Machine Intelligence & Robotic Control, Vol. 1, No. 2, 71–78 (1999)

Paper

Stable Adaptive Control for Robot Trajectory Tracking Using Dynamic Neural Networks*

Fuchun Sun†, Zengqi Sun‡, Nan Li†, and Lingbo Zhang†

Abstract: This paper presents a stable adaptive control approach based on dynamic neural networks (DNNs) for robot manipulators with unknown dynamics nonlinearities. DNNs in the proposed control scheme are used to approximate the whole robot dynamic systems, i.e. the dynamic behavior of the closed-loop systems, which is distinguished from static feedforward NNs (SNNs) being used to approximate the nonlinear components in robot dynamic systems. One benefit of using DNNs is that the approximation precision and convergence of tracking errors can be improved, and especially, some specific dynamic behaviors, such as limit cycles and chaos, etc. that can not be dealt with SNNs can be approximated in this case. On the other hand, by using the dynamic inversion of the DNN system, the DNN dynamics behavior can be predetermined by design. As a result, the perceptive input of the DNNs will strictly lie in some compact set, this excludes the assumption that is often used in the existing literature, i.e., the robot states are assumed to be within a compact set. The robot control is composed of the dynamic inversion of the DNN system, adaptive compensation and a sliding control. The proof of a complete stability and the tracking error convergence is given by using Lyapunov stability theory that results in novel learning algorithms for the DNN systems. Simulations of a two-link manipulator show that stable adaptive control approach using DNNs performs better than that using SNNs.

Keywords: Robot, Dynamic Neural Networks (DNNs), Adaptive Tracking Control, Stability

1. Introduction

NEURAL Networks (NNs) are capable of learning and reconstructing complex nonlinear mapping and have been widely used in the identification and control for robotic manipulators by many researchers. The early NN applications in the control of robotic manipulators include Albus and Miller's CMAC Controller [1], [2], Iiguni's linear optimal control techniques with back-propagation NNs [3], Kawato and Ozaki's feed-forward compensators using back-propagation NNs [4], [5] for improving the control performance, etc. These NN-based control approaches could give good simulations or even experimental results. However, lack of theoretical analysis and stability security makes industrialists wary of using the results in real industrial environments.

To cope with these problems, stable NN-based on-line adaptive control both in continuous and discrete time for robots has been recently investigated by many researchers [6]-[8]. Representatives of these researches are multilayer NN-based adaptive controllers for robotic manipulators by Lewis [6], [7] and linearly parameterized NN-based adaptive control by Slotine *et al.* [8], [9]. In the proposed control framework, multilayer NNs and linearly parameterized NNs were used to approximate the nonlinear components in the robot dynamic system, respectively, and Lyapunov stability theory or passive theory is employed to design a closed loop control system with stability, convergence and improved robustness. As a result, the designed system is stable, and an on-line NN weight updating law is yielded for the function approximations. However, as one of the consequences of these applications, some of the system dynamic behaviors, such as limit cycles and chaos, etc., will lose if SNNs are used to approximate the nonlinear components in robot dynamic equation [10].

The DNN has advantages over a SNN due to the feedback connections between layers and neurons forming complicated dynamics, it could deal with time-varying input or output through their own natural temporal operation. Thus, the DNN is a dynamic mapping and is more suitable for the control of nonlinear dynamic systems than the SNN [11]. Recently, stable adaptive control for nonlinear dynamic systems using DNNs has emerged as one of the important research fields in NN applications. Rovithakis and Christodoulou *et al.* [12], [13] have contributed more in this respect. They presented direct and indirect adaptive control schemes based on a recurrent NN model of the unknown system. Lyapunov stability theory was also used to provide answers to the problem of stability, convergence and robustness. However, there are two shortcomings existing in the proposed algorithms:

1. The proposed control scheme assumes that the system states are within some compact set, which is also used in the other NN applications. Actually, without proving the stability of the whole system, the system states may be unbounded. Therefore, the NN approximation equation is not necessarily true during on-line learning.
2. In the DNN system, the perceptive inputs of the NN nonlinear components are the state of the constructed system, not the DNN system itself. So it is not a real

* Received November 5, 1999; accepted January 23, 2000. This work was supported jointly by National Natural Science Foundation of China under Grant 69934010 and by Science Foundation of the College of the Information and Technology of Tsinghua University.

† Dept. of Computer Science and Technology, State Key Lab of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, P.R. China. E-mail: {sfc, zlb}@s1000e.cs.tsinghua.edu.cn

‡ Dept. of Computer Science and Technology, State Key Lab of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, P.R. China. E-mail: szq-dcs@mail.tsinghua.edu.cn

Paper No. 1345–269X/99/020071-08

DNN system.

This paper is concerned with the stable adaptive control for robotic manipulators using DNNs. In the DNN systems constructed in this paper, the input of the NN nonlinear components in the DNNs is from the DNNs state itself such that a real DNN system is built. Furthermore, use of the dynamic inversion of the DNN system as the control input of the robot system make the state of the DNN system possible to be predetermined by design, which overcomes the demerit presented in item 1. Main theoretical results for designing such an adaptive controller of robotic manipulators are given. The effectiveness and efficiency of the proposed adaptive controller using DNNs are demonstrated in comparison studies with the adaptive control algorithm using SNNs by simulations of a two-link manipulator.

The paper is organized as follows. In Section 2, some basics for the robot model and its properties as well as those for controller design are reviewed. Then in Section 3, the stable adaptive control approach based on DNNs for robot trajectory tracking is given, where a complete control structure and the learning algorithms for the free adaptive parameters are presented. Stability and tracking error convergence proof is also given in this section. An application example is given in Section 4. Finally, Section 5 concludes the paper by highlighting the feature properties of the proposed DNN-based controller.

2. Preliminaries and Methodology Framework

2.1 Definitions

Let Z_+ be the natural number set and $\Re$ the real number set. Let $\Re^n$ be the n-dimensional vector space defined in $\Re$, and $\Re^{n\times n}$ be $n\times n$ real matrix space. The norms of a vector $\boldsymbol{x}=(x_1,\cdots,x_n)$ and a matrix $A=(a_{i,j})\in\Re^{n\times n}$ are defined as

$$\begin{cases}\|\boldsymbol{x}\| &= \sqrt{\boldsymbol{x}^T\boldsymbol{x}}\\ \|A\| &= \sqrt{\text{eig}(A^TA)_{\max}}\end{cases} \tag{1}$$

with $\text{eig}(\cdot)_{\max}$ the maximum eigenvalue. For any positive definite symmetric matrix $A(\boldsymbol{x})$, and for any $\boldsymbol{x}$, let A_m and A_M denote the minimum and maximum eigenvalues of $A(\boldsymbol{x})$, respectively. In addition, the following two norm forms are useful for the subsequent development

$$\begin{cases}\|\boldsymbol{x}\|_1 &= \sum_{i=1}^n |x_i|\\ \|A\|_F^2 &= \text{tr}(AA^T).\end{cases} \tag{2}$$

Definition 1: Consider the nonlinear system $\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}),\boldsymbol{y}=\boldsymbol{h}(\boldsymbol{x})$ where $\boldsymbol{x}$ is a state vector, $\boldsymbol{u}$ is the input vector and $\boldsymbol{y}$ is the output vector. The solution is uniformly ultimately bounded (UUB) if for all $\boldsymbol{x}(t_0)=\boldsymbol{x}_0$, there exists $\bar{\epsilon}>0$ and $T(\bar{\epsilon},\boldsymbol{x}_0)$ such that $\|\boldsymbol{x}(t)\|<\bar{\epsilon}$ for all $t\geq t_0+T$.

