An Adaptive Neural Network Control Method for Robot Manipulators

Min-Jung Lee* · Young-Kiu Choi**

* Dept. of Electrical Engineering, Pusan National University, Pusan, 609-735, Korea
** School of Electrical and Computer Engineering, Pusan National University, Pusan, 609-735, Korea

Abstract -In recent years the neural network known as a sort of the intelligent control strategy is used as a powerful tool for designing control system since it has learning ability. But it is difficult for neural network controllers to guarantee the stability of control systems. In this paper we try connecting a radial basis function network to an adaptive control strategy. Radial basis function networks are simpler and easier to handle than multilayer perceptrons. We use the radial basis function network to generate control input signals that are similar to the control inputs of adaptive control using linear reparameterization of the robot manipulator. We adopt the saturation function as an auxiliary controller. This paper also proves mathematically the stability of the control system under the existence of disturbances and modeling errors.

1. Introduction

Generally, robot manipulators used as industrial automatic elements are known as systems with high nonlinearities that are often unknown and time-varying. If we want to design a controller for robot manipulator, we should consider the exact trajectory tracking performance for references input and the robustness for the existence of external disturbances. The conventional feedback controllers such as PID controllers, are commonly used in the field of industries because their control architectures are very simple and easy to implement. But when these conventional feedback controllers are directly applied to nonlinear systems, they suffer from the poor performance and low robustness due to the unknown nonlinearities and the external disturbances. During decades, to deal with the unknown nonlinearities and the external disturbances, various control strategies are proposed in the forms of the automatic tuning of PID control, variable structure control, feedback linearization, adaptive control, intelligent control, etc [1-4].

Recently, neural networks that are kinds of intelligent control methods have been investigated to control robot manipulators. The goal of the neural network controller is similar to an adaptive controller whose goal is to realize a robust controller under unknown time-varying system parameters. There are several different control strategies related to neural networks: learning control action based on reinforcement signals, learning the inverse dynamics, indirect adaptive control method based on neural network identification, and direct adaptive control method with guaranteed stability [5-10].

In this paper, we try connecting the radial basis function network (RBFN) to an adaptive control strategy. The RBFN is known as another candidate of neural networks. It is simpler and easier to handle than the multilayer perceptron that is conventionally used in neural network control strategies. Additionally, the RBFN is mathematically tractable, and its structure is similar to the fuzzy inference system. In the adaptive controller, the RBFN generates control input signals based on the Lyapunov method that is used in the conventional adaptive control. This paper uses the RBFN to approximate the robot dynamics that may be written in terms of the filtered tracking errors. The learning rule of RBFN is derived to guarantee the stability and robustness of the control system under the existence of disturbance and modeling errors. Experimental results for a SCARA-type manipulator show that the proposed adaptive controller based on the RBFN has the fast convergence time when compared to the multilayer neural network controller with the error back-propagation algorithm. And we show that the proposed method is more robust than the conventional controllers.

2. Robot Dynamics

The dynamics of an n-link robotic manipulator may be expressed in the Euler-Lagrangian form [2, 4].

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + \tau_d = \tau \qquad (1)$$

where $q \in R^n$ is the joint variable vector, $D(q)$ is the inertial matrix, $C(q,\dot{q})$ is the Coriolis/centrifugal matrix, and $G(q)$ is the gravity vector. The bounded disturbance is τ_d and the control input torque is τ.

In general, we have the following properties of the robot dynamics. They hold for all rigid-link manipulators [4].

Property 1: The inertia matrix $D(q)$ is symmetric and positive definite, and there exist scalars, d_1 and d_2, such that

$$d_1 I \le D(q) \le d_2 I$$

Property 2: The Coriolis/centrifugal matrix $C(q,\dot{q})$ is bounded by $c_b(q)\|\dot{q}\|^2$, with $c_b(q) \in C^1(S)$. S is a compact simply connected set of R^n.

