Neural Comput & Applic (2015) 26:693–703
DOI 10.1007/s00521-014-1744-4

ORIGINAL ARTICLE

# Solving time-varying quadratic programs based on finite-time Zhang neural networks and their application to robot tracking

**Peng Miao · Yanjun Shen · Yuehua Huang · Yan-Wu Wang**

Received: 8 June 2014 / Accepted: 22 September 2014 / Published online: 18 October 2014


**Abstract** In this paper, finite-time Zhang neural networks (ZNNs) are designed to solve time-varying quadratic program (QP) problems and applied to robot tracking. Firstly, finite-time criteria and upper bounds of the convergent time are reviewed. Secondly, finite-time ZNNs with two tunable activation functions are proposed and applied to solve the time-varying QP problems. Finite-time convergent theorems of the proposed neural networks are presented and proved. The upper bounds of the convergent time are estimated less conservatively. The proposed neural networks also have superior robustness performance against perturbation with large implementation errors. Thirdly, feasibility and superiority of our method are shown by numerical simulations. At last, the proposed neural networks are applied to robot tracking. Simulation results also show the effectiveness of the proposed methods.

**Keywords** Time-varying QP problems · Finite-time ZNN · Tunable activation function · Upper bound of convergent time · Robot tracking

P. Miao
College of science, China Three Gorges University, Yichang 443002, China

Y. Shen (✉) · Y. Huang
Hubei Provincial Collaborative Innovation Center for New Energy Microgrid, China Three Gorges University, Yichang 443002, China
e-mail: shenyj@ctgu.edu.cn

Y.-W. Wang
School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China

## 1 Introduction

Quadratic program (QP) problems are widely applied in science and engineering fields, such as, $k$-winner-take-all problem [1–4], optimal controller design [5, 6], digital signal processing [7], robot-arm motion planning [8–12] and so on. In some cases of their application areas, it is needed to solve time-varying QP problems. There are many algorithms or methods to solve the time-varying QP problems. Numerical methods can be applied to linear-equality constraint, while sequential quadratic programming (SQP) is widely used to solve nonlinear programming (NLP) problems [13–15]. In addition, neural networks also play an important role in solving the time-varying QP problems. They possess the characteristics of parallel-distributed nature and hardware-realization convenience and receive considerable studies in many scientific and engineering fields. For instance, Zhang neural network (ZNN) has been proposed to solve the time-varying QP problems [8, 16–20]. It is shown that the exact solutions can be achieved when time goes to infinity. In [8], Zhang and Yang presented a special type of ZNN model with a power-sum activation function. It has better robustness against large implementation errors than the one with linear function. However, the proposed ZNN converges to the actual solution after infinitely long time. In order to accelerate ZNN to finite-time convergence, activation functions, $\text{sign}(x)|x|^r$ with $0<r<1$ and $\text{sign}(x)(|x|^r+|x|^{1/r})$ with $0<r<1$, are used to solve QP problem in [21] and to solve complex time-varying Sylvester equation in [22], respectively. More recently, ZNN with a tunable activation function is proposed to solve QP problems [23]. It is shown ZNN with the tunable activation function has a faster convergent speed and less sensitivity to additive noise [23].

Based on the ideas in [8, 21–26], we design ZNNs with two tunable activation functions to solve the time-varying QP problems. The exact solutions of these problems can be derived in finite time. Moreover, ZNNs with these tunable activation functions have superior robustness performance against perturbation with large implementation errors than the one proposed in [8]. At the same time, we can estimate the upper bounds of the convergence time with less conservation.

Our paper is organized as follows. In Sect. 2, we give some preliminaries about finite-time stability and estimation of the upper bounds of the convergence time. In Sect. 3, time-varying QP problems are formulated and solved by ZNNs with two tunable activation functions. Sect. 4 gives robust analysis of the proposed neural networks and shows that the proposed neural networks have the superior robust performance against perturbation with large implementation errors than the model in [8]. In Sect. 5, feasibility and superiority of our method are validated by numerical simulations. In Sect. 6, our neural networks are applied to robot tracking. The superiority is also shown by computer simulation. At last, Sect. 7 concludes this 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** [27] The equilibrium $x = 0$ of (1) is finite-time convergent if there is 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.

We give the following sufficient conditions such that the system (1) is finite-time stable.

**Lemma 1** [24, 25] *If there is 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*

$$\dot{V}(x)|_{(1)} \leq -k_1 V(x)^r - k_2 V(x), \forall x \in \mathcal{U}. \tag{2}$$

*Then, the origin of system (1) is finite-time stable. The convergence time $T_1$ satisfies*

$$T_1(x_0) \leq \frac{\ln\left[1 + \frac{k_2}{k_1} V(x_0)^{1-r}\right]}{k_2(1-r)}, \tag{3}$$

*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.*

**Lemma 2** *Suppose there is a positive definite function $V(x) \in C^1$ defined on a neighborhood $\mathcal{U} \subset \mathcal{R}^n$ of the origin, such that*

$$\dot{V}(x)|_{(1)} \leq -k_1 V(x)^r - k_2 V(x) - k_3 V(x)^{\frac{1}{r}}, \forall x \in \mathcal{U}, \tag{4}$$

*where $k_1$, $k_2$, $k_3 > 0$ and $0 < r < 1$. Then, the origin of system (1) is locally finite-time stable. The convergence time $T_2$ satisfies*

$$T_2(x_0) \leq \begin{cases} \dfrac{r \ln\left[\dfrac{k_2 + k_3}{k_2 V(x_0)^{(r-1)/r} + k_3}\right]}{k_2(1-r)} + \dfrac{\ln\left(1 + \dfrac{k_2}{k_1}\right)}{k_2(1-r)}, V(x_0) \geq 1, \\ \dfrac{\ln\left[1 + \dfrac{k_2}{k_1} V(x_0)^{1-r}\right]}{k_2(1-r)}, V(x_0) < 1, \end{cases} \tag{5}$$

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

*Proof* Since $V(x) \geq 0$, it follows from (4) that

$$\dot{V}(x)|_{(1)} \leq -k_1 V(x)^r - k_2 V(x).$$

Then, Lemma 1 implies that the system (1) is finite-time stable.

Now, we prove that the inequality (5) holds.

If $V(x_0) \geq 1$, from (4), it follows that

$$\dot{V}(x) \leq -k_2 V(x) - k_3 V(x)^{\frac{1}{r}}. \tag{6}$$

Multiplying (6) by $e^{k_2 t}$, we have,

$$e^{k_2 t} \dot{V}(x) + e^{k_2 t} k_2 V(x) \leq -e^{k_2 t} k_3 V(x)^{\frac{1}{r}}.$$

Then,

$$\frac{d(e^{k_2 t} V(x))}{(e^{k_2 t} V(x))^{1/r}} \leq -k_3 e^{(1-\frac{1}{r}) k_2 t} dt.$$

Integrating the above differential inequality from 0 to $t$, we have

$$V(x) \leq e^{-k_2 t} \left[ V(x_0)^{1-\frac{1}{r}} + \frac{k_3}{k_2} - \frac{k_3}{k_2} e^{(1-\frac{1}{r}) k_2 t} \right]^{\frac{r}{r-1}}. \tag{7}$$

Let

$$t_1 = \frac{r \ln\left[\frac{k_2 + k_3}{k_2 V(x_0)^{(r-1)/r} + k_3}\right]}{k_2(1-r)}. \tag{8}$$

It is easy to check that

$$V(x) \leq 1, \ t \geq t_1. \tag{9}$$

When $V(x) \leq 1$, from (4), we have

$$\dot{V}(x) \leq -k_1 V(x)^r - k_2 V(x). \tag{10}$$

By Lemma 1, we obtain $t_2$,

$$t_2 = \frac{\ln\left(1 + \frac{k_2}{k_1}\right)}{k_2(1-r)}, \tag{11}$$

such that $x(t) = 0$, $t \geq t_2$. Therefore, if $V(x_0) > 1$, we have $x(t) = 0$, when, $t > t_1 + t_2$. Thus, the first inequality in (5) holds.