2.2 Models of robot dynamics

Consider the dynamic equation of a robot manipulator

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}}+C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+G(\boldsymbol{q})+F(\boldsymbol{q},\dot{\boldsymbol{q}})=\boldsymbol{u}(t) \tag{3}$$

where $\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}\in\Re^n$ are the vectors of generalized coordinates, velocity and acceleration, $M(\boldsymbol{q})\in\Re^{n\times n}$ the positive inertia matrix, $C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\in\Re^n$ the Coriolis and centrifugal torques, $G(\boldsymbol{q})\in\Re^n$ the gravitational torques, $\boldsymbol{u}(t)\in\Re^n$ the applied torque. $F(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the unstructured uncertainties of the dynamics including friction and other disturbances, which is assumed to be a continuous function of the robot joint angles. The following properties are required for the subsequent development.

Property 1: $M(\boldsymbol{q})$ is a positive symmetric matrix defined by $M_m\leq\|M(\boldsymbol{q})\|\leq M_M$ with M_m,M_M being known constants.

Property 2: $C(\boldsymbol{q},\dot{\boldsymbol{q}})$, defined by using the Christoffel symbols, satisfies that

- $\dot{M}(\boldsymbol{q})-2C(\boldsymbol{q},\dot{\boldsymbol{q}})$ is skew symmetric;
- $\|C(\boldsymbol{q},\dot{\boldsymbol{q}})\|\leq C_M\|\dot{\boldsymbol{q}}\|$ and $C(\boldsymbol{q},\boldsymbol{x})\boldsymbol{y}=C(\boldsymbol{q},\boldsymbol{y})\boldsymbol{x}$, $\boldsymbol{x},\boldsymbol{y}\in\Re^n$, $C_M>0$.

2.3 Design framework

The robot dynamic equation (3) can be written in state space form

$$\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x})+B(\boldsymbol{x})\boldsymbol{u}(t) \tag{4}$$

where $\boldsymbol{x}=[\boldsymbol{q}^T\ \dot{\boldsymbol{q}}^T]^T$ is the system state, and

$$\boldsymbol{f}(\boldsymbol{x})=\begin{bmatrix}\dot{\boldsymbol{q}}\\ -M^{-1}(\boldsymbol{q})(C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+G(\boldsymbol{q})+F(\boldsymbol{q},\dot{\boldsymbol{q}}))\end{bmatrix},$$
$$B(\boldsymbol{x})=\begin{bmatrix}0\\ M^{-1}(\boldsymbol{q})\end{bmatrix}. \tag{5}$$

To develop a stable NN-based adaptive control law, the tracking error metric [8] is defined as

$$S=C(\boldsymbol{x}-\boldsymbol{x}_d) \tag{6}$$

where $\boldsymbol{x}_d=[\boldsymbol{q}_d^T\ \dot{\boldsymbol{q}}_d^T]^T\in\Re^{2n}$ is the desired trajectories to be tracked, $C=[\Lambda\ I]\in\Re^{n\times 2n}$, $\Lambda=\Lambda^T>0$.

Differentiating (6) with respect to time and using (4) gives

$$\begin{aligned}\dot{S}&=C[\boldsymbol{f}(\boldsymbol{x})+B(\boldsymbol{x})\boldsymbol{u}(t)-\dot{\boldsymbol{x}}_d]\\ &=-\boldsymbol{r}S+\boldsymbol{r}S-C\dot{\boldsymbol{x}}_d+C\boldsymbol{f}(\boldsymbol{x})+CB(\boldsymbol{x})\boldsymbol{u}(t)\end{aligned} \tag{7}$$

where $\boldsymbol{r}=\text{diag}(r_1,\cdots,r_n)$, related to the closed-loop control bandwidth of the system, is usually defined as $\boldsymbol{r}=I-\bar{\boldsymbol{r}}$ with $\bar{\boldsymbol{r}}=\text{diag}(\bar{r}_1,\cdots,\bar{r}_n)$, in which $\bar{r}_i(i=1,\cdots,n)$ are determined according to (12a) in [15].

With $M^{-1}=CB(\boldsymbol{x})$, multiply M to both sides of (7) leads to

$$M\tilde{S}=M\boldsymbol{h}+W(\boldsymbol{x})+\boldsymbol{u}(t) \tag{8}$$

where $\tilde{S}=\dot{S}+\boldsymbol{r}S$, $\boldsymbol{h}=\boldsymbol{r}S-C\dot{\boldsymbol{x}}_d$, $W(\boldsymbol{x})=MC\boldsymbol{f}(\boldsymbol{x})$.

The following DNN is used to approximate the robot dynamics (5)

$$\dot{\hat{\boldsymbol{x}}}=\boldsymbol{f}_N(\hat{\boldsymbol{x}})+G_N(\hat{\boldsymbol{x}})\boldsymbol{u}(t) \tag{9}$$

where $\hat{\boldsymbol{x}}=[\hat{\boldsymbol{q}}^T\ \dot{\hat{\boldsymbol{q}}}^T]^T$ is the DNN state, $\boldsymbol{f}_N(\hat{\boldsymbol{x}})\in\Re^{2n}$, $G_N(\hat{\boldsymbol{x}})\in\Re^{2n\times n}$, defined as the nonlinear components of the DNN, are usually presented by two strictly feedforward NNs or two linearly parameterized NNs with $\hat{\boldsymbol{x}}$ as their perceptive inputs.

For a DNN system, the tracking error metric is defined as follows

$$S_o=C(\hat{\boldsymbol{x}}-\boldsymbol{x}_d) \tag{10}$$

where C is defined as before. Combining (9) and (10) gives

$$\begin{aligned}\dot{S}_o &= C(\dot{\hat{x}} - \dot{x}_d) \\ &= C\{\boldsymbol{f}_N(\hat{\boldsymbol{x}}) + G_N(\hat{\boldsymbol{x}})[\boldsymbol{u}(t) - \bar{\boldsymbol{u}}(t)] - \dot{\boldsymbol{x}}_d\} \\ &= -\boldsymbol{r}S_o + \boldsymbol{r}S_o - C\dot{\boldsymbol{x}}_d + C\boldsymbol{f}_N(\hat{\boldsymbol{x}}) \\ &\quad + CG_N(\hat{\boldsymbol{x}})[\boldsymbol{u}(t) - \bar{\boldsymbol{u}}(t)] \end{aligned} \tag{11}$$

where $\bar{\boldsymbol{u}}(t)$ presents the robust control components for robot control. Define $M_N = (CG_N(\hat{\boldsymbol{x}}))^{-1}$, and multiplying M_N to both sides of (11) yields

$$M_N\tilde{S}_o = M_N\boldsymbol{h}_N + W_N(\hat{\boldsymbol{x}}) + \boldsymbol{u}(t) - \bar{\boldsymbol{u}}(t) \tag{12}$$

where $\tilde{S}_o = \dot{S}_o + \boldsymbol{r}S_o$, $\boldsymbol{h}_N = \boldsymbol{r}S_o - C\dot{\boldsymbol{x}}_d$, $W_N(\hat{\boldsymbol{x}}) = M_NC\boldsymbol{f}_N(\hat{\boldsymbol{x}})$. When the DNN state $\hat{\boldsymbol{x}}$ approaches the robot state $\boldsymbol{x}$, we have $\bar{\boldsymbol{u}}(t) \to 0$, then the DNN system is equivalent to the robot system (4) in terms of state and control input.

