Neurocomputing 143 (2014) 80–89

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Letters

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

Peng Miao [a], Yanjun Shen [b,*], Xiaohua Xia [c]

[a] College of Science, China Three Gorges University, Yichang, Hubei 443002, China
[b] College of Electrical Engineering and New Energy, China Three Gorges University, Yichang, Hubei 443002, China
[c] Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa

ARTICLE INFO

Article history:
Received 14 December 2013
Received in revised form
17 May 2014
Accepted 4 June 2014
Communicated by Long Cheng
Available online 21 June 2014

Keywords:
Recurrent neural networks
Finite-time stability
Tunable activation function
Quadratic programming problem

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 the neural networks. By Lyapunov theorem, finite-time stability can be derived for the proposed neural networks, 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, they can reduce oscillation when delay appears, and have less sensitivity to additive noise with careful selection of the parameters. Simulations are presented to evaluate the performance of the neural networks with the tunable activation function. In addition, the proposed neural networks are applied to estimate parameters for an energy model of belt conveyors. The effectiveness of our methods are validated by theoretical analysis and numerical simulations.

1. Introduction

Recently, recurrent neural networks have made great development. They are widely applied in scientific and engineering field, for example, optimization [1,2], control of chaos [3], pattern classification [4,5], signal processing [6], robotics [7], solving time-varying Sylvester equation [8], the winners-take-all competition [9–11], etc.

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, recurrent neural networks are designed to solve the constrained optimization problem in [12–14]. However, the designed neural networks only converge to the optimal solution asymptotically and the convergence time is infinite. In order to obtain the accurate solution, in [15] and [16], 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 networks 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 number of neurons in neural networks. Then, research on reduction of neuron number without losing efficiency and accuracy receives some researchers' attention [17–33]. For examples, Zhang investigated and analyzed the performance of gradient neural network applied to time-varying quadratic minimization and quadratic programming problems in [20]. Wang presented a k-winners-take-all (kWTA) neural network and proved its global stability and finite-time convergence in [23]. Bian and Chen proposed a smoothing neural network with a differential equation which could be implemented easily in [32]. There are also some neural network models for solving optimization problems, which can efficiently deal with time delays [34–38]. For instance, Liu and Cao proposed a delayed neural network which could effectively solve a class of linear projection equations and some quadratic programming problems in [34]. In order to deal with the convex optimization problem in finite time, the authors firstly presented a recurrent neural network with a continuous function, $|x|^r \operatorname{sign}(x)(0 < r < 1)$ in [39]. The activation function was also applied to a dual neural network model in [25]. 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, 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

* Corresponding author.
E-mail address: shenyj@ctgu.edu.cn (Y. Shen).

http://dx.doi.org/10.1016/j.neucom.2014.06.018
0925-2312/

additive noise. Therefore, it is worth while to study finite-time dual neural network for solving quadratic programming problems with a relative high robustness against time delay and noise.

In the paper, our main contribution is to present recurrent neural networks with a tunable activation function to solve quadratic programming problems. The tunable activation function is $k_1|x|^r \operatorname{sign}(x)+k_2x$, where k_1, k_2 are tunable positive parameters. These parameters are not only helpful to accelerate convergence speed, but also helpful to improve robustness of the neural networks with appearance of time delay and noise. The motivation comes from the idea of Gao and Hung in [43]. Adding the term $k_2x(t)$ to the activation function is a standard technique in sliding mode control. It can be used to suppress chattering [43].

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:

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

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 [40]*). The equilibrium $x=0$ of (1) is finite-time convergent if there are an open neighborhood $\mathcal{U}$ of the origin and a function $T_x:\mathcal{U}\backslash\{0\}\to(0,\infty)$, such that every solution trajectory $x(t,x_0)$ of (1) starting from the initial point $x_0\in\mathcal{U}\backslash\{0\}$ 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 [40]*). *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*

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

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 [41], Shen and Huang [42]*). *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*

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

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

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

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 Lemmas 1 and 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 $|x|^r\operatorname{sign}(x)$ is very close to $\operatorname{sign}(x)$ for x with small absolute values. Therefore, it may yield chattering phenomenon. For example, consider the following scalar differential equation:

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

The trajectories of (6) with different value of r are given in Fig. 1.

From Fig. 1, 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:

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

Fig. 2 shows the trajectory of (7). From Fig. 2, we can see the chattering phenomenon disappears.

In sliding mode control, the introduction of k_2x, called reaching control, to suppress chattering is the idea of Gao and Hung in [43].

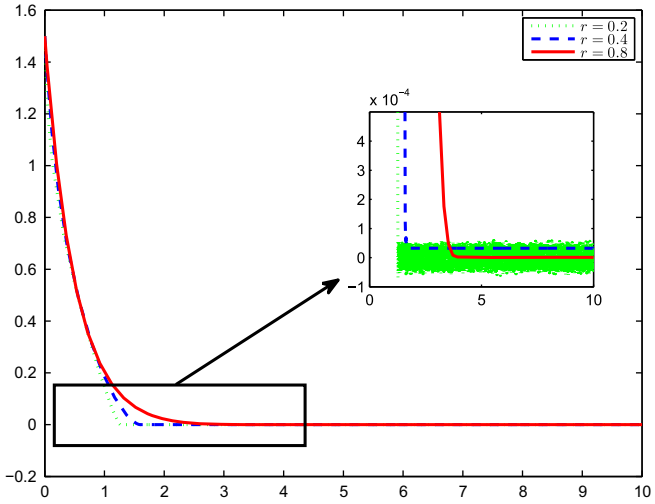

Fig. 1. The trajectories of (6) with different value of r.

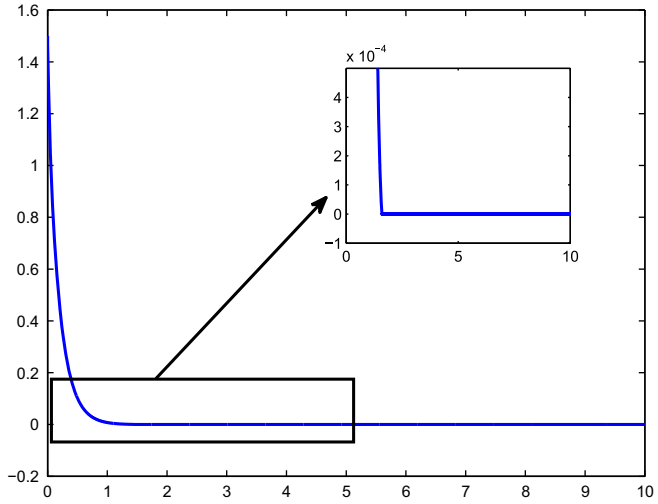

Fig. 2. The trajectory of (7) with $r=0.2$, $k_1=0.0001$, $k_2=15$.

