CEREBELLAR MODEL ARITHMETIC COMPUTERS

The nonlinearities in the dynamics of practical physical systems make their control a complex problem. Traditionally the plant dynamics were first modeled and verified through off-line experimentation. The control was then designed using linear system design techniques or geometric techniques with linear analogues. These techniques were successful in the control of systems when the model accurately described the process. The results for systems with unknown dynamics were at first limited by-and-large to ad hoc techniques and simulations involving assumptions such as certainty equivalence. These approaches are limited by the complexity of the model and cannot accommodate the variation of systems parameters. This has resulted in the development of controllers that can learn the process dynamics, as well as adapt to parametric changes in the system.

Adaptive controllers attempt to learn the plant characteristics while simultaneously achieving the control objectives. These controllers tune the adaptation parameters using the input-output measurements of the plant (1–4). While the classical adaptive methods guarantee stability for a large class of systems, the system must satisfy assumptions on linearity in the unknown system parameters. A regression matrix must be computed for each system by often tedious preliminary off-line analysis.

In recent years learning-based control has emerged as an alternative to adaptive control. Notable among this class of controllers are the neural network (NN) and fuzzy logic-based controllers. In the neural network, learning was accomplished in an off-line fashion by associating input-output pairs during training cycles. While neural networks are very successful in a variety of applications like pattern recognition, classification, and system identification to name a few, their applications in closed-loop is fundamentally different. In the literature neural networks have been utilized mostly in indirect control configuration, that is, identification-based control where back-propagation NN weight tuning is used to identify the system off-line (5–7). These methods are essentially open-loop control and do not guarantee stability in closed-loop applications. Unfortunately, when the neural network is employed in the feedback configuration, the gradients required for tuning cannot be found if the plant has unknown dynamics. Thus proofs of stability and guaranteed tracking performance are absent in these works (6–10).

Rigorous research in NN for closed-loop control is being pursued by several research groups. Narendra et al. (5,6,9) emphasizes the finding of gradients needed for backprop tuning. Sadegh (11) employs approximate calculations of the gradient to establish stability results, and Cristodoulou (12), Ioannou (13), Sadegh (11), and Slotine (14) offer rigorous proofs of performance in terms of tracking error stability and bounded NN weights. All these works assume that the NN is linear in the unknown parameters by employing single-layer NNs or recursive NNs with special structures. While the use of multilayer NNs in system identification was rigorously investigated, only recently researchers have focused on closed-loop control of nonlinear systems using multilayer NNs either in the continuous-time or discrete-time domains. In Refs. 15–17 it has been shown that NN controllers can effectively control complex nonlinear systems without requiring assumptions like linearity in parameters, availability of a regression matrix, and persistency of excitation. There are NNs are all multilayer nonlinear networks, and tuning laws guaranteeing tracking as well as stability of both the closed-loop system and the NN have been established, both for continuous-time and discrete-time cases.

The approximation property of fully connected NNs is basic to their application in control of complex dynamical systems. It has been shown that multilayer feed-forward NNs are theoretically capable of representing arbitrary mappings if a sufficiently large number of nodes are included in the hidden layers (15). Since all the weights are updated during each learning cycle, the learning is essentially global in nature. This global nature of the weight updating does not utilize the information on local NN structure and thus slows down the speed of learning. Furthermore, it is necessary to have a large number of nodes in each layer to guarantee a good function approximation. It has been shown that the speed of learning is inversely proportional to the number of nodes in a layer (15). The fully connected NNs suffer from an additional drawback in the sense that the function approximation is sensitive to the training data. Thus the effectiveness of a general multilayer NN is limited in problems requiring on-line learning.

To address these issues, the cerebellar model articulation controller (CMAC) NN (18) was proposed for closed-loop control of complex dynamical systems (19–25). The CMAC is a nonfully connected perceptronlike associative memory network that computes a nonlinear function over a domain of interest. The CMAC NN is capable of learning nonlinear functions extremely quickly due to the local nature of its weight updating (26).

