, \\ f_E(x) &= 2x_1 + 5x_2^2 - 3x_3^2 + 2x_4 - x_5 x_6 - 7. \end{aligned}$$

Example 5.4. Consider the system of equalities and inequalities (1.5), where

$$f(x) := \left(f_I(x)^T, f_E(x)^T\right)^T \quad x \in \mathbb{R}^6,$$

with

$$\begin{aligned} f_I(x) &= \begin{bmatrix} -e^{5x_1} + x_2 \\ x_2 + x_3^3 \\ -3e^{x_4} \\ 5x_5 - x_6 \end{bmatrix} \preceq_{\mathcal{K}^4 = \mathcal{K}^2 \times \mathcal{K}^2} 0, \\ f_E(x) &= \begin{bmatrix} 3x_1 + e^{x_2 + x_3} - 2x_4 - 7x_5 + x_6 - 3 \\ 2x_1^2 + x_2 + 3x_3 - (x_4 - x_5)^2 + 2x_6 - 13 \end{bmatrix} = 0. \end{aligned}$$

Example 5.5. Consider the system of equalities and inequalities (1.5), where

$$f(x) := \left(f_I(x)^T, f_E(x)^T\right)^T \quad x \in \mathbb{R}^7,$$

with

$$\begin{aligned} f_I(x) &= \begin{bmatrix} 3x_1^3 \\ x_2 - x_3 \\ -2(x_4-1)^2 \\ \sin(x_5+x_6) \\ 2x_6 + x_7 \end{bmatrix} \preceq_{\mathcal{K}^5=\mathcal{K}^2\times\mathcal{K}^3} 0, \\ f_E(x) &= \begin{bmatrix} x_1 + x_2 + 2x_3x_4 + \sin x_5 + \cos x_6 + 2x_7 \\ x_1^3 + x_2 + \sqrt{x_3^2+3} + 2x_4 + x_5 + x_6 + 6x_7 \end{bmatrix} = 0. \end{aligned}$$

Exam	fun	suc	c	σ	iter	cpu	res
5.2	ϕ_1	20	5	0.02	13.500	0.002	5.835e-08
5.2	ϕ_2	20	5	0.02	8.450	0.001	5.134e-07
5.2	ϕ_3	20	5	0.02	8.600	0.002	2.260e-07
5.3	ϕ_1	20	1	0.02	21.083	0.009	8.165e-07
5.3	ϕ_2	17	1	0.02	14.647	0.001	2.899e-07??
5.3	ϕ_3	17	1	0.02	18.529	0.002	7.167e-07
5.4	ϕ_1	20	0.5	0.002	46.750	0.033	1.648e-07
5.4	ϕ_2	2	0.5	0.002	420.000	0.499	9.964e-07
5.4	ϕ_3	0	0.5	0.002	Fail	Fail	Fail
5.5	ϕ_1	20	0.1	0.002	14.250	0.009	6.251e-07
5.5	ϕ_2	20	0.1	0.002	13.250	0.001	6.532e-07
5.5	ϕ_3	20	0.1	0.002	12.650	0.001	6.016e-07

Table 4: Average performance of Algorithm4.1 for Examples 5.2-5.5

Table 4 shows the numerical results including three smoothing functions (fun) used to solve the problems, the number (suc) that Algorithm 4.1 successfully solves every generated problem, the parameters c and σ, the average iteration numbers (iter), the average CPU time (cpu) in seconds and the average residual norm $\|H(z)\|$ (res) for Examples 5.2-5.5 with random initializations, respectively. Performance profiles are provided as below.

Figure 8 and Figure 9 are the performance profiles in terms of iteration number for Example 5.3 and Example 5.5. From the Figure 8, we see that although the best probability of the function ϕ_3 is lower, but the ratio that can be solved in a large number of problems is higher than that of the other two. In this case, the difference between the three functions is not obvious. From the Figure 9, we can also see the function ϕ_3 has best performance.

In summary, below are our numerical observations and conclusions.

1. The Algorithm 4.1 is effective. In particular, the numerical results show that our proposed method is better than the algorithm with monotone line search studied in [25] when solving the system of inequalities under the order induced by second-order cone.

