Finite time dual neural networks with a tunable activation function for solving quadratic programming problems and its application

Peng Miao[1], Yanjun Shen[2,*], Xiaohua Xia[3]

[1]*College of science, China Three Gorges University, Yichang, Hubei, China, 443002*

[2]*College of Electrical Engineering and New Energy, China Three Gorges University, Yichang, Hubei, China, 443002*

[3]*Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa*

Abstract

In this paper, finite time dual neural networks with a new activation function are presented to solve quadratic programming problems. The activation function has two tunable parameters, which give more flexibility to design a neural network. By Lyapunov theorem, the finite-time stability can be derived for the proposed neural networks model, and the actual optimal solutions of the quadratic programming problems can be obtained in finite time interval. Different from the existing recurrent neural networks for solving the quadratic programming problems, the neural networks of this paper have a faster convergent speed, at the same time, reduced oscillation when delay appears, and less sensitivity to the additive noise with careful selection the parameters. The effectiveness of our methods are validated by theoretical analysis and numerical simulations.

Keywords:
recurrent neural networks, finite-time stability, tunable activation function, quadratic programming

1. Introduction

Recently, recurrent neural networks have made great development. They are widely applied in scientific and engineering field, for example, optimization (Smith (1999); Li, Lou and Liu (2012)), control of chaos (Lin, Li and Liu (2012)), pattern classification (Burrows and Niranjan (1994); Husken and Stagge (2003)), signal processing (Skowronski and Harris (2007)), robotics (Li, Chen, Liu, Li and Liang (2012)), solving time-varying Sylvester equation (Li, Chen and Liu (2013)), the winners-take-all competition (Li, Liu and Li (2013); Liu and Wang (2008); Hu and Zhang (2009)), convex quadratic programming (Zhang and Wang (2002); Xia and Sun (2009); Xia Feng and Wang (2004)), kinematic control of redundant manipulators (Zhang, Wang and Xia (2003)) etc. Particularly, the Hopfield neural networks (Hopfield (1984)), the recurrent neural networks, can be used to online optimization.

With the development of recurrent neural networks, remarkable advances have been made in the field of online optimization. For example, by removing the explicit constraints and by introducing a penalty term into the cost function, and a recurrent neural network is designed to solve the constrained optimization problem in (Hopfield (1984); Rodriguez, Dominguez, Rueda and Sanchez (1990); Kennedy and Chua (1988)). However, the designed neural network only converges to the optimal solution asymptotically. In order to obtain the accurate solution, some scholars have done a lot of work. For example, in (Wu, Xia, Li and Chen (1996)) and (Zhang and Constantinides (1992)), dynamic Lagrange multipliers are introduced to regulate the constraints and the optimal solution can be obtained in finite time. However, the number of neurons in the neural network is increased. The reason is that extra neurons are required for the dynamics of the Lagrange multipliers. It is well known that the complexity and cost of its hardware implementation are relevant to the the number of neurons in the neural network. Then, the research on reduction of neuron number without losing efficiency and accuracy receives some researchers' attention (Hu and Zhang (2009); Liu and Wang (2006); Wang (2010); Liu and Wang (2011); Li, Li and Wang (2013). For instance, the authors in (Li, Li and Wang (2013)) present a dual neural network model with a continuous function, $|x|^r \mathrm{sign}(x)$ $(0 < r < 1)$. The finite-time convergence property and the optimality of the proposed neural network for solving the quadratic programming problem are proven. The parameter r has an effect on the convergence time. The neural network has a faster convergent speed with a smaller r. However, the chattering phenomenon will happen, especially in the case when time delay appears. On the other hand, the neural network with a smaller r is less sensitive to additive noise. Therefore, it is worth while to study finite-time dual networks for solving quadratic programming problems with a relative high robustness against time delay and noise.

*Corresponding author

[1]Peng Miao is a postgraduate student in the College of Science, China Three Gorges University.

[2]Yanjun Shen is a professor in the College of Electrical Engineering and New Energy, China Three Gorges University.

[3]Xiaohua Xia is a professor in the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria.

In the paper, we present recurrent neural networks with a tunable active function, $k_1|x|^r\text{sign}(x)+k_2x$, where k_1, k_2 are tunable positive parameters. These parameters are not only helpful to accelerate convergence speed, but also helpful to improve the robustness of the neural network with appearance of time delay and noise.

The paper is organized as follows. In Section 2, finite-time criteria and upper bounds of the convergence time are reviewed. In section 3, we present finite-time recurrent neural networks with a tunable activation function for solving quadratic programming problems. In section 4, numerical simulations are given to show the effectiveness of our methods. Section 5 concludes the paper.

2. Preliminaries

Consider the following system:

$$\dot{x}(t) = f(x(t)), f(0) = 0, x \in \mathcal{R}^n, x(0) = x_0, \tag{1}$$

where $f : \mathcal{D} \to \mathcal{R}^n$ is continuous on an open neighborhood $\mathcal{D}$ of the origin $x = 0$.