Define the state deflection metric of the robot system from the DNN system as

$$\begin{aligned} S_e &= C(\boldsymbol{x} - \hat{\boldsymbol{x}}) \\ &= \Lambda(\boldsymbol{q} - \hat{\boldsymbol{q}}) + \dot{\boldsymbol{q}} - \dot{\hat{\boldsymbol{q}}} \end{aligned} \tag{13}$$

where $S_e = [s_{e1}, s_{e2}, \cdots, s_{en}]^T$, and C is defined as before.

If the following control is considered for a robot (4)

$$\boldsymbol{u}(t) = -M_N\boldsymbol{h}_N - W_N(\hat{\boldsymbol{x}}) + \bar{\boldsymbol{u}}(t) \tag{14}$$

then (12) can be written as

$$M_N(\hat{\boldsymbol{x}})\tilde{S}_o = \boldsymbol{0}. \tag{15}$$

Since $M_N(\hat{\boldsymbol{x}})$ is a positive definite matrix, the system behavior of the DNN can be predetermined by $\tilde{S}_o = \boldsymbol{0}$, i.e.,

$$\dot{S}_o + \boldsymbol{r}S_o = \boldsymbol{0}. \tag{16}$$

Subtracting (12) from (8) and using (16) lead to

$$\begin{aligned} M(\boldsymbol{q})\dot{S}_e = &-M(\boldsymbol{q})\boldsymbol{r}S_e + (M(\hat{\boldsymbol{q}}) - M_N(\hat{\boldsymbol{q}}))\boldsymbol{h} \\ &+W(\hat{\boldsymbol{x}}) - W_N(\hat{\boldsymbol{x}}) + Q + \bar{\boldsymbol{u}} \end{aligned} \tag{17}$$

where

$$\begin{aligned} M(\boldsymbol{q}) &= (M_{ij}(\boldsymbol{q})) \in \Re^{n\times n}, \\ M_N(\hat{\boldsymbol{q}}) &= (\hat{M}_{ij}(\hat{\boldsymbol{q}})) \in \Re^{n\times n}, \quad i,j = 1,\cdots,n \\ W(\hat{\boldsymbol{x}}) &= [w_1,\cdots,w_n]^T \in \Re^n, \\ W_N(\hat{\boldsymbol{x}}) &= [\hat{w}_1,\cdots,\hat{w}_n]^T \in \Re^n, \\ \boldsymbol{h} &= [h_1,\cdots,h_n]^T \in \Re^n, \\ Q &= [M(\boldsymbol{q}) - M(\hat{\boldsymbol{q}})]\boldsymbol{h} + W(\boldsymbol{x}) - W(\hat{\boldsymbol{x}}) \\ &\quad -M_N(\hat{\boldsymbol{q}})(\boldsymbol{h} - \boldsymbol{h}_N). \end{aligned} \tag{18}$$

The goal of the paper is to propose a novel control algorithm based on the DNN learning so as to make the DNN system defined in (9) approximate the robot dynamics presented in (4) as accurate as possible. Actually , we can achieve this by only making $S_e(t) \to 0$, $M_N(\hat{q}) \to M(\hat{q})$, and $W_N(\hat{x}) \to W(\hat{x})$ through the DNN learning. In the subsequent section, we will research the stable adaptive control approach for robot trajectory tracking using DNNs.

3. Robot Adaptive Control Using DNNs

Consider the following control law

$$\begin{cases} \boldsymbol{u}(t) &= \boldsymbol{u}_I(t) + \bar{\boldsymbol{u}}(t) \\ &= -M_N\boldsymbol{h}_N - W_N(\hat{\boldsymbol{x}}) + \bar{\boldsymbol{u}}(t) \\ \bar{\boldsymbol{u}}(t) &= \boldsymbol{u}_p(t) + \boldsymbol{u}_n(t). \end{cases} \tag{19}$$

It is shown by (19) that the robot control law is composed of the dynamic inversion $\boldsymbol{u}_I(t)$ of the DNN system, adaptive compensation term $\boldsymbol{u}_p(t)$ and the nonlinear control component $\boldsymbol{u}_n(t)$. The nonlinear control component $\boldsymbol{u}_n(t)$, which includes a proportional and derivative control (PD) component and a sliding control [14], is used to improve the system stability and compensate for inherent network approximation errors.

Since $M(\boldsymbol{q})$ and $W(\boldsymbol{x})$ are the continuous time functions of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, the following can be obtained by the expansion of the Taylor series

$$\begin{aligned} M(\boldsymbol{q})\boldsymbol{h} + W(\boldsymbol{x}) &= D_H\boldsymbol{x} + O_1(t), \\ M(\hat{\boldsymbol{q}})\boldsymbol{h} + W(\hat{\boldsymbol{x}}) &= D_H\hat{\boldsymbol{x}} + O_2(t), \\ D_H &= \left[\frac{\partial M(\boldsymbol{x})}{\partial \boldsymbol{x}^T} \quad \frac{\partial W(\boldsymbol{x})}{\partial \boldsymbol{x}^T}\right]\Bigg|_{\boldsymbol{x}=\boldsymbol{0}} \end{aligned} \tag{20}$$

which leads to

$$\begin{aligned} Q &= [M(\boldsymbol{q}) - M(\hat{\boldsymbol{q}})]\boldsymbol{h} + W(\boldsymbol{x}) - W(\hat{\boldsymbol{x}}) \\ &\quad +M_N(\hat{\boldsymbol{q}})(\boldsymbol{h} - \boldsymbol{h}_N) \\ &= D_H\boldsymbol{y}_H + O(t) + M_N(\hat{\boldsymbol{q}})(\boldsymbol{h} - \boldsymbol{h}_N) \end{aligned} \tag{21}$$

where $\boldsymbol{y}_H = [\bar{\boldsymbol{e}}^T h_1,\cdots,\bar{\boldsymbol{e}}^T h_n, \bar{\boldsymbol{e}}^T]^T \in \Re^{n_H}$, $\bar{\boldsymbol{e}} = \boldsymbol{x} - \hat{\boldsymbol{x}}$, $n_H = 2n(n+1)$, $D_H = [d_{H,1},\cdots,d_{H,n}]^T \in \Re^{n\times n_H}$, and the remaining term $O(t) = O_1(t) + O_2(t) = [o_1(t),\cdots,o_n(t)]^T$, is in small quantity when the DNN state is near the robot system state $\boldsymbol{x}$. Assume

$$\boldsymbol{u}_p(t) = -M_N(\hat{\boldsymbol{q}})(\boldsymbol{h} - \boldsymbol{h}_N) - \hat{D}_H\boldsymbol{y}_H \tag{22}$$

where $\hat{D}_H$ is the estimate of D_H.

Substituting (20) and (22) into (17) gives

$$\begin{aligned} M(\boldsymbol{q})\tilde{S}_e = &[M(\hat{\boldsymbol{q}}) - M_N(\hat{\boldsymbol{q}})]\boldsymbol{h} + W(\hat{\boldsymbol{x}}) - W_N(\hat{\boldsymbol{x}}) \\ &+O(t) + \tilde{D}_H\boldsymbol{y}_H + \boldsymbol{u}_n(t) \end{aligned} \tag{23}$$

where $\tilde{S}_e = \dot{S}_e + rS_e$, $\tilde{D}_H = D_H - \hat{D}_H$.

