Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 4607853, 12 pages
https://doi.org/10.1155/2018/4607853

Research Article

A Projection Neural Network for Circular Cone Programming

Yaling Zhang [1,2] **and Hongwei Liu** [1]

[1] *School of Mathematics and Statistics, Xidian University, Xi'an 710071, China*
[2] *School of Computer Science, Xi'an Science and Technology University, Xi'an 710054, China*

Correspondence should be addressed to Yaling Zhang; zyldella@126.com

Received 5 January 2018; Revised 24 April 2018; Accepted 3 May 2018; Published 10 June 2018

Academic Editor: Nibaldo Rodríguez

A projection neural network method for circular cone programming is proposed. In the KKT condition for the circular cone programming, the complementary slack equation is transformed into an equivalent projection equation. The energy function is constructed by the distance function and the dynamic differential equation is given by the descent direction of the energy function. Since the projection on the circular cone is simple and costs less computation time, the proposed neural network requires less state variables and leads to low complexity. We prove that the proposed neural network is stable in the sense of Lyapunov and globally convergent. The simulation experiments show our method is efficient and effective.

1. Introduction

The circular cone is a pointed, closed, convex cone having hyperspherical sections orthogonal to its axis of revolution about which the cone is invariant to rotation [1–3]. Let its half-aperture angle be $\theta_i \in (0, \pi/2)$, $i = 1, 2, \cdots, N$, and then the n_i-dimensional circular cone denoted by L_{θ_i} can be expressed as follows:

$$L_{\theta_i} = \left\{ x_i = \begin{bmatrix} x_{i1} \\ x_{i0} \end{bmatrix} \in R^{n_i - 1} \times R : \|x_{i1}\| \le \tan\theta_i x_{i0} \right\}, \quad (1)$$

where $\|\cdot\|$ is the standard Euclidean norm.

When $\theta_i = \pi/4$, the circular cone reduces to the well-known second-order cone given by

$$K^{n_i} = \left\{ x_i = \begin{bmatrix} x_{i1} \\ x_{i0} \end{bmatrix} \in R^{n_i - 1} \times R : \|x_{i1}\| \le x_{i0} \right\}. \quad (2)$$

In this paper, we consider a linear circular cone programming (LCCP), which is described as follows:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \ x \in L_\theta, \end{aligned} \quad (3)$$

where $A \in R^{m \times n}$, $c \in R^n$, $b \in R^m$ and $x = [x_1, \cdots, x_N] \in R^{n_1} \times \cdots \times R^{n_N}$, and $\sum_{i=0}^{N} n_i = n$. Here, $x = [x_1, \cdots, x_N] \in R^{n_1} \times \cdots \times R^{n_N}$ is viewed as a column vector in $R^{n_1 + \cdots + n_N}$. In addition,

$$L_\theta = L_{\theta_1} \times L_{\theta_2} \times \cdots \times L_{\theta_N}, \quad (4)$$

and $x_i \in L_{\theta_i}$. When $\theta_i = \pi/4$, $i = 1, 2, \cdots, N$, we have

$$K = K^{n_1} \times K^{n_2} \times \cdots \times K^{n_N}. \quad (5)$$

The circular cone programming arises in some real-life engineering problems, such as the optimal grasping manipulation problems for multifingered robots [4, 5]. In addition, circular cone programming is applied in the perturbation analysis of second-order cone programming problems [6].

By the Lagrangian method, the dual programming to LCCP can be given as

$$\begin{aligned} \min \quad & b^T y \\ \text{s.t.} \quad & A^T y + z = c, \\ & z \in L_\theta^*, \end{aligned} \quad (6)$$

where $y \in R^n$ is the Lagrange multipliers, and L_θ^* is the dual cone of cone L_θ, which is defined as

$$L_\theta^* = L_{\theta_1}^* \times L_{\theta_2}^* \times \cdots \times L_{\theta_N}^*, \tag{7}$$

where

$$L_{\theta_i}^* := \left\{ z_i \mid z_i^T x_i \geq 0,\ x_i \in L_{\theta_i} \right\}. \tag{8}$$

Under mild constraint qualifications (e.g., Slater condition), strong duality holds for problem (3). Then, by the strong duality theorem for general conic programming problems [7], the KKT condition for (3) is given as

$$\begin{aligned} Ax &= b, \\ A^T y + z &= c, \\ x^T z &= 0, \quad x \in L_\theta,\ z \in L_\theta^*. \end{aligned} \tag{9}$$

Circular cone programming is a nonlinear convex programming, and the second-order cone programming is a special case [8, 9]. There are many efficient methods to deal with the second-order cone programming [8, 10, 11]. However, different from the second-order cone, the circular cone L_θ is nonsymmetric. Some properties holding in the second-order cone can be extended to the circular cone [12]. However, some other second-order cone properties fail to be satisfied for the circular cone [3, 12].

Recently, an interior point method [13] is proposed for the circular cone programming based on self-concordant barrier functions for its cones. However, in many science and engineering applications, real-time solutions of circular cone problems are often desired. For the large-scale problems, the traditional algorithm may not be efficient due to the complexity of the algorithm used. The artificial neural network based circuit implementation is an efficient approach for the real-time problem. At present, two types of neural networks are developed for the second-order cone programming and have shown some computational advantages. One type is the smooth neural network, including the merit function method derived from the Fischer-Burmeister function [14], the smoothed natural residual merit function method [15], and the merit function method based on the generalized Fischer-Burmeister function [16]. The other is the projection neural network by replacing the scalar projection function with the cone projection function [14, 17]. In paper [18], professor He proposed a neural network based on the simple projection and contraction technique for linear asymmetric variational inequalities. Inspired by the ideal projection and contraction technique and the new results about the projection conclusions on the circular cone [2], we can develop the projection neural network for the linear circular cone programming.

In this paper, we focus on neural network approach to the circular cone programming problem. The energy function is constructed by the distance function based on a cone projection function, whose solutions correspond to the KKT points of the circular cone programming. The dynamic differential equation is given by the descent direction of the energy function based on the projection and contraction technique. The proposed neural network requires less state variables and leads to low complexity. Its Lyapunov stability and global convergence are proved under some conditions. Finally, we test the projection neural network by some numerical examples and the optimal grasping manipulation problems for multifingered robots and also compare the neural network with the second neural network for some second-order cone programming problems in paper [14]. Simulation results demonstrate the effectiveness of the proposed neural network.

2. Preliminaries

In this section, firstly, we introduced some concepts about the Lyapunov stability of the first-order differential equation [16, 19]. Then, we briefly introduce some properties of circular cone and the projection on the circular cone, which are proposed in paper [2].

2.1. Lyapunov Stability of the First-Order Differential Equation. Given a mapping $f : R^n \to R^n$, the following first-order differential equation is

$$\begin{aligned} \frac{du}{dt} &= f(u(t)), \\ u(t_0) &= u_0 \in R^n. \end{aligned} \tag{10}$$

Next, we give the definition of Lyapunov stability [19].

Definition 1 ([19]).

(1) For (10), a point $u^* \in R^n$ is called an equilibrium point of (10) if $f(u^*) = 0$.

(2) If there is a neighborhood $N \subseteq R^n$ of u^* such that $f(u^*) = 0$ and $f(u) \neq 0$ for $u \in N$, then u^* is called an isolated equilibrium point.

