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

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, $x^r \text{sign}(x)(0 < r < 1)$ 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
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 \( k_1|x|^r \text{sign}(x) + k_2x \), where \( k_1, k_2 \) are tunable positive parameters. These parameters are not only helpful to accelerate convergence speed, but also helpful to improve 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:
\[
x(t) = f(x(t)), \quad f(0) = 0, \quad x \in \mathbb{R}^n, \quad x(0) = x_0,
\]
where \( f: \mathcal{D} \rightarrow \mathbb{R}^n \) 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 \( T_2: \mathcal{U} \rightarrow (0, \infty) \), such that every solution trajectory \( x(t, x_0) \) of (1) starting from the initial point \( x_0 \in \mathcal{U} \) is well-defined and unique in forward time for \( t \in [0, T_2(x_0)) \), and \( \lim_{t \to T_2(x_0)} x(t, x_0) = 0 \). Then, \( T_2(x_0) \) 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 \( \mathcal{U} = \mathcal{D} = \mathbb{R}^n \), 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 \( \mathcal{U} \subset \mathbb{R}^n \) of the origin, and real numbers \( k_1 > 0 \) and \( 0 < r < 1 \), such that
\[
V(x)|_{x=1} \leq -k_1 V(x)^r, \quad \forall x \in \mathcal{U}.
\]
Then, the origin of the system (1) is locally finite-time stable. The convergence time \( T_1 \) is, depending on the initial state \( x_0 \), satisfies
\[
T_1(x_0) \leq \frac{V(x_0)^{1-r}}{k_1(1-r)}
\]
for all \( x_0 \in \mathcal{U} \). Further, if \( \mathcal{U} = \mathbb{R}^n \) and \( V(x) \) is radially unbounded (that is \( V(x) \rightarrow +\infty \) as \( \|x\| \rightarrow +\infty \)), 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 \( \mathcal{U} \subset \mathbb{R}^n \) of the origin, and real numbers \( k_1, k_2 > 0 \) and \( 0 < r < 1 \), such that
\[
V(x)|_{x=1} \leq -k_1 V(x)^r - k_2 V(x), \quad \forall x \in \mathcal{U}.
\]
Then, the origin of system (1) is finite-time stable. The convergence time \( T_2 \) satisfies
\[
T_2(x_0) \leq \frac{\ln \left( 1 + \frac{k_2 V(x_0)^{1-r}}{k_1 \mathcal{U}} \right)}{k_2(1-r)}
\]
for all \( x_0 \in \mathcal{U} \). If \( \mathcal{U} = \mathbb{R}^n \) 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 \( |x|^r \text{sign}(x) \) is very close to \( \text{sign}(x) \) for \( x \) with small absolute values. Therefore, it may yield chattering phenomenon. For example, consider the following scalar differential equation:
\[
x(t) = -|x(t)|^r \text{sign}(x(t)), \quad 0 < r < 1.
\]
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:
\[
x(t) = -k_1|x(t)|^r \text{sign}(x(t)) - k_2x(t), \quad 0 < r < 1.
\]
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.](image1)

![Fig. 2. The trajectory of (7) with r=0.2, k_1 = 0.0001, k_2 = 15.](image2)
The following Lemmas are also useful for our main results.

**Lemma 3** (Li et al. [25]). Let $\|x\|_b$ 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 $\|x\|_a \leq \|x\|_b$, (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 $\lambda_n \leq \lambda_{n-1} \leq \ldots \leq \lambda_2 \leq \lambda_1$, lists the eigenvalues of $A$ and $\mu_n \leq \mu_{n-1} \leq \ldots \leq \mu_2 \leq \mu_1$ the eigenvalues of $B$, then $\lambda_n \leq \mu_n \leq \lambda_{n-1} \leq \ldots \leq \lambda_2 \leq \mu_2 \leq \lambda_1$. (9)

**Lemma 5** (Li et al. [25]). Let $\epsilon_1 = \lambda_{\min}(EMET)$, $\epsilon_2 = \lambda_{\max}(EMET)$, where $E \in \mathbb{R}^{q \times q}$, $M \in \mathbb{R}^{n \times n}$, $M = M^T$, and let $A_1 = D(l - \rho EMET) + \rho EMET$, where $D$ is an identity matrix of proper dimensions, $D = \text{diag}(d_1, d_2, \ldots, d_q)$, with $0 \leq d_i \leq 1$ for $i = 1, 2, \ldots, q$, $\rho \in \mathbb{R}$, $0 < \rho \leq \epsilon_2 / \epsilon_1$. Then, $A_1 + A_1^T \geq \rho \epsilon_1 I$ (10)