The earliest contributions in the study of the behavior and properties of the CMACs were by H. Tolle and his group of researchers. Their finding on the approximation properties and learning in CMAC is well-presented in their classic book *NeuroControl* (27). Brown and Harris (28,29) also studied the use of the CMAC in adaptive modeling and control of systems. The importance of the convergence and stability properties of the CMAC in closed-loop control was established by Parks and Militzer (30,31). Ellison (32) independently presented similar results using CMACs for closed-loop control of robots. Recently, Commuri (33–39) established a method for passivity-based design of the learning laws for the CMAC that enables modular design for on-line learning and guaranteed closed-loop control. This article presents a comprehensive study of the use of CMAC NNs in control applications. The structure and properties of the CMAC NN that make it highly

suited for closed-loop control are studied. The weight-update laws for guaranteed stability, tracking performance, and robustness issues are discussed.

BACKGROUND ON CMAC NEURAL NETWORKS

Structure of CMAC Neural Networks

The cerebellar model arithmetic computer (CMAC) is a perceptronlike associative memory that performs nonlinear function mapping over a region of the function space. This highly structured nonfully connected neural network model was established by J. Albus (22,26) based on a model of the human memory and neuromuscular control system. Figure 1 shows a typical application of a CMAC neural network where the CMAC is used to manufacture a continuous function $g(\underline{x}) = [g_1(\underline{x}), g_2(\underline{x}), \ldots, g_m(\underline{x})]^T$, where $\underline{x} \in R^n$, and $g: R^n \rightarrow R^m$.

The nonlinear function $g(\underline{x})$ produced by the CMAC is composed of two primary functions

$$\begin{aligned} R &: X \Rightarrow A \\ P &: A \Rightarrow Y \end{aligned} \tag{1}$$

where X is the continuous n-dimensional input space, A is an N_A-dimensional *association space*, and Y is the m-dimensional output space. The function $R(.)$ is fixed and maps each point $\underline{x}$ in the input space X onto a association vector $a = R(\underline{x})$ in the association space A. The function $P(a)$ computes an output $y \in Y$ by projecting the association vector determined by $R(\underline{x})$ onto a vector of adjustable weights w such that

$$y = P(\alpha) = w^T \alpha \tag{2}$$

$R(\underline{x})$ in Eq. (1) is the multidimensional *receptive field function* which assigns an activation value to each point $\underline{x}$ in the input space $X\{\underline{x} = (x_1, \ldots, x_n) \in X\}$. From Eq. (2), it can be seen that the output of the CMAC is a linear combination of the weights (18).

In order to obtain the multidimensional receptive field functions, the input space is first discretized, and activation functions of finite span are defined on each of the intervals. A receptive field function is said to be active if it has a nonzero activation value for a particular input. Standard CMAC implementations have a finite number of maximally active receptive field functions for any given input vector. Figure 2 depicts some standard receptive functions, and Fig. 3 shows a multidimensional receptive field function of order 2 with an overlap of 4. The width of the receptive field function controls the output generalization of the CMAC and the offset between adjacent receptive field functions controls the input quantization and the output resolution (18). Further the function generated by the CMAC depends on the type of receptive fields used. Splines of order one generate staircase functions, while splines of order two generate linear output functions.

The CMAC is a nonfully connected perceptronlike network that computes a nonlinear function over a domain of interest. Since the receptive field functions have a finite span, an element in the input space excites only a finite number of these receptive field functions. Let $\underline{x}$ be the input vector presented to the network and α be the corresponding vector in the association space A. Let α^* be the set of active or nonzero elements of α. Since the output is a linear combination of these nonzero values, it is then necessary only to adjust the weights w attached to α^* in Eq. (2) to change the output. Thus the CMAC NN is capable of learning nonlinear functions extremely quickly due to this local nature of its weight updating (26).