Property 3: The matrix $\dot{D} - 2C$ is skew-symmetric, that is, the matrix is satisfied with $x^T(\dot{D} - 2C)x = 0$ for $\forall x \in R^n$. Strongly related to the skew symmetric property is the passive property.

Property 4: The unknown disturbance satisfies $\|\tau_d\| < b_d$, with b_d a known positive constant.

3. Radial Basis Function Networks

The locally tuned and overlapped receptive field is a well-known structure that has been studied in regions of cerebral cortex, visual cortex, and so on. Based on the biological receptive fields, Moody and Darken proposed a network structure, namely, a RBFN that employs local receptive fields to perform function mappings. The RBFN is known in the field of an approximation of non-linear function and pattern recognition. Especially the RBFN has a faster convergence property than the multiplayer neural network because the RBFN has simple architecture. Fig. 1 shows the schematic diagram of a RBFN with M receptive field units where the j-th receptive field unit is usually a Gaussian function or a logistic function [5,7, 9,10].

The output of a RBFN can be computed in two ways. For the simpler one, the output is a weighted sum of the function values associated with each receptive field. For the other one, the network produces the normalized response function as the weighted average of the strengths.

In this paper Gaussian function is selected as a basis function, and the output of the RBFN is calculated by the weighted sum method. The i-th output of the RBFN is defined by

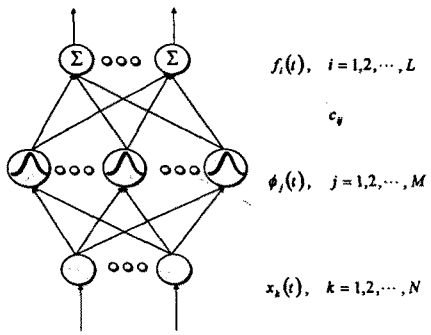

Fig. 1. Structure of the RBFN.

$$f_i = \sum_{j=1}^{M} c_{ij}\phi_j \quad i = 1,2,\cdots,L \tag{2}$$

$$\phi_j(x) = \exp\left(-\frac{\|x - m_j\|^2}{\sigma_j^2}\right)$$

where M and L are the number of hidden nodes and output nodes, respectively. Note that c_{ij} is the weight connecting the j-th hidden node to the i-th output node, $\phi_j(x)$ is the j-th basis function, $m_j \in R^n$ is the center vector and $\sigma_j \in R^n$ is the j-th standard deviation.

The system model based upon the RBFN with M nodes is expressed as:

$$f = c^T\Phi + \varepsilon \tag{3}$$

where

$$f = [f_1 \;\; f_2 \;\; \cdots \;\; f_L]^T,\quad \Phi = [\phi_1 \;\; \phi_2 \;\; \cdots \;\; \phi_M]^T,$$

$$c^T = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1M} \\ c_{21} & c_{22} & \cdots & c_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ c_{L1} & c_{L2} & \cdots & c_{LM} \end{bmatrix},\ \text{and}\ \varepsilon = [\varepsilon_1 \;\; \varepsilon_2 \;\; \cdots \;\; \varepsilon_L]^T.$$

The second element of the left side in (3) is an approximation error. From the approximation theorem, ε can be made very small. Therefore, we can set the norm of ε bounded by a known constant value [5,6], as follows:

$$\|\varepsilon\| \le \varepsilon_N. \tag{4}$$

4. Adaptive controller using rbfn

Fig. 2 shows the architecture of the adaptive neural network controller. In this structure, the RBFN makes control input signals for robot manipulators. The RBFN approximates the robot dynamics that may be written in terms of the filtered tracking errors. The adaptation laws for updating the weights of RBFN are derived mathematically to guarantees the stability of control system. And from the proposed method, the control inputs for robot manipulator are composed two parts: one is the RBFN approximator, and another is the robust term to guarantee the robustness under the existence of nonlinearities and the external disturbances. This approach is somewhat different from the conventional adaptive control scheme whose control inputs is included three part. One part is linear reparameterization in the unknown terms and the rests are explicitly expressed with tracking error term. The proposed robust term is usually used in the variable structure control.