Definition 2 ([19]). Let $u(t)$ be a solution of (10). For an isolated equilibrium point $u^* \in R^n$, if, for any $u(t_0) = u_0$ and $\epsilon > 0$, there exists a $\delta > 0$ such that

$$\|u_0 - u^*\| < \delta \Longrightarrow \|u(t) - u^*\| < \epsilon \quad \text{for } t \geq t_0, \tag{11}$$

then u^* is Lyapunov stable.

Definition 3 (Lyapunov function [19]). Let $N \subseteq R^n$ be an open neighborhood of $\widetilde{u}$. A continuously differential function $g : R^n \to R$ is said to be a Lyapunov function (or energy function) at the state $\widetilde{u}$ for (10) if

$$\begin{aligned} g(\widetilde{u}) &= 0, \\ g(\widetilde{u}) &> 0, \quad \forall u \in N \setminus \widetilde{u}; \\ \frac{dg(u(t))}{dt} &\leq 0, \quad \forall u \in N. \end{aligned} \tag{12}$$

The relationship between stabilities and a Lyapunov function is given as follows, which is proposed in paper [20].

Lemma 4 ([20]). *An isolated equilibrium point u^* is Lyapunov stable if there exists a Lyapunov function over some neighborhood N of u^*.*

2.2. The Properties and Projection on the Circular Cone. Let

$$T_i = \begin{bmatrix} I_{n_i-1} & 0 \\ 0 & \tan\theta_i \end{bmatrix}, \tag{13}$$

for $i = 1, 2, \cdots, N$, where I_{n_i-1} is the $n_i - 1$ dimension unit matrix. The following lemma describes the relationship between the second-order cone K^{n_i} and the circular cone L_{θ_i}, where $i = 1, 2, \cdots, N$.

Lemma 5 ([2]). *Let L_{θ_i} and K^{n_i} be defined as in (1) and (2) for $i = 1, 2, \cdots, N$. Then we obtain*

(1) $L_{\theta_i} = T_i^{-1} K^{n_i}$ *and* $K^{n_i} = T_i L_{\theta_i}$,

(2) $T_i K^{n_i} = L_{\pi/2-\theta_i}$ *and* $L_{\pi/2-\theta_i} = T_i^2 L_{\theta_i}$,

(3) $L_{\theta_i}^* = L_{\pi/2-\theta_i}$ *and* $(L_{\theta_i}^*)^* = L_{\theta_i}$.

From Lemma 5 and the definition of L_{θ_i}, $L_{\theta_i}^*$, $i = 1, 2, \cdots, N$, we have

$$x_i \in L_{\theta_i} \Longleftrightarrow T_i x_i \in K^{n_i}, \quad \text{for } x_i = \begin{bmatrix} x_{i_1} \\ x_{i_0} \end{bmatrix} \in R^{n_i-1} \times R \tag{14}$$

and

$$z_i \in L_{\theta_i}^* \Longleftrightarrow T_i^{-1} z_i \in K^{n_i}, \quad \text{for } z_i = \begin{bmatrix} z_{i_1} \\ z_{i_0} \end{bmatrix} \in R^{n_i-1} \times R. \tag{15}$$

Let

$$T = \begin{bmatrix} T_1 & & 0 \\ & \ddots & \\ 0 & & T_N \end{bmatrix}. \tag{16}$$

Then, from the conclusion above, we have

$$x \in L_\theta \Longleftrightarrow Tx \in K, \quad \text{for } x = [x_1, \cdots, x_N] \in R^{n_1} \times \cdots \times R^{n_N} \tag{17}$$

and

$$x \in L_\theta^* \Longleftrightarrow T^{-1} x \in K, \quad \text{for } z = [z_1, \cdots, z_N] \in R^{n_1} \times \cdots \times R^{n_N}. \tag{18}$$

Let $x_i = \begin{bmatrix} x_{i_1} \\ x_{i_0} \end{bmatrix} \in R^{n_i-1} \times R$, $i = 1, 2, \cdots, N$. Then the spectral decomposition of x_i on circular cone L_{θ_i} can be given as [2]

$$x_i = \lambda_1(x_i)\, c_1(x_i) + \lambda_2(x_i)\, c_2(x_i), \tag{19}$$

where

$$\begin{aligned} \lambda_1(x_i) &= x_{i_0} - \left\| x_{i_1} \right\| \cot\theta_i, \\ \lambda_2(x_i) &= x_{i_0} + \left\| x_{i_1} \right\| \tan\theta_i \end{aligned} \tag{20}$$

and

$$\begin{aligned} c_1(x_i) &= \frac{1}{1+\cot^2\theta_i} \begin{bmatrix} \cot\theta_i I_{n_i-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -w \\ 1 \end{bmatrix}, \\ c_2(x_i) &= \frac{1}{1+\tan^2\theta_i} \begin{bmatrix} \tan\theta_i I_{n_i-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} w \\ 1 \end{bmatrix} \end{aligned} \tag{21}$$

with $w = -x_{i_1}/\|x_{i_1}\|_2$ if $x_{i_1} \neq 0$ and any vector in R^{n_i-1} satisfying $\|w\| = 1$ if $x_{i_1} = 0$.

Next we introduce the projection on the circular cone [2].

Lemma 6 ([2]). *For any $x_i = \begin{bmatrix} x_{i_1} \\ x_{i_0} \end{bmatrix} \in R^{n_i-1} \times R$, $i = 1, 2, \cdots, N$, let $P_{L_{\theta_i}}(x_i)$ be the projection of x_i on the circular cone L_{θ_i}. Then we obtain*

$$P_{L_{\theta_i}}(x_i) = (\lambda_1(x_i))_+ c_1(x_i) + (\lambda_2(x_i))_+ c_2(x_i), \tag{22}$$

where $s_+ := \max(0, s)$ for $s \in R$.

Let $x = [x_1, \cdots, x_N] \in R^{n_1} \times \cdots R^{n_N}$. Then the projection $P_{L_\theta}(x)$ of x on cone L_θ is described as

$$P_{L_\theta}(x) = \left[P_{L_{\theta_1}}(x_1), \cdots, P_{L_{\theta_N}}(x_N) \right] \in R^{n_1} \times \cdots \times R^{n_N}. \tag{23}$$

In particular, when $\theta_i = \pi/4$, the projection on second-order cone K^{n_i} can be obtained, which have been proposed in papers [21–23]:

$$P_{K^{n_i}}(x_i) = (x_{i0} - \|x_{i1}\|)_+ \frac{1}{2} \begin{bmatrix} -w \\ 1 \end{bmatrix} + (x_{i0} + \|x_{i1}\|)_+ \frac{1}{2} \begin{bmatrix} w \\ 1 \end{bmatrix}, \tag{24}$$

where $w = -x_1/\|x_{i_1}\|$ when $x_{i_1} \neq 0$ and any vector in R^{n_i-1} satisfying $\|w\| = 1$ when $x_{i_1} = 0$. Then the projection $P_K(x)$ of x on the cone K is described as

$$P_K(x) = \left[P_{K^{n_1}}(x_1), \cdots, P_{K^{n_N}}(x_N) \right] \in R^{n_1} \times \cdots \times R^{n_N}. \tag{25}$$

It is well known that any $z \in R^n$ can be written as

$$z = P_\Theta(z) + P_{-\Theta^*}(z) \tag{26}$$

where Θ denotes any closed convex cone and Θ^* represents the dual cone of Θ. Hence, when $\Theta = L_\theta$ and $\Theta = K$, respectively, we have

$$z = P_{L_\theta}(z) + P_{-L_\theta^*}(z) = P_{L_\theta}(z) - P_{-L_\theta^*}(-z) \tag{27}$$

and

$$z = P_K(z) + P_{-K}(z) = P_K(z) - P_{(K)^*}(-z). \quad (28)$$

Since the second-order cone is self-dual, then $K = K^*$. We have

$$z = P_K(z) + P_{-K}(z) = P_K(z) - P_{-K}(-z). \quad (29)$$

3. A Projection-Based Neural Network Model for Circular Cone Programming

In this section, we build a projection-based neural network model for circular cone programming.