If $V(x_0) \leq 1$. The inequality (10) also holds. By Lemma 1, we have $x(t) = 0$, $t \geq T_1(x_0)$.

## 3 Problem formulation and neural network solvers with two tunable activation functions

A time-varying QP problem with time-varying linear-equality constraint is given as follows,

$$\text{minimze}: \quad \frac{1}{2}x^{\mathrm{T}}(t)P(t)x(t) + q^{\mathrm{T}}(t)x(t), \tag{12a}$$

$$\text{subject to}: \quad A(t)x(t) = b(t), \tag{12b}$$

where $P(t) \in R^{n\times n}$ is a smooth time-varying, positive-defined, symmetric Hessian matrix for any $t \in [0, +\infty)$, $q(t) \in R^n$ is defined as a smooth time-varying coefficient vector, $A(t) \in R^{m\times n}$ is a full row rank, smooth time-varying coefficient matrix, and $b(t) \in R^m$ is defined as smooth time-varying vector.

From [20], the problem (12) can be solved as follows:

$$W(t)Y(t) = u(t), \tag{13}$$

where $W(t) = \begin{bmatrix} P(t) & A^{\mathrm{T}}(t) \\ A(t) & \mathbf{0} \end{bmatrix}$, $Y(t) = \begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix}$, $u(t) = \begin{bmatrix} -q(t) \\ b(t) \end{bmatrix}$, $\lambda(t) \in R^m$ is assumed as the Lagrange multiplier vector.

As in [16–20], let $e(t) = W(t)Y(t) - u(t)$ denote the vector-valued error function. The time derivative $\dot{e}(t)$ is formulated as

$$\dot{e}(t) = -\varepsilon \mathcal{F}(e(t)), \tag{14}$$

where $\mathcal{F}(x)$ is defined as the following tunable activation functions:

$$\mathcal{F}(x) = \text{sign}(x)(k_1|x|^r + k_2|x| + k_3|x|^{\frac{1}{r}}), \tag{15}$$

or

$$\mathcal{F}(x) = \sum_{k=1}^{N} \text{sign}(x)(k_1|x|^{\frac{1}{k}} + k_2|x| + k_3|x|^k), \tag{16}$$

where $N > 1$ is an integer parameter, $0 < r < 1$, $k_1 > 0$, $k_2 > 0$, and $k_3 > 0$ are tunable parameters.

From (14), we have the following finite-time neural network model for solving the problem (12):

$$W(t)\dot{Y}(t) = -\varepsilon \mathcal{F}(W(t)Y(t) - u(t)) - \dot{W}(t)Y(t) + \dot{u}(t), \tag{17}$$

where the parameter $\varepsilon > 0$. Then, we have

$$\begin{aligned} \dot{Y}_i = &-\sum_{k=1}^{n+m} \dot{w}_{ik} Y_k + \sum_{k=1}^{n+m} (\sigma_{ik} - w_{ik})\dot{Y}_k \\ &-\varepsilon \mathcal{F}\left(\sum_{k=1}^{n+m} w_{ik} Y_k - u_i\right) + \dot{u}_i, \end{aligned} \tag{18}$$

where $Y_i$, $\dot{Y}_i$, $u_i$, and $\dot{u}_i$ are the $i$th elements of $Y(t)$, $\dot{Y}(t)$, $u(t)$, and $\dot{u}(t)$, respectively, $i = 1, 2, \cdots, n+m$, and $w_{ik}$, $\dot{w}_{ik}$, and $\sigma_{ik}$ are the $i$th line and $k$th column elements of $W(t)$, $\dot{W}(t)$, and identity matrix $I$, respectively.

Now, we give our main results.

**Theorem 1** *If the time-varying matrices $P(t)$, $q(t)$, $A(t)$, and $b(t)$ are given, starting from any initial value $Y(0)$, the neural network (17) with (15) always converges to its equilibrium state $[x^*(t), \lambda^*(t)]^{\mathrm{T}}$ in finite time, where $x^*(t)$ is the time-varying theoretical solution of (12).*

*Proof* Let $e^+(0) = \max_{1 \leq i \leq n+m}\{|e_i(0)|\}$. Note that every $e_i(t)$ in $e(t)$ has identical dynamic $\dot{e}_i(t) = \mathcal{F}(e_i(t))$. Then, we have $-|e^+(t)| \leq e_i(t) \leq |e^+(t)|$, for $t \geq 0$ and $1 \leq i \leq n+m$. That implies that $e_i(t)$ $(i = 1, \cdots, n+m)$ converge to zero when $e^+(t)$ reaches zero. Note that

$$\dot{e}^+(t) = -\varepsilon \mathcal{F}(e^+(t)), e^+(0) = \max\{|e_i(0)|\}.$$

Defining a Lyapunov function $V = |e^+(t)|^2$, then we have the time derivative of $V$:

$$\begin{aligned} \dot{V} &= 2|\dot{e}^+(t)|e^+(t) \\ &= -2\varepsilon \mathcal{F}(|e^+(t)|)e^+(t) \\ &= -2\varepsilon\left(k_1|e^+(t)|^{r+1} + k_2|e^+(t)|^2 + k_3|e^+(t)|^{\frac{1}{r}+1}\right) \\ &= -2\varepsilon\left(k_1 V^{\frac{r+1}{2}} + k_2 V + k_3 V^{\frac{1+r}{2r}}\right). \end{aligned}$$

From the proof of Lemma 2, we can obtain that neural network (17) with (15) converges to its equilibrium state $[x^*(t), \lambda^*(t)]^{\mathrm{T}}$ in finite-time interval. Then, the time-varying theoretical solution of (12) can be obtained in finite time, and the upper bound of convergence time $T_{f1}$ satisfies

$$T_{f1}=\begin{cases}\dfrac{r\ln\left[\dfrac{k_2+k_3}{k_2V_0^{(r-1)/2r}+k_3}\right]}{\varepsilon k_2(1-r)}+\dfrac{\ln\left(1+\dfrac{k_2}{k_1}\right)}{\varepsilon k_2(1-r)}, V_0\geq 1,\\ \dfrac{\ln\left[1+\dfrac{k_2}{k_1}V_0^{(1-r)/2}\right]}{\varepsilon k_2(1-r)}, V_0<1,\end{cases} \tag{19}$$

where $V_0=\max\{|e_1(0)|^2,|e_2(0)|^2,\cdots,|e_{n+m}(0)|^2\}$. □

**Theorem 2** *When the time-varying matrices $P(t)$, $q(t)$, $A(t)$, and $b(t)$ are given, the neural network (17) with (16) always converges to its equilibrium state $[x^*(t),\lambda^*(t)]^{\mathrm{T}}$ in finite time for any initial value $Y(0)$, where $x^*(t)$ is the time-varying theoretical solution of (12).*

*Proof* Choosing the Lyapunov function $V=|e^+(t)|^2$, we have

$$\begin{aligned}\dot V&=2|\dot e^+(t)|e^+(t)\\&=-2\varepsilon\mathcal{F}(|e^+(t)|)e^+(t)\\&=-2\varepsilon\sum_{k=1}^{N}\left(k_1|e^+(t)|^{\frac{1}{k}+1}+k_2|e^+(t)|^2+k_3|e^+(t)|^{k+1}\right)\\&=-2\varepsilon\sum_{k=1}^{N}\left(k_1V^{\frac{k+1}{2k}}+k_2V+k_3V^{\frac{1+k}{2}}\right).\end{aligned} \tag{20}$$

**I**. If $V_0>1$ ($V_0$ is given in the proof of Theorem 1), from (20), we have

$$\begin{aligned}\dot V&\leq-2\varepsilon\sum_{k=1}^{N}\left(k_3V^{\frac{k+1}{2}}+k_2V\right)\\&=-2\varepsilon\left(k_3\sum_{k=1}^{N}V^{\frac{k+1}{2}}+k_2NV\right)\\&\leq-2\varepsilon(k_3NV^2+k_2NV).\end{aligned}$$

As in the proof of Lemma 2, we can obtain $V(t)\leq 1$ when $t\geq t_1=\frac{\ln\left[\frac{k_2+k_3}{k_2/\sqrt{V_0}+k_3}\right]}{\varepsilon k_2N}$. When $V(t)\leq 1$, from (20), it follows that

$$\begin{aligned}\dot V&\leq-2\varepsilon\sum_{k=1}^{N}\left(k_1V^{\frac{k+1}{2k}}+k_2V\right)\\&=-2\varepsilon\left(k_1\sum_{k=1}^{N}V^{\frac{k+1}{2k}}+k_2NV\right)\\&\leq-2\varepsilon(k_1NV^{\frac{1}{2}}+k_2NV).\end{aligned}$$

Then, we have $V(t)=0$, when $t>t_2=2\ln(1+k_2/k_1)/(\varepsilon k_2N)$.

