# An Analytical Comparison of a Neural Network and a Model-Based Adaptive Controller

Richard E. Nordgren and Peter H. Meckl

*Abstract*—**A neural network inverse dynamics controller with adjustable weights is compared with a computed-torque type adaptive controller. Adaptive control techniques have been designed to accomplish the same objectives as neural network controllers, namely to achieve good performance when the system parameters are poorly known. Thus, a detailed comparison of the two approaches highlights some of the similarities and differences, both with respect to learning/adaptation laws and system performance. Lyapunov stability techniques, usually applied to adaptive systems, are used to derive a globally asymptotically stable adaptation law for a single-layer neural network controller that bears similarities to the well-known delta rule for neural networks. This alternative learning rule allows the learning rates of each connection weight to be individually adjusted to give faster convergence. The role of persistently exciting inputs to ensure parameter convergence, often mentioned in the context of adaptive systems, is emphasized in relation to the convergence of neural network weights. A coupled, compound pendulum system is used to develop inverse dynamics controllers based on adaptive and neural network techniques. Adaptation performance is compared for a model-based adaptive controller and a simple neural network utilizing both delta-rule learning and the alternative adaptation law.**

## I. Introduction

Neural networks [1], [2] show great potential for controlling systems that are difficult or impossible to model using traditional techniques. Instead of basing the control algorithm on a known model, the neural network can configure itself with appropriate connection weights in order to "learn" a control scheme that generates desired system performance. Mechanical systems exhibiting joint friction and other nonlinear systems lend themselves to such a learning-based control. In this paper, we show that the traditional mathematical tools used for adaptive systems may be applied to neural network controllers to derive adaptation laws that can be shown to be globally asymptotically stable. These adaptation laws bear a strong resemblance to the familiar delta rule for neural networks. More importantly, they permit a detailed look at the dynamics of a neural network controller during training.

The area of learning control has been studied by many researchers, with a few of the notable ones surveyed here. (Although adaptation and learning usually have different definitions, in this paper, since only system parameters are to be learned, they will be treated as being equivalent.) Raibert [3] proposed a model of the central nervous system that is capable of learning the dominant dynamic parameters (inertia, friction, spring constant, and gravitational force) of a limb while performing a specific movement. Raibert used the idea of a state vector addressable memory to store this parametric data during learning. This data was then accessed as a look-up table to generate parameters necessary to compute joint torques for a desired movement. Albus [4] generalized this approach in his cerebellar model articulation controller (CMAC) to store parametric data in locally overlapping regions so that each table entry is influenced by neighboring data points during learning. Atkeson and Reinkensmeyer [5] used an alternate approach in which the parametric data is stored locally during learning but the parameter value obtained during recall is averaged over neighboring entries. All of these techniques essentially store parametric data in a large look-up table.

Neural networks represent an alternative method for storing this parametric data. Kawato *et al.* [6] proposed an adaptive model of the human nervous system similar to that of Raibert. Here, a neural network was used as a feedforward inverse dynamics controller that was capable of learning a model of a three-axis robot with unknown dynamic parameters. The desired trajectory data was fed into a number of subfunctions characterizing the system dynamics. The outputs of these subfunctions were then appropriately weighted and summed using several neurons configured like adaptive linear combiners [7]. Although the implementation is simple, this neural network can capture rich nonlinear dynamics when a sufficiently large catalog of subfunctions is used.

In this simple form, the neural network controller looks very much like an adaptive element as described by Widrow and Stearns [7] and Fujita [8]. Each of these researchers presents an alternative method to show convergence of the final network output to its specified desired value. However, neither of these analyses treats the learning system as a feedback system, as proposed by Kawato *et al.*, with the training signal coming from the proportional-derivative (PD) controller, and hence changing as the learning process proceeds.

Stability techniques used for adaptive systems take into account the dynamic feedback nature of the adaptation process. Parks [9] was one of the first researchers to use Lyapunov stability techniques for model reference adaptive control (MRAC) to derive globally stable adaptation laws, thereby placing adaptive control on a solid mathematical footing. Craig *et al.* [10], [11] extended the work of Parks to include adaptive control of nonlinear multidimensional systems. He developed a mathematically rigorous adaptation law using Lyapunov

Manuscript received November 16, 1991; revised June 26, 1992.

The authors are with the School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907-1288

IEEE Log Number 9203726.

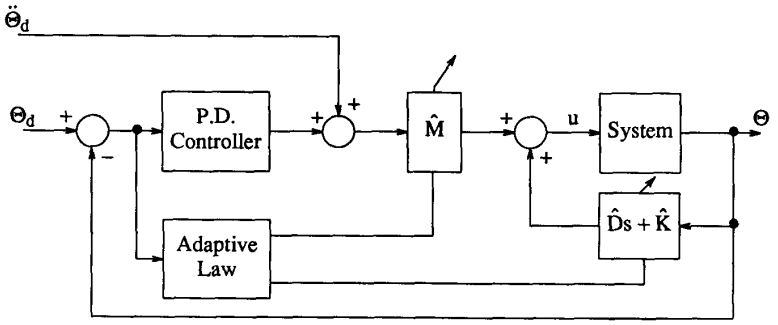



Fig. 1. Craig's model-based adaptive controller.

stability theory for the control of a three-axis robot arm. A learning scheme for handling unmodelable joint friction was also developed.

The connection between neural network controllers and adaptive controllers was exploited by Narendra and Parthasarathy [12]. They see neural networks as highly nonlinear control elements that offer distinct advantages over the more conventional linear parameter adaptive controllers in achieving desired system performance. A variety of neural network implementations, including dynamic back propagation, are discussed.

Actual comparisons between neural controllers and adaptive controllers have also been documented. Kraft and Campagna [13] discuss a comparison of three different control schemes and their application to a single problem; that of tracking a filtered square wave signal of unknown magnitude and frequency. The performance of a model reference adaptive controller, self-tuning regulator, and cerebellar model articulation controller (CMAC) were compared.

This paper goes beyond the results described by Kraft and Campagna [13] to compare not only the performance of adaptive and neural controllers but also their respective learning/adaptation laws to gain an understanding of why they perform as they do. Several adaptive linear combiners as described by Kawato [6] will be used as the neural network controller. In order to keep the analysis straightforward, only linear system models with unknown parameters will be used.

After briefly describing the adaptive control approach formulated by Craig *et al.* [10], a Lyapunov analysis will be used to derive a globally asymptotically stable adaptation law for a neural network inverse dynamics model using desired trajectory signals as inputs. The training signal will be derived from a proportional-plus-derivative (PD) controller, based on the system error trajectory. This adaptation law will then be compared to the familiar delta rule for neural networks based on a coupled, compound pendulum system model. Performance of the adaptive controller and neural controller are compared for this system. Finally, the role of persistent excitation at the inputs is highlighted for proper convergence of the system parameters (connection weights) to their actual values. This paper emphasizes the advantages of viewing neural network controllers from an adaptive control framework.