The following Lemmas are also useful for our main results.

Lemma 3 (*Li et al. [25]*). *Let $\|x\|_a$ be the a-norm of $x=[x_1,x_2,\ldots,x_n]^T$, $\|x\|_a=(\sum_{i=1}^n|x_i|^a)^{1/a}$, for $0<b<a$, we have*

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

Lemma 4 (*Hwang [45]*). *Let A be a Hermitian matrix of n, and let B be a principal sub-matrix of A of order $n-1$. If $\lambda_n \le \lambda_{n-1} \le \cdots \le \lambda_2 \le \lambda_1$, lists the eigenvalues of A and $\mu_n \le \mu_{n-1} \le \cdots \le \mu_3 \le \mu_2$ the eigenvalues of B, then*

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

Lemma 5 (*Li et al. [25]*). *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, $D=\mathrm{diag}(d_1,d_2,\ldots,d_q)$ with $0\le d_i\le 1$ for $i=1,2,\ldots,q$, $\rho\in\mathcal{R}$, $0<\rho\le 2/\varepsilon_q$. Then,*

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

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

In addition, $ME^Tx=0$, 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:

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

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

$$l\le Ex\le h, \tag{12c}$$

where $x\in\mathcal{R}^n$, $W\in\mathcal{R}^{n\times n}$ is a positive definite 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 $h\ge l$. As in [22], we assume that the equality constraint is irredundant, that is, $\mathrm{rank}(A)=m$.

According to Karush–Kuhn–Tucker (KKT) conditions [44], we have

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

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

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

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 $\mathrm{g}(x)=[\mathrm{g}_1(x_1),\mathrm{g}_2(x_2),\ldots,\mathrm{g}_q(x_q)]^T$ is defined as

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

As in [25], we obtain

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

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

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

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

In order to solve μ in (22), a layer of dynamic neurons is given as follows

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

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 $z=[z_1,z_2,\ldots,z_q]^T$,

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

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

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

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

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

The following Lemmas are needed for the main results.

Lemma 6. *There exists a solution of the neural network (26) on $[0,\infty)$.*

Proof. Because (17) and (24) are continuous functions on $[0,\infty)$, then, $\mathcal{F}(\mathrm{g}((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)$ is also continuous on $[0,\infty)$. Therefore, there exists a solution of the neural network (26) on $[0,\infty)$ [46]. □

Lemma 7 (*Li et al. [25]*). *$x^*=-\rho ME^T\mu^*+s$ is the optimal solution of 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 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*

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

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

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

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

$$\begin{aligned} \dot{V}(t)=&\dot{\mu}^T(J(I-\rho EME^T)+\rho EME^T)^T\mathcal{F}(\mathrm{g}((I-\rho EME^T)\mu\\ &+\rho Es)+\rho EME^T\mu-\rho Es)\\ =&-\frac{1}{\varepsilon}(\mathcal{F}(\mathrm{g}((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es))^T\\ &(J(I-\rho EME^T)+\rho EME^T)^T\mathcal{F}(\mathrm{g}((I-\rho EME^T)\mu\\ &+\rho Es)+\rho EME^T\mu-\rho Es), \end{aligned} \tag{29}$$

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

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

where

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

From Lemmas 4 and 5, it follows that

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

Bringing (31) into (29), we have

$$\begin{aligned}\dot V(t)&\le-\frac{\rho\varepsilon_1}{2\varepsilon}\|(\mathcal{F}(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es))^T\|\\&\quad\|\mathcal{F}(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)\|\\&=-\frac{\rho\varepsilon_1}{2\varepsilon}\|\mathcal{F}(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)\|^2\\&=-\frac{\rho\varepsilon_1}{2\varepsilon}\|k_1(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)^r\\&\quad+k_2(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)\|^2.\end{aligned}\tag{32}$$

Lemma 3 implies that

$$\begin{aligned}\dot V(t)&\le-\frac{\rho\varepsilon_1}{2\varepsilon}(k_1^2\|(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)\|_{2r}^{2r}\\&\quad+k_2^2\|(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)\|_2^2)\\&\le-\frac{\rho\varepsilon_1}{2\varepsilon}(k_1^2\|(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)\|_{r+1}^{r(r+1)}\\&\quad+k_2^2\|(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)\|_{r+1}^{r+1})\end{aligned}$$

Then, we have

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

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

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

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

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

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*

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

where V_0 is given in Theorem 1.

Table 1
Comparisons of (26) with other neural network models by solving problem (12).

Neural network models	*Theoretical error*	*Convergent time*	*Number of neurons*
[14]	Non-zero	Infinite	n
[16]	Zero	Infinite	n+m+4q
[17]	Zero	Infinite	n
[22]	Zero	Infinite	q
[26]	Zero	Infinite	m+q
[21]	Zero	Finite	n+2q
[39]	Zero	Finite	n
[25]	Zero	Finite	q
(26)	Zero	Finite	q

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

Remark 2. In Theorem 1, the term k_2x 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, chattering phenomenon will be avoided and the convergent speed can be accelerated. Moreover, the tunable activation function can also decrease the sensitivity to additive noise. In practice, when r is close to 1, we can select $k_1=1$, $k_2=1$. When r is close to 0, we can select $k_1<0.05$ and $k_2\ge1$.

Remark 3. $\mathcal{F}(g((I-\rho EME^T)\mu+\rho Es)+\rho EME^T\mu-\rho Es)$ in (23) is continuous everywhere and locally Lipschitz everywhere except on $u=u^*$. Hence forward uniqueness for all initial conditions except the equilibrium point follows from [47].

Remark 4. There exist several other neural network models which can be used to solve quadratic programming problem (12) [14,16,17,22,26,21,39,25]. In Table 1, we summarize the comparisons of these neural network models with our model (26).

4. Numerical simulations and application

In the section, we present two numerical examples to illustrate the efficiency of the neural network (26) with the tunable activation function (24): online calculation of a quadratic programming problem and parameters estimation for an energy model of belt conveyors.