Therefore, if $V_0\geq 1$, we have $V(t)=0$ when $t\geq T_{f2}=t_1+t_2$.

**II**. If $V_0<1$, by Lemma 1, we have $V=0$ when $t\geq T_{f2}=\ln(1+k_2V_0^{\frac{1}{2}}/k_1)/(\varepsilon k_2N)$.

In conclusion, we can obtain that neural network (17) with (16) converges to its equilibrium state in finite time. Therefore, the time-varying theoretical solution of (12) can be obtained in finite-time interval and the upper bound of convergence time $T_{f2}$ satisfies

$$T_{f2}=\begin{cases}\dfrac{\ln\left[\dfrac{k_2+k_3}{k_2/\sqrt{V_0}+k_3}\right]}{\varepsilon k_2N}+\dfrac{2\ln\left(1+\dfrac{k_2}{k_1}\right)}{\varepsilon k_2N}, V_0\geq 1,\\ \dfrac{2\ln\left(1+\dfrac{k_2}{k_1}V_0^{\frac{1}{4}}\right)}{\varepsilon k_2N}, V_0<1.\end{cases} \tag{21}$$

*Remark 1* The neural networks proposed in [16–20], especially in [8], converge to the exact solution of (12) asymptotically. In other word, it requires infinitely long time to achieve the exact solution. From Theorem 1 and 2, we know that the neural network (17) with the activation function (15) or (16) can converge to the desired solution of (12) in finite time. Moreover, the upper bounds of convergence time $T_{f1}$ and $T_{f2}$ can be determined by $k_1$, $k_2$, $k_3$, $r$, $\varepsilon$, and $N$.

*Remark 2* If $V_0<1$, we can obtain $\frac{2\ln\left(1+\frac{k_2}{k_1}V_0^{\frac{1}{4}}\right)}{\varepsilon k_2N}<\frac{\ln\left(1+\frac{k_2}{k_1}V_0^{\frac{1}{4}}\right)}{\varepsilon k_2}=\frac{(1-r)\ln\left(1+\frac{k_2}{k_1}V_0^{\frac{1}{4}}\right)}{\varepsilon k_2(1-r)}$. Moreover, let $F(r)=\ln\left[1+\frac{k_1}{k_2}V_0^{(1-r)/2}\right]-(1-r)\ln\left(1+\frac{k_2}{k_1}V_0^{\frac{1}{4}}\right)$, we have

$$F'(r)=\frac{1}{2}\frac{\ln\frac{1}{V_0}k_1/k_2V_0^{\frac{1-r}{2}}}{1+k_1/k_2V_0^{\frac{1-r}{2}}}+\ln\left(1+k_1/k_2V_0^{\frac{1}{4}}\right)>0.$$

Note that $F(0)>0$, then $F(r)>0$. Therefore, $T_{f2}<T_{f1}$. If $V_0\geq 1$, using the same method, we also have $T_{f2}<T_{f1}$.

## 4 Robustness analysis

We discuss the robustness of the neural network (17) by adding a large model-implementation error $\Delta\omega\in\mathbf{R}^{n+m}$ to (14),

$$\dot e(t)=-\varepsilon\mathcal{F}(e(t))+\Delta\omega, \tag{22a}$$

$$\dot e_i(t)=-\varepsilon\mathcal{F}(e_i(t))+\Delta\omega_i, \tag{22b}$$

where $e_i(t)$ and $\Delta\omega_i$ are the $i$th elements of $e(t)$ and $\Delta\omega$, respectively, $i=1,2,\cdots,n+m$.

In the following theorem, we will show the robustness of the proposed neural networks with a large implementation error.

**Theorem 3** *Consider the perturbed neural network* (22) *with the activation function* (15), *the steady-state error satisfies the following inequality*:

$$\lim_{t\to\infty}\|e(t)\|_2 \le \sqrt{\frac{n+m}{\left(27k_1^2k_2^2k_3^2\right)^{\frac{1}{r+1+\frac{1}{r}}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{6}{r+1+\frac{1}{r}}}},$$

where $|\Delta\omega_i| \le \delta \le +\infty$ *for any* $t \in [0,+\infty)$ *with* $\delta \gg \varepsilon$ *or at least* $\delta \ge \varepsilon$. *For the perturbed neural network* (22) *with the activation function* (16), *the steady-state error satisfies the following inequality*:

$$\lim_{t\to\infty}\|e(t)\|_2 \le \sqrt{\frac{n+m}{\left(432N^3k_1^{\frac{3}{2}}k_2^3k_3^{\frac{3}{2}}\right)^{\frac{4}{9+3N}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{8}{3+N}}},$$

for error range $\lim_{t\to\infty}|e_i| \ge 1$ *and* $N > 1$.

*Proof* Define a Lyapunov function $v_i = |e_i(t)|^2/2$ for the perturbed neural network (22). Then, we have

$$\dot{v}_i(t) = |e_i(t)|\dot{e}_i(t) = -\varepsilon|e_i(t)|\mathcal{F}(e_i(t)) + |e_i(t)|\Delta\omega_i.$$

Note that $|\Delta\omega_i| \le \delta$. We can obtain $|e_i||\Delta\omega_i| \le \delta|e_i|$. Then, we have

$$\dot{v}_i(t) \le -\varepsilon|e_i(t)|\mathcal{F}(e_i) + \delta|e_i| = -|e_i|(\varepsilon\mathcal{F}(e_i) - \delta). \tag{23}$$

The time evolution $e_i(t)$ $(1 \le i \le n+m)$ has three cases: (**i**). $\varepsilon\mathcal{F}(e_i) - \delta > 0$, (**ii**). $\varepsilon\mathcal{F}(e_i) - \delta = 0$, (**iii**). $\varepsilon\mathcal{F}(e_i) - \delta < 0$. In the following part, we will give a detailed discussion.

(**a**) When the perturbed neural network (22) with $t \in [t_0, t_1]$ is in the case (i), then $\dot{v}_i < 0$ and (23) imply that $e_i(t)$ converges to zero as $t$ evolves.

(**b**) If the perturbed neural network (22) with any time $t$ is in the case (ii), then $\dot{v}_i \le 0$ implies that $e_i$ converges to zero or $\mathcal{F}(e_i) = \delta/\varepsilon$.

(**c**) For the case (iii), the worst case, (23) implies that the situation $\dot{v}_i > 0$ will happen. So, $|e_i|$ may not converge to zero. For the worst case, note that $0 < \dot{v}_i \le -|e_i|(\varepsilon\mathcal{F}(e_i) - \delta)$, as $\varepsilon\mathcal{F}(e_i) - \delta$ range in value from a positive value to 0, $v_i$ and $|e_i|$ will increase. So, there exists $t_2$ such that $\varepsilon\mathcal{F}(e_i(t_2)) - \delta = 0$, which returns to the case (ii), $\mathcal{F}(e_i) = \delta/\varepsilon$.

