Adaptive Control of a Class of Nonlinear Discrete-Time Systems Using Neural Networks

Fu-Chuang Chen, *Member, IEEE*, and Hassan K. Khalil, *Fellow, IEEE*,

Abstract— **Layered neural networks are used in a nonlinear self-tuning adaptive control problem. The plant is an unknown feedback-linearizable discrete-time system, represented by an input–output model. To derive the linearizing-stabilizing feedback control, a (possibly nonminimal) state-space model of the plant is obtained. This model is used to define the zero dynamics, which are assumed to be stable, i.e., the system is assumed to be minimum phase. A linearizing feedback control is derived in terms of some unknown nonlinear functions. A layered neural network is used to model the unknown system and generate the feedback control. Based on the error between the plant output and the model output, the weights of the neural network are updated. A local convergence result is given. The result says that, for any bounded initial conditions of the plant, if the neural network model contains enough number of nonlinear hidden neurons and if the initial guess of the network weights is sufficiently close to the correct weights, then the tracking error between the plant output and the reference command will converge to a bounded ball, whose size is determined by a dead-zone nonlinearity. Computer simulations verify the theoretical result.**

I. Introduction

ADAPTIVE control of linear systems has been an active research area in the past two decades. It is only recently that issues related to adaptive control of feedback-linearizable nonlinear systems are addressed, e.g., [1], [3], and [4]. An important assumption in previous work on nonlinear adaptive control is the linear dependence on the unknown parameters, i.e., the unknown nonlinear functions in the plant have the form

$$f(\cdot) = \sum_{i=1}^{n} \theta_i f_i(\cdot) \tag{1}$$

where f_i's are known functions. The linear parameterization (1) is also used in the recent work [2] on applying Gaussian networks to an adaptive control problem.

Multilayer neural networks can be considered as general tools for modeling nonlinear functions. The network comprises fixed (sigmoid-type) nonlinearities and adjustable weights which appear nonlinearly. Learning algorithms are used to adjust the weights so as to reduce the modeling error. With the introduction of the back-propagation learning algorithm by Rumelhart *et al.* [5] in 1986, the multilayer neural network has become a popular architecture for practical applications in many areas, including system identification and control, signal processing, and pattern classification. Some of the activities in identification and control problems have been reported in [6].

The idea of applying multilayer neural networks to adaptive control of feedback-linearizable discrete-time systems appeared in [7] and [8]. In [13] we presented a convergence result for adaptive regulation using multilayer neural networks. Many restrictive assumptions, however, were made in [13]. As we worked on relaxing these assumptions, it became clear that the updating rule used in [13] would not be adequate. To cope with this problem, we incorporate a dead-zone nonlinearity in the weight updating rule, adapted from Kreisselmeier and Anderson [14]. The control algorithm, the modified updating rule, and a convergence theorem are given in Section III.

The control scheme and the convergence result presented in this paper are applicable to nonlinear discrete-time systems with a general relative degree. For a system with a relative degree higher than one, the cancellation control cannot be defined explicitly because of a causality problem, i.e., current controls depend on future outputs. A similar problem appears in the linear case and has been discussed in [15] and [16]. In Section II, we generalize the results of [15] and [16] to the nonlinear case. We also define the zero dynamics and the minimum phase property of a nonlinear system. Minimum phase is one of the conditions of our convergence theorem. Finally, simulation results are given in Section IV.

II. Linearizing Feedback Control

We are interested in the single-input/single-output nonlinear discrete-time system

$$y_{k+1} = f_0(\cdot) + g_0(\cdot)u_{k-d+1} \tag{2}$$

where f_0 and g_0 are smooth (i.e., infinitely differentiable) functions of

$$y_{k-n+1}, \cdots, y_k, u_{k-d-m+1}, \cdots, u_{k-d}$$

$m \leq n$, y is the output, u is the input, d is the relative degree of the system, and g_0 is bounded away from zero. The arguments of f_0 and g_0 are real variables.

Manuscript received October 8, 1991; revised May 7, 1993. Recommended by Past Associate Editor, A. Arapostathis. This work was supported in part by National Science Foundation Grants ECS-8912827 and ECS-9121501 and National Science Council of the Republic of China Grant NSC-81-0404-E-009-005.

F.-C. Chen is with the Department of Control Engineering, National Chiao Tung University, Hsinchu, Taiwan R.O.C.

H. K. Khalil is with the Department of Electrical Engineering, Michigan State University, East Lansing, MI 48824 USA.

IEEE Log Number 9409416.

There are two difficulties when we try to linearize system (2) via feedback. First, the control law cannot simply be

$$u_{k-d+1} = -\frac{f_0(\cdot)}{g_0(\cdot)} + \frac{r(k-d+1)}{g_0(\cdot)}$$

with $r(k)$ being the reference command, because this control is noncausal when $d > 1$. Second, since f_0 and g_0 depend on past inputs, the system may become internally unstable after the feedback control, if it exists, cancels the plant dynamics. These two issues are well known for linear discrete-time systems [15], [16]. The purpose of this section is to resolve these difficulties for the nonlinear system (2).

To define the zero dynamics and to facilitate developing the convergence proof, the input/output form (2) of the system is converted into a state-space form. We select the state variables as the current output and all past inputs and outputs up to the most delayed input or output on the right-hand side of (2), i.e.,

$$\begin{aligned} x_i(k) &= y_{k-n+i}, \qquad \text{for} \quad i = 1, 2, \cdots, n \\ x_{n+i}(k) &= u_{k-m-d+i}, \qquad \text{for} \quad i = 1, 2, \cdots, m+d-1. \end{aligned}$$

Let $\mathbf{x}(k)$ be the state vector. A state-space model of (2) is constructed accordingly as

$$\begin{aligned} x_i(k+1) &= x_{i+1}(k), \qquad \text{for} \quad i = 1, 2, \cdots, n-1 \\ x_n(k+1) &= f_0[\mathbf{x}(k)] + g_0[\mathbf{x}(k)] x_{n+m+1}(k) \\ x_{n+i}(k+1) &= x_{n+i+1}(k), \\ &\qquad \text{for} \quad i = 1, 2, \cdots, m+d-2 \\ x_{n+m+d-1}(k+1) &= u_k \\ y(k) &= x_n(k). \end{aligned} \tag{3}$$

When $d = 1$, the second equation of (3) is

$$x_n(k+1) = f_0[\mathbf{x}(k)] + g_0[\mathbf{x}(k)] u_k.$$

Hence, the input–output map of the system can be linearized by a causal state feedback control that cancels the nonlinearity. When $d > 1$, we need to do more work to be able to cancel the nonlinearity by a causal state feedback control. The trick is to represent future plant outputs in terms of elements of $\mathbf{x}(k)$. Notice that $x_n(k+1) = y_{k+1}$. Then

$$x_n(k+2) = f_0[\mathbf{x}(k+1)] + g_0[\mathbf{x}(k+1)] x_{n+m+1}(k+1). \tag{4}$$

Replacing $\mathbf{x}(k+1)$ in (4) by the right-hand side of (3), we have[1]

$$x_n(k+2) = f_1[\mathbf{x}(k)] + g_1[\mathbf{x}(k)] x_{n+m+2}(k).$$

By applying the same technique recursively, one gets

$$\begin{aligned} x_n(k+3) &= f_2[\mathbf{x}(k)] + g_2[\mathbf{x}(k)] x_{n+m+3}(k), \\ &\vdots \\ x_n(k+d-1) &= f_{d-2}[\mathbf{x}(k)] + g_{d-2}[\mathbf{x}(k)] x_{n+m+d-1}(k). \end{aligned}$$

[1] Notice that f_0 and g_0 depend only on x_1 to x_{n+m}. Therefore, substitution of $\mathbf{x}(k+1)$ from (3) does not bring u_k in the argument of f_0 and g_0. The new functions f_1 and g_1 depend on x_1 to x_{n+m+1}.

Consider the state transformation

$$\mathbf{z}(k) = \begin{bmatrix} z_{11}(k) \\ \vdots \\ z_{1n}(k) \\ z_{1,n+1}(k) \\ \vdots \\ z_{1,n+d-1}(k) \\ z_{21}(k) \\ \vdots \\ z_{2m}(k) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ \vdots \\ x_n(k) \\ x_n(k+1) \\ \vdots \\ x_n(k+d-1) \\ x_{n+1}(k) \\ \vdots \\ x_{n+m}(k) \end{bmatrix} = T[\mathbf{x}(k)]. \tag{5}$$

It can be shown that the inverse of (5), i.e., $\mathbf{x} = T^{-1}(\mathbf{z})$, exists provided $g_0(\mathbf{x}), g_1(\mathbf{x}), \cdots, g_{d-2}(\mathbf{x})$ are bounded away from zero over the domain of interest. After application of the transformation (5), (3) becomes

$$\begin{aligned} z_{1i}(k+1) &= z_{1,i+1}(k), \\ &\qquad \text{for} \quad i = 1, 2, \cdots, n+d-2 \\ z_{1,n+d-1}(k+1) &= f_{d-1}[\mathbf{x}(k)] \\ &\quad + g_{d-1}[\mathbf{x}(k)] x_{n+m+d-1}(k+1) \\ &= f_{d-1}\{T^{-1}[\mathbf{z}(k)]\} \\ &\quad + g_{d-1}\{T^{-1}[\mathbf{z}(k)]\} u_k \\ &= F[\mathbf{z}(k)] + G[\mathbf{z}(k)] u_k \\ z_{2i}(k+1) &= z_{2,i+1}(k), \\ &\qquad \text{for} \quad i = 1, 2, \cdots, m-1 \\ z_{2m}(k+1) &= \frac{z_{1,n+1}(k) - f_0[\mathbf{x}(k)]}{g_0[\mathbf{x}(k)]} \\ &= \frac{z_{1,n+1}(k) - f_0\{T^{-1}[\mathbf{z}(k)]\}}{g_0\{T^{-1}[\mathbf{z}(k)]\}} \\ y(k) &= z_{1n}(k). \end{aligned} \tag{6}$$