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−5.

Example 1. The quadratic programming problem is to

$$\text{minimize}\quad 3x_1^2+3x_2^2+4x_3^2+5x_4^2+3x_1x_2+5x_1x_3+x_2x_4-11x_1-5x_4,\tag{36a}$$

$$\text{subject to}\quad 3x_1-3x_2-2x_3+4x_4=0,\tag{36b}$$

$$4x_1+x_2-x_3-24x_4=0,\tag{36c}$$

$$-73\le-50x_1+50x_2\le-50,\tag{36d}$$

$$-20\le32x_1+10x_3\le41.\tag{36e}$$

For the example, we have

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

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

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

The largest and smallest eigenvalues of the matrix EME^T are $\varepsilon_q=138.4337$ and $\varepsilon_1=18.5319$, respectively. Moreover, EME^T has full rank and satisfies the conditions of Theorem 1. Then, the neural network (26) converges to the optimal solution in finite time. In addition, we choose $\rho=0.01\le2/\varepsilon_q$, the scaling factor $\varepsilon=10^{-8}$ in the simulation.

In the following, we will 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 this part, we compare the convergence speed and the chattering phenomenon with the activation function (24), and the following activation function given in [25]:

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

Let T_5 and t_5 denote the upper bound of the convergence time and the actual convergence time of the neural network model [25], respectively, and T_3 and t_3 denote the upper bound of the convergence time and the actual convergence time of our model (26), respectively. The initial value of μ is given by $\mu_0 = [^{0.17}_{0.09}]$. By simple computation, we can obtain that the upper bound T_3 and the actual convergence time t_3 are 4.5774×10^{-8} and 0.83×10^{-8} s, respectively, with r=0.6, $k_1 = 1$, $k_2 = 1$. The upper bound T_5 and the actual convergence time t_5 are 14.3147×10^{-8} and 1.1×10^{-8} s, respectively. Obviously, we have $14.3147 \times 10^{-8} > 4.5774 \times 10^{-8}$ and $1.1 \times 10^{-8} > 0.83 \times 10^{-8}$. The simulation results are shown in Figs. 3 and 4.

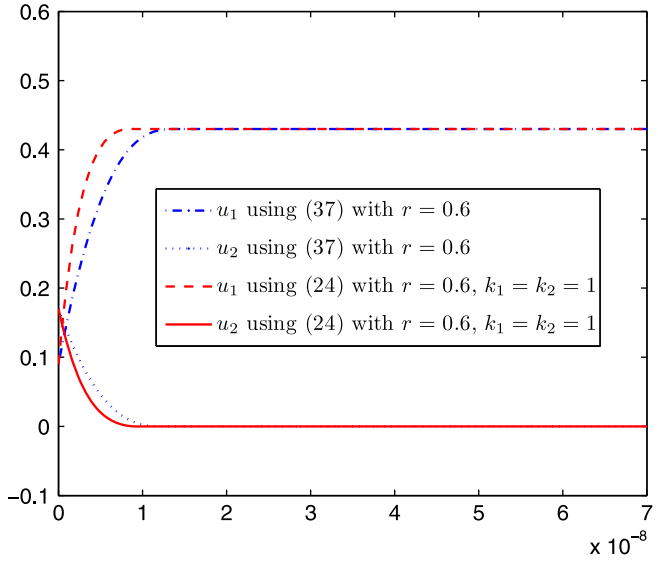

Fig. 3. The transient behavior of μ with the activation functions (24) and (37).

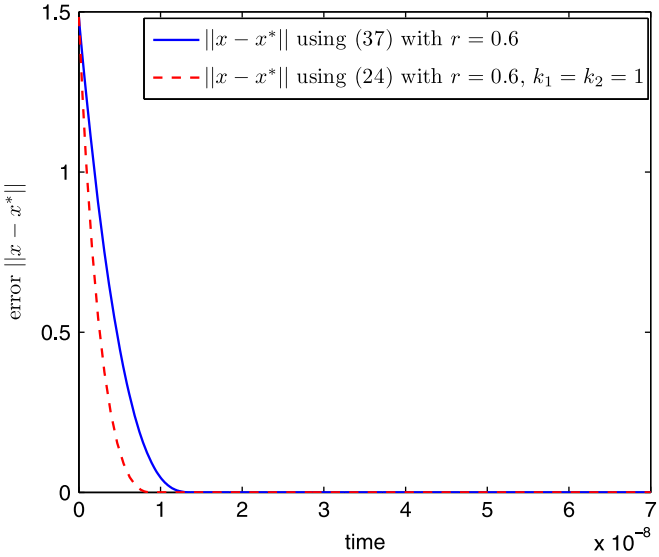

Fig. 4. The transient behavior of errors with the activation functions (24) and (37).

In order to eliminate the effect of random initialization, we compare the performance of the neural network model [25] and our model (26) with ten independently generated initializations of uniform distribution on the interval [0, 1]. The unit of time is 10^{-8} s. Table 2 shows the simulation results.

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 very 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 the convergent speed. Figs. 5 and 6 show the simulation results.

(2) *Sensitivity 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 functions (24) and (37), respectively. For simplicity, we only consider the presence of noise in the state equation:

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

$$\text{Output equation}: x = -ME^T\mu + s, \tag{38b}$$

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

Table 2
Comparisons the upper bound of the convergence time and the actual convergence time of the neural network model [25] and our model (r=0.5, $k_1 = 1$, $k_2 = 1$).

Number of random initialization	μ_0	T_5	t_5	T_3	t_3
1	[0.81;0.16]	11.01	1.12	4.71	0.91
2	[0.91;0.97]	6.74	2.51	2.51	1.92
3	[0.13;0.96]	16.60	2.21	7.54	1.62
4	[0.91;0.49]	9.91	1.83	4.13	1.31
5	[0.63;0.80]	8.09	2.21	3.19	1.62
6	[0.10;0.14]	11.66	0.96	5.04	0.71
7	[0.28;0.42]	10.99	1.28	4.69	1.01
8	[0.55;0.92]	10.78	2.42	4.58	1.72
9	[0.96;0.79]	8.24	2.31	3.27	1.70
10	[0.96;0.93]	7.53	2.50	2.90	1.87
Mean		10.16	1.94	4.26	1.44
Var		8.02	0.36	2.10	0.18

