Applications of Mathematics

Jingyong Tang; Li Dong; Liang Fang; Li Sun

Analysis of a non-monotone smoothing-type algorithm for the second-order cone programming

Applications of Mathematics, Vol. 60 (2015), No. 1, 35–49

Persistent URL: http://dml.cz/dmlcz/144093

ANALYSIS OF A NON-MONOTONE SMOOTHING-TYPE ALGORITHM FOR THE SECOND-ORDER CONE PROGRAMMING

Jingyong Tang, Li Dong, Xinyang, Liang Fang, Li Sun, Tai'an

(Received April 18, 2013)

Abstract. The smoothing-type algorithm is a powerful tool for solving the second-order cone programming (SOCP), which is in general designed based on a monotone line search. In this paper, we propose a smoothing-type algorithm for solving the SOCP with a non-monotone line search. By using the theory of Euclidean Jordan algebras, we prove that the proposed algorithm is globally and locally quadratically convergent under suitable assumptions. The preliminary numerical results are also reported which indicate that the non-monotone smoothing-type algorithm is promising for solving the SOCP.

Keywords: second-order cone programming; smoothing Newton algorithm; non-monotone line search; convergence

MSC 2010: 90C25, 90C30, 65K05

1. Introduction

The *second-order cone* (SOC) in $\mathcal{R}^n$ $(n \geqslant 1)$, also called the Lorentz cone, is defined to be

$$\mathcal{K}^n := \{(x_1, \tilde{x}^{\mathrm{T}})^{\mathrm{T}} \in \mathcal{R} \times \mathcal{R}^{n-1} \colon\ x_1 \geqslant \|\tilde{x}\|\},$$

where $\|\cdot\|$ denotes the Euclidean norm. If $n = 1$, then $\mathcal{K}^1$ is the set of nonnegative reals $\mathcal{R}_+$ (the nonnegative orthant in $\mathcal{R}$).

The second-order cone programming (SOCP) problem is a class of convex optimization problems in which a linear function is minimized over the intersection of

This paper was partly supported by Excellent Young Scientist Foundation of Shandong Province (BS2011SF024, BS2012SF025), Science Technology Research Projects of Education Department of Henan Province (13A110767), and Basic and Frontier Technology Research Project of Henan Province (142300410318).

an affine linear manifold with the Cartesian product of second-order cones. In this paper we consider the SOCP in the standard format

$$\text{(P)} \qquad \min\{c^{\mathrm{T}}x:\ Ax = b,\ x \in \mathcal{K}\},$$

and the dual problem of (P) is given by

$$\text{(D)} \qquad \max\{b^{\mathrm{T}}y:\ A^{\mathrm{T}}y + s = c,\ s \in \mathcal{K}\},$$

where $A \in \mathcal{R}^{m\times n}, c \in \mathcal{R}^n$ and $b \in \mathcal{R}^m$, and $\mathcal{K} \subset \mathcal{R}^n$ is the Cartesian product of second-order cones, i.e., $\mathcal{K} = \mathcal{K}^{n_1} \times \ldots \times \mathcal{K}^{n_r}$, with $\mathcal{K}^{n_i} \subset \mathcal{R}^{n_i}$ for each $i = 1, \ldots, r$, and $n = \sum_{i=1}^{r} n_i$. In the subsequent analysis, we focus on the case $\mathcal{K} = \mathcal{K}^n$ for simplicity. Our analysis can be easily extended to general cases.

Throughout the paper, we make the following assumption.

Assumption 1.1. Both (P) and (D) are strictly feasible.

Under Assumption 1.1, it is well-known that both (P) and (D) have optimal solutions and their optimal values coincide [1], and the SOCP is equivalent to its *optimality conditions*:

$$Ax = b, \quad A^{\mathrm{T}}y + s = c, \quad x \circ s = 0, \quad x, s \in \mathcal{K},\ y \in \mathcal{R}^m, \tag{1.1}$$

where "$\circ$" denotes the Jordan product, which will be presented in the next section.

The SOCP problem includes the linear programming problem, the convex quadratic programming problem and the quadratically constrained convex quadratic programming problem as special cases [1]. In recent years, the SOCP has been studied extensively due to its various applications in many fields (see [1], [15]). Various methods have been developed for solving the SOCP, where the smoothing-type algorithms (e.g., [2], [3], [5], [6], [18], [19]) are one of the most effective methods. This kind of algorithms reformulates the system (1.1) as a family of parameterized smooth equations and solve the smooth equations approximately by using Newton's method at each iteration. By driving the parameter to converge to zero, one can expect to find a solution to the SOCP. In general, the smoothing-type algorithms for the SOCP are designed based on a monotone line search (e.g., [2], [3], [5], [6], [18], [19]). It is well known that the non-monotone line search rule has many advantages, especially in the case of iterates trapped in a narrow curved valley of objective functions (e.g., [8], [20]). Some non-monotone line search schemes have been applied to the smoothing-type algorithms for nonlinear complementarity problems (e.g., [9], [16]) and symmetric cone complementarity problems (e.g., [11]). A crucial question

in this respect is whether the smoothing-type algorithm with a non-monotone line search for the SOCP not only possesses locally fast convergence properties but also has encouraging numerical results.

In this paper, we propose a non-monotone smoothing-type algorithm for solving the SOCP. The proposed algorithm is based on a non-monotone line search scheme, which was first introduced by Zhang and Hager [20], and includes the usual monotone line search (e.g., see [2], [3], [5], [6], [18], [19]). By using the theory of Euclidean Jordan algebras, we prove the global and local quadratical convergence of the proposed algorithm under suitable assumptions. Some preliminary numerical results are also reported which demonstrate that the non-monotone smoothing-type algorithm has some advantage over the monotone one.