Definition 1 (Bhat and Bernstein (2000)): The equilibrium $x = 0$ of (1) is finite-time convergent if there are an open neighborhood $\mathcal{U}$ of the origin and a function $T_x : \mathcal{U} \setminus \{0\} \to (0, \infty)$, such that every solution trajectory $x(t, x_0)$ of (1) starting from the initial point $x_0 \in \mathcal{U} \setminus \{0\}$ is well-defined and unique in forward time for $t \in [0, T_x(x_0))$, and $\lim_{t\to T_x(x_0)} x(t, x_0) = 0$. Then, $T_x(x_0)$ is called the *convergence time* (of the initial state x_0). The equilibrium of (1) is finite-time stable if it is Lyapunov stable and finite-time convergent. If $U = \mathcal{D} = \mathcal{R}^n$, the origin is a globally finite-time stable equilibrium.

The following Lemmas provide sufficient conditions for the origin of the system (1) to be a finite-time stable equilibrium.

Lemma 1 (Bhat and Bernstein (2000)): Suppose there are a C^1 positive definite function $V(x)$ defined on a neighborhood $\mathcal{U} \subset \mathcal{R}^n$ of the origin, and real numbers $k_1 > 0$ and $0 < r < 1$, such that

$$\dot{V}(x)|_{(1)} \le -k_1V(x)^r, \forall x \in \mathcal{U}. \tag{2}$$

Then, the origin of the system (1) is locally finite-time stable. The convergence time T_1, depending on the initial state x_0, satisfies

$$T_1(x_0) \le \frac{V(x_0)^{1-r}}{k_1(1-r)}, \tag{3}$$

for all $x_0 \in \mathcal{U}$. Further, if $\mathcal{U} = \mathcal{R}^n$ and $V(x)$ is radially unbounded (that is $V(x) \to +\infty$ as $\|x\| \to +\infty$), the origin of system (1) is globally finite-time stable.

Lemma 2 (Shen and Xia (2008); Shen and Huang (2012)): If there are a C^1 positive definite function $V(x)$ defined on a neighborhood $\mathcal{U} \subset \mathcal{R}^n$ of the origin, and real numbers $k_1, k_2 > 0$ and $0 < r < 1$, such that

$$\dot{V}(x)|_{(1)} \le -k_1V(x)^r - k_2V(x), \forall x \in \mathcal{U}. \tag{4}$$

Then, the origin of system (1) is finite-time stable. The convergence time T_2 satisfies

$$T_2(x_0) \le \frac{\ln[1 + \frac{k_2}{k_1}V(x_0)^{1-r}]}{k_2(1-r)}, \tag{5}$$

for all $x_0 \in \mathcal{U}$. If $\mathcal{U} = \mathcal{R}^n$ and $V(x)$ is radially unbounded, the origin of system (1) is globally finite-time stable.

Remark 1: From Lemma 1 and Lemma 2, we can see the upper bound of the convergence time is relevant to r. It decreases with decrease of r. When r is greater than 0 but sufficiently close to 0, the term $|x|^r\text{sign}(x)$ is very close to $\text{sign}(x)$ for x with small absolute values values. Therefore, it may yield chattering phenomenon. For example, consider the following scalar differential equation:

$$\dot{x}(t) = -|x(t)|^r\text{sign}(x),\ 0 < r < 1. \tag{6}$$

The trajectories of (6) with different value of r are given in Fig.1. From Fig. 1 and Fig. 2, we can see that the trajectory of (6) with $r = 0.2$ has the fastest convergence speed, but the chatting phenomenon happens also. To overcome the problem, we can select a small value of k_1 and a large value of k_2 for the following scalar differential equation:

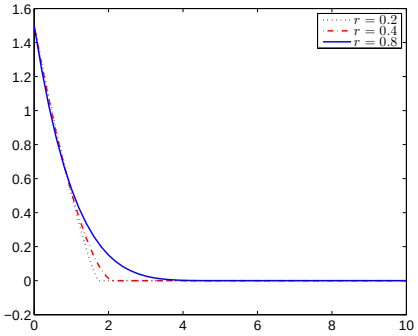

Figure 1: The trajectories of (6) with different value of r

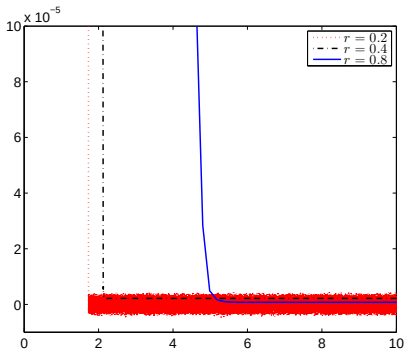

Figure 2: Partial enlarged view of Figure 1

$$\dot{x}(t) = -k_1|x(t)|^r\text{sign}(x) - k_2x,\ 0 < r < 1. \tag{7}$$

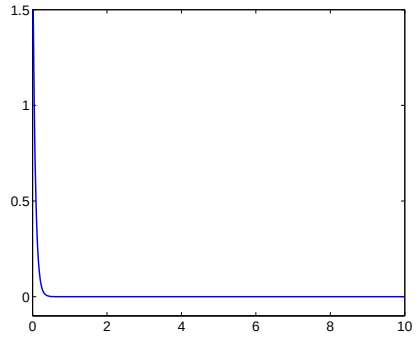

Figure 3: The trajectory of (7) with $r = 0.2$, $k_1 = 0.0001$, $k_2 = 15$

The trajectory of (7) is shown in Fig. 3. From the Fig. 4, we can see the chattering phenomenon disappears. In sliding mode control, the introduction of $-k_2 x$, called reaching control, to suppress chattering is the ideal of Gao and Hung in (Gao and Hung (1993)).