Since the function inputs $\hat{\boldsymbol{q}}, \dot{\hat{\boldsymbol{q}}}$ are bounded and $M_{ij}(\hat{\boldsymbol{q}}), w_i(\hat{\boldsymbol{x}})$ are continuous with their inputs, they can be represented by linearly parameterized NNs over some compact sets where $\hat{\boldsymbol{q}}, \dot{\hat{\boldsymbol{q}}}$ are defined, with approximation errors as small as possible by carefully choosing the NN structure, i.e.

$$M_{ij}(\hat{\boldsymbol{q}}) = \psi_{ij}^T\boldsymbol{y}_\psi(\hat{\boldsymbol{q}}) + v_{ij}, \quad w_i(\hat{\boldsymbol{x}}) = \varphi_i^T\boldsymbol{y}_\varphi(\hat{\boldsymbol{x}}) + \gamma_i \tag{24}$$

where $\boldsymbol{y}_\psi(\hat{\boldsymbol{q}}) = [y_{\psi,1},\cdots,y_{\psi,n_\psi}]^T \in \Re^{n_\psi}, \boldsymbol{y}_\varphi(\hat{\boldsymbol{x}}) = [y_{\varphi,1},\cdots,y_{\varphi,n_\varphi}]^T \in \Re^{n_\varphi}$ represent the basis function vectors, $\psi_{ij} \in \Re^{n_\psi}, \varphi_i \in \Re^{n_\varphi}$ are NN weights chosen as the values of $\hat{\psi}_{ij}$ and $\hat{\varphi}_i$ that minimize the NN approximation errors $v_{ij}(\hat{\boldsymbol{q}})$, $\gamma_i(\hat{\boldsymbol{x}})$ for all $\hat{\boldsymbol{q}} \in \Omega_1$, $\dot{\hat{\boldsymbol{q}}} \in \Omega_2$, where

$\Omega_1 \in \Re^n$, $\Omega_2 \in \Re^n$are sufficiently large compact sets. The following DNN function components are defined as

$$\hat{M}_{ij}(\hat{\boldsymbol{q}}) = \hat{\psi}_{ij}^T \boldsymbol{y}_\psi(\hat{\boldsymbol{q}}), \quad \hat{w}_i(\hat{\boldsymbol{x}}) = \hat{\varphi}_i^T \boldsymbol{y}_\varphi(\hat{\boldsymbol{x}}) \tag{25}$$

with $\hat{\psi}_{ij} \in \Re^{n_\psi}, \hat{\varphi}_i \in \Re^{n_\varphi}$, defined as the NN weight estimates.

Substituting (24) and (25) into (23) gives

$$\begin{aligned} M(\boldsymbol{q})\tilde{S}_e = & \begin{bmatrix} (\psi_{11} - \hat{\psi}_{11})^T & \cdots & (\psi_{1n} - \hat{\psi}_{1n})^T \\ \vdots & \cdots & \vdots \\ (\psi_{n1} - \hat{\psi}_{n1})^T & \cdots & (\psi_{nn} - \hat{\psi}_{nn})^T \end{bmatrix} \\ & \times \begin{bmatrix} h_1 \boldsymbol{y}_\psi(\hat{\boldsymbol{q}}) \\ \vdots \\ h_n \boldsymbol{y}_\psi(\hat{\boldsymbol{q}}) \end{bmatrix} \\ & + [(\varphi_1 - \hat{\varphi}_1), \cdots, (\varphi_n - \hat{\varphi}_n)]^T \boldsymbol{y}_\varphi(\hat{\boldsymbol{x}}) \\ & + \boldsymbol{\epsilon} + \boldsymbol{u}_n(t). \end{aligned} \tag{26}$$

Define

$$M(\boldsymbol{q})\tilde{S}_e = \tilde{\Theta}^T Y(\hat{\boldsymbol{x}}) + \boldsymbol{\epsilon} + \boldsymbol{u}_n(t) \tag{27}$$

where

$$\begin{aligned} \Theta^T &= \begin{bmatrix} \psi_{11}^T & \cdots & \psi_{1n}^T & \varphi_1^T & d_{H,1}^T \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \psi_{n1}^T & \cdots & \psi_{nn}^T & \varphi_n^T & d_{H,n}^T \end{bmatrix} \in \Re^{n \times n_e} \\ \hat{\Theta}^T &= \begin{bmatrix} \hat{\psi}_{11}^T & \cdots & \hat{\psi}_{1n}^T & \hat{\varphi}_1^T & \hat{d}_{H,1}^T \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{\psi}_{n1}^T & \cdots & \hat{\psi}_{nn}^T & \hat{\varphi}_n^T & \hat{d}_{H,n}^T \end{bmatrix} \in \Re^{n \times n_e}, \\ \tilde{\Theta} &= \Theta - \hat{\Theta} = [\tilde{\boldsymbol{\theta}}_1, \cdots, \tilde{\boldsymbol{\theta}}_n] = (\tilde{\theta}_{ij}) \in \Re^{n_e \times n}, \\ Y(\hat{\boldsymbol{x}}) &= \left[h_1 \boldsymbol{y}_\psi^T, \cdots, h_n \boldsymbol{y}_\psi^T, \boldsymbol{y}_\varphi^T, \boldsymbol{y}_H^T\right]^T \in \Re^{n_e}, \\ \boldsymbol{\epsilon} &= [\epsilon_1, \cdots, \epsilon_n]^T, \quad \epsilon_i = \sum_{j=1}^{n} v_{ij} h_j + \gamma_i + o_i, \\ \epsilon_m &= \sup \|\epsilon(t)\|_\infty, \quad n_e = n \times n_\psi + n_\varphi + n_H. \end{aligned} \tag{28}$$

Usually, $\epsilon_m \geq 0$ is an unknown quantity, and should be estimated with the estimation law to control the system. The estimate of ϵ_m is defined as $\hat{\epsilon}$ and the estimate error as $\tilde{\epsilon} = \epsilon_m - \hat{\epsilon}$. The following theorem gives a stable learning law, and guarantees that the tracking error metric of the robot is uniformly ultimately bounded.

The following theorem yields for stabilizing the closed loop system.

Theorem 1: Consider the n-link rigid robot manipulator described by the dynamic model given in (3), by applying the control law (19) with

$$\boldsymbol{u}_n(t) = -KS_e - \hat{\epsilon}\text{sgn}(S_e) \tag{29}$$

and the NN learning algorithm

$$\dot{\hat{\Theta}} = \eta[Y(\hat{\boldsymbol{x}})S_e^T + \sigma_1(\Theta_0 - \hat{\Theta})] \tag{30}$$

and the bound estimate algorithm

$$\dot{\hat{\epsilon}} = \eta_0[\|S_e\|_1 + \sigma_2(\epsilon_0 - \hat{\epsilon})] \tag{31}$$

where
$\eta = \text{diag}(\eta_1 I^{n_\psi}, \cdots, \eta_n I^{n_\psi}, \eta_{n+1} I^{n_\varphi}, \eta_{n+2} I^{n_H}) \in \Re^{n_e}$ is a learning rate matrix with $\eta_i > 0$, $I^m \in \Re^{m \times m}$denotes an identity matrix, $\sigma_1 > 0$, $\sigma_2 > 0$, Θ_0, ϵ_0 are design parameters. If the minimum eigenvalue of the positive symmetric matrix K, K_m, satisfy

$$K_m \geq \frac{1}{2}\left\|\dot{M}(\boldsymbol{q})\right\| \tag{32}$$

then the tracking error metric of the robot is uniformly ultimately bounded.