Firstly, an equivalent projection equation is built for the KKT condition (9). Since the circular cone is a non-self-dual cone, our analysis is based on the relationship of circular cone and second-order cone.

Lemma 7. *Assume $\overline{x} \in L_\theta$, $\overline{z} \in L_\theta^*$. Then $\overline{x}^T\overline{z} = 0$ if and only if $e(\overline{x},\overline{z}) = 0$, where $e(\overline{x},\overline{z}) = T\overline{x} - P_K(T\overline{x} - T^{-1}\overline{z})$.*

Proof. "⇒" The projection on the closed convex set K has an important property [24],

$$\langle x - P_K(x), T\overline{x} - P_K(x)\rangle \le 0, \quad \forall\, x \in R^n \quad (30)$$

where $\langle\cdot\rangle$ denotes the inner product of two vectors.

Let $x = T\overline{x} - T^{-1}\overline{z}$ in the inequality above. Then we have

$$0 \le \|e(\overline{x},\overline{z})\|^2 \le \langle e(\overline{x},\overline{z}), Tz\rangle. \quad (31)$$

Moreover, because

$$\begin{aligned}\left\langle e(\overline{x},\overline{z}), T^{-1}\overline{z}\right\rangle &= \left\langle T\overline{x} - P_K\left(T\overline{x} - T^{-1}\overline{z}\right), T^{-1}\overline{z}\right\rangle \\ &= \langle\overline{x},\overline{z}\rangle - \left\langle P_K\left(T\overline{x} - T^{-1}\overline{z}\right), T^{-1}\overline{z}\right\rangle\end{aligned} \quad (32)$$

and

$$\begin{aligned}\langle\overline{x},\overline{z}\rangle &= 0, \\ \left\langle P_K\left(T\overline{x} - T^{-1}\overline{z}\right), \overline{z}\right\rangle &\ge 0,\end{aligned} \quad (33)$$

we have

$$\langle e(\overline{x},\overline{z}), T\overline{z}\rangle \le 0. \quad (34)$$

By (31) and (34), $e(\overline{x},\overline{z}) = 0$ is obtained.

"⇐" It follows from $e(\overline{x},\overline{z}) = 0$ that

$$T\overline{x} = P_K\left(T\overline{x} - T^{-1}\overline{z}\right) \in K. \quad (35)$$

From (17), we have $\overline{x} \in L_\theta$.

Based on (29), we have

$$T\overline{x} - T^{-1}\overline{z} = P_K\left(T\overline{x} - T^{-1}\overline{z}\right) + P_{-K}\left(T\overline{x} - T^{-1}\overline{z}\right). \quad (36)$$

Then, we get

$$\begin{aligned}T^{-1}\overline{z} &= -P_{-K}\left(T\overline{x} - T^{-1}\overline{z}\right) = P_K\left(-\left(T\overline{x} - T^{-1}\overline{z}\right)\right) \\ &\in K.\end{aligned} \quad (37)$$

From (18), it is easy to obtain $\overline{z} \in L_\theta^*$. For any $Tx \in K$, we have

$$\begin{aligned}&\left\langle T\overline{x} - T^{-1}\overline{z} - P_K\left(T\overline{x} - T^{-1}\overline{z}\right), Tx\right. \\ &\left. - P_K\left(T\overline{x} - T^{-1}\overline{z}\right)\right\rangle \le 0.\end{aligned} \quad (38)$$

Since $T\overline{x} = P_K(T\overline{x} - T^{-1}\overline{z})$, we get

$$\left\langle Tx - T\overline{x}, T^{-1}\overline{z}\right\rangle \ge 0. \quad (39)$$

Substituting $Tx = 0 \in K$ and $Tx = 2T\overline{x} \in K$ in inequality (39), we have

$$\langle\overline{x},\overline{z}\rangle = \overline{x}^T\overline{z} = 0. \quad (40)$$

The proof is completed. □

Let

$$e(u) \coloneqq Wu - P_\Omega\left(Wu - W^{-1}(Mu + q)\right), \quad (41)$$

where $P_\Omega(\cdot)$ denotes the projection on the set $\Omega = K \times R^m$ and

$$\begin{aligned}u &= \begin{bmatrix} x \\ y \end{bmatrix}, \\ W &= \begin{bmatrix} T & 0 \\ 0 & I_m \end{bmatrix}, \\ M &= \begin{bmatrix} 0 & -A^T \\ A & 0 \end{bmatrix}, \\ q &= \begin{bmatrix} c \\ -b \end{bmatrix}.\end{aligned} \quad (42)$$

Obviously, M is a block asymmetric matrix; i.e., $M^T = -M$. So, we have

$$e(u) = \begin{bmatrix} Tx - P_K\left(Tx - T^{-1}\left(c - A^T y\right)\right) \\ Ax - b \end{bmatrix}. \quad (43)$$

From Lemma 7, we know that the KKT condition (9) is equivalent to

$$e(u) = 0. \quad (44)$$

Let $\Omega_\theta = L_\theta \times R^m$. Then it is easy to prove that

$$u \in \Omega_\theta \Longleftrightarrow Wu \in \Omega. \quad (45)$$

Next, we give the following neural network model for circular cone programming, which consists of the following energy function and dynamical system.

The energy function is given as follows:

$$H(u) = \frac{1}{2}\|u - u^*\|^2, \quad (46)$$

where $u^* \in \Omega_\theta^*$.

Here, a dynamical system is proposed to solve (44). The dynamical system is given as follows:

$$\frac{du}{dt} = F(u) = -r\rho(u)\left(W - W^{-1}M\right)e(u), \tag{47}$$

where $r \in \{0, 1\}$ and

$$\rho(u) = \frac{\|e(u)\|^2}{\left\|\left(W - W^{-1}M\right)e(u)\right\|^2}. \tag{48}$$

The system described by (47) can be realized by a recurrent neural network with two-layer structure.

4. Stability Analysis

In this section, the stability analysis of projection neural network for circular cone programming is given.

For the dynamical system (47), we have the following result.

Lemma 8. *$u^* \in \Omega_\theta^*$ if and only if u^* is an equilibrium point of network (47), where Ω_θ^* is the solution set of problem (9).*

Proof. "$\Rightarrow$" Since $u^* \in \Omega_\theta^*$, we have $e(u^*) = 0$. It is easy to know that

$$\left.\frac{du}{dt}\right|_{u=u^*} = 0, \tag{49}$$

so u^* is an equilibrium point of network (47).

"$\Leftarrow$" Since u^* is an equilibrium point of network (47), we obtain $F(u^*) = -r\rho(u)(W - W^{-1}M)e(u^*) = 0$.

