to appear in Neurocomputing, 2011

Recurrent Neural Networks for Solving Second-Order Cone Programs

Chun-Hsu Ko [1]
Department of Electrical Engineering,
I-Shou University
Kaohsiung County 840, Taiwan

Jein-Shan Chen [2]
Department of Mathematics
National Taiwan Normal University
Taipei 11677, Taiwan

Ching-Yu Yang [3]
Department of Mathematics
National Taiwan Normal University
Taipei, Taiwan 11677

January 5, 2011
(revised on June 1, 2011)

Abstract This paper proposes using the neural networks to efficiently solve the second-order cone programs (SOCP). To establish the neural networks, the SOCP is first reformulated as a second-order cone complementarity problem (SOCCP) with the Karush-Kuhn-Tucker conditions of the SOCP. The SOCCP functions, which transform the SOCCP into a set of nonlinear equations, are then utilized to design the neural networks. We propose two kinds of neural networks with the different SOCCP functions. The first neural network uses the Fischer-Burmeister function to achieve an unconstrained minimization with a merit function. We show that the merit function is a Lyapunov function and this neural network is asymptotically stable. The second neural network utilizes the natural residual function with the cone projection function to achieve low computation complexity. It is shown to be Lyapunov stable and converges globally to an optimal solution

[1]E-mail: chko@isu.edu.tw

[2]Corresponding author, Member of Mathematics Division, National Center for Theoretical Sciences, Taipei Office. The author's work is partially supported by National Science Council of Taiwan. E-mail: jschen@math.ntnu.edu.tw

[3]E-mail: yangcy@math.ntnu.edu.tw

under some condition. The SOCP simulation results demonstrate the effectiveness of the proposed neural networks.

Key words. SOCP, Neural network, Merit function, Fischer-Burmeister function, Cone projection function, Lyapunov stable.

AMS subject classifications. 92B20, 90C33, 65K05

1 Introduction

Second-order cone program (SOCP) has been widely applied in engineering optimization [1]. It requires solving the optimization problem subject to the linear equality and second-order cone inequality constraints [2]. Numerical approaches such as the interior-point method [1] or the merit function method [3] can effectively solve the SOCP. However, many engineering dynamic systems, such as force analysis in robot grasping [1, 4] and control applications [5, 6], require the real-time SOCP solutions. As a result, efficient approaches for solving the real-time SOCP are needed. Prior research [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18] indicates that the neural networks can be used to solve various optimization problems. Furthermore, the neural networks based on circuit implementation exhibit the real-time processing ability. We consider it is appropriate to utilize the neural networks for efficiently solving the SOCP problems.

The recurrent neural network was introduced by Hopfield and Tank [7] for solving linear programming problems. Kennedy and Chua [8] proposed an extended neural network for solving nonlinear convex programming problems thereafter, while their approach involves the penalty parameter which affects the neural network accuracy. To find the exact solutions, more neural networks for optimization have been further developed. Among them, the primal-dual neural network [9, 10, 11] with the global stability is proposed for providing the exact solutions of the linear and quadratic programming problems. The projection neural network, developed by Xia and Wang [12, 14, 15], was proposed to efficiently solve many optimization problems and variational inequalities. Since the SOCP is a nonlinear convex problem, both primal-dual neural network [16] and projection neural network [17] can be used to provide the SOCP solution. However, they require many state variables, leading to high model complexity. It thus motivates the development of more compact neural networks for SOCP.

The SOCP can be solved by analyzing its Karush-Kuhn-Tucker (KKT) optimality conditions which leads to the second-order cone complementarity problem (SOCCP) [3, 19, 20]. The approaches [3, 20] based on the SOCCP functions, such as Fischer-Burmeister (FB) and natural residual functions, can be further utilized for solving the SOCCP. In the merit function approach [3], an unconstrained smooth minimization with

the FB function is achieved in finding the SOCCP solution. On the other hand, the semi-smooth approach [20] uses the natural residual function with the cone projection (CP) function to reformulate the SOCCP as a set of nonlinear equations and then apply the non-smooth Newton method to obtain the solution. Previous studies have demonstrated the feasibility of these SOCCP functions in solving the SOCP problems. We also use them in our neural network design. In this paper, we propose two novel neural networks for efficiently solving the SOCP problems. One is based on the gradient of the smooth merit function derived from the FB function [18]. The other is an extended projection neural network by replacing the scalar projection function [12, 14, 15] with the CP function. These neural networks are with less state variables than those previously proposed [16, 17] for solving the SOCP. Furthermore, they are shown to be stable and globally convergent to the SOCP solutions.

This paper is organized as follows. Section 2 introduces the second-order cone program and its SOCCP formulation. In Section 3, the neural network based on the Fischer-Burmeister function is proposed and analyzed. In Section 4, the second neural network based on the cone projection function is proposed. Its global stability is also verified. In Section 5, several SOCP examples are presented to demonstrate the effectiveness of the proposed neural networks. Finally, the conclusions are given in Section 6.

2 Problem Formulation

In this section, we introduce the second-order cone program and reformulate it as a second-order cone complementarity problem. The second-order cone program is in the form of

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b,\ x \in K. \end{array} \tag{1}$$

Here $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear continuously differentiable function, $A \in \mathbb{R}^{m\times n}$ is a full row rank matrix, $b \in \mathbb{R}^m$ is a vector, and K is a Cartesian product of second-order cones (or Lorentz cones), expressed as

$$K = K^{n_1} \times K^{n_2} \times \cdots \times K^{n_N} \tag{2}$$

where $N, n_1, \cdots, n_N \geq 1, n_1 + \cdots + n_N = n$, and

$$K^{n_i} := \left\{(x_{i1}, x_{i2}, \cdots, x_{in_i})^T \in \mathbb{R}^{n_i} |\ \|(x_{i2}, \cdots, x_{in_i})\| \leq x_{i1}\right\}$$

with $\|\cdot\|$ denoting the Euclidean norm and K^1 the set of nonnegative reals $\mathbb{R}_+$. A special case of equation (2) is $K = \mathbb{R}^n_+$, namely the nonnegative orthant in $\mathbb{R}^n$, which corresponds to $N = n$ and $n_1 = \cdots = n_N = 1$. When f is linear, i.e., $f = c^T x$ with $c \in \mathbb{R}^n$, SOCP (1) reduces to the following linear SOCP:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b,\ x \in K. \end{array} \tag{3}$$

The KKT optimality conditions for (1) are given by

$$\begin{cases} \nabla f(x) - A^T y - \lambda = 0, \\ x^T \lambda = 0,\ x \in K,\ \lambda \in K, \\ Ax = b, \end{cases} \tag{4}$$

where $y \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^n$. When f is convex, these conditions are sufficient for optimality. It also can be written as

$$\begin{cases} x^T(\nabla f(x) - A^T y) = 0,\ x \in K,\ \nabla f(x) - A^T y \in K, \\ Ax = b. \end{cases} \tag{5}$$

By solving the system (5), we may obtain a primal-dual optimal solution of SOCP (1). Note that system (5) involves the SOCCP. To efficiently solve it, we propose using the neural network approaches with the FB function and CP function, respectively, described below.