Proof: The following Lyapunov function candidate is considered

$$V = \frac{1}{2}S_e^T M(\boldsymbol{q})S_e + \frac{1}{2}S_o^T S_o + \frac{1}{2}\text{tr}(\tilde{\Theta}^T \eta^{-1}\tilde{\Theta}) + \frac{1}{2}\eta_0^{-1}\tilde{\epsilon}^2 \tag{33}$$

which satisfies

$$\begin{gathered} \frac{1}{2}P_m \boldsymbol{y}^T \boldsymbol{y} \leq V \leq \frac{1}{2}P_M \boldsymbol{y}^T \boldsymbol{y}, \\ \boldsymbol{y} = [S_e^T, S_o^T, \|\tilde{\Theta}\|, \tilde{\epsilon}]^T \\ P_m = \min\{M_m, 1.0, \eta_M^{-1}, \eta_0^{-1}\}, \\ P_M = \max\{M_M, 1.0, \eta_m^{-1}, \eta_0^{-1}\}. \end{gathered}$$

Using Properties 1 and 2, it is easy to obtain

$$\begin{aligned} \dot{V} = & \frac{1}{2}S_e^T \dot{M}(\boldsymbol{q})S_e + S_e^T[-M(\boldsymbol{q})\boldsymbol{r}S_e + \tilde{\Theta}^T Y(\hat{\boldsymbol{x}}) \\ & - KS_e + \boldsymbol{\epsilon} - \hat{\epsilon}\text{sgn}(S_e)] \\ & - S_o^T \boldsymbol{r} S_o - \text{tr}(\tilde{\Theta}^T \eta^{-1}\dot{\hat{\Theta}}) - \eta_0^{-1}\tilde{\epsilon}\dot{\hat{\epsilon}} \\ = & \frac{1}{2}S_e^T \dot{M}(\boldsymbol{q})S_e - S_e^T M(\boldsymbol{q})\boldsymbol{r}S_e - S_e^T K S_e + S_e^T \boldsymbol{\epsilon} \\ & - \hat{\epsilon}S_e^T \text{sgn}(S_e) - S_o^T \boldsymbol{r} S_o + \text{tr}[\tilde{\Theta}^T(-\eta^{-1}\dot{\hat{\Theta}} \\ & + Y(\hat{\boldsymbol{x}})S_e^T)] - \eta_0^{-1}\tilde{\epsilon}\dot{\hat{\epsilon}} \\ \leq & -\left(K_m - \frac{1}{2}\left\|\dot{M}(\boldsymbol{q})\right\|\right)\|S_e\|^2 - M_m r_m \|S_e\|^2 \\ & - r_m \|S_o\|^2 - \sigma_1 \text{tr}[\tilde{\Theta}^T(\Theta_0 - \hat{\Theta})] \\ & - \sigma_2 \tilde{\epsilon}(\epsilon_0 - \hat{\epsilon}). \end{aligned} \tag{34}$$

By completing the square, it can be shown that

$$\begin{aligned} -\sigma_1 \text{tr}[\tilde{\Theta}^T(\Theta_0 - \hat{\Theta})] = & -\frac{1}{2}\sigma_1 \|\tilde{\Theta}\|^2 - \frac{1}{2}\sigma_1 \|\hat{\Theta} - \Theta_0\|^2 \\ & + \frac{1}{2}\sigma_1 \|\Theta - \Theta_0\|^2 \end{aligned} \tag{35}$$

$$-\tilde{\epsilon}\sigma_2(\epsilon_0 - \epsilon_m) = -\frac{1}{2}\sigma_2 \tilde{\epsilon}^2 - \frac{1}{2}\sigma_2(\epsilon_0 - \hat{\epsilon})^2 + \frac{1}{2}\sigma_2(\epsilon_0 - \epsilon_m)^2. \tag{36}$$

Substituting (35) and (36) into (34) yields

$$\begin{aligned} \dot{V} \leq & -M_m r_m \|S_e\|^2 - r_m \|S_o\|^2 - \frac{1}{2}\sigma_1 \|\tilde{\Theta}\|^2 \\ & - \frac{1}{2}\sigma_2 \tilde{\epsilon}^2 + \gamma \\ \leq & -\lambda V / P_M + \gamma \end{aligned} \tag{37}$$

where

$$\begin{aligned} \lambda &= \min\{2M_m r_m, 2r_m, \sigma_1, \sigma_2\}, \\ \gamma &= \frac{1}{2}\sigma_1 \|\Theta - \Theta_0\|^2 + \frac{1}{2}\sigma_2(\epsilon_0 - \epsilon_m)^2. \end{aligned} \tag{38}$$

From (37), we can conclude that $\|S_e\|$ will eventually fall into a residual set with size $O(\gamma)$, this concludes our proof. ■

Remark 1: In Theorem 1, the design parameter Θ_0 can be considered as an initial estimate of the unknown weight Θ, allowing the designer to incorporate any prior parameter knowledge that may be available through off-line identification or other methods. As shown in (37), the closer Θ_0 is to its true value, the smaller the residual tracking error becomes. Besides, the adaptive law (30) and (31) incorporate a leakage term based on a variant of the σ-modification, which prevents drift of the free learning parameters.

Remark 2: $M(\boldsymbol{q})$ only contains trigonometric functions of $\boldsymbol{q}$, hence the derivative of each element with respect to $\boldsymbol{q}$ is bounded such that $\|\dot{M}(\boldsymbol{q})\|$ is bounded in (32).

Remark 3: Theorem 1 can be easily extended to the case using SNNs. If $S_e, Y(\hat{\boldsymbol{x}})$ in (19), (29)–(31) are replaced by $S, \bar{Y}(\boldsymbol{x})$ with

$$\bar{Y}(\boldsymbol{x}) = [h_1 \boldsymbol{y}_\psi^T(\boldsymbol{x}), \cdots, h_n \boldsymbol{y}_\psi^T(\boldsymbol{x}), \boldsymbol{y}_\varphi^T(\boldsymbol{x})]^T \in \Re^{\bar{n}_e},$$
$$\bar{n}_e = n \times n_\psi + n_\varphi \tag{39}$$

then the results given in theorem 1 will be suitable for stable adaptive control of robotic manipulators using SNNs. A similar derivation can be found in [9], [15].