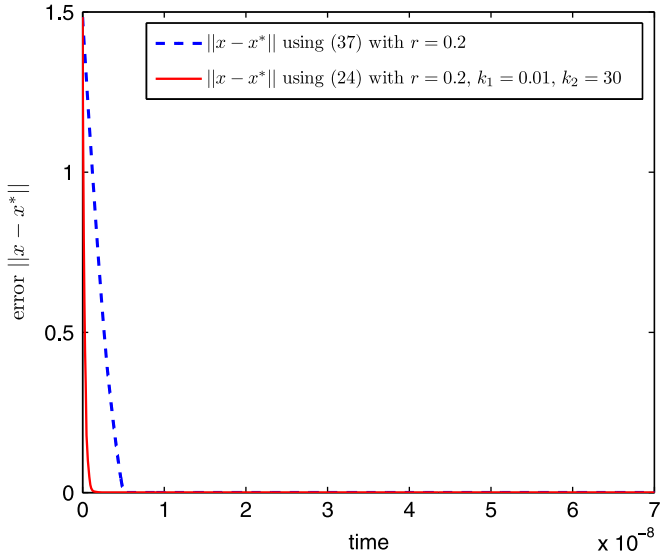

Fig. 5. The transient behavior of errors under r=0.2 with the activation functions (24) and (37).

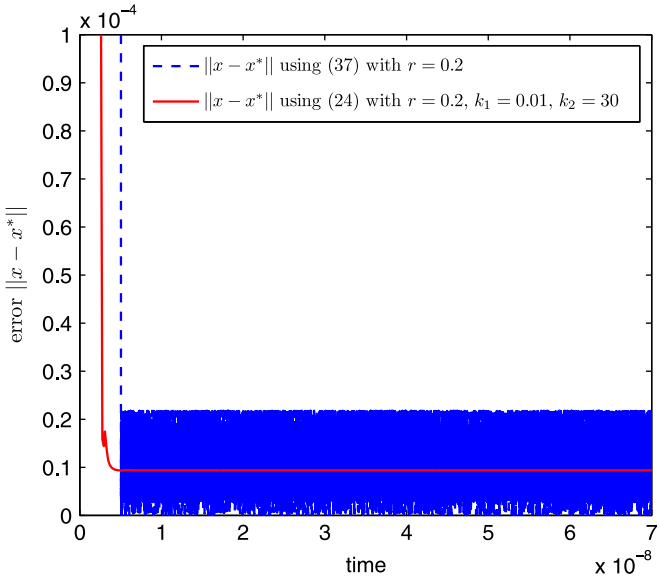

Fig. 6. Partial enlarged view of Fig. 5.

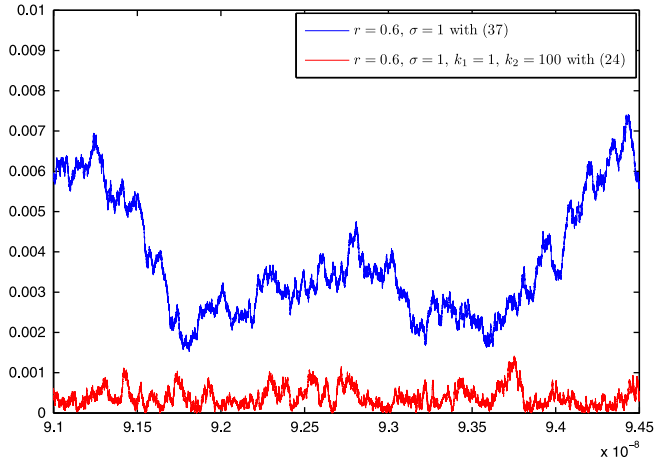

Fig. 7. Error under $r=0.6$ with noise level $\sigma=1$ by the method in [25] and our method.

The simulation results are given in Fig. 7. From Fig. 7, 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 [25] under different time delay. We consider the feedback channel of the state equation with time delay:

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

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

where is the time delay. As in [25], we use the $1/\varepsilon$ as the time unit τ.

The neural network in [25] will be more likely 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:

Fig. 8: $r=1$, time delay 1τ;
Fig. 9: $r=0.8$, time delay 0.2τ;
Fig. 10: $r=0.2$, time delay 0.008τ;
Fig. 11: $r=0$, time delay 0.04τ.

Example 2. Parameters estimation for an energy model of belt conveyors.

Belt conveyors are widely used to transfer bulk material in mining, metallurgical and coal industry because of high transfer capacity and long transfer distance. Fig. 12 shows a typical belt conveyor.

As in [48–50], the mechanical power of a belt conveyor P_T is given by

$$P_T = F_U V, \tag{40}$$

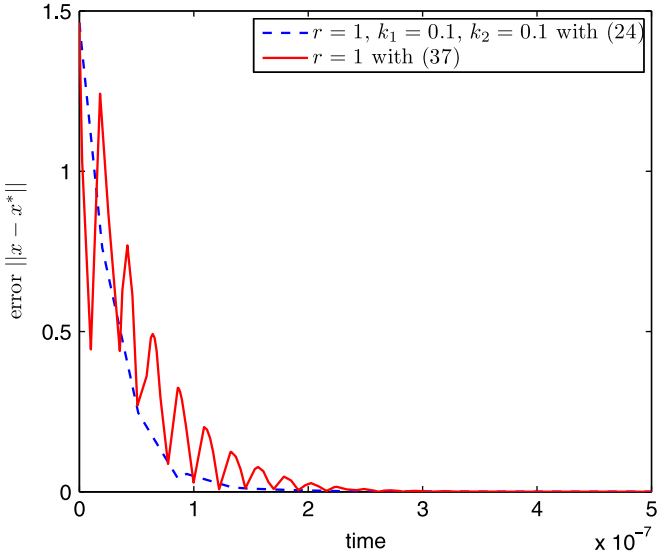

Fig. 8. Comparisons of errors under $r=1$ with the time delay equal to 1τ by the activation functions (24) and (37).