3 Neural Network Design with Fischer-Burmeister Function

It is known that the merit function approach [3] can be used for solving system (5). Motivated by this approach, we propose a neural network with the Fischer-Burmeister function to find the minimal of the merit function and study its global stability.

In [3], system (5) is shown to be equivalent to an unconstrained smooth minimization problem via the merit function approach, described as

$$\min\ E(x,y) = \Psi_{FB}(x, \nabla f(x) - A^T y) + \frac{1}{2}\|Ax - b\|^2, \tag{6}$$

where $E(x,y)$ is a merit function, $\Psi_{FB}(x,y) = \dfrac{1}{2}\displaystyle\sum_{i=1}^{N} \|\phi_{FB}(x_i, y_i)\|^2$, $x = (x_1, \cdots, x_N)^T$, $y = (y_1, \cdots, y_N)^T \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_N}$, and ϕ_{FB} is the Fischer-Burmeister function defined as

$$\phi_{FB}(x_i, y_i) := (x_i^2 + y_i^2)^{1/2} - x_i - y_i. \tag{7}$$

Based on the gradient of the objective $E(x,y)$ in minimization problem (6), we propose the first neural network for solving the SOCP, with the following dynamic equation

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \rho \begin{pmatrix} -\nabla_x E(x,y) \\ -\nabla_y E(x,y) \end{pmatrix}, \tag{8}$$

where ρ is a positive scaling factor and

$$\begin{cases} \nabla_x E(x,y) &= \nabla_x \Psi_{FB}(x, \nabla f(x) - A^T y) \\ & \quad +\nabla^2 f(x) \cdot \nabla_y \Psi_{FB}(x, \nabla f(x) - A^T y) + A^T(Ax - b), \\ \nabla_y E(x,y) &= -A \cdot \nabla_y \Psi_{FB}(x, \nabla f(x) - A^T y). \end{cases} \tag{9}$$

For linear SOCP (3), the above equations reduce to

$$\begin{cases} \nabla_x E(x,y) = \nabla_x \Psi_{FB}(x, c - A^T y) + A^T(Ax - b), \\ \nabla_y E(x,y) = -A \cdot \nabla_y \Psi_{FB}(x, c - A^T y). \end{cases} \tag{10}$$

Note that the Jordan product [3] is required for calculating $\nabla_x \Psi_{FB}$ and $\nabla_y \Psi_{FB}$ which are introduced in the appendix. And, the dynamic equation (8) can be realized by a recurrent neural network with FB function as shown in Figure 1. The circuit for the neural network realization requires $n+m$ integrators, n processors for $\nabla f(x)$, n^2 processors for $\nabla^2 f(x)$, n processors for $\nabla_x \Psi_{FB}$, m processors for $\nabla_y \Psi_{FB}$, $4mn$ connection weights and some summers. Furthermore, the neural network (8) is asymptotically stable, as proven in the following theorem.

Theorem 3.1 *If $u^* = (x^*, y^*)$ is an isolated equilibrium point of neural network (8), then $u^* = (x^*, y^*)$ is asymptotically stable for (8).*

Proof. We assume that $u^* = (x^*, y^*)$ is an isolated equilibrium point of neural network (8) over a neighborhood $\Omega_* \subseteq \mathbb{R}^n$ of u^* such that $\nabla E(x^*, y^*) = 0$ and $\nabla E(x, y) \neq 0$, $\forall (x, y) \in \Omega_* \setminus \{(x^*, y^*)\}$. First we show that $E(x, y)$ is a Lyapunov function for u^* at Ω_*. Since

$$\nabla_y E(x^*, y^*) = -A \cdot \nabla_y \Psi_{FB}(x^*, \nabla f(x^*) - A^T y^*) = 0,$$

from Lemma 3 and Proposition 1 of [3], we have

$$\nabla_x \Psi_{FB}(x^*, \nabla f(x^*) - A^T y^*) = \nabla_y \Psi_{FB}(x^*, \nabla f(x^*) - A^T y^*) = 0.$$

Moreover, from Proposition 1 of [3], this says

$$\Psi_{FB}(x^*, \nabla f(x^*) - A^T y^*) = 0.$$

Then from equation (9),

$$\begin{aligned} \nabla_x E(x^*, y^*) &= \nabla_x \Psi_{FB}(x^*, \nabla f(x^*) - A^T y^*) \\ &\quad + \nabla^2 f(x^*) \cdot \nabla_y \Psi_{FB}(x^*, \nabla f(x^*) - A^T y^*) + A^T(Ax^* - b) = 0, \end{aligned}$$

which implies that $A^T(Ax^* - b) = 0$. Because $A \in \mathbb{R}^{m \times n}$ is a full row rank matrix, we must have $Ax^* - b = 0$, which yields

$$E(x^*, y^*) = \Psi_{FB}(x^*, \nabla f(x^*) - A^T y^*) + \frac{1}{2}\|Ax^* - b\|^2 = 0.$$

Next, we claim that $E(x, y) > 0$, $\forall (x, y) \in \Omega_* \setminus \{(x^*, y^*)\}$. If not, there is an $(x, y) \in \Omega_* \setminus \{(x^*, y^*)\}$ such that $E(x, y) = 0$, this says that $\Psi_{FB}(x, \nabla f(x) - A^T y) = 0$ and $Ax = b$, then $\nabla_x E(x, y) = 0$ and $\nabla_y E(x, y) = 0$. Hence, (x, y) is an equilibrium point of

neural network (8), contradicting with that $u^* = (x^*, y^*)$ is an isolate equilibrium point. Finally,

$$\begin{aligned} & \frac{dE(x(t), y(t))}{dt} \\ = \; & [\nabla_{(x(t),y(t))} E(x(t), y(t))]^T (-\rho \nabla_{(x(t),y(t))} E(x(t), y(t))) \\ = \; & -\rho \|\nabla_{(x(t),y(t))} E(x(t), y(t))\|^2 \\ \leq \; & 0. \end{aligned}$$

Therefore, the function $E(x, y)$ is a Lyapunov function for neural network (8) over the set Ω_*. Since $u^* = (x^*, y^*)$ is an isolated equilibrium point of neural network (8), we have

$$\frac{dE(x(t), y(t))}{dt} < 0, \; \forall (x(t), y(t)) \in \Omega_* \setminus \{(x^*, y^*)\}.$$

Thus, u^* is asymptotically stable for neural network (8). $\square$

4 Neural Network Design with Cone Projection Function

In this section, we propose another neural network associated with the cone projection function to solve system (5) for obtaining the SOCP solution and study its stability. In fact, from [24, Prop. 3.3], we know that such cone projection onto K has a special formula given as

$$P_K(z) = [\lambda_1(z)]_+ u_z^{(1)} + [\lambda_2(z)]_+ u_z^{(2)},$$

where $[\cdot]_+$ means the scalar projection, $\lambda_1(z)$, $\lambda_2(z)$ and $u_z^{(1)}$, $u_z^{(2)}$ are the spectral values and the associated spectral vectors of $z = (z_1, z_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$, respectively, given by

$$\begin{cases} \lambda_i(z) = z_1 + (-1)^i \|z_2\|, \\ u_z^{(i)} = \dfrac{1}{2} \left(1, (-1)^i \dfrac{z_2}{\|z_2\|} \right), \end{cases}$$

for $i = 1, 2$. The CP function $P_K(z)$ has the following property, called projection theorem [21], which is useful in our subsequent analysis.