The paper is organized as follows. In the next section, we briefly introduce some preliminaries which will be used in the subsequent sections. In Section 3, we propose a non-monotone smoothing-type algorithm for solving the SOCP. The global convergence and local quadratic convergence of the proposed algorithm are investigated in Section 4. Preliminary numerical results are reported in Section 5. The conclusions are given in Section 6.

The following notations are used throughout this paper: $\mathcal{R}^n$ denotes the space of n-dimensional real column vectors, and $\mathcal{R}^n_+$ and $\mathcal{R}^n_{++}$ denote the non-negative and positive orthant in $\mathcal{R}^n$, respectively. For convenience, we write $(u^{\mathrm{T}}, v^{\mathrm{T}})^{\mathrm{T}}$ as (u, v) for any vectors $u, v \in \mathcal{R}^n$. The symbol I represents the identity matrix with suitable dimension and $\mathcal{J}$ denotes the set of all nonnegative integers, i.e., $\mathcal{J} := \{1, 2, \ldots\}$. By $\operatorname{int} \mathcal{K}$ we denote the interior of $\mathcal{K}$. For any $x, y \in \mathcal{R}^n$, we write $x \succeq_{\mathcal{K}} y$ if $x - y \in \mathcal{K}$, and $x \succ_{\mathcal{K}} y$ if $x - y \in \operatorname{int} \mathcal{K}$. The Euclidean inner product is denoted by $\langle \cdot, \cdot \rangle$. For any $\alpha, \beta > 0$, $\alpha = O(\beta)$ means that α/β is uniformly bounded as $\beta \to 0$, and $\alpha = o(\beta)$ means that α/β tends to zero as $\beta \to 0$.

2. Preliminaries

Smoothing-type algorithms for solving the SOCP are based on the Euclidean Jordan algebra associated with the SOC $\mathcal{K}$. For any vectors $x = (x_1, \tilde{x}) \in \mathcal{R} \times \mathcal{R}^{n-1}$ and $s = (s_1, \tilde{s}) \in \mathcal{R} \times \mathcal{R}^{n-1}$, their Jordan product associated with $\mathcal{K}$ is defined by

$$x \circ s := (x^{\mathrm{T}} s, x_1 \tilde{s} + s_1 \tilde{x}). \tag{2.1}$$

The identity element under this product is $\mathbf{e} := (1, 0, \ldots, 0)^{\mathrm{T}} \in \mathcal{R}^n$. For any $x = (x_1, \tilde{x}) \in \mathcal{R} \times \mathcal{R}^{n-1}$, we define the symmetric matrix

$$L_x := \begin{bmatrix} x_1 & \tilde{x}^{\mathrm{T}} \\ \tilde{x} & x_1 I \end{bmatrix} \in \mathcal{R}^{n \times n}, \tag{2.2}$$

where I represents the $(n-1)\times(n-1)$ identity matrix and L_x can be viewed as a linear mapping from $\mathcal{R}^n$ to $\mathcal{R}^n$ given by $L_x s = x \circ s$ for any $x, s \in \mathcal{R}^n$. It is easy to see that L_x is positive semidefinite if and only if $x \in \mathcal{K}$, and positive definite if and only if $x \in \operatorname{int}\mathcal{K}$.

For any $x = (x_1, \tilde{x}) \in \mathcal{R}\times\mathcal{R}^{n-1}$, its *spectral decomposition* with respect to the SOC $\mathcal{K}$ is

$$x = \lambda_1(x)c_1 + \lambda_2(x)c_2, \tag{2.3}$$

where $\lambda_1(x), \lambda_2(x)$ and c_1, c_2 are the spectral values and the associated spectral vectors of x given by

$$\lambda_i(x) = x_1 + (-1)^i\|\tilde{x}\|, \quad i = 1, 2, \tag{2.4}$$

$$c_i = \begin{cases} \dfrac{1}{2}\Big(1, (-1)^i\dfrac{\tilde{x}}{\|\tilde{x}\|}\Big), & \tilde{x} \neq 0, \\ \dfrac{1}{2}(1, (-1)^i\omega), & \tilde{x} = 0, \end{cases} \qquad i = 1, 2, \tag{2.5}$$

with any $\omega \in \mathcal{R}^{n-1}$ such that $\|\omega\| = 1$.

For any $x = (x_1, \tilde{x}) \in \mathcal{R}\times\mathcal{R}^{n-1}$, with spectral values $\lambda_1(x), \lambda_2(x)$ and spectral vectors c_1, c_2 given in (2.4) and (2.5), the following results hold (see Properties 2.1 and 2.2 in [17]):

(1) $x^2 := \lambda_1(x)^2c_1 + \lambda_2(x)^2c_2 \in \mathcal{K}$.

(2) If $x \in \mathcal{K}$, then $\lambda_2(x) \geqslant \lambda_1(x) \geqslant 0$, and $\sqrt{x} := \sqrt{\lambda_1(x)}c_1 + \sqrt{\lambda_2(x)}c_2$.

(3) If $x \in \operatorname{int}\mathcal{K}$, then $\lambda_2(x) \geqslant \lambda_1(x) > 0$, and $x^{-1} := \lambda_1(x)^{-1}c_1 + \lambda_2(x)^{-1}c_2$. Moreover, L_x is invertible with

$$L_x^{-1} = \frac{1}{\det(x)}\begin{bmatrix} x_1 & -\tilde{x}^{\mathrm{T}} \\ -\tilde{x} & \frac{\det(x)}{x_1}I + \frac{\tilde{x}\tilde{x}^{\mathrm{T}}}{x_1} \end{bmatrix},$$

where $\det(x) := x_1^2 - \|\tilde{x}\|^2$.