Given a reference trajectory $q_d(t) \in R^n$, the tracking error $\tilde{q}(t)$ is

$$\tilde{q}(t) = q_d(t) - q(t) \tag{5}$$

and the filtered tracking error s and the control input τ are

$$s = \dot{\tilde{q}} + \Lambda\tilde{q} = \dot{q}_r - \dot{q} \tag{6}$$

$$\tau = \hat{c}^T\Phi(x) + Ksat(s) \tag{7}$$

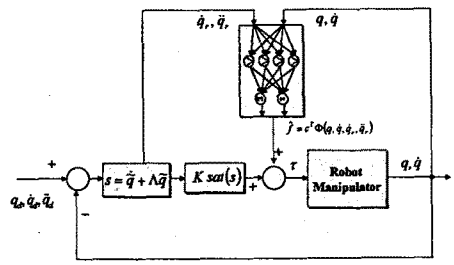

Fig. 2. Structure of the proposed controller.

where $x = [q \;\; \dot{q} \;\; \dot{q}_r \;\; \ddot{q}_r]$, $\dot{q}_r = \dot{q}_d + \Lambda\tilde{q}$, $\Lambda = \Lambda^T > 0$, and $K > 0$.

In (7), we select the saturation function including a deadzone. Fig. 3 shows the saturation function. From Fig. 3, the saturation function is approximated using two functions such as a signum function and a $\rho(|s_i|)$, as follows:

$$sat(s) = \sum_{i=1}^{n} \mathrm{sgn}(s_i)\cdot\rho(|s_i|) \tag{8}$$

$$\rho(|s_i|) = \begin{cases} 1 & if\ |s_i| \ge \delta \\ \dfrac{(1-\rho_{\min})}{\delta}|s_i| + \rho_{\min} & if\ 0 \le |s_i| \le \delta \end{cases}. \tag{9}$$

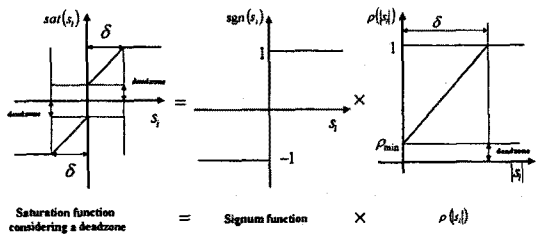

Fig. 3. Saturation Function.

To guarantee the stability of the total control system, we first select the positive-definite Lyapunov candidate function

$$V = \frac{1}{2}\left\{s^T Ds + tr\left(\tilde{c}^T\Gamma_1^{-1}\tilde{c}\right) + tr\left(\tilde{m}^T\Gamma_2^{-1}\tilde{m}\right) + tr\left(\tilde{\sigma}^T\Gamma_3^{-1}\tilde{\sigma}\right)\right. \tag{10}$$

where $\tilde{c} = c^* - \hat{c}$ is an error between the optimal weight c^* and an estimated weight $\hat{c}$. Γ_1, Γ_2, and Γ_3 are diagonal, symmetric and positive definite matrices.

Differentiating (10) with respect to time yields

$$\dot{V} = s^T M\dot{s} + \frac{1}{2}s^T\dot{M}s - tr\left(\tilde{c}^T\Gamma_1^{-1}\dot{\hat{c}}\right) - tr\left(\tilde{m}^T\Gamma_2^{-1}\dot{\hat{m}}\right) - tr\left(\tilde{\sigma}^T\Gamma_3^{-1}\dot{\hat{\sigma}}\right). \tag{11}$$

