Journal of Intelligent and Robotic Systems **21:** 221–238, 1998.

Robot Control Optimization Using Reinforcement Learning

KAI-TAI SONG* and WEN-YU SUN
Department of Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, R.O.C.; e-mail: ktsong@cc.nctu.edu.tw

(Received: 3 December 1996; accepted in revised form: 5 September 1997)

Abstract. Conventional robot control schemes are basically model-based methods. However, exact modeling of robot dynamics poses considerable problems and faces various uncertainties in task execution. This paper proposes a reinforcement learning control approach for overcoming such drawbacks. An artificial neural network (ANN) serves as the learning structure, and an applied stochastic real-valued (SRV) unit as the learning method. Initially, force tracking control of a two-link robot arm is simulated to verify the control design. The simulation results confirm that even without information related to the robot dynamic model and environment states, operation rules for simultaneous controlling force and velocity are achievable by repetitive exploration. Hitherto, however, an acceptable performance has demanded many learning iterations and the learning speed proved too slow for practical applications. The approach herein, therefore, improves the tracking performance by combining a conventional controller with a reinforcement learning strategy. Experimental results demonstrate improved trajectory tracking performance of a two-link direct-drive robot manipulator using the proposed method.

Key words: artificial neural network, dynamic control, reinforcement learning, robot control.

1. Introduction

Industrial robots are extensively employed in factories for welding, painting and for conveying materials. At present, attempts to apply them to more complex tasks such as assembling, grinding and deburring are underway. It is desirable for a robot manipulator to acquire skills and operate in an unstructured and uncertain environment autonomously. In such applications, robots must react more closely with the environment and the inclusion of learning and adaptation in robot motion control is therefore necessary. Conventional robot dynamic control schemes are basically model-based approaches [1, 11]. They assume the dynamic model of the manipulator and the environment states (shape and stiffness) are either known for the designer or can be estimated on-line, although during execution it is not always practically possible. Such model-based control schemes share two common problems: 1) the uncertainties in the dynamic modeling of the manipulator and 2) the uncertainties in the environment states.

* This work was supported by the National Science Council under grant NSC-85-2213-E-009-094.

Recently, learning control of robot manipulators has been investigated by many researchers [4, 9]. Learning control is a control method that can achieve a desired level of control performance when a priori information concerning the system is unknown or incompletely known. In this approach, performance is improved by iterative practice. Miller et al. [9] presented a neural network based learning control design for dynamic control of an industrial robot. Gullapalli et al. [4] applied reinforcement learning control for acquiring robot skills and demonstrated its capacity to learn nonlinear control functions. Song and Chu [11] employed reinforcement learning for force tracking with an industrial robot. Even with no information about the environment states available for control design, satisfactory force tracking results were obtained. Their experimental results demonstrated the controller's learning capacity under environment variations. But in that work, the learning controller was designed to deal with the uncertainties in the contace environment alone. To extend a solution to the problem, this study proposed a reinforcement learning control design that not only deals with environment states but also the dynamics of a two-link robot manipulator. Those uncertainties mentioned above will not occur in such circumstance. Moreover, since the applied stochastic reinforcement learning algorithm can estimate the optimal action for rendering the desired response [7], the performance of the robot control system will be optimized for a repetitive trajectory. However, because the learning speed proved to be very slow, applying the reinforcement learning scheme to real-world problems was a problem. For practical applications, an alternative strategy is proposed, which involves combining a conventional PID controller with the reinforcement scheme to deal with uncertainties and improve control performance under system-model variation. The rest of this paper is organized as follows. Section 2 introduces the reinforcement learning control structure. Section 3 describes a controller design for a two-link direct-drive manipulator. Section 4 presents the simulation results. Practical experimental results, given in Section 5, demonstrate the feasibility of reinforcement learning in real-world applications. Section 6 presents conclusion.

2. Reinforcement Learning

Figure 1 depicts the basic concepts of reinforcement learning. 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 $r(t) \in \Re$. That signal is termed 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 a unique function $y^*(x)$ exists which optimizes the reinforcement over the input space. That function $y^*(x)$ is termed the optimal law. Hence the objective of reinforcement learning is to learn an optimal

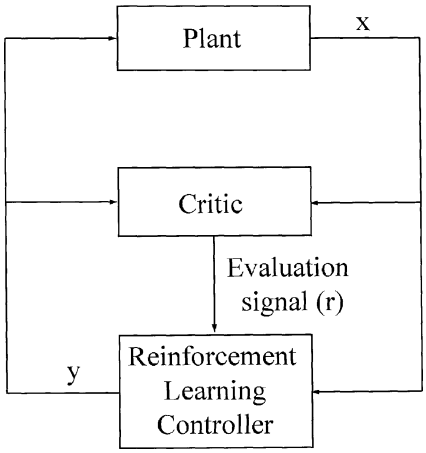

Figure 1. Reinforcement learning control structure.

controller such that for each input $x(t)$ an optimal performance evaluation r is returned. Therefore, there are two main issues pertain to reinforcement learning: the first to construct a critic that fits control purposes and contains adequate information for performance evaluation, and the second to determine a method of updating the controller to improve performance. For tasks such as playing chess or devising an inverted pendulum, where the final results are influenced by a multiple sequence of previous control actions, the first issue becomes how to translate an overall performance evaluation of a series of control actions into an immediate performance evaluation of particular control actions [3]. Sutton [15] proposed an adaptive heuristic critic and temporal difference to solve this problem. In this study, however, because the controller is immediately provided with a performance evaluation for each selected control action from the environment and these evaluation signals are directly employed to improve the control performance, the focus will be on the second problem.