In general, we have $x^2 = x \circ x$. If $x \in \operatorname{int}\mathcal{K}$, then $x \circ x^{-1} = \mathbf{e}$.

3. The algorithm

The smoothing-type algorithms are typically developed by an SOC complementarity function [7]. Recall that a mapping $\varphi^{\mathrm{SOC}}\colon \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}^n$ is an SOC complementarity function associated with $\mathcal{K}$ if

$$\varphi^{\mathrm{SOC}}(x, s) = 0 \Longleftrightarrow x \circ s = 0,\ x \in \mathcal{K},\ s \in \mathcal{K}. \tag{3.1}$$

With an SOC complementarity function, one (e.g., [2], [3], [5], [6], [18], [19]) can rewrite (1.1) as:

$$G(x,y) := \begin{pmatrix} b - Ax \\ \varphi^{\mathrm{SOC}}(x, c - A^{\mathrm{T}}y) \end{pmatrix} = 0. \tag{3.2}$$

In this paper, we regard the well-known vector-valued natural residual (NR) function $\varphi_{\mathrm{NR}}(x,s)\colon\ \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}^n$ as the function φ^{SOC}, which is defined by

$$\varphi_{\mathrm{NR}}(x,s) := x + s - \sqrt{(x-s)^2}. \tag{3.3}$$

By using such a function, we rewrite (1.1) as

$$G_{\mathrm{NR}}(x,y,s) := \begin{pmatrix} b - Ax \\ c - A^{\mathrm{T}}y - s \\ \varphi_{\mathrm{NR}}(x,s) \end{pmatrix} = 0. \tag{3.4}$$

Notice that φ_{NR} is typically non-smooth because it is not differentiable at $(0,0) \in \mathcal{R}^n \times \mathcal{R}^n$. This implies that the function $G_{\mathrm{NR}}(x,y,s)$ is not differentiable at some points and the traditional Newton's methods cannot immediately apply to $G_{\mathrm{NR}}(x,y,s) = 0$. To overcome this difficulty, we use the following smoothing function $\varphi(\mu,x,s)\colon\ \mathcal{R}_+ \times \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}^n$:

$$\varphi(\mu,x,s) = (1+\mu)(x+s) - \sqrt{(1-\mu)^2(x-s)^2 + 4\mu^2\mathbf{e}}. \tag{3.5}$$

This function was first introduced by Huang, Han and Chen [10] for the nonlinear complementarity problem, and was extended to the SOCP by Chi and Liu [2]. Recently, Huang and Ni [12] have further studied its properties in the context of symmetric cone. Notice that the smoothing-type algorithms investigated in [2], [10], [12] are all designed with a monotone line search. In the following, we will study a non-monotone smoothing-type algorithm based on $\varphi(\mu,x,s)$.

Theorem 3.1 (Theorem 2.4, [2]). *Let $\varphi(\mu,x,s)$ be defined by* (3.5). *Then the following results hold.*

(i) *φ is globally Lipschitz continuous and strongly semi-smooth everywhere. Moreover, φ is continuously differentiable at any $(\mu,x,s) \in \mathcal{R}_{++} \times \mathcal{R}^n \times \mathcal{R}^n$ with its Jacobian*

$$\varphi'(\mu,x,s) = \begin{pmatrix} x + s - L_Q^{-1}[-(1-\mu)q^2 + 4\mu\mathbf{e}] \\ (1+\mu)I - (1-\mu)^2 L_Q^{-1} L_q \\ (1+\mu)I + (1-\mu)^2 L_Q^{-1} L_q \end{pmatrix}, \tag{3.6}$$

where

$$q := q(\mu, x, s) = x - s, \quad Q := Q(\mu, x, s) = \sqrt{(1-\mu)^2 q^2 + 4\mu^2 \mathbf{e}}. \tag{3.7}$$

(ii) $\lim_{\mu \downarrow 0} \varphi(\mu, x, s) = \varphi_{\rm NR}(x, s)$ *for any* $(x, s) \in \mathcal{R}^n \times \mathcal{R}^n$. *Thus,* $\varphi(\mu, x, s)$ *is a smoothing function of* $\varphi_{\rm NR}(x, s)$.

By using $\varphi(\mu, x, s)$ given in (3.5), we can also rewrite (1.1) as

$$H(z) = H(\mu, x, y, s) := \begin{pmatrix} \mu \\ b - Ax \\ c - A^{\rm T} y - s \\ \varphi(\mu, x, s) \end{pmatrix} = 0, \tag{3.8}$$

which is continuously differentiable in $\mathcal{R}_{++} \times \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n$. Thus, to solve (P) and (D), one can apply Newton-type methods to solve (3.8) and make $\mu \downarrow 0$.

Let $\Psi\colon\ \mathcal{R}_+ \times \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{R}_+$ be the natural merit function:

$$\Psi(z) := \|H(z)\|^2 \quad \forall\, z = (\mu, x, y, s) \in \mathcal{R}_+ \times \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n.$$

A l g o r i t h m 3.1 (A non-monotone smoothing-type algorithm for the SOCP).

Step 0: Choose constants $\delta \in (0,1), \sigma \in (0, 1/2)$ and $\mu_0 \in \mathcal{R}_{++}$, and let $\overline{z} := (\mu_0, 0, 0, 0) \in \mathcal{R}_{++} \times \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n$. Choose $\gamma \in (0,1)$ such that $\mu_0 \gamma < 1$. Let $(x^0, y^0, s^0) \in \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n$ be an arbitrary initial point. Let $z^0 := (\mu_0, x^0, y^0, s^0)$, $\Gamma_0 := \Psi(z^0)$ and $\beta(z^0) := \gamma \min\{1, \Psi(z^0)\}$. Let $Q_0 := 1$. Choose $\lambda_{\min}$ and $\lambda_{\max}$ such that $0 \leqslant \lambda_{\min} < \lambda_{\max} < 1$. Set $k := 0$.