Property 4.1 *Let K be a nonempty closed convex subset of $\mathbb{R}^n$. Then, for each $z \in \mathbb{R}^n$, $P_K(z)$ is the unique vector $\bar{z} \in K$ satisfying $(y - \bar{z})^T (z - \bar{z}) \leq 0, \; \forall y \in K$.*

Employing the natural residual function with the CP function [19, 20], system (5) can be equivalently written as

$$\begin{cases} x - P_K(x - \nabla f(x) + A^T y) = 0, \\ Ax - b = 0, \end{cases} \tag{11}$$

where $x = (x_1, \cdots, x_N)^T \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_N}$ with $x_i = (x_{i1}, x_{i2}, \cdots, x_{in_i})^T,\ i = 1, \cdots, N$, and $P_K(x) = [P_K(x_1), \cdots, P_K(x_N)]^T$.

Based on the equivalent formulation in (11) and employing the ideas for networks used in [12, 13], we consider the second neural network for solving the SOCP, with the following dynamic equations:

$$\frac{d}{dt}\begin{pmatrix} x \\ y \end{pmatrix} = \rho \begin{pmatrix} -x + P_K(x - \nabla f(x) + A^T y) \\ -Ax + b \end{pmatrix}, \tag{12}$$

where ρ is a positive scaling factor. The dynamic equations can be realized by a recurrent neural network with the cone projection function as shown in Figure 2. The circuit for the neural network realization requires $n + m$ integrators, n processors for $\nabla f(x)$, N processors for cone projection mapping P_K, $2mn$ connection weights and some summers. Compared with the first neural network in (8), the second neural network (12) dose not require to calculate $\nabla^2 f(x)$, resulting in lower model complexity.

To analyze the stability of the neural network in equation (12), we first give three lemmas and one proposition.

Lemma 4.1 *Let $F(u)$ be defined as*

$$F(u) := F(x, y) := \begin{pmatrix} -x + P_K(x - \nabla f(x) + A^T y) \\ -Ax + b \end{pmatrix}. \tag{13}$$

Then, $F(u)$ is semi-smooth. Moreover, $F(u)$ is strongly semi-smooth if $\nabla^2 f(x)$ is locally Lipschitz continuous.

Proof. This is an immediate consequence of [20, Theorem 1]. □

Proposition 4.1 *For any initial point $u_0 = (x_0, y_0)$ where $x_0 := x(t_0) \in K$, there exists a unique solution $u(t) = (x(t), y(t))$ for neural network (12). Moreover, $x(t) \in K$.*

Proof. For simplicity, we assume $K = K^n$. The analysis can be carried over to the general case where K is the Cartesian product of second-order cones. From Lemma 4.1, $F(u) := F(x, y)$ is semi-smooth and Lipschitz continuous. Thus, there exists a unique solution $u(t) = (x(t), y(t))$ for neural network (12). Therefore, it remains to show that $x(t) \in K^n$. For convenience, we denote $x(t) := (x_1(t), x_2(t)) \in \mathbb{R} \times \mathbb{R}^{n-1}$. To complete the proof, we need to verify two things: (i) $x_1(t) \geq 0$ and (ii) $\|x_2(t)\| \leq x_1(t)$. First, from (12), we have

$$\frac{dx}{dt} + \rho x(t) = \rho P_K(x - \nabla f(x) + A^T y).$$

The solution of the first-order ordinary differential equation above is

$$x(t) = e^{-\rho(t-t_0)}x(t_0) + \rho e^{-\rho t}\int_{t_0}^{t} e^{\rho s}P_K(x - \nabla f(x) + A^T y)ds.$$

If we let $x(t_0) := (x_1(t_0), x_2(t_0)) \in \mathbb{R} \times \mathbb{R}^{n-1}$ and denote $z(t) := (z_1(t_0), z_2(t_0))$ as the term $P_K(x - (\nabla f(x) - A^T y))$, which leads to

$$\begin{aligned} x_1(t) &= e^{-\rho(t-t_0)}x_1(t_0) + \rho e^{-\rho t}\int_{t_0}^{t} e^{\rho s}z_1(s)ds, \\ x_2(t) &= e^{-\rho(t-t_0)}x_2(t_0) + \rho e^{-\rho t}\int_{t_0}^{t} e^{\rho s}z_2(s)ds. \end{aligned}$$

Due to both $x_0(t)$ and $z(t)$ belong to K^n, there have $x_1(t_0) \geq 0$, $\|x_2(t_0)\| \leq x_1(t_0)$ and $z_1(t) \geq 0$, $\|z_2(t)\| \leq z_1(t)$. Therefore, $x_1(t) \geq 0$ since both terms in the right-hand side are nonnegative. In addition,

$$\begin{aligned} \|x_2(t)\| &\leq e^{-\rho(t-t_0)}\|x_2(t_0)\| + \rho e^{-\rho t}\int_{t_0}^{t} e^{\rho s}\|z_2(s)\|ds \\ &\leq e^{-\rho(t-t_0)}x_1(t_0) + \rho e^{-\rho t}\int_{t_0}^{t} e^{\rho s}z_1(s)ds \\ &= x_1(t), \end{aligned}$$

which implies that $x(t) \in K^n$. $\square$

Lemma 4.2 *Let $H(u)$ be defined as*

$$H(u) := H(x, y) := \begin{pmatrix} \nabla f(x) - A^T y \\ Ax - b \end{pmatrix}. \tag{14}$$

Then, H is a monotone function if f is a convex function. Moreover, $\nabla H(u)$ is positive semi-definite if and only if $\nabla^2 f(x)$ is positive semi-definite.

Proof. Let $u = (x, y)$ and $\tilde{u} = (\tilde{x}, \tilde{y})$. Then, the monotonicity of H holds since

$$\begin{aligned} & (u - \tilde{u})^T(H(u) - H(\tilde{u})) \\ = \; & (x - \tilde{x})^T(\nabla f(x) - \nabla f(\tilde{x})) - (x - \tilde{x})^T(A^T(y - \tilde{y})) + (y - \tilde{y})^T(A(x - \tilde{x})) \\ = \; & (x - \tilde{x})^T(\nabla f(x) - \nabla f(\tilde{x})) \\ \geq \; & 0, \end{aligned}$$

where the last inequality is due to the convexity of $f(x)$, see [22, Theorem 3.4.5]. Furthermore, we observe that

$$\nabla H(u) = \begin{bmatrix} \nabla^2 f(x) & -A^T \\ A & 0 \end{bmatrix}.$$

Thus, we have

$$\begin{aligned} & u^T \nabla H(u) u \\ = & \begin{bmatrix} x^T & y^T \end{bmatrix} \begin{bmatrix} \nabla^2 f(x) & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ = & x^T \nabla^2 f(x) x, \end{aligned}$$

which indicates that the positive semi-definiteness of $\nabla H(u)$ is equivalent to the positive semi-definiteness of $\nabla^2 f(x)$. $\square$