## II. Adaptive Control Development

The model-based adaptive controller, proposed by Craig *et al.* [10] for robot manipulator control, is described in this section. This control scheme combines a proportional-plus-derivative (PD) controller with an adaptive computed-torque scheme based on an estimated system model. The model parameters are updated by an adaptation law to establish a good match between the model and the actual system.

A block diagram of this control scheme is shown in Fig. 1. This diagram has been rearranged slightly from that presented in [10] to make it easier to compare with a neural network controller in the next section. In addition, the dynamic system has been restricted to a linear representation in order to clarify the analysis. The system is assumed to be described by the following matrix differential equation:

$$M\ddot{\Theta} + D\dot{\Theta} + K\Theta = \boldsymbol{u} \tag{1}$$

where $\Theta$ represents the $n \times 1$ output position vector, $\boldsymbol{u}$ represents the $n \times 1$ control input torque vector, and $M$, $D$, and $K$ are $n \times n$ time-invariant mass, damping, and stiffness matrices, respectively. The model-based adaptive controller is given by

$$\boldsymbol{u} = \hat{M}\ddot{\Theta}_d + \hat{D}\dot{\Theta} + \hat{K}\Theta + \hat{M}\left(K_d\dot{E} + K_pE\right) \tag{2}$$

where $E = \Theta_d - \Theta$ is the position error vector, $\Theta_d$ represents the desired position vector, $K_p$ and $K_d$ represent diagonal proportional and derivative gain matrices, and $\hat{M}, \hat{D}, \hat{K}$ represent the estimates of $M$, $D$, and $K$.

Combining the dynamics of the system and controller leads to the governing dynamic equation in terms of position error:

$$\ddot{E} + K_d\dot{E} + K_pE = \hat{M}^{-1}[M-\hat{M} \quad D-\hat{D} \quad K-\hat{K}]\begin{bmatrix}\ddot{\Theta}\\ \dot{\Theta}\\ \Theta\end{bmatrix}. \tag{3}$$

Craig *et al.* rewrote the right-hand-side of (3) by defining an $r \times 1$ parameter error vector $\Phi$, whose $r$ elements contain

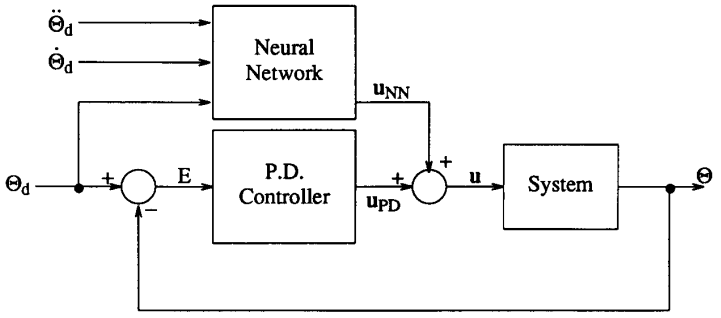



Fig. 2. Neural network control system.

all the unknown system parameters whose values are to be estimated. This leads to

$$\ddot{E} + K_d\dot{E} + K_pE = \hat{M}^{-1}W\Phi \tag{4}$$

where $W$ is an $n \times r$ matrix whose elements are functions of $\Theta$, $\dot{\Theta}$, and $\ddot{\Theta}$, and $\Phi = \Pi - \hat{\Pi}$, where $\Pi$ and $\hat{\Pi}$ represent the actual and estimated parameter vectors, respectively.

Notice that the dynamics governing the left-hand-side of (4) depend only on the controller gains. Thus, stability is easily guaranteed. The right-hand side can be treated as a forcing function driving the error dynamics. Note that this driving function depends only on the *actual* trajectory and the parameter error. The form of the controller given by (2) was specially constructed to achieve this result.

Using Lyapunov stability analysis on (4), Craig *et al.* [10] developed an adaptation law to update the parameter estimates in $\Phi$. The resulting expression is given as follows:

$$\dot{\Phi} = -\frac{d\hat{\Pi}}{dt} = -\Gamma W^T\hat{M}^{-1}E_f \tag{5}$$

where $\Gamma$ is an $n \times n$ diagonal adaptation rate gain matrix, $W^T$ denotes the matrix transpose of $W$, and $E_f$ represents a "filtered error" [10] given by

$$E_f = \dot{E} + \Psi E \tag{6}$$

where $\Psi$ is another constant diagonal matrix. The filtered error is used to ensure that the adaptive closed-loop system is strictly positive real. The importance of this condition will become clear in the next section.

## III. Derivation of Adaptation Law for Neural Controller

The technique used by Craig *et al.* [10] to develop the model-based adaptive controller will now be used to derive a similar adaptation law (or learning rule) for a neural network controller. A block diagram of the basic control structure, as described by Kawato *et al.* [6], is shown in Fig. 2. The system to be controlled is again described by (1). The basic control scheme consists of a feedforward neural network controller and a fixed-gain PD controller. The neural net is simply a series of adaptive linear combiners, for which the outputs are linear combinations of the desired trajectory signals and their derivatives. The network weights ultimately represent system parameters and are initially set to zero. Once trained, the neural network controller will represent the inverse dynamics model of the system to be controlled. The fixed-gain PD controller ensures adequate performance prior to convergence of the network weights and provides the training signal to the network for the purpose of adapting the weights. It also reduces steady-state output errors due to disturbance inputs.

Following the development of Craig *et al.* [10], we can derive a globally asymptotically stable adaptation law for a neural network. For the system given by (1), the neural network inverse dynamics model is described by:

$$\boldsymbol{u}_{NN}(t) = \hat{M}\ddot{\Theta}_d + \hat{D}\dot{\Theta}_d + \hat{K}\Theta_d \tag{7}$$

where $\Theta_d$ is the desired position vector and $\hat{M}$, $\hat{D}$, and $\hat{K}$ are the estimated mass, damping, and stiffness matrices, respectively. The PD controller is represented by

$$\boldsymbol{u}_{PD}(t) = K_p(\Theta_d - \Theta) + K_d\left(\dot{\Theta}_d - \dot{\Theta}\right) \tag{8}$$

where $K_p$and $K_d$ are diagonal proportional and derivative feedback gain matrices, respectively. Combining these equations, with $\boldsymbol{u} = \boldsymbol{u}_{NN} + \boldsymbol{u}_{PD}$, the system equation for the block diagram of Fig. 2 is given by:

$$\begin{aligned}\hat{M}\ddot{\Theta}_d + \hat{D}\dot{\Theta}_d + \hat{K}\Theta_d + K_p(\Theta_d - \Theta) + K_d\left(\dot{\Theta}_d - \dot{\Theta}\right)\\ = M\ddot{\Theta} + D\dot{\Theta} + K\Theta.\end{aligned} \tag{9}$$