Step 1: If $\|H(z^k)\| = 0$, then stop.

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

$$H(z^k) + H'(z^k)\Delta z^k = \beta(z^k)\overline{z}. \tag{3.9}$$

Step 3: Let α_k be the maximum of the values $1, \delta, \delta^2, \ldots$ such that

$$\Psi(z^k + \alpha_k \Delta z^k) \leqslant [1 - 2\sigma(1 - \mu_0\gamma)\alpha_k]\Gamma_k. \tag{3.10}$$

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

Step 5: Choose $\lambda_k \in [\lambda_{\min}, \lambda_{\max}]$. Set

$$Q_{k+1} := \lambda_k Q_k + 1, \tag{3.11}$$

$$\Gamma_{k+1} := \frac{\lambda_k Q_k \Gamma_k + \Psi(z^{k+1})}{Q_{k+1}}, \tag{3.12}$$

$$\beta(z^{k+1}) := \min\{\gamma, \gamma\Psi(z^{k+1}), \beta(z^k)\}. \tag{3.13}$$

Set $k := k + 1$. Go to Step 1.

A similar algorithmic framework has been extensively studied for solving the SOCP (see, e.g., [2], [3], [5], [6], [18], [19]). However, all papers mentioned above discussed this kind of algorithms based on a monotone line search. In Algorithm 3.1, the line search (3.10) is a non-monotone search scheme, which was first introduced by Zhang and Hager [20] for the unconstrained optimization problem. If we choose $\lambda_k = 0$ for all $k \in \mathcal{J}$, then the line search (3.10) is the usual monotone line search. It is easy to see that Γ_{k+1} is a convex combination of Γ_k and $\Psi(z^{k+1})$. Since $\Gamma_0 = \Psi(z^0)$, it follows that Γ_k is a convex combination of $\Psi(z^0), \Psi(z^1), \ldots, \Psi(z^k)$.

For the non-monotone line search given in Algorithm 3.1, some basic results are included in the following lemma, whose proof can be found in Remark 3.4, [11].

Lemma 3.1. *Suppose that the sequences $\{\Gamma_k\}$ and $\{z^k := (\mu_k, x^k, y^k, s^k)\}$ are generated by Algorithm* 3.1. *Then $\{\Gamma_k\}$ is monotonically decreasing. Moreover, $\Psi(z^k) \leqslant \Gamma_k$ for all $k \in \mathcal{J}$.*

In Algorithm 3.1, it is essential that the Jacobian matrix of $H(z)$ is invertible, since the descent direction should be well-defined and unique.

Lemma 3.2. *Let $H(z)$ be defined by* (3.8). *Then the following results hold.*

(i) *H is globally Lipschitz continuous, strongly semi-smooth and continuously differentiable at any $z := (\mu, x, y, s) \in \mathcal{R}_{++} \times \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n$ with its Jacobian*

$$H'(z) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -A & 0 & 0 \\ 0 & 0 & -A^{\mathrm{T}} & -I \\ \varphi'_\mu(\mu, x, s) & \varphi'_x(\mu, x, s) & 0 & \varphi'_s(\mu, x, s) \end{pmatrix}, \tag{3.14}$$

where $\varphi'_\mu(\mu, x, s)$, $\varphi'_x(\mu, x, s)$ and $\varphi'_s(\mu, x, s)$ are defined by (3.6).

(ii) *If A has full row rank, then $H'(z)$ is invertible for any $z := (\mu, x, y, s) \in \mathcal{R}_{++} \times \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n$.*

P r o o f. By Theorem 3.1, it is easy to see that (i) holds. The result (ii) can be proved in a similar way as the one in Theorem 5 of [14]. We omit it here. □

Theorem 3.2. *Suppose that A has full row rank. Then Algorithm* 3.1 *is well-defined.*

P r o o f. By the first equation of (3.9), we have $\Delta\mu_k = -\mu_k + \mu_0\beta(z^k)$, and thus

$$\mu_{k+1} = \mu_k + \alpha_k\Delta\mu_k = (1 - \alpha_k)\mu_k + \alpha_k\mu_0\beta(z^k) > 0, \tag{3.15}$$

which implies that $\mu_k > 0$ for all $k \geqslant 0$. Since A has full row rank, it follows from Lemma 3.2 that $H'(z^k)$ is non-singular for any $z^k = (\mu_k, x^k, y^k, s^k)$. Hence, Step 2 is well-defined at the kth iteration. For any $\alpha \in (0,1)$, we denote

$$F_k(\alpha) := \Psi(z^k + \alpha\Delta z^k) - \Psi(z^k) - \alpha\Psi'(z^k)\Delta z^k. \tag{3.16}$$

Then $F_k(\alpha) = o(\alpha)$, since Ψ is continuously differentiable for any $z^k \in \mathcal{R}_{++} \times \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{R}^n$. By the definition of $\beta(z^k)$, we have $\beta(z^k) \leqslant \gamma \min\{1, \Psi(z^k)\}$ for all $k \in \mathcal{J}$. Thus, $\beta(z^k) \leqslant \gamma\Psi(z^k) \leqslant \gamma\|H(z^k)\|$ if $\|H(z^k)\| \leqslant 1$, and $\beta(z^k) \leqslant \gamma \leqslant \gamma\|H(z^k)\|$ if $\|H(z^k)\| > 1$. This shows that

$$\beta(z^k) \leqslant \gamma\|H(z^k)\| \quad \forall\, k \in \mathcal{J}. \tag{3.17}$$