Consider

$$\begin{aligned}\left\|F(u^*)\right\|_2^2 &= r^2\rho(u^*)^2\left[e(u^*)\right]^T\left(W - W^{-1}M\right)^T \\ &\cdot\left(W - W^{-1}M\right)e(u^*) = 0.\end{aligned} \tag{50}$$

Since $(W - W^{-1}M)^T(W - W^{-1}M)$ is positive definite matrix, $e(u^*) = 0$ is obtained. From (45), we know $u^* \in \Omega_\theta^*$. □

The following result guarantees the existence and uniqueness for the solution $u(t)$ of neural network (47).

Lemma 9. *For any $u_0 \in \Omega$, there exists a unique continuous solution $u(t)$ of (47) with $u(0) = u_0$ for all $t \geq 0$.*

Proof. Because

$$e(u) = \begin{bmatrix} Tx - P_K\left(Tx - T^{-1}\left(c - A^T y\right)\right) \\ Ax - b \end{bmatrix}, \tag{51}$$

from the results in paper [11], we know that $e(u)$ is semismooth. From all of the above, we conclude that $F(u)$ is semismooth. Thus, there exists a unique solution $u(t)$ for neural network (47). □

The results of Lemmas 8 and 9 indicate that neural network model (47) is well defined. Now, we give the Lyapunov stability of neural network (47).

Theorem 10. *The solution of neural network (47), with initial point u_0, is Lyapunov stable.*

Proof. From Lemma 9, there exists a unique continuous solution $u(t)$ of (47) with $u(0) = u_0$ for all $t \geq 0$. Since $u^* \in \Omega_\theta^*$, we have

$$\left\langle u^*, Mu^* + q\right\rangle = \left\langle Wu^*, W^{-1}\left(Mu^* + q\right)\right\rangle = 0. \tag{52}$$

In addition, for any $v \in \Omega_\theta$, we have

$$\left\langle v, Mu^* + q\right\rangle = \left\langle Wv, W^{-1}\left(Mu^* + q\right)\right\rangle \geq 0. \tag{53}$$

From (52) and (53), we have

$$\left\langle W\left(v - u^*\right), W^{-1}\left(Mu^* + q\right)\right\rangle \geq 0. \tag{54}$$

Setting $Wv = P_\Omega(Wu - W^{-1}(Mu + q))$ in (54), it follows that

$$\begin{aligned}&\left\langle W^{-1}\left(Mu^* + q\right), P_\Omega\left(Wu - W^{-1}\left(Mu + q\right)\right)\right. \\ &\left. - Wu^*\right\rangle \geq 0\end{aligned} \tag{55}$$

for any $u \in R^{n+m}$. For any $v \in R^{n+m}$ and $w \in \Omega_\theta$, we have

$$\left\langle v - P_\Omega(v), P_\Omega(v) - Ww\right\rangle \geq 0. \tag{56}$$

Let $v = Wu - W^{-1}(Mu + q)$, $w = u^*$ in (30). We obtain

$$\begin{aligned}&\left\langle e(u) - W^{-1}\left(Mu + q\right), P_\Omega\left(Wu - W^{-1}\left(Mu + q\right)\right)\right. \\ &\left. - Wu^*\right\rangle \geq 0.\end{aligned} \tag{57}$$

Based on (55) and (57), we get

$$\left\langle e(u) - W^{-1}M\left(u - u^*\right), W\left(u - u^*\right) - e(u)\right\rangle \geq 0. \tag{58}$$

It follows that

$$\begin{aligned}&\left\langle u - u^*, \left(W - W^{-1}M\right)e(u)\right\rangle \\ &\quad\geq \|e(u)\|^2 + \left\langle u - u^*, M\left(u - u^*\right)\right\rangle.\end{aligned} \tag{59}$$

Because $M^T = -M$, we have $\langle u - u^*, M(u - u^*)\rangle = 0$. Thus we get

$$\left\langle u - u^*, \left(W - W^{-1}M\right)e(u)\right\rangle \geq \|e(u)\|^2. \tag{60}$$

By the equation above, we obtain

$$\begin{aligned}\frac{d}{dt}H(u(t)) &= \left(u - u^*\right)^T\frac{du}{dt} \\ &= -r\rho(u)\left(u - u^*\right)^T\left(W - W^{-1}M\right)e(u) \\ &\leq -r\rho(u)\|e(u)\|^2 \leq 0,\end{aligned} \tag{61}$$

which shows that the solution of neural network (47) is Lyapunov stable. □

Next, we give the globally convergent result of the trajectory of neural network (47).

Theorem 11. *Let u^* is the equilibrium point of neural network (47). The solution trajectory of neural network (47) with initial point u_0 is globally convergent to u^* and has finite convergence time.*

Proof. From (61), we know that the level set

$$L\left(u^0\right) := \left\{u \mid H(u) \le H\left(u^0\right)\right\} \tag{62}$$

is bounded. Then, based on the Invariant Set Theorem [25], we have that the solution trajectory $u(t)$ converges to $\overline{u}$ as $t \to +\infty$, where $\overline{u}$ is the largest invariant set in

$$\Lambda = \left\{u \in L\left(u^0\right) \mid \frac{dH(u(t))}{dt} = 0\right\}. \tag{63}$$

□

Now, we prove the result that $(du/dt)|_{u=\widetilde{u}} = 0$ if and only if $(dH(u(t))/dt)|_{u=\widetilde{u}} = 0$, and then we can obtain the globe convergence of neural network (47).

If $(du/dt)|_{u=\widetilde{u}} = 0$, then $(dH(u(t))/dt)|_{u=\widetilde{u}} = (\widetilde{u} - u^*)^T (du/dt)|_{u=\widetilde{u}} = 0$.

If $(dH(u(t))/dt)|_{u=\widetilde{u}} = 0$, from (61), we have

$$\begin{aligned} 0 &= \left.\frac{dH(u(t))}{dt}\right|_{u=\widetilde{u}} = \left(\widetilde{u} - u^*\right)^T \left.\frac{du}{dt}\right|_{u=\widetilde{u}} \\ &\le -r\rho(\widetilde{u}) \|e(u)\|^2 \le 0. \end{aligned} \tag{64}$$

So $e(\widetilde{u}) = 0$. Then

$$\left.\frac{du}{dt}\right|_{u=\widetilde{u}} = -r\rho(\widetilde{u})\left(W - W^{-1}M\right)e(\widetilde{u}) = 0. \tag{65}$$

From the conclusions above, $u(t)$ converges globally to the equilibrium point u^*. Moreover, from Lemma 9 and the same argument as in [14, 26], the neural network (47) has finite convergence time.

5. Simulation Experiments

In this section, we give some examples to test the simulation performance of neural network (47). The neural network is run in the MATLAB 7.0 environment on an Intel Core processor 1.80GHZ personal computer with 4.0GB of RAM. The numerical examples are solved by ODE23 in the ode solver, which is a nonstiff medium order method. In the ode solver,$r = 0.8$, $RelTol = 10^{-6}$, and $AbsTol = 10^{-9}$ for the test examples.