Adding $M\ddot{\Theta}_d + D\dot{\Theta}_d + K\Theta_d$ to both sides and rearranging yields:

$$\begin{aligned}\hat{M}\ddot{\Theta}_d + \left(M\ddot{\Theta}_d - M\ddot{\Theta}\right) + \hat{D}\dot{\Theta}_d + \left(D\dot{\Theta}_d - D\dot{\Theta}\right)\\ + \hat{K}\Theta_d + (K\Theta_d - K\Theta)\\ + K_p(\Theta_d - \Theta) + K_d\left(\dot{\Theta}_d - \dot{\Theta}\right)\\ = M\ddot{\Theta}_d + D\dot{\Theta}_d + K\Theta_d.\end{aligned} \tag{10}$$

If we now define $E = \Theta_d - \Theta$, the foregoing equation may

be written as

$$\ddot{E} + M^{-1}(D + K_d)\dot{E} + M^{-1}(K + K_p)E = M^{-1}[M - \hat{M} \quad D - \hat{D} \quad K - \hat{K}]\begin{bmatrix}\ddot{\Theta}_d \\ \dot{\Theta}_d \\ \Theta_d\end{bmatrix} = M^{-1}W_d\Phi \tag{11}$$

where $W_d$ is an $n \times r$ matrix of functions of the desired trajectory and its derivatives and $\Phi$ is a vector of $r$ parameter errors. Notice that $W_d$ may contain known system parameters, while $\Phi$ only needs to contain the unknown parameters.

When (11) for the neural network controller is compared with (4) for the adaptive controller, some differences become apparent. The right-hand-side of (11) depends on functions of the *desired* trajectory, rather than the actual trajectory, as was the case in (4). Also, the actual inverse mass matrix scales the right-hand-side magnitude instead of the estimated inverse mass matrix. Finally, the left-hand-side of (11) depends on the actual system parameters as well as the controller parameters. This makes it more difficult to ensure stable performance.

In order to ascertain stability of the forced system given by (11), it is necessary to investigate the stability of the unforced system, which can be represented as

$$\ddot{E} + G\dot{E} + HE = 0 \tag{12}$$

where $G$ and $H$ represent the coefficient matrices of $\dot{E}$ and $E$ given in (11). The unforced system (12) may be shown to be stable for relatively mild constraints on the $G$ and $H$ matrices. If $G = G^T > 0$, $H = H^T > 0$, then the Lyapunov function, defined by

$$V = \dot{E}^T G\dot{E} + E^T GHE \tag{13}$$

can be used to show that this system is globally asymptotically stable. By differentiating the expression for $V$, we obtain

$$\dot{V} = -2\dot{E}^T G^2 \dot{E} \tag{14}$$

which is negative definite, showing that the origin is globally asymptotically stable. Thus, the unforced system can be stabilized by proper choice of controller gain matrices $K_p$ and $K_d$.

The problem with the formulation of (11) is that although the error equation has constant coefficients, the matrices $M$, $D$, and $K$ are unknown. If we assume that the elements of these matrices lie in some bounded intervals, we can choose appropriate values of $K_p$ and $K_d$ such that the error equation poles are sufficiently damped and have acceptable time constants to ensure convergence. However, the uncertainty in parameters results in overdesign of the gains for the PD controller.

We now take the Laplace transform of (11), where we define the transform of the term on the far right as $F(s)$, an effective forcing function. We refer to (15) as the error transfer function:

$$E(s) = \left[s^2 I + sM^{-1}(D + K_d) + M^{-1}(K + K_p)\right]^{-1} F(s). \tag{15}$$

Since the error transfer function is not strictly positive real (it has two more poles than zeros), it is cascaded with a noncausal filter $Z(s) = [sI + \Psi]$, thereby adding a zero. This filter may be implemented by using rate sensors to eliminate the causality problem. If we define the filter

$$E_f = \dot{E} + \Psi E, \tag{16}$$

we may write our filtered error transfer function as

$$E_f(s) = [sI + \Psi] \cdot \left[s^2 I + sM^{-1}(D + K_d) + M^{-1}(K + K_p)\right]^{-1} F(s). \tag{17}$$

This transfer function is strictly positive real (SPR), therefore, by the SPR lemma [14], we are guaranteed the existence of two symmetric positive definite matrices $P$ and $Q$ that satisfy

$$A^T P + PA = -Q \qquad PB = C^T \tag{18}$$

where $A$, $B$, and $C$ are the matrices for (11) written in state-space form, shown as follows:

$$\dot{X} = AX + BM^{-1}W_d\Phi \qquad E_f = CX \tag{19}$$

where $X$ is the error state vector defined by $X = [E \quad \dot{E}]^T$.

A globally asymptotically stable (GAS) adaptation scheme may be developed using Lyapunov stability theory. Using the same choice as Craig *et al.* [10], we define a Lyapunov function of the form:

$$V(X, \Phi) = X^T PX + \Phi^T \Gamma^{-1} \Phi \tag{20}$$

where $P$ is the positive definite matrix defined in (18) and $\Gamma$ is an arbitrary $n \times n$ diagonal matrix. Differentiating $V$ in (20) and substituting $\dot{X}$ from (19) leads to

$$\dot{V}(X, \Phi) = X^T(A^T P + PA)X + 2\Phi^T \Gamma^{-1}\dot{\Phi} + X^T PBM^{-1}W_d\Phi + \Phi^T W_d^T M^{-T} B^T PX \tag{21}$$

where $M^{-T}$ denotes the transpose of the inverse of the mass matrix $M$. Since the mass matrix is in general symmetric, $M^{-T}$ can be replaced by $M^{-1}$. Substituting $Q$ and $C$ from (18) yields

$$\dot{V}(X, \Phi) = -X^T QX + 2\Phi^T \Gamma^{-1}\dot{\Phi} + X^T C^T M^{-1} W_d \Phi + \Phi^T W_d^T M^{-1} CX \tag{22}$$

Substituting $E_f$ from (19) gives

$$\dot{V}(X, \Phi) = -X^T QX + 2\Phi^T \Gamma^{-1}\dot{\Phi} + E_f^T M^{-1} W_d \Phi + \Phi^T W_d^T M^{-1} E_f. \tag{23}$$

Finally,

$$\dot{V}(X, \Phi) = -X^T QX + 2\Phi^T \left(W_d^T M^{-1} E_f + \Gamma^{-1}\dot{\Phi}\right). \tag{24}$$