Therefore, it follows from (3.16) and (3.17) that for any $\alpha \in (0,1)$

$$\begin{aligned}
\Psi(z^k + \alpha\Delta z^k) &= \Psi(z^k) + \alpha\Psi'(z^k)\Delta z^k + F_k(\alpha) \\
&= \Psi(z^k) + 2\alpha H(z^k)^{\mathrm{T}} H'(z^k)\Delta z^k + F_k(\alpha) \\
&= \Psi(z^k) + 2\alpha H(z^k)^{\mathrm{T}}[-H(z^k) + \beta(z^k)\overline{z}] + F_k(\alpha) \\
&\leqslant (1 - 2\alpha)\Psi(z^k) + 2\alpha\mu_0\|H(z^k)\|\beta(z^k) + F_k(\alpha) \\
&\leqslant [1 - 2(1 - \mu_0\gamma)\alpha]\Psi(z^k) + o(\alpha).
\end{aligned}$$

Since $\mu_0\gamma < 1$, there exists a constant $\overline{\alpha} \in (0,1)$ such that for any $\alpha \in (0, \overline{\alpha}]$ and $\sigma \in (0, 1/2)$

$$\Psi(z^k + \alpha\Delta z^k) \leqslant [1 - 2\sigma(1 - \mu_0\gamma)\alpha]\Psi(z^k) \leqslant [1 - 2\sigma(1 - \mu_0\gamma)\alpha]\Gamma_k,$$

where the second inequality uses the second result in Lemma 3.1. This demonstrates that Step 3 is well-defined at the kth iteration. Therefore, Algorithm 3.1 is well-defined. □

4. Convergence analysis

In this section, it is proved that if an accumulation point z^* of the iteration sequence $\{z^k\}$ generated by Algorithm 3.1 satisfies the non-singularity assumption, then the iteration sequence converges to the accumulation point globally and locally quadratically without strict complementarity. To show the global convergence of Algorithm 3.1, we need the following lemma.

Lemma 4.1. *Suppose that A has full row rank and that $\{z^k := (\mu_k, x^k, y^k, s^k)\}$ is the iteration sequence generated by Algorithm* 3.1. *Then, $\beta(z^{k+1}) \leqslant \beta(z^k)$, $\mu_0\beta(z^k) \leqslant \mu_k$, $\mu_{k+1} \leqslant \mu_k$ for all $k \in \mathcal{J}$.*

P r o o f. From the definition of $\beta(z^k)$ given in (3.13), we know that $\beta(z^{k+1}) \leqslant \beta(z^k)$ for all $k \in \mathcal{J}$. From Step 0 in Algorithm 3.1, it is easy to see that $\mu_0\beta(z^0) \leqslant \mu_0\gamma \leqslant \mu_0$. Suppose that $\mu_0\beta(z^k) \leqslant \mu_k$ for some k. By (3.15) we have

$$\mu_{k+1} = \mu_k + \alpha_k\Delta\mu_k = (1-\alpha_k)\mu_k + \alpha_k\mu_0\beta(z^k), \tag{4.1}$$

which, together with $\mu_0\beta(z^k) \leqslant \mu_k$, yields that

$$\mu_{k+1} \geqslant (1-\alpha_k)\mu_0\beta(z^k) + \alpha_k\mu_0\beta(z^k) = \mu_0\beta(z^k) \geqslant \mu_0\beta(z^{k+1}),$$

where the last inequality holds, since $\{\beta(z^k)\}$ is monotonically decreasing. This shows that $\mu_0\beta(z^k) \leqslant \mu_k$ for all $k \in \mathcal{J}$. Moreover, it follows from (4.1) that for all $k \in \mathcal{J}$

$$\mu_{k+1} = (1-\alpha_k)\mu_k + \alpha_k\mu_0\beta(z^k) \leqslant (1-\alpha_k)\mu_k + \alpha_k\mu_k = \mu_k.$$

Thus, we complete the proof. □

Now we give the global convergence of Algorithm 3.1. The proof technique is taken from Theorem 4.3 of [11].

Theorem 4.1. *Suppose that A has full row rank and that $\{z^k\}$ is the iteration sequence generated by Algorithm* 3.1. *Then any accumulation point z^* of $\{z^k\}$ is a solution to $\Psi(z) = 0$.*

P r o o f. Without loss of generality, we assume that $z^k = (\mu_k, x^k, y^k, s^k)$ converges to $z^* = (\mu^*, x^*, y^*, s^*)$ as $k \to \infty$. From Lemma 4.1 we know that $\{\beta(z^k)\}$ is monotonically decreasing and bounded from below and thus convergent. Therefore, there exists $\beta^* \geqslant 0$ such that $\lim\limits_{k\to\infty} \beta(z^k) = \beta^*$. We now assume that $\beta^* > 0$ and derive a contradiction. By Lemma 3.1 we have

$$0 \leqslant \Psi(z^k) \leqslant \Gamma_k \leqslant \Gamma_{k-1} \leqslant \Gamma_0, \tag{4.2}$$

that is, the sequence $\{\Psi(z^k)\}$ is bounded. Hence, $\{\Psi(z^k)\}$ has a convergent subsequence, denoted by $\{\Psi(z^k)\}_{k\in\overline{\mathcal{J}}}$, where $\overline{\mathcal{J}} \subset \mathcal{J}$. Then, it follows from the continuity of Ψ that

$$\lim_{\overline{\mathcal{J}}\ni k\to\infty} \Psi(z^k) = \Psi(z^*).$$

Also $\lim\limits_{\overline{\mathcal{J}}\ni k\to\infty} \beta(z^k) = \beta^*$. Since $\beta^* > 0$, by the definition of $\beta(z^k)$, we have $\Psi(z^*) > 0$. On the other hand, since $\{\Gamma_k\}$ is monotonically decreasing and bounded from below

and thus convergent, there exists $\Gamma^* \geqslant 0$ such that $\varliminf_{\overline{\mathcal{J}} \ni k \to \infty} \Gamma(z^k) = \Gamma^*$. By (4.2), we have $\Gamma^* \geqslant \Psi(z^*) > 0$. Now, we consider the following cases.

