Proceedings of International Conference
on Artificial Intelligence

Control Optimization of Robotic Manipulators Using Reinforcement Learning

Kai-Tai Song and Wen-Yu Sun
Department of Control Engineering
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

Abstract

Conventional robot force control schemes are basically model-based methods. However, it is very difficult to obtain exact modelling of robot dynamics and there are various uncertainties in the contact environment. In this paper, we propose a reinforcement learning control approach to overcome these drawbacks. We used artificial neural network (ANN) as the learning structure, and applied the stochastic real-valued (SRV) unit as the learning method. Firstly, simulations of force tracking control of a two-link robot arm were carried out to verify the control design. The results show that even without the information of the robot dynamic model and environment states, operation rules for simultaneous controlling force and velocity can be achieved via repetitive exploration. However, the learning speed was quite slow. Therefore in practical control applications, our approach is to enhance the performance optimization. We propose to combine a conventional controller with the reinforcement learning strategy. Finally, experimental results are presented to demonstrate the proposed method.

1 Introduction

Industrial robots are used widely in factories for welding, painting and material transfer. Attempts are now to put them to more difficult tasks such as assembling, grinding and deburring. In all these operations, the robot must come into physical contact with the environment and the inclusion of force information in the control of robot motion is therefore necessary. A number of force control schemes have been proposed [6,9]. These force control schemes almost all assume that the dynamic model of the manipulator and the environment states (shape and stiffness) are known for the designer. However, this is not always true in practice. Therefore, these model-based control schemes have two problems in common: 1) the uncertainty of the dynamic modeling of the manipulator and 2) the uncertainty of the environment states.

Learning control is a control method which can achieve desired control performance when a priori information of the system is unknown or incompletely known. In this approach the performance is improved by iterative practicing. Song and Chu [11] had used reinforcement learning for force tracking of an industrial robot. Under no information about the environment states, satisfactory force tracking results were obtained under environment variations. But in that experiment the learning controller did not learn the dynamics of the manipulator. To have a complete solution of the problem, we propose in this paper a reinforcement learning control design which not only learn the environment states but also the dynamics of the manipulator. Nevertheless, the learning speed would be very slow, it would be very difficult to apply the reinforcement learning scheme for real-world problems. For practical applications, we propose alternatively a strategy to combine a conventional PID controller with the reinforcement scheme to achieve performance optimization under load disturbance. The rest of this paper is organized as follows. In section 2, we introduce the reinforcement learning control structure. In section 3, a controller design based on *SRV* unit is described. Section 4 presents simulation results. Practical experimental results are presented in section 5 to demonstrate the possibility of reinforcement learning in real-world applications. We conclude in section 6.

2 Reinforcement learning

Barto[2] pointed out that reinforcement learning essentially involves two main issues: 1) how to translate an overall performance evaluation of series of control actions into immediate performance evaluation of the individual control actions, and 2) how to update the controller based on the immediate evaluations. In this study, because the controller is immediately provided

with a performance evaluation for each chosen control action from the environment and these evaluation signals are directly used to improve the control performance, we will focus on the second problem.

Fig. 1 is the basic reinforcement learning concept. At time step t, the controller receives a vector of state inputs $x(t)$ from the environment $X \subseteq \Re^n$, where $\Re$ is the set of real numbers. The controller provides an output $y(t) \in Y \subseteq \Re^m$ based on the current control law $y(x)$. The critic evaluates the output $y(t)$ in the context of input $x(t)$ and sends to the controller an evaluation signal $r(t) \in \Re$. The evaluation signal is called reinforcement. The reinforcement $r(t)$ is determined by the critic according to an external reinforcement function $r(x(t), y(t)) \in \Re$. It is assumed that there exists an unique function $y^*(x)$ that optimizes the reinforcement over the input space. The function $y^*(x)$ is called the optimal law. Hence the objective of the reinforcement learning is to learn the optimal controller. In order to improve the performance, it is necessary to acquire the gradient of the reinforcement function in terms of control actions for adjustment of the controller's parameters. Since the function of the reinforcement is unknown to the controller, reinforcement learning algorithms are acquired to have an approach to estimate the gradient of the reinforcement value. Werbos [13] divided reinforcement learning algorithms into two main approaches to find the gradient: 1) direct approach and 2) indirect approach. In this paper, we chose the direct approach as our learning method. We introduce it in the next section.