Lemma 4.3 *Let $F(u)$, $H(u)$ be defined as in (13) and (14), respectively. Also, let $u^* = (x^*, y^*)$ be an equilibrium point of neural network (12) with x^* being an optimal solution of SOCP. Then, the following inequalities hold:*

$$(F(u) + u - u^*)^T (-F(u) - H(u)) \geq 0. \tag{15}$$

Proof. First, we denote $\lambda := \nabla f(x) - A^T y$. Then, we obtain

$$\begin{aligned} & (F(u) + u - u^*)^T (-F(u) - H(u)) \\ = & \begin{bmatrix} -x + P_K(x - \lambda) + (x - x^*) \\ (-Ax + b) + (y - y^*) \end{bmatrix}^T \begin{bmatrix} x - P_K(x - \lambda) - \lambda \\ (Ax - b) - (Ax - b) \end{bmatrix} \\ = & \begin{bmatrix} -x^* + P_K(x - \lambda) \\ (-Ax + b) + (y - y^*) \end{bmatrix}^T \begin{bmatrix} (x - \lambda) - P_K(x - \lambda) \\ 0 \end{bmatrix} \\ = & -(x^* - P_K(x - \lambda))^T ((x - \lambda) - P_K(x - \lambda)). \end{aligned}$$

Since $x^* \in K$, applying Property 4.1 gives

$$(x^* - P_K(x - \lambda))^T ((x - \lambda) - P_K(x - \lambda)) \leq 0.$$

Thus, inequality (15) is proved. $\square$

We now investigate the stability and convergence issues of neural network (12). First, we analyze the behavior of the solution trajectory of neural network (12) including existence and convergence. We then establish two kinds of stability for an isolated equilibrium point.

We know that every solution u^* to SOCP is an equilibrium point of neural network (12). If further u^* is an isolated equilibrium point of neural network (12), we show that u^* is Lyapunov stable.

Theorem 4.1 *If f is convex and twice differentiable, then the solution of neural network (12), with initial point $u_0 = (x_0, y_0)$ where $x_0 \in K$, is Lyapunov stable. Moreover, the solution trajectory of neural network (12) is extendable to the global existence.*

Proof. Again, for simplicity, we assume $K = K^n$. From Proposition 4.1, there exists a unique solution $u(t) = (x(t), y(t))$ for neural network (12) and $x(t) \in K^n$. Let $u^* = (x^*, y^*)$ be an equilibrium point of neural network (12) with x^* being an optimal solution of SOCP. We define a Lyapunov function as below:

$$E(u) := E(x, y) := -H(u)^T F(u) - \frac{1}{2}\|F(u)\|^2 + \frac{1}{2}\|u - u^*\|^2, \tag{16}$$

where $F(u)$ and $H(u)$ are given as in (13) and (14), respectively. From [23, Theorem 3.2], we know that E is continuously differentiable with

$$\nabla E(u) = H(u) - [\nabla H(u) - I]F(u) + (u - u^*).$$

It is also trivial that $E(u^*) = 0$. Then, we have

$$\begin{array}{rcl} \dfrac{dE(u(t))}{dt} & = & \nabla E(u(t))^T \dfrac{du}{dt} \\ & = & \{H(u) - [\nabla H(u) - I]F(u) + (u - u^*)\}^T \rho F(u) \\ & = & \rho\left\{[H(u) + (u - u^*)]^T F(u) + \|F(u)\|^2 - F(u)^T \nabla H(u) F(u)\right\}. \end{array}$$

Hence, inequality (15) in Lemma 4.3 implies

$$(H(u) + u - u^*)^T F(u) \le -H(u)^T (u - u^*) - \|F(u)\|^2,$$

which yields

$$\begin{array}{rcl} & & \dfrac{dE(u(t))}{dt} \\ & \le & \rho\left\{-H(u)^T (u - u^*) - F(u)^T \nabla H(u) F(u)\right\} \\ & = & \rho\left\{-H(u^*)^T (u - u^*) - (H(u) - H(u^*))^T (u - u^*) - F(u)^T \nabla H(u) F(u)\right\}. \end{array} \tag{17}$$

On the other hand, we know that

$$\begin{array}{rcl} & & (F(u^*) + u^* - u)^T (-F(u^*) - H(u^*)) \\ & = & -(x - P_K(x^* - \lambda^*))^T ((x^* - \lambda^*) - P_K(x^* - \lambda^*)). \end{array}$$

Since $x \in K^n$, applying Property 4.1 gives

$$(x - P_K(x^* - \lambda^*))^T ((x^* - \lambda^*) - P_K(x^* - \lambda^*)) \le 0.$$

Thus, we have $(F(u^*) + u^* - u)^T (-F(u^*) - H(u^*)) \ge 0$. Note that $F(u^*) = 0$, we therefore obtain $-H(u^*)^T (u - u^*)^T \le 0$. Also the monotonicity of H implies $-(H(u) - H(u^*))^T (u - u^*) \le 0$. In addition, f is convex and twice differentiable if and only if $\nabla^2 f(x)$ is positive semidefinite and hence ∇H is positive semidefinite by Lemma 4.2, i.e., the second term $-F(u)^T \nabla H(u) F(u) \le 0$. The above discussions lead to $dE(u(t))/dt \le 0$.

In order to obtain $E(u)$ is a Lyapunov function and u^* is Lyapunov stable, we will show the following inequality:

$$-H(u)^T F(u) \ge \|F(u)\|^2. \tag{18}$$

To see this, we first observe that

$$\begin{aligned} & \|F(u)\|^2 + H(u)^T F(u) \\ = \; & (x - P_K(x-\lambda))^T((x-\lambda) - P_K(x-\lambda)). \end{aligned}$$

Since $x \in K$, applying Property 4.1 again, there holds

$$(x - P_K(x-\lambda))^T((x-\lambda) - P_K(x-\lambda)) \leq 0,$$

which yields the desired inequality (18). By combining equation (16) and inequality (18), we have

$$E(u) \geq \frac{1}{2}\|F(u)\|^2 + \frac{1}{2}\|u - u^*\|^2,$$

which says $E(u) > 0$ if $u \neq u^*$. Hence $E(u)$ is indeed a Lyapunov function and u^* is Lyapunov stable. Moreover, it holds that

$$E(u_0) \geq E(u) \geq \frac{1}{2}\|u - u^*\|^2 \quad \text{for} \quad t \geq t_0, \tag{19}$$

which means the solution trajectory $u(t)$ is bounded. Hence, it can be extended to global existence. □

Theorem 4.2 *Let $u^* = (x^*, y^*)$ be an equilibrium point of (12) with x^* being an optimal solution of SOCP. If f is twice differentiable and $\nabla^2 f(x)$ is positive definite, the solution of neural network (12), with initial point $u_0 = (x_0, y_0)$ where $x_0 \in K$, is globally convergent to u^* and has finite convergence time.*