▷ Suppose that $\alpha_k \geqslant \varepsilon > 0$ for all $k \in \overline{\mathcal{J}}$, where ε is a fixed constant. By the definition of Q_k and the fact that $\lambda_{\max} \in [0, 1)$, we have

$$Q_{k+1} = 1 + \sum_{i=0}^{k} \prod_{j=0}^{i} \lambda_{k-j} \leqslant 1 + \sum_{i=0}^{k} \lambda_{\max}^{i+1} \leqslant \sum_{i=0}^{\infty} \lambda_{\max}^{i} = \frac{1}{1 - \lambda_{\max}}$$

for any $k \in \overline{\mathcal{J}}$. Hence, by (3.12), we have for any $k \in \overline{\mathcal{J}}$

$$\begin{aligned} \Gamma_{k+1} &= \frac{\lambda_k Q_k \Gamma_k + \Psi(z^{k+1})}{Q_{k+1}} \\ &\leqslant \frac{\lambda_k Q_k \Gamma_k + [1 - 2\sigma(1 - \mu_0\gamma)\alpha_k]\Gamma_k}{Q_{k+1}} \\ &= \Gamma_k - \frac{2\sigma(1 - \mu_0\gamma)\alpha_k \Gamma_k}{Q_{k+1}} \\ &\leqslant [1 - 2\sigma(1 - \lambda_{\max})(1 - \mu_0\gamma)\varepsilon]\Gamma_k, \end{aligned} \tag{4.3}$$

where the first inequality follows from (3.10). Taking limits on both sides of the inequality (4.3), also using $\Gamma^* > 0$, we have $1 \leqslant 1 - 2\sigma(1 - \lambda_{\max})(1 - \mu_0\gamma)\varepsilon$, which contradicts $0 \leqslant \lambda_{\max} < 1$ and $0 < \mu_0\gamma < 1$.

▷ Suppose that $\varliminf_{\overline{\mathcal{J}} \ni k \to \infty} \alpha_k = 0$. Then, the step size $\hat{\alpha}_k := \alpha_k/\delta$ does not satisfy the line search criterion (3.10) for any sufficiently large $k \in \overline{\mathcal{J}}$, i.e.,

$$\Psi(z^k + \hat{\alpha}_k \Delta z^k) > [1 - 2\sigma(1 - \mu_0\gamma)\hat{\alpha}_k]\Gamma_k \geqslant [1 - 2\sigma(1 - \mu_0\gamma)\hat{\alpha}_k]\Psi(z^k). \tag{4.4}$$

Thus

$$\frac{\Psi(z^k + \hat{\alpha}_k \Delta z^k) - \Psi(z^k)}{\hat{\alpha}_k} > -2\sigma(1 - \mu_0\gamma)\Psi(z^k), \quad k \in \overline{\mathcal{J}}. \tag{4.5}$$

Since $\beta^* > 0$, by Lemma 4.1 we obtain that $\mu^* = \lim_{k \to \infty} \mu_k \geqslant \mu_0 \beta^* > 0$. It follows that $\Psi(z)$ is continuously differentiable at z^*. Moreover, from Lemma 3.2 we know that $H'(z^k)$ is an invertible continuously linear operator for all sufficiently large $k \in \overline{\mathcal{J}}$. Thus, by using (3.9) we obtain that $\{\Delta z^k\}_{k \in \overline{\mathcal{J}}}$ is convergent. Define $\Delta z^* := \varliminf_{\overline{\mathcal{J}} \ni k \to \infty} \Delta z^k$. Hence, taking limits on both sides of the inequality (4.5), we have

$$\begin{aligned} -2\sigma(1 - \mu_0\gamma)\Psi(z^*) &\leqslant 2H(z^*)^{\mathrm{T}} H'(z^*)\Delta z^* \\ &= 2H(z^*)^{\mathrm{T}}(-H(z^*) + \beta^* \overline{z}) \\ &\leqslant 2(-\Psi(z^*) + \mu_0 \|H(z^*)\|\beta^*) \\ &\leqslant -2(1 - \mu_0\gamma)\Psi(z^*), \end{aligned}$$

where the last inequality follows from the fact that $\beta^* \leqslant \gamma\|H(z^*)\|$ by (3.17). This indicates that $\sigma(1-\mu_0\gamma) \geqslant (1-\mu_0\gamma)$, which contradicts $\mu_0\gamma < 1$ and $0 < \sigma < 1/2$. Thus, $\beta^* = 0$. It follows that $\lim\limits_{k_n\to\infty} \beta(z^{k_n}) = 0$ for any subsequence $\{z^{k_n}\}$ of $\{z^k\}$. This, together with the definition of $\beta(z^k)$, implies that $\lim\limits_{k_n\to\infty} \Psi(z^{k_n}) = 0$ for any subsequence $\{z^{k_n}\}$ of $\{z^k\}$. By a simple continuity discussion we obtain that every accumulation point z^* of $\{z^k\}$ is a solution to $\Psi(z) = 0$. □

Let $\psi\colon\ \mathcal{R}_+ \times \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}^n$ be the CHKS smoothing function defined by

$$\psi(\mu, x, s) = x + s - \sqrt{(x-s)^2 + 4\mu^2\mathbf{e}}.$$

By using the same arguments as that in our paper, we can prove that Algorithm 3.1 based on ψ is also well-defined and has global and local quadratical convergence properties. Notice that Liu and Huang [13] recently proposed a smoothing Newton algorithm based on ψ for linear programming over symmetric cones. In [13], it was proved that the iteration sequence is bounded under Assumption 1.1. By using the method given in the proof of Theorem 4.1 in [13], we can obtain the following result.