3 Control design for robot force tracking

In the direct approach, the controller actively explores the control action space and finds gradient information of the reinforcement function by comparing reinforcement values in the control space. It is often referred to as *stochastic reinforcement learning*. In early direct approaches, such as learning automata[8] and the associative reward-penalty algorithm[3], the controller only has discrete action spaces. It can not satisfy many practical control applications that require continuous control signals. A stochastic reinforcement learning algorithm for learning function with continuous outputs was proposed by Gullapalli[5]. The algorithm is based on the stochastic real valued (SRV) unit. It contains two parts. The two parts simultaneously learn in real-time. Random actions that might lead to better performance must be tried. In order to enable the exploration, the output of unit 1 μ is treated as mean of the desired stochastic action. The output of unit 2 is an estimation of the expected value of the reinforcement signal, which is used to compute the standard deviation σ. This can be interpreted as a search-extent for a better action. The random search is accomplished by taking the normal distribution $N(\mu, \sigma)$ for each input. A normal distributed random variable a is produced. An output function f maps a to the control output y. Fig.2 shows the robot force tracking control design based on stochastic reinforcement learning. Since the original SRV structure is for single output learning element, we put two of them in parallel to learn respectively the joint 1 torque and joint 2 torque. Because single performance evaluation is required, there is only one reinforcement predictor (unit 2). The learning algorithms of unit 1 and unit 2 are discussed below.

3.1 The learning element

Unit 1 is the learning element. In general, its output can be expressed as:

$$y(k) = N_c[W, x(k)] \tag{1}$$

where N_c represents the controller network, W is the weight matrix. Our aim is to obtain optimal control actions in terms of the critic r, therefore the weight parameters are adjusted by:

$$\omega(k+1) = \omega(k) + \triangle\omega(k) \tag{2}$$

and

$$\begin{aligned} \triangle\omega(k) &= \alpha\frac{\partial r(k+1)}{\partial\omega(k)} \\ &= \alpha\frac{\partial r(k+1)}{\partial\mu(k)}\frac{\partial\mu(k)}{\partial\omega(k)} \end{aligned} \tag{3}$$

where α is the learning rate, ω is an element of W. Because the system model is unobtainable, therefore we can not have the derivative of critic r relative to the mean μ, $(\frac{\partial r(k+1)}{\partial\mu(k)})$, Gullapalli[5] proposed a random search approach to solve this problem. Suppose the system critic is a valve between 0 and 1, and the smaller the better performance, then we have:

$$\frac{\partial r(k+1)}{\partial\mu(k)} = (\hat{r}(k) - r(k+1))\frac{(a(k) - \mu(k))}{\sigma(k)} \tag{4}$$

If the actual critic is smaller than the predicted critic, or when $\hat{r}(k) - r(k+1) > 0$, the direction of searching is correct, so the mean value $\mu(k)$ is adjusted toward $a(k)$. Conversely, if the actual critic is larger than the predicted critic, then $\mu(k)$ is adjusted away

from $a(k)$. Standard deviation σ can be treated as search amount; when $\hat{r}$ is smaller, then the system performance is better, and the control action is closer to the ideal value, so the search amount should be smaller, in other wards, the standard deviation σ becomes smaller. This means the function s linking $\hat{r}$ and σ should be a monotonically increasing function of $\hat{r}$.

3.2 The reinforcement predictor

Unit 2 predicts(tracks) the evaluation feedback. An ANN, for instance, can be used for this purpose. The predicted critic can be expressed as:

$$\hat{r}(k) = N_e[V, x(k)] \tag{5}$$

Where N_e represents the critic network, $[x(k), \hat{r}]$ is the input-output pair at time instant k, V is the weight matrix of the ANN. The method of least mean squre(LMS) is usually used as the learning rule for adjusting the weight parameters. The cost function is :

$$E = \frac{1}{2}\sum_k [r(k+1) - \hat{r}(k)]^2 \tag{6}$$

The updating rules are:

$$v(k+1) = v(k) + \triangle v(k) \tag{7}$$

and

$$\triangle v(k) = \beta(r(k+1) - \hat{r}(k))\frac{\partial \hat{r}(k)}{\partial v(k)} + \lambda \triangle v(k-1) \tag{8}$$

where β is the learning rate, λ is the momentum constant added to speed-up convergence, which are positive and less than one; v is an element of V.