Proof. From (19), the level set

$$L(u_0) := \{u \mid E(u) \leq E(u_0)\}$$

is bounded. Then, the Invariant Set Theorem [25] implies the solution trajectory $u(t)$ converges to θ as $t \to \infty$ where θ is the largest invariant set in

$$\Pi = \left\{ u \in L(u_0) \mid \frac{dE(u(t))}{dt} = 0 \right\}.$$

We will show that $du/dt = 0$ if and only if $dE(u(t))/dt = 0$ which yields that $u(t)$ converges globally to the equilibrium point $u^* = (x^*, y^*)$. Suppose $du/dt = 0$, then it is clear that $dE(u(t))/dt = \nabla E(u)^T(du/dt) = 0$. Let $\hat{u} = (\hat{x}, \hat{y}) \in \Pi$ which says $dE(\hat{u}(t))/dt = 0$. From (17), we know that

$$\frac{dE(\hat{u}(t))}{dt} \leq \rho\left\{-(H(\hat{u}) - H(u^*))^T(\hat{u} - u^*) - F(\hat{u})^T \nabla H(\hat{u}) F(\hat{u})\right\}.$$

Both terms inside the big parenthesis are nonpositive as shown in Lemma 4.2, so $(H(\hat{u}) - H(u^*))^T(\hat{u} - u^*) = 0$, $F(\hat{u})^T \nabla H(\hat{u}) F(\hat{u}) = 0$, and

$$\begin{aligned} & F(\hat{u})^T \nabla H(\hat{u}) F(\hat{u}) \\ = \; & \{-\hat{x} + P_K(\hat{x} - \nabla f(\hat{x}) + A^T\hat{y})\}^T \nabla^2 f(\hat{x}) \{-\hat{x} + P_K(\hat{x} - \nabla f(\hat{x}) + A^T\hat{y})\} \\ = \; & 0. \end{aligned}$$

The condition of $\nabla^2 f(\hat{x})$ being positive definite leads to

$$-\hat{x} + P_K(\hat{x} - \nabla f(\hat{x}) + A^T\hat{y}) = 0,$$

which is equivalent to $d\hat{x}/dt = 0$. On the other hand, similar to the arguments in Lemma 4.2, we have

$$\begin{aligned} & (\hat{u} - u^*)^T (H(\hat{u}) - H(u^*)) \\ = \; & (\hat{x} - x^*)^T (\nabla f(\hat{x}) - \nabla f(x^*)) \\ = \; & (\hat{x} - x^*)^T \nabla^2 f(x_s)(\hat{x} - x^*) \\ = \; & 0, \end{aligned}$$

where $x_s \in [x^*, \hat{x}]$. Again, the condition of $\nabla^2 f(x_s)$ being positive definite yields $\hat{x} = x^*$. Hence $d\hat{y}/dt = 0$ and therefore $d\hat{u}(t)/dt = 0$. From above, $u(t)$ converges globally to the equilibrium point $u^* = (x^*, y^*)$. Moreover, with Theorem 4.1 and following the same arguments as in [12, Theorem 2], the neural network (12) has finite convergence time.

5 Simulations

To demonstrate the effectiveness of the proposed neural networks, three illustrative SOCP problems are tested, described as below.

Example 5.1 *Consider the nonlinear convex SOCP [20] given by*

$$\begin{array}{ll} \text{minimize} & \exp(x_1 - x_3) + 3(2x_1 - x_2)^4 + \sqrt{1 + (3x_2 + 5x_3)^2} \\ \text{subject to} & Ax = b, \; x \in K^3 \times K^2 \end{array}$$

where

$$A = \begin{bmatrix} 4 & 6 & 3 & -1 & 0 \\ -1 & 7 & -5 & 0 & -1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

This problem has an approximate solution $x^* = [0.2324, -0.07309, 0.2206, 0.153, 0.153]^T$. We use the proposed neural networks with the FB and CP functions, respectively, to solve the problem with the trajectories obtained by them shown in Figures 3 and 4. From the simulation results, we found that both trajectories are globally convergent to x^* and the neural network with the CP function converged to x^* quicker than that with the FB function. On the other hand, the neural network with the CP function also has lower

model complexity than that with the FB function as mentioned in Sec. 4. Hence, the neural network with the CP function is preferable to the neural network with the FB function when both can globally converge to the optimal solution.

Example 5.2 *Consider the following linear SOCP given by*

$$\begin{array}{ll} \text{minimize} & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ \text{subject to} & Ax = b,\ x \in K^3 \times K^3 \end{array}$$

where

$$A = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 4 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 9 \\ 20 \\ 6 \\ 4 \\ 8 \end{bmatrix}$$

This problem has an optimal solution $x^* = [3, 1, 2, 5, 3, 4]^T$. Note that, its objective function is convex and the Hessian matrix $\nabla^2 f(x)$ is a zero matrix. Hence, the neural network with the FB function is asymptotically stable from Theorem 3.1 while the neural network with the CP function is Lyapunov stable from Theorem 4.1. Figures 5 and 6 display the trajectories obtained using the neural networks with the FB and CP functions, respectively. The simulation results show that both trajectories are convergent to x^*. Coinciding with above results of Theorems 3.1 and 4.1, the neural network with the CP function yields the oscillating trajectory and has longer convergence time than the neural network with the FB function.

Example 5.3 *Consider the grasping force optimization problem for the multi-fingered robotic hand [1, 4, 17]. Its goal is to find the minimum grasping force for moving an object. For the robotic hand with m fingers, the optimization problem can be formulated as*

$$\begin{array}{ll} \text{minimize} & \dfrac{1}{2} f^T f \\ \text{subject to} & Gf = -f_{ext} \\ & \|(f_{i1}, f_{i2})\| \le \mu f_{i3},\ (i = 1, \cdots, m) \end{array}$$

where $f = [f_{11}, f_{12}, \cdots, f_{m3}]^T$ is the grasping force, G the grasping transformation matrix, f_{ext} the time-varying external wrench, and μ the friction coefficient.

Letting $[x_{i1}, x_{i2}, x_{i3}] = [\mu f_{i3}, f_{i1}, f_{i2}], i = 1, \cdots, m$, and $x = [x_{11}, x_{12}, \cdots, x_{m3}]^T$, the problem can be reformulated as a nonlinear convex SOCP. For the three-finger grasp example in [17], the robot hand grasps a polyhedral with the grasp points $[0, 1, 1]^T$,

$[1, 0.5, 0]^T$, and $[0, -1, 0]^T$, and the robot hand moves along a vertical circular trajectory of radius r with a constant velocity ν. We reformulate the example as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Q x \\ \text{subject to} & Ax = b,\ x \in K^3 \times K^3 \times K^3 \end{array} \tag{20}$$

where $Q = \text{diag}(1/\mu^2, 1, 1, 1/\mu^2, 1, 1, 1/\mu^2, 1, 1)$

$$A = \begin{bmatrix} 0 & 0 & 1 & -1/\mu & 0 & 0 & 0 & 1 & 0 \\ -1/\mu & 0 & 0 & 0 & 0 & -1 & 1/\mu & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & -0.5 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0.5/\mu & 0 & -1 & 0 & 1 & 0 \end{bmatrix}$$

$$\text{and} \quad b = \begin{bmatrix} 0 \\ -f_c \sin\theta(t) \\ Mg - f_c \cos\theta(t) \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

where M is the mass of the polyhedral, $g = 9.8m/s^2$, $f_c = \dfrac{M\nu^2}{r}$ the centripetal force, t the time, and $\theta = \dfrac{\nu t}{r} \in [0, 2\pi]$. Note that problem (20) is a nonlinear convex SOCP and the matrix Q is positive definite. We know from Theorems 3.1 and 4.2 that both the proposed neural networks are globally convergent to the optimal solution. Under the conditions $M = 0.1kg$, $r = 0.2m$, $\nu = 0.4\pi\ m/s$, and $\mu = 0.6$, the time-varying grasping force obtained from the proposed neural networks is shown in Figure 7. We found that the maximum grasping force occurs at the position $\theta = \pi$ $(t = 0.5s)$ which corresponds to the maximum downward wrench. The simulation results demonstrate that the neural networks are effective in the SOCP applications.