Theorem 4.2. *Let Algorithm* 3.1 *be based on ψ. If we choose $x^0 \in \mathcal{K}$ such that $Ax^0 = b$, and choose $y^0 \in \mathcal{R}^m$ and set $s^0 := -A^{\mathrm{T}}y^0 + c$, then the iteration sequence $\{z^k\}$ generated by Algorithm* 3.1 *is bounded under Assumption* 1.1.

Now, we discuss the local quadratical convergence of Algorithm 3.1. By the definition of $\beta(z^k)$, we have $\beta(z^k) \leqslant \gamma\|H(z^k)\|^2$ for all z^k generated by Algorithm 3.1. Using this fact, the following convergence theorem can be proved in a similar way as in Theorem 4.3 of [2]. We omit the proof.

Theorem 4.3. *Suppose that A has full row rank and that z^* is an accumulation point of the iteration sequence $\{z^k\}$ generated by Algorithm* 3.1. *If all $V \in \partial H(z^*)$ are non-singular, where ∂H stands for the generalized Jacobian of H in the sense of Clarke* [4], *then $\{z^k\}$ converges to z^* quadratically, i.e., $\|z^{k+1} - z^*\| = O(\|z^k - z^*\|^2)$. Moreover, $\mu_{k+1} = O(\mu_k^2)$.*

5. Numerical results

In this section, we give some numerical results of Algorithm 3.1 for solving the SOCP. All experiments were performed on a personal computer with 1.96GB memory and Pentium(R) Dual-Core CPU 2.93 GHz×2. The operating system was Windows XP and the computer codes were all written in Matlab 7.0.1.

We consider the SOCP problem with sizes $n(=2m)$ from 100 to 600 with $n_i = 5$ for each $i = 1, \ldots, r$. The test problems are randomly generated. To be specific, we generate a random matrix A and a random vector x in the second-order cone which gives a right-hand side $b = Ax$ and hence, the problem is feasible. Moreover, we generate a random vector c in the second-order cone so that the optimal value of the problem is obtainable.

Throughout the computational experiments, the parameters used in Algorithm 3.1 are chosen as $\delta := 0.85$, $\sigma := 10^{-4}$, $\mu_0 := 0.1$, $\gamma := 0.2$, $y^0 := 0$, $s^0 := c$. Denote $\mathbf{e}^{n_i} := (1, 0, \ldots, 0)^{\mathrm{T}} \in \mathcal{R}^{n_i}$ and $\mathbf{e} := (\mathbf{e}^{n_1}, \ldots, \mathbf{e}^{n_r})^{\mathrm{T}}$. Let x^0 be chosen according to the values listed in Tables 1 and 2. We use $\|H(z^k)\| < 10^{-6}$ as the stopping criterion.

In our computation, in order to see the behavior of Algorithm 3.1 with a monotone line search, we set $\lambda_k = 0$ for all $k \in \mathcal{J}$, and in order to see the behavior of Algorithm 3.1 with a non-monotone line search, we set $\lambda_k = 0.2$ for all $k \in \mathcal{J}$. The random problems of each case are generated 10 times, and the tested results are listed in Table 1, where AIT and ACPU denote the average values of the number of iterations and the CPU time in seconds, respectively.

			$\lambda_k = 0$		$\lambda_k = 0.2$	
x^0	m	n	AIT	ACPU	AIT	ACPU
$\mathbf{e}$	50	100	8.0	0.066	8.1	0.066
	100	200	9.1	0.209	9.1	0.200
	150	300	9.7	0.594	9.5	0.597
	200	400	11.1	1.461	10.4	1.388
	250	500	10.8	3.458	10.2	2.458
	300	600	11.1	5.172	10.4	4.609
$0.5\mathbf{e}$	50	100	8.2	0.066	8.3	0.069
	100	200	9.1	0.208	9.1	0.206
	150	300	9.6	0.625	9.3	0.597
	200	400	10.6	1.492	10.1	1.373
	250	500	11.1	3.023	10.1	2.456
	300	600	11.1	5.216	10.4	4.592
$0.2\mathbf{e}$	50	100	8.2	0.066	8.3	0.066
	100	200	9.0	0.208	9.0	0.219
	150	300	9.7	0.625	9.3	0.588
	200	400	10.6	1.420	10.0	1.323
	250	500	11.0	3.052	10.1	2.547
	300	600	10.9	5.220	10.4	4.786

Table 1. Numerical results for Algorithm 3.1

Notice that by using Zhang-Hager's non-monotone scheme [20], Huang, Hu, and Han [11] proposed a non-monotone smoothing-type algorithm to solve symmetric

cone complementarity problems. For comparison purpose, we also use their algorithmic framework to solve the above test problem. In the experiments, we choose parameters and initial points in a way similar to the previous experiment and choose $\lambda_k = 0.2$. In addition, we choose $\beta = (\|H(z^0)\| + 1)/\mu_0$ to insure that the condition $\|H(z^0)\| \leqslant \beta\mu_0$ holds. The numerical results are listed in Table 2.

