WM14 - 1:15

Proceedings of the
American Control Conference
Seattle, Washington • June 1995

A New Neural Network Control Technique for Robot Manipulators *

Seul Jung and T.C. Hsia
Robotics Research Laboratory
Department of Electrical and Computer Engineering
University of California, Davis
Davis, CA 95616
e-mail:hsia@ece.ucdavis.edu and jung@ece.ucdavis.edu

Abstract

A new neural network (NN) control technique for robot manipulators is introduced in this paper. The fundamental robot control technique is the model-based computed-torque control which is subjected to performance degradation due to model uncertainty. NN controllers have been traditionally used to generate a compensating joint torque to account for the effects of the uncertainties. The proposed NN control approach is conceptually different in that it is aimed at prefiltering the desired joint trajectories before they are used to command the computed-torque-controlled robot system (the plant) to counteract performance degradation due to plant uncertainties. In this framework, the NN controller serves as the inverse model of the plant, which can be trained on-line using joint tracking error. Several variations of this basic technique is introduced. Back-propagation training algorithms for the NN controller have been developed. Simulation results have demonstrated the excellent tracking performance of the proposed control technique.

I Introduction

For the past several years, there have been a lot of interests in applying artificial neural networks (NNs) to solve the problems of identification and control of complex nonlinear system by exploiting the nonlinear mapping abilities of the NN [1, 2, 3]. At the same time, extensive investigations have been carried out to design NN controllers for robot manipulators [4, 5, 6]. The fundamental approach for robot manipulator control is the model-based computed-torque control scheme. Within this framework, the use of NN controller is basically to generate auxiliary joint torques to compensate for the uncertainties in the nonlinear robot dynamic model. Variations of this basic idea can be found in other designs[4, 6, 7]. NN can also be trained off-line to model the nonlinear robot dynamics, and then is used to implement the computed-torque controller with further on-line modifications to account for uncertainties [6].

*This work is supported in part by NITTA Corporation of Japan

In this paper we investigate a new robot control technique as depicted in Figure 1. Compensation of robot model uncertainty is achieved by modifying the desired trajectory, q_d, using an appropriately trained NN serving as a nonlinear filter. The filtered trajectories q_r $\dot{q}_r$ $\ddot{q}_r$ are the desired reference inputs to the computed-torque robot control system (referred to as the plant). The trajectory tracking error $\varepsilon = q_d - q$ is used as training signal. In this arrangement, the NN acts as the inverse of the plant. Hence we refer the proposed technique as neural network inverse control.

The control problem posed in Figure 1 is an example of the general problem of NN control of nonlinear plants. For this class of problems, plant dynamics must be known (or a model is identified) to implement the error-back-propagation algorithms for training the NN. We also note that the control system of Figure 1 can be viewed as an inverse system model identification problem [1]. Thus a necessary condition by the proposed scheme is that the plant be stable.

In most practical cases, we can assume that the plant based on the nominal robot model is stable when it is subjected to all uncertainties encountered in performing tasks. Under this condition, the robot tracking error $e = q_r - q$ satisfies approximately a second order linear differential equation. It is this property that allows us to carry out the required error-back-propagation computations. Combining this result with the assumed stability property for the plant makes the proposed control scheme in Figure 1 feasible. Several variations of this basic approach is examined as shown in Figures 2 and 3. In the following, we will develop the NN controller training algorithm as well as presenting simulation results for on-line trajectory tracking. It is shown that the control scheme is practical and the control system performance is excellent.