If we choose our adaptation law to be

$$\dot{\Phi} = -\frac{d\hat{\Pi}}{dt} = -\Gamma W_d^T M^{-1} E_f \tag{25}$$

then

$$\dot{V}(X, \Phi) = -X^T QX \tag{26}$$

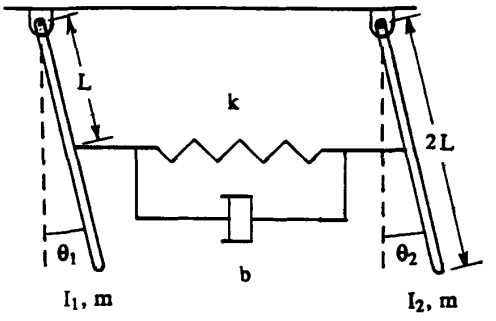



Fig. 3. Schematic of coupled compound pendulum system

which is nonpositive. We have now satisfied the stipulations of Lyapunov's second law (the direct method) for the trajectory error ($E$ and $\dot{E}$) to asymptotically converge to zero using the adaptation law given by (25). Although $M$ is unknown in this adaptation law, if it is a diagonal matrix of constants, it may be lumped in with $\Gamma$, an adaptation rate gain matrix. Since the magnitudes of the elements of the $\Gamma$ matrix are theoretically unconstrained, stability is maintained over any range of actual inertia parameters in $M$.

## IV. Application to a Coupled Compound Pendulum

In order to compare the adaptive controller to a neural network controller, it is useful to look at a specific example. The system to be investigated here consists of a double compound pendulum with a spring and damper providing coupling between the two pendulums (Fig. 3).

The linearized double-pendulum dynamics may be modeled as:

$$M\ddot{\Theta} + D\dot{\Theta} + K\Theta = \boldsymbol{u} \tag{27}$$

where $\Theta = [\theta_1\, \theta_2]^T$, the inertia matrix $M = \text{diag}(I_1, I_2)$, torque vector $\boldsymbol{u} = [\tau_1\, \tau_2]^T$, the damping matrix

$$D = \begin{bmatrix} bL^2 & -bL^2 \\ -bL^2 & bL^2 \end{bmatrix} \tag{28}$$

and the stiffness matrix

$$K = \begin{bmatrix} (mgL + kL^2) & -kL^2 \\ -kL^2 & (mgL + kL^2) \end{bmatrix}. \tag{29}$$

Before comparing the neural network controller derived in the previous section to Craig's adaptive controller, it is instructive to compare the algorithm of (25) with a standard neural network learning rule. For the double-pendulum system of Fig. 3, two separate control torques are required to provide arbitrary trajectories in $\theta_1$ and $\theta_2$. Thus, two adaptive linear combiners are necessary to represent the inverse dynamics model for the system, as given by the following expression:

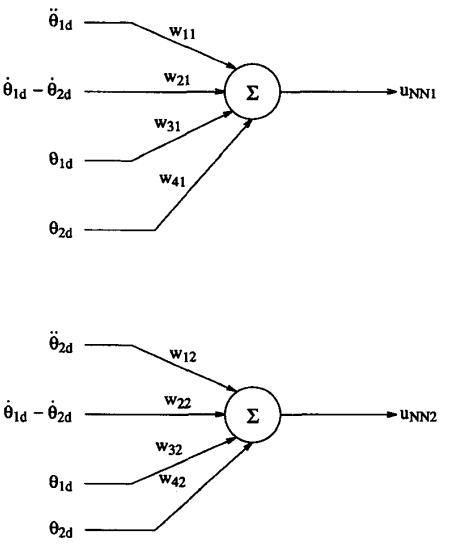



Fig. 4. Neural network inverse dynamics model.

$$\boldsymbol{u}_{NN} = \hat{M}\ddot{\Theta}_d + \hat{D}\dot{\Theta}_d + \hat{K}\Theta_d. \tag{30}$$

Each adaptive linear combiner can be considered a neuron of a simple neural network.

Since the inertia matrix is diagonal, it is reasonable to leave out one of the angular acceleration inputs for each neuron. In addition, the form of the damping matrix in (28) suggests that only the signal $\dot{\theta}_{1d} - \dot{\theta}_{2d}$ must be provided to both neurons to estimate the damping parameter $b$. This leaves four connection weights per neuron, or a total of eight connection weights to represent the unknown system parameters (Fig. 4). The elements of the control input vector $\tau_i$ provided by the inverse dynamics controller can be written in terms of these connection weights as follows:

$$\tau_i = \boldsymbol{w}_i^T \begin{bmatrix} \ddot{\theta}_{id} & \left(\dot{\theta}_{1d} - \dot{\theta}_{2d}\right) & \theta_{1d} & \theta_{2d} \end{bmatrix}^T, i = 1, 2 \tag{31}$$

where

$$\boldsymbol{w}_i^T = [\, w_{1i} \quad w_{2i} \quad w_{3i} \quad w_{4i}\,]. \tag{32}$$

For an adaptive linear combiner, the traditional learning algorithm is the delta rule, a classic gradient descent technique

$$\Pi = [\, I_1 \quad bL^2 \quad (mgL + kL^2) \quad -kL^2 \quad I_2 \quad -bL^2 \quad -kL^2 \quad (mgL + kL^2)\,]^T \tag{37}$$

$$W_d = \begin{bmatrix} \ddot{\theta}_{1d} & \dot{\theta}_{1d} - \dot{\theta}_{2d} & \theta_{1d} & \theta_{2d} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddot{\theta}_{2d} & \dot{\theta}_{1d} - \dot{\theta}_{2d} & \theta_{1d} & \theta_{2d} \end{bmatrix} \tag{38}$$

[7]. For each neuron, this algorithm is given by

$$\Delta \boldsymbol{w}_i = \eta_i \delta_i \begin{bmatrix} \ddot{\theta}_{id} \\ \dot{\theta}_{1d} - \dot{\theta}_{2d} \\ \theta_{1d} \\ \theta_{2d} \end{bmatrix}, i = 1, 2 \tag{33}$$

where $\delta_i$ represents the difference between desired output and actual output and $\eta_i$ represents a learning rate constant, for each combiner.

Since the neural controller is to generate an inverse dynamics model, the desired output would be the appropriate torque signals to achieve a desired trajectory. Since this cannot be known without an accurate system model in the first place, an alternative strategy was proposed by Kawato *et al.* [6] to establish $\delta_i$. In essence, the output of the PD controller is an indication of the network error $\delta_i$, since if the true inverse dynamics model has been learned, the neural controller alone will provide the necessary torques to achieve the desired trajectory (see Fig. 2). With zero trajectory error, the PD controller produces no output and hence indicates that learning has been completed.