4. Application

In this section, the above developed control approach is employed in the position control of a 2-link manipulator. The robot dynamics equation is

$$\begin{bmatrix} D_{11}(q_2) & D_{12}(q_2) \\ D_{12}(q_2) & D_{22}(q_2) \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} F_{12}(q_2)\dot{q}_2^2 + 2F_{12}(q_2)\dot{q}_1\dot{q}_2 \\ -F_{12}(q_2)\dot{q}_1^2 \end{bmatrix} + \begin{bmatrix} G_1(q_1, q_2)g \\ G_2(q_1, q_2)g \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{40}$$

where

$$\begin{aligned} D_{11}(q_2) &= (m_1 + m_2)\bar{l}_1^2 + m_2\bar{l}_2^2 + 2m_2\bar{l}_1\bar{l}_2\cos(q_2) + J_1 \\ D_{12}(q_2) &= m_2\bar{l}_2^2 + m_2\bar{l}_1\bar{l}_2\cos(q_2) \\ D_{22}(q_2) &= m_2\bar{l}_2^2 + J_2 \\ F_{12}(q_2) &= m_2\bar{l}_1\bar{l}_2\sin(q_2) \\ G_1(q_1, q_2) &= -(m_1 + m_2)\bar{l}_1\cos(q_1) - m_2\bar{l}_2\cos(q_1 + q_2) \\ G_2(q_1, q_2) &= -m_2\bar{l}_2\cos(q_1 + q_2) \end{aligned}$$

with

$$\|u_1\| \le 360\ [\text{kg}\cdot\text{m}^2/\text{s}], \quad \|u_2\| \le 182\ [\text{kg}\cdot\text{m}^2/\text{s}],$$
$$g = 9.8\ [\text{m}/\text{s}^2].$$

The physical parameters of the robot are

$$\bar{l}_1 = 1\ [\text{m}], \quad \bar{l}_2 = 0.8\ [\text{m}], \quad J_1 = J_2 = 5\ [\text{kg}\cdot\text{m}],$$
$$m_1 = 0.5\ [\text{kg}], \quad m_2 = 6.25\ [\text{kg}]. \tag{41}$$

The desired joint angle trajectory for the robot to follow is

$$q_{1d}(t) = 0.5[\sin(t) + \sin(2t)],$$
$$q_{2d}(t) = 0.5[\cos(3t) + \cos(4t)]. \tag{42}$$

And the initial simulation condition for robot motion is

$$q_1(0) = 1.0, \quad \dot{q}_1(0) = -0.5,$$
$$q_2(0) = 1.0, \quad \dot{q}_2(0) = -2.0. \tag{43}$$

In the following, the control performance of the proposed robot controller based on DNNs will be illustrated in comparison studies with the robot controller based on SNNs mentioned in Remark 3. In simulations, the design parameters of each controller are tuned to their best values, in terms of the conflicting requirements of tracking accuracy and controller stability, so that the best performances of these two types of controllers can be compared.

The controller based on DNNs (or SNNs) is designed according to (19), (29)–(32) (or Remark 3). Among them, the learning algorithm for DNNs is determined by (30), and the bound estimate algorithm is given by (31). It can be verified that the DNN state is strictly on the compact set $\Omega = [-1.5, 1.5] \times [-4.0, 4.0] \times [-1.5, 1.5] \times [-4.0, 4.0]$. For these, the centers of the DNN and SNN can be assigned in the same procedures.

In a four-dimension input space on Ω, with 11 levels for each input variable, there are 121 divisions to position the centers of basis functions according to the orthogonal design. Thus 121 basis functions are required to construct the nonlinear function $W_N(\hat{\boldsymbol{x}})$ (or $W_N(\boldsymbol{x})$). Similarly, in one dimensional input space $\hat{q}_2$ (or q_2) $\in [-1.5, 1.5]$ [rad] with 21 even partitions, 21 basis functions are required to approximate the nonlinear function $G_N^{-1}(\hat{\boldsymbol{q}})$ (or $G_N^{-1}(\boldsymbol{q})$) according to the orthogonal design. After the initial assignment of centers of basis functions following abovementioned procedures, centers can be further modified on line by the unsupervised competitive clustering algorithm [15] along with the robot simulations for trajectory tracking. The widths are determined by the P-nearest neighbor heuristics [15].

Simulations are done using a fourth-order Runge–Kutta algorithm with an integral step of 0.001 [s]. and a controller sampling interval $\delta = 0.02$ [s]. The same design parameters for both DNN and SNN controllers are chosen as

$$K = \text{diag}(80, 80), \quad \Lambda = \text{diag}(12, 12), \quad \eta_0 = 0.5,$$
$$\sigma_1 = 0.001, \quad \sigma_2 = 0.01, \quad \theta_0 = 0, \quad \epsilon_0 = 0.05. \tag{44}$$

The learning rates for DNN controller are chosen as

$$\eta_1 = 0.92, \quad \eta_2 = 0.93, \quad \eta_3 = 123.5, \quad \eta_4 = 100.5 \tag{45}$$

and the learning rates for SNN controller are chosen as

$$\eta_1 = 0.42, \quad \eta_2 = 0.43, \quad \eta_3 = 109.0 \tag{46}$$

and the initial learning rates for further updating the centers of the basis functions are chosen as

$$\alpha_\psi(0) = \alpha_\varphi(0) = 0.25.$$

Figures 1 to **4** present the angle tracking errors for two joints during the first 60 seconds and the last 10 seconds of operation for both DNN controller and SNN controller of robotic manipulators. **Figures 5** to **8** show the corresponding control torques. To evaluate the robustness against

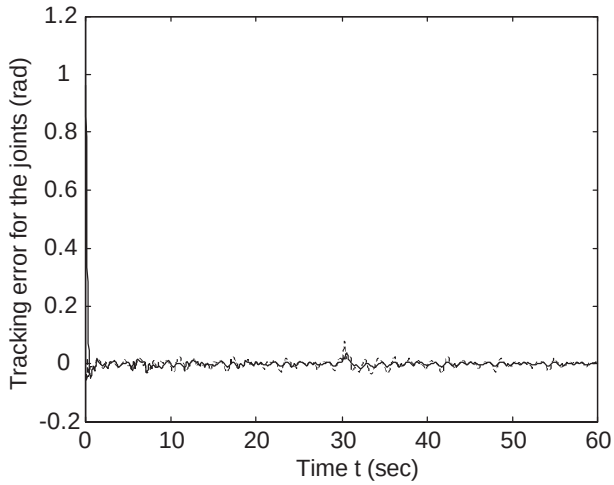

Fig. 1 Robot joint angle tracking errors for the first 60 seconds of operations using the DNN controller, where the solid line is for $q_1(t)$ and the dashed line is for $q_2(t)$

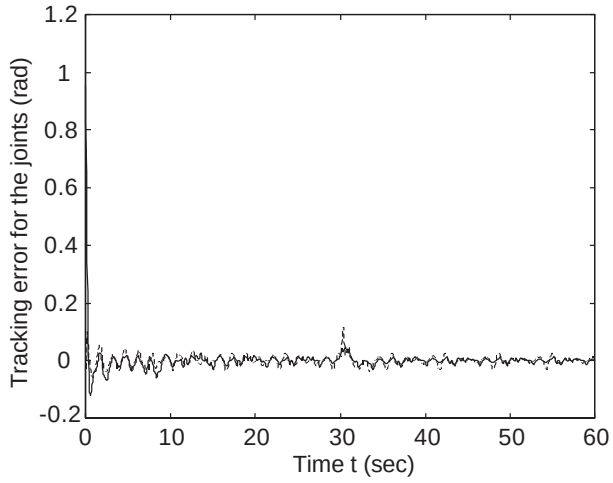

Fig. 3 Robot joint angle tracking errors for the first 60 seconds of operations using the SNN controller, where the solid line is for $q_1(t)$ and the dashed line is for $q_2(t)$