When $d = 1$, the state vector z does not contain the components z_{1i} for $i > n$, and the state equation in the z coordinates takes the form (6) except that $z_{2m}(k+1) = u_k$. To deal with the cases $d = 1$ and $d > 1$ simultaneously we rewrite (6) as

$$\begin{aligned} z_{1i}(k+1) &= z_{1,i+1}(k), \\ &\qquad \text{for} \quad i = 1, 2, \cdots, n+d-2 \\ z_{1,n+d-1}(k+1) &= F[\mathbf{z}(k)] + G[\mathbf{z}(k)] u_k \\ z_{2i}(k+1) &= z_{2,i+1}(k), \\ &\qquad \text{for} \quad i = 1, 2, \cdots, m-1 \\ z_{2m}(k+1) &= u_{k-d+1} \\ y(k) &= z_{1n}(k) \end{aligned} \tag{7}$$

where $u_{k-d+1} = u_k$ when $d = 1$, while for $d > 1$ we have

$$u_{k-d+1} = \frac{z_{1,n+1}(k) - f_0\{T^{-1}[\mathbf{z}(k)]\}}{g_0\{T^{-1}[\mathbf{z}(k)]\}}. \tag{8}$$

The feedback control

$$u_k = \frac{-F[\mathbf{z}(k)] + r(k)}{G[\mathbf{z}(k)]} \tag{9}$$

linearizes the input–output map of the system and $r(k)$ appears as desired output d steps later. This transformation procedure is illustrated by the following example.

Example: For $n = m = d = 2$, the system takes the form

$$y_{k+1} = f_0(y_{k-1}, y_k, u_{k-3}, u_{k-2}) + g_0(y_{k-1}, y_k, u_{k-3}, u_{k-2})u_{k-1}. \quad (10)$$

Then

$$y_{k+2} = f_0(y_k, y_{k+1}, u_{k-2}, u_{k-1}) + g_0(y_k, y_{k+1}, u_{k-2}, u_{k-1})u_k. \quad (11)$$

Replace the y_{k+1} term in (11) by the right-hand side of (10) to obtain

$$\begin{aligned} y_{k+2} &= f_0[y_k, f_0(y_{k-1}, y_k, u_{k-3}, u_{k-2}) \\ &\quad + g_0(y_{k-1}, y_k, u_{k-3}, u_{k-2}) \\ &\quad \cdot u_{k-1}, u_{k-2}, u_{k-1}] \\ &\quad + g_0[y_k, f_0(y_{k-1}, y_k, u_{k-3}, u_{k-2}) \\ &\quad + g_0(y_{k-1}, y_k, u_{k-3}, u_{k-2}) \\ &\quad \cdot u_{k-1}, u_{k-2}, u_{k-1}]u_k \\ &= f_1(y_{k-1}, y_k, u_{k-3}, u_{k-2}, u_{k-1}) \\ &\quad + g_1(y_{k-1}, y_k, u_{k-3}, u_{k-2}, u_{k-1})u_k. \end{aligned}$$

The two functions f_1 and g_1 depend on past inputs and outputs; thus, the control u_k can be defined in terms of f_1 and g_1. □

Following Monaco and Norman-Cyrot [17], the zero dynamics are defined as the unobservable dynamics when the control (9) is used, namely

$$\begin{aligned} z_{1i}(k+1) &= z_{1,i+1}(k), \quad \text{for} \quad i = 1, 2, \cdots, n-2 \\ z_{1,n-1}(k+1) &= 0, \\ z_{2i}(k+1) &= z_{2,i+1}(k), \quad \text{for} \quad i = 1, 2, \cdots, m-1 \\ z_{2m}(k+1) &= \left. \frac{-f_0\{T^{-1}[\mathbf{z}(k)]\}}{g_0\{T^{-1}[\mathbf{z}(k)]\}} \right|_{z_{1i}=0,\ i=n,\cdots,n+d-1}. \end{aligned} \quad (12)$$

Since the dynamics associated with z_{1i}, for $i = 1, \cdots, n-1$, are always stable, we define the system to be minimum phase if

$$\begin{aligned} z_{2i}(k+1) &= z_{2,i+1}(k), \quad \text{for} \quad i = 1, 2, \cdots, m-1 \\ z_{2m}(k+1) &= \frac{-f_0\{T^{-1}[\mathbf{0}, \mathbf{z}_2(k)]\}}{g_0\{T^{-1}[\mathbf{0}, \mathbf{z}_2(k)]\}} \end{aligned} \quad (13)$$

has an asymptotically stable equilibrium point $C = [c, c, \cdots, c]'$. Equation (12) defines the internal dynamics of the system when the reference command $r(k)$ and the plant output y_k are constrained to be identically zero. Since z_{1i}, for $i = 1, 2, \cdots, n+d-1$ are either forward or backward shifts of y_k, constraining y_k to be identically zero implies that $z_{1i} = 0$, for $i = 1, 2, \cdots, n+d-1$ and the closed-loop equation reduces to (13). Note that (13) is the same equation that is obtained from (2) by constraining y_k to be identically zero.

III. Adaptive Control Using Neural Networks

Consider now the case when n, m, and d are known, but the nonlinear functions $F(\bullet)$ and $G(\bullet)$ in (7) are unknown. We use multilayer neural networks to model the unknown nonlinearities. It has been shown by Funahashi [10], Cybenko [11], Hornik *et al.* [12], and Hecht-Nielson [9], using different techniques, that multilayer neural networks can approximate any "well-behaved" nonlinear function to any desired accuracy. The statement of Funahashi's theorem [10] is quoted here.

Theorem: Let $\phi(x)$ be a nonconstant, bounded, and monotonically increasing continuous function. Let S be a compact subset of R^n and $f(x_1, \cdots, x_n)$ be a real valued continuous function on S. Then for any $\epsilon > 0$, there exists an integer N and real constants $c_i, \theta_i (i = 1, \cdots, N), w_{ij} (i = 1, \cdots, N, j = 1, \cdots, n)$ such that

$$\tilde{f}(x_1, \cdots, x_n) = \sum_{i=1}^{N} c_i \phi \left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i \right) \quad (14)$$

satisfies $\max_{x \in S} |f(x_1, \cdots, x_n) - \tilde{f}(x_1, \cdots, x_n)| < \epsilon$.

The network $\tilde{f}$ in (14) contains one nonlinear hidden layer. Similar results for neural networks with more than one hidden layer can be derived from the theorem above or be shown from scratch [10]. Funahashi's theorem, however, provides only an "existence" result. The network size N and the network parameters $c_i's, w_{ij}'s, \theta_i's$ are not determined by the theorem.

To proceed with our development, we state our assumptions on the plant.

Assumption 1: $g_0(\mathbf{x}), \cdots, g_{d-1}(\mathbf{x})$ are bounded away from zero over S, a compact subset of $R^{n+m+d-1}$, that is

$$|g_i(\mathbf{x})| \geq b > 0, \quad \forall \mathbf{x} \in S. \quad (15)$$

Assumption 2 (The Minimum Phase Assumption): The change of variables $e_{2i} = z_{2i} - c$ transforms (13) into

$$\begin{aligned} e_{2i}(k+1) &= e_{2,i+1}(k), \quad \text{for} \quad i = 1, 2, \cdots, m-1 \\ e_{2m}(k+1) &= \frac{-f_0\{T^{-1}[0, \mathbf{e}_2(k) + C]\}}{g_0\{T^{-1}[0, \mathbf{e}_2(k) + C]\}} - c. \end{aligned} \quad (16)$$

We assume that the origin of (16) is exponentially stable, and there is a Lyapunov function $V_2(\mathbf{e}_2)$ which satisfies

$$\begin{aligned} c_1 |\mathbf{e}_2(k)|^2 \leq V_2[\mathbf{e}_2(k)] \leq c_2 |\mathbf{e}_2(k)|^2, \\ V_2[\mathbf{e}_2(k+1)] - V_2(\mathbf{e}_2(k)) \leq -\alpha |\mathbf{e}_2(k)|^2 \end{aligned}$$

and

$$\left| \frac{\partial V_2(\mathbf{e}_2)}{\partial \mathbf{e}_2} \right| \leq L |\mathbf{e}_2| \quad (17)$$

in some ball $B_{\rho_2} \subset R^m$. The existence of a Lyapunov function satisfying these condition is guaranteed by a converse Lyapunov theorem [13].