To improve performance, it is necessary to determine the gradient of the reinforcement function in terms of control actions for adjusting the controller's parameters. Since the reinforcement function is unknown to the controller, reinforcement learning algorithms are acquired for estimating the gradient of the reinforcement value. Werbos [16] employed reinforcement learning algorithms in two main approaches to find the gradient: 1) the indirect approach, where first the environment is identified before utilizing it to train the controller, and 2) the direct approach, which is the one selected for this study.

In the direct approach, the controller actively explores the control action space to acquire gradient information about the reinforcement function by comparing the reinforcement values in that space. This process is frequently termed *stochastic reinforcement learning*. Earlier direct approaches, such as learning automata [10] and the associative reward-penalty algorithm [4], provided the controller

with only discrete action space and were not satisfactory for many practical control applications requiring continuous control signals. Gullapalli [7] proposed a stochastic reinforcement learning algorithm for a learning function with continuous outputs. That algorithm is based on the stochastic real-valued (SRV) unit. It is in two parts, unit 1 which is the learning element, producing an output that is a real-value function of the inputs, and unit 2 which is the reinforcement predictor, estimating the expected value of the reinforcement signal. Both parts learn simultaneously in real-time. The algorithm is stochastic, requiring random actions that might lead to improved performance to be tried. To enable such exploration, the output of unit 1 μ is treated as a mean, and that of unit 2 σ is the standard deviation, which can be interpreted as extent search for a better action. These two parameters are adjusted in the learning process to increase the probability of producing the optimal control action. The algorithm does this by maintaining the mean of the output an estimate of the optimal control action, such that it has the minimum reinforcement from the environment. The random search is accomplished by taking the normal distribution $N(\mu, \sigma)$ for each input, which yields a random variable a which is then mapped by an output function f to the control output y.

3. Design for Robot Dynamic Control

Figure 2 illustrates a dynamic control design based-on stochastic reinforcement learning of a two-link robot manipulator. Since the original SRV structure is for a single output learning element, in this study two of them are placed parallel to learn, respectively joint 1 and joint 2 torque. Because only single performance evaluation is required, only one reinforcement predictor (unit 2) is employed. The learning algorithms for both units are described below.

3.1. THE REINFORCEMENT PREDICTOR

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

$$\hat{r}(k) = N_e[V, x(k)], \tag{1}$$

where N_e represents the critic network, $[x(k), \hat{r}]$ is the input-output pair at time instant k, and V the weight matrix of the ANN. The least mean square (LMS) method [17] usually serves as the learning rule for adjusting the weight parameters. The cost function is:

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

The updating rule is based on the gradient descent method:

$$v(k+1) = v(k) + \Delta v(k) \tag{3}$$

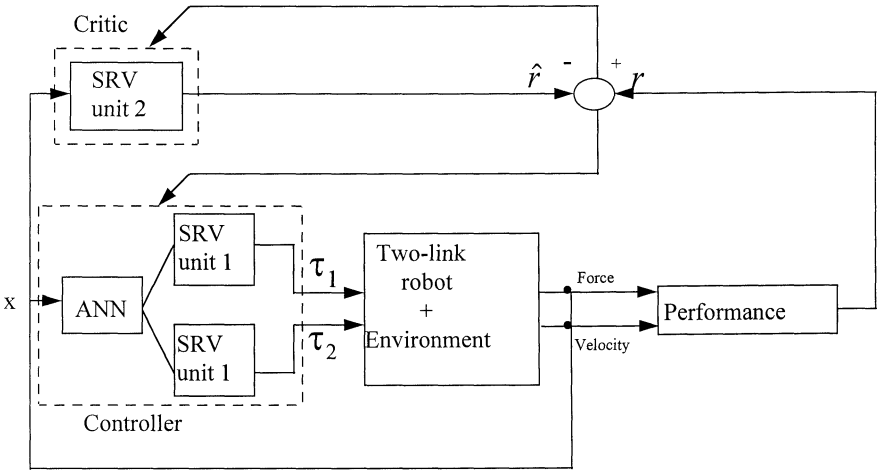

Figure 2. Robot dynamic control design using stochastic reinforcement learning.

and

$$\Delta v(k) = \beta\big(r(k+1) - \hat{r}(k)\big)\frac{\partial \hat{r}(k)}{\partial v(k)} + \lambda \Delta v(k-1), \tag{4}$$

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.

3.2. THE LEARNING ELEMENT

Unit 1 is the learning element, whose function is controller in this study. Its output, in general, can be expressed as:

$$y(k) = N_c\big[W, x(k)\big], \tag{5}$$

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

$$\omega(k+1) = \omega(k) + \Delta\omega(k) \tag{6}$$

and

$$\Delta\omega(k) = \alpha\frac{\partial r(k+1)}{\partial \omega(k)} = \alpha\frac{\partial r(k+1)}{\partial \mu(k)}\,\frac{\partial \mu(k)}{\partial \omega(k)}, \tag{7}$$

where α is the learning rate, and ω an element of W. Because the system model is unobtainable, it is not possible to calculate the derivative of critic r relative to the mean μ, $(\partial r(k+1)/\partial \mu(k))$. Gullapalli [7] proposed a random search approach for this problem. Suppose the system critic is a value between 0 and 1,

and it is assumed that a smaller value represents the better performance, this then yields:

$$\frac{\partial r(k+1)}{\partial \mu(k)} = (\hat{r}(k) - r(k+1))\frac{a(k) - \mu(k)}{\sigma(k)}. \tag{8}$$

This implies that when the actual critic is smaller than the predicted critic, or when $\hat{r}(k) - r(k+1) > 0$, the direction of searching is correct, and therefore 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 $a(k)$. As pointed out in [7], the fraction in the above equation may be regarded as normalized noise. If the noise causes the unit to receive a reinforcement signal that is less than the expected reinforcement, then it will be desirable for the unit to have an action closer to the current action. It should therefore update its mean action in the direction of the noise. On the other hand, if the noise causes the unit to receive a reinforcement signal that is more than the expected reinforcement, then it should update its mean action in the opposite direction. Standard deviation σ can be treated as search amount; when $\hat{r}$ is smaller, then the system performance improves, and the control action becomes closer to the ideal value, such that the search amount should decrease. This implies the standard deviation σ becomes smaller. This signifies the function s linking $\hat{r}$ and σ should be a monotonically decreasing function of $\hat{r}$.