II Computed-Torque Control

The dynamic equation of an n degrees-of-freedom manipulator in joint space coordinates is given by :

$$D(q)\ddot{q} + H(q,\dot{q}) + F(\dot{q}) = \tau \tag{1}$$

where the vectors $q, \dot{q}, \ddot{q}$ are the joint angle, joint velocity, and joint acceleration, respectively; $D(q)$ is the n x n symmetric positive definite inertia matrix; $H(q,\dot{q}) = C(q,\dot{q})\dot{q} + G(q)$; $C(q,\dot{q})$ is the n x 1 vector of Coriolis and centrifugal torques; $G(q)$ is the n x 1 gravitational torques; τ is the n x 1 vector of joint actuator torques; $F(\dot{q})$ is an n x 1 vector representing Coulomb friction and viscous friction forces. Computed-torque method is the basic approach to robot control when nominal robot dynamic model is available. That is, the control law can be written as

$$\tau(t) = \hat{D}(q)u(t) + \hat{H}(q,\dot{q}) \tag{2}$$

where $\hat{D}(q)$ and $\hat{H}(q,\dot{q})$ are estimates of $D(q)$ and $H(q,\dot{q})$, and $u(t)$ is given by

$$u(t) = \ddot{q}_r + K_D(\dot{q}_r - \dot{q}) + K_P(q_r - q) \tag{3}$$

where K_P and K_D are n x n symmetric positive definite gain matrices. Combining (1),(2), and (3) yields the closed loop dynamic equation

$$\ddot{e} + K_D\dot{e} + K_P e = \hat{D}^{-1}(\Delta D(q)\ddot{q} + \Delta H(q,\dot{q}) + F(\dot{q})) \tag{4}$$

where $\Delta D(q) = D(q) - \hat{D}(q), \Delta H(q,\dot{q}) = H(q,\dot{q}) - \hat{H}(q,\dot{q})$, $e = (q_r - q)$ and $q_r = q_d + \phi_p$. If $\hat{D} = D, \hat{H} = H$, $F = 0$ and $\Phi = 0$, then $q_r = q_d$ and equation (4) becomes the following ideal linear second order equation of motion in error space as follows:

$$\ddot{\varepsilon} + K_D\dot{\varepsilon} + K_P\varepsilon = 0 \tag{5}$$

where $\varepsilon = q_d - q$. Since there are always uncertainties in the robot dynamic model, the ideal error response (5) can not be achieved and the performance is degraded as indicated by (4). Thus the computed-torque based control is not robust in practice. To improve robustness, NN controllers are proposed to compensate for the uncertainties as shown in Figures 1 and 2.

III Neural Network Inverse Control

A new robot control scheme is proposed in this paper as depicted in Figure 1. The basic concept of this scheme is that the NN controller acts as the inverse of the plant(the robot under computed-torque control) so that the robot response q tracks q_d with minimal distortion. So when the NN controller is updated online to account for the uncertainties in the robot dynamics, optimal trajectory tracking by the robot can be achieved. The tracking errors $(\varepsilon, \dot{\varepsilon}, \ddot{\varepsilon})$ are used to update the NN weights. From Figure 1 (called Scheme A) the control input vector $u(t)$ is

$$u(t) = \ddot{q}_r + K_D(\dot{q}_r - \dot{q}) + K_P(q_r - q) \tag{6}$$

where $\ddot{q}_r$ $\dot{q}_r$ q_r are reference input vectors to the plant and they are independently generated by the NN (so they are independent to one another, *i.e.*derivative relations among them do not necessarily hold). Combining (1) and (6) yields

$$(\ddot{q}_r - \ddot{q}) + K_D(\dot{q}_r - \dot{q}) + K_P(q_r - q) = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + f) \tag{7}$$

To analyze how the NN controller works, let us denote its three outputs as $\phi_p, \phi_d,$ and ϕ_a where $\ddot{q}_r = \phi_a$, $\dot{q}_r = \phi_d$, $q_r = \phi_p$ as shown in Figure 1. Using the definition of tracking error $\varepsilon = q_d - q$, (7) can be rewritten as

$$\ddot{\varepsilon} + K_D\dot{\varepsilon} + K_P\varepsilon = \delta + \ddot{q}_d + K_D\dot{q}_d + K_P q_d - \Psi_A \tag{8}$$

where $\delta = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + f)$ and $\Psi_A = (\phi_a + K_D\phi_d + K_P\phi_p)$ which represents the total contribution of the NN outputs to the tracking error equation (8). When the NN controller is converged, $\varepsilon = 0$ and the ideal NN outputs ϕ_p ϕ_d ϕ_a satisfy the relationship

$$\Psi_A = \delta + \ddot{q}_d + K_D\dot{q}_d + K_P q_d \tag{9}$$

It is seen that Ψ_A is equal to a combination of the desired trajectories and robot model uncertainties. This means that the function of the NN controller is to modify the desired trajectories such that its outputs ϕ_p ϕ_d ϕ_a satisfy the relation (9).