Differentiating $s(t)$ and using (1), the robot dynamics may be written in terms of the filtered tracking errors as

$$D\dot{s} = D\ddot{q}_r + C\dot{q}_r + G + \tau_d - Cs - \tau. \tag{12}$$

Substituting (12) into (10) yields

$$\dot{V} = s^T\left[D\ddot{q}_r + C\dot{q}_r + G + \tau_d - \tau\right] + \frac{1}{2}s^T\left(\dot{D} - 2C\right)s - tr\left(\tilde{c}^T\Gamma_1^{-1}\dot{\hat{c}}\right) - tr\left(\tilde{m}^T\Gamma_2^{-1}\dot{\hat{m}}\right) - tr\left(\tilde{\sigma}^T\Gamma_3^{-1}\dot{\hat{\sigma}}\right). \tag{13}$$

In property 3, $s^T\left(\dot{D} - 2C\right)s = 0$. In this paper, we set the robot dynamics written in terms of the filtered tracking errors to the nonlinear robot function expressed as:

$$f(q,\dot{q},\dot{q}_r,\ddot{q}_r) = D\ddot{q}_r + C\dot{q}_r + G \approx c^{*T}\Phi(q,\dot{q},\dot{q}_r,\ddot{q}_r) + \varepsilon. \tag{14}$$

From (14), the RBFN is used to approximate the nonlinear robot function. And substituting (7) and (14) into (13) yields

$$\dot{V} = s^T\left(\varepsilon + \tau_d - Ksat(s)\right) - tr\left(\tilde{c}^T\Phi s^T - \Gamma_1^{-1}\dot{\hat{c}}\right) - tr\left(\tilde{m}^T\Gamma_2^{-1}\dot{\hat{m}}\right) - tr\left(\tilde{\sigma}^T\Gamma_3^{-1}\dot{\hat{\sigma}}\right). \tag{15}$$

If we defined the adaptation laws such that

$$\dot{\hat{c}} = \Gamma\Phi(q,\dot{q},\dot{q}_r,\ddot{q}_r)s^T \tag{16}$$

$$\dot{\hat{m}} = -\Gamma_2\|s\|_1\hat{m} \tag{17}$$

$$\dot{\hat{\sigma}} = -\Gamma_3 \|s\|_1 \hat{\sigma} \tag{18}$$

then

$$\dot{V} = s^T(\varepsilon + \tau_d) - \sum_{i=1}^{n} K_i |s_i| \rho(|s_i|) + \|s\|_1 tr\{\tilde{m}^T(m^* - \tilde{m})\} + \|s\|_1 tr\{\tilde{\sigma}^T(\sigma^* - \tilde{\sigma})$$

$$\le \|s\|_1 \|\varepsilon + \tau_d\|_1 - \sum_{i=1}^{n} K_i |s|_i \rho_{\min} + \|s\|_1 tr\{\tilde{m}^T(m^* - \tilde{m})\} + \|s\|_1 tr\{\tilde{\sigma}^T(\sigma^* - \tilde{\sigma})\}$$

$$= \sum_{i=1}^{n} \left[|s_i| \{ \|\varepsilon + \tau_d\|_1 - K_i \rho_{\min} + tr\{\tilde{m}^T(m^* - \tilde{m})\} + tr\{\tilde{\sigma}^T(\sigma^* - \tilde{\sigma})\} \} \right]. \tag{19}$$

From the property of Frobenius norm, $tr\{\tilde{m}^T(u^* - \tilde{u})\}$ and $tr\{\tilde{\sigma}^T(\sigma^* - \tilde{\sigma})$ are defined by

$$tr\{\tilde{m}^T(m^* - \tilde{m})\} = \langle \tilde{m}, m^* \rangle_F - \|\tilde{m}\|_F^2 \le \|\tilde{m}\|_F \|m^*\|_F - \|\tilde{m}\|_F^2$$