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. However, the SRV unit solves only reinforcement learning problem, it is non-associative. Many structures for associative memory are available, such as a boxes system [8], a cerebellar model articulation controller (CMAC) [2], artificial neural networks (ANNs) and fuzzy-neural networks. In these approaches, the boxes system and a CMAC require proper partition of the input space, and a fuzzy-neural network structure requires setting up of the fuzzy rules and membership functions [14]. ANNs have been utilized for associative memory in many applications. If the hidden layer elements are adequate, any nonlinear mapping can be accomplished by a three layered feedforward network. Its generalization property makes ANN a popular choice, and it was thus applied as the learning structure herein and trained by an error-backpropagation algorithm [12]. Figure 3 illustrates the ANN structure in the proposed reinforcement learning controller. One neural network is designed to work as the reinforcement predictor, which gives the predicted critic. Another neural network is the controller, which produces the control torques to the robot manipulator. The ANNs possesss two hidden layers, each of which has twelve processing elements. There are six inputs for the ANN: force error, velocity error, angular position of joint 1 and 2 (θ_1 and θ_2), angular velocity of joint 1 and 2 ($\dot{\theta}_1$ and $\dot{\theta}_2$). The control

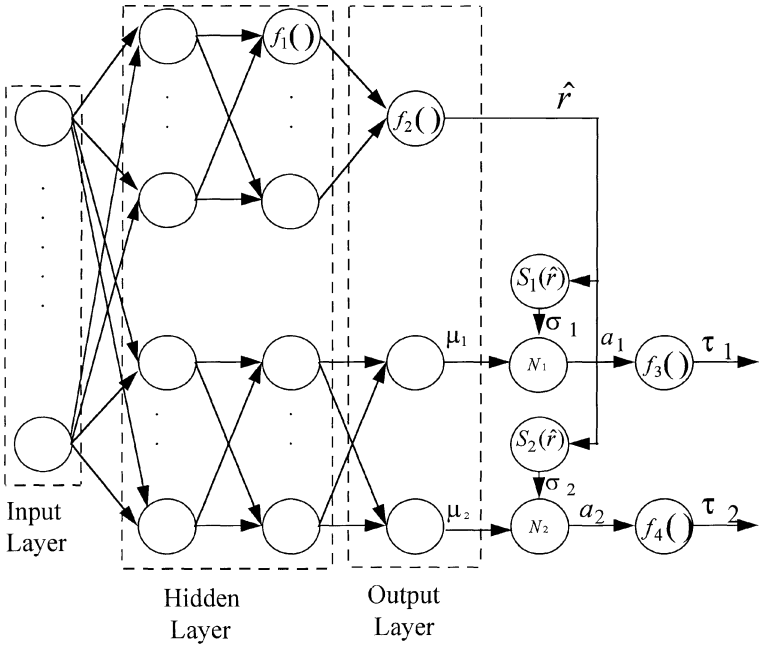

Figure 3. ANN structure for reinforcement learning.

actions are the torques for the two joint motors. The activation functions of neuron f_1 and neuron f_2 are

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

The activation functions f_3 and f_4 are

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

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

The functions s_1 and s_2 are linear function

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

4. Simulation Results

To verify the proposed design, computer simulations for force tracking control of a two-link robot arm were carried out. The SCARA type robot manipulator possesses two parallel links, the length and mass of each of which are: $l_1 = 0.4$ m, $l_2 = 0.3$ m, $m_1 = 15$ kg, $m_2 = 3$ kg. The environment was modeled as a stiffness,

which is to say that the normal force is proportional to the difference between the environment surface position and the set-point position of the end-effector. Under the condition of no prior information about neither the robot model nor the environment, the controller was set to learn the relationship between the dynamics of the robot arm and the stiffness of the environment. The dynamic model employed in these simulations is described below. A robot manipulator with two parallel-link (without gravity terms), yields:

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

where $M(\Theta)$ is the inertial term, and $V(\Theta, \dot{\Theta})$ 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 a 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: $F_d = 1.0$ N, $V_d =$ 5 cm/s, respectively. In simulation of robot compliant motion, the performance can be characterized by the actual contact force and the moving velocity of the tool tip. Therefore, the reinforcement signal was designed by taking into account the tracking errors of specified desired contact force and moving velocity:

$$r = \frac{1}{2}\left(|F_d - F| + \left|\frac{V_d - V}{5}\right|\right). \tag{17}$$

If r exceeded 1, the failure signal would be recognized, preventing damage to the tool, and the system would reset to the initial setting ($\theta_1 = 45°$, $\theta_2 = -45°$). The stiffness of the contact environment was 100 N/m. The control design was tested in two environmental configurations: 1) a 50 cm line path and 2) in a circular path of 8 cm radius. Only the results for circular path, which is more difficult to track, are presented here. Figure 4 illustrates the simulation results. Figure 4(a) indicates the length of time the robot arm had moved before the failure occurred. If a failure did not occur during the execution of the specified task, then the robot was able to complete the circular path. Otherwise, it would return to the