By the above analysis, the steady-state error is $\lim_{t\to\infty}\mathcal{F}(|e_i|) \approx \frac{\delta}{\varepsilon}$. For $\delta \gg \varepsilon$ or at least $\delta \ge \varepsilon$, we can obtain $\lim_{t\to\infty}\mathcal{F}(|e_i|) \approx \delta/\varepsilon \gg 1$ or at least $\ge 1$.

For the perturbed neural network (22) with (15), we have

$$\begin{aligned}
\frac{\delta^2}{\varepsilon^2} &\approx \lim_{t\to\infty}\mathcal{F}^2(|e_i(t)|)\\
&= \lim_{t\to\infty}\left(k_1|e_i(t)|^r + k_2|e_i(t)| + k_3|e_i(t)|^{\frac{1}{r}}\right)^2\\
&\ge \lim_{t\to\infty}\left(k_1^2|e_i(t)|^{2r} + k_2^2|e_i(t)|^2 + k_3^2|e_i(t)|^{\frac{2}{r}}\right)\\
&\ge \lim_{t\to\infty} 3\left[k_1^2k_2^2k_3^2|e_i(t)|^{2r+2+\frac{2}{r}}\right]^{\frac{1}{3}}.
\end{aligned}$$

Thus,

$$\lim_{t\to\infty}|e_i(t)|^2 \le \left(\frac{1}{27k_1^2k_2^2k_3^2}\right)^{\frac{1}{r+1+\frac{1}{r}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{6}{r+1+\frac{1}{r}}}.$$

Then,

$$\lim_{t\to\infty}\|e(t)\|_2 \le \sqrt{\frac{n+m}{\left(27k_1^2k_2^2k_3^2\right)^{\frac{1}{r+1+\frac{1}{r}}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{6}{r+1+\frac{1}{r}}}}. \tag{24}$$

For the perturbed formula (22) with (16), we have

$$\begin{aligned}
\frac{\delta^2}{\varepsilon^2} &\approx \lim_{t\to\infty}\mathcal{F}^2(|e_i(t)|)\\
&= \lim_{t\to\infty}\left(\sum_{k=1}^{N}(k_1|e_i(t)|^{\frac{1}{k}} + k_2|e_i(t)| + k_3|e_i(t)|^k)\right)^2\\
&= \lim_{t\to\infty}\left(Nk_2|e_i(t)| + \sum_{k=1}^{N}k_1|e_i(t)|^{\frac{1}{k}} + \sum_{k=1}^{N}k_3|e_i(t)|^k\right)^2\\
&\ge \lim_{t\to\infty}\left(Nk_2|e_i(t)| + 2\sqrt{k_1k_3\textstyle\sum_{k=1}^{N}|e_i(t)|^{\frac{1}{k}}\sum_{k=1}^{N}|e_i(t)|^k}\right)^2\\
&= \lim_{t\to\infty}\left(N^2k_2^2|e_i(t)|^2 + 4k_1k_3\sum_{k=1}^{N}|e_i(t)|^{\frac{1}{k}}\sum_{k=1}^{N}|e_i(t)|^k\right.\\
&\quad \left. + 4Nk_2|e_i(t)|\sqrt{k_1k_3\textstyle\sum_{k=1}^{N}|e_i(t)|^{\frac{1}{k}}\sum_{k=1}^{N}|e_i(t)|^k}\right)\\
&\ge \lim_{t\to\infty}\left(N^2k_2^2|e_i(t)|^2 + 4k_1k_3|e_i(t)|\sum_{k=1}^{N}|e_i(t)|^k\right.\\
&\quad \left. + 4Nk_2|e_i(t)|\sqrt{k_1k_3|e_i(t)|\textstyle\sum_{k=1}^{N}|e_i(t)|^k}\right)\\
&\ge \lim_{t\to\infty}\left(N^2k_2^2|e_i(t)|^2 + 4k_1k_3|e_i(t)|^{N+1}\right.\\
&\quad \left. + 4Nk_2k_1^{\frac{1}{2}}k_3^{\frac{1}{3}}|e_i(t)|^{\frac{N+3}{2}}\right)\\
&\ge \lim_{t\to\infty} 3\sqrt[3]{16N^3k_1^{\frac{3}{2}}k_2^3k_3^{\frac{3}{2}}|e_i(t)|^{3+N+\frac{N+3}{2}}}.
\end{aligned}$$

Thus,

$$\lim_{t\to\infty}|e_i(t)|^2 \le \left(\frac{1}{432N^3k_1^{\frac{3}{2}}k_2^3k_3^{\frac{3}{2}}}\right)^{\frac{4}{9+3N}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{8}{3+N}}.$$

Then,

$$\lim_{t\to\infty}\|e(t)\|_2 \le \sqrt{\frac{n+m}{\left(432N^3k_1^{\frac{3}{2}}k_2^3k_3^{\frac{3}{2}}\right)^{\frac{4}{9+3N}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{8}{3+N}}}. \quad (25)$$

The proof is completed. □

*Remark 3* In [8], the steady-state errors are $\sqrt{n+m}\delta/\varepsilon$ and $\sqrt{n+m}\delta/(N\varepsilon)$ with the linear activation function and power-sum activation function, respectively. Note that $6/(1+r+1/r) \le 2$ and $8/(3+N)\le 2$, so the steady-state errors with (15) and (16) are smaller than the ones with the linear activation function and with the power-sum activation function by setting appropriate values of $k_1$, $k_2$, $k_3$ and $N$.

*Remark 4* Note that $6/(1+r+1/r) > 8/(3+N)$ for any $N > 1$ and $0.43 \le r < 1$, we know that the steady-state error with (16) is smaller than the one with (15) for the same parameters. For $0 < r < 0.43$, we also obtain that the steady-state error with (16) is smaller the one with (15) by setting the values of the parameters $k_1$, $k_2$, $k_3$, and $N$.

## 5 Numerical simulations

In this part, we give numerical simulations for solving the time-varying QP by the neural network (17) with the activation function (15) and with the activation function (16), and with the power-sum activation function in [8], respectively. We will show superiority of our methods from two aspects: finite-time convergence and robustness.

*Example 1* For (12), we set $P(t) = \begin{bmatrix} 8+\sin t & 0.9\cos t \\ 0.9\cos t & 10-0.5\cos t \end{bmatrix}$, $q(t) = \begin{bmatrix} -2\cos 2t \\ -2\sin 2t \end{bmatrix}$, $A(t) = [2\cos 3t, \sin 3t]$, $b(t) = \sin t$.

From (13), we have $u(t) = \begin{bmatrix} 2\cos 2t \\ 2\sin 2t \\ \sin t \end{bmatrix}$, $W(t) = \begin{bmatrix} 8+\sin t & 0.9\cos t & 2\cos 3t \\ 0.9\cos t & 10-0.5\cos t & \sin 3t \\ 2\cos 3t & \sin 3t & 0 \end{bmatrix}$.