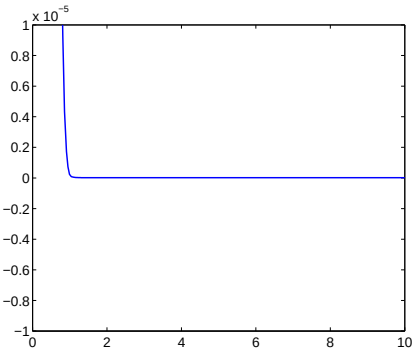

Figure 4: Partial enlarged view of Figure 3

The following Lemmas are also useful for our main results.

Lemma 3 (Li, Li and Wang (2013)): Let $\|x\|_a$ is the a-norm of $x = [x_1, x_2, ..., x_n]^T$, $\|x\|_a = (\sum_{i=1}^{n} |x_i|^a)^{\frac{1}{a}}$, for $0 < b < a$, we have

$$\|x\|_a \leq \|x\|_b. \tag{8}$$

Lemma 4 (Hwang (2004)): Let A be a Hermitian matrix of n, and let B be a principal sub-matrix of A of order $n-1$. If $\lambda_n \leq \lambda_{n-1} \leq ... \leq \lambda_2 \leq \lambda_1$, lists the eigenvalues of A and $\mu_n \leq \mu_{n-1} \leq ... \leq \mu_3 \leq \mu_2$ the eigenvalues of B, then

$$\lambda_n \leq \mu_n \leq \lambda_{n-1} \leq ... \leq \lambda_2 \leq \mu_2 \leq \lambda_1. \tag{9}$$

Lemma 5 (Li, Li and Wang (2013)): Let $\varepsilon_1 = \lambda_{\min}(EME^T)$, $\varepsilon_q = \lambda_{\max}(EME^T)$, where $E \in \mathcal{R}^{q\times n}$, $M \in \mathcal{R}^{n\times n}$, $M = M^T$, and let $A_1 = D(I - \rho EME^T) + \rho EME^T$, where I is an identity matrix of proper dimensions, $D = \mathrm{diag}(d_1, d_2, ..., d_q)$ with $0 \leq d_i \leq 1$ for $i = 1, 2, ..., q$, $\rho \in \mathcal{R}$, $0 < \rho \leq \frac{2}{\varepsilon_q}$. Then,

$$A_1 + A_1^T \geq \rho\varepsilon_1 I \tag{10}$$

$$x^T(A_1 + A_1^T)x \geq \rho\varepsilon_1 x^T x, \text{ for } \forall x \in \mathcal{R}^n. \tag{11}$$

In addition, $ME^T x = 0$, when $x^T(A_1 + A_1^T)x = 0$.

3. Dual neural networks with a tunable activation function for solving quadratic programming problem

Consider the following quadratic programming problem:

$$\text{minimize} \quad \frac{1}{2}x^T W x + c^T x, \tag{12a}$$

$$\text{subject to} \quad Ax = b, \tag{12b}$$

$$l \leq Ex \leq \mathrm{h}, \tag{12c}$$

where $x \in \mathcal{R}^n$, $W \in \mathcal{R}^{n\times n}$ is a positive matrix, $c \in \mathcal{R}^n$, $A \in \mathcal{R}^{m\times n}$, $b \in \mathcal{R}^m$, $E \in \mathcal{R}^{q\times n}$, $h \in \mathcal{R}^q$, $l \in \mathcal{R}^q$, $m < n$ and $h \geq l$. As in (Liu and Wang (2006)), we assume that the equality constraint is irredundant, that is, $\mathrm{rank}(A) = m$.

According to Karush-Kuhn-Tucker (KKT) conditions (Boyd and Vandenberghe (2004)), we have:

$$Wx + c + A^T\lambda + E^T\mu = 0, \tag{13}$$

$$Ax = b, \tag{14}$$

$$\begin{cases} Ex = h, & \text{if} \quad \mu > 0, \\ l \leq Ex \leq h, & \text{if} \quad \mu = 0, \\ Ex = l, & \text{if} \quad \mu < 0, \end{cases} \tag{15}$$

where $\lambda \in \mathcal{R}^m$ and $\mu \in \mathcal{R}^q$ are dual variables to the equality constraint (12b) and the inequality constraint (12c), respectively. Introducing a saturation function, we have

$$\rho Ex = \mathrm{g}(\rho Ex + \mu), \tag{16}$$

where $\rho \in \mathcal{R}$, $\rho > 0$ is a scaling factor, and the saturation function $\mathrm{g}(x) = [\mathrm{g}_1(x_1), \mathrm{g}_2(x_2), ..., \mathrm{g}_q(x_q)]^T$ is defined as

$$\mathrm{g}_i(x_i) = \begin{cases} \rho h_i, & \text{if} \quad x_i > \rho h_i, \\ x_i, & \text{if} \quad \rho l_i \leq x_i \leq \rho h_i, \\ \rho l_i, & \text{if} \quad x_i < \rho l_i. \end{cases} \tag{17}$$