In view of the above analysis it is clear that the NN controller can be redesigned as shown in Figure 2 (called Scheme B). In this scheme, the NN outputs are added to the desired trajectories such that $\ddot{q}_r = \phi_a + \ddot{q}_d$, $\dot{q}_r = \phi_d + \dot{q}_d$, and $q_r = \phi_p + q_d$. Substituting q_r $\dot{q}_r$ $\ddot{q}_r$ into (7) yields

$$\ddot{\varepsilon} + K_D\dot{\varepsilon} + K_P\varepsilon = \delta - \Psi_B \tag{10}$$

where $\Psi_B = [\phi_a + K_D\phi_d + K_P\phi_p]$. Thus, for Scheme B, the ideal output of NN at convergence is

$$\Psi_B = \delta \tag{11}$$

In this case the NN is only required to cancel out the robot model uncertainty. Simulations presented later show that Scheme B is more efficient than Scheme A. Comparing the new results with those schemes which modify the robot joint torques , the NN in Schemes A and B have higher dimensional output than that of scheme by *Ishiguro et. al* [4]. This increased NN complexity should provide better controller performance. Our simulations show that this is true. We propose to use multilayer feedforward neural network with back-propagation updating algorithms.

IV Neural Network Controller Design

The standard two layer feedforward neural network is used as the controller. It is composed of an input buffer, a non-linear hidden layer and a linear output layer.

The weight updating law minimizes the objective function $E(\mathcal{E})$ which is a quadratic function of tracking errors ε:

$$E(\mathcal{E}) = \frac{1}{2}(\mathcal{E}^T\mathcal{E}) \tag{12}$$

where $\mathcal{E} = [\varepsilon^T\dot{\varepsilon}^T\ddot{\varepsilon}^T]^T, \varepsilon = (q_d - q), \dot{\varepsilon} = (\dot{q}_d - \dot{q})$, and $\ddot{\varepsilon} = \ddot{q}_d - \ddot{q}$. The gradient of $E(\mathcal{E})$ is

$$\frac{\partial E(\mathcal{E})}{\partial w} = \frac{\partial \mathcal{E}^T}{\partial w}\mathcal{E} = -[\ \frac{\partial q^T}{\partial w}\ \frac{\partial \dot{q}^T}{\partial w}\ \frac{\partial \ddot{q}^T}{\partial w}]\ \mathcal{E} \tag{13}$$

In both Scheme A and Scheme B, $q_r\ \dot{q}_r\ \ddot{q}_r$ are functions of w so

$$\begin{aligned}\frac{\partial \mathcal{E}^T}{\partial w} &= -[\ \frac{\partial q^T}{\partial w}\ \frac{\partial \dot{q}^T}{\partial w}\ \frac{\partial \ddot{q}^T}{\partial w}] \\ &= -[\ \frac{\partial q^T}{\partial q_r^T}\frac{\partial q_r^T}{\partial w}\ \frac{\partial \dot{q}^T}{\partial \dot{q}_r^T}\frac{\partial \dot{q}_r^T}{\partial w}\ \frac{\partial \ddot{q}^T}{\partial \ddot{q}_r^T}\ \frac{\partial \ddot{q}_r^T}{\partial w}]\end{aligned} \tag{14}$$