Rewrite the plant in an input–output form as

$$y_{k+d} = f_{d-1}[\mathbf{x}(k)] + g_{d-1}[\mathbf{x}(k)]u_k. \quad (18)$$

Recall from (6) that $f_{d-1}(\mathbf{x}) = F(\mathbf{z})$ and $g_{d-1}(\mathbf{x}) = G(\mathbf{z})$. Plant (18) is modeled by the neural network

$$\hat{y}_{k+d} = \hat{f}_{d-1}[\mathbf{x}(k), \mathbf{w}] + \hat{g}_{d-1}[\mathbf{x}(k), \mathbf{v}]u_k. \quad (19)$$

The functions $\hat{f}_{d-1}(\cdot, \cdot)$ and $\hat{g}_{d-1}(\cdot, \cdot)$ depend on the structure of the neural network and the number of neurons. For example, if $\hat{f}_{d-1}(\cdot, \cdot)$ and $\hat{g}_{d-1}(\cdot, \cdot)$ are three-layer neural networks with p and q hidden neurons, respectively, then they can be expressed as

$$\hat{f}_{d-1}[\mathbf{x}(k), \mathbf{w}] = \sum_{i=1}^{p} w_i H \left(\sum_{j=1}^{m+n+d-1} w_{ij} x_j + \hat{w}_i \right) \quad (20)$$

and

$$\hat{y}_{d-1}[\mathbf{x}(k),\mathbf{v}] = \sum_{i=1}^{q} v_i H\left(\sum_{j=1}^{m+n+d-1} v_{ij}x_j + \hat{v}_i\right). \quad (21)$$

According to Funahashi's theorem, the function H in (20) and (21) has to be continuous, bounded, nonconstant, and monotonically increasing. We require, in addition, that H be differentiable. The differentiability of H is needed in our updating rule and convergence analysis. Throughout this research, we have used the hyperbolic tangent function

$$H(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

in computer simulations, but other functions having the foregoing properties could be used.

Assumption 3: Given a positive constant ϵ and a compact set $S \subset R^{n+m+d-1}$, there exist coefficients $\mathbf{w}$ and $\mathbf{v}$ such that $\hat{f}_{d-1}$ and $\hat{g}_{d-1}$ approximate the continuous functions f_{d-1} and g_{d-1}, with accuracy ϵ over S, that is

$$\exists\, \mathbf{w},\mathbf{v}\ s.t.\ \max |\hat{f}_{d-1}(\mathbf{x},\mathbf{w}) - f_{d-1}(\mathbf{x})| \le \epsilon$$

and

$$\max |\hat{g}_{d-1}(\mathbf{x},\mathbf{v}) - g_{d-1}(\mathbf{x})| \le \epsilon, \quad \forall \mathbf{x} \in S. \quad (22)$$

This assumption is justified by the approximation results of [9]–[12]. In our work we assume that the structure of the network and the number of neurons have been already specified, and (22) holds for the plant under consideration, but we do not assume that we know the weights $\mathbf{w}$ and $\mathbf{v}$ for which (22) is satisfied. Let $\mathbf{w}(k)$ and $\mathbf{v}(k)$ denote the estimates of $\mathbf{w}$ and $\mathbf{v}$ at time k. Then the control u_k can be defined as the following.

Control Law:

$$u_k = \frac{-\hat{f}_{d-1}[\mathbf{x}(k),\mathbf{w}(k)] + r(k)}{\hat{g}_{d-1}[\mathbf{x}(k),\mathbf{v}(k)]} \quad (23)$$

where $r(k)$ is the reference command.

The control u_k is applied to both the plant and the neural network model. The network weights are updated according to the error between the plant and model outputs. To better define the error, rewrite (18) and (19) as

$$y_{k+1} = f_{d-1}[\mathbf{x}(k-d+1)] + g_{d-1}[\mathbf{x}(k-d+1)]u_{k-d+1} \quad (24)$$

and

$$\hat{y}_{k+1} = \hat{f}_{d-1}[\mathbf{x}(k-d+1),\mathbf{w}] + \hat{g}_{d-1}[\mathbf{x}(k-d+1),\mathbf{v}]u_{k-d+1}. \quad (25)$$

The estimated plant output is

$$y^*_{k+1} = \hat{f}_{d-1}[\mathbf{x}(k-d+1),\mathbf{w}(k)] + \hat{g}_{d-1}[\mathbf{x}(k-d+1),\mathbf{v}(k)]u_{k-d+1}. \quad (26)$$

The error e^*_{k+1} is defined as

$$e^*_{k+1} = y^*_{k+1} - y_{k+1} \quad (27)$$

which will be used in the weight updating rule to be described next.

Let $\Theta = \begin{bmatrix}\mathbf{w}\\ \mathbf{v}\end{bmatrix}$. The problem of adjusting the estimate $\Theta(k)$ on-line is a typical problem in adaptive control. The new element here is the fact that the functions $\hat{f}_{d-1}$ and $\hat{g}_{d-1}$ depend nonlinearly on the parameter Θ. In the adaptive control of linear systems, or even linearizable continuous-time systems [3], the corresponding functions depend linearly on the unknown parameter Θ. This nonlinear dependence on Θ is the main challenge in the current problem. In a previous work [13], we studied an updating rule of the form

$$\Theta(k+1) = \Theta(k) - \frac{1}{r_k} e^*_{k+1}\mathbf{J}_{k-d+1} \quad (28)$$

where

$$\mathbf{J}_{k-d+1} = \left[\left.\frac{\partial y^*_{k+1}}{\partial \Theta}\right|_{\Theta(k)}\right]' = \begin{bmatrix}\left(\left.\frac{\partial \hat{f}_{d-1}[\mathbf{x}(k-d+1),\mathbf{w}]}{\partial \mathbf{w}}\right|_{\mathbf{w}(k)}\right)'\\ \left(\left.\frac{\partial \hat{g}_{d-1}[\mathbf{x}(k-d+1),\mathbf{v}]}{\partial \mathbf{v}}\right|_{\mathbf{v}(k)}\right)' u_{k-d+1}\end{bmatrix}$$

and

$$r_k = 1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}.$$

The variable y^*_{k+1} is the output of a multilayer neural network. Hence, the Jacobian matrix $\mathbf{J}_{k-d+1}$ can be calculated using the routines of the backpropagation algorithm [5]. We were able to prove an asymptotic regulation result, under some restrictive assumptions, like assuming that the nonlinearity f_{d-1} vanishes at the origin and that f_{d-1} and g_{d-1} can be modeled perfectly by neural networks, that is, $\epsilon = 0$ in (22). As we tried to relax these restrictions and work on the tracking problem, it became clear that the learning rule (28) would be inadequate. The source of the problem is partly from model uncertainties in the stability analysis. Related problems have been extensively studied in the literature on robust adaptive control [18], where a number of modifications of the simple gradient algorithm have been proposed to cope with robustness problems. We are going to employ a dead-zone algorithm for updating the weights which has been adapted from [14]. At each time step, if the error between the plant output and the model output is larger than a certain threshold, the weights are updated. Otherwise, the weights are not changed. To implement this, the error e^*_{k+1} defined in (27) is applied as input to a dead-zone function $D(e)$, defined by

$$D(e) = \begin{cases} 0 & \text{if } |e| \le d_0\\ e - d_0 & \text{if } e > d_0\\ e + d_0 & \text{if } e < -d_0. \end{cases} \quad (29)$$

The output of the dead-zone function is used in the updating rule.

Updating Rule:

$$\Theta(k+1) = \Theta(k) - \frac{1}{r_k} D(e^*_{k+1})\mathbf{J}_{k-d+1}. \quad (30)$$

Define the parameter error as

$$\tilde{\Theta}(k) = \Theta(k) - \Theta. \quad (31)$$

Theorem 1: Suppose $|r(k)| \leq d_1$ for all $k \geq 0$. Given any constant $\rho > 0$ and any small constant $d_0 > 0$, there exist positive constants $\rho_1 = \rho_1(\rho, d_1), \rho_2 = \rho_2(\rho, d_1), \epsilon^* = \epsilon^*(\rho, d_0, d_1)$, and $\delta^* = \delta^*(\rho, d_0, d_1)$ such that if Assumptions 1 and 3 are satisfied on $S \supset B_{\rho_1}$ with $\epsilon < \epsilon^*$, Assumption 2 is satisfied on $B_{\rho_2}, |\mathbf{x}(0)| \leq \rho$, and $|\tilde{\Theta}(0)| \leq \delta < \delta^*$, then

1) $\left|\tilde{\Theta}(k)\right|$ will be monotonically nonincreasing, and $|\Theta(k+1) - \Theta(k)|$ will converge to zero.
2) The tracking error between the plant output and the reference command will converge to a ball of radius d_0 centered at the origin.