Example 12. Consider the following problem with two 3-dimensional circular cones:

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ \text{s.t.} \quad & x_1 + 2x_2 + x_6 = 9 \\ & x_1 + x_4 + 4x_5 = 20 \\ & x_2 + x_3 + x_5 = 6 \\ & x_1 + x_2 = 4 \\ & x_3 + 2x_5 = 8 \\ & \left\|\left(x_2, x_3\right)^T\right\| \le \tan\theta_1 x_1 \\ & \left\|\left(x_5, x_6\right)^T\right\| \le \tan\theta_2 x_4, \end{aligned} \tag{66}$$

where

$$\begin{aligned} A &= \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 \end{pmatrix}, \\ b &= \begin{pmatrix} 9 \\ 20 \\ 6 \\ 4 \\ 8 \end{pmatrix}, \\ c &= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \\ \theta_1 &= \frac{\pi}{3}, \\ \theta_2 &= \frac{\pi}{3}. \end{aligned} \tag{67}$$

The optimal solution of Example 12 is

$$x^* = (2.288, 1.712, 0.575, 2.863, 3.712, 3.288)^T \tag{68}$$

and

$$y^* = (1.628, -0.640, -2.267, 0.012, 3.267)^T. \tag{69}$$

Figures 1 and 2 depict the trajectories of neural network (47) with the initial values $x_0 = (0, 0, 0, 0, 0, 0)^T$ and $y_0 = (0, 0, 0, 0, 0)^T$ converging to its solutions x^* and y^*, respectively.

Example 13. Consider the following problem with three 3-dimensional circular cones:

$$\begin{aligned} \min \quad & x_2 + 2x_3 + 3x_4 + x_5 + 2x_6 + 2x_7 + x_8 + x_9 \\ \text{s.t.} \quad & x_1 + x_2 + 2x_3 + 2x_5 + x_7 = 15 \end{aligned}$$

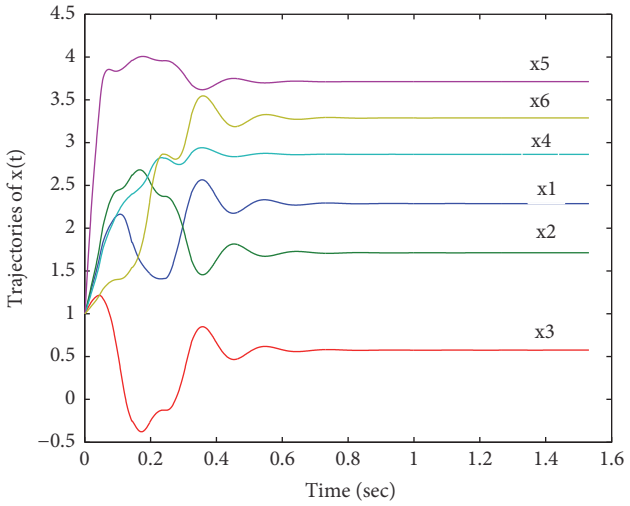

Figure 1: The x trajectories followed from Example 12.

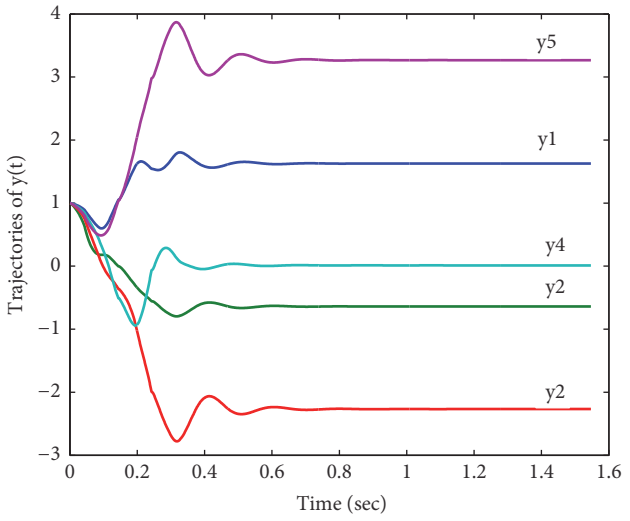

Figure 2: The y trajectories followed from Example 12.

$$
\begin{aligned}
&4x_1 + 6x_2 + 3x_3 + x_5 + 2x_7 + x_8 = 34\\
&x_3 + x_4 + x_5 + x_6 + 3x_7 + x_8 + x_9 = 24\\
&2x_2 + x_3 + x_6 + x_7 + x_8 + x_9 = 10\\
&x_2 + x_5 + 2x_6 + 2x_7 + x_8 + 2x_9 = 16\\
&\left\|\left(x_2, x_3\right)^T\right\| \le \tan\theta_1 x_1\\
&\left\|\left(x_5, x_6\right)^T\right\| \le \tan\theta_2 x_4\\
&\left\|\left(x_8, x_9\right)^T\right\| \le \tan\theta_3 x_7,
\end{aligned}
\tag{70}
$$

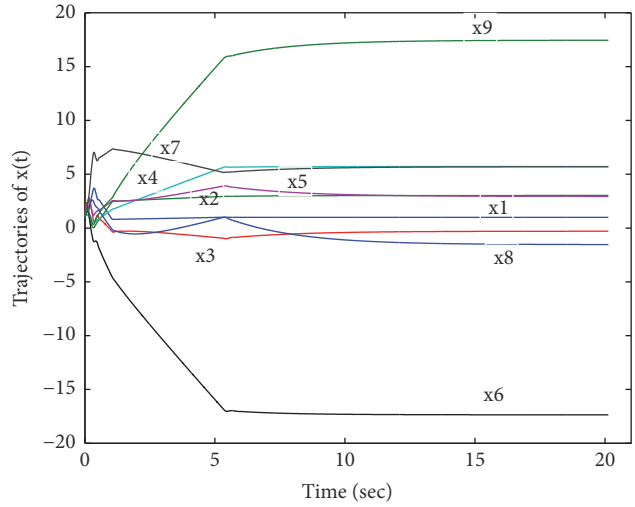

Figure 3: The x trajectories followed from Example 13.

where

$$
A = \begin{pmatrix}
1 & 1 & 2 & 0 & 2 & 0 & 1 & 0 & 0\\
4 & 6 & 3 & 0 & 1 & 0 & 2 & 1 & 0\\
0 & 0 & 1 & 1 & 1 & 1 & 3 & 1 & 1\\
0 & 2 & 1 & 0 & 0 & 1 & 1 & 1 & 1\\
0 & 1 & 0 & 0 & 1 & 2 & 2 & 1 & 2
\end{pmatrix},
$$

$$
b = \begin{pmatrix}
15\\
34\\
24\\
10\\
16
\end{pmatrix},
\tag{71}
$$

$$
c = \begin{pmatrix}0 & 1 & 2 & 3 & 1 & 2 & 2 & 1 & 1\end{pmatrix},
$$

$$
\theta_1 = 0.4\pi,
$$

$$
\theta_2 = 0.4\pi,
$$

$$
\theta_3 = 0.4\pi.
$$

The optimal solution of Example 13 is

$$
\begin{aligned}
x^* = (&0.987, 3.025, -0.297, 5.721, 2.945,\\
&-17.359, 5.692, -1.536, 17.450)^T
\end{aligned}
\tag{72}
$$

and

$$
y^* = (0.714, -0.314, 0.228, 1.268, -0.192)^T. \tag{73}
$$

Figures 3 and 4 depict the trajectories of neural network (47) with the initial values $x_0 = (0,0,0,0,0,0,0,0,0)^T$ and $y_0 = (0,0,0,0,0)^T$ converging to its solutions x^* and y^*, respectively.