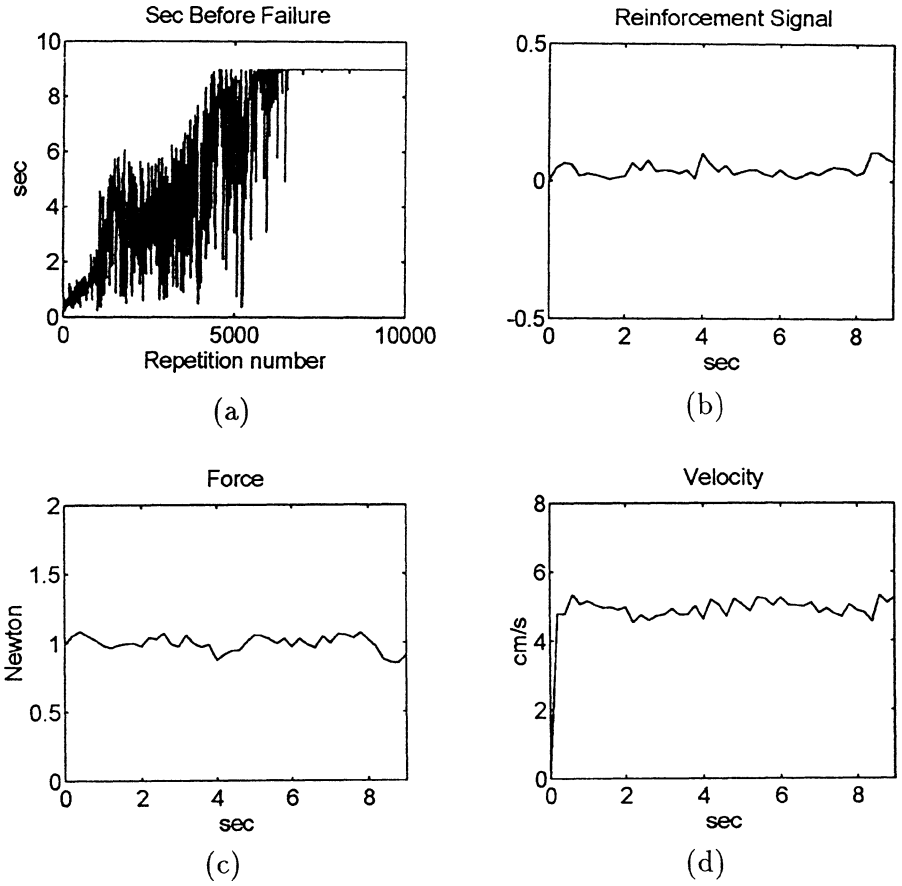

Figure 4. Simulation result of robot tracking control of a two-link robot manipulator: (a) execution time before failure, (b) trajectory of reinforcement signal, (c) force response, (d) velocity response.

initial position and start a new trial. Note that it took 9 seconds for the robot to complete the circular path. We see from the figure that after about 6000 trials, the robot was able to finish the task without failure. This means the dynamics of the two-link robot manipulator as well as the contact environment dynamics was learned by the proposed reinforcement learning controller. Figure 4(b) shows the reinforcement signal trajectory. It is practically zero during the execution of the final trial and this indicates the desired control optimization has been achieved. Figures 4(c) and (d) depict force and velocity responses, respectively. Figure 5 illustrates the trajectories of joint 1 and joint 2. Figure 6 shows the output torques of each joint. The above simulation results confirm that the neural network has learned the dynamic relationship between the robot manipulator and the environment. However, the initial instability and slow learning speed are great problems in practical applications.

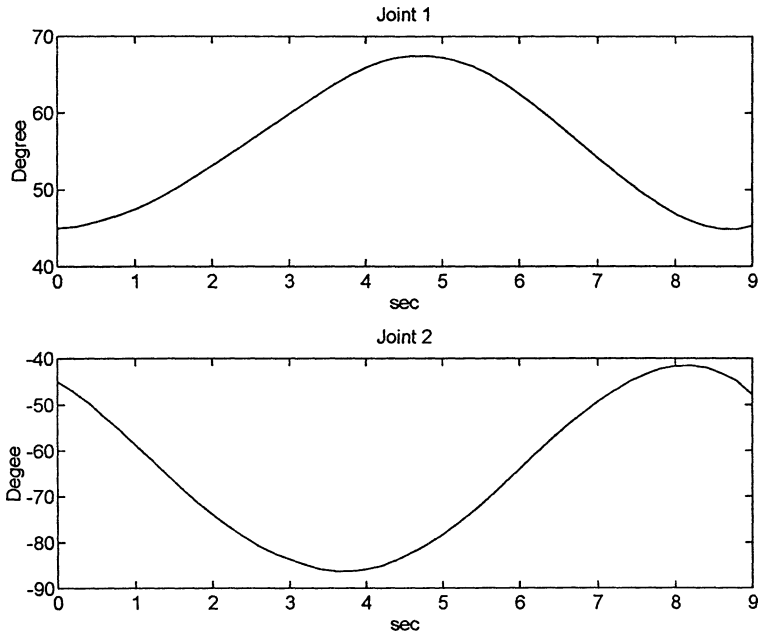

Figure 5. Simulation result of trajectories of each joint.