As in (Li, Li and Wang (2013)), we obtain

$$x = -[W^{-1}E^T - W^{-1}A^T(AW^{-1})^{-1}AW^{-1}E^T]\mu - W^{-1}c + W^{-1}A^T(AW^{-1}A^T)^{-1}(b + AW^{-1}c), \tag{18}$$

$$\lambda = -(AW^{-1}A^T)^{-1}AW^{-1}E^T\mu - (AW^{-1}A^T)^{-1}(b + AW^{-1}c). \tag{19}$$

Define the following constant vector

$$s = W^{-1}A^T(AW^{-1}A^T)^{-1}(b + AW^{-1}c) - W^{-1}c, \tag{20}$$

$$M = W^{-1} - W^{-1}A^T(AW^{-1}A^T)^{-1}AW^{-1}. \tag{21}$$

Then

$$-\rho EME^T\mu + \rho Es = \mathrm{g}\big((I - \rho EME^T)\mu + \rho Es\big). \tag{22}$$

We use the following a layer of dynamic neurons to solve μ in (22)

$$\varepsilon\dot{\mu} = - \mathcal{F}\Big(\mathrm{g}((I-\rho EME^T)\mu+\rho Es) +\rho EME^T\mu-\rho Es\Big), \tag{23}$$

where ε is a scaling positive parameter, $\mathcal{F}(x)$ is a tunable activation function,

$$\mathcal{F}(x) = k_1|x|^r\mathrm{sign}(x)+k_2x, \tag{24}$$

$0 < r < 1$, k_1, k_2 are tunable positive parameters, for $z = [z_1, z_2, ..., z_q]^T$,

$$\mathcal{F}(z) = [\mathcal{F}(z_1), \mathcal{F}(z_2), \cdots, \mathcal{F}(z_q)]^T. \tag{25}$$

The following dual network can be used to solve the programming problem (12),

$$\text{state equation}: \ \varepsilon\dot{\mu} = -\mathcal{F}\Big(\mathrm{g}((I-\rho EME^T)\mu+\rho Es) + \rho EME^T\mu-\rho Es\Big), \tag{26a}$$

$$\text{output equation}: \ x = -ME^T\mu+s, \tag{26b}$$

where $\mathrm{g}(\cdot)$ is given by (17).

The following Lemma is needed for the main results.

Lemma 6 (Li, Li and Wang (2013)): $x^* = -\rho ME^T\mu^* + s$ is the optimal solution to the programming problem (12), where $\mu = \mu^*$ is an equilibrium point of (26).

Now we present the main result.

Theorem 1: With the tunable activation function (24), the neural network (26) is stable in the sense of Lyapunov. When the EME^T has full rank, the neural network converges to an equilibrium point μ^* in finite time and the upper bound of the convergence time T_3 satisfies

$$T_3 \le \frac{2\varepsilon}{\rho\varepsilon_1k_2^2(1-r^2)}\ln\Big[1+\frac{V_0^{1-r}k_2^2(1+r)^{1-r}}{k_1^2}\Big], \tag{27}$$

where $V_0 = \frac{1}{r+1}\|\mathrm{g}((I-\rho EME^T)\mu_0+\rho Es)+\rho EME^T\mu_0-\rho Es\|_{r+1}^{r+1}$ is the initial value of $V(t)$, and $V(t)$ is given as follows:

$$V(t) = \frac{1}{r+1}\Big\|\mathrm{g}((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es\Big\|_{r+1}^{r+1}. \tag{28}$$

Proof: Along the trajectory of (26), the time derivative of $V(t)$ defined by (28) is given by

$$\dot{V}(t) = \dot{\mu}^T\Big(J(I-\rho EME^T)+\rho EME^T\Big)^T\mathcal{F}\Big(\mathrm{g}\Big((I-\rho EME^T)\mu +\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)$$

$$= -\frac{1}{\varepsilon}\Big(\mathcal{F}\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big)^T \Big(J(I-\rho EME^T)+\rho EME^T\Big)^T\mathcal{F}\Big(\mathrm{g}\Big((I-\rho EME^T)\mu +\rho Es\Big)+\rho EME^T\mu-\rho Es\Big), \tag{29}$$

where $J = D^+\mathrm{g}$ is the upper-right Dini derivative of $\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)$. According to (17), we have

$$J = \mathrm{diag}(J_1, J_2, ..., J_n),$$

where

$$J_i = \begin{cases} 1, & \text{if} \quad \rho l_i \le \Big((I-\rho EME^T)\mu+\rho Es\Big)_i \le \rho h_i \\ 0, & \text{if} \quad \Big((I-\rho EME^T)\mu+\rho Es\Big)_i \le \rho l_i \quad \text{or} \\ & \quad \Big((I-\rho EME^T)\mu+\rho Es\Big)_i \ge \rho h_i. \end{cases} \tag{30}$$

From Lemma 4 and Lemma 5, it follows

$$\Big(J(I-\rho EME^T)+\rho EME^T\Big)^T+\Big(J(I-\rho EME^T)+\rho EME^T\Big) \ge \rho\varepsilon_1 I. \tag{31}$$