Evaluation of the derivatives $\frac{\partial q^T}{\partial q_r^T}\ \frac{\partial \dot{q}^T}{\partial \dot{q}_r^T}\ \frac{\partial \ddot{q}^T}{\partial \ddot{q}_r^T}$ require the knowledge of the plant model. Clearly the plant dynamics is not precisely known in our case. Our proposal here is to use the ideal linear error equation(q_d is replaced by q_r in (5))

$$(\ddot{q}_r - \ddot{q}) + K_D(\dot{q}_r - \dot{q}) + K_P(q_r - q) = 0 \tag{15}$$

as an approximation to the plant dynamics by neglecting the plant uncertainties. The most significant consequence of this approximation is that we have a linear plant model which provides simple results to (15). In view of (14), we have

$$\frac{\partial E(\mathcal{E})}{\partial w} = -[\ \frac{\partial q_r^T}{\partial w}\ \frac{\partial \dot{q}_r^T}{\partial w}\ \frac{\partial \ddot{q}_r^T}{\partial w}]\ J^T \begin{bmatrix} \varepsilon \\ \dot{\varepsilon} \\ \ddot{\varepsilon} \end{bmatrix} \tag{16}$$

where

$$J = \begin{bmatrix} \frac{\partial q}{\partial q_r} & \frac{\partial q}{\partial \dot{q}_r} & \frac{\partial q}{\partial \ddot{q}_r} \\ \frac{\partial \dot{q}}{\partial q_r} & \frac{\partial \dot{q}}{\partial \dot{q}_r} & \frac{\partial \dot{q}}{\partial \ddot{q}_r} \\ \frac{\partial \ddot{q}}{\partial q_r} & \frac{\partial \ddot{q}}{\partial \dot{q}_r} & \frac{\partial \ddot{q}}{\partial \ddot{q}_r} \end{bmatrix} = \begin{bmatrix} I & K_P^{-1}K_D & K_P^{-1} \\ K_D^{-1}K_P & I & K_D^{-1} \\ K_P & K_D & I \end{bmatrix} \tag{17}$$

J is a Jacobian for the plant which depends only on the PD gains. The gradient rule for weight update is

$$\Delta w(t) = -\eta\ \frac{\partial E}{\partial w} + \alpha\ \Delta w(t-1) \tag{18}$$

where η is the update rate and α is the momentum coefficient.

V Simplification for Controller Design

The control Schemes A an B involve simultaneous generation of the robot reference trajectories $q_r, \dot{q}_r$, and $\ddot{q}_r$ using three separate tracking errors, $\varepsilon, \dot{\varepsilon}$, and $\ddot{\varepsilon}$. Thus the total number of weights is $(n_I+1)n_H+(n_H+1)n_O$. But in both Ψ_A and Ψ_B, the NN outputs ϕ_p and ϕ_d are weighted more heavily than ϕ_a due to the large values of K_D and K_P. Therefore we can reduce the complexity of the NN if we eliminate ϕ_a from the output and replace it with the finite difference approximation

$$\phi_a(t) \cong \frac{\phi_d(t) - \phi_d(t-1)}{T} \tag{19}$$

where T is the sampling time. This simplified controller, call Scheme C, is shown in Figure 3 in which only ε and $\dot{\varepsilon}$ are employed for NN training. The NN controller in Scheme C is represented as

$$\Psi_C = \dot{\phi}_d + K_D\phi_d + K_P\phi_p \tag{20}$$

Consequently the number of internal weights are reduced approximately by one third. We note however that one degree of freedom is lost in (20) so that its effectiveness in compensating robot uncertainties is reduced. Simulations have confirmed this point.

VI Simulation Results

A three link elbow robot manipulator whose parameters are taken from the first three links of Puma 560 robot are used for simulation studies. The nominal system parameters are used as the basis in forming the robot model $\hat{D}(q)$ and $\hat{h}(q,\dot{q})$. A 10 Kg payload uncertainty is attached to the third link. Furthermore Coulomb friction and viscous friction forces are added to each joint where $f(\dot{q}) = 5.0sgn(\dot{q}) + 8.0(\dot{q})$. The sampling rate, T, is 5ms. Six hidden units are used in the NN controller. The controller gains are selected as $K_D = diag[20,20,20]$ and $K_P = diag[100,100,100]$. The values for η and α are listed in Table 1. The performances of the basic control Scheme A and Scheme B are tested by commanding the end point to track a circular trajectory as shown in Figure 4. The circle has a period of 4 *secs*. The corresponding joint position and velocity errors for 2 cycles are shown in Figure 5. It is clear that Scheme B converges faster with better accuracy than Scheme A. It is also seen that the robot joint trajectories converged rapidly in about 1 *sec* (one fourth of a cycle). The simulation results demonstrate the fast adaptive rate of the NN controller for meaningful on-line application.