$x^T(A_1 + A_1^T)x \geq \rho \epsilon_1 \|x\|_2^2$ for $\forall x \in \mathbb{R}^n$. (11)

In addition, $\text{ME}^T x = 0$, 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:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} x^T W x + c^T x, \\
\text{subject to} & \quad Ax = b, \\
& \quad l \leq Ex \leq h,
\end{align*}
\]

(12a)

where $x \in \mathbb{R}^n$, $W \in \mathbb{R}^{n \times n}$ is a positive definite matrix, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $E \in \mathbb{R}^{q \times n}$, $h \in \mathbb{R}^q$, $l \in \mathbb{R}^q$, $m < n$ and $h \geq l$. As in [22], we assume that the equality constraint is irredundant, that is, rank($A$) = $m$.

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

\[
\begin{align*}
Ax = b, \\
\begin{cases}
Ex = h & \text{if} \quad \mu > 0, \\
l \leq Ex \leq h & \text{if} \quad \mu = 0, \\
Ex = l & \text{if} \quad \mu < 0,
\end{cases}
\end{align*}
\]

(14)

(15)

where $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^q$ are dual variables to the equality constraint (12b) and the inequality constraint (12c), respectively. Introducing a saturation function, we have $\rho \mathbf{Ex} = g(\mathbf{Ex} + \mu)$, (16)

where $\rho \in \mathbb{R}$, $\rho > 0$ is a scaling factor, and the saturation function $g(x) = g_1(x_1), g_2(x_2), \ldots, g_n(x_n)^T$ is defined as

\[
\begin{align*}
g_i(x_i) = \begin{cases}
\rho_i x_i & \text{if} \quad x_i > \rho_i, \\
x_i & \text{if} \quad \rho_i \leq x_i \leq \rho_i, \\
\rho_i & \text{if} \quad x_i < \rho_i.
\end{cases}
\end{align*}
\]

(17)

As in [25], we obtain

\[
\begin{align*}
x = -(W^{-1}E)^T - W^{-1}A^T(AW^{-1})^{-1}AW^{-1}E^T \mu - W^{-1}c
\quad + W^{-1}A^T(AM^{-1}A)^{-1}(b + AW^{-1}c), \\
\lambda = -(AW^{-1}A)^{-1}AW^{-1}E^T \mu - (AW^{-1}A)^{-1}(b + AW^{-1}c).
\end{align*}
\]

(18)

(19)

Define the following constant vector:

\[
s = W^{-1}A^T(AM^{-1}A)^{-1}(b + AW^{-1}c),
\]

(20)

\[
M = W^{-1} - W^{-1}A^T(AM^{-1}A)^{-1}AW^{-1}.
\]

(21)

Then

\[
-\rho EMET \mu + \rho \mathbf{Es} = g(l - \rho EMET \mu + \rho \mathbf{Es}).
\]

(22)

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

\[
\mathbf{e} \mu = -F(g(l - \rho EMET \mu + \rho \mathbf{Es}) + \rho EMET \mu - \rho \mathbf{Es}),
\]

(23)

where $\epsilon$ is a scaling positive parameter, $F(x)$ is a tunable activation function,

\[
F(z) = [F(z_1), F(z_2), \ldots, F(z_q)]^T.
\]

(25)

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

State equation: \[\mathbf{e} \mu = -F(g(l - \rho EMET \mu + \rho \mathbf{Es}) + \rho EMET \mu - \rho \mathbf{Es}),\]

(26a)

Output equation: \[x = -ME^T \mu + s,\]

(26b)

where $g(\cdot)$ 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 $F(g(l - \rho EMET \mu + \rho \mathbf{Es}) + \rho EMET \mu - \rho \mathbf{Es})$ 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]). $x^* = -\rho EMET \mu^* + s$ 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 $EMET$ has full rank, the neural network converges to an equilibrium point $\mu^*$ in finite time and the upper bound of the convergence time $T_s$ satisfies

\[
T_s \leq \frac{2\epsilon}{\rho^2 k_1^2 (1 - r^2)} \ln \left[ 1 + \frac{V_0^{1-r} k_2^2 (1 + r)^{-1}}{k_1^2} \right],
\]