Bringing (31) into (29), we have

$$\dot{V}(t) \le -\frac{\rho\varepsilon_1}{2\varepsilon}\Big\|\Big(\mathcal{F}\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big)^T\Big\| \Big\|\mathcal{F}\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big\|$$

$$= -\frac{\rho\varepsilon_1}{2\varepsilon}\Big\|\mathcal{F}\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big\|^2$$

$$= -\frac{\rho\varepsilon_1}{2\varepsilon}\Big\|k_1\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)^r +k_2\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big\|^2. \tag{32}$$

Lemma 3 implies that

$$\dot{V}(t) \le -\frac{\rho\varepsilon_1}{2\varepsilon}\Big(k_1^2\Big\|\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big\|_{2r}^{2r} +k_2^2\Big\|\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big\|_2^2\Big)$$

$$\le -\frac{\rho\varepsilon_1}{2\varepsilon}\Big(k_1^2\Big\|\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big\|_{r+1}^{r(r+1)} +k_2^2\Big\|\Big(\mathrm{g}\Big((I-\rho EME^T)\mu+\rho Es\Big)+\rho EME^T\mu-\rho Es\Big)\Big\|_{r+1}^{r+1}\Big)$$

Then, we have

$$\dot{V}(t) \le -\frac{\rho\varepsilon_1}{2\varepsilon}[k_1^2\Big((1+r)V(t)\Big)^r + k_2^2(1+r)V(t)]. \tag{33}$$

By Lemma 2, we can get that $V(t) = 0$ when $t > T_3$. This completes the proof.

In addition, by Lemma 6 we can get the optimal solution to the programming problem in finite time.

If $k_1 = k_2 = 1$, we have the activation function

$$\mathcal{F}(x) = |x|^r \text{sign}(x) + x. \tag{34}$$

We have the following corollary for (26) with the activation function (34).

Corollary 1: The neural network (26) with the activation function (34) is stable in the sense of Lyapunov. In addition, if EME^T has full rank, the neural network converges to an equilibrium point μ^* in finite time and the upper bound of the convergence time T_4 satisfies

$$T_4 \leq \frac{2\varepsilon \ln[1 + V_0^{1-r}(1+r)^{1-r}]}{\rho\varepsilon_1(1-r^2)}, \tag{35}$$

where V_0 is given in Theorem 1.

Proof : Using the same method as Theorem 1, we can obtain the result.

Remark 2: In Theorem 1, the term $k_2 x$ is added to accelerate the convergent speed of the neural network. The parameters k_1, k_2 give more flexibility to solve the quadratic programming problem. By careful section of k_1, k_2, the chattering phenomenon will be avoided. Moreover, the tunable activation function can also decrease the sensitivity to additive noise.

4. Numerical Simulations

In the section, we give a numerical example to illustrative the neural network (26) with the tunable activation function (24). The simulation is performed with the programming language Matlab 7.10.0 on a desktop computer with the Intel (R) Core(TM) G640 Duo CPU at 2.80 GHz, 1.59 GHz and 1.90 GB of RAM. The configuration parameters are given as: variable-step, ode45, relative tolerance 1e-10.

Example: The quadratic programming problem is to

$$\begin{aligned}
\text{minimize} \quad & 3x_1^2 + 3x_2^2 + 4x_3^2 + 5x_4^2 + 3x_1x_2 + \\
& 5x_1x_3 + x_2x_4 - 11x_1 - 5x_4, && \text{(36a)}\\
\text{subject to} \quad & 3x_1 - 3x_2 - 2x_3 + 4x_4 = 0, && \text{(36b)}\\
& 4x_1 + x_2 - x_3 - 24x_4 = 0, && \text{(36c)}\\
& -73 \leq -50x_1 + 50x_2 \leq -50, && \text{(36d)}\\
& -20 \leq 32x_1 + 10x_3 \leq 41. && \text{(36e)}
\end{aligned}$$

For the example, we have

$$W = \begin{bmatrix} 6 & 3 & 5 & 0 \\ 3 & 6 & 0 & 1 \\ 5 & 0 & 8 & 0 \\ 0 & 1 & 0 & 10 \end{bmatrix}, c = \begin{bmatrix} -11 \\ 0 \\ 0 \\ -5 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$A = \begin{bmatrix} 3 & -3 & -2 & 1 \\ 4 & 1 & -1 & -2 \end{bmatrix}, E = \begin{bmatrix} -50 & 50 & 0 & 0 \\ 32 & 0 & 10 & 0 \end{bmatrix},$$

$$l = \begin{bmatrix} -73 \\ -20 \end{bmatrix}, h = \begin{bmatrix} -50 \\ 41 \end{bmatrix}.$$

The largest and smallest eigenvalue of the matrix EME^T are $\varepsilon_q = 138.4337$ and $\varepsilon_1 = 18.5319$, respectively. Moreover, EME^T has full rank, satisfying the conditions of Theorem 1. Then, the neural network (26) converges to the optimal solution in finite time. In addition, we choose $\rho = 0.01 \leq \frac{2}{\varepsilon_q}$, the scaling factor $\varepsilon = 10^{-8}$ in the simulation. We use this example to systematically evaluate the performance of the proposed activation function in three aspects: convergence speed, sensitively to additive noise and robustness against time delay.