Letting $Y_0 = [2,1,1]^{\mathrm{T}}$, $\varepsilon = 1$, $r = 0.5$, $k_1 = k_2 = k_3 = 1$, $N = 3$, we have $e(0) = W(0)Y(0) - u(0) = [17.8, 10.4, 4]^{\mathrm{T}}$, $V_0 = 17.8^2 > 1$. Then, we have $T_{f1} = 2.0248s$ and $T_{f2} = 0.6749s$ by calculating (19) and (21), respectively. The convergence time $T_{fc}$ is determined such that

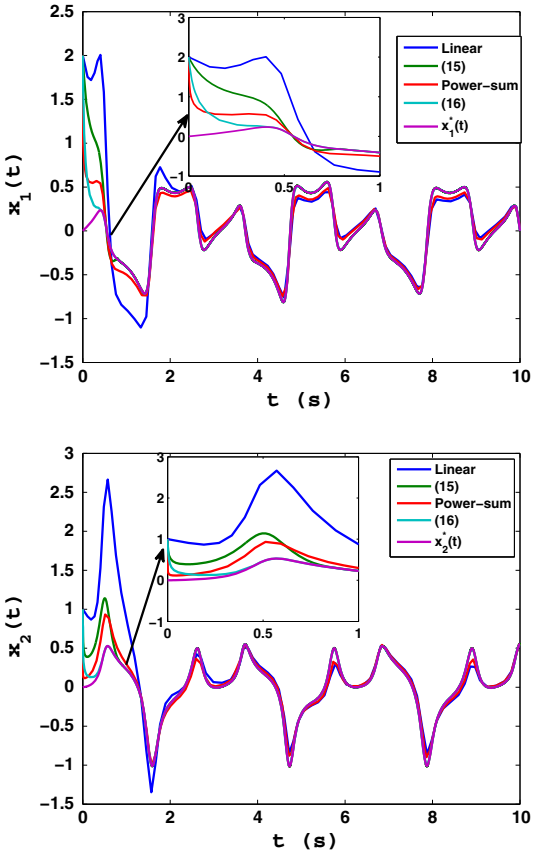



**Fig. 1** Online solutions of Example 1 by the neural network (17) with (15) , (16), and the neural network ($\gamma = 1$, $N = 3$) in [8]

$\|x(t) - x^*(t)\| \le 10^{-6}$, $t > T_{fc}$. It is observed form Fig. 1 and 2 that the neural networks can converge to the theoretical solution in finite time and the convergence time $T_{fc1} = 0.82$ and $T_{fc2} = 0.45$, respectively, while the neural network in [8] can not converge to the theoretical solution in finite time.

Let $\Delta\omega = [10^2, 10^2, 10^2]^{\mathrm{T}}$ in (22) and $\varepsilon = 1$, $r = 0.5$, $k_1 = k_2 = k_3 = 5$, $N = 2$, $\delta = 100\sqrt{3}$. From (24) and (25), we obtain that the steady-state errors are given as

$$\sqrt{\frac{n+m}{(27k_1^2k_2^2k_3^2)^{\frac{1}{r+1+\frac{1}{r}}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{6}{r+1+\frac{1}{r}}}} \approx 22.58,$$

and

$$\sqrt{\frac{n+m}{\left(432N^3k_1^{\frac{3}{2}}k_2^3k_3^{\frac{3}{2}}\right)^{\frac{4}{9+3N}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{8}{3+N}}} \approx 9.9638.$$

However, the steady-state error $\sqrt{n+m}\delta/(\varepsilon N) \approx 100$ [8]. The steady-state errors obtained by our methods are smaller than the one obtained by [8]. Moreover, the neural network with the activation function (16) has the most superior robustness performance. The simulation results

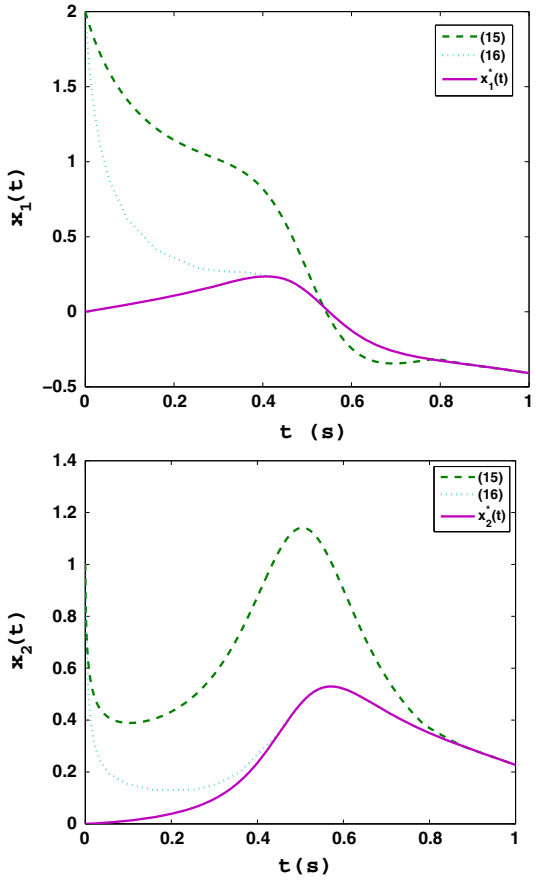



**Fig. 2** The convergence time of online solution of Example 1 by the neural network (17) with (15) and (16)

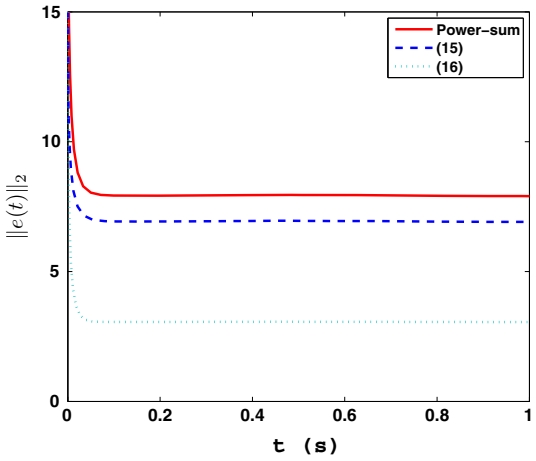



**Fig. 3** The robustness performance of online solution of Example 1 by the neural network (17) with (15), (16), and the neural network ($\gamma = 1$, $N = 3$) in [8]

are given in Fig. 3. It also shows that our method has a superior robustness performance.

In conclusion, theoretical analysis and numerical simulations demonstrate the effectiveness of our methods.

*Example 2* We select a time-varying Toeplitz matrix

$$P(t) = \begin{bmatrix} p_1(t) & p_2(t) & p_3(t) & \cdots & p_n(t) \\ p_2(t) & p_1(t) & p_3(t) & \cdots & p_{n-1}(t) \\ p_3(t) & p_2(t) & p_1(t) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & p_2(t) \\ p_n(t) & p_{n-1}(t) & \cdots & p_2(t) & p_1(t) \end{bmatrix} \in R^{n\times n},$$

where $p_1(t) = 8 + \cos t$, $p_i(t) = \frac{\sin t}{i-1}$, $i = 2, 3, \cdots, n$. Let $q(t) = \left[-2\cos 2t, 2\cos\left(2t + \frac{\pi}{2}\right), \cdots, 2\cos\left(2t + \frac{(n-1)\pi}{2}\right)\right]$ $\mathrm{T} \in R^{n\times 1}$, $A(t) = \left[\sin t, \sin\left(t - \frac{\pi}{3}\right), \sin\left(t - \frac{2\pi}{3}\right), \cdots, \sin\left(t - \frac{(n-1)\pi}{3}\right)\right] \in R^{1\times n}$, and $b(t) = 2\cos\left(2t + \frac{n\pi}{2}\right) \in R$ for (12).

In the simulation, let $n = 4$, $\varepsilon = 2$, $r = 0.5$, $k_1 = k_3 = 1$, $k_2 = 2$, $N = 4$, $Y_0 = \left[-\frac{1}{4}, \frac{1}{16}, -\frac{1}{4}, \frac{1}{16}, \frac{1}{4}\right]^{\mathrm{T}}$. Then, $e(0) = W(0)Y(0) - u(0) = \left[0, \frac{1}{2}, 0, \frac{1}{2}, 0\right]^{\mathrm{T}}$ and $V_0 = 0.5^2 < 1$. Therefore, we can obtain $T_{f1} = 0.4407s$ and $T_{f2} = 0.1102s$ by computing (19) and (21), respectively. From Figs. 4 and 5, it is observed that the convergence time $T_{fc_1} = 0.38s$ and $T_{fc_2} = 0.095s$, respectively.