6 Conclusion

In this paper, we have proposed two neural networks for efficiently solving the SOCP. The first neural network is based on gradient of the merit function derived from the FB function and was shown to be asymptotically stable. The second neural network with the CP function has low model complexity, and has been shown to be Lyapunov stable and converge globally to the SOCP solution under the positive definite condition of Hessian matrix of the objective function. The convergence of the neural networks has been validated with the simulation results of the SOCP examples. When the second

neural network with the CP functions yields oscillating trajectory, we can employ the neural network based on FB function instead, though it has higher model complexity. The proposed neural networks are thus ready for the SOCP applications.

During the reviewing process of this paper, we published another paper [26] which focuses on second-order cone constrained variational inequality problem. Since the KKT conditions of second-order cone programs can be recast as variational inequality problem, the paper [26] indeed deals with a broader class of optimization problems. However, the two neural networks considered therein are different from the two neural networks studied in this paper. More specifically, the FB method used in [26] is based on the smoothed FB function while the one studied here is based on regular FB function; the CP method in [26] is based on a Lagrangian model which is, even when it reduces to SOCP, not the same as the one investigated here. Due to the essential difference, the assumptions used to establish stability are also different. In view of this, it will be an interesting topic to do numerical comparison among these neural networks for SOCP.

Acknowledgement

The work was supported by National Science Council of Taiwan under the Grant NSC 97-2221-E-214-034.

References

[1] 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, pp. 193-228, 1998.

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

[3] J.-S. Chen and P. Tseng, *An unconstrained smooth minimization reformulation of the second-order cone complementarity problem*, Mathematical Programming, vol. 104, pp. 293–327, 2005.

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

[5] S. Boyd, C. Crusius and A. Hansson, *Control applications of nonlinear convex programming*, Journal of Control Process, vol. 8, no. 5, pp. 313-324, 1998.

[6] D. Bertsimas and D.B. Brown, *Constrained stochastic LQC: a tractable approach*, IEEE Transactions on Automatic Control, vol. 52, no. 10, pp. 1826-1841, 2007.

[7] D.W. Tank and J.J. Hopfield, *Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit*, IEEE Transactions on Circuits and Systems, vol. 33, no. 5, pp. 533-541, 1986.

[8] M.P. Kennedy and L.O. Chua, *A Neural network for nonlinear programming*, IEEE Transaction on Circuits and Systems, Vol. 35, No. 5, pp. 554-562, 1988.

[9] Y.S. Xia, *A new neural network for solving linear and quadratic programming problems*, IEEE Transactions on Neural Networks, vol. 7, no. 6, pp. 1544-1547, 1996.

[10] Q. Tao, J.D. Cao, M.S. Xue and H. Qiao, *A high performance neural network for solving nonlinear programming problems with hybrid constraints*, Physics Letters A, vol. 288, no. 2, pp. 88-94, 2001.

[11] J. Wang, Q. Hu, and D. Jiang, *A Lagrangian neural network for kinematic control of redundant robot manipulators*, IEEE Transactions on Neural Networks, vol. 10, no. 5, pp. 1123-1132, 1999.

[12] Y. Xia and J. Wang, *A recurrent neural network for solving nonlinear convex programs subject to linear constraints*, IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 379-386, 2005.

[13] Y. Xia, H. Leung and J. Wang, *A projection neural network and its application to constrained optimization problems*, IEEE Transactions on Circuits and Systems - Part I, vol. 49, pp. 447-458, 2002.

[14] Y. Xia and J. Wang, *A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints*, IEEE Transactions on Circuits and Systems-I: Regular Papers, vol. 51, no. 7, pp. 1385-1394, 2004.

[15] Y. Xia, H. Leung and J. Wang, *A general projection neural network for solving monotone variational inequalities and related optimization problems*, IEEE Transactions on Neural Networks, vol. 15, no. 2, pp. 318-328, 2004.

[16] X. Mu, S. Liu and Y. Zhang, *A neural network algorithm for second-order conic programming*, Second International Symposium on Neural Networks, Chongqing, China, Proceedings Part II, pp. 718-724, 2005.

[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] L. Z. Liao and H. D. Qi, *A neural network for the linear complementarity problem*, Mathematical and Computer Modeling, vol. 29, no. 3, pp. 9-18, 1999.

[19] J. S. Chen, X. Chen and P. Tseng, *Analysis of nonsmooth vector-valued function associated with second-order cone*, Mathematical Programming, vol. 101, no. 1, pp. 95-117, 2004.

[20] 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, pp. 297–320, 2009.

[21] D. P. Bertsekas, *Nonlinear Programming*, Belmont, MA: Athena Scientific, 1995.

[22] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Philadelphia: SIAM, 2000.

[23] M. Fukushima, *Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems*, Mathematical Programming, vol. 53, no. 1, pp. 99-110, 1992.

[24] M. Fukushima, Z.-Q. Luo, and P. Tseng, *Smoothing functions for second-order-cone complimentarity problems*, SIAM Journal on Optimization, vol. 12, pp. 436–460, 2002.

[25] R. Golden, *Mathematical Methods for Neural Network Analysis and Design*, Cambridge, MA: The MIT Press, 1996.

[26] J. Sun, J.-S. Chen and C.-H. Ko, *Neural networks for solving second-order cone constrained variational inequality problem*, to appear in Computational Optimization and Applications, 2011.

Appendix

In Appendix, we introduce the Jordan product and its properties used in the neural network with the FB function, which are needed when we write codes for simulations.

For any $x = (x_1, x_2) \in \mathbb{R} \times \mathbb{R}^{n-1}$, their Jordan product is defined as

$$x \circ y = (x^T y, y_1 x_2 + x_1 y_2).$$

Their sum of square is calculated by

$$x^2 + y^2 = (\|x\|^2 + \|y\|^2, 2x_1 x_2 + 2y_1 y_2).$$

The square root of x is

$$x^{1/2} = (s, \frac{x_2}{2s}), s = \sqrt{\frac{1}{2}\left(x_1 + \sqrt{x_1^2 - \|x_2\|^2}\right)}, \text{ if } x = 0, x^{1/2} = 0$$

and the determinant of x is $\det(x) = x_1^2 - \|x_2\|^2$. Furthermore, a matrix L_x is defined as

$$L_x = \begin{bmatrix} x_1 & x_2^T \\ x_2 & x_1 I \end{bmatrix},$$

and when $\det(x) \neq 0$, L_x is invertible with

$$L_x^{-1} = \frac{1}{\det(x)} \begin{bmatrix} x_1 & -x_2^T \\ -x_2 & \frac{\det(x)}{x_1} I + \frac{1}{x_1} x_2 x_2^T \end{bmatrix}.$$