With the error signal $\delta_i$ defined as follows,

$$\delta_i = K_{di}\dot{E}_i + K_{pi}E_i \tag{34}$$

the delta rule can be written as

$$\frac{d\boldsymbol{w}_i}{dt} \approx \frac{\Delta \boldsymbol{w}_i}{\Delta t} = \frac{\eta_i}{\Delta t}\left(K_{di}\dot{E}_i + K_{pi}E_i\right) \begin{bmatrix} \ddot{\theta}_{id} \\ \dot{\theta}_{1d} - \dot{\theta}_{2d} \\ \theta_{1d} \\ \theta_{2d} \end{bmatrix}, i = 1, 2. \tag{35}$$

The difference quotient $\Delta \boldsymbol{w}_i/\Delta t$ can be approximated by the derivative $d\boldsymbol{w}_i/dt$ for a sufficiently small time step $\Delta t$. Since the connection weights are updated at a constant rate, the time step $\Delta t$ scales the learning rate $\eta_i$. Thus, $\eta_i$ must be adjusted to maintain the same weight adaptation rate if the corresponding time step $\Delta t$ is changed.

To compare this learning rule with the GAS learning rule derived in the previous section, it is necessary to define the parameter error vector $\Phi$ and matrix $W_d$ used in (25):

$$\Phi = \Pi - \hat{\Pi} = \Pi - \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix} \tag{36}$$

where $\Pi$ represents the vector of actual parameter values (assumed constant): see (37) and (38) at the bottom of the previous page. The resulting adaptation law using (25) can be written as

$$\frac{d\boldsymbol{w}_i}{dt} = \frac{1}{I_i}\Gamma_i\left(\dot{E}_i + \psi_i E_i\right) \begin{bmatrix} \ddot{\theta}_{id} \\ \dot{\theta}_{1d} - \dot{\theta}_{2d} \\ \theta_{1d} \\ \theta_{2d} \end{bmatrix}, i = 1, 2 \tag{39}$$

where $\Gamma_i$ represents a 4×4 diagonal matrix of adaptation rate gains. In contrast to delta rule learning, the rate of weight adaptation is unaffected by the time step.

When the delta rule expressed by (35) is compared with the GAS adaptation law in (39), a number of interesting observations can be made. First, notice that the use of the PD controller output as training signal in the delta rule plays the same role as the filtered error in the GAS adaptation law. Thus, an alternative approach has been used to indicate that the delta rule, using the PD control signal as output error, is in fact a globally asymptotically stable adaptation law for appropriate values of the PD controller gains. The advantage of the Lyapunov approach used here is that it takes into account all of the coupled dynamics that occur when the PD controller output is used as a training signal. Since this signal changes continuously as the neural network is learning, the overall stability of the learning algorithm is not obvious. The Lyapunov approach suggests that the delta rule can indeed provide stable performance and give convergence to appropriate parameter values under these conditions.

Another important observation is the presence of a diagonal matrix $\Gamma_i$ in the GAS algorithm, in place of a single adaptation gain $\eta_i$ in the delta rule. This matrix provides added flexibility in the selection of adaptation gains for the GAS algorithm. Instead of requiring that all connection weights (parameter estimates) for each neuron converge at the same rate, as is the case for the delta rule, the GAS algorithm allows different convergence rates for different parameters. It should be emphasized that these stability results are limited to single-layer neural networks configured as adaptive linear combiners. However, a similar approach can also be used for multilayer neural networks to enhance learning. The advantage of this added flexibility will become clear when the learning performance for the different algorithms is compared.

## V. Learning Performance Comparison

The double-pendulum system of Fig. 3 will now be used to compare the learning performance of the adaptive controller and three different neural network controllers. Craig's model-based adaptive controller will be compared with the neural network inverse dynamics controller utilizing either the GAS adaptation law or the delta learning rule. The GAS adaptation law will be tested with two different parameter error vectors, one resembling that used with the adaptive controller, the other resembling that used with the neural network of Fig. 4.

For the specific double-pendulum system, parameters $I_1, I_2$, $b$, and $k$ are considered unknown, while the rest of the parameters, $m$ and $L$, are considered known. Thus, the most compact form for the parameter error vector $\Phi$ for the model-based adaptive controller is given by

$$\Phi = \Pi - \hat{\Pi} = \left[\left(I_1 - \hat{I}_1\right)\left(I_2 - \hat{I}_2\right)\left(b - \hat{b}\right)\left(k - \hat{k}\right)\right]^T \tag{40}$$

and the corresponding adaptation law is given by

$$\frac{d\hat{\Pi}}{dt} = \Gamma W^T \hat{M}^{-1} E_f \tag{41}$$

where $\Gamma$ = diag $(\gamma_1, \gamma_2, \gamma_3, \gamma_4)$,

$$W = \begin{bmatrix} \ddot{\theta}_1 & 0 & L^2\left(\dot{\theta}_1 - \dot{\theta}_2\right) & L^2(\theta_1 - \theta_2) \\ 0 & \ddot{\theta}_2 & L^2\left(\dot{\theta}_2 - \dot{\theta}_1\right) & L^2(\theta_2 - \theta_1) \end{bmatrix} \tag{42}$$

and

$$E_f = \begin{bmatrix} \left(\dot{\theta}_{1d} - \dot{\theta}_1\right) + \psi_1(\theta_{1d} - \theta_1) \\ \left(\dot{\theta}_{2d} - \dot{\theta}_2\right) + \psi_2(\theta_{2d} - \theta_2) \end{bmatrix}. \tag{43}$$

The adaptive controller must identify four parameters for the system, that is, $r = 4$.

The adaptation algorithm for the neural network controller using the GAS adaptation law, and the same form of parameter error vector as the adaptive controller, is given by

$$\frac{d\hat{\Pi}}{dt} = \Gamma W_d^T M^{-1} E_f \tag{44}$$

where $\hat{\Pi} = \begin{bmatrix} \hat{I}_1 & \hat{I}_2 & \hat{b} & \hat{k} \end{bmatrix}^T$ as in Craig's scheme and $W_d$ is the same as $W$ in Craig's scheme with the actual joint angles replaced by desired joint angles. Again the number of parameters to be identified is four ($r = 4$).