Proof:

Step 1: The dynamics associated with $\mathbf{z}_1$ [see (7)] are

$$\begin{aligned} z_{1i}(k+1) &= z_{1,i+1}(k), \qquad \text{for} \quad i = 1, 2, \cdots, n+d-2 \\ z_{1,n+d-1}(k) &= F[\mathbf{z}(k)] + G[\mathbf{z}(k)]u_k. \end{aligned} \tag{32}$$

The last equation can be rewritten as

$$\begin{aligned} z_{1,n+d-1}(k+1) &= F[\mathbf{z}(k)] + G[\mathbf{z}(k)]u_k \\ &= \hat{F}[\mathbf{z}(k), \mathbf{w}] + \hat{G}[\mathbf{z}(k), \mathbf{v}]u_k \\ &\quad + (F[\mathbf{z}(k)] - \hat{F}[\mathbf{z}(k), \mathbf{w}] \\ &\quad + \{G[\mathbf{z}(k)] - \hat{G}[\mathbf{z}(k), \mathbf{v}]\}u_k)_1 \\ &= \hat{F}[\mathbf{z}(k), \mathbf{w}] + \hat{G}[\mathbf{z}(k), \mathbf{v}]u_k + \{\bullet\}_1 \end{aligned} \tag{33}$$

where $\hat{F}[\mathbf{z}(k), \mathbf{w}] = \hat{f}_{d-1}\{T^{-1}[\mathbf{z}(k)], \mathbf{w}\}$ and $\hat{G}[\mathbf{z}(k), \mathbf{v}] = \hat{g}_{d-1}\{T^{-1}[\mathbf{z}(k)], \mathbf{v}\}$. Plugging u_k into (33), we have

$$\begin{aligned} z_{1,n+d-1}(k+1) &= \hat{F}[\mathbf{z}(k), \mathbf{w}] + \hat{G}[\mathbf{z}(k), \mathbf{v}] \\ &\quad \cdot \frac{-\hat{F}[\mathbf{z}(k), \mathbf{w}(k)] + r(k)}{\hat{G}[\mathbf{z}(k), \mathbf{v}(k)]} + \{\bullet\}_1 \\ &= \hat{F}[\mathbf{z}(k), \mathbf{w}] + \hat{G}[\mathbf{z}(k), \mathbf{v}] \\ &\quad \cdot \left[\frac{-\hat{F}[\mathbf{z}(k), \mathbf{w}(k)] + r(k)}{\hat{G}[\mathbf{z}(k), \mathbf{v}(k)]}\right. \\ &\quad + \left(\frac{-\hat{F}[\mathbf{z}(k), \mathbf{w}] + r(k)}{\hat{G}[\mathbf{z}(k), \mathbf{v}]}\right. \\ &\quad \left.\left. - \frac{-\hat{F}[\mathbf{z}(k), \mathbf{w}] + r(k)}{\hat{G}[\mathbf{z}(k), \mathbf{v}]}\right)\right] + \{\bullet\}_1 \\ &= r(k) + \left[\hat{F}[\mathbf{z}(k), \mathbf{w}(k)] - r(k)\right. \\ &\quad + \hat{G}[\mathbf{z}(k), \mathbf{v}] \\ &\quad \left. \cdot \left(\frac{-\hat{F}[\mathbf{z}(k), \mathbf{w}(k)] + r(k)}{\hat{G}[\mathbf{z}(k), \mathbf{v}(k)]}\right)\right] \\ &\quad + \hat{F}[\mathbf{z}(k), \mathbf{w}] - \hat{F}[\mathbf{z}(k), \mathbf{w}(k)] + \{\bullet\}_1 \\ &= r(k) + \{\bullet\}_1 \\ &\quad + (\hat{F}[\mathbf{z}(k), \mathbf{w}] - \hat{F}[\mathbf{z}(k), \mathbf{w}(k)] \\ &\quad + \{\hat{G}[\mathbf{z}(k), \mathbf{v}] - \hat{G}[\mathbf{z}(k), \mathbf{v}(k)]\}u_k)_2 \\ &= r(k) + \{\bullet\}_1 + \{\bullet\}_2. \end{aligned} \tag{34}$$

Define

$$e_{1i}(k) = z_{1i}(k) - r(k - n - d + i). \tag{35}$$

Then (32) can be represented in new state variables $\mathbf{e}_1$ as

$$\begin{aligned} e_{1i}(k+1) &= e_{1,i+1}(k), \qquad \text{for} \quad i = 1, 2, \cdots, n+d-2 \\ e_{1,n+d-1}(k) &= \{\bullet\}_1 + \{\bullet\}_2. \end{aligned} \tag{36}$$

With the transformation

$$e_{2i} = z_{2i} - c \tag{37}$$

the dynamics associated with $\mathbf{z}_2$ are transformed into

$$\begin{aligned} e_{2i}(k+1) &= e_{2,i+1}(k), \qquad \text{for} \quad i = 1, 2, \cdots, m-1 \\ e_{2m}(k+1) &= u_{k-d+1} - c. \end{aligned} \tag{38}$$

Thus, (36) and (38) together is the new state-space representation of the closed-loop system.

Let

$$\begin{aligned} \mathbf{e}_1(k) &= [e_{11}(k), e_{12}(k), \cdots, e_{1,n+d-1}(k)]', \\ \mathbf{e}_2(k) &= [e_{21}(k), e_{22}(k), \cdots, e_{2m}(k)]' \end{aligned}$$

and

$$\Pi(k) = [r(k-1), r(k-2), \cdots, r(k-n-d+1)]'.$$

Step 2: Consider the set

$$I_{\mathbf{e}} = \left\{ \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{pmatrix} \mid |\mathbf{e}_1| \leq \mu_1, |\mathbf{e}_2| \leq \mu_2 \right\} \tag{39}$$

where the positive constants μ_1 and μ_2 will be chosen as we go along. To start with, we choose them to ensure that, for all $|\mathbf{x}(0)| \leq \rho$, the initial vector $\mathbf{e}(0)$ will be in the interior of $I_{\mathbf{e}}$. Since $\mathbf{z}(k) = \mathbf{e}(k) + [\Pi(k) \quad C]'$ and $\mathbf{x}(k) = T^{-1}[\mathbf{z}(k)]$, it is clear that for all $\mathbf{e}$ in $I_{\mathbf{e}}$, the vector $\mathbf{x}$ belongs to a ball B_{ρ_1}, where ρ_1 depends on μ_1, μ_2, d_1 and $|C|$. We assume that Assumptions 1 and 3 hold on a compact set S containing B_{ρ_1}. Consider also the set

$$I_{\Theta} = \{\tilde{\Theta} \mid |\tilde{\Theta}| \leq \delta\}. \tag{40}$$

Our goal in this step is to show that as long as $\mathbf{e}(k)$ remains in $I_{\mathbf{e}}$, the set I_{Θ} will be a positively invariant set, provided ϵ and δ are sufficiently small. Toward that end, suppose that $\tilde{\Theta}(k)$ belongs to I_{Θ}. The input–output form of the system is

$$y_{k+1} = f_{d-1}[\mathbf{x}(k-d+1)] + g_{d-1}[\mathbf{x}(k-d+1)]u_{k-d+1}. \tag{41}$$

By (22), system (41) can be rewritten as

$$\begin{aligned} y_{k+1} &= \hat{f}_{d-1}[\mathbf{x}(k-d+1), \mathbf{w}] \\ &\quad + \hat{g}_{d-1}[\mathbf{x}(k-d+1), \mathbf{v}]u_{k-d+1} \\ &\quad + \{f_{d-1}[\mathbf{x}(k-d+1)] \\ &\quad - \hat{f}_{d-1}[\mathbf{x}(k-d+1), \mathbf{w}]\} \\ &\quad + (\{g_{d-1}[\mathbf{x}(k-d+1)] \\ &\quad - \hat{g}_{d-1}[\mathbf{x}(k-d+1), \mathbf{v}]\}u_{k-d+1}) \\ &= \hat{f}_{d-1}[\mathbf{x}(k-d+1), \mathbf{w}] \\ &\quad + \hat{g}_{d-1}[\mathbf{x}(k-d+1), \mathbf{v}]u_{k-d+1} + O(\epsilon). \end{aligned} \tag{42}$$

The estimate of the plant output is

$$y^*_{k+1} = \hat{f}_{d-1}[\mathbf{x}(k-d+1), \mathbf{w}(k)] + \hat{g}_{d-1}[\mathbf{x}(k-d+1), \mathbf{v}(k)]u_{k-d+1}. \quad (43)$$

The error between the neural network output and the plant output is

$$\begin{aligned} e^*_{k+1} &= y^*_{k+1} - y_{k+1} \\ &= \hat{f}_{d-1}[\mathbf{x}(k-d+1), \mathbf{w}(k)] - \hat{f}_{d-1}[\mathbf{x}(k-d+1), \mathbf{w}] \\ &\quad + \{\hat{g}_{d-1}[\mathbf{x}(k-d+1), \mathbf{v}(k)] \\ &\quad - \hat{g}_{d-1}[\mathbf{x}(k-d+1), \mathbf{v}]\}u_{k-d+1} + O(\epsilon) \\ &= \tilde{\Theta}(k)' \begin{bmatrix} \left(\dfrac{\partial \hat{f}_{d-1}[\mathbf{x}(k-d+1), \mathbf{w}(k)]}{\partial \mathbf{w}(k)}\right)' \\ \left(\dfrac{\partial \hat{g}_{d-1}[\mathbf{x}(k-d+1), \mathbf{v}(k)]}{\partial \mathbf{v}(k)}\right)' u_{k-d+1} \end{bmatrix} \\ &\quad + O(|\tilde{\Theta}(k)|^2) + O(\epsilon) \\ &= \tilde{\Theta}(k)'\mathbf{J}_{k-d+1} + \eta(k). \end{aligned}$$

Since $\mathbf{x}(k)$ is bounded, there exist c_3 and c_4 (depending on μ_1 and μ_2) such that

$$|\eta(k)| \le c_3|\tilde{\Theta}(k)|^2 + c_4\epsilon. \quad (44)$$

Assume that δ and ϵ are small enough such that

$$|\eta(k)| \le M < d_0 \quad (45)$$

(d_0 defined in (29)). Using the definition of the dead-zone function (29), we can easily verify the following claims:

- If $|e^*_{k+1}| \le d_0$, then $D(e^*_{k+1}) = 0$.
- If $e^*_{k+1} > d_0$, i.e., $\tilde{\Theta}(k)'\mathbf{J}_{k-d+1} + \eta(k) > d_0$, then $\tilde{\Theta}(k)'\mathbf{J}_{k-d+1} > 0$, since $|\eta(k)| < d_0$

$$\begin{aligned} D(e^*_{k+1}) &= \tilde{\Theta}(k)'\mathbf{J}_{k-d+1} + \eta(k) \\ &\quad - d_0 < \tilde{\Theta}(k)'\mathbf{J}_{k-d+1} + d_0 - d_0 \\ &\Longrightarrow D(e^*_{k+1}) < \tilde{\Theta}(k)'\mathbf{J}_{k-d+1}. \end{aligned}$$