(27)

where $V_0 = \frac{1}{(1 + r)^2} \|g(l - \rho EMET \mu_0 + \rho \mathbf{Es}) + \rho EMET \mu_0 - \rho \mathbf{Es}\|_{l+1}^{-1}$ is the initial value of $V(t)$, and $V(t)$ is given as follows:

\[
V(t) = \frac{1}{r + 1} \|g(l - \rho EMET \mu + \rho \mathbf{Es}) + \rho EMET \mu - \rho \mathbf{Es}\|_{l+1}^{-1}.
\]

(28)

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

\[
\dot{V}(t) = \mu^T (l - \rho EMET) + \rho EMET \mu^T F(g(l - \rho EMET) \mu + \rho \mathbf{Es}) + \rho EMET \mu - \rho \mathbf{Es})
\]

\[
= -\frac{1}{\epsilon} \frac{1}{F(g(l - \rho EMET \mu + \rho \mathbf{Es}) + \rho EMET \mu - \rho \mathbf{Es})^T}
\]

\[
(l - \rho EMET \mu + \rho \mathbf{Es})^T F(g(l - \rho EMET) \mu + \rho \mathbf{Es}) \rho EMET \mu - \rho \mathbf{Es}),
\]

(29)
Then, we have $T_{\text{function}}(34)$. is stable in the sense of Lyapunov. In addition to Lemma 4 and 5, it follows that $J(t) = \rho E(t)^2 + J(t) + (l - \rho E(t))^2 + \rho E(t)^2 \geq \rho E(t)^2$.

Bringing (31) into (29), we have

$$V(t) = \frac{-DE}{2e} \|F(g((l - \rho E(t))^2 + \rho E(t)^2)^2)\| + \frac{-DE}{2e} \|F(g((l - \rho E(t))^2 + \rho E(t)^2)^2)\|^2$$

$$= -DE \|F(g((l - \rho E(t))^2 + \rho E(t)^2)^2)\| + DE \|F(g((l - \rho E(t))^2 + \rho E(t)^2)^2)\|^2$$

$$= -DE \|k_1(g((l - \rho E(t))^2 + \rho E(t)^2)^2) + k_2(g((l - \rho E(t))^2 + \rho E(t)^2)^2)^2 + k_3$$

Then, we have

$$V(t) \leq \frac{-DE}{2e} \|k_1(g((l - \rho E(t))^2 + \rho E(t)^2)^2)\|^2$$

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

$$F(x) = \|x\| \text{sign}(x) + x.$$ (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 $\mu^*$ in finite time and the upper bound of the convergence time $T_4$ satisfies

$$T_4 \leq \frac{2 \ln(1 + V_{\text{in}}(1 + \frac{1}{n})^{-1}}{\rho E_V(1 - r)}.$$ (35)

where $V_0$ is given in Theorem 1.

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

<table>
<thead>
<tr>
<th>Neural network models</th>
<th>Theoretical error</th>
<th>Convergent time</th>
<th>Number of neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14]</td>
<td>Non-zero</td>
<td>Infinite</td>
<td>n</td>
</tr>
<tr>
<td>[16]</td>
<td>Zero</td>
<td>Infinite</td>
<td>n + m + 4q</td>
</tr>
<tr>
<td>[17]</td>
<td>Zero</td>
<td>Infinite</td>
<td>n</td>
</tr>
<tr>
<td>[22]</td>
<td>Zero</td>
<td>Infinite</td>
<td>q</td>
</tr>
<tr>
<td>[26]</td>
<td>Zero</td>
<td>Infinite</td>
<td>m + q</td>
</tr>
<tr>
<td>[21]</td>
<td>Zero</td>
<td>Finite</td>
<td>n + 2q</td>
</tr>
<tr>
<td>[9]</td>
<td>Zero</td>
<td>Finite</td>
<td>n</td>
</tr>
<tr>
<td>[25]</td>
<td>Zero</td>
<td>Finite</td>
<td>q</td>
</tr>
<tr>
<td>[26]</td>
<td>Zero</td>
<td>Finite</td>
<td>q</td>
</tr>
</tbody>
</table>

The largest and smallest eigenvalues of the matrix $EME^T$ are $\eta = 138.4337$ and $\xi = 18.5319$, 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 $\rho = 0.01 \leq 2/\xi$, the scaling factor $\varepsilon = 10^{-3}$ in the simulation.