The adaptation algorithm for the neural network controller again utilizing the GAS adaptation law, but with the parameter error vector corresponding to the neural network of Fig. 4, is given by

$$\frac{d\boldsymbol{w}_i}{dt} = \frac{1}{I_i}\Gamma_i\left(\dot{E}_i + \psi_i E_i\right)\begin{bmatrix} \ddot{\theta}_{id} \\ \dot{\theta}_{1d} - \dot{\theta}_{2d} \\ \theta_{1d} \\ \theta_{2d} \end{bmatrix}, i = 1, 2 \tag{45}$$

where $\Gamma_i = \text{diag}[\gamma_{1i}, \gamma_{2i}, \gamma_{3i}, \gamma_{4i}]$ represents individual adaptation rate gains for each parameter, and the connection weight vectors $\boldsymbol{w}_i$ represent the four unknown parameters for each neuron, eight in total. Notice that (44) incorporates the fact that $\theta_1$ and $\theta_2$ only appear as the difference $\theta_1 - \theta_2$, while the formulation of (45) requires the network to determine this fact.

The adaptation law for the neural network controller of Fig. 4 utilizing delta rule learning is given by

$$\frac{d\boldsymbol{w}_i}{dt} \approx \frac{\Delta \boldsymbol{w}_i}{\Delta t} = \frac{\eta_i}{\Delta t}\left(K_{di}\dot{E}_i + K_{pi}E_i\right)\begin{bmatrix} \ddot{\theta}_{id} \\ \dot{\theta}_{1d} - \dot{\theta}_{2d} \\ \theta_{1d} \\ \theta_{2d} \end{bmatrix}, i = 1, 2 \tag{46}$$

where the connection weight vectors $\boldsymbol{w}_i$ represent the four unknown parameters for each neuron.

Values for each of the system parameters, controller parameters, and adaptation gains are given in Tables I and II. The PD controller parameters have been selected to give a closed-loop natural frequency of 5 rad/s with critical damping for the adaptive control system. Initial parameter or weight values were selected so that $I_{1o} = I_{2o} = 1$ kg-m$^2$, $b_o = 0.8$ N-s/m, and $k_o = 0.8$ N/m.

The desired trajectories to be followed by the two pendulums are given by

$$\theta_{1d}(t) = 2sin4t - \sin 8t \tag{47}$$

and

$$\theta_{2d}(t) = 1 + \cos 4t - \cos 8t. \tag{48}$$

TABLE I
DOUBLE PENDULUM SYSTEM PARAMETERS

| | | |
|---|---|---|
| $L$ | = | 1 m |
| $m$ | = | 1 kg |
| $I_1$ | = | 1.33 kg-m$^2$ |
| $I_2$ | = | 1.33 kg-m$^2$ |
| $b$ | = | 1 N-s/m |
| $k$ | = | 1 N/m |

These expressions were chosen to guarantee persistent excitation, which will be explained in detail in the next section. Initial conditions for the system were selected to coincide with the initial desired angles and corresponding zero velocities to minimize transient effects during parameter adaptation.

Simulation results for each type of system are given in Figs. 5, 6, and 7. Only the result for inertia estimate $I_1$ is given since the behavior for the other parameters is similar. In each case, the inertia estimate error is given as a fraction of the "true" value. Fig. 5 shows the inertia estimate error as a function of time for the adaptive controller. Fig. 6 shows the inertia estimate error for the neural network controller using the GAS adaptation law for two different choices of parameter error vector. Fig. 7 shows the inertia estimate error for the neural network controller with delta rule learning.

In every case except the neural network with delta rule learning, the inertia parameter converges within approximately 4 s. Whether the GAS adaptation law is used with the parameter error vector of the adaptive controller (44) or that of the more conventional neural network of Fig. 4 (45), the parameter convergence is similar. However, somewhat faster convergence is achieved for the parameter error vector associated with the adaptive controller (Fig. 6(a)) compared to the conventional scheme (Fig. 6(b)) since additional information about the system model is incorporated into the formulation of (44). Notice that (45) can be viewed as a modified delta rule in which each parameter adaptation rate can be individually adjusted.

The inertia parameter estimate for the neural network with conventional delta rule learning fails to converge within the 12 s shown in Fig. 7. This is primarily the result of a relatively large learning rate required to improve the convergence rate of the stiffness parameter. Better inertia parameter convergence could be obtained at the expense of dramatically longer convergence times for the stiffness parameter.

The major reason that the neural network with delta rule learning takes longer to converge is that individual parameter adaptation rates cannot be independently set, as is the case for GAS adaptation. Basically, some parameters converge rapidly, while others converge slowly, even with the same adaptation rate gain. In this example, the weights corresponding to derivatives of the input converge significantly faster than those corresponding to the input alone. If the gain is increased to speed up learning of the slow parameters, the fast parameters begin to exhibit highly erratic adaptation, as shown in Fig. 7. The advantage of the GAS adaptation law derived here is that

TABLE II
ADAPTATION PARAMETERS

| Adaptive Controller | Neural Controller with GAS Adaptation (44) | Neural Controller with GAS Adaptation (45) | Neural Controller with Delta Rule Learning |
|---|---|---|---|
| $\Psi_1 = 1$ | $\Psi_1 = 1$ | $\Psi_1 = 1$ | |
| $\Psi_2 = 1$ | $\Psi_2 = 1$ | $\Psi_2 = 1$ | |
| $\gamma_1 = 1$ | $\gamma_1 = 2$ | $\gamma_{11} = \gamma_{12} = 1$ | $\eta_1 = 0.1$ |
| $\gamma_2 = 1$ | $\gamma_2 = 2$ | $\gamma_{21} = \gamma_{22} = 5$ | $\eta_2 = 0.1$ |
| $\gamma_3 = 2$ | $\gamma_3 = 4$ | $\gamma_{31} = \gamma_{42} = 100$ | |
| $\gamma_4 = 15$ | $\gamma_4 = 20$ | $\gamma_{41} = \gamma_{32} = 50$ | |
| $\Delta t = 0.1$ ms | $\Delta t = 0.1$ ms | $\Delta t = 0.1$ ms | $\Delta t = 0.1$ ms |

it allows each of the parameter rate gains to be so adjusted that all parameters converge at the same fast rate (Fig. 6(b)).

## VI. PERSISTENT EXCITATION

The adaptation law of (25), derived using Lyapunov stability analysis, only ensures that the error between desired and actual trajectory will eventually go to zero. It does not guarantee that the final parameter estimates actually converge to their "true" values. In order to guarantee parameter convergence, the parameter error vector $\Phi$ must also eventually go to zero. Thus, global asymptotic stability must be sought for the expanded system including the dynamics of the parameter error vector $\Phi$:

$$\begin{bmatrix} \dot{X} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} A & BM^{-1}W_d \\ -\Gamma W_d^T M^{-1} C & 0 \end{bmatrix} \begin{bmatrix} X \\ \Phi \end{bmatrix} \tag{49}$$

Conditions that guarantee that the origin of this expanded system is globally asymptotically stable (GAS) are that the $A$ matrix be stable, $(A, B)$ be a controllable pair, and the matrix $W_d$ of internal signals be piecewise continuous, bounded, and persistently exciting [14].