The results of E_p E_v E_c in Table 1 are computed during the second cycle of tracking (controller is well converged after the first cycle). The update rate η is optimized for each case while α is fixed at $\alpha = 0.9$. We see that Schemes A, B and C all performed well and Scheme B is the best as expected. The improvement in accuracies provided by all the NN controller is clearly demonstrated.

Table 1. Circular tracking errors during the second iteration(η is optimized.)

	Scheme A	Scheme B	Scheme C
η	0.006	0.06	0.05
α	0.9	0.9	0.9
E_p	0.0124	0.0034	0.0177
E_v	0.5330	0.4586	3.0141
E_c	0.0011	0.0002	0.0020

$$E_p = \sum_{t=1}^{P/T} \| q_c - q \|^2 \ (rad)^2$$
$$E_v = \sum_{t=1}^{P/T} \| \dot{q}_c - \dot{q} \|^2 \ (rad/sec)^2$$
$$E_c = \sum_{j=1}^{P/T} [(x_d - x)^2 + (y_d - y)^2 + (z_d - z)^2] (m)^2$$
where $P = 4sec$ and $T = 5ms$.

Table 2 lists the simulation results of tracking a composite trajectory Again Scheme B is the best. The tracking response of Scheme B is shown in Figure 6.

Table 2. Composite tracking errors during the second iteration(η is optimized.)

	Scheme A	Scheme B	Scheme C
η	0.005	0.03	0.01
α	0.9	0.9	0.9
E_p	0.20940	0.07511	3.5344

$$E_p = \sum_{t=1}^{P/T} \| q_c - q \|^2 \ (rad)^2$$
$$E_v = \sum_{t=1}^{P/T} \| \dot{q}_c - \dot{q} \|^2 \ (rad/sec)^2$$
where $P = 16sec$ and $T = 5ms$.

VII Conclusion

A new neural network control technique for robot manipulator control is presented in this paper. The NN controller serves as the inverse model of the computed-torque controlled robot system. Three possible schemes are introduced for implementing the control strategy with varying degree of complexities. Simulation studies on trajectory tracking showed that the proposed control technique works extremely well and the controller converges very fast for on-line control. The performance of Scheme B clearly demonstrates the superiority of the proposed technique. Since the proposed schemes are aimed at modifying the desired input trajectory, it can be easily implemented at the command trajectory planning level of an existing controller without having to modify the controller's internal structure as would be required by other existing schemes.

References

[1] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. on Neural Networks*, vol. 1, pp. 4-27, 1990.

[2] T. Yabuta and T. Yamada, "Learning control using neural networks", *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 740-745, 1991.

[3] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for nonlinear system control", *IJCNN*, pp. 315-319, 1992.

[4] A. Ishiguro, T. Furuhashii, S. Okuma, and Y. Uchikawa, "A neural network compensator for uncertainties of robot manipulator", *IEEE Trans. on Industrial Electronics*, vol. 39, pp. 61-66, December, 1992.

[5] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback error learning neural network for trajectory control of a robotic manipula-

tor", *IEEE Trans. on Neural Networks*, vol. 1, pp. 251–265, 1988.

[6] T. Ozaky, T. Suzuki, T. Furuhashi, S. Okuma, and Y. Uchikawa, "Trajectory control of robotic manipulators using neural networks", *IEEE Trans. on Industrial Electronics*, vol. 38, pp. 195–202, 1991.

[7] S. Jung and T.C. Hsia, "On-line neural network control of robot manipulators", *International Conference on Neural Information Processing*, pp. 1663–1668, Seoul, 1994.