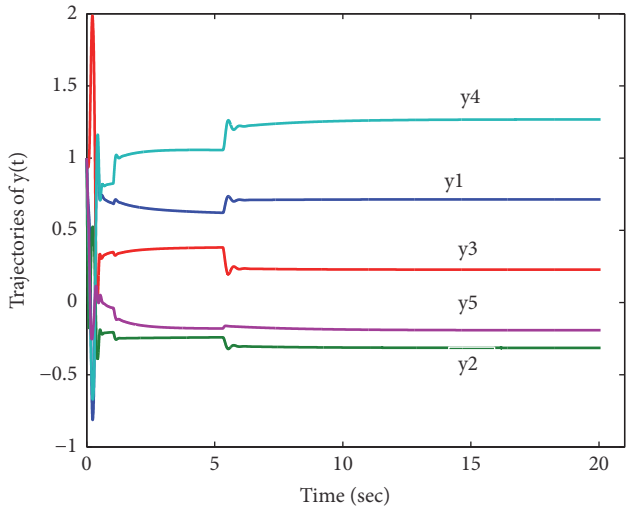

Figure 4: The y trajectories followed from Example 13.

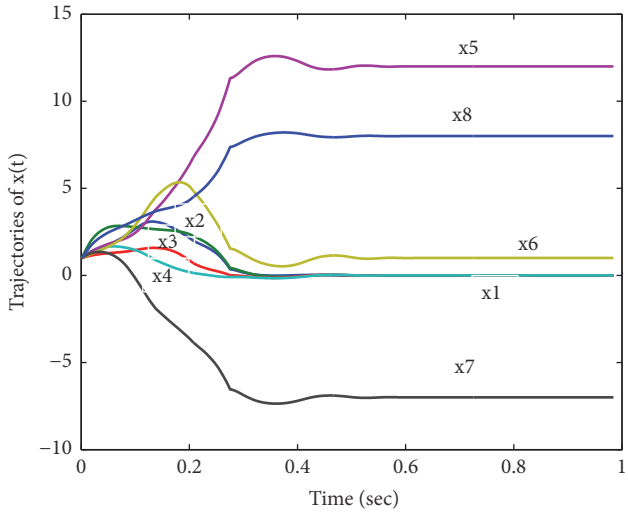

Figure 5: The x trajectories followed from Example 14.

Example 14. Consider the problem with two 4-dimensional circular cones as follows:

$$\begin{aligned}
\min \quad & 6x_1 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \\
\text{s.t.} \quad & 3x_2 + x_4 + 2x_8 = 16 \\
& 2x_2 + x_3 + x_4 + x_5 = 12 \\
& x_1 + x_5 + x_6 + x_7 + x_8 = 14 \\
& x_1 + x_4 + x_5 + x_7 = 5 \\
& x_2 + x_5 + 2x_7 + x_8 = 6 \\
& \left\| (x_2, x_3, x_4)^T \right\| \le \tan\theta_1 x_1 \\
& \left\| (x_6, x_7, x_8)^T \right\| \le \tan\theta_2 x_5,
\end{aligned} \tag{74}$$

where

$$A = \begin{pmatrix} 0 & 3 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 & 1 \end{pmatrix},$$

$$b = \begin{pmatrix} 16 \\ 12 \\ 14 \\ 5 \\ 6 \end{pmatrix},$$

$$c = \begin{pmatrix} 6 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

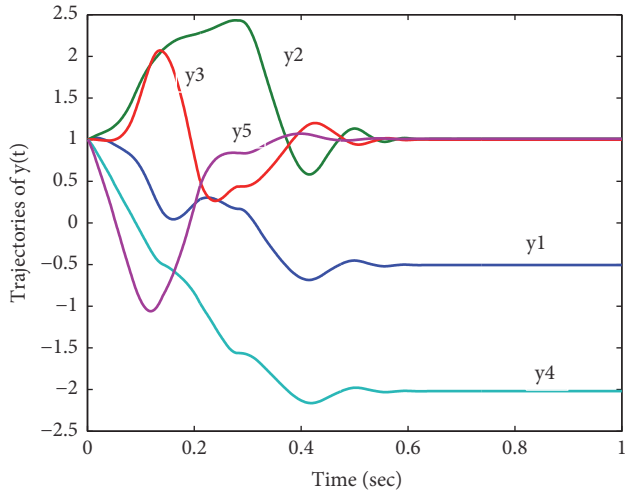

Figure 6: The y trajectories followed from Example 14.

$$\begin{aligned}
\theta_1 &= \frac{\pi}{4}, \\
\theta_2 &= \frac{\pi}{3}.
\end{aligned} \tag{75}$$

The optimal solution of Example 14 is

$$x^* = (0, 0, 0, 0, 12, 1, -7, 8)^T \tag{76}$$

and

$$y^* = (-0.505, 1.010, 1.000, -2.019, 1.010)^T. \tag{77}$$

Figures 5 and 6 depict the trajectories of neural network (47) with the initial values $x_0 = (0, 0, 0, 0, 0, 0, 0, 0)^T$ and $y_0 = (0, 0, 0, 0, 0)^T$ converging to its solutions x^* and y^*, respectively.

In paper [14], two neural networks are proposed for the second-order cone programs. The first neural network

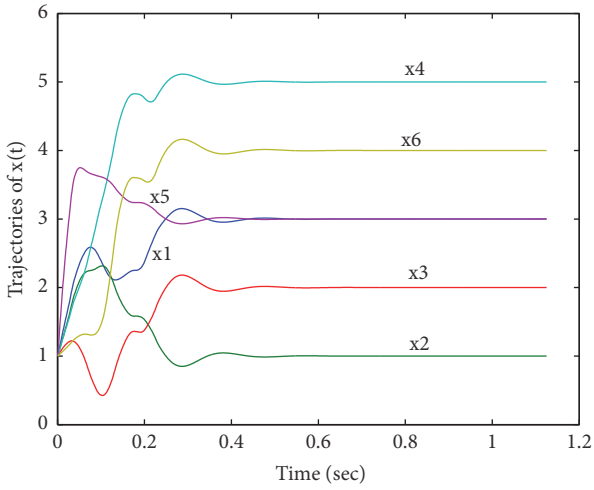

Figure 7: Transient behavior of the neural network (47) in Example 15.

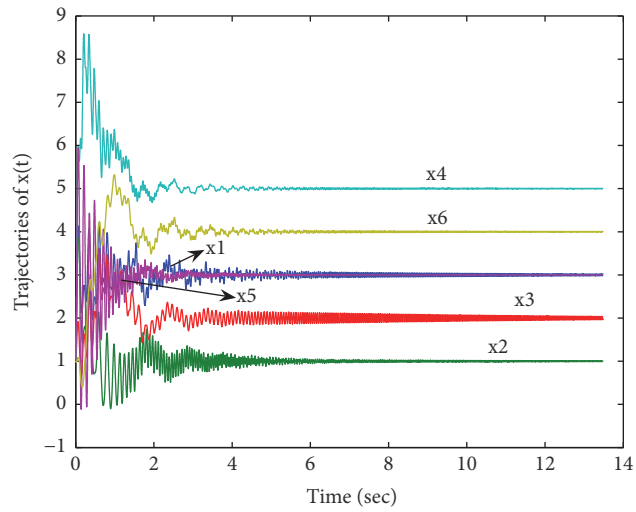

Figure 8: Transient behavior of the neural network with CP function in Example 15.

uses the Fischer-Burmeister function to achieve an unconstrained minimization with a merit function. The second neural network utilizes the natural residual function with the cone projection (CP) function to achieve low computation complexity. The second neural network is a projection neural network. Here we compared our projection neural network method with the neural network based on the cone projection function in paper [14] by the next two test examples.