$$tr\{\tilde{\sigma}^T(\sigma^* - \tilde{\sigma})\} = \langle \tilde{\sigma}, \sigma^* \rangle_F - \|\tilde{\sigma}\|_F^2 \le \|\tilde{\sigma}\|_F \|\sigma^*\|_F - \|\tilde{\sigma}\|_F^2$$

then

$$\dot{V} = -\sum_{i=1}^{n} \left[|s_i| \left\{ K_i \rho_{\min} - \|\varepsilon + \tau_d\|_1 + \left(\|\tilde{m}\|_F - \frac{m_{\max}}{2} \right)^2 - \frac{m_{\max}^2}{4} + \left(\|\tilde{\sigma}\|_F - \frac{\sigma_{\max}}{2} \right)^2 - \frac{\sigma_{\max}^2}{4} \right\} \right]. \tag{20}$$

When an approximation error term ε in (19) is replaced by (5) and a disturbance term τ_d by upper bound b_d, the derivative of Lyapunov function can be written. And if we define gain K_i that satisfied the following inequality:

$$K_i \ge \frac{\frac{m_{\max}^2}{4} + \frac{\sigma_{\max}^2}{4} + (\varepsilon_N + b_d)}{\rho_{\min}} \quad \forall i = i,2,\cdots,m \tag{21}$$

then yields

$$\dot{V} \le 0. \tag{22}$$

Lyapunov candidate function (10) is positive definite, and the differential form of (10) is negative semi-definite with the assumption of (221). According to the Lyapunov 2nd method, the total system is stable.

5. Experimental Results

In order to show the validation of the adaptive controller based on the RBFN, we applied it to the SCARA-type robotic manipulator shown in Fig. 4. And Fig. 5 shows the configuration to control the robot manipulator. The setup consist of an IBM-PC computer, a DSP processor board equipped with a TMS320C40 chip to calculate the control input on-line, a DIO board to acquire the error signals and position data, and a D/A board to send the command signal to the robot manipulator.

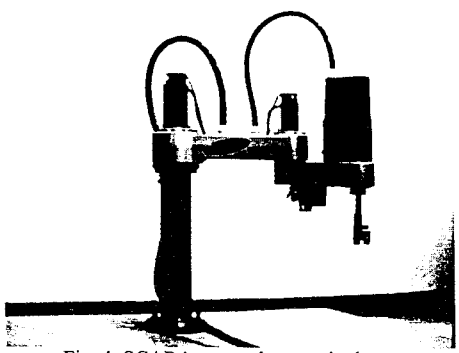

Fig. 4. SCARA-type robot manipulator.

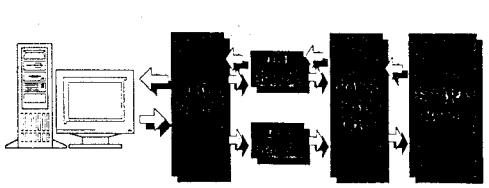

Fig. 5. Configuration of the robot system.

We set the control parameters λ=[8,9], $K = diag[1.8,1.9]$, Γ_1=0.7×I , $\Gamma_2 = \Gamma_3$ =0.001×I, σ=$diag$[0.5,0.5]. The number of neurons in the input layer is eight, and the number of neurons in the hidden layer is fifteen. The sampling time in this experiment is set to 5[ms]. We have four different experiments. The first one is that the reference trajectories have the frequency, $\omega = 1.88$ [*rad/sec*]. And the second one is 2 times faster than the first case, $\omega = 3.76$ [*rad/sec*]. The third is 4[kg] load with the same frequency of the second case. Finally, we consider the circle trajectory in the Cartesian space shown in Fig. 9. Additionally, we compared two different controllers, the PID controller and the series neuro-control using multilayer perceptron.