Next, we compare the robustness performance of the neural networks proposed in this paper with the one in [8]. Let $\Delta\omega = [10^2, 10^2, 10^2]^{\mathrm{T}}$ in (22), $\varepsilon = 2$, $r = 0.5$, $k_1 = k_3 = 1$, $k_2 = 2$, $N = 4$, $\delta = 100\sqrt{3}$. From (24) and (25), we obtain that the steady-state errors are given as

$$\sqrt{\frac{n+m}{(27k_1^2k_2^2k_3^2)^{\frac{1}{r+1+\frac{1}{r}}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{6}{r+1+\frac{1}{r}}}} \approx 43.0257,$$

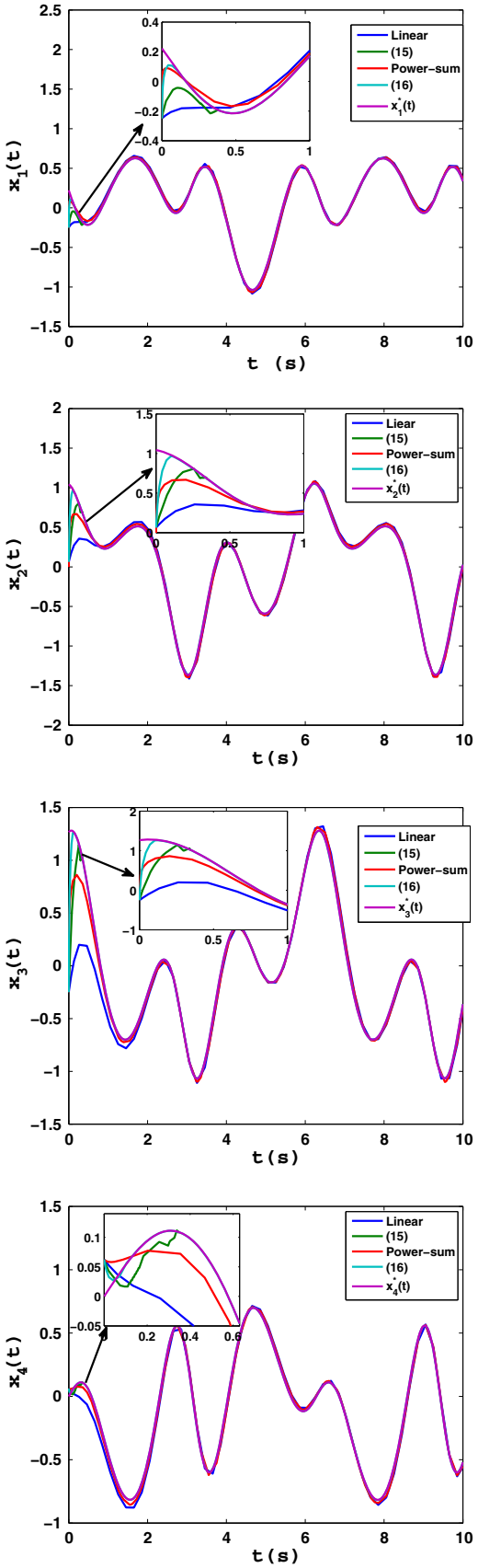
and

$$\sqrt{\frac{n+m}{\left(432N^3k_1^{\frac{3}{2}}k_2^3k_3^{\frac{3}{2}}\right)^{\frac{4}{9+3N}}}\left(\frac{\delta}{\varepsilon}\right)^{\frac{8}{3+N}}} \approx 8.8638.$$

However, the steady-state errors $\sqrt{n+m}\delta/(\varepsilon N) \approx 48.4123$ and $\sqrt{n+m}\delta/\varepsilon \approx 193.6492$ [8]. Obviously, the steady-state errors obtained by our methods are smaller than the ones obtained by [8]. Moreover, the neural network with the activation function (16) has the most superior robustness performance. The simulation results are shown in Fig. 6.

## 6 Application to robot tracking

The neural network (17) can be applied to robot tracking by solving a time-varying QP problem. Compared with the neural networks proposed in [8–12, 28–31], the theoretical solution can be achieved in finite time by our method. If there exists a large implementation error, our neural networks also have a superior robustness performance.




◀**Fig. 4** Online solution of Example 2 by the neural network (17) with (15), (16), and the neural network ($\gamma = 1$, $N = 3$) in [8]

### 6.1 Model

Consider the following redundant robot manipulator [8],

$$r(t) = f(\theta(t)), \tag{26}$$

where $r(t) \in R^m$ denotes the end-effector position vector in Cartesian space, $\theta(t) \in R^n$ denotes the joint-space vector, and $f(\cdot)$ denotes a continuous nonlinear function with known structure and parameters. The inverse kinematic problem is to find the joint variable $\theta(t)$ for any given $r(t)$. By (26), the relation between $\dot{r}(t)$ and $\dot{\theta}(t)$ can be described as

$$J(\theta(t))\dot{\theta}(t) = \dot{r}(t), \tag{27}$$

where $J(\theta(t)) = \partial f(\theta)/\partial\theta$ is the Jacobian matrix.

By the previous work [8, 10], we have the following time-varying QP problem:

$$\text{minimze}: \quad \frac{1}{2}x^{\mathrm{T}}(t)P(t)x(t) + q^{\mathrm{T}}(t)x(t), \tag{28a}$$

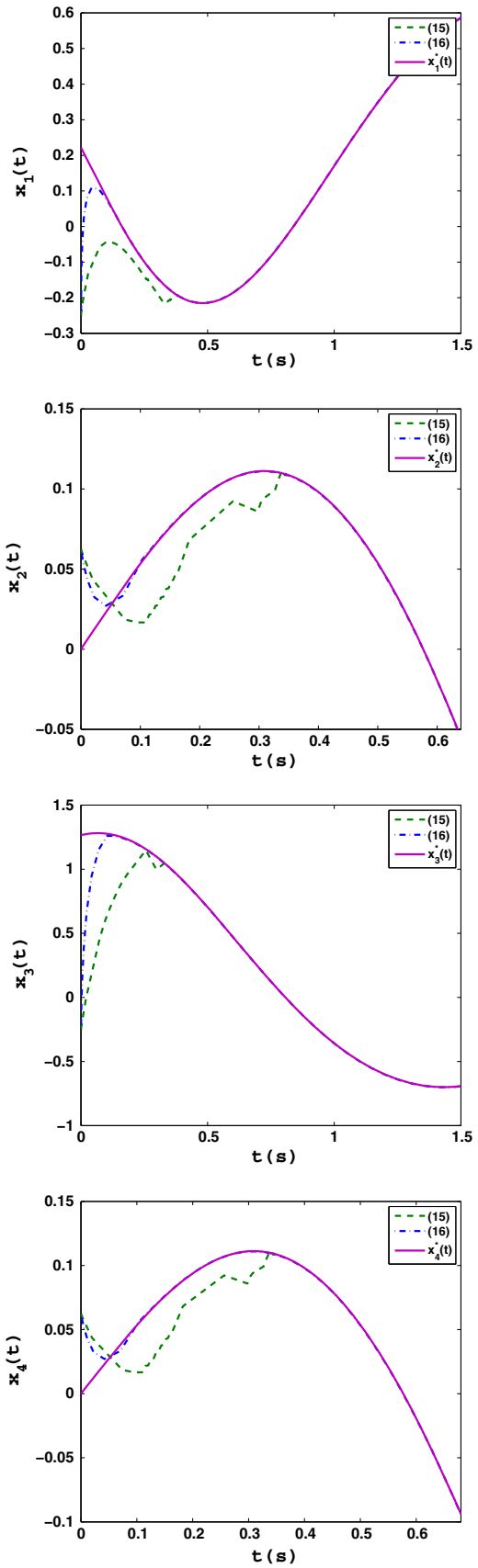
$$\text{subject to}: \quad A(t)x(t) = b(t), \tag{28b}$$

where $P = I$, $q(t) = \mu(\theta(t) - \theta(0))$, $A(t) = J(t)$, and $b(t) = \dot{r}(t)$. $\theta(t) \in R^n$ is the joint variable vector, $\mu$ is a parameter used to scale the magnitude of the manipulator response to joint displacements, and $x(t) = \dot{\theta}(t)$ will be solved online.