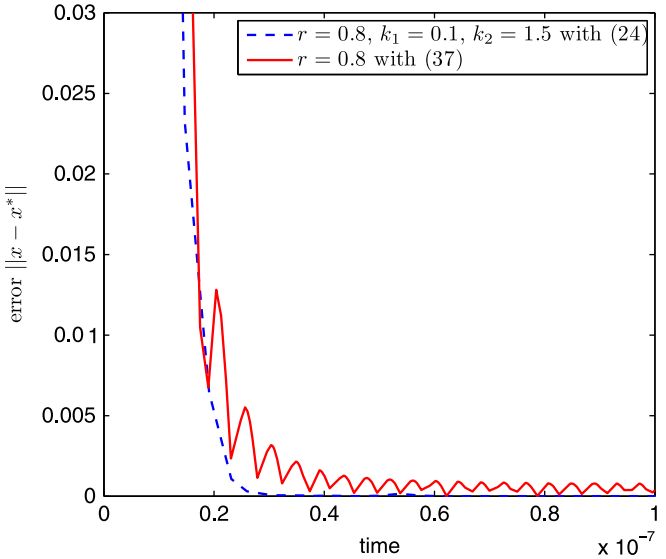

Fig. 9. Comparisons of errors under $r=0.8$ with the time delay equal to 0.2τ by the activation functions (24) and (37).

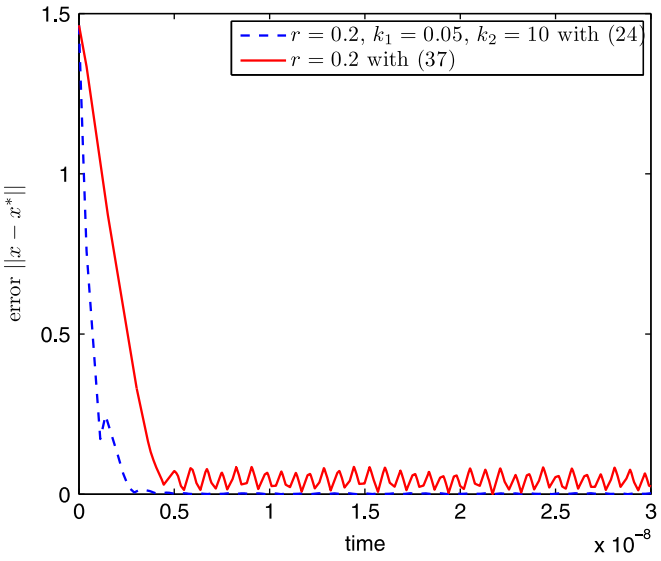

Fig. 10. Comparisons of errors under $r=0.2$ with the time delay equal to 0.008τ by the activation functions (24) and (37).

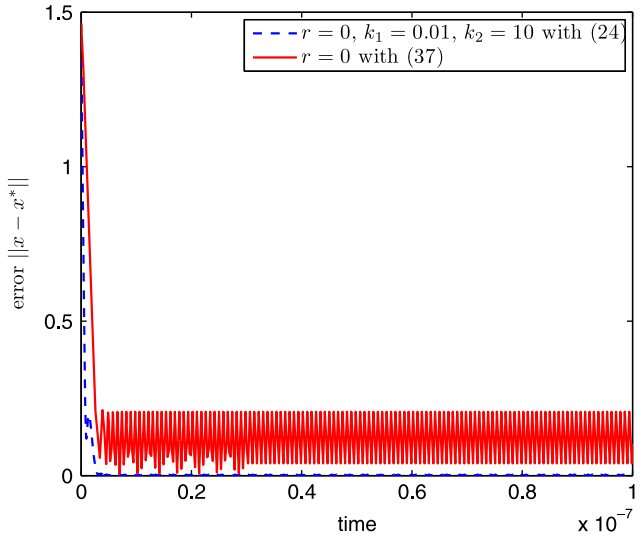

Fig. 11. Comparisons of errors under $r=0$ with the time delay equal to 0.04τ by the activation functions (24) and (37).

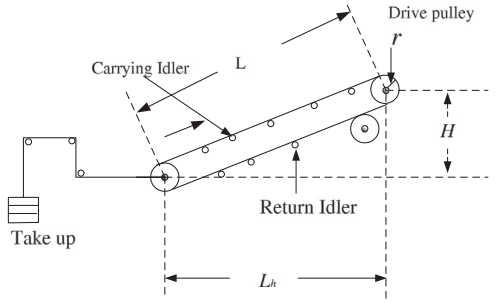

Fig. 12. Typical profile of belt conveyors.

and

$$F_U = F_H + F_N + F_S + F_{st}, \tag{41}$$

where F_H, F_N, F_S, and F_{st} are the main resistance, the secondary resistance, the slop resistance, and the special resistance, and can be expressed as

$$\begin{cases} F_H = fLg[Q_{RO} + Q_{RU} + (2Q_B + Q_G)\cos\ \delta], \\ F_N = \dfrac{TV}{3.6} + \dfrac{T^2}{6.48\rho b_1^2} + C_{Ft}, \\ F_S = K_1\dfrac{T^2}{V^2} + K_2\dfrac{T}{V} + K_3, \\ F_{st} = Q_G Hg, \end{cases} \tag{42}$$

where V is the belt speed (m/s), T is the feed rate (t/h), $Q_G = T/3.6V$, K_1, K_2 and K_3 are constant coefficients related to the structural parameters of the belt conveyor, C_{Ft} is a constant, Q_{RO}, Q_{RU}, Q_B, δ, f, L, H, ρ, b_1 and g are the unit mass of the rotating parts of carrying idler rollers (kg/m), the unit mass of rotating parts of the return idler rollers (kg/m), the unit mass of the belt (kg/m), the unit mass of the load (kg/m), the inclination angle (°), the artificial friction factor, the center-to-center distance (m), the net change in elevation (m), the bulk density of material (kg/m^3), the width between the skirt boards (m) and the gravitational acceleration (9.8 m/s^2).

Let

$$\begin{cases} \theta_1 = \dfrac{1}{6.48\rho b_1^2}, \\ \theta_2 = gf(Q_{RO} + Q_{RU} + 2Q_B)[L\ \cos\ \delta \\ \qquad + L(1 - \cos\ \delta)\left(1 - \dfrac{2Q_B}{Q_{RO} + Q_{RU} + 2Q_B}\right)] + K_3 + C_{Ft}, \\ \theta_3 = K_1, \\ \theta_4 = \dfrac{gL\ \sin\ \delta + gfL\ \cos\ \delta}{3.6} + K_2. \end{cases} \tag{43}$$

From (40)–(43), we can obtain

$$P_T = \theta_1 T^2 V + \theta_2 V + \theta_3\frac{T^2}{V} + \theta_4 T + \frac{V^2 T}{3.6}. \tag{44}$$

The parameter vector $\Theta = [\theta_1,\ \theta_2,\ \theta_3,\ \theta_4]^T$ is determined by the structural parameters and components of a belt conveyor, by the operation circumstance and by the characteristic of the material handled. It is relatively a constant vector for a certain belt conveyor. In [49,50], $\Theta = [2.3733 \times 10^{-4},\ 8.5663 \times 10^3,\ 0.0031,\ 51.6804]^T$.