The SRV unit only solves the problem of reinforcement learning. It is non-associative. In practice, memory is an important element in learning control design. The learning controller must remember the correct mapping of the input and output variables via learning. There are many structures for associative memory, such as boxes system[7], cerebellar model articulation controller (CMAC)[1], artificial neural networks(ANN) and fuzzy-neural networks. In these approaches, boxes system and CMAC approaches require proper partition of the input space, and fuzzy-neural network structure requires the set-up of the fuzzy rules and membership functions[12]. ANN have been used for associative memory in many applications. Any nonlinear mapping can be accomplished by a three layered feedforward network if the hidden layer unit is adequate. The generalization property makes ANN popular, we adopted it as the learning structure and train it by error-backpropagation algorithm[10]. Fig. 3 shows the ANN structure used in the reinforcement learning controller. It has two hidden layers. Each hidden layer has 12 processing elements. There are six inputs for the ANN: force error, velocity error, angular position of joint 1 and joint 2 (θ_1, θ_2), angular velocity of joint 1 and joint 2 ($\dot{\theta}_1$ and $\dot{\theta}_2$). The control actions are the torques for the two joint motors. The activation functions of neuron f_1 and neuron f_2 are

$$f_1(x) = f_2(x) = \frac{1}{1+e^{-2x}}. \tag{9}$$

The activation functions f_3 and f_4 are

$$f_3(x) = (1 - \frac{2}{1+e^{-}x}) \times 6 \tag{10}$$

$$f_4(x) = (1 - \frac{2}{1+e^{-}x}) \times 3. \tag{11}$$

The functions $s_1$and s_2 are linear function

$$s_1(x) = s_2(x) = 0.15x. \tag{12}$$

4 Simulation results

To verified the proposed design, computer simulations for force tracking control of a two-link robot arm have been carried out. The SCARA type robot manipulator has two parallel links. The length and mass of each link are: $l_1 = 0.4m, l_2 = 0.3m, m_1 = 15kg, m_2 = 3kg$. We assumed the environment can be modeled as a stiffness. In other words, the normal force is proportional to the difference between environment surface position and set-point position of the end-effector. Under the condition of no prior information about the robot model and environment, the controller was set to learn the relationship between the dynamics of the robot arm and the environment stiffness. The dynamic model used in these simulations is described below. For a robot manipulator with two parallel-link (without gravity terms), we have:

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) \tag{13}$$

where $M(\Theta)$ is the inertial term, and $V(\Theta, \dot{\Theta})$ is the nonlinear term:
$M(\Theta) =$

$$\begin{bmatrix} l_2^2 m_2 + 2l_1 l_2 m_2 \cos\theta_2 + l_1^2(m_1+m_2) & l_2^2 m_2 + l_1 l_2 m_2 \cos\theta_2 \\ l_2^2 m_2 + l_1 l_2 m_2 \cos\theta_2 & l_2^2 m_2 \end{bmatrix} \tag{14}$$

$$V(\Theta, \dot{\Theta}) = \begin{bmatrix} -m_2 l_1 l_2 \dot{\theta_2}^2 \sin\theta_2 - 2m_2 l_1 l_2 \dot{\theta_1}\dot{\theta_2}\sin\theta_2 \\ m_2 l_1 l_2 \dot{\theta_1}^2 \sin\theta_2 \end{bmatrix} \tag{15}$$

In Cartesian coordinate system, the dynamics equation can be expressed as:

$$f = M_x(\Theta) + V_x(\Theta, \dot{\Theta}) \tag{16}$$

where

$$\begin{aligned} f &= J^{-T}(\Theta)\tau, \\ M_x(\Theta) &= J^{-T}(\Theta)M(\Theta)J^{-1}(\Theta), \\ V_x(\Theta, \dot{\Theta}) &= J^{-T}(\Theta)(V(\Theta, \dot{\Theta}) - M(\Theta)J^{-1}(\Theta)\dot{J}(\Theta)\dot{\Theta}). \end{aligned}$$

In the simulations, the desired contact force and velocity were: $F_d = 1.0N$, $V_d = 5cm/s$ respectively. The reinforcement signal was

$$r = max(1.0, \frac{1}{2}(\mid F_d - F \mid + \mid \frac{(V_d - V)}{5} \mid)) \tag{17}$$

If r=1, the failure signal would be recognized for safety reason, preventing the damage of the tool. The system would reset to initial condition($\theta_1 = 45°, \theta_2 = -45°$) as this occurred. The stiffness of the contact environment was $100N/m$. We have tested the control design using two environment configurations: 1) a line path of $50cm$ and 2) a circular path of $8cm$ radius. Only the results of circular path, which is more difficult to track, are presented here. Fig. 4 shows the time when the robot arm had moved before the failure occurred. Note that the robot was set to move 10 seconds. Fig. 5 and Fig. 6 show force and velocity response respectively. Fig. 7 shows the trajectories of joint 1 and joint 2. From the above simulation results, it can be seen that the ANN controller have learned the dynamic relationship between the robot manipulator and the environment. However, the learning speed is slow. This is a great problem in practical applications.