Based on the properties of the Jordan product described above, the formulae of $\nabla_x \Psi_{FB}(x, y)$ and $\nabla_y \Psi_{FB}(x, y)$ in neural network (8) are calculated (see [3]) as

$$\nabla_x \Psi_{FB}(x, y) = \left(L_x L^{-1}_{(x^2+y^2)^{\frac{1}{2}}} - I\right) \phi_{FB}(x, y),$$

and

$$\nabla_y \Psi_{FB}(x, y) = \left(L_y L^{-1}_{(x^2+y^2)^{\frac{1}{2}}} - I\right) \phi_{FB}(x, y).$$

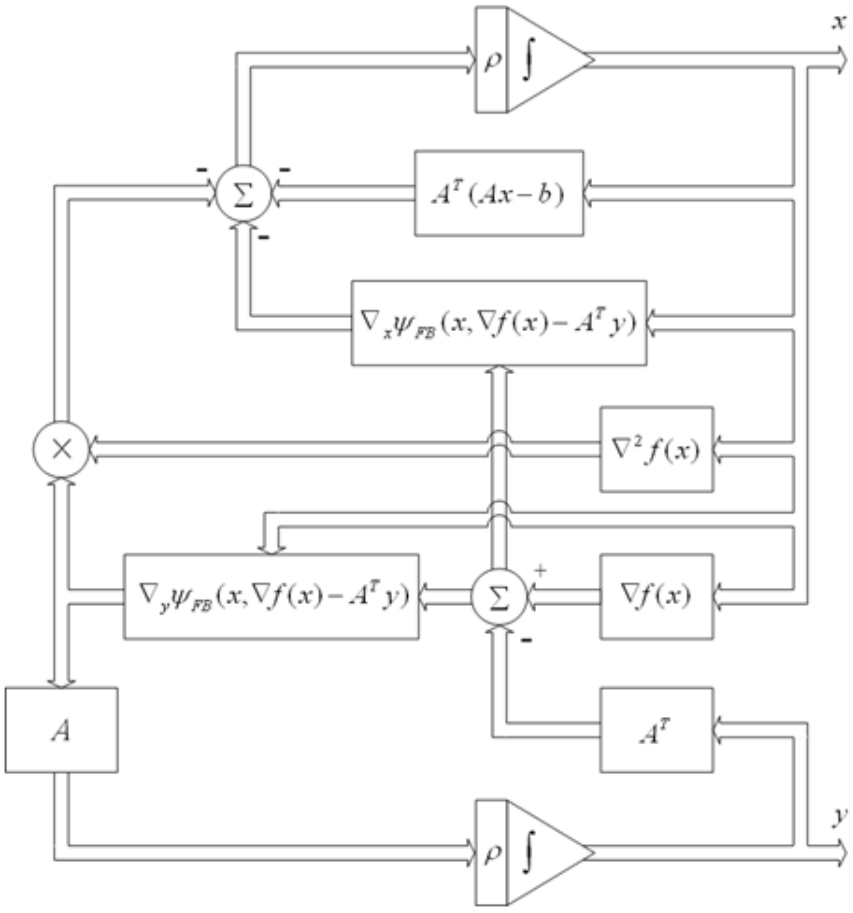

Figure 1: Block diagram of the proposed neural network with FB function.

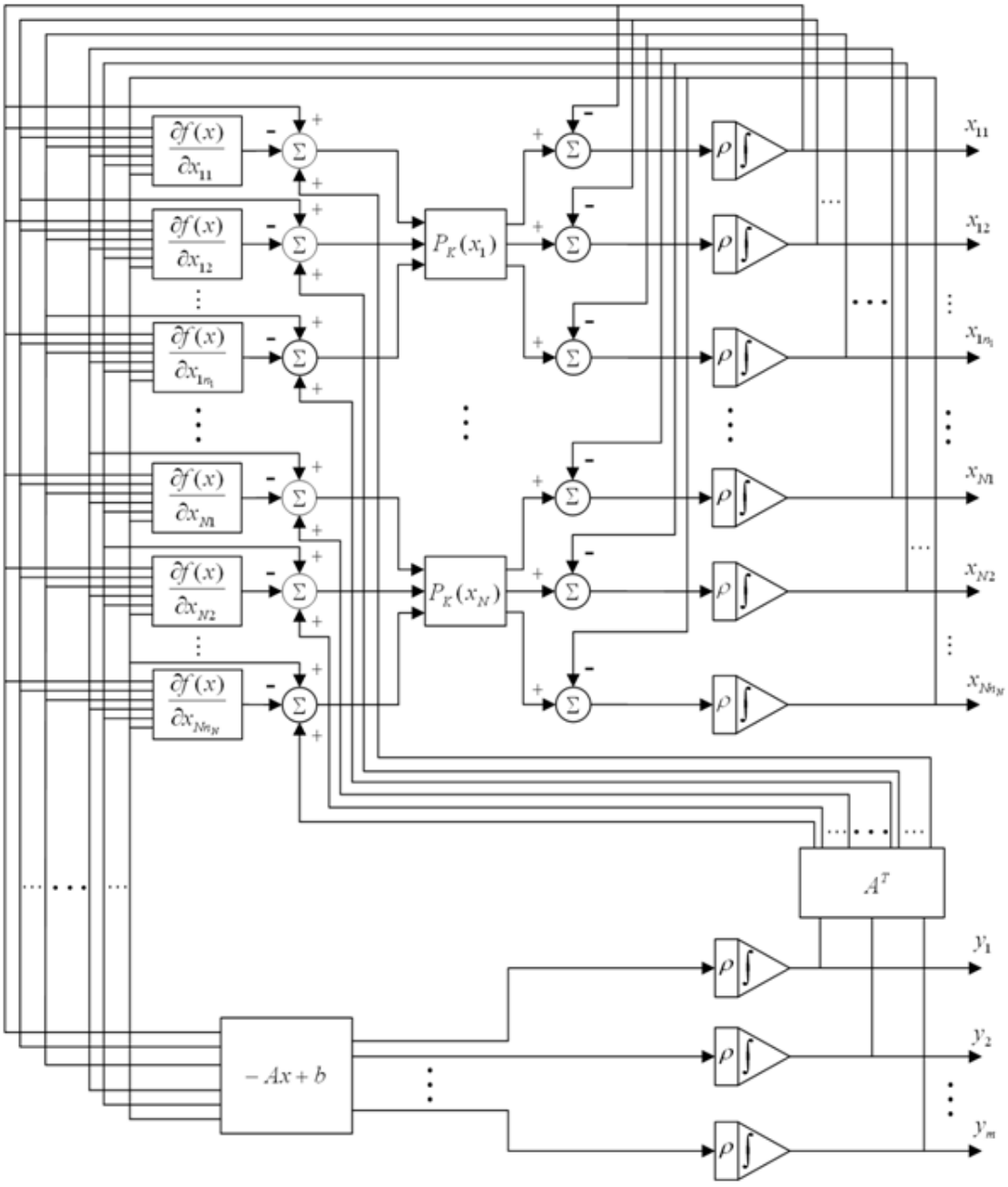

Figure 2: Block diagram of the proposed neural network with CP function.