- If $e^*_{k+1} < -d_0$, i.e., $\tilde{\Theta}(k)'\mathbf{J}_{k-d+1} + \eta(k) < -d_0$, then $\tilde{\Theta}(k)'\mathbf{J}_{k-d+1} < 0$, since $|\eta(k)| < d_0$.

$$\begin{aligned} D(e^*_{k+1}) &= \tilde{\Theta}(k)'\mathbf{J}_{k-d+1} + \eta(k) \\ &\quad + d_0 > \tilde{\Theta}(k)'\mathbf{J}_{k-d+1} - d_0 + d_0 \\ &\Longrightarrow D(e^*_{k+1}) > \tilde{\Theta}(k)'\mathbf{J}_{k-d+1}. \end{aligned}$$

Thus in all cases we can represent $D(e^*_{k+1})$ as

$$D(e^*_{k+1}) = \alpha(k)\tilde{\Theta}(k)'\mathbf{J}_{k-d+1} \quad (46)$$

where $0 \le \alpha(k) < 1$. Substituting (46) into the updating rule (30), we obtain

$$\Theta(k+1) = \Theta(k) - \alpha(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]\mathbf{J}_{k-d+1}}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}}. \quad (47)$$

Subtracting Θ from both sides of (47), it becomes

$$\tilde{\Theta}(k+1) = \tilde{\Theta}(k) - \alpha(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]\mathbf{J}_{k-d+1}}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}}. \quad (48)$$

Then

$$\begin{aligned} &\tilde{\Theta}(k+1)'\tilde{\Theta}(k+1) - \tilde{\Theta}(k)'\tilde{\Theta}(k) \\ &\quad = -2\alpha(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]^2}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}} \\ &\qquad + \alpha^2(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]^2\mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}}{(1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1})^2} \\ &\quad \le -2\alpha(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]^2}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}} \\ &\qquad + \alpha^2(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]^2}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}} \quad (49) \\ &\quad \le -\alpha^2(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]^2}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}} \\ &\qquad - [2\alpha(k) - 2\alpha^2(k)]\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]^2}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}} \\ &\quad \le -\alpha^2(k)\frac{[\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}]^2}{1 + \mathbf{J}'_{k-d+1}\mathbf{J}_{k-d+1}} \le 0 \quad (50) \end{aligned}$$

which shows that I_Θ is positively invariant.

Step 3: Rewrite the dynamics associated with $\mathbf{e}_1$ as

$$\mathbf{e}_1(k+1) = A\mathbf{e}_1(k) + B[\bullet]_1 \quad (51)$$

where

$$A = \begin{bmatrix} 0 & 1 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

$$[\bullet]_1 = \{\bullet\}_1 + \{\bullet\}_2.$$

For all $\mathbf{e} \in I_\mathbf{e}$ there exist constants c_5 and c_6 (depending on μ_1 and μ_2) such that

$$[\bullet]_1 \le c_5\delta + c_6\epsilon. \quad (52)$$

The matrix A is a stability matrix since all its eigenvalues are at the origin. Hence, given any symmetric $Q > 0, \exists$ a symmetric $P > 0$ such that $A'PA - P = -Q$. Consider the quadratic function $V_1(\mathbf{e}_1(k)) = \mathbf{e}'_1(k)P\mathbf{e}_1(k)$. Then

$$\begin{aligned} &V_1(\mathbf{e}_1(k+1)) - V_1(\mathbf{e}_1(k)) \\ &\quad = -\mathbf{e}'_1(k)Q\mathbf{e}_1(k) \\ &\qquad + 2\mathbf{e}'_1(k)A'PB[\bullet]_1 + [\bullet]_1^2 B'PB \\ &\quad \le -\lambda_{\min}(Q)|\mathbf{e}_1(k)|^2 \\ &\qquad + c_7(c_5\delta + c_6\epsilon)|\mathbf{e}_1(k)| + c_8(c_5\delta + c_6\epsilon)^2 \\ &\quad \le -\tfrac{1}{2}\lambda_{\min}(Q)|\mathbf{e}_1(k)|^2 \\ &\qquad + c_9(c_5\delta + c_6\epsilon)^2 \\ &\quad \le -\lambda V_1[\mathbf{e}_1(k)] + c_9(c_5\delta + c_6\epsilon)^2, \\ &\qquad \text{where} \quad \lambda = \frac{\lambda_{\min}(Q)}{2\lambda_{max}(P)} \\ &\quad \Longrightarrow \text{The R.H.S. will be negative} \\ &\qquad \text{whenever} \quad V_1[\mathbf{e}_1(k)] > \frac{c_9}{\lambda}(c_5\delta + c_6\epsilon)^2. \quad (53) \end{aligned}$$

The conclusion from (53) is that, given any $\mu_3 > 0$, if δ and ϵ are small enough such that

$$\frac{c_9}{\lambda}(c_5\delta + c_6\epsilon)^2 < \mu_3 \tag{54}$$

then $\{V_1(\mathbf{e}_1) \le \mu_3\}$ will be a positively invariant set. By choosing μ_3 large enough so that $|\mathbf{e}_1(0)| \le \sqrt{[\mu_3/\lambda_{\max}(P)]}$, we can be sure that $\mathbf{e}_1(0) \in \{V_1(\mathbf{e}_1) \le \mu_3\}$. Moreover, by choosing $\mu_1 \ge \sqrt{[\mu_3/\lambda_{\max}(P)]}$, we can make $\{V_1(\mathbf{e}_1) \le \mu_3\} \subset \{|\mathbf{e}_1| \le \mu_1\}$.

Step 4: The dynamics associated with $\mathbf{e}_2$ are

$$\begin{aligned} e_{2i}(k+1) &= e_{2,i+1}(k), \quad \text{for} \quad i = 1, 2, \cdots, m-1 \\ e_{2m}(k+1) &= u_{k-d+1} - c \end{aligned}$$

where

$$\begin{aligned} u_{k-d+1} = \{&-\hat{F}[\mathbf{e}_1(k-d+1) + \Pi(k-d+1), \\ &\mathbf{e}_2(k-d+1) + C, \mathbf{w}(k-d+1)] \\ &+ r(k-d+1)\}/\{\hat{G}[\mathbf{e}_1(k-d+1) \\ &+ \Pi(k-d+1), \mathbf{e}_2(k-d+1) \\ &+ C, \mathbf{v}(k-d+1)]\}. \end{aligned}$$

After some manipulation similar to previous steps, we can show that as long as $\mathbf{e}(k)$ belongs to I_e

$$\begin{aligned} u_{k-d+1} = &\frac{-f_0\{T^{-1}[0, \mathbf{e}_2(k) + C]\}}{g_0\{T^{-1}[0, \mathbf{e}_2(k) + C]\}} \\ &+ [O(d_1) + O(\mu_1) + O(\delta) + O(\epsilon)]_2. \end{aligned}$$

Let

$$\begin{aligned} s_k &= \left[e_{22}(k), \cdots, e_{2m}(k), \frac{-f_0\{T^{-1}[0, \mathbf{e}_2(k) + C]\}}{g_0\{T^{-1}[0, \mathbf{e}_2(k) + C]\}} - c\right]' \\ q_k &= [0, \cdots, 0, [\bullet]_2]' \end{aligned}$$

and rewrite the equation for e_2 as

$$e_2(k+1) = s_k + q_k.$$

Using the function $V_2(\mathbf{e}_2(k))$ described in (17), we obtain

$$\begin{aligned} V_2[\mathbf{e}_2(k+1)] - V_2[\mathbf{e}_2(k)] &= V_2(s_k + q_k) - V_2[\mathbf{e}_2(k)] \\ &= \{V_2(s_k) - V_2[\mathbf{e}_2(k)]\} \\ &\quad + V_2(s_k + q_k) - V_2(s_k) \\ &\le -\hat{k}|\mathbf{e}_2(k)|^2 + c_{10}|s_k| \\ &\quad \cdot |q_k| + c_{11}|q_k|^2 \\ &\le -\hat{k}|\mathbf{e}_2(k)|^2 \\ &\quad + c_{12}|\mathbf{e}_2(k)| + c_{13}. \end{aligned} \tag{55}$$

Thus, if μ_2 is chosen large enough, there will be a positively invariant set $\{V_2[\mathbf{e}_2(k)] \le \mu_4\} \subset \{|\mathbf{e}_2| \le \mu_2\}$. By choosing μ_4 large enough we can be sure that $\mathbf{e}_2(0) \in \{V_2(\mathbf{e}_2) \le \mu_4\}$.

Step 5: We combine the results of Steps 2, 3, and 4 to conclude that as long as $\mathbf{e}(k)$ remains in I_e, the sets $I_\Theta = \{\tilde{\Theta} \le \delta\}$, $\{V_1(\mathbf{e}_1) \le \mu_3\}$, and $\{V_2(\mathbf{e}_2) \le \mu_4\}$ are positively invariant sets for sufficiently small ϵ and δ. Since

$$\mathbf{e}(0) \in \{V_1(\mathbf{e}_1) \le \mu_3\} \times \{V_2(\mathbf{e}_2) \le \mu_4\} \subset I_e$$

we see that $\mathbf{e}(k)$ remains in I_e for all $k \ge 0$. Hence, our conclusions so far are indeed valid for all $k \ge 0$.