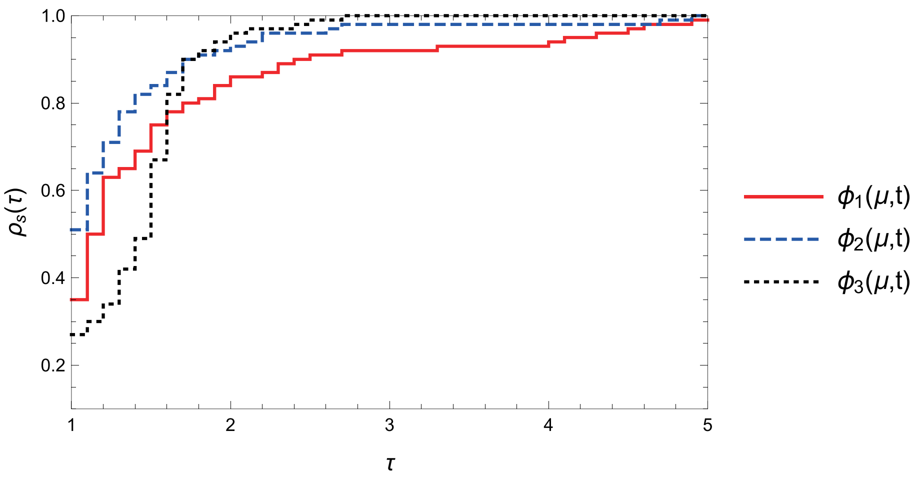

Figure 8: Performance profile of iteration number for Example 5.3.

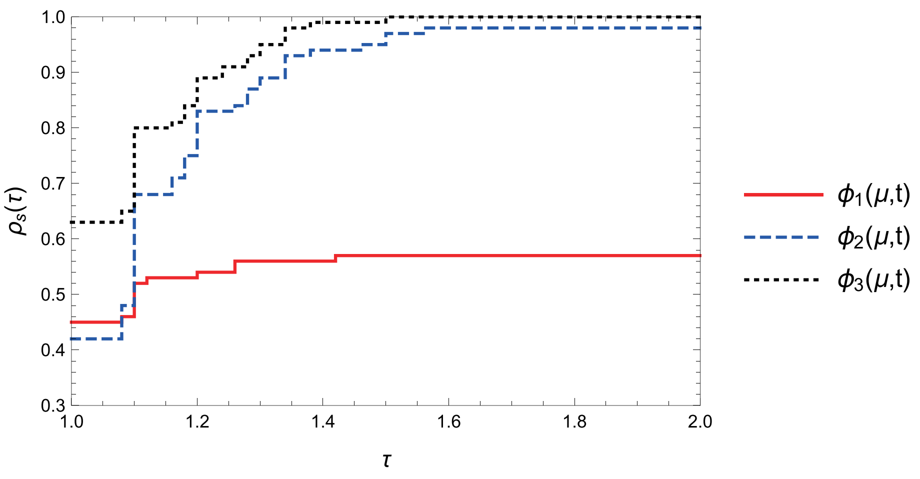

Figure 9: Performance profile of iteration number for Example 5.5.

2. For Examples 5.1 and 5.2, the function ϕ_3 outperforms much better than the others. For the rest problems, the difference of their numerical performance is very marginal.

3. For future topics, it is interesting to discover more efficient smoothing functions and to apply the type of SOC-functions to other optimization problems involved second-order cones.

References

[1] F. Alizadeh, D. Goldfarb, Second-order cone programming, *Math. Program.* 95 (2003) 3–51.

[2] F.H. Clark, *Optimizaton and Nonsmooth Analysis*, Wiley, New York, 1983.

[3] J-S. Chen, The convex and monotone functions associated with second-order cone, *Optimization.* 55 (2006) 363–385.

[4] J.-S. Chen, X. Chen, S.-H. Pan and J. Zhang, Some characterizations for SOC-monotone and SOC-convex functions, *J. Global Optim.* 45 (2009) 259–279.

[5] J-S. Chen, X. Chen and P. Tseng, Analysis of nonsmooth vector-valued functions associated with second-order cones, *Math. Program.* 101 (2004) 95–117.

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

[7] J.-S. Chen, T.-K. Liao and S.-H. Pan, Using Schur Complement Theorem to prove convexity of some SOC-functions, *J. Nonlinear Convex Anal.* 13 (2012) 421–431.

[8] J.-S. Chen and S.-H. Pan, A survey on SOC complementarity functions and solution methods for SOCPs and SOCCPs, *Pac. J. Optim.* 8 (2012) 33–74.

[9] J-S. Chen, C.-H. Ko, Y.-D. Liu and S.-P. Wang, New smoothing functions for solving a system of equalities and inequalities, *Pac. J. Optim.* 12 (2016) 185–206.

[10] J.W. Daniel, Newton's method for nonlinear inequalities, *Numer. Math.* 21 (1973) 381–387.

[11] U. Faraut and A. Korányi, *Analysis on Symmetric Cones*, Oxford Mathematical Monographs, Oxford University Press, New York, 1994.

[12] A. Fischer, Solution of monotone complementarity problems with locally Lipschitzian functions, *Math. Program.* 76 (1997) 513–532.

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

[14] F. Facchinei and J.S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Volume-I, Springer, New York, 2003.

[15] Z.-H. Huang, S.-L. Hu, and J. Han, Global convergence of a smoothing algorithm for symmetric cone complementarity problems with a nonmonotone line search, *Sci. China Ser. A* 52 (2009) 833–848.

[16] Z.-H. Huang and T. Ni, Smoothing algorithms for complementarity problems over symmetric cones, *Comput. Optim. Appl.* 45 (2010) 557–579.