Generalization versus Dichotomy. Since the network need not have an unique set of association cells for every possible input pattern, a given association cell can be activated by different input patterns. For example, let two inputs x_1 and x_2 activate two overlapping sets of association vectors α_1^* and α_2^*. Now adjustment in the weights corresponds to α_1^* will have the unintended consequence of influencing the output due to α_2^*, which can either be beneficial or detrimental to the implementation. In general, the networks ability to generalize between similar input patterns in determined by the overlap of $\alpha_1^* \wedge \alpha_2^*$. If $\alpha_1^* \wedge \alpha_2^*$ is null, then the two input patterns will be independent. The amount by which the outputs will be

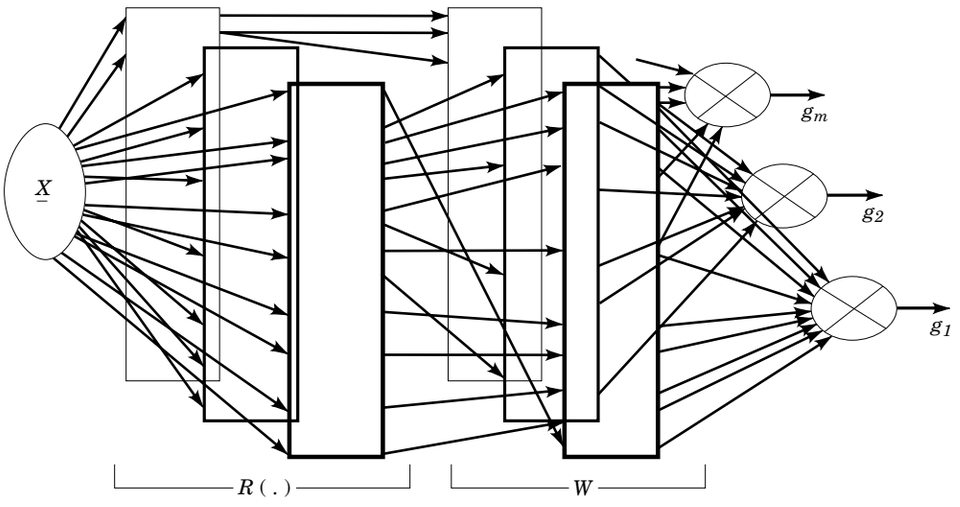

Figure 1. CMAC architecture for the approximation of a vector function.