Example 15. In Example 12, let $\theta_1 = \pi/4$, $\theta_2 = \pi/4$. Then the circular cone programming problem is converted into a second-order cone programming problem. This test problem is from Example 5.2 in paper [14]. This problem has an optimal solution $x^* = (3, 1, 2, 5, 3, 4)^T$.

Figures 7 and 8 depict the trajectories obtained using neural network (47) and the neural network with CP function in [14], respectively.

The simulation results show that both trajectories are convergent to x^*, but the neural network with the CP function yields the oscillating trajectory and has longer convergence time than the neural network (47).

Example 16. In Example 14, let $\theta_1 = \pi/4$, $\theta_2 = \pi/4$. Then a second-order cone problem is obtained, which has an optimal solution $x^* = (0, 0, 0, 0, 12, 1, -7, 8)^T$.

Figures 9 and 10 depict the trajectories obtained using neural network (47) and the neural network with CP function in [14], respectively.

The simulation results show that both trajectories are convergent to x^*. In addition, neural network with the CP function yields the oscillating trajectory and has longer convergence time than the neural network (47).

Example 17. In this example, we use the grasping force optimization problem for the multifingered robotic hand [4, 8, 14] to demonstrate the effectiveness of the proposed neural network. The force optimization is to minimize the

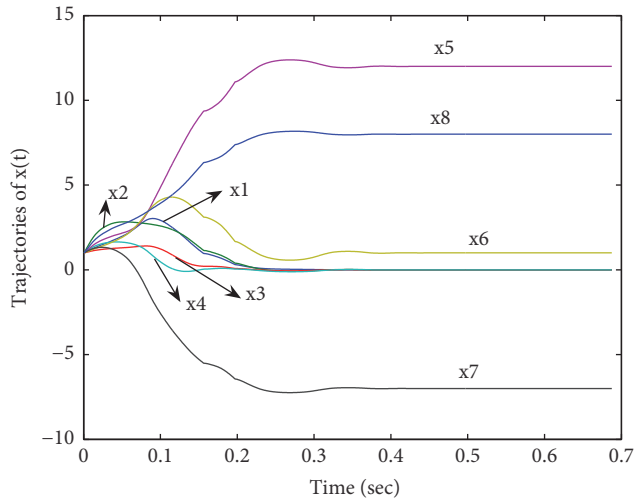

Figure 9: Transient behavior of the neural network (47) in Example 16.

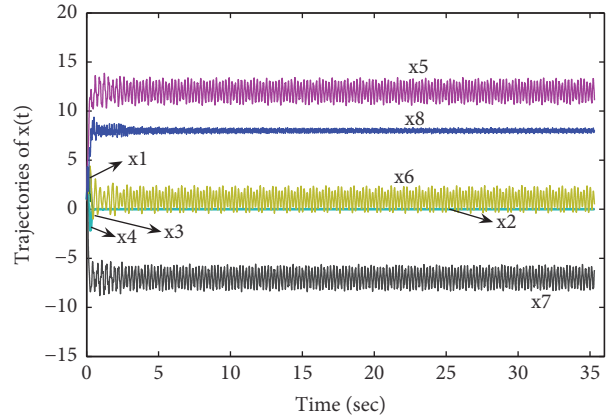

Figure 10: Transient behavior of the neural network with CP function in Example 16.

magnitude of grasping force from every finger applied to an object. For the robotic hand with m fingers, the optimization problem can be formulated as

$$\begin{aligned} \min \quad & \frac{1}{2} f^T f \\ \text{s.t.} \quad & Gf = -f_{ext} \\ & \left\| (f_{i1}, f_{i2})^T \right\| \le \mu f_{i3} \quad (i = 1, 2, \cdots, m), \end{aligned} \tag{78}$$

where $f = [f_{11}, f_{12}, \cdots, f_{m3}]$ is the grasping force, G is the grasping transformation matrix, f_{ext} is the time-varying external wrench, and μ is the friction coefficient. Problem (78) is a convex quadratic circular cone programming problem. In [14], problem (78) is formulated as a convex quadratic second-order cone programming problem by variable transformation.

In this example, we consider a three-fingered grasping force optimization example [14]. The three-finger robot hand grasps a polyhedral with the grasp points $[0, 1, 0]^T$, $[1, 0.5, 0]^T$, and $[0, -1, 0]^T$, and the robot hand moves along a vertical circular trajectory of radius r with a constant velocity v. Let $x = [f_{13}, f_{11}, f_{12}, f_{23}, f_{21}, f_{22}, f_{33}, f_{31}, f_{32}]^T$. Then problem (78) is reformulated as

$$\begin{aligned} \min \quad & \frac{1}{2} x^T Q x \\ \text{s.t.} \quad & Ax = b \\ & \left\| (x_2, x_3) \right\| \le \tan(\theta_1)\, x_1 \\ & \left\| (x_5, x_6) \right\| \le \tan(\theta_2)\, x_4 \\ & \left\| (x_8, x_9) \right\| \| \le \tan(\theta_3)\, x_7, \end{aligned} \tag{79}$$

where $Q = \mathrm{diag}(1, 1, 1, 1, 1, 1, 1, 1, 1)$ and

$$\begin{aligned} A &= \begin{pmatrix} 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & -0.5 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0.5 & 0 & -1 & 0 & 1 & 0 \end{pmatrix}, \\ b &= \begin{pmatrix} 0 \\ -f_c \sin\theta(t) \\ M_1 g - f_c \cos\theta(t) \\ 0 \\ 0 \\ 0 \end{pmatrix}, \\ c &= (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)^T, \\ \theta_1 &= \theta_2 = \theta_3 = \operatorname{actan}\left(\mu^{-1}\right). \end{aligned} \tag{80}$$

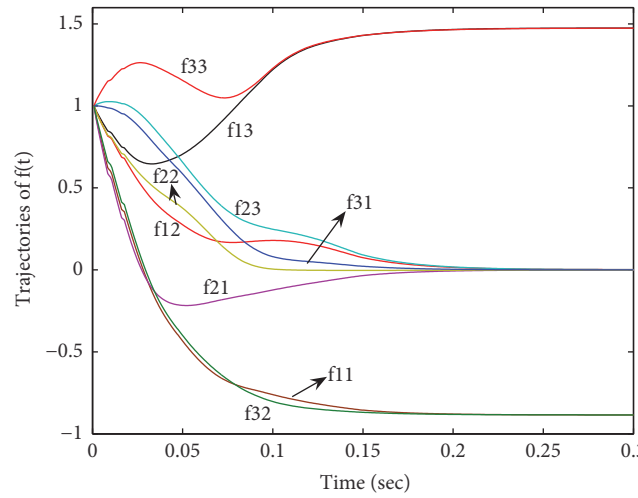

Figure 11: Transient behavior of the neural network (47) when $t = 0.5s$.

Here, M_1 is the mass of the polyhedral, $g = 9.8m/s^2$, $f_c = Mv^2/r$ the centripetal force, t the time, and $\theta(t) = vt/r \in [0, 2\pi]$. In this example, we set the data as follows: $M_1 = 0.1kg$, $r = 0.2m$, $n = 0.4\pi m/s$, and $\mu = 0.6$. In the time-varying grasping force, we only test numerical examples when $t = 0s$ and $t = 0.5s$. The other results are easily given when $t \in (0, 0.5)$.