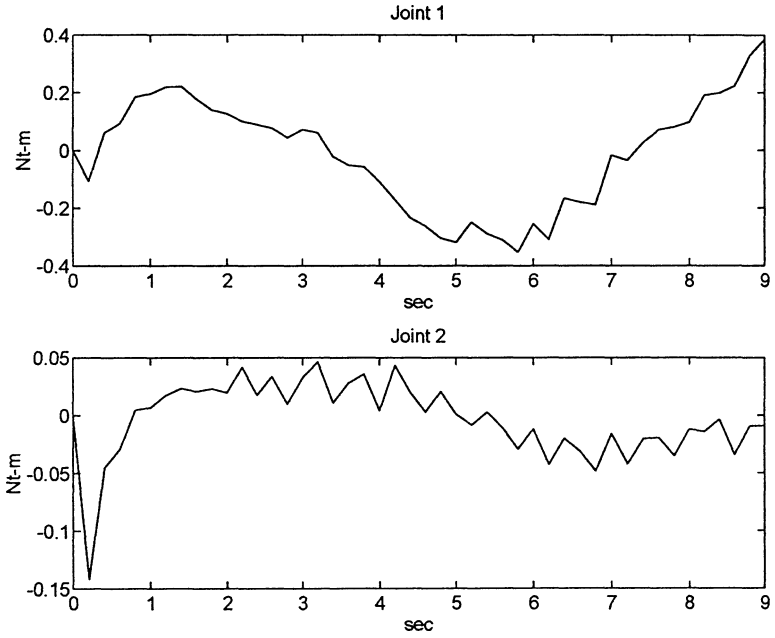

Figure 6. Simulation result of control torques of each joint.

5. Experimental Results

To demonstrate the feasibility of reinforcement learning control scheme in real-world applications, a new control scheme is proposed for applying reinforcement learning to dynamic control of a robot manipulator. Figure 7 depicts the proposed control system block diagram. Conventional PID controllers were implemented to control the two-link robot arm for tracking the desired trajectory with preliminary performance. Problems caused by nonlinearity and uncertainties in robot tracking control are well recognized features, as is that load-disturbance may deteriorate performance. The control scheme herein proposed improves the tracking performance by adding an on-line learning controller. The inputs of reinforcement learning controller were designed to include current position error $e(k)$, position error of the previous sampling instant $e(k-1)$, two-time-step previous position error $e(k-2)$, the previous control output $u(k-1)$, joint angular position $\theta(k)$ and joint angular velocity $\theta(k)$, where $e(k)$, $e(k-1)$, $e(k-2)$ and $u(k-1)$ were also the inputs of the PID controller. The tuned PID controller stabilizes the system when the system is initialized. In the following successive control cycles, the learning controller estimates the output of the PID controller and compensates with an adaptive torque. When the system dynamics are learned through repetitive practicing, the control will be taken over by the reinforcement learning controller. The training is based on SRV unit and the convergence theorem applies [6]. Figure 8 depicts the experimental setup. A laboratory-made two-link SCARA type manipulator was driven by two direct-drive motors. The maximum torque of direct-drive motors are $60Nt - m$ and $15Nt - m$, respectively. The control scheme was implemented on a dSPACE DSP controller card. A PC/AT-486 served as the host computer. Two experiments were conducted to justify the control performance can be improved using the proposed control scheme.

5.1. TRACKING CONTROL OF ONE-LINK MANIPULATOR

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

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

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

where T is the sampling period, which was 1 ms in the experiment; k represents the sample instant; $\dot{\theta}_d$ is the desired angular velocity; θ_{od} is the goal value of previous time step. Since for these experiments only position tracking was implemented, a simplified ANN structure was employed. The ANN possessed

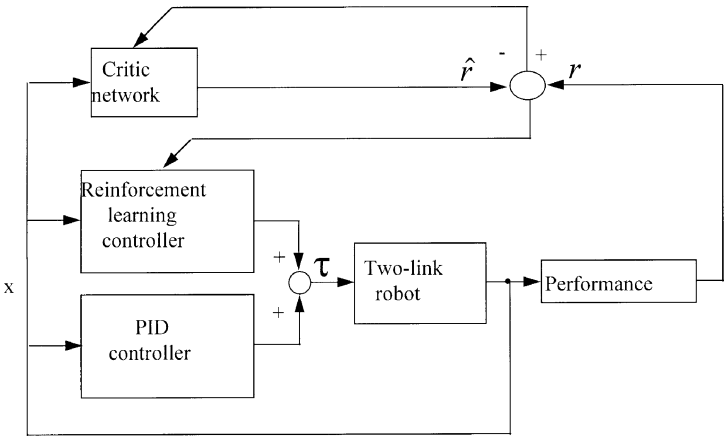

Figure 7. Block diagram of reinforcement learning control experiment.

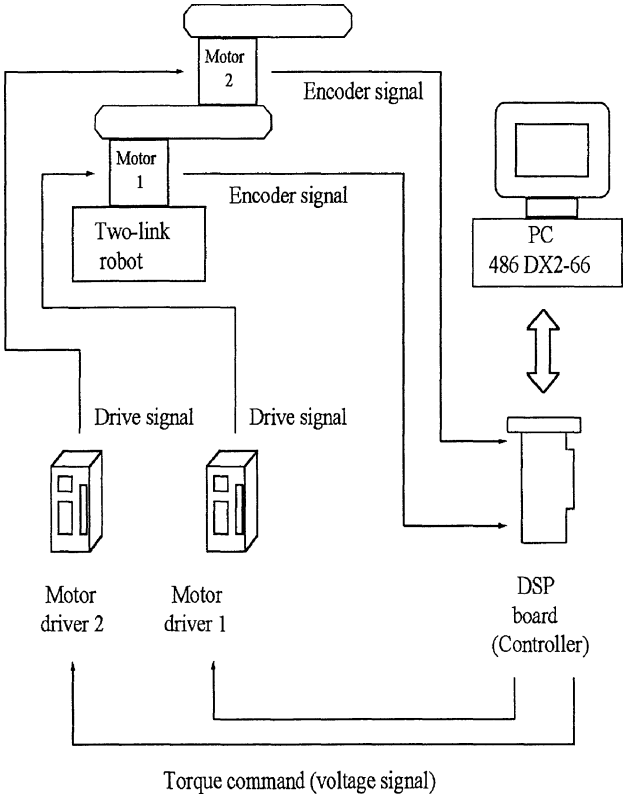

Figure 8. Experimental set-up.