In the following, we will use this example to systematically evaluate the performance of the proposed activation function in

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

**Remark 2.** In Theorem 1, the term $k_2$ 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 $k_2 \geq 1$.

**Remark 3.** $F(g((l - \rho E(t))^2 + \rho E(t)^2)^2)$ 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 this 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) i5-5080 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 minimize $3x_1^2 + 3x_2^2 + 4x_3^2 + 5x_4^2 + 3x_1x_2 + 5x_1x_3 + 2x_4 - 11x_1 - 5x_4$.

subject to $x_1 - 3x_2 - 2x_3 + 4x_4 = 0$, $4x_1 + x_2 - x_3 - 24x_4 = 0$, $-73 \leq -50x_1 + 50x_3 \leq -50$, $-20 \leq 32x_1 + 10x_3 \leq 41$.

For the example, we have

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

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

$$I = \begin{bmatrix} -73 \\ -20 \end{bmatrix}, \quad h = \begin{bmatrix} -50 \\ -41 \end{bmatrix}.$$
three aspects: convergence speed, sensitivity 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]:

\[
F(x) = |x|^r \text{sign}(x), \quad 0 < r < 1.
\]  

(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 \( \mu \) 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 \times 10^{-8} \text{ s} and 0.83 \times 10^{-8} \text{ 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 \times 10^{-8} \text{ s} and 1.1 \times 10^{-8} \text{ s}, respectively. Obviously, we have 14.3147 \times 10^{-8} > 4.5774 \times 10^{-8} > 0.83 \times 10^{-8}. The simulation results are shown in Figs. 3 and 4.

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} \text{ 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 be disturbed by noise. In this 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:

State equation:

\[
\epsilon \dot{\mu} = -F_1(g(I - \rho \text{EMET} \mu + \rho \text{Es}) + \rho \text{EMET} \mu - \rho \text{Es}) + v,
\]  

(38a)

Output equation:

\[
x = -\text{MET} \mu + s,
\]  

(38b)

where \( v \) is zero mean Gaussian white noise with covariance \( \sigma I \).

Table 2

<table>
<thead>
<tr>
<th>Number of random initialization</th>
<th>( \mu_0 )</th>
<th>( T_5 )</th>
<th>( t_5 )</th>
<th>( T_3 )</th>
<th>( t_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0.81;0.16]</td>
<td>11.01</td>
<td>1.12</td>
<td>4.71</td>
<td>0.91</td>
</tr>
<tr>
<td>2</td>
<td>[0.91;0.97]</td>
<td>6.74</td>
<td>2.51</td>
<td>2.51</td>
<td>1.92</td>
</tr>
<tr>
<td>3</td>
<td>[0.13;0.96]</td>
<td>16.60</td>
<td>2.21</td>
<td>7.54</td>
<td>1.62</td>
</tr>
<tr>
<td>4</td>
<td>[0.91;0.49]</td>
<td>9.91</td>
<td>1.83</td>
<td>4.13</td>
<td>1.31</td>
</tr>
<tr>
<td>5</td>
<td>[0.63;0.80]</td>
<td>8.09</td>
<td>2.21</td>
<td>3.19</td>
<td>1.62</td>
</tr>
<tr>
<td>6</td>
<td>[0.10;0.14]</td>
<td>11.66</td>
<td>0.96</td>
<td>5.04</td>
<td>0.71</td>
</tr>
<tr>
<td>7</td>
<td>[0.26;0.42]</td>
<td>10.99</td>
<td>1.28</td>
<td>4.69</td>
<td>1.01</td>
</tr>
<tr>
<td>8</td>
<td>[0.55;0.92]</td>
<td>10.78</td>
<td>2.42</td>
<td>4.58</td>
<td>1.72</td>
</tr>
<tr>
<td>9</td>
<td>[0.96;0.79]</td>
<td>8.24</td>
<td>2.31</td>
<td>3.27</td>
<td>1.70</td>
</tr>
<tr>
<td>10</td>
<td>[0.96;0.93]</td>
<td>7.53</td>
<td>2.50</td>
<td>2.90</td>
<td>1.87</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>10.16</td>
<td>1.94</td>
<td>4.26</td>
<td>1.44</td>
</tr>
<tr>
<td>Var</td>
<td></td>
<td>8.02</td>
<td>0.36</td>
<td>2.10</td>
<td>0.18</td>
</tr>
</tbody>
</table>

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

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

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

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:

State equation: \[ \varepsilon \dot{\mu}(t) = -F_1(g(t-\rho E^T\mu(t-\tau)) \mu(t-\tau) + \rho Es) + \rho E^T \mu(t-\tau) - \rho Es, \] (39a)

Output equation: \[ x = -ME^T \mu(t-\tau) + s, \] (39b)

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

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 1\( \tau \);
- Fig. 9: \( r = 0.8 \), time delay 0.2\( \tau \);
- Fig. 10: \( r = 0.2 \), time delay 0.008\( \tau \);
- Fig. 11: \( r = 0 \), time delay 0.04\( \tau \).