In order for $W_d$ to be persistently exciting, it must satisfy the following condition [11]:

$$\alpha I_r \leq \int_{t_o}^{t_o+\rho} W_d^T W_d dt \leq \beta I_r \tag{50}$$

where $I_r$ is an $r \times r$ identity matrix and $\rho$ represents a suitable time interval. This condition requires that $W_d$ "must vary sufficiently over the interval $\rho$ so that the entire $r$-dimensional space is spanned" [11]. Since $W_d$ is a matrix of functions of the desired input trajectories, the persistent excitation condition can be satisfied with desired trajectory signals $\Theta_d$ that have as many spectral lines as there are unknown parameters in $\Phi$. A simple sinusoidal signal actually contains two spectral lines, hence is persistently exciting of order 2. Thus, in the compound pendulum example, desired trajectory signals $\theta_{1d}$ and $\theta_{2d}$ that are different from each other and contain at least two distinct frequencies will satisfy the persistently exciting condition, since in each controller implementation the number of unknown parameters is four. For the neural network implementation, even though there are a total of eight parameters, each neuron can be treated separately as containing only four parameters.

Using the persistently exciting trajectories given by (47) and (48) results in parameter estimates that converge to their appropriate values, as seen in Figs 5-7. If, however, input signals are used which are not persistently exciting, then the estimated parameters may converge to entirely different values, with disturbing consequences. A particular example of what can happen for the neural network controller is indicated in Fig. 8. The input trajectory $\theta_{1d}$ of Fig. 8(a) consists of two distinctly different waveforms, with a nonpersistently exciting waveform used for adaptation. The effects on parameter error in $I_1$ and trajectory error in $\theta_1$ are shown in Fig. 8(b) and (c), respectively, with the delta rule learning law used for parameter adaptation. To enhance learning, different learning rates have been used for each input signal: $\eta = 1 \times 10^{-3}$ for $\theta_{1d}$ and $\theta_{2d}$, $\eta = 5 \times 10^{-5}$ for $\dot{\theta}_{1d} - \dot{\theta}_{2d}$, and $\eta = 1 \times 10^{-5}$ for $\ddot{\theta}_{1d}$ and $\ddot{\theta}_{2d}$.

For the first 5 s, the input trajectories are $\theta_{1d}(t) = 1 - \cos 2t$ and $\theta_{2d}(t) = \sin 2t$, which are clearly not persistently exciting for this system. During this period, the trajectory error does converge to zero, as the Lyapunov analysis suggests. However, the parameter error converges to a nonzero value. At 5 s, learning is turned off, and the neural network controller is exercised for the same nonpersistently exciting inputs. After 6.28 s, the system is exercised on the trajectories given by (47) and (48), which differ from those used for adaptation. From 5–6.28 s, the trajectory error remains close to zero, despite the erroneous parameter estimates. After 6.28 s, however, when different trajectories are exercised, the erroneous parameter estimates in the inverse dynamics model result in significant trajectory error. Thus, it is important to train a neural network on persistently exciting input trajectories to guarantee acceptable trajectory error for "novel" trajectory inputs.

Although it is easy to recognize the importance of persistently exciting inputs, it is much more difficult to ensure that practical trajectory input signals will be persistently exciting. The foregoing analysis has been restricted to a simple class of single-layer neural networks. Similar results are likely to hold for more complicated multilayer networks, requiring inputs with even more frequency components to ensure convergence of the large number of connection weights. Achieving this condition may be impossible. Nonetheless,

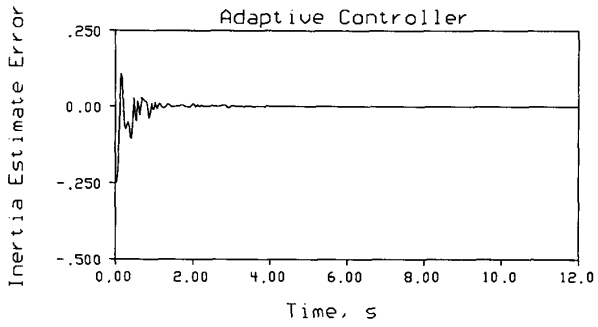



Fig. 5. Inertia estimate error for an adaptive controller.

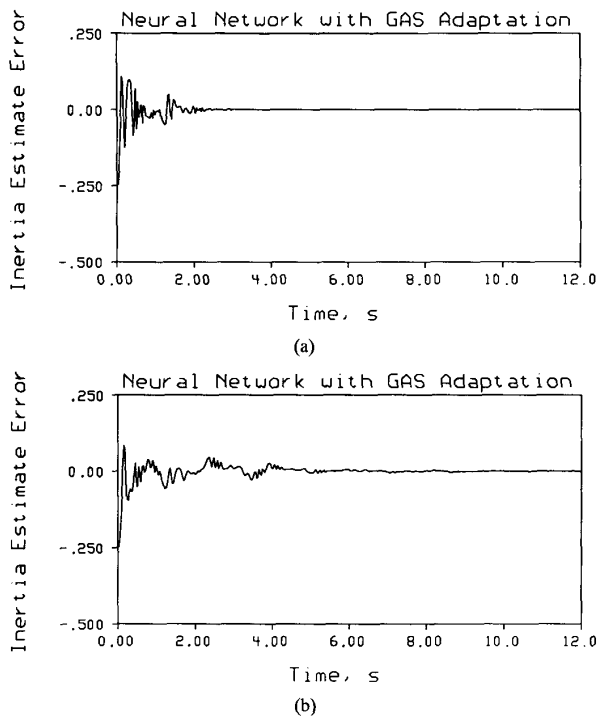



Fig. 6. Inertia estimate error for neural controller with the GAS adaptation law. (a) Using (44). (b) Using (45).

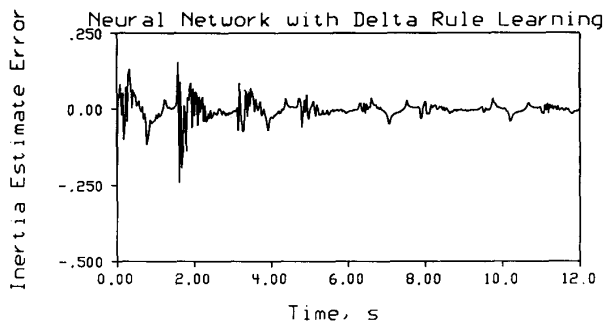



Fig. 7. Inertia estimate error for neural controller with delta rule learning.