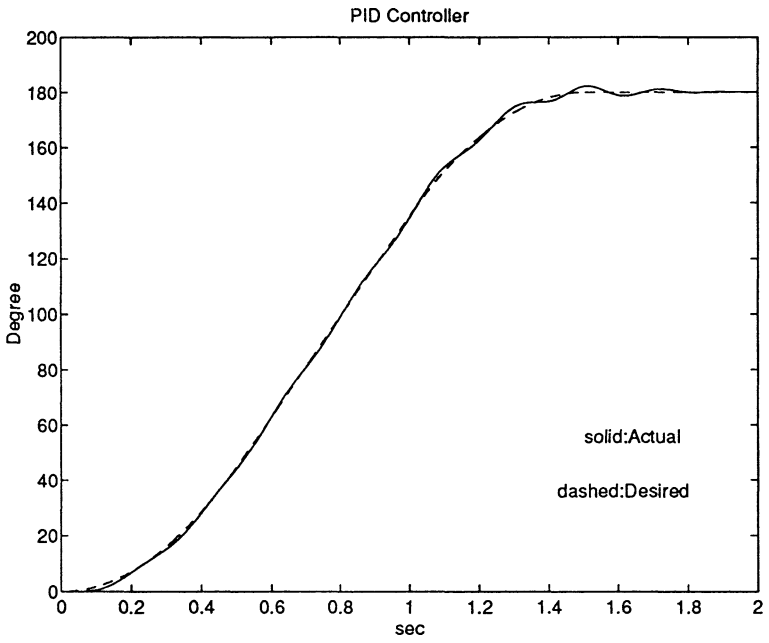

Figure 9. Experimental result of one-link manipulator using PID controller alone.

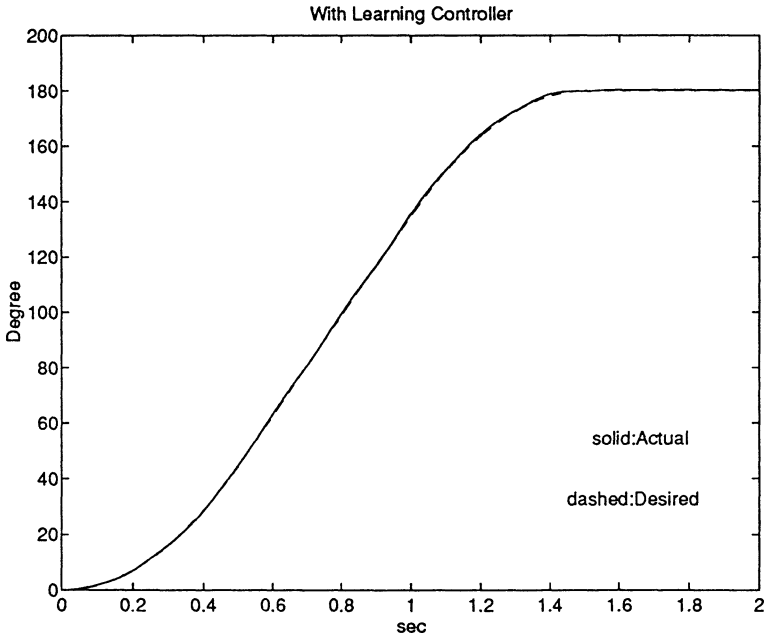

Figure 10. Experimental result after learning.

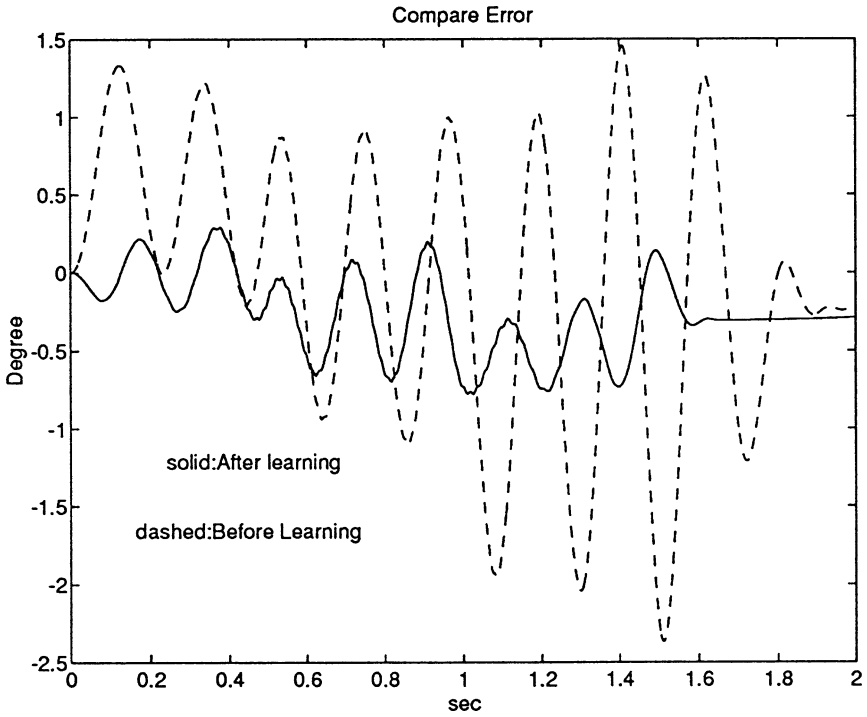

Figure 11. Comparison of error before (using PID controller alone) and after learning.