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

\[ P_T = F_U V, \] (40)
where $V$ is the belt speed (m/s), $T$ is the feed rate (t/h), $Q_c = T/3.6V$, $K_1$, $K_2$ and $K_0$ are constant coefficients related to the structural parameters of the belt conveyor, $C_F$ is a constant, $Q_{ro}$, $Q_{ru}$, $Q_{bo}$, $Q_{bu}$, $\delta$, $f$, $L$, $H$, $\rho$, $b_1$ 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 ($^\circ$), 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

$$\theta_1 = \frac{1}{6.48 \rho b_1},$$

$$\theta_2 = gf(Q_{ro} + Q_{ru} + 2Q_B) L \cos \delta + L(1 - \cos \delta) \left(1 - \frac{2Q_B}{Q_{ro} + Q_{ru} + 2Q_B}\right) + K_3 + C_F,$$

$$\theta_3 = K_1,$$

$$\theta_4 = \frac{g L \sin \delta + g f L \cos \delta}{3.6} + K_2.$$  

From (40)–(43), we can obtain

$$P_T = \theta_1 T^3 V + \theta_2 T^2 V + \theta_3 T + \frac{V^2 T}{3.6}.$$  

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 be 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 $\Theta$ by the following method:

$$\begin{aligned}
\min & \sum_{i=1}^{n} \left( P_T(i) - L(i) \hat{\Theta} - \frac{V^2(i) T(i)}{3.6} \right)^2, \\
\text{subject to} & \quad \hat{\Theta}^T \sum_{i=1}^{n} L(i)^T L(i) \hat{\Theta} + \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right)^2 - 2 \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right) L(i) \hat{\Theta}, \\
\text{subject to} & \quad \hat{\Theta}^T \leq \hat{h}, \quad -2 \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right) L(i) \hat{\Theta},
\end{aligned}$$

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

$$\begin{aligned}
\min & \quad \hat{\Theta}^T \sum_{i=1}^{n} L(i)^T L(i) \hat{\Theta} + \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right)^2 - 2 \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right) L(i) \hat{\Theta}, \\
\text{subject to} & \quad \hat{\Theta}^T \leq \hat{h}, \quad -2 \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right) L(i) \hat{\Theta},
\end{aligned}$$

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

$$\begin{aligned}
\min & \quad \hat{\Theta}^T \sum_{i=1}^{n} L(i)^T L(i) \hat{\Theta} + \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right)^2 - 2 \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right) L(i) \hat{\Theta}, \\
\text{subject to} & \quad \hat{\Theta}^T \leq \hat{h}, \quad -2 \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right) L(i) \hat{\Theta},
\end{aligned}$$

Then, we have $W = 2 \sum_{i=1}^{n} L(i)^T L(i), c = -2 \sum_{i=1}^{n} \left( P_T(i) - \frac{V^2(i) T(i)}{3.6} \right) L(i), E = \hat{h} \hat{h}^T, I = [10^2 \ 0 \ 0 ^10^2], s = -W^{-1} c,$ and $M = W^{-1}$. (By properly selecting $V(i)$ and $T(i)$, we can ensure that $W$ is positive definite, and $EMEM^T$ has full rank). Therefore, we
can use the neural network (26) to solve the quadratic programming problem (46).

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

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

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

The eigenvalues of the matrix \(W\) are \(0.0825, 1.4392 \times 10^{4}, 2.8265 \times 10^{9}, 3.7681 \times 10^{10}\) and the eigenvalues of the matrix \(EME^T\) are \(2.6538 \times 10^{-11}, 3.5380 \times 10^{-10}, 6.9482 \times 10^{-5}, 12.1188\). Therefore, the matrix \(W\) is positive definite and the matrix \(EME^T\) has full rank. The largest eigenvalue of the matrix \(EME^T\) is \(\epsilon_q = 12.1188\). We choose \(\rho = 0.16 \leq 2/\epsilon_q\), \(\epsilon = 1\). By implementation of the neural network, the output of (26) is \(\hat{q} = [2.37330935 \times 10^{-4}, 8.566299 \times 10^{4}, 0.00309999, 51.68040010]\). 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 (2013CF050), the
References

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.