Step 6: Since (50) is valid for all $k \ge 0$, we conclude that $\tilde{\Theta}(k)'\tilde{\Theta}(k)$ is monotonically nonincreasing and

$$\tilde{\Theta}(k)'\tilde{\Theta}(k) \longrightarrow C_1 \quad \text{as} \quad k \longrightarrow \infty \tag{56}$$

where C_1 is a constant. Moreover, (56) implies that

$$\alpha(k)\frac{\tilde{\Theta}(k)'\mathbf{J}_{k-d+1}}{\sqrt{1 + \mathbf{J}_{k-d+1}'\mathbf{J}_{k-d+1}}} \longrightarrow 0 \quad \text{as} \quad k \longrightarrow \infty. \tag{57}$$

Using (57) in (47) shows that

$$\Theta(k+1) - \Theta(k) \longrightarrow 0 \quad \text{as} \quad k \longrightarrow \infty. \tag{58}$$

Another related point to be shown is that, since

$$\begin{aligned} |\hat{G}[\mathbf{z}(k), \mathbf{v}(k)] - G[\mathbf{z}(k)]| \le &|\hat{G}[\mathbf{z}(k), \mathbf{v}(k)] \\ &- \hat{G}_{d-1}[\mathbf{z}(k), \mathbf{v}]| \\ &+ |\hat{g}_{d-1}\{T^{-1}[\mathbf{z}(k)], \mathbf{v}\} \\ &- g_{d-1}\{T^{-1}[\mathbf{z}(k)]\}| \\ \le &\bar{c}|\tilde{\mathbf{v}}(k)| + \epsilon \le \bar{c}\delta + \epsilon \end{aligned} \tag{59}$$

the function $\hat{G}[\mathbf{z}(k), \mathbf{v}(k)]$ will be bounded away from zero and will have the same sign as $G[\mathbf{z}(k)], \forall k \ge 0$, provided δ and ϵ are small enough to satisfy $\bar{c}\delta + \epsilon \le b/2$. A direct implication of this result is that the uniform boundedness of $\mathbf{e}(k)$ will ensure uniform boundedness of u_k.

Step 7: Finally, we show that the plant output will eventually track the reference command with an error less than d_0.

Since $\mathbf{x}(k)$ and u_k are bounded for all k, it can be verified that $\mathbf{J}_{k-d+1}$ is bounded. Hence, (57) implies that

$$\begin{aligned} &\alpha(k)\tilde{\Theta}(k)\mathbf{J}_{k-d+1} \longrightarrow 0 \quad \text{as} \quad k \longrightarrow \infty \\ \Longrightarrow\ &D(e^*_{k+1}) \longrightarrow 0 \quad \text{as} \quad k \longrightarrow \infty \\ \Longrightarrow\ &|e^*_{k+1}| < d_0 \quad \text{as} \quad k \longrightarrow \infty \\ \Longrightarrow\ &|y^*_{k+1} - y_{k+1}| < d_0 \quad \text{as} \quad k \longrightarrow \infty. \end{aligned}$$

Recall that

$$\begin{aligned} y^*_{k+d} = &\hat{f}_{d-1}[\mathbf{x}(k), \mathbf{w}(k+d-1)] \\ &+ \hat{g}_{d-1}[\mathbf{x}(k), \mathbf{v}(k+d-1)]u_k \end{aligned}$$

while the control u_k is generated from

$$r(k) = \hat{f}_{d-1}[\mathbf{x}(k), \mathbf{w}(k)] + \hat{g}_{d-1}[\mathbf{x}(k), \mathbf{v}(k)]u_k.$$

Then

$$\begin{aligned} |y^*_{k+d} - r(k)| \le &|\hat{f}_{d-1}[\mathbf{x}(k), \mathbf{w}(k+d-1)] \\ &- \hat{f}_{d-1}[\mathbf{x}(k), \mathbf{w}(k)]| \\ &+ |\{\hat{g}_{d-1}[\mathbf{x}(k), \mathbf{v}(k+d-1)] \\ &- \hat{g}_{d-1}[\mathbf{x}(k), \mathbf{v}(k)]\}u_k| \\ \le &K|\Theta(k-d+1) - \Theta(k)| \longrightarrow 0, \\ &\text{as} \quad k \longrightarrow \infty. \\ \Longrightarrow\ &|r(k) - y_{k+d}| < d_0 \quad \text{as} \quad k \longrightarrow \infty. \ \square \end{aligned}$$

IV. Simulation

The simulation is divided into three parts. Part I shows how the dead-zone size d_0 is related to the modeling error ϵ and the initial parameter error $\tilde{\Theta}(0)$. Part II emphasizes that our result is nonlocal in the initial state of the plant. In Part III, the neural network is used to control a relative-degree-two system. The simulation programs are written in Microsoft C and run on an IBM PC compatible machine.

Part I: The result of Theorem 1 is local in the sense that if the initial parameter error $\tilde{\Theta}$ (0) and the modeling error ϵ are small enough (both depending on the size of d_0), then the tracking error will converge to a ball of radius d_0. Here we want to demonstrate through simulation that this local convergence theorem is not a conservative result.

The system is modeled by

$$y_{k+1}^* = \hat{f}[y_k, y_{k-1}, \mathbf{w}(k)] + \hat{g}_k u_k \tag{60}$$

where $\hat{f}$ is the output of a neural network and $\hat{g}$ is a scalar. Our goal is to control the plant

$$y_{k+1} = \frac{1.5 y_k y_{k-1}}{1 + y_k^2 + y_{k-1}^2} + 0.35 \sin(y_k + y_{k-1}) + 1.2 u_k \tag{61}$$

to track a reference command. The network $\hat{f}$ contains two nonlinear hidden layers, with four nonlinear neurons in each hidden layer. In practice, the modeling error ϵ is determined once the neural network is determined. To find out the modeling error, we have the neural network model (60) undergo an off-line training. During the training, the control u_k is selected randomly from $[-2.2, 2.2]$ and applied to the plant and the model. Then, based on the error between the plant output y_{k+1} and the model output y_{k+1}^*, the network parameters $\mathbf{w}(k)$ and $\hat{g}_k$ are updated to $\mathbf{w}(k+1)$ and $\hat{g}_{k+1}$ using the standard backpropagation algorithm [5]. The average output error (averaged over 1000 iterations) drops to 0.05 after 10 000 iterations. It is observed that training for more than 10 000 iterations would do little for reducing the error.

The simulation proceeds as follows: The neural network model is pretrained for 20 000 iterations, and the network parameters at the end of the training (denoted as $\Theta(20k)$) are perturbed as follows: all the elements in $\mathbf{w}(20k)$ are increased by the amount Δ, and $\hat{g}(20k)$ is decreased by the amount Δ. The perturbed network model is used to control the plant, which initially rests at -2.0, to track a sinusoidal reference command $r(k) = 1.5 \sin(\pi k/50)$, using the control law and the updating rule described in the previous section. The closed-loop control system is run for 6000 time steps and then checked for convergence of the tracking error. It is clear from the updating rule (30) that if $|e| \leq d_0$, then $\Theta(k+1) = \Theta(k)$. To check the convergence of the tracking error, the system is run for another 8000 time steps. If no network parameter changes are observed during this period, the tracking error is said to have converged to the dead zone. The network parameters are checked 14 digits below the decimal point under MATLAB.

The simulation result is described next.

- When $\Delta = 0$, i.e., when the pretrained network is not perturbed, the error would not converge to the dead zone unless d_0 is increased to 0.075. Obviously, $d_0 = 0.075$ is needed to compensate for the modeling error.
- Continuing with $d_0 = 0.075$ but gradually increasing Δ, we found that the tracking error converges if $\Delta \leq 0.088$. When Δ is larger than 0.088, d_0 has to increase such that the tracking error converges. The upper bounds imposed on Δ are

$$\begin{aligned} &\Delta \leq 0.088, && \text{when} \quad d_0 = 0.075 \\ &\Delta \leq 0.100, && \text{when} \quad d_0 = 0.090 \\ &\Delta \leq 0.136, && \text{when} \quad d_0 = 0.130 \\ &\Delta \leq 0.198, && \text{when} \quad d_0 = 0.200 \\ &\Delta \leq 0.249, && \text{when} \quad d_0 = 0.320. \end{aligned} \tag{62}$$

  This shows that convergence in this example is local, with the allowable size of Δ clearly depending on the size of the dead zone (i.e., d_0).
  Although the convergence is checked only 6000 time steps after the control is implemented, this does not affect the accuracy of (62). Once it was tested with $\Delta = 0.136$ and d_0 a little less than 0.130 (for example $d_0 = 0.129$), we found the error did not converge even when the control system is run up to 100 000 time steps.
- Equations (44) and (45) in the proof of Theorem 1 say that the following condition has to be satisfied

$$c_3 |\tilde{\Theta}(0)|^2 + c_4 \epsilon \leq d_0. \tag{63}$$

  For this example, we want to see (63) hold and to determine a possible value of c_3. Substituting the values of (62) into (63) and assuming the equality holds, one has

$$\begin{aligned} c_3(0.088)^2 + c_4\epsilon &= 0.075 \\ c_3(0.100)^2 + c_4\epsilon &= 0.090 \\ c_3(0.136)^2 + c_4\epsilon &= 0.130 \\ c_3(0.198)^2 + c_4\epsilon &= 0.200 \\ c_3(0.249)^2 + c_4\epsilon &= 0.320. \end{aligned} \tag{64}$$

  Computing c_3 from the four pairs of neighboring equations developed from (64), one obtains the four values for c_3

$$6.649, \quad 4.708, \quad 3.863, \quad 4.825.$$

  They are of the same order and are indeed close. This shows that there is a quadratic relationship between d_0 and the maximum allowable $\tilde{\Theta}(0)$.