5 Proposed approach

For demonstrating the possibility of using reinforcement learning control scheme in real-world applications, and taking structure safety into consideration, we propose a new control strategy and apply reinforcement learning to performance optimization. It is similar to the idea of Franklin[4]. Fig. 8 is the control block diagram. Conventional PID controllers are implemented to control the two-link robot arm to track a desired trajectory with satisfactory performance. It is well known that there exists nonlinearity and uncertainties in robot tracking control. Furthermore, load-disturbance will deteriorate the performance. We proposed this control scheme to improve the tracking performance by adding an on-line learning controller. To eliminate the interference between the PID controllers and the learning controller, we design the inputs of learning controller to include e(k), e(k–1), e(k-2), u(k-1), θ and $\dot{\theta}$, where e(k), e(k-1), e(k-2) and u(k-1) are also the inputs of the PID controllers. It is desirable that the learning controller estimates the output of the PID controllers and compensates an adaptive torque. Fig. 9 shows the experimental setup. The laboratory-made two-link SCARA type manipulator was powered by two direct-drive motors. The control scheme was implemented on a dSPACE DSP controller card. Two experiments have been conducted for verification of the proposed control scheme.

5.1 Experimental results of one-link manipulator

In this experiment, only the second link was used. The trajectory for this link was given below:

$$\dot{\theta}_d = \begin{cases} 180° \times kT \times 2, & \text{if } kT < 0.5 \\ 180°, & \text{if } 0.5 \le kT < 1.0 \\ 180° - 180° \times (kT - 1) \times 2, & \text{if } 1.0 \le kT < 1.5 \\ 0°, & \text{if } 1.5 \le kT \end{cases} \tag{18}$$

$$\theta_d = \theta_{od} + \dot{\theta}_d T \tag{19}$$

where T is the sampling period, which was $1ms$; k is sampling instance; $\dot{\theta}_d$ is the desired angular velocity; θ_{od} is the goal value of previous time step. In the experiments, we verified only the position tracking. Force control was not implemented. The ANN had one hidden layer with 10 processing node. The critic was

$$r = \frac{\mid \theta_d - \theta \mid}{1.5} \tag{20}$$

The PID controller was designed as follows:

$$\frac{U(S)}{E(S)} = K_p + K_i\frac{1}{S} + K_d S \tag{21}$$

$$\begin{aligned} U(k) &= U(k-1) + K_p(e(k) - e(k-1)) + K_i^{'}e(k) \\ &\quad + K_d^{'}(e(k) - 2e(k-1) + e(k-2)) \end{aligned} \tag{22}$$

where $K_i^{'} = K_i P$, $K_d^{'} = \frac{K_d}{T}$, k is the sampling instant. A set of controller coefficients, $K_p = 3.0$, $K_i^{'} = 0.005$, $K_d^{'} = 1.5$ were set to give satisfactory performance. Then we added a 3kg load as disturbance to the system, the system response oscillated. Fig. 10 shows the experimental result after 100 training iterations. It can be seen from the figure that the learning controller not only eliminated the disturbance but also demonstrated faster rise time. Fig. 11 shows the comparison of the position error before and after learning.

5.2 Experimental results of two-link manipulator

In this experiment, the two-link manipulator was commanded to follow a desired trajectory given below:

$$\theta_d = -90^\circ \times cos(180^\circ kT) \quad (23)$$

In this case, the coefficient $K_i = 0$, therefore we have

$$\frac{U(S)}{E(S)} = K_p + K_d S. \quad (24)$$

$$U(k) = K_p e(k) + K_d(e(k) - e(k-1)) \quad (25)$$

We set $K_p = \binom{3}{1}$ and $K_d' = \binom{1.5}{0.5}$ to obtain satisfactory responses. In the experiments, during $0^\circ < \theta_1 < 15^\circ$, we added a $20N.m$ noise torque to motor 1, and during $0^\circ < \theta_2 < 15^\circ$, we added a $5N.m$ noise torque to motor 2. The Fig. 12 shows the experimental results of tracking performance after 180 training iterations.