(1) Convergence speed

In the part, we compare the convergence speed and the chattering phenomenon with the activation function (24), and the following activation function given in (Li, Li and Wang (2013)):

$$\mathcal{F}(x) = |x|^r \text{sign}(x), \quad 0 < r < 1. \tag{37}$$

The initial value of μ is given by $\mu_0 = \begin{bmatrix} 0.17 \\ 0.09 \end{bmatrix}$. By simple computation, we can obtain the upper bound of T_3 in (27) is 4.5774×10^{-8} with $r = 0.6$, $k_1 = 1$, $k_2 = 1$. The the upper bound of the convergence time T_5 is 14.3147×10^{-8} in (Li, Li and Wang (2013)). Obviously, we have $14.3147 \times 10^{-8} > 4.5774 \times 10^{-8}$. Fig. 5 and Fig. 6 show the results.

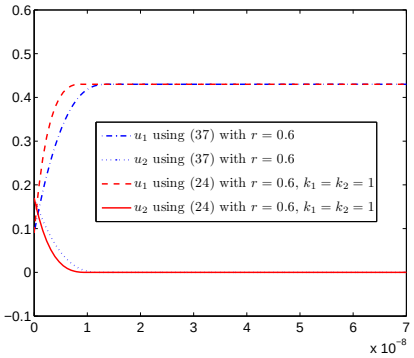

Figure 5: The transient behavior of μ with the activation function (24) and (37)

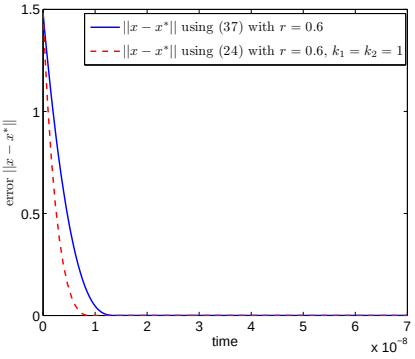

Figure 6: The transient behavior of errors with the activation function (24) and (37)

Although reducing the value of r can speed up the convergent speed of the neural network with the activation function (37), it will present the chattering phenomenon when r is sufficiently close to 0. With the proposed activation function (24), we can make the value of k_2 larger and the value of k_1 smaller to reduce or even eliminate the chattering phenomenon and at the same time to accelerate up the convergent speed. Fig. 7 and Fig. 8 show the simulation results.

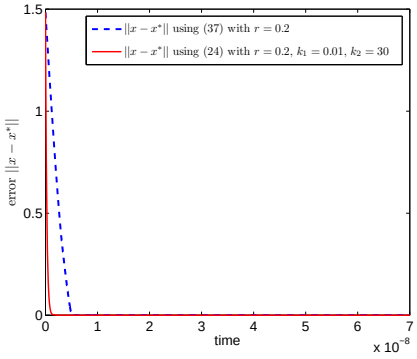

Figure 7: The transient behavior of errors under $r = 0.2$ with the activation function (24) and (37)

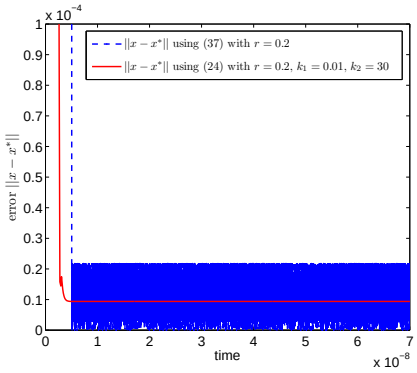

Figure 8: Partial enlarged view of Figure 7

(2) Sensitively to additive noise

In practice, the dynamics of the neural network may been disturbed by noise. In the part, we will compare the sensitivity to additive noise with the activation function (24) and (37), respectively. For simplicity, we only consider the presence of noise in the state equation.

$$\text{state equation}: \ \varepsilon\dot{\mu} = -\mathcal{F}_1(\mathrm{g}((I - \rho EME^T)\mu + \rho Es) + \rho EME^T\mu - \rho Es) + v, \quad (38a)$$

$$\text{output equation}: \ x = -ME^T\mu + s, \quad (38b)$$

where v is zero mean Gaussian white noise with covariance σI.

The simulation results are given in Fig. 9 and Fig. 10.

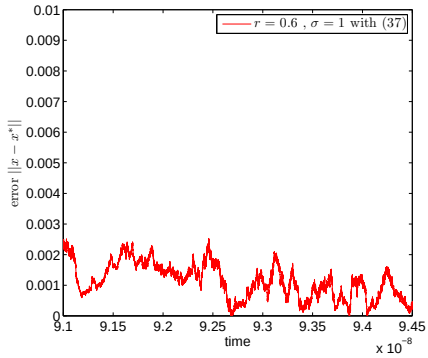

Figure 9: Error under $r = 0.6$ with noise level $\sigma = 1$ by the method in (Li, Li and Wang (2013))

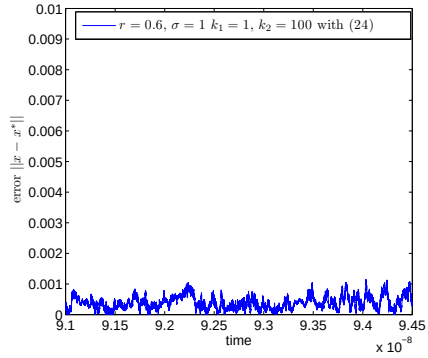

Figure 10: Error under $r = 0.6$ with noise level $\sigma = 1$ by our method

From Fig. 9 and Fig. 10, we can see that the neural network with the tunable activation function (24) is less sensitive to additive noise than the one with (37).