If we set the values of V and T, then the value of P_T can been obtained by (44). Therefore, we obtain the data, $\{P_T(i), V(i), T(i), i = 1, \ldots, N\}$, where N is sampling number. Using the data, we can apply the proposed neural network to solve the approximate solution of Θ by the following method:

$$\text{minimize} \quad \sum_{i=1}^{n}\left(P_T(i) - L(i)\hat{\Theta} - \frac{V^2(i)T(i)}{3.6}\right)^2, \tag{45a}$$

$$\text{subject to} \quad l \le \hat{\Theta} \le h, \tag{45b}$$

where $L(i) = [T^2(i)V(i),\ V(i),\ (T^2(i))/(V(i)),\ T(i)]$. Note that the problem (45) can been transformed into the following quadratic programming problem:

$$\text{minimize} \quad \hat{\Theta}^T\sum_{i=1}^{n}L^T(i)L(i)\hat{\Theta} + \sum_{i=1}^{n}\left(P_T(i) - \frac{V^2(i)T(i)}{3.6}\right)^2 - 2\sum_{i=1}^{n}\left(P_T(i) - \frac{V^2(i)T(i)}{3.6}\right)L(i)\hat{\Theta}, \tag{46a}$$

$$\text{subject to} \quad l \le \hat{\Theta} \le h. \tag{46b}$$

Then, we have $W = 2\sum_{i=1}^{n}L^T(i)L(i)$, $c = -2\sum_{i=1}^{n}(P_T(i) - (V^2(i)T(i))/3.6)L(i)$, $E = I_{4\times 4}$, $l = [0\ 10^2\ 0\ 10]^T$, $h = [1\ 10^4\ 1\ 100]^T$, $s = -W^{-1}c$, and $M = W^{-1}$ (By properly selecting $V(i)$ and $T(i)$, we can ensure that W is positive definite, and EME^T has full rank). Therefore, we

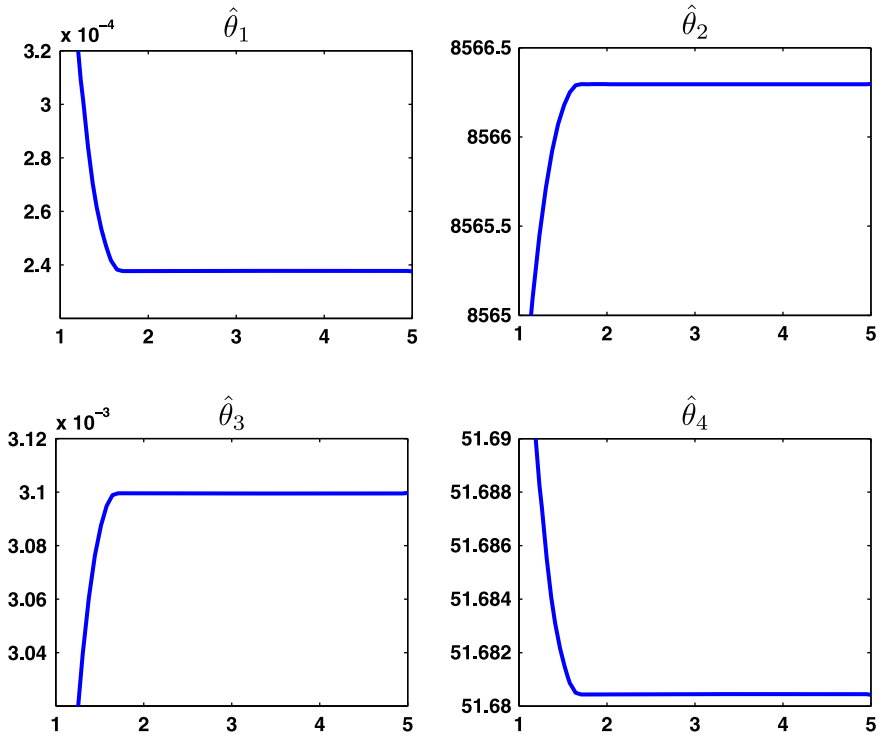

Fig. 13. Transient behavior of $\hat{\Theta}$ ($k_1 = k_2 = 1$, $r = 0.5$).

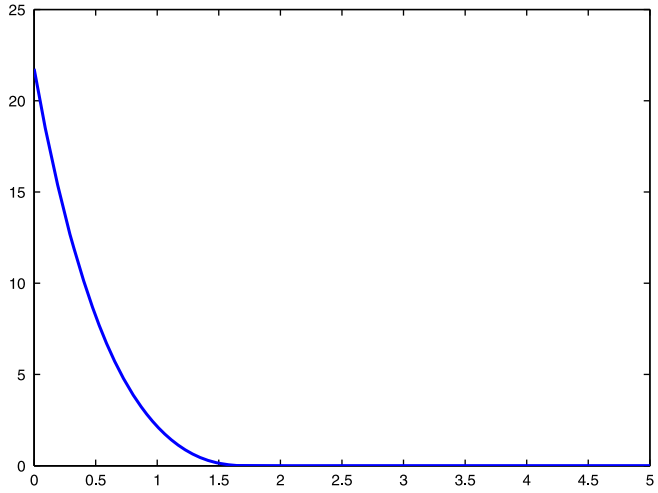

Fig. 14. Transient behavior of $\|\hat{\Theta} - \Theta\|$ ($k_1 = k_2 = 1$, $r = 0.5$).

can use the neural network (26) to solve the quadratic programming problem (46).