6 Conclusions

In this paper, we proposed a reinforcement learning control strategy, which was tested for robot manipulators. In the simulations, the controller learned the robot dynamic model and the environment states by reinforcement learning method. The uncertainty problem is solved , because the controller has the online learning capability. However, the learning speed is slow for complex nonlinear dynamics of robot manipulation in compliant motion. To be more practical, the reinforcement learning scheme is combined with a conventional PID controller to obtain faster convergence speed of learning, while improving the system performance. In the experiments, we have demonstrated the capacity of reinforcement learning to eliminate load disturbance applied to robot manipulators. In the future, more efficient learning structure needs to be further investigated to increase the convergence of the learning phase.

Acknowledgment

This work was supported by National Science Council under the grant NSC85-2213-E-009-094.

References

[1] J. S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)", *Tran. of ASME, Series G*, Vol. 97, No. 3, 1975, pp.220-227

[2] A. G. Barto, "Connectionist Learning for Control", *Neural Networks for Control*, W. T. Miller, R. Sutton, and P. Werbos, Eds, MIT Press, Cambridge, MA, 1990

[3] A. G. Barto and P. Anandan, "Pattern-Recognizing Stochastic Learning Automata", *IEEE Trans. Sys. Man. Cyber.*, Vol.SMC-15, No. 3, 1985, pp.360-375

[4] J. Franklin, "Input Space Representation for Reinforcement Learning Control", *Proc. of the 28th Conf. on Decision and Control*, 1989, pp.115-122

[5] V. Gullapulli, "A Stochastic Reinforcement Learning Algorithm for Learning Real-Valued Functions", *Neural Networks*, Vol. 3, 1990, pp.671-692

[6] N. Hogan, "Impedance Control: An Approach to Manipulation: part I,II,III", *TRANS. ASME J. Dyn. Sys. Meas. and Control*, Vol. 107, March 1985, pp.1-24

[7] D. Michie and R. A. Chambers, "BOXES: An Experiment in Adaptive Control", *Machine Intelligence 2*, E. Dale and D. Michie Eds., 1986, pp.137-152

[8] K. S. Narendra and M. A. L. Thathachar, "Learning Automata-A Survey", *IEEE Trans. on Sys. Man and Cyber.*, 14, 1974, pp.323-334

[9] M. H. Raibert and J. Craig, "Hybrid Position/Force Control of Manipulators", *Trans. ASME J. Dyn. Sys. Meas. and Control*, Vol. 102, June 1981, pp.126-133

[10] D. E. Rumelhart, G. E. Hinton and R. J. Williams, *Parallel Distributed Processing*, Cambridge, MA: The MIT Press, 1986

[11] T. S. Chu and K. T. Song, "An Experimental Study of Force Tracking Control by Reinforcement Learning", *1994 Inter. Symposium on Artificial Neural Networks*, Tainan, Taiwan, 1994, pp.728-734

[12] W. Y. Sun, "Control Design and Experimental Study of a Robot Using Reinforcement Learning", *Master Thesis*, National Chiao Tung Univ., 1995

[13] P. J. Werbos, "Generalization of Back Propagation with Application to a Recurrent Gas Marke Model", *Neural Networks*, I(October), 1988