First, the reference input trajectories are defined by $q_1^r(t) = 0.4\cos(1.88t)[rad]$, and $q_2^r(t) = 0.4\{\sin(1.88t)+1\}[rad]$ for the joints 1 and 2, respectively. Fig. 6 (a) and (b) show the errors of the PID controller, the multilayer neural network and the proposed controller. In the case of the series neuro-control using multilayer perceptron., the tracking errors are reduced during the learning process but the rate of reduction of the error is very slow. By the way the proposed method is faster that the series neuro-control using multilayer perceptron.

Another reference input trajectories are defined as $q_1^r(t) = 0.4\cos(3.76t)[rad]$, and $q_2^r(t) = 0.4\{\sin(3.76t)+1\}[rad]$ for the joints 1 and 2, respectively, and applied to the robot manipulator. Fig. 7 shows the tracking errors. From Fig. 7, the performance of the PID controller is deteriorated since the trajectories are 2 times faster than the first trajectories. However, the errors of the series neuro-control using multilayer perceptron and those of the proposed controller are reduced as the time goes on. The error reduction rate of the proposed method is faster than that of the series neuro-control using multilayer perceptorn.

To show the robustness and performance of the proposed method, we have 4*kg* load and the circle trajectory on the Cartesian axes. Fig. 8 (a) and (b) show the tracking errors for 4*kg* load when the reference input trajectories are set to $q_1^r(t) = 0.4\cos(3.76t)[rad]$ and $q_2^r(t) = 0.4\{\sin(3.76t)+1\}[rad]$ for the joints 1 and 2, respectively. During the learning process, Fig. 8 shows the series neuro-control using multilayer perceptorn and the proposed method is less sensitive to the disturbance than the PID controller.

Fig. 10 shows the tracking errors when the circle input trajectories are applied to the manipulator. The proposed method shows that tracking errors are reduced vary fast.

6. Conclusions

In this paper, we try connecting the RBFN to an adaptive control strategy. The RBFN is known as another candidate of neural networks. The structure of proposed method is composed of two parts. The first part is the RBFN approximator. The RBFN generates control input signals based on the Lyapunov method that is used in the conventional adaptive control method. We set the robot dynamics to the nonlinear robot function expressed by the filtered tracking errors, and used the RBFN to approximate the nonlinear robot function. This proposed method is somewhat different from the conventional adaptive control schemes that used the linear reparameterization of robot dynamics. The adaptation laws of the RBFN are calculated to guarantee the stability of the control system based on the Lyapuove method. And we adopt the saturation function as an auxiliary controller in order to guarantee the stability and robustness of control system. Finally, Experimental results show that the proposed adaptive controller is more adaptable to the environment changes and is

more robust than the conventional PID controller and series neuro-controller using the multilayer perceptron.

[References]

[1] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, Prentice-Hall, 1991.

[2] F.L. Lewis, C. T. Abdallah, and D. M. Dawson, *Control of Robot Manipulators*, Macmillian Publishing Company, 1993.

[3] Sigeru Omatu, Marzuki Khalid, and Rubiyah Yusof, *Neuro-Control and Its Applications: Adavances in Industrial Control*, Springer-Verlag, 1996.

[4] Frank L. Lewis, Kai Liu, and Aydin Yesildirek, "Neural net robot controller with guaranteed tracking performance," *IEEE Trans. on Neural Networks*, vol. 6, no. 3, May 1995.

[5] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 4-27, March 1990.

[6] A. S. Morris and S. Khemaissia, "A neural network based adaptive robot controller," *Journal of Intelligent and Robotic Systems*, vol. 15, pp. 3-10, 1996.

[7] Lin-Xin Wang, "Stable adaptive fuzzy controllers with application to inverted pendulum tracking," *IEEE Trans. on Systems, Man, and Cybernetics-Part b: Cybernetics*, vol. 26, no. 5, October 1996.

[8] R. M. Sanner and J.-J.E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. on Neural Networks*, vol. 3, no. 6, November 1992.

[9] Lin-Xin Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, Prentice-Hall, 1994.