In the simulation, we select $[V(1), V(2), V(3), V(4), V(5)] = [1, 3, 1.2, 5, 2]$, $[T(1), T(2), T(3), T(4), T(5)] = [300, 100, 100, 100, 100]$. Then, we have

$$W = \begin{bmatrix} 2.409 \times 10^{10} & 9.688 \times 10^{5} & 1.700 \times 10^{10} & 7.640 \times 10^{7} \\ 9.688 \times 10^{5} & 8.088 \times 10 & 2.600 \times 10^{5} & 2.840 \times 10^{3} \\ 1.700 \times 10^{10} & 2.600 \times 10^{5} & 1.642 \times 10^{10} & 5.773 \times 10^{7} \\ 7.640 \times 10^{7} & 2.840 \times 10^{3} & 5.773 \times 10^{7} & 2.600 \times 10^{5} \end{bmatrix},$$

$$EME^T = \begin{bmatrix} 8.653 \times 10^{-8} & -0.001 & -1.059 \times 10^{-7} & 9.238 \times 10^{-6} \\ -0.001 & 12.118 & 0.001 & -0.112 \\ -1.059 \times 10^{-7} & 0.001 & 1.308 \times 10^{-7} & -1.167 \times 10^{-5} \\ 9.238 \times 10^{-6} & -0.112 & -1.167 \times 10^{-5} & 0.001 \end{bmatrix}.$$

The eigenvalues of the matrix W are 0.0825, 1.4392×10^4, 2.8265×10^9, 3.7681×10^{10} and the eigenvalues of the matrix EME^T are 2.6538×10^{-11}, 3.5380×10^{-10}, 6.9482×10^{-5}, 12.1188. Therefore, the matrix W is positive definite and the matrix EME^T has full rank. The largest eigenvalue of the matrix EME^T is $\varepsilon_q = 12.1188$. We choose $\rho = 0.16 \le 2/\varepsilon_q$, $\varepsilon = 1$. By implementation of the neural network, the output of (26) is $\hat{\Theta} = [2.37330935 \times 10^{-4}, 8.566299\,98 \times 10^3, 0.00309999, 51.68040010]$. Fig. 13 shows the results. The transient behavior of $\|\hat{\Theta} - \Theta\|$ is shown in Fig. 14 as well.

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, finite-time stability could be derived for the proposed neural networks, 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, it could reduce oscillation when delay appears, and have less sensitivity to the additive noise with careful selection the parameters. The effectiveness of our methods were validated by theoretical analysis and numerical simulations.

Acknowledgments

This work was supported by the National Science Foundation of China (61374028, 61174216 and 51177088), the Grant National Science Foundation of Hubei Provincial (2013CFA050), the

Scientific Innovation Team Project of Hubei Provincial Department of Education (T201103), the Graduate Scientific Research Foundation of China Three Gorges University (2014PY069).