### 6.2 Simulation results

In this part, a five-link planar redundant robot manipulator is applied to simulation. The five-link robot has three redundant degrees (due to $n = 5$, $m = 2$); the desired path of its end effector is an ellipse with the major radius being 0.6m and the minor radius being 0.3m. The initial conditions are $\theta_0 = [3\pi/4, -\pi/2, -\pi/4, \pi/6, \pi/3]^{\mathrm{T}}$ rad and $Y_0 = [0, 0, 0, 0, 0, 0, 0]^{\mathrm{T}}$. In addition, let $N = 3$, $\mu = 4$, and $\varepsilon = 10$. We select the activation function (16) with $k_1 = k_2 = k_3 = 1$. Then, $e(0) = W(0)Y(0) - u(0) = [0, 0, 0, 0, 0, 0, 0.3]^{\mathrm{T}}$ and $V_0 = 0.3^2 < 1$. Therefore, we have $T_{f2} = 0.0291s$ by calculating (21).

Figure 7 gives the motion trajectories of the five-link planar robot manipulator operating in two-dimensional space. Figure 8 shows that the actual trajectory of the robot's end effector, and the desired elliptical path are sufficiently match. In addition, the trajectories of $\dot{\theta}(t)$ and $\theta(t)$ are shown in Figs. 9 and 10, respectively.




◀**Fig. 5** The convergence time of online solution of Example 2 by the neural network (17) with (15) and (16)

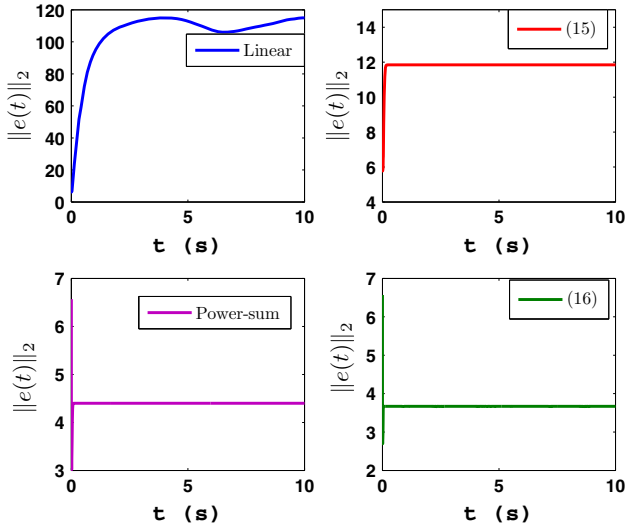



**Fig. 6** The robustness performance of online solution of Example 2 by using the linear function ($\gamma = 2$), the power-sum function ($\gamma = 2$, $N = 4$), and the functions (15) and (16)

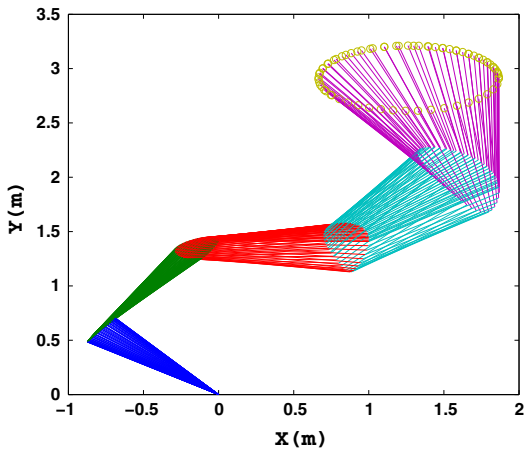



**Fig. 7** Motion trajectories of a five-link planar manipulator synthesized by the neural network (17) with (16) ($k_1 = k_2 = k_3 = 1, N = 3$)

It is observed from Fig. 11 that our neural network can converge to the theoretical solution in finite time comparing to the one in [8], and the actual convergence time is $T_{fc2} = 0.024s$.

Let $\Delta\omega = [30, 30, 30, 30, 30, 30, 30]^{\mathrm{T}}$ in (22), $\varepsilon = 10$, $k_1 = k_2 = k_3 = 1$. Figure 12 shows that our neural network has a superior robustness performance than the one in [8].

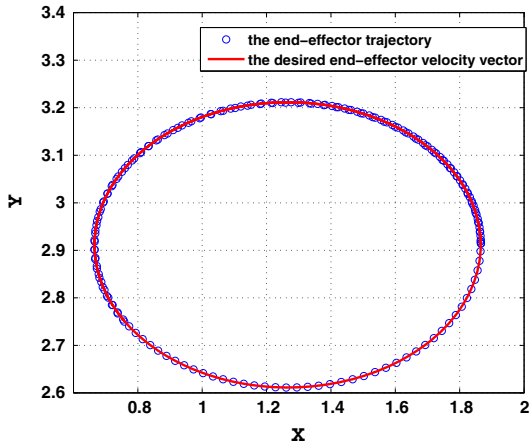



**Fig. 8** The end-effector trajectory of a five-link planar manipulator and the desired end-effector velocity vector

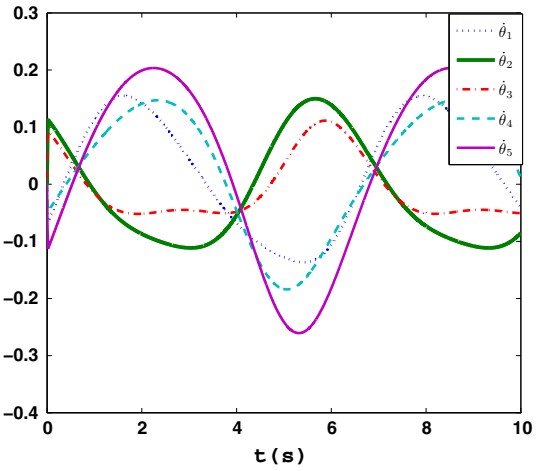



**Fig. 9** Joint velocity profiles, $\dot{\theta}(t)$, of a five-link planar robot manipulator synthesized when tracking an elliptical path

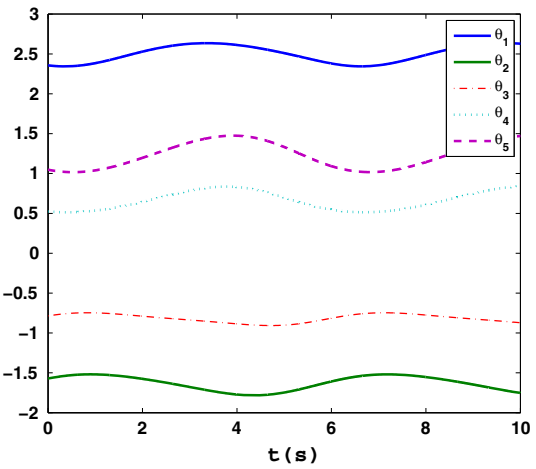



**Fig. 10** Joint variable, $\theta(t)$, of a five-link planar robot manipulator synthesized when tracking an elliptical path

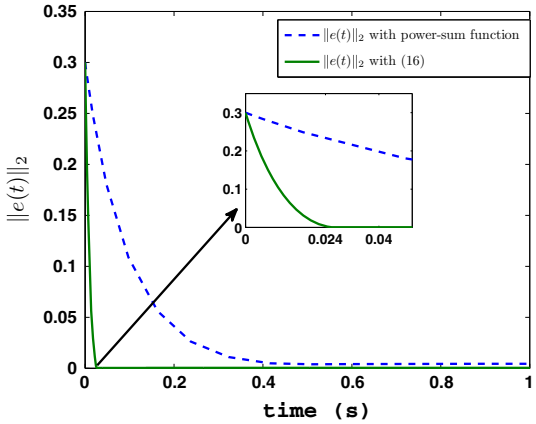



**Fig. 11** Residual errors, $\|e(t)\|_2$, of the neural network (17) with (16) ($k_1 = k_2 = k_3 = 1, N = 3, \varepsilon = 10$) and the one in [8] with power-sum function ($N = 3, \gamma = 10$)