one hidden layer with ten processing elements. Since this was a trajectory tracking task, the critic was designed to be:

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

The PID controller was designed as follows:

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

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

where $K_i' = K_i T$, $K_d' = K_d/T$. A set of controller coefficients, $K_p = 3.0$, $K_i' = 0.005$, $K_d' = 1.5$ were set to give preliminary performance. After tuning the PID controller, a 3 kg load disturbance was added to the system. Figure 9 shows the experimental result using the PID controller alone. It can be seen that there was a lag in the beginning and the arm oscillated. Figure 10 illustrates the experimental results after 100 training iterations. It can be seen from the

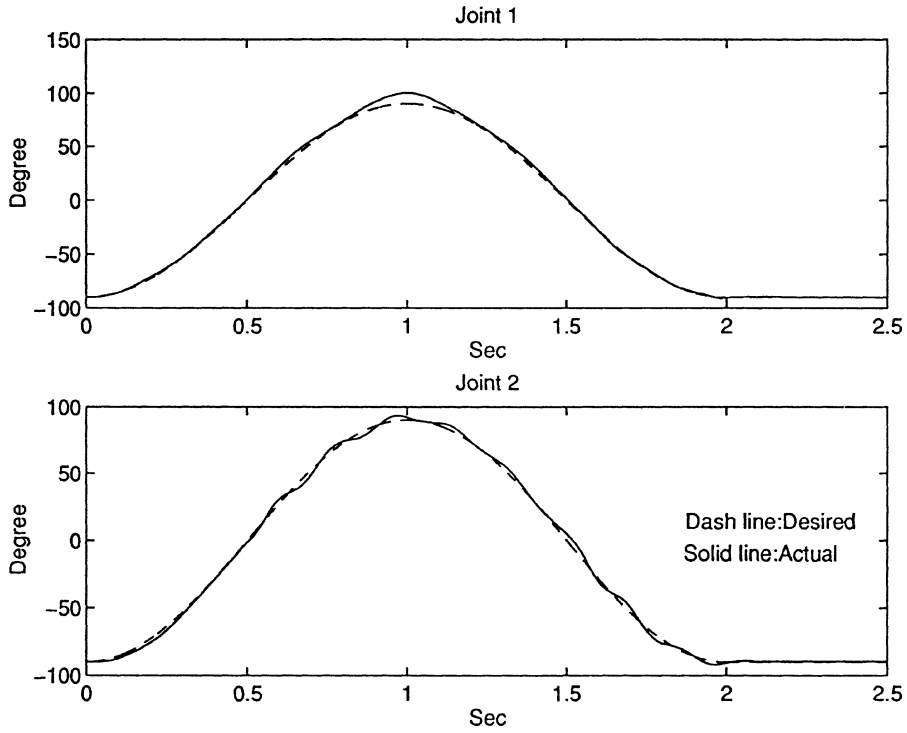

Figure 12. Experimental result of trajectory tracking after 180 learning iterations of the two-link robot manipulator.

figure that the learning controller not only eliminated the disturbance but also demonstrated enhanced rise time. Figure 11 illustrates the comparison of the position error before and after learning.

5.2. TRACKING CONTROL OF TWO-LINK MANIPULATOR

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

$$\theta_d = -90^\circ \times \cos(180^\circ kT). \tag{23}$$

In this case, the coefficient $K_i = 0$, therefore yielding

$$\frac{U(s)}{E(s)} = K_p + K_d s, \tag{24}$$

$$U(k) = K_p e(k) + K'_d\big(e(k) - e(k-1)\big). \tag{25}$$

The settings

$$K_p = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad \text{and} \quad K'_d = \begin{pmatrix} 1.5 \\ 0.5 \end{pmatrix}$$