References

[1] K.A. Smith, Neural networks for combinatorial optimization: a review of more than a decade of research, Inf. J. Comput. 11 (1999) 15–34.
[2] S. Li, Y. Lou, B. Liu, Bluetooth aided mobile phone localization: a nonlinear neural circuit approach, ACM Trans. Embedded Comput. Syst. January (2013).
[3] S. Lin, Y. Li, B. Liu, Model-free control of Lorenz chaos using an approximate optimal control strategy, Commun. Nonlinear Sci. Numer. Simul. 12 (7) (2012) 4891–4900.
[4] N. Burrows, M. Niranjan, The use of recurrent neural networks for classification, in: IEEE Workshop on Neural Networks for Signal Processing IV, Scotland, 1994, pp. 117–125.
[5] M. Husken, P. Stagge, Recurrent neural networks for time series classification, Neurocomputing 50 (2003) 223–235.
[6] M.D. Skowronski, J.G. Harris, Noise-robust automatic speech recognition using a predictive echo state network, IEEE Trans. Audio Speech Lang. Process. 15 (5) (2007) 1724–1730.
[7] S. Li, S. Chen, B. Liu, Y. Li, Y. Liang, Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks, Neurocomputing 91 (2012) 1–10.
[8] S. Li, S.F. Chen, B. Liu, Accelerating 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 (2013) 189–205.
[9] S. Li, B. Liu, Y.M. Li, Selective positive-negative feedback produces the winner-take-all competition in recurrent neural networks, IEEE Trans. Neural Netw. Learn. Syst. 24 (2) (2013) 301–309.
[10] Q. Liu, J. Wang, Two k-winners-take-all networks with discontinuous activation functions, Neural Netw. 21 (2–3) (2008) 406–413.
[11] X. Hu, B. Zhang, A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem, IEEE Trans. Neural Netw. 20 (4) (2009) 654–664.
[12] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, Proc. Natl. Acad. Sci. 81 (10) (1984) 3088–3092.
[13] A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, J.L. Huertas, E. Sanchez-Sinencio, Nonlinear switched capacitor neural networks for optimization problems, IEEE Trans. Circuits Syst. 37 (3) (1990) 384–398.
[14] M.P. Kennedy, L.O. Chua, Neural networks for nonlinear programming, IEEE Trans. Circuits Syst. 35 (5) (1988) 554–562.
[15] X.Y. Wu, Y. Xia, J. Li, W.K. Chen, A high-performance neural network for solving linear and quadratic programming problems, IEEE Trans. Neural Netw. 7 (3) (1996) 643–651.
[16] S. Zhang, A.G. Constantinides, Lagrange programming neural networks, IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process. 39 (7) (1992) 441–452.
[17] Q.S. Liu, J.D. Cau, Global exponential stability of discrete-time recurrent neural network for solving quadratic programming problems subject to linear constraints, Neurocomputing 74 (2011) 3494–3501.
[18] X.L. Hu, Applications of the general projection neural network in solving extended linear-quadratic programming problems with linear constraints, Neurocomputing 72 (2009) 1131–1137.
[19] Y. Tan, C. Deng, Solving for a quadratic programming with a quadratic constraint based on a neural network frame, Neurocomputing 30 (2000) 117–128.
[20] Y.N. Zhang, Y.W. Yang, G.Q. Ruan, Performance analysis of gradient neural network exploited for online time-varying quadratic minimization and equality-constrained quadratic programming, Neurocomputing 74 (2011) 1710–1719.
[21] X. Hu, B. Zhang, A new recurrent neural network for solving convex quadratic programming problems with an application to the k-winners-take-all problem, IEEE Trans. Neural Netw. 20 (4) (2009) 654–664.
[22] S. Liu, J. Wang, A simplified dual neural network for quadratic programming with its kwta application, IEEE Trans. Neural Netw. 17 (6) (2006) 1500–1510.
[23] J. Wang, Analysis and design of a k-winners-take-all model with a single state variable and the heaviside step activation function, IEEE Trans. Neural Netw. 21 (9) (2010) 1496–1506.
[24] Q. Liu, J. Wang, Finite-time convergent recurrent neural network with a hard-limiting activation function for constrained optimization with piecewise linear objective functions, IEEE Trans. Neural Netw. 22 (4) (2011) 601–613.
[25] S. Li, Y.M. Li, Z. Wang, A class of finite-time dual neural networks for solving quadratic programming problems and its k-winners-take-all application, Neural Netw. 39 (2013) 27–39.
[26] Y. Zhang, J. Wang, A dual neural network for convex quadratic programming subject to linear equality and inequality constraints, Phys. Lett. A (2002) 271–278.
[27] Y. Xia, C. Sun, A novel neural dynamical approach to convex quadratic program and its efficient applications, Neural Netw. 22 (2009) 1463–1470.
[28] Y. Xia, G. Feng, J. Wang, A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations, Neural Netw. 17 (7) (2004) 1003–1015.
[29] Y. Zhang, J. Wang, Y. Xia, A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits, IEEE Trans. Neural Netw. 14 (2003) 658–667.
[30] Q.S. Liu, J. Wang, A one-layer recurrent neural network for constrained nonsmooth optimization, IEEE Trans. Circuits Syst. 41 (5) (2011) 1323–1333.
[31] L. Cheng, Z.G. Hou, Y.Z. Lin, M. Tan, W.J. Chris Zhang, F.X. Wu, Recurrent neural network for non-smooth convex optimization problems with application to the identification of genetic regulatory networks, IEEE Trans. Neural Netw. 21 (5) (2011) 714–726.
[32] W. Bian, X.J. Chen, Smoothing neural network for constrained non-Lipschitz optimization with applications, IEEE Trans. Neural Netw. Learn. Syst. 23 (3) (2012) 399–411.
[33] Q. S Liu, J. Wang, A one-layer projection neural network for nonsmooth optimization subject to linear equalities and bound constraints, IEEE Trans. Neural Netw. Learn. Syst. 24 (5) (2013) 812–824.
[34] Q.S. Liu, J.D. Cao, Y.S. Xia, A delayed neural network for solving linear projection equations and its analysis, IEEE Trans. Neural Netw. 16 (4) (2005) 834–843.
[35] Y.Q. Yang, J.D. Cao, Solving quadratic programming problems by delayed projection neural network, IEEE Trans. Neural Netw. 17 (6) (2006) 1630–1634.
[36] L. Cheng, Z.G. Hou, M. Tan, A neural-type delayed projection neural network for solving nonlinear variational inequalities, IEEE Trans. Circuits Syst. II: Express Briefs 55 (8) (2008) 806–810.
[37] B.N. Huang, G.T. Hui, D.W. Gong, Z.S. Wang, X.P. Meng, A projection neural network with mixed delays for solving linear variational inequality, Neurocomputing 125 (2014) 28–32.
[38] L. Cheng, Z.G. Hou, M. Tan, A delayed projection neural network for solving linear variational inequalities, IEEE Trans. Neural Netw. 20 (6) (2009) 915–925.
[39] L. Cheng, Z.G. Hou, Noriyasu Homma, M. Tan, Madam M. Gupta, Solving convex optimization problems using recurrent neural networks in finite time, in: Proceedings of the 2009 International Joint Conference on Neural Networks, 2009, pp. 538–543.
[40] S. Bhat, D. Bernstein, Finite-time stability of continuous autonomous systems, SIAM J. Control Optim. 38 (2000) 751–766.
[41] Y.J. Shen, X.H. Xia, Semi-global finite-time observers for nonlinear systems, Automatica 44 (2008) 3152–3156.
[42] Y.J. Shen, Y.H. Huang, Global finite-time stabilisation for a class of nonlinear systems, Int. J. Syst. Sci. 43 (1) (2012) 73–78.
[43] W.B. Gao, J.C. Hung, Variable structure control of nonlinear systems: a new approach, IEEE Trans. Ind. Electron. 40 (1993) 45–55.
[44] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, 2004.
[45] S.G. Hwang, Cauchys interlace theorem for eigenvalues of Hermitian matrices, Am. Math. Mon. (2004) 157–159.
[46] J.K. Hale, Ordinary differential equations, Pure and Applied Mathematics XXI, second ed., Krieger, Malabar, FL, 1980.
[47] P. Bhat, S. Bernstein, Geometric homogeneity with applications to finite-time stability, Math. Control Signals Syst. 17 (2005) 101–127.
[48] Y.J. Shen, X.H. Xia, Adaptive parameter estimation for an energy model of belt conveyor with DC motor, Asian J. Control 16 (3) (2014) 1–11.
[49] S.R. Zhang, X.H. Xia, Modeling and energy efficiency optimization of belt conveyors, Appl. Energy 88 (9) (2011) 3061–3071.
[50] S.R. Zhang, X.H. Xia, A new energy calculation models of belt conveyors, IEEE AFRICON (2009) 1–6.

Peng Miao received the bachelor's degree from the Department of Mathematics at the Normal University of Nanyang of China, in 2012. Now, he is a postgraduate student in the College of Science, China Three Gorges University. His research interests include nonlinear systems and neural networks.

Yanjun Shen received the bachelor's degree from the Department of Mathematics at the Normal University of Huazhong of China, in 1992, the master's degree from the Department of Mathematics at Wuhan University, in 2001, and the Ph.D. degree in the Department of Control and Engineering at Huazhong University of Science and Technology in 2004. Now he is currently a professor in the College of Electrical Engineering and New Energy, China Three Gorges University. His research interests include robust control, nonlinear systems and neural networks.

Xiaohua Xia is a professor in the Electrical, Electronic and Computer Engineering at the University of Pretoria, South Africa, director of the Centre of New Energy Systems, and the director of the National Hub for the Postgraduate Programme in Energy Efficiency and Demand-side Management. He was academically affiliated with the University of Stuttgart, Germany, the Ecole Centrale de Nantes, France, and the National University of Singapore before joining the University of Pretoria in 1998. He is a fellow of the Institute for Electronic and Electrical Engineers (IEEE), a fellow of the South African Academy of Engineering (SAAE), a member of the Academy of Science of South Africa (ASSAF), and he has an A rating from the South African National Research Foundation (NRF). He has been an associate editor of Automatica, IEEE Transactions on Circuits and Systems II, IEEE Transactions on Automatic Control, and specialist editor (control) of the SAIEE Africa Research Journal. His research interests are modeling and control of nonlinear systems, energy systems and optimization.