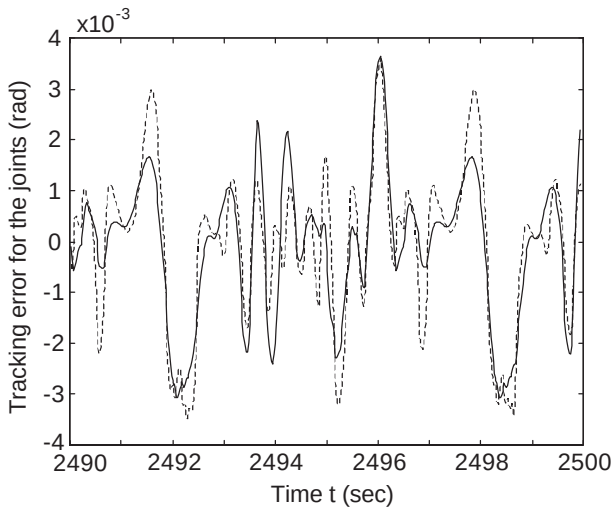

Fig. 2 Robot joint angle tracking errors for the last 10 seconds of operations using the DNN controller, where the solid line is for $q_1(t)$ and the dashed line is for $q_2(t)$

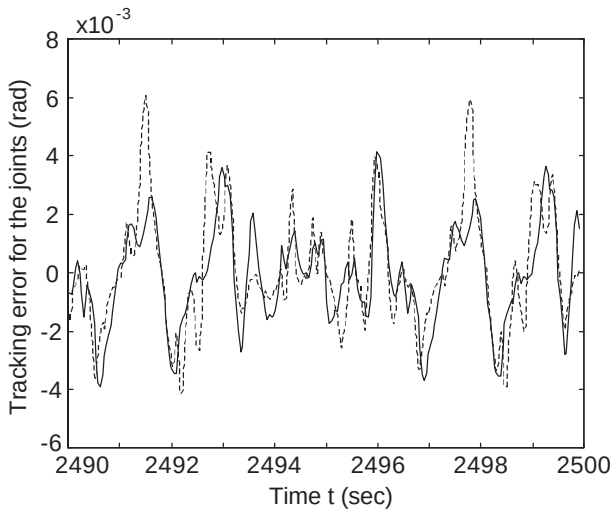

Fig. 4 Robot joint angle tracking errors for the last 10 seconds of operations using the SNN controller, where the solid line is for $q_1(t)$ and the dashed line is for $q_2(t)$

the external disturbances for both of the robot controllers, a disturbance control torque with magnitude of 200.0 is added to two joints of the robot during time interval [30, 35] in the simulations. **Figures 9** and **10** present bound estimations on the NN construction errors during the whole operation processes. It has been shown that the DNN controller results in a better control performance than the SNN controller for robot trajectory tracking. Besides, the results given in Figs. 1 and 2 also are superior to that shown in [16] for SNN-based robot control using the desired trajectory learning.

5. Conclusions

A stable adaptive controller design for robot trajectory tracking using DNNs has been developed in this paper, in which the proposed dynamic inversion of the DNN system plays a central role. One of the advantages of the method proposed is that it excludes the assumption that is often used in the existing literature, i.e., the robot states are assumed to be within a compact set. Actually, without proving the stability of the whole system, the robot joint values may be unbounded. Therefore, the approximation equation is not necessarily true during on-line learning. By using the dynamic inversion control, this problem is solved because the DNN state is normally bounded without noise. Besides, a simple estimation law for the bound on the NN reconstruction errors is added in the control scheme to exclude the bound estimation. It has been proved that the tracking error metric, the state deflection metric of the DNN system from the robot system, and the NN weights are convergent. The method is applied in an example to illustrate the remarkable performance improvement of the method.

References

[1] J. S. Albus, "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)," *J. of Dynamic Systems, Measurement and Control*, vol. 97, pp. 220–227, 1975.

[2] W. T. Miller III, F. H. Glanz, and L. G. Kraft III, "Application of a general learning algorithm to the control of robotic manipulators,"

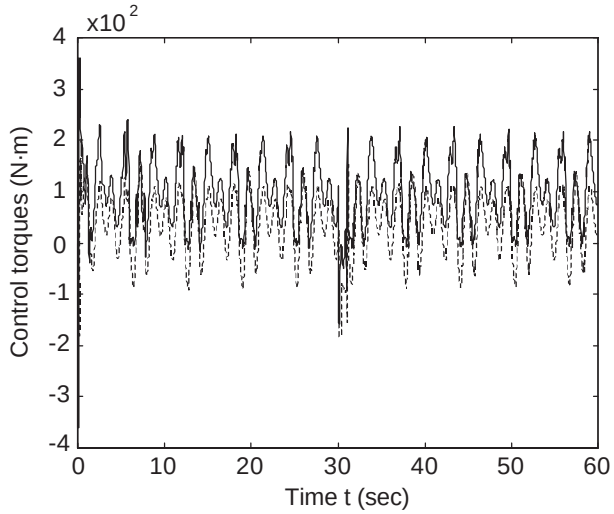

Fig. 5 Robot control torques for the first 60 seconds of operations using the DNN controller, where the solid line is for $u_1(t)$ and the dashed line is for $u_2(t)$

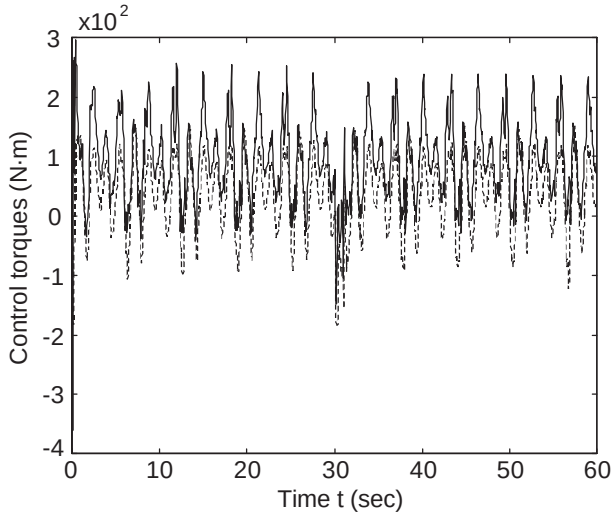

Fig. 7 Robot control torques for the first 60 seconds of operations using the SNN controller, where the solid line is for $u_1(t)$ and the dashed line is for $u_2(t)$

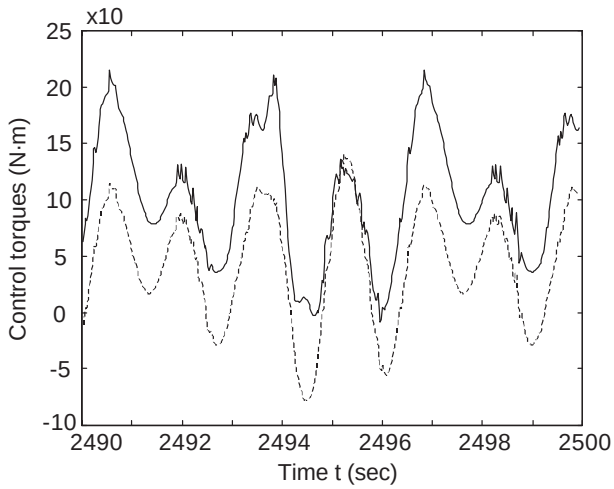

Fig. 6 Robot control torques for the last 10 seconds of operations using the DNN controller, where the solid line is for $u_1(t)$ and the dashed line is for $u_2(t)$