Part II: In this part of the simulation, we want to emphasize that the result of Theorem 1 is nonlocal in the initial state of the plant. With $\Delta = 0.195$ and $d_0 = 0.21$, three initial conditions of the plant are tried: 3.0, 0.0, and -3.0. Fig. 1 shows the plant outputs for the first 100 time steps. For all of the three initial conditions, the tracking errors are observed to converge to the dead zone after 6000 time steps.

There are two remarks concerning the results of Part I and Part II.

Remark 1: Although a suitable dead-zone size is needed to guarantee the convergence of the tracking error to the dead zone, without using dead zone we may achieve better results in terms of how close the plant output is to the reference

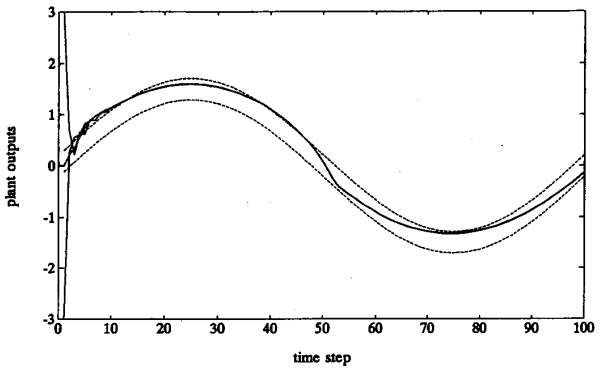


Fig. 1. The plant output for three different initial conditions.

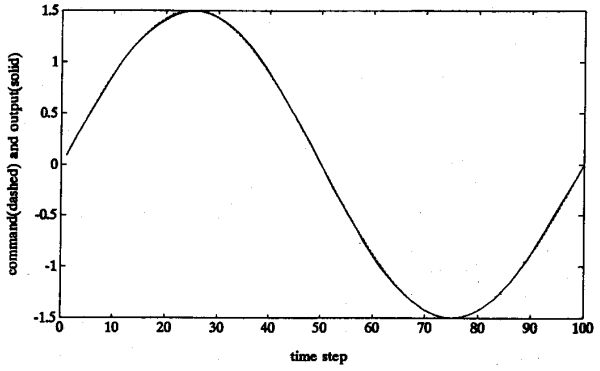


Fig. 2. The plant output when no dead zone is used.

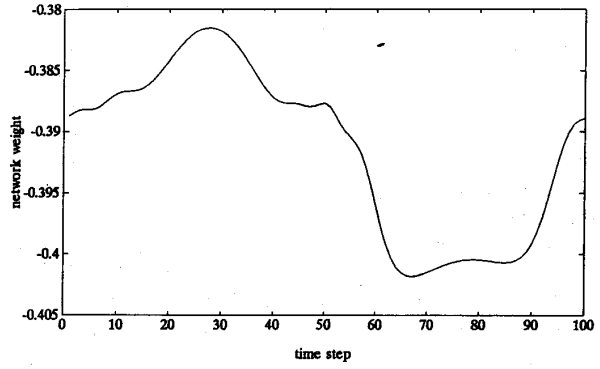


Fig. 3. The behavior of the weight w_{11} when no dead zone is used.

command. To illustrate this point, set $\Delta = 0.30$ and $d_0 = 0$ and rerun the simulation in Part I. Fig. 2 shows that, after 6000 time steps, only small error exists between the reference command and the plant output. The parameters, however, do not converge. Fig. 3 shows the behavior of the typical weight w_{11} in $\hat{f}$. Taking into consideration that in Part I the network weights are checked 14 digits below the decimal point for convergence, the weight fluctuation in Fig. 3 should be considered very large.

Remark 2: Although Part I and Part II pertinently address our main theoretical results in this paper, they do not illustrate the crucial role of dead-zone nonlinearity in accommodating

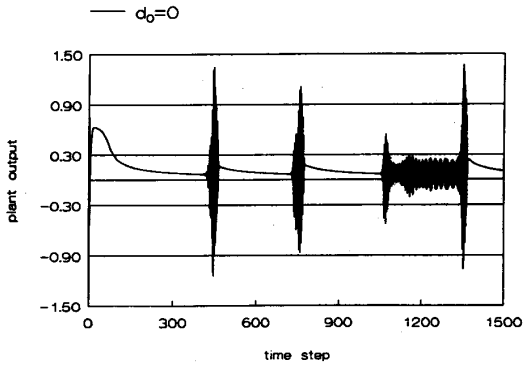


Fig. 4. The plant output for the regulation problem when no dead zone is used.

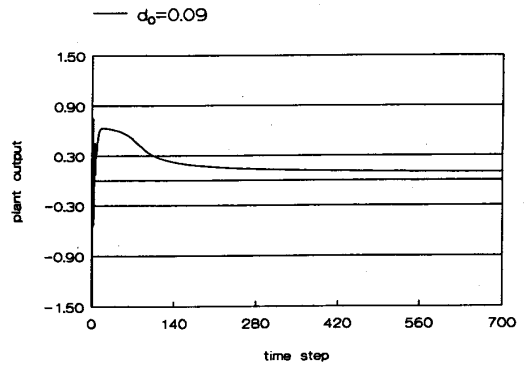


Fig. 5. The plant output for the regulation problem when $d_0 = 0.09$.

modeling errors. To emphasize this point, let us consider another example. Here neural network (60) is used to regulate the output of plant (65) to zero

$$y_{k+1} = \frac{2.5y_k y_{k-1}}{1 + y_k^2 + y_{k-1}^2} + 0.3\cos(y_k + y_{k-1}) + 1.2u_k. \quad (65)$$

We artificially construct the following situation: all the bias weights in the neural network are eliminated so that $\hat{f}$ becomes unable to model a nonlinear function which does not vanish at the origin. The plant contains a cos term; thus, it can not be properly modeled by the neural network around the origin. The neural network is pretrained for 1000 time steps. Fig. 4 shows the simulation result when no dead zone is used. The initial condition of the plant is $(y_0, y_{-1}) = (-1.5, -1.5)$. After some initial transient, the plant output is brought toward zero. The plant output, however, bursts into wild oscillations every time it comes close to zero. This is because the neural network controller can not provide correct cancellation control around zero plant output. To cope with this situation, a dead zone of size $d_0 > 0$ is specified in the updating rule. Fig. 5 shows the result when $d_0 = 0.09$. After some fast initial transient, the plant output is observed to gradually decay toward the origin and finally stay at 0.09. It is interesting to turn to Fig. 6 to look at the behavior of a randomly chosen weight in the neural network model. When $d_0 = 0.09$ is specified, the weight converges to a constant. Fig. 6 also shows that the same weight

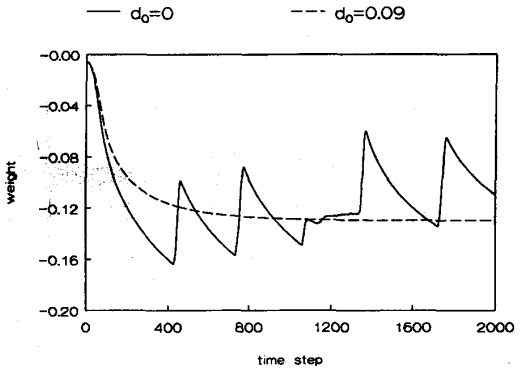


Fig. 6. The effect of dead zone on the behavior of a randomly chosen weight.

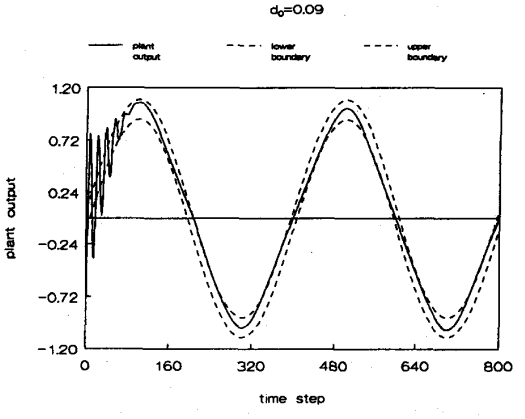


Fig. 7. The pendulum output.

moves around without settling to any point when $d_0 = 0$, i.e., when no dead zone is used.