x^0	m	n	AIT	ACPU
e	50	100	11.0	0.106
	100	200	10.9	0.281
	150	300	10.5	1.038
	200	400	10.8	2.478
	250	500	10.8	4.720
	300	600	10.9	7.120
0.5**e**	50	100	10.9	0.097
	100	200	10.8	0.281
	150	300	10.6	1.702
	200	400	10.9	2.514
	250	500	10.7	4.672
	300	600	10.9	7.338
0.2**e**	50	100	10.9	0.086
	100	200	10.7	0.277
	150	300	10.6	1.070
	200	400	10.9	2.509
	250	500	10.7	4.680
	300	600	10.8	7.592

Table 2. Numerical results for Huang-Hu-Han's algorithm

From the numerical results listed in Table 1, we may see that Algorithm 3.1 works better with the non-monotone line search than the monotone line search in the sense that the former could find the optimizer more efficiently than the latter; and the former has less number of iterations and less CPU time than the latter for most cases. Moreover, by the numerical results in Tables 1 and 2, we may find that our non-monotone smoothing algorithm seems to be more effective than Huang-Hu-Han's algorithm [11]. These demonstrate that the non-monotone smoothing-type algorithm proposed in this paper has some advantages.

6. Conclusions

In this paper, we propose a non-monotone smoothing-type algorithm to solve the SOCP. By using the theory of Euclidean Jordan algebras, we prove that the proposed algorithm is globally and locally quadratically convergent under some suitable assumptions. The numerical results show that our algorithm performs well.

A c k n o w l e d g e m e n t . The authors thank the anonymous referees for their valuable comments and suggestions on the paper, which have considerably improved the paper. Especially, we sincerely thank Dr. Tie Ni for providing us with his codes.

References

[1] *F. Alizadeh, D. Goldfarb*: Second-order cone programming. Math. Program. *95* (2003), 3–51.
[2] *X. Chi, S. Liu*: A non-interior continuation method for second-order cone programming. Optimization *58* (2009), 965–979.
[3] *X. Chi, S. Liu*: A one-step smoothing Newton method for second-order cone programming. J. Comput. Appl. Math. *223* (2009), 114–123.
[4] *F. H. Clarke*: Optimization and Nonsmooth Analysis. Canadian Mathematical Society Series of Monographs and Advanced Texts. John Wiley & Sons, New York, 1983; reprinted by SIAM, Philadelphia, 1990.
[5] *L. Fang, Z. Feng*: A smoothing Newton-type method for second-order cone programming problems based on a new smoothing Fischer-Burmeister function. Comput. Appl. Math. *30* (2011), 569–588.
[6] *L. Fang, G. He, Y. Hu*: A new smoothing Newton-type method for second-order cone programming problems. Appl. Math. Comput. *215* (2009), 1020–1029.
[7] *M. Fukushima, Z.-Q. Luo, P. Tseng*: Smoothing functions for second-order-cone complementarity problems. SIAM J. Optim. *12* (2002), 436–460.
[8] *L. Grippo, F. Lampariello, S. Lucidi*: A nonmonotone line search technique for Newton's method. SIAM J. Numer. Anal. *23* (1986), 707–716.
[9] *S.-L. Hu, Z.-H. Huang, P. Wang*: A nonmonotone smoothing Newton algorithm for solving nonlinear complementarity problems. Optim. Methods Softw. *24* (2009), 447–460.
[10] *Z. H. Huang, J. Han, Z. Chen*: Predictor-corrector smoothing Newton method, based on a new smoothing function, for solving the nonlinear complementarity problem with a P_0-function. J. Optimization Theory Appl. *117* (2003), 39–68.
[11] *Z. H. Huang, S. L. Hu, J. Han*: Convergence of a smoothing algorithm for symmetric cone complementarity problems with a nonmonotone line search. Sci. China, Ser. A *52* (2009), 833–848.
[12] *Z.-H. Huang, T. Ni*: Smoothing algorithms for complementarity problems over symmetric cones. Comput. Optim. Appl. *45* (2010), 557–579.
[13] *X.-H. Liu, 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.
[14] *Y.-J. Liu, L.-W. Zhang, Y.-H. Wang*: Analysis of a smoothing method for symmetric conic linear programming. J. Appl. Math. Comput. *22* (2006), 133–148.
[15] *M. S. Lobo, L. Vandenberghe, S. Boyd, H. Lebret*: Applications of second-order cone programming. Linear Algebra Appl. *284* (1998), 193–228.

[16] *T. Ni, P. Wang*: A smoothing-type algorithm for solving nonlinear complementarity problems with a non-monotone line search. Appl. Math. Comput. *216* (2010), 2207–2214.
[17] *S. Pan, J.-S. Chen*: A damped Gauss-Newton method for the second-order cone complementarity problem. Appl. Math. Optim. *59* (2009), 293–318.
[18] *J. Tang, G. He, L. Dong, L. Fang*: A smoothing Newton method for second-order cone optimization based on a new smoothing function. Appl. Math. Comput. *218* (2011), 1317–1329.
[19] *J. Tang, G. He, L. Dong, L. Fang*: A new one-step smoothing Newton method for second-order cone programming. Appl. Math., Praha *57* (2012), 311–331.
[20] *H. Zhang, W. W. Hager*: A nonmonotone line search technique and its application to unconstrained optimization. SIAM J. Optim. *14* (2004), 1043–1056.

Authors' addresses: *Jingyong Tang*, *Li Dong*, College of Mathematics and Information Science, Xinyang Normal University, Xinyang 464000, China, e-mail: `tangjingyong926@163.com`, `citycity926@163.com`; *Liang Fang*, College of Mathematics and System Science, Taishan University, Tai'an 271021, China, e-mail: `fangliang3@163.com`; *Li Sun*, College of Information Science and Engineering, Shandong Agricultural University, Tai'an 271018, China, e-mail: `sunlishi@hotmail.com`.