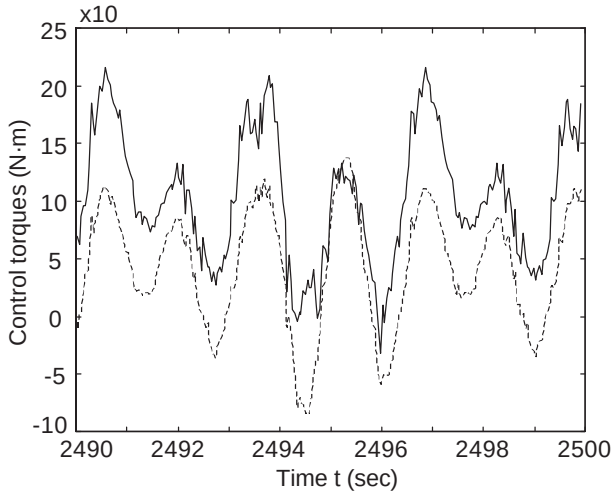

Fig. 8 Robot control torques for the last 10 seconds of operations using the SNN controller, where the solid line is for $u_1(t)$ and the dashed line is for $u_2(t)$

Int. J. of Robotics Research, vol. 6, no. 2, pp. 84–98, 1987.

[3] Y. Iiguni, H. Sakai, H. Tokumaru, "A nonlinear regulator design in the presence of system uncertainties using multilayer neural networks," *IEEE Trans. on Neural Networks*, vol. 2, no. 2, pp. 410–417, 1991.

[4] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE Control System Magazine*, vol. 8, no. 2, pp. 8–15, 1988.

[5] T. Ozaki, T. Suzuki, T. Furuhashi, S. Okuma, and Y. Uchikawa, "Trajectory control of robotic manipulator using neural networks," *IEEE Trans. on Industrial Electronics*, vol. 38, no. 3, pp. 195–202, 1991.

[6] F. L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controller with guaranteed tracking performance," *IEEE Trans. on Neural Networks*, vol. 6, no. 3, pp. 703–715, 1995.

[7] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural net robot controller: structure and stability proofs," *IEEE Trans. on Neural Networks*, vol. 7, no. 2, pp. 388–399, 1996.

[8] R. M. Sanner and J. J. E. Slotine, "Stable adaptive control of robot manipulators using 'neural' networks," *Neural Computation*, vol. 7, no. 3, pp. 753–790, 1995.

[9] F. C. Sun and Z. Q. Sun, "Stable sampled-data adaptive control of robot arms using neural networks," *J. of Intelligent and Robotic*

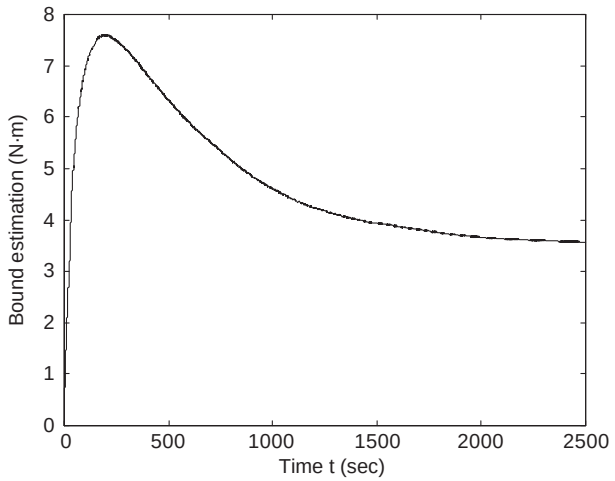

Fig. 9 Bound estimates for the DNN controller

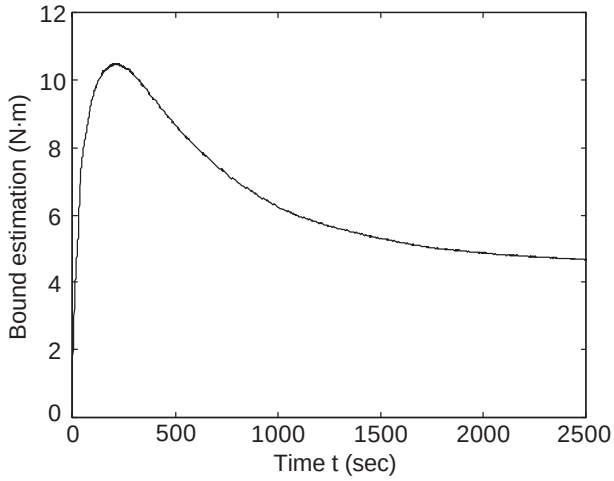

Fig. 10 Bound estimates for the SNN controller

Systems, vol. 20, no. 2, pp. 131–155, 1997.

[10] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice-Hall, 1996.

[11] T. W. S. Chow and Y. Fang, "A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics," *IEEE Trans. on Industrial Electronics*, vol. 45, no. 1, pp. 151–161, 1998.

[12] G. A. Rovithakis and M. A. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 400–412, 1994.

[13] G. A. Rovithakis, "Tracking control of multi-input affine nonlinear dynamical systems with unknown nonlinearities using dynamical neural networks," *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 29, no. 2, pp. 179–189, 1999.

[14] E. Tzirkel-Hancock and F. Fallside, "Stable control of nonlinear systems using neural networks," *Int. J. of Robust and Nonlinear Control*, vol. 2, pp. 63–86, 1992.

[15] F. C. Sun, Z. Q. Sun, and P. Y. Woo, "Stable neural network-based adaptive control for sampled-data nonlinear systems," *IEEE Trans. on Neural Networks*, vol. 9, no. 5, pp. 956–968, 1998.

[16] F. C. Sun, Z. Q. Sun, Y. Y. Zhu, and W. J. Lu, "Stable neuro-adaptive control for robots with unknown dynamics," *J. of Intelligent and Robotic Systems*, vol. 26, no. 1, pp. 91–100, 1999.

Biographies

Fuchun Sun was born in Jiansu Province, China, in 1964. He received the B.S., M.S. degrees from Naval Aeronautical Engineering Academy, Yantai, China, in 1986 and 1989, respectively, and Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 1998.

He is currently engaged in postdoctoral research in the Department of Automation, Tsinghua University, Beijing, China. His research interests include intelligent control, neural networks, fuzzy systems, variable structure control, nonlinear systems and robotics.

Zengqi Sun graduated from the Department of Automatic Control, Tsinghua University, China, in 1966 and received the Ph.D. degree in control engineering from the Chalmas University of Technology, Sweden, in 1981.

He is currently a professor of the Department of Computer Science and Technology, Tsinghua University, China. He is also a IEEE Senior Member, a executive member of IEEE Beijing Section, and a council member of the Chinese Association of Automation.

He is the author or co-author of over 200 papers and seven books on intelligent control and robotics. His current research interests include intelligent control, robotics, fuzzy systems, neural networks and evolutionary computing, etc.

Nan Li was born in Hubei Province, China, in 1978. She is currently an undergraduate student in the Department of Computer Science and Technology, Tsinghua University, Beijing, China.

Lingbo Zhang was born in Shandong Province, China, in 1971. He received the B.S. degree from Naval Engineering University, Wuhan, China, in 1993, and M.S. degree from the Department of Precision Instruments & Mechanics, Tsinghua University, Beijing, China, in 1999. He is currently a Ph.D. candidate in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include fuzzy control, measurement systems and robotics.