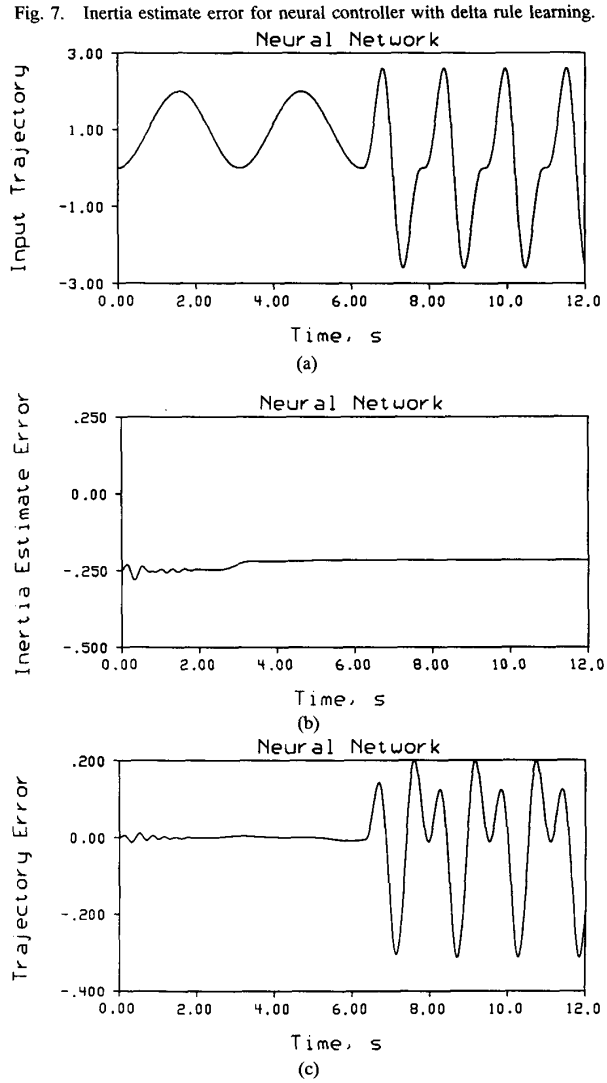



Fig. 8. Demonstration of nonpersistently exciting input. (a) Desired input trajectory $\theta_{1d}$. (b) Error in inertia estimate $I_1$. (c) Error in trajectory $\theta_1$.

it is crucial that the consequences of such a limitation be clearly understood. The aim of this paper is to point out the importance of picking suitable training trajectories that ensure good performance once learning has ceased. If a particular nonpersistently exciting signal is used during network training and a sufficiently different trajectory is used during execution, significant trajectory errors may result.

## VII. Conclusions

Concepts from adaptive control have been used to derive an alternative learning rule for a simple single-layer neural network configured as an inverse dynamics controller. This globally asymptotically stable adaptation law bears similarities to the traditional delta rule for a simple adaptive linear combiner. However, the stability analysis takes into account the dynamic interaction between the neural network controller and the rest of the system when the output of a PD controller is used as a training signal.

The advantage of the alternative adaptation law is that it allows selective tuning of the individual learning rates in order to optimize the time it takes for the trajectory error to converge to zero. This will usually lead to faster learning, as demonstrated by a comparison of adaptive and neural network controllers on a compound pendulum.

Finally, the role of persistently exciting inputs in training neural network controllers is highlighted. It is shown that if different trajectories are used to train and execute a neural network controller, then unless the training input is persistently exciting, the trajectory during execution can have significant errors.

The foregoing conclusions have been derived for a single-layer neural network consisting of several adaptive linear combiners. As such, they represent preliminary results regarding neural network stability and convergence that may be extended to the case of multilayer networks in the future.

## REFERENCES


[1] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986.

[2] W T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*. Cambridge, MA: MIT Press, 1990.

[3] M. H. Raibert, "A model for sensorimotor control and learning," *Biological Cybern.*, vol. 29, pp. 29-36, 1978.

[4] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *ASME J. Dynamic Syst., Measurement, Contr.*, vol. 97, pp. 220-227, Sept. 1975.

[5] C G. Atkeson and D. J. Reinkensmeyer, "Using associative content-addressable memories to control robots," in *Proc. IEEE Conf. on Decision and Control*, Austin, TX, Dec. 1988, pp. 792-797.

[6] M. Kawato, K. Furukawa, and R. Suzuki, "A hierarchical neural-network model for control and learning of voluntary movement," *Biological Cybern.*, vol. 57, pp. 169-185, 1987.

[7] B. Widrow, and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[8] M. Fujita, "Adaptive filter model of the cerebellum," *Biological Cybern.*, vol. 45, pp. 195-206, 1982.

[9] P. C. Parks, "Lyapunov redesign of model reference adaptive control systems," *IEEE Trans. Automatic Contr.*, vol. AC-11, no. 3, pp. 362-367. July 1966.

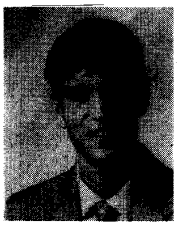
[10] J. J. Craig, P. Hsu, and S. S. Sastry, "Adaptive control of mechanical manipulators," *The Int. J. Robotics Res.*, vol. 6, no. 2, pp. 16-28, Summer 1987.

[11] J. J. Craig, *Adaptive Control of Mechanical Manipulators*. Reading, MA: Addison-Wesley, 1988.

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

[13] L. G. Kraft and D. P. Campagna, "A comparison of CMAC neural network and traditional adaptive control systems," in *Proc. Am. Control Conf.*, Pittsburgh, PA, June 1989, pp. 456-477.

[14] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Englewood Cliffs, NJ, Prentice-Hall, 1989.


**Richard E. Nordgren** was born in Ann Arbor, MI. He received the B.S. degree from Worcester Polytechnic Institute, Worcester, MA, and the M.S. degree from Purdue University, West Lafayette, IN, in 1984 and 1991, respectively, both in mechanical engineering.

From 1984 to 1987 he worked as a Nuclear Engineer with Westinghouse Electric Corp. at the Naval Reactors Facility, Idaho Falls, ID. Since 1988 he has been a Teaching Assistant and Research Assistant in the Department of Mechanical Engineering at Purdue, where he is currently working toward the Ph.D. degree. His research interests are in the areas of neural networks, frequency-domain techniques in robust control design, and quantitative feedback theory.



**Peter H. Meckl** received the B.S. degree in mechanical engineering from Northwestern University, Evanston, IL, and the S.M. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1981, 1984, and 1988, respectively.

He is currently an Assistant Professor of Mechanical Engineering at Purdue University, West Lafayette, IN, where he teaches automatic control systems and microprocessors in electromechanical systems. His research interests are in the area of dynamic systems and control, with emphasis on active vibration control and intelligent control systems utilizing neural networks. His application interests include robotic manipulators, flexible structures, and computer disk drives.