(3) Robustness against time delay

In the above two parts, time delay is not taken into account for the neural network model. But in implementation of the neural network, for example implementation with analog circuits by neural network, time delay is always inevitable due to limited response rate and sometimes it is crucial to the stability of the system. So, the following part, we evaluate the influence of time delay on the neural computing with our tunable activation function and the activation function in (Li, Li and Wang (2013)) under different time delay. We consider the feedback channel of the state equation with time delay:

$$\text{state equation}: \ \varepsilon\dot{\mu}(t) = -\mathcal{F}_1(\mathrm{g}((I - \rho EME^T)\mu(t - \phi) + \rho Es) + \rho EME^T\mu(t - \phi) - \rho Es), \quad (39a)$$

$$\text{output equation}: \ x = -ME^T\mu(t - \phi) + s, \quad (39b)$$

where ϕ is the time delay. As in (Li, Li and Wang (2013)), we use the $\frac{1}{\varepsilon}$ as the time unit τ.

The neural network in (Li, Li and Wang (2013)) will be more possible to oscillate with a smaller r when time delay appears. So making the value of r larger will be helpful to reduce

the oscillate phenomenon. But we know the convergence time will become longer with a larger value of r. We can further decrease the oscillation and obtain a shorter convergent time by carefully section of parameters k_1, k_2. We give the simulations in the following situations, respectively:

Fig. 11: $r = 1$, time delay 1τ;
Fig. 12: $r = 0.8$, time delay 0.2τ;
Fig. 13: $r = 0.2$, time delay 0.008τ;
Fig. 14: $r = 0$, time delay 0.04τ.

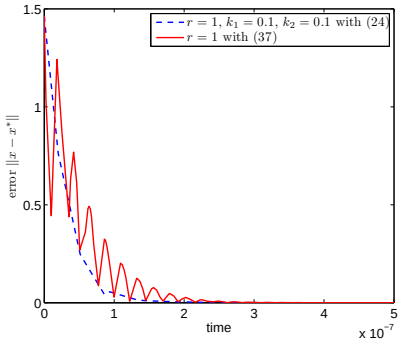

Figure 11: Comparisons of errors under $r = 1$ with the time delay equal to 1τ by the activation functions (24) and (37)

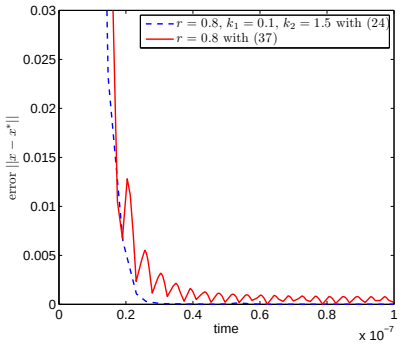

Figure 12: Comparisons of errors under $r = 0.8$ with the time delay equal to 0.2τ by the activation functions (24) and (37)

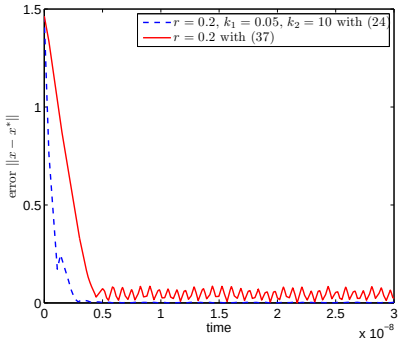

Figure 13: Comparisons of errors under $r = 0.2$ with the time delay equal to 0.008τ by the activation functions (24) and (37)

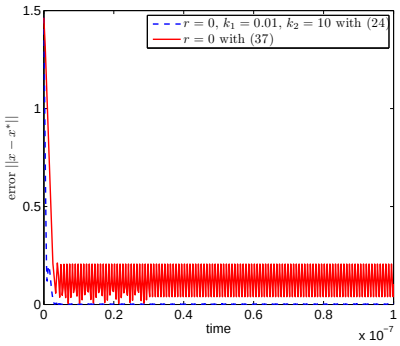

Figure 14: Comparisons of errors under $r = 0$ with the time delay equal to 0.04τ by the activation functions (24) and (37)

5. Conclusion

In the paper, finite time dual neural networks with a new activation function were presented to solve quadratic programming problems. The activation function has two tunable parameters, which give more flexibility to design a neural network. By Lyapunov theorem, the finite-time stability could be derived for the proposed neural networks model, and the actual optimal solutions of the quadratic programming problems could be obtained in finite time interval. Different from the existing recurrent neural networks for solving the quadratic programming problems, the neural networks of this paper have a faster convergent speed, at the same time, reduced oscillation when delay appears, and less sensitivity to the additive noise with careful selection the parameters. The effectiveness of our methods were validated by theoretical analysis and numerical simulations.