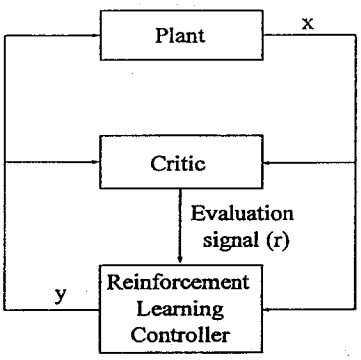

Fig. 1 Control structure of reinforcement learning

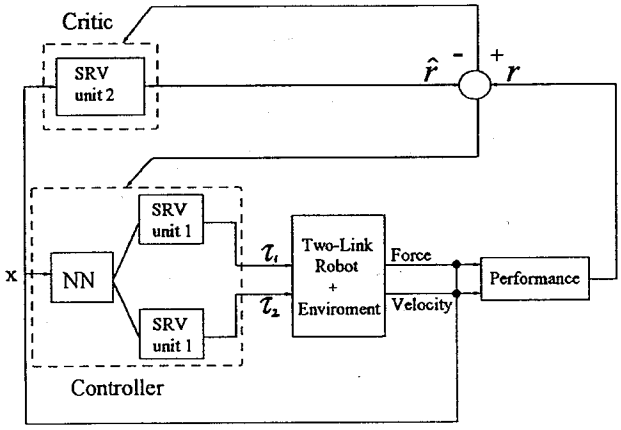

Fig. 2 Robot force tracking control design

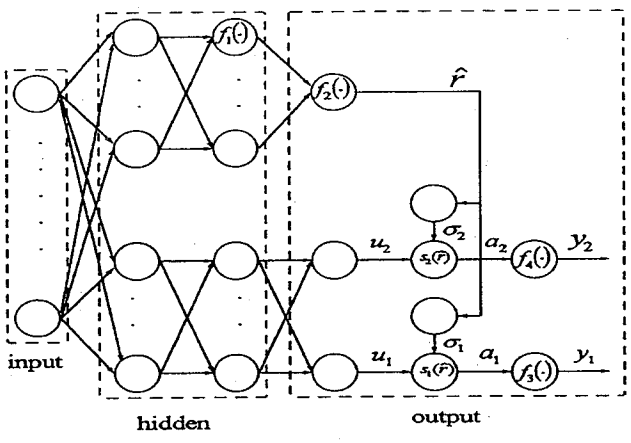

Fig. 3 ANN structure for reinforcement learning

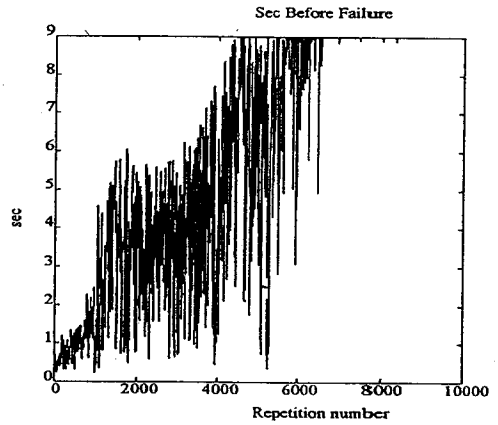

Fig. 4 Circular path: execution time before failure

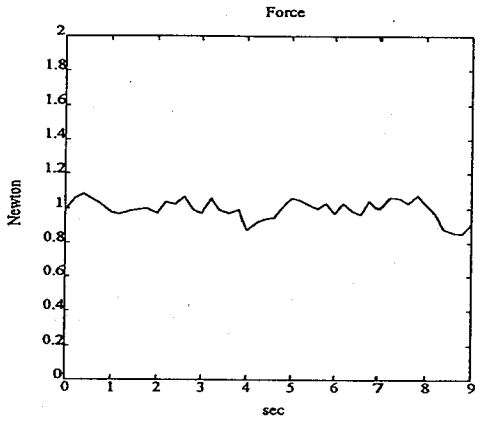

Fig. 5 Circular path: Simulation result of force

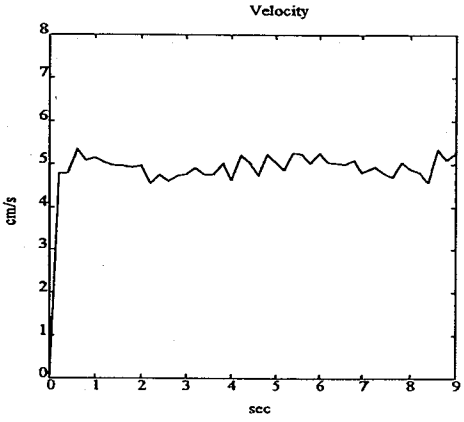

Fig. 6 Circular path: Simulation result of velocity