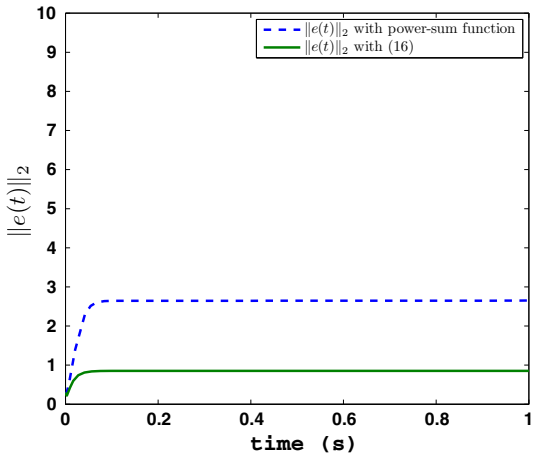



**Fig. 12** The robustness performances of the neural network (17) with (16) ($k_1 = k_2 = k_3 = 1, N = 3, \varepsilon = 10$) and the model in [8] with power-sum function ($N = 3, \gamma = 10$)

## 7 Conclusions

In this paper, finite-time recurrent neural network has been designed to solve time-varying QP problems and applied to robot tracking. Firstly, finite-time criteria and upper bounds of the convergent time were reviewed. Then, finite-time neural network with tunable activation function was proposed and applied to solve the time-varying QP problems. Finite-time convergent theorems of the proposed neural network were presented and proved. The upper bounds of the convergent time were estimated less conservatively. The neural network also has a superior robustness performance against perturbation with large implementation errors. Thirdly, feasibility and superiority of our method were shown by numerical simulations. At last, the proposed neural network was applied to robot tracking. Simulation

results also showed the effectiveness of the proposed methods.

**Acknowledgments** This work was supported by the National Science Foundation of China (Nos. 61374028, 61374171, 51177088, 61273183, 61304162), the Grant National Science Foundation of Hubei Provincial (2013CFA050), and the Graduate Scientific Research Foundation of China Three Gorges University (2014PY064, 2014PY069).

## References

1. Li S, Liu B, Li Y (2013) Selective positive–negative feedback produces the winner-take-all competition in recurrent neural networks. IEEE Trans Neural Netw Learn Syst 24(2):301–309
2. Liu Q, Wang J (2008) Two k-winners-take-all networks with discontinuous activation functions. Neural Netw 21(2–3): 406–413
3. Hu X, 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 Trans Neural Netw 20(4):654–664
4. Tymoshchuk P (2013) A model of analogue k-winners-take-all neural circuit. Neural Netw 42:44–61
5. Johansen TA, Fossen TI, Berge SP (2004) Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. IEEE Trans Control Syst Technol 12:211–216
6. Fares B, Noll D, Apkarian P (2002) Robust control via sequential semidefinite programming. SIAM J Control Optim 40:1791–1820
7. Leithead WE, Zhang Y (2007) O($N^2$)-operation approximation of covariance matrix inverse in Gaussian process regression based on quasi-Newton BFGS method. Commun Stat Simul Comput 36:367–380
8. Yang Y, Zhang Y (2013) Superior robustness of power-sum activation functions in Zhang neural networks for time-varying quadratic programs perturbed with large implementation errors. Neural Comput Appl 22:175–185
9. Wang J, Zhang Y (2004) Recurrent neural networks for real-time computation of inverse kinematics of redundant manipulators. Machine intelligence quo vadis?. World Scientific, Singapore
10. Zhang Y, Tan Z, Chen K, Yang Z, Lv X (2009) Repetitive motion of redundant robots planned by three kinds of recurrent neural networks and illustrated with a four-link planar manipulator's straight-line example. Robot Auton Syst 57:645–651
11. Zhang Y, Ma W, Li X, Tan H, Chen K (2009) MATLAB Simulink modeling and simulation of LVI-based primal-dual neural network for solving linear and quadratic programs. Neurocomputing 72:1679–1687
12. Zhang Y, Tan Z, Yang Z, Lv X (2008) A dual neural network applied to drift-free resolution of five-link planar robot arm. In: Proceedings of the 2008 IEEE international conference on information and automation. Zhangjiajie, China, 20–23 June 2008
13. Boggs PT, Tolle JW (1995) Sequential quadratic programming. Acta Numer 4:1–51
14. Murray W (1997) Sequential quadratic programming methods for large-scale problems. Comput Optim Appl 7:127–142
15. Hu J, Wu Z, McCann H, Davis LE, Xie C (2005) Sequential quadratic programming method for solution of electromagnetic inverse problems. IEEE Trans Antennas Propag 53:2680–2687
16. Zhang Y, Ma W, Cai B (2009) From Zhang neural network to Newton iteration for matrix inversion. IEEE Trans Circuits Syst I 56(7):1405–1415
17. Zhang Y, Ge SS (2005) Design and analysis of a general recurrent neural network model for time-varying matrix inversion. IEEE Trans Neural Netw 16(6):1477–1490
18. Zhang Y, Jiang D, Wang J (2002) A recurrent neural network for solving Sylvester equation with time-varying coefficients. IEEE Trans Neural Netw 13(5):1053–1063
19. Li Z, Zhang Y (2010) Improved Zhang neural network model and its solution of time-varying generalized linear matrix equations. Expert Syst Appl 37(10):7213–7218
20. Zhang Y, Li Z (2009) Zhang neural network for online solution of time-varying convex quadratic program subject to time-varying linear-equality constraints. Phys Lett A 373:1639–1643
21. Li S, Li Y, Wang Z (2013) A class of finite-time dual neural networks for solving quadratic programming problems and its k-winners-take-all application. Neural Netw 39:27–39
22. Li S, Chen S, Liu B (2013) 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 37:189–205
23. Miao P, Shen Y, Xia X (2014) Finite time dual neural networks with a tunable activation function for solving quadratic programming problems and its application. Neurocomputing 143:80–89
24. Shen Y, Xia X (2008) Semi-global finite-time observers for nonlinear systems. Automatica 44:3152–3156
25. Shen Y, Huang Y (2012) Global finite-time stabilisation for a class of nonlinear systems. Int J Syst Sci 43(1):73–78
26. Miao P, Shen Y, Hou J, Shen Y (2014) A recurrent neural network with a tunable activation function for solving k-winners-take-all. In: Proceedings of the 33rd Chinese control conference July 28–30. Nanjing, China, pp 4957–4962
27. Bhat S, Bernstein D (2000) Finite-time stability of continuous autonomous systems. SIAM J Control Optim 38:751–766
28. Kumar Naveen, Panwar Vikas, Borm Jin-Hwan, Chai Jangbom, Yoon Jungwon (2013) Adaptive neural controller for space robot system with an attitude controlled base. Neural Comput Appl 23:2333–2340
29. Li S, Cui H, Li Y, Liu B, Lou Y (2013) Decentralized control of collaborative redundant manipulators with partial command coverage via locally connected recurrent neural networks. Neural Comput Appl 23:1051–1060
30. Torres C, de Jesús Rubio J, Aguilar-Ibáñez CF, Pérez-Cruz JH (2014) Stable optimal control applied to a cylindrical robotic arm. Neural Comput Appl 24:937–944
31. Samy Assal FM (2013) Learning from hint for the conservative motion of the constrained industrial redundant manipulators. Neural Comput Appl 23:1649–1660