6. Acknowledgment

This work was supported by the National Science Foundation of China (61174216, 61273183, 51177088, 61374028), the National Science Foundation of Hubei Provincial (2011CDB187), the Scientific Innovation Team Project of Hubei Provincial Department of Education (T201103).

References

Smith KA (1999). *Neural networks for combinatorial optimization: a review of more than a decade of research.* Informs Journal Computing, 11, 15-34

Li S, Lou Y and Liu B (2012). *Bluetooth aided mobile phone localization: a nonlinear neural circuit approach.* ACM Trans Embedded Comput Syst (accepted)

Lin S, Li Y and Liu B (2012). *Model-free control of lorenz chaos using an approximate optimal control strategy.* Commun Nonlinear Sci Numer Simul, 12(7), 4891-4900

Burrows N and Niranjan M (1994). *The use of recurrent neural networks for classification.* In: IEEE workshop on neural networks for signal processing IV, Scotland, Pages: 117-125

Husken M and Stagge P (2003). *Recurrent neural networks for time series classification.* Neurocomputing, 50, 223-235

Skowronski MD and Harris JG (2007). *Noise-robust automatic speech recognition using a predictive echo state network.* IEEE Trans Audio Speech Lang Process, 15(5), 1724-1730

Li S, Chen S, Liu B, Li Y and Liang Y (2012). *Decentralized kinematic control of a class of collaborayive redundant manipulators via recurrent neural networks.* Neurocomputing, 91, 1-10.

Li S, Chen SF and Liu B (2013). *Acecelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by using a sign-bi-power activation function.* Neural Process Lett, 37, 189-205

Li S, Liu B and Li YM (2013). *Selective positive-negative feedback produces the winner-take-all competition in recurrent neural networks.* IEEE Transactions on Neural Networks and Learning Systems, 24(2), 301-309.

Liu Q and Wang J (2008). *Two k-winners-take-all networks with discontinuous activation functions.* Neural Networks, 21(2-3), 406-413.

Hu X and Zhang B (2009). *A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem.* IEEE Transactions on Neural Networks, 20(4), 654-664.

Zhang Y and Wang J (2002). *A dual neural network for convex quadratic programming subject to linear equality and inequality constraints.* Physics Letters, A, 271-278.

Xia Y and Sun C (2009). *A novel neural dynamical approach to convex quadratic program and its efficient applications.* Neural Networks, 22, 1463-1470.

Xia Y, Feng G and Wang J (2004). *A recurrent neural network with exponential convergence for solving convex quadratic programand related linear piecewise equations.* Neural Networks, 17(7), 1003-1015.

Zhang Y, Wang J and Xia Y(2003). *A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits.* IEEE Transactions on Neural Networks, 14, 658-667.

Hopfield, JJ (1984). *Neurons with graded response have collective computational properties like those of two-state neurons.* Proceedings of the National Academy of Sciences, 81(10), 3088-3092.

Rodriguez-Vazquez A, Dominguez-Castro R, Rueda A, Huertas JL and Sanchez-Sinencio E (1990). *Nonlinear switched capacitor neural networks foroptimization problems.* IEEE Transactions on Circuits and Systems, 37(3),384-398.

Kennedy MP and Chua LO (1988). *Neural networks for nonlinear programming.* IEEE Transactions on Circuits and Systems, 35(5), 554-562.

Wu XY, Xia Y, Li J and Chen WK (1996). *A high-performance neural network for solving linear and quadratic programming problems.* IEEE Transactions on Neural Networks, 7(3), 643-651.

Zhang S and Constantinides AG (1992). *Lagrange programming neural networks.* IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 39(7), 441-452.

Hu X and Zhang B (2009). *A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem.* IEEE Transactions on Neural Networks, 20(4), 654-664.

Liu S and Wang J (2006). *A simplified dual neural network for quadratic programming with its kwta application.* IEEE Transactions on Neural Networks,17(6), 1500-1510.

Wang J (2010). *Analysis and design of a k-winners-take-all model with a single state variable and the heaviside step activation function.* IEEE Transactions on Neural Networks, 21(9), 1496-1506.

Liu Q and Wang J (2011). *Finite-time convergent recurrent neural network with a hard-limiting activation function for constrained optimization with piecewiselinear objective functions.* IEEE Transactions on Neural Networks, 22(4), 601-613.

Li S, Li YM, and Wang Z (2013). *A class of finite-time dual neural networks for solving quadratic programming problems and its k-winners-take-all application.* Neural Networks, 39, 27-39

Bhat S and Bernstein D (2000). *Finite-time stability of continuous autonomous systems.* SIAM Journal of Control and Optimization, 38, 751-766.

Shen YJ and Xia XH (2008). *Semi-global finite-time observers for nonlinear systems.* Automatica, 44, 3152-3156

Shen YJ and Huang YH (2012). *Global finite-time stabilisation for a class of nonlinear systems.* International Journal of Systems Science, 43(1), 73-78

Gao, WB and Hung, JC (1993). *Variable structure control of nonlinear systems: a new approach.* IEEE Transactions on Industrial Electronics, 40, 45-55.

Boyd S and Vandenberghe L (2004). *Convex optimization.* Cambridge UniversityPress.

Hwang SG (2004). *Cauchys interlace theorem for eigenvalues of Hermitian matrices.* American Mathematical Monthly, 157-159.