Part III: In this part we are going to apply the neural network to control a pendulum, which is a relative-degree-two system. Suppose that the equation of motion of the pendulum is

$$\ddot{\theta}(t) + \dot{\theta}(t) + \sin(\theta(t)) = u(t). \tag{66}$$

Let T denote the sampling period. Equation (66) is discretized (using Euler's Rule) as

$$\theta_{k+1} = (2-T)\theta_k + (T-1)\theta_{k-1} - T^2 \sin(\theta_{k-1}) + T^2 u_{k-1}. \tag{67}$$

To define the control, the same transformation as described in Section II is performed

$$\begin{aligned}\theta_{k+2} &= (2-T)\theta_{k+1} + (T-1)\theta_k - T^2 sin(\theta_k) + T^2 u_k \\ &= (2-T)[(2-T)\theta_k + (T-1)\theta_{k-1} - T^2 sin(\theta_{k-1}) \\ &\quad + T^2 u_{k-1}] + (T-1)\theta_k - T^2 sin(\theta_k) + T^2 u_k\end{aligned}$$

which is modeled by

$$\hat{\theta}_{k+2} = \hat{f}[\theta_k, \theta_{k-1}, u_{k-1}, \mathbf{w}(k)] + \hat{g}_k u_k \tag{68}$$

where $\hat{f}$ is a neural network that contains two hidden layers with only two neurons in each hidden layer, while $\hat{g}$ is a scalar that is updated directly. At each time step, the control u_k is

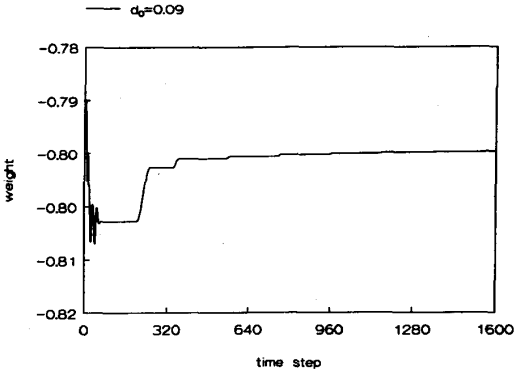


Fig. 8. After some initial transient, the weight converges.

generated from (68). The updating of the weights is based on the error between θ_{k+1} and

$$\hat{\theta}_{k+1} = \hat{f}[\theta_{k-1}, \theta_{k-2}, u_{k-2}, \mathbf{w}(k)] + \hat{g}_{k-1} u_{k-1} \tag{69}$$

as was outlined in Section III. Figs. 7 and 8 show the simulation results when $T = 0.3$ and a dead zone of size $d_0 = 0.09$ is used. The plant output is supposed to track a sinusoidal reference command of magnitude 1. Before the network is used for controlling the pendulum, it is trained for 12 000 time steps. The initial condition of the pendulum is $\theta_0 = -1$ radian. The plant output, shown in Fig. 7, converges toward the dead zone quickly after the control process starts. The behavior of a randomly chosen weight in the neural network is shown in Fig. 8. The weight converges after some initial quick changes.

V. Conclusions

The use of neural networks in control has drawn much interest in the control community in the past few years. In this work we presented an analytical study of the use of multilayer neural networks in the control of a class of nonlinear discrete-time systems with relative degree possibly higher than one. The convergence result of this paper is local with respect to the initial parameters but nonlocal with respect to the initial states of the plant. This feature of the result distinguishes it from a simple local result that could have been obtained by Taylor linearization about an equilibrium point and a set of nominal parameters. The fact that the initial states of the plant are allowed to belong to any compact set required a careful Lyapunov-type analysis of the closed-loop system. On the other hand, having a local result with respect to the initial parameters is not the best that one would have hoped for. In the lack of analytical results in the area, however, the result we obtained here is definitely a welcome conceptual contribution. The result, actually, is a little bit more than a conceptual one. The need to start from initial parameters sufficiently close to the exact ones can be addressed by pretraining. In all our simulations, the neural network used to model an unknown nonlinearity has gone through intensive off-line training using the backpropagation algorithm. This off-line training provided a good starting point when the network was used in on-line adaptive control. The idea of performing off-line training definitely has some merits, especially when such training is

performed on a prototype sample of the system. When the network is then used in an on-line adaptive control system, it may not provide an acceptable model of the actual nonlinearity of the system, but it could provide a model that is good enough for the initial parameters to be within their domain of attraction.

Our analytical study has also pointed out the need to incorporate a dead-zone nonlinearity to account for modeling errors between the actual nonlinearity and its neural network model. We have demonstrated via simulations the crucial role played by the dead-zone nonlinearity.

The convergence result we obtained in Theorem 1 is stated in a generic form that could be applied to many nonlinear parameterization schemes, provided the stated assumptions are satisfied. In particular, any function approximator that would satisfy Assumption 3 with arbitrarily small ϵ would be a scheme to which we could apply Theorem 1. The link between our work and multilayer neural networks comes at three points. First, we used the well-known results on the use of the multilayer neural network as a universal function approximator to justify Assumption 3. Second, our updating rule requires the calculation of a Jacobian matrix which can be calculated using the routines of the backpropagation algorithm. This is particularly important because those routines have the advantage that when the number of neuron layers is fixed (in practice, less than four layers are used), the computation time of the Jacobian is independent of the network complexity (that is, the number of neurons used in each layer which increases with the complexity of the approximated function), provided appropriate parallel computing hardware is available. Third, all our simulations have been for the case of multilayer neural networks.

The fact that Theorem 1 is stated in a generic form that is not limited to multilayer neural networks is a good feature and a bad feature at the same time. It is good because our analysis could be useful in other situations. But, it is bad because the analysis does not take advantage of properties that might be unique to multilayer neural networks. It is not clear at this time what could be such properties or how could they be used to obtain sharper results.

REFERENCES

[1] D. G. Taylor, P. V. Kokotovic, R. Marino, and I. Kanellakopoulos, "Adaptive regulation of nonlinear systems with unmodeled dynamics," *IEEE Trans. Automat. Contr.*, vol. 34, no. 4, pp. 405–412, Apr. 1989.

[2] R. M. Sanner and J.-J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 837–863, Nov. 1992.

[3] S. S. Sastry and A. Isidori, "Adaptive control of linearizable systems," *IEEE Trans. Automat. Contr.*, vol. 34, no. 11, pp. 1123–1131, Nov. 1989.

[4] I. Kanellakopoulos, P. V. Kokotovic, and A. S. Morse, "Adaptive output feedback control of a systems with output nonlinearities," *IEEE Trans. Automat. Contr.*, vol. 37, no. 11, pp. 1666–1682, Nov. 1992.

[5] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, vol. 1, Rumelhart and McClelland, Ed. Cambridge, MA: MIT Press, 1986.

[6] *IEEE Contr. Syst. Mag.*, special issues on neural network control systems, 1988–1990.

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

[8] F.-C. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," *IEEE Contr. Syst. Mag., Special Issue on Neural Networks for Control Systems*, Apr. 1990.

[9] R. Hecht-Nielsen, "Theory of the back-propagation neural network," in *Proc. Int. Joint Conf. Neural Networks*, June 1989, pp. I-593–605.

[10] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[11] G. Cybenko, "Approximation by superpositions of a sigmoidal function," Dept. Elec. Computer Eng., Univ. Illinois at Urbana-Champaign, Tech. Rep. 856, 1988.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[13] F.-C. Chen and H. K. Khalil, "Adaptive control of nonlinear systems using neural networks," *Int. J. Contr.*, vol. 55, pp. 1299–1317, 1992.

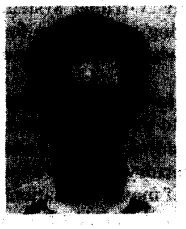
[14] G. Kreisselmeier and B. Anderson, "Robust model reference adaptive control," *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 2, pp. 127–133, Feb. 1986.

[15] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[16] K. J. Astrom and B. Wittenmark, *Adaptive Control*. Reading, MA: Addison-Wesley, 1989.

[17] S. Monaco and D. Normand-Cyrot, "Minimum-phase nonlinear discrete-time systems and feedback stabilization," in *Proc. 1987 IEEE Conf. Decis. Contr.*, 1987, pp. 979–986.

[18] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness*. Englewood Cliffs, NJ: Prentice-Hall, 1989.



Fu-Chuang Chen (S'88–M'90) was born in I-Lan, Taiwan, Republic of China on Dec. 30, 1960. He received the B.S. degree in power mechanical engineering from National Tsing Hua University, Taiwan, ROC, in 1983. After the two-year military service in Taiwan, he then attended the Michigan State University and received the M.S. degree in systems science and the Ph.D. degree in electrical engineering in 1986 and 1990, respectively.

Since 1990, Dr. Chen has been on the faculty of control engineering, National Chiao-Tung University, Taiwan, ROC. In the past few years, he has worked on problems related to singular perturbation theory, airplane auto pilot design, H_∞ theory, neural networks and adaptive control. His current research interests include the experimental phase of learning control problems involving neural networks and image feedback.

Hassan K. Khalil (S'77–M'78–SM'85–F'89) received the B.S. and the M.S. degrees from Cairo University, Cairo, Egypt, and the Ph.D. degree from the University of Illinois, Urbana-Champaign, in 1973, 1975, and 1978, respectively, all in electrical engineering.

Since 1978, he has been with Michigan State University, East Lansing, where he is currently Professor of Electrical Engineering. He has consulted for General Motors and Delco Products.

Dr. Khalil has published several papers on singular perturbation methods, decentralized control, robustness, and nonlinear control. He is author of the book *Nonlinear Systems* (New York: Macmillan, 1992), coauthor with P. Kokotović and J. O'Reilly, of the book *Singular Perturbation Methods in Control: Analysis and Design* (New York: Academic, 1986), and coeditor, with P. Kokotović, of the book *Singular Perturbation in Systems and Control* (New York: IEEE Press, 1986). He was the recipient of the 1983 Michigan State University Teacher Scholar Award, the 1989 George S. Axelby Outstanding Paper Award of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the 1994 Michigan State University Withrow Distinguished Scholar Award, and the 1995 Michigan State University Distinguished Faculty Award. He served as Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL, 1984–1985; Registration Chairman of the IEEE-CDC Conference, 1984; CSS Board of Governors, 1985; Program Committee member, IEEE-CDC Conference, 1986; Finance Chairman of the 1987 American Control Conference (ACC); Program Chairman of the 1988 ACC; Program Committee member, 1989 ACC; and General Chair of the 1994 ACC. He is now serving as Associate Editor of *Automatica*.