[10] J.-S.R. Jang, C.-T.Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, 1997.

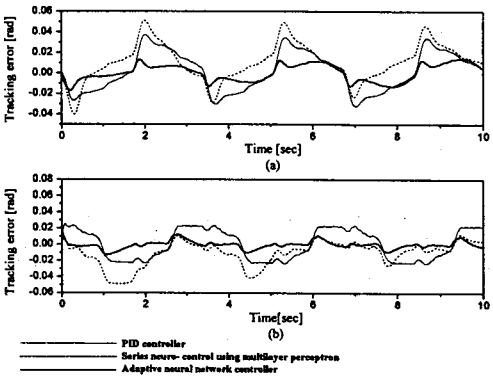

Fig. 6. Tracking errors under reference inputs with $\omega = 1.88[rad/\sec]$.(a) Joint 1. (b) Joint 2.

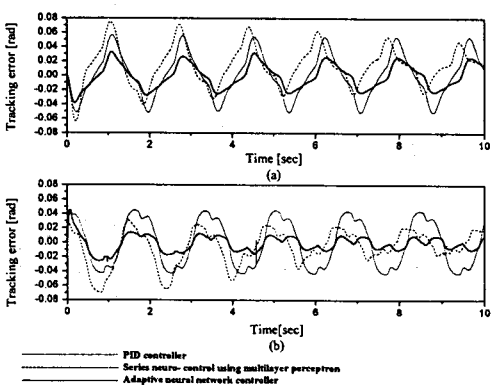

Fig. 7. Tracking errors under reference inputs with $\omega = 3.76[rad/\sec]$.(a) Joint 1. (b) Joint 2.

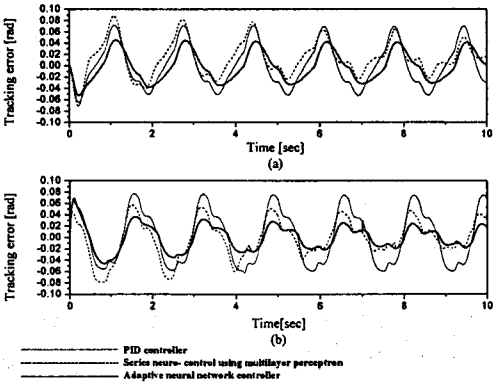

Fig. 8. Tracking errors under disturbance torque. (a) Joint 1. (b) Joint 2.

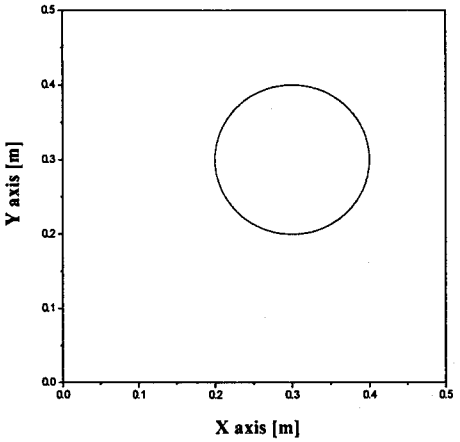

Fig. 9. Circle trajectory in the Cartesian space.

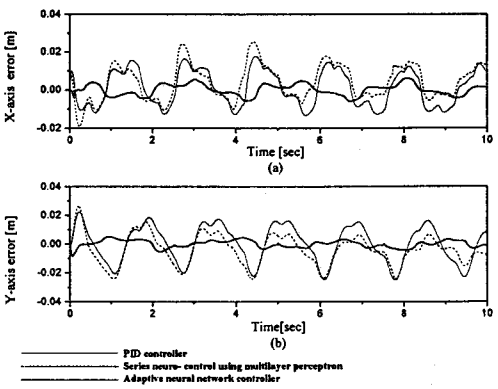

Fig. 10. Tracking errors for the circle trajecory in the Cartesian space. (a) Joint 1. (b) Joint 2.