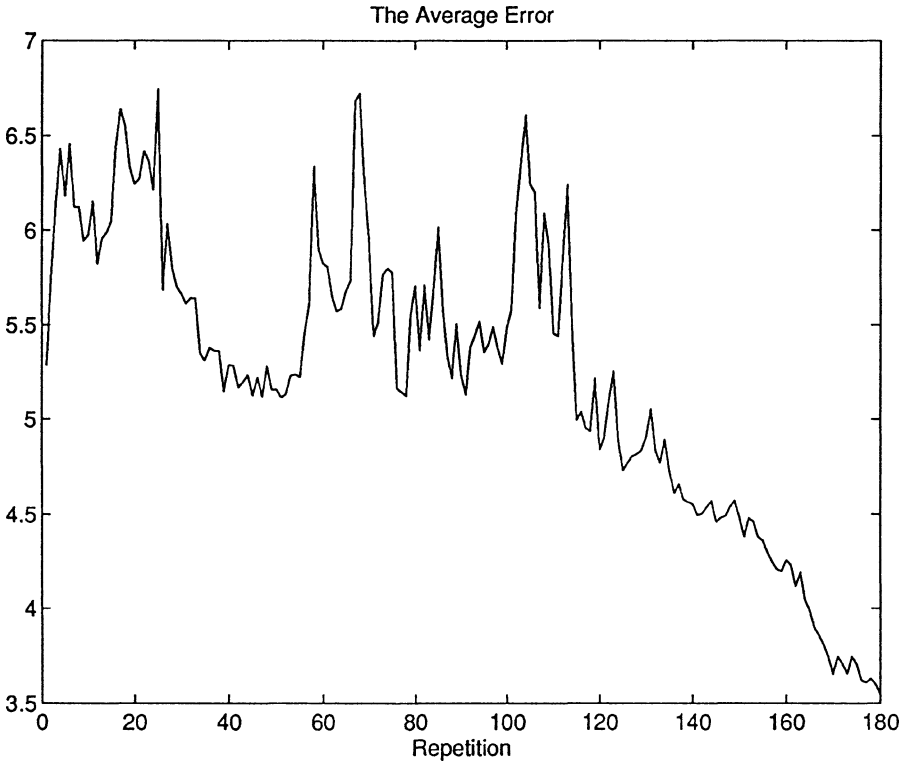

Figure 13. Convergence of average error in the experiment.

were intended to provide preliminary responses. In the experiments, during $0^\circ < \theta_1 < 15^\circ$, a $20Nt-m$ noise torque was added to motor 1, and during $0^\circ < \theta_2 < 15^\circ$, a $5Nt-m$ noise torque added to motor 2. Figure 12 shows the experimental results of tracking performance after 180 training iterations. Figure 13 gives the experimental result of average error defined below:

$$\frac{\sum_k |\theta_d(k) - \theta_1(k)| + |\theta_d(k) - \theta_2(k)|}{\sum k}, \tag{26}$$

where k is the sampling numbers. Notably, the average error decays after repetitive learning, decreasing from 5.4° before learning (using PID controllers alone) to 3.5° after learning. As the experiments demonstrate, the learning controller improves performance within 180 learning cycles.

6. Conclusion

In this paper a reinforcement learning control design has been presented for dynamic control of a two-link robot manipulator. The advantage of the applied stochastic reinforcement learning structure is that desired control actions can be

found for a specific task by actively exploring the control space, without requiring knowledge of system dynamics. The learning controller can achieve the desired performance via repetitive learning. Simulation results show that robot force and position tracking can be achieved simultaneously using the proposed reinforcement learning method. The controller learned the two-link robot dynamics and the contact-environment states. However, because the learning speed for complex nonlinear characteristics was slow, for more practical, real-world applications, the reinforcement learning scheme was combined with a conventional PID controller to enhance convergence speed and improve system performance. Practical experiments on a two-link direct-drive robot demonstrated the capacity of reinforcement learning to eliminate load disturbance applied to the robot manipulator. Moreover, the improved performance using the proposed learning control scheme was justified when compared with using a PID controller alone. The need for a more efficient learning structure to expedite the convergence of the learning phase remains a task for future investigation.

References

1. An, C. H., Atkeson, C. G., and Hollerbach, J. M.: *Model Based Control of a Robot Manipulator*, MIT Press, Cambridge, MA, 1988.
2. Albus, J. S.: A new approach to manipulator control: the cerebellar model articulation controller (CMAC), *Trans. of ASME, Series G* **97**(3) (1975), 220–227.
3. Barto, A. G.: Connectionist learning for control, in: W. T. Miller, R. Sutton, and P. Werbos (eds), *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990.
4. Barto, A. G. and Anandan, P.: Pattern-recognizing stochastic learning automata, *IEEE Trans. Systems Man Cybernet.* **15**(3) (1985), 360–375.
5. Gullapalli, V., Franklin, J. A., and Benbrahim, H.: Acquiring robot skills via reinforcement learning, *IEEE Control Systems* **14**(1) (1994), 13–24.
6. Gullapalli, V.: Associative reinforcement learning of real-valued functions, in: *Proc. of the IEEE Int. Conf. on Systems Man Cybernet.*, 1991, pp. 1453–1458.
7. Gullapalli, V.: A stochastic reinforcement learning algorithm for learning real-valued functions, *Neural Networks* **3** (1990), 671–692.
8. Michie, D. and Chambers, R. A.: BOXES: an experiment in adaptive control, in: E. Dale and D. Michie (eds), *Machine Intelligence 2*, 1986, pp. 137–152.
9. Miller, W. T., Hewes, R. P., Glanz, F. H., and Kraft, L. G.: Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller, *IEEE Trans. Robot. Automat.* **6**(1) (1990), 1–9.
10. Narendra, K. S. and Thathachar, M. A. L.: Learning automata – a survey, *IEEE Trans. Systems Man Cybernet.* **14** (1974), 323–334.
11. Raibert, M. H. and Craig, J.: Hybrid position/force control of manipulators, *Trans. ASME J. Dyn. Systems Meas. Control* **102** (1981), 126–133.
12. Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: *Parallel Distributed Processing*, MIT Press, Cambridge, MA, 1986.
13. Song, K. T. and Chu, T. S.: An experimental study of force tracking control by reinforcement learning, in: *Proc. 1994 Internat. Symp. on Artificial Neural Networks*, Taiwan, 1994, pp. 728–734.
14. Sun, W. Y.: Control design and experimental study of a robot using reinforcement learning, Master thesis, National Chiao Tung Univ., 1995.
15. Sutton, R. S.: Learning to predict by the method of temporal difference, *Machine Learning* **3**(1) (1988), 9–44.

16. Werbos, P. J.: Generalization of back propagation with application to a recurrent gas market model, *Neural Networks* **1** (October, 1988), 339–356.
17. Widrow, B. and Stearns, S. D.: *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.