Because the neural network (47) is designed to solve the linear circular cone programming problem, it cannot be used to solve problem (34) directly. But our neural network can be extended to convex quadratic circular cone programming problem easily. The extension method is only to change the residual function $e(u)$ and the matrix M in neural network (47) as follows:

$$\begin{aligned} M &= \begin{bmatrix} Q & -A^T \\ A & 0 \end{bmatrix}, \\ e(u) &= \begin{bmatrix} Tx - P_K\left(Tx - T^{-1}\left(Qx + c - A^T y\right)\right) \\ Ax - b \end{bmatrix}. \end{aligned} \tag{81}$$

The results of the projection neural networks (47) to solve Example 17 are shown in Figures 11 and 12 when $t = 0s$ and $t = 0.5s$, respectively. The simulation results demonstrate that the neural networks are convergent for the grasping force optimization problem.

For these simulation examples, the proposed network in (47) with other initial points always converges to the theoretical optimal solution. The simulation results demonstrate that the neural networks are effective for the circular cone programming.

6. Conclusion

In the paper, a projection neural network method is proposed for the circular cone programming. The projection equation and the contraction technique are used to construct the energy function and dynamical system. In the method,

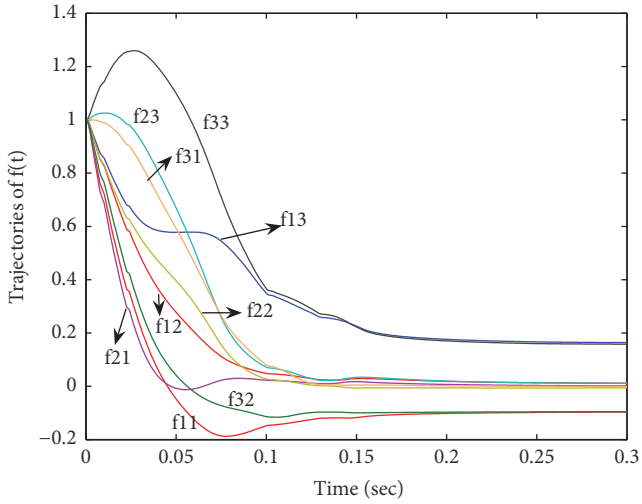

FIGURE 12: Transient behavior of the neural network (47) when $t = 0s$.

the projection and contraction technique accelerates the convergence of the neural network method. In addition, the proposed neural network requires less state variables, so the neural network method is simple and efficient.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Science Foundations for Young Scientists of China (11101320, 61201297), National Science Basic Research Plan in Shaanxi Province of China (2015JM1031), and the Fundamental Research Funds for the Central Universities (JB150713).

References

[1] J. Dattorro, *Meboo Publishing*, Convex Optimization and Euclidean Distance Geometry. Meboo Publishin, Palo Alto, 2005.

[2] J. Zhou and J.-S. Chen, "Properties of circular cone and spectral factorization associated with circular cone," *Journal of Nonlinear and Convex Analysis. An International Journal*, vol. 14, no. 4, pp. 807–816, 2013.

[3] J. Zhou, J.-S. Chen, and B. S. Mordukhovich, "Variational analysis of circular cone programs," *Optimization. A Journal of Mathematical Programming and Operations Research*, vol. 64, no. 1, pp. 113–147, 2015.

[4] S. P. Boyd and B. Wegbreit, "Fast computation of optimal contact forces," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1117–1132, 2007.

[5] B. León, A. Morales, and J. Sancho-Bru, "Robot Grasping Foundations," in *From Robot to Human Grasping Simulation*, vol. 19 of *Cognitive Systems Monographs*, pp. 15–31, Springer International Publishing, Cham, 2014.

[6] JF. Bonnans and HR. Cabrera, "Perturbation analysis of second-order cone programming problems," *Math Program*, vol. 104, pp. 205–227, 2005.

[7] A. Ruszczynski, *Nonlinear Optimization*, Princeton University Press, New Jersey, 2006.

[8] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical Programming*, vol. 95, no. 1, Ser. B, pp. 3–51, 2003.

[9] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra and its Applications*, vol. 284, no. 1-3, pp. 193–228, 1998.

[10] X. D. Chen, D. Sun, and J. Sun, "Complementarity functions and numerical experiments on some smoothing Newton methods for second-order-cone complementarity problems," *Computational optimization and applications*, vol. 25, no. 1-3, pp. 39–56, 2003.

[11] 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 Journal on Optimization*, vol. 20, no. 1, pp. 297–320, 2009.

[12] J. Zhou, J.-S. Chen, and H.-F. Hung, "Circular cone convexity and some inequalities associated with circular cones," *Journal of Inequalities and Applications*, 2013:571, 17 pages, 2013.

[13] P. Ma, Y. Bai, and J.-S. Chen, "A self-concordant interior point algorithm for nonsymmetric circular cone programming," *Journal of Nonlinear and Convex Analysis. An International Journal*, vol. 17, no. 2, pp. 225–241, 2016.

[14] C.-H. Ko, J.-S. Chen, and C.-Y. Yang, "Recurrent neural networks for solving second-order cone programs," *Neurocomputing*, vol. 74, no. 17, pp. 3646–3653, 2011.

[15] X. Miao, J.-S. Chen, and C.-H. Ko, "A smoothed NR neural network for solving nonlinear convex programs with second-order cone constraints," *Information Sciences*, vol. 268, pp. 255–270, 2014.

[16] X. Miao, J.-S. Chen, and C.-H. Ko, "A neural network based on the generalized FB function for nonlinear convex programs with second-order cone constraints," *Neurocomputing*, vol. 203, pp. 62–72, 2016.

[17] Y. Xia, J. Wang, and L.-M. Fok, "Grasping-force optimization for multifingered robotic hands using a recurrent neural network," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 549–554, 2004.

[18] B. He and H. Yang, "A neural-network model for monotone linear asymmetric variational inequalities," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 11, no. 1, pp. 3–16, 2000.

[19] R. K. Miller and A. N. Michel, *Ordinary differential equations*, Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York-London, 1982.

[20] J. Zabczyk, *Mathematical Control Theory: An Introduction*, Birkhuser, Boston, Mass, USA, 1992.

[21] J. Faraut and A. Korányi, *Analysis on Symmetric Cones*, Oxford University Press, New York, NY, USA, 1994.

[22] J. r. Outrata and D. Sun, "On the coderivative of the projection operator onto the second-order cone," *Set-Valued Analysis*, vol. 16, no. 7-8, pp. 999–1014, 2008.

[23] LC. Kong, L. Tuncel, and NH. Xiu, "Clarke Generalized Jacobian of the Projection onto Symmetric Cones," *Set-Valued and Variational Analysis*, vol. 17, pp. 135–151, 2009.

[24] D. G. Luenberger, *Introduction to Linear and Nolinear Programming*, Addison-wesley Publishing Company, Boston, 1973.

[25] R. M. Golden, *Mathematical methods for neural network analysis and design*, MIT Press, Cambridge, 1996.

[26] Y. Xia and J. Wang, "A recurrent neural network for solving nonlinear convex programs subject to linear constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 16, no. 2, pp. 379–386, 2005.