[17] 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) 593–615.

[18] Z.H. Huang, Y. Zhang and W. Wu, A smoothing-type algorithm for solving a system of inequalities, *J. Comput. Appl. Math.* 220 (2008) 355–363.

[19] C. Kanzow, I. Ferenczi and M. Fukushima, On the local convergence of semismooth Newton methods for linear and nonlinear second-order cone programs without strict complementarity, *SIAM J. Optim.* 20 (2009) 297–320.

[20] L.-C. Kong, J. Sun, and N.-H. Xiu, A regularized smoothing Newton method for symmetric cone complementarity problems, *SIAM J. Optim.* 19 (2008) 1028–1047.

[21] X.-H. Liu and W.-Z. Gu, Smoothing Newton algorithm based on a regularized one-parametric class of smoothing functions for generalized complementarity problems over symmetric cones, *J. Ind. Manag. Optim.* 6 (2010) 363–380.

[22] N. Lu and Z.-H. Huang, Convergence of a non-interior continuation algorithm for the monotone SCCP, Acta Math. Appl. Sin. Engl. Ser. 26 (2010) 543–556.

[23] X.-H. Liu and Z.-H. Huang, A smoothing Newton algorithm based on a one-parametric class of smoothing functions for linear programming over symmetric cones, *Math. Methods Oper. Res.* 70 (2009) 385–404.

[24] M.S. Lobo, L. Vandenberghe, S. Boyd and H. Lebret, Applications of second-order cone programming, *Linear Algebra Appl.* 284 (1998) 193–228.

[25] N. Lu and Y. Zhang, A smoothing-type algorithm for solving inequalities under the order induced by a symmetric cone, *J. Inequal. Appl.* 4 (2011) 2011.

[26] Y.-J. Liu, L.-W. Zhang, and Y.-H. Wang, Analysis of smoothing method for symmetric conic linear programming, *J. Appl. Math. Comput.* 22 (2006) 133–148.

[27] M. Macconi, B. Morini and M. Porcelli, Trust-region quadratic methods for nonlinear systems of mixed equalities and inequalities, *Appl. Numer. Math.* 59 (2009) 859–876.

[28] D.Q. Mayne, E. Polak and A.J. Heunis, Solving nonlinear inequalities in a finite number of iterations, *J. Optim. Theory Appl.* 33 (1981) 207–221.

[29] S.-H. Pan and J.-S. Chen, A class of interior proximal-like algorithms for convex second-order cone programming, *SIAM J. Optim.* 19 (2008) 883–910.

[30] S.-H. Pan and J.-S. Chen, Interior proximal methods and central paths for convex second-order cone programming, *Nonlinear Anal.* 73 (2010) 3083–3100.

[31] S.-H. Pan and J.-S. Chen, A least-square semismooth Newton method for the second-order cone complementarity problem, *Optim. Methods Softw.* 26 (2011) 1–22.

[32] S.-H. Pan, Y. Chiang and J.-S. Chen, SOC-monotone and SOC-convex functions v.s. matrix-monotone and matrix-convex functions, *Linear Algebra Appl.* 437 (2012) 1264–1284.

[33] M. Sahba, On the solution of nonlinear inequalities in a finite number of iterations, *Numer. Math.* 46 (1985) 229–236.

[34] J.-H. Sun, J.-S. Chen, and C.-H. Ko, Neural networks for solving second-order cone constrained variational inequality problem, *Comput. Optim. Appl.* 51 (2012) 623–648.

[35] J. Wu and J.-S. Chen, A proximal point algorithm for the monotone second-order cone complementarity problem, *Comput. Optim. Appl.* 51 (2012) 1037–1063.

[36] H.-C. Zhang and W. Hager, A nonmontone line search technique and its application to unconstrained optimization, *SIAM J. Optim.* 14 (2004) 1043–1056.

[37] Y. Zhang and Z.-H. Huang, A nonmonotone smoothing-type algorithm for solving a system of equalities and inequalities, *J. Comput. Appl. Math.* 233 (2010) 2312–2321.

[38] J.G. Zhu, H.W. Liu and X.L. Li, A regularized smoothing-type algorithm for solving a system of inequalities with a P_0-function, *J. Comput. Appl. Math.* 233 (2010) 2611–2619.

[39] E. D. Dolan and J. J. More, Benchmarking optimization software with performance profiles, *Math. Program.* 91 (2002) 201–213.

Manuscript received 28 August 2016
revised 15 July 2017
accepted for publication 20 July 2017

Xin-He Miao
School of Mathematics, Tianjin University
E-mail address: xinhemiao@tju.edu.cn

Nuo Qi
School of Mathematics, Tianjin University
E-mail address: qinuotju@126.com

B. Saheya
College of Mathematical Science
Inner Mongolia Normal University
E-mail address: saheya@imnu.edu.cn

Jein-Shan Chen
Department of Mathematics
National Taiwan Normal University
E-mail address: jschen@math.ntnu.edu.tw