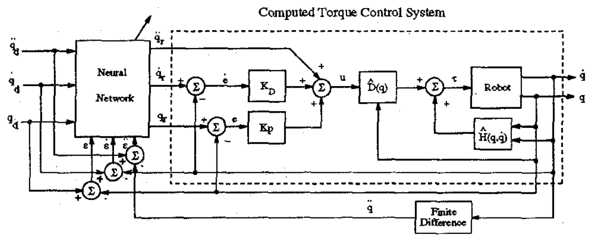

Figure 1: Scheme A:Inverse Control Structure for Computed-Torque Control

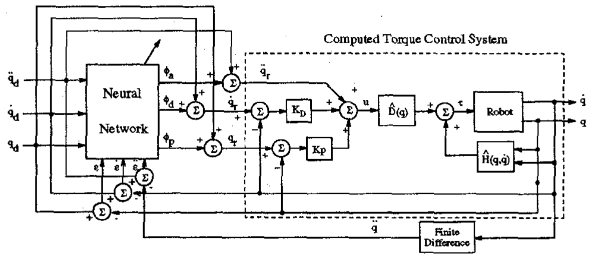

Figure 2: Scheme B : NN Compensator Structure for Computed-Torque-Control

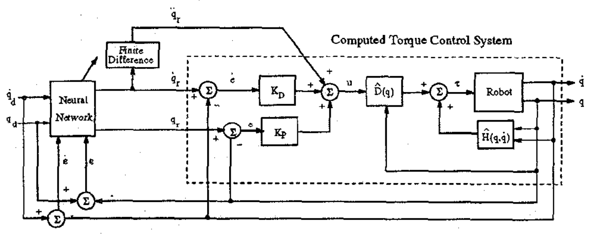

Figure 3: Scheme C:Inverse Control Structure for Computed-Torque Control

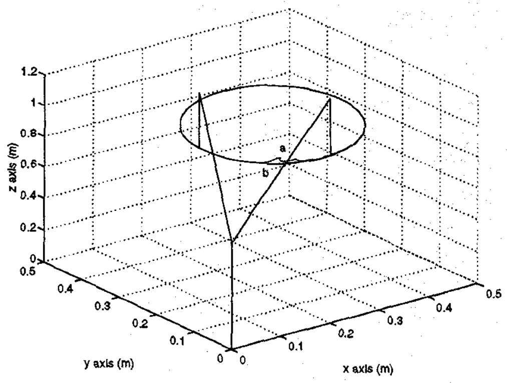

Figure 4: End Point Tracking of a Circular Trajectory for a Three Link Robot using control (a) Scheme A and (b) Scheme B

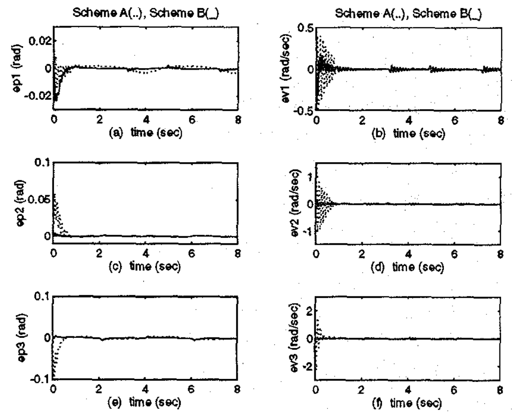

Figure 5: Joint Angle Tracking Errors for Circle Trajectory in Figure 4 where $e_p = q_d - q$ $e_v = \dot{q}_d - \dot{q}$.

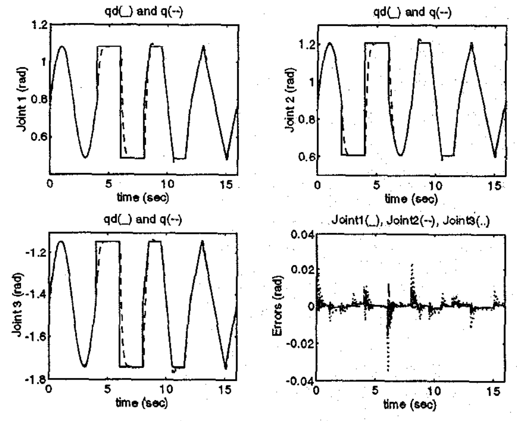

Figure 6: Composite Trajectory Tracking Response and Joint errors for Scheme B