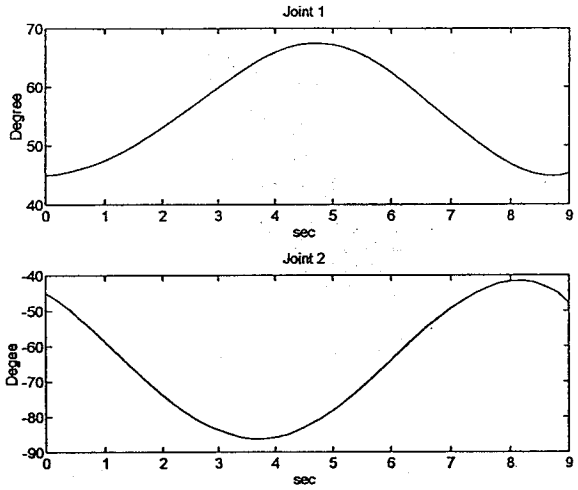

Fig. 7 Circular path: Simulation result of trajectories

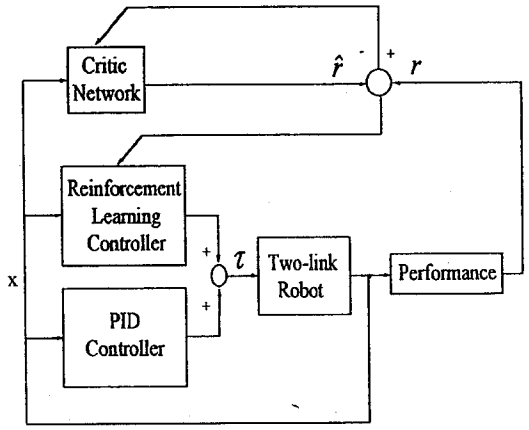

Fig. 8 Block diagram for control optimization

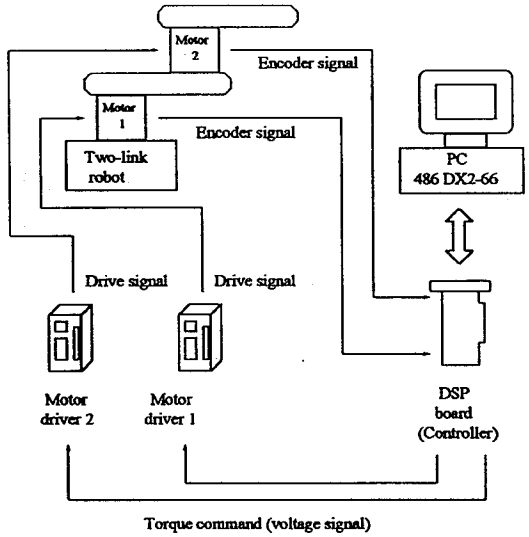

Fig. 9 Experimental setup

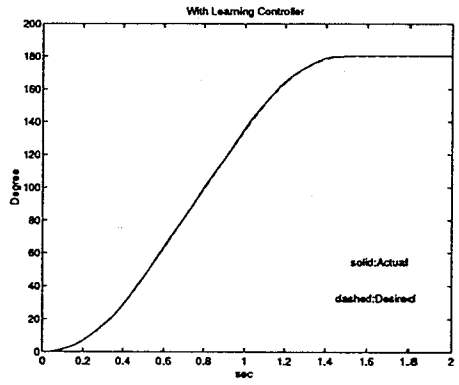

Fig. 10 One-link: experimental result after learning

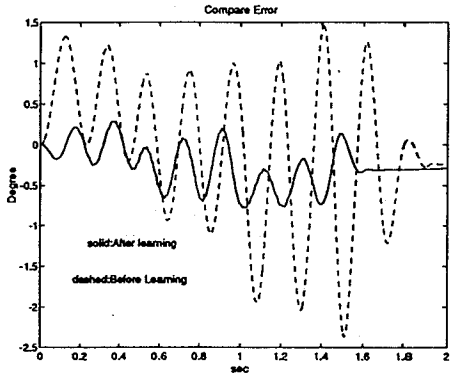

Fig. 11 One-link: comparison of errors after learning

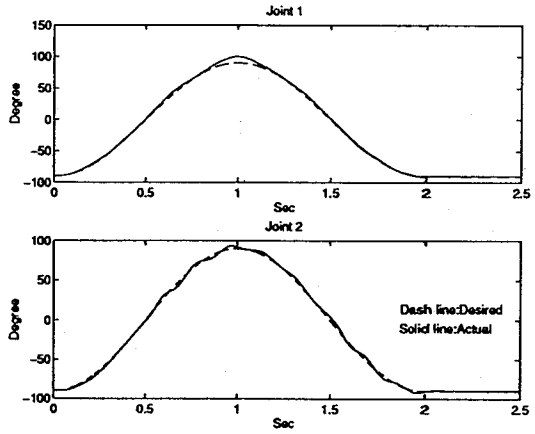

Fig. 12 Two-link: experimental result after learning