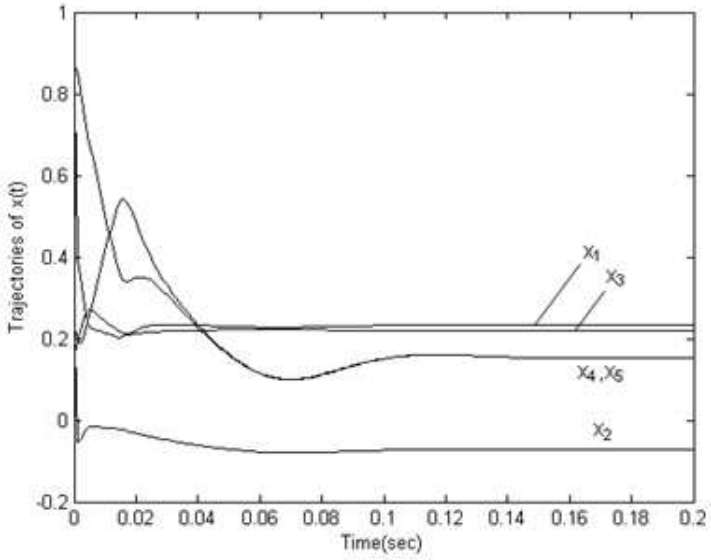

Figure 3: Transient behavior of the neural network with FB function in Example 5.1.

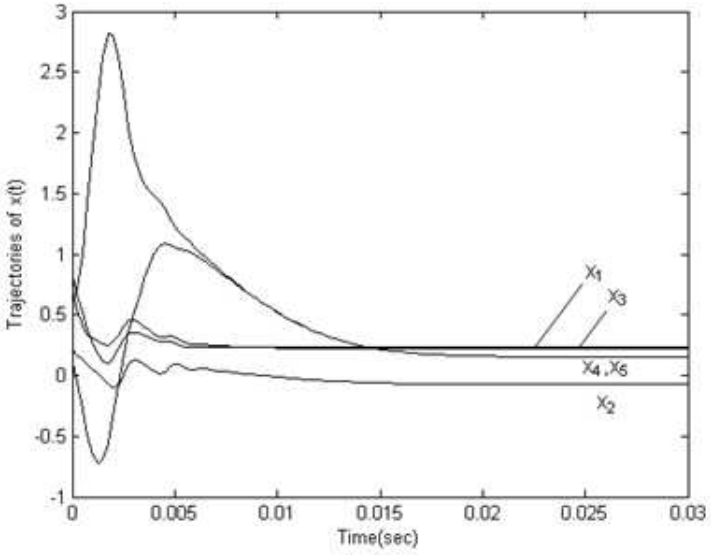

Figure 4: Transient behavior of the neural network with CP function in Example 5.1.

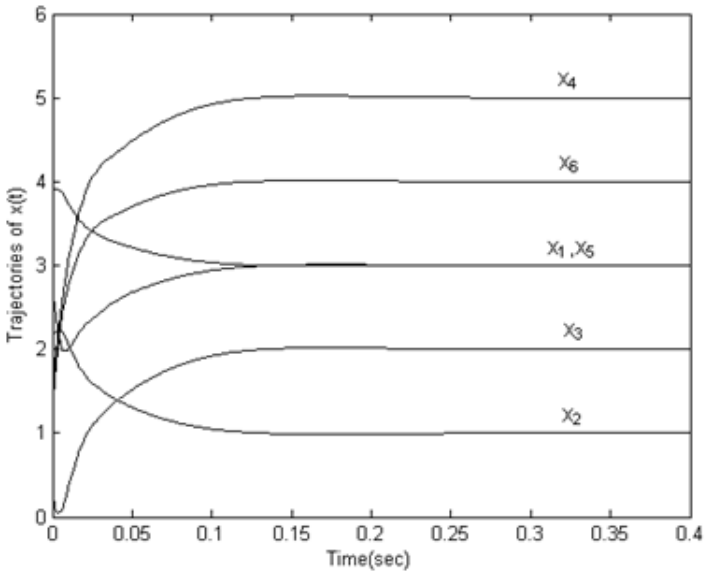

Figure 5: Transient behavior of the neural network with FB function in Example 5.2.

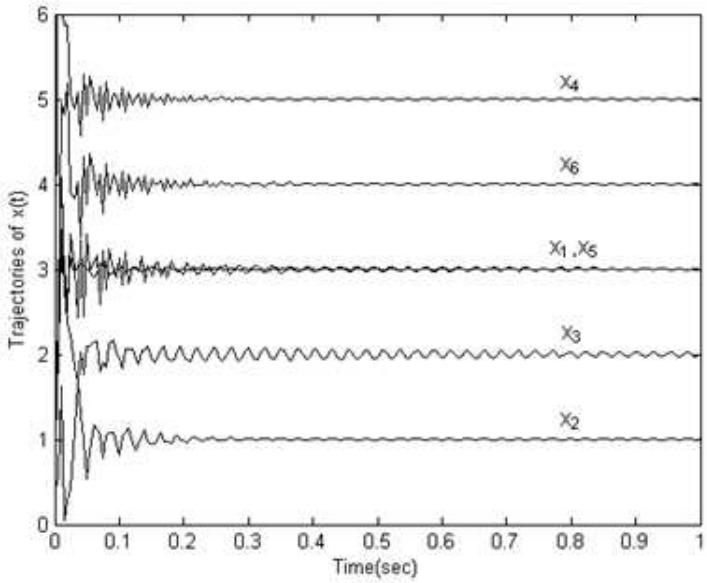

Figure 6: Transient behavior of the neural network with CP function in Example 5.2.

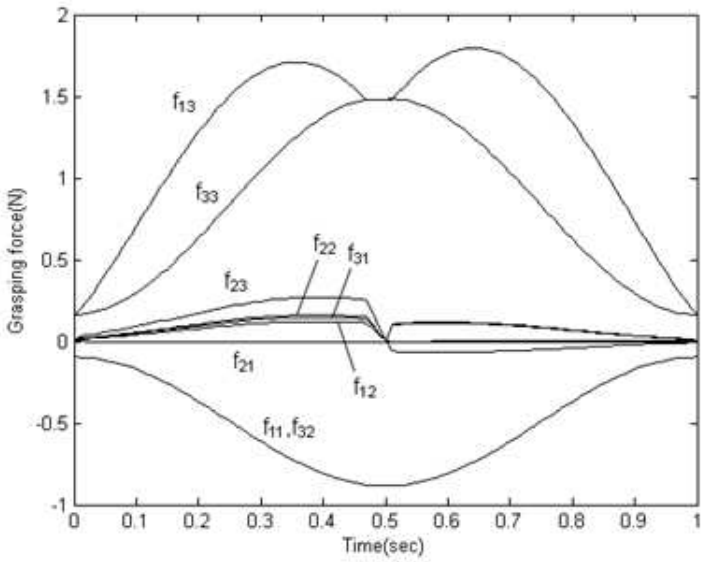

Figure 7: Grasping force obtained by using proposed neural networks in Example 5.3.