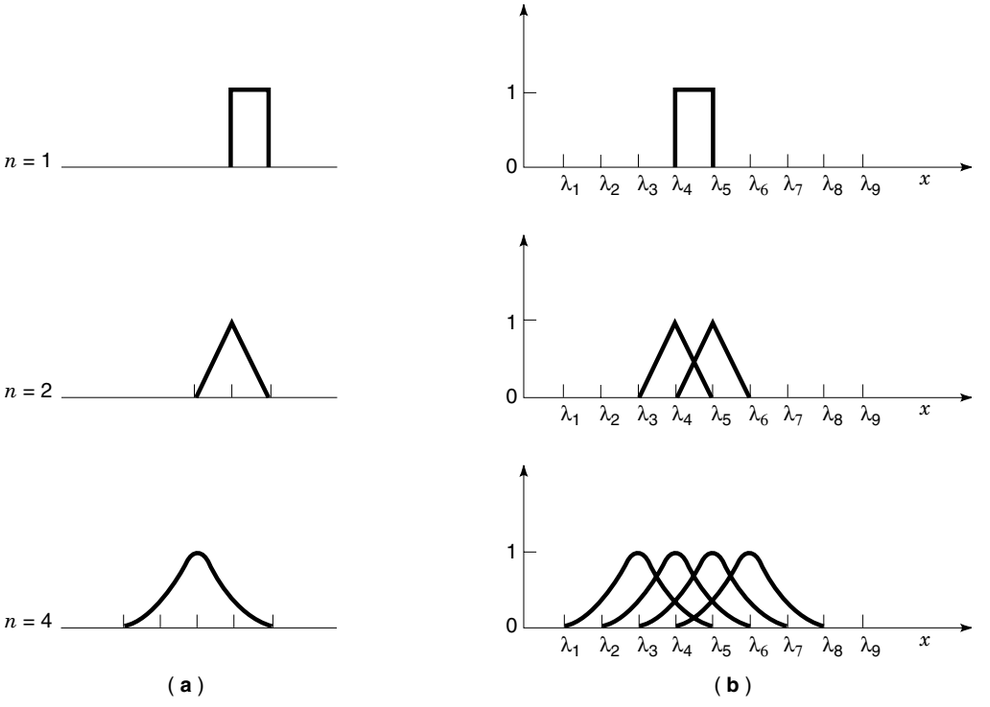

Figure 2. Standard CMAC receptive field functions of orders 1, 2, and 4.

similar to two input patterns x_1 and x_2 will be determined by extent of overlap of $\alpha_1^* $ and α_2^* (22). Similarly the network's ability to dichotomize or produce dissimilar outputs for the two inputs patterns x_1 and x_2 depends on the nonintersecting elements of $\alpha_1^* \wedge \alpha_2^*$.

Effects of Hash Coding. It can be seen from the above discussion that CMAC can learn any function by proper choice of the weights. The mapping generated, however, is dependent on the actual implementation of the CMAC.

Let A_p be the number of association cells physically implemented by CMAC and A^* be the number of maximally active elements of A for any given pattern. In practice, A_p is chosen to be at least 100 times A^*. Then it can be shown that a unique mapping from $X \rightarrow A$ is theoretically possible if $R^n < 99^{|A^*|}$, where R^n is the number of possible input patterns (22).

The number of association cells in any CMAC is determined by the level of discretization of the input space. If the level of discretization is very fine, there will be too many association cells and it becomes physically impossible to implement the CMAC. This problem can be solved by hash coding (22,40) where the size of physical memory is maintained at manageable size by mapping many association cells to the same physical memory locations. Hashing has the undesirable side effect of "collisions." If the actual number of memory locations available is two thousand, namely $A_p = 2000$ and $A^* = 20$, then the probability of two or more cells being mapped into the same cell in A is approximately 0.1 (22). Therefore, as long as this probability is low, collisions are not a serious problem and only results in reduced resolution of the output.

Another effect of hashing is the interference in the form of unwanted generalization between input vectors. It can be shown that this effect is insignificant as long as the overlap is not large compared to the total number of cells in A^*. If, for example, $A_p = 20{,}000$ and $A^* = 20$, then the probability of two or more collisions is 0.01, and the probability of two or more cells spuriously overlapping is 0.0002. Thus, in the implementation of CMAC, it is desirable to keep A^* small to minimize the amount of computations required. It is also desirable to keep the ratio A^*/A_p small to minimize the probability of overlap between widely separated input patterns.

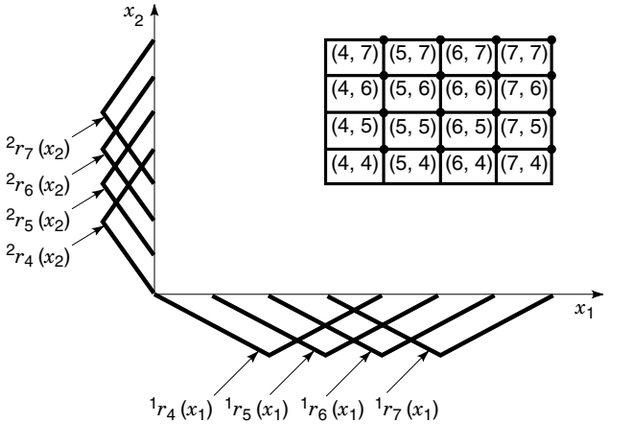

Figure 3. Multidimensional receptive field functions of order 2 and overlap 4.

Constructive Method for Linear Multidimensional Receptive Field Functions

The structure of the CMAC discussed in preceding section gives insight into the nature of the function generated by the

CMAC. However, in practical applications the reverse is often necessary, that is, when, given a particular function to approximate, the task is to determine the CMAC structure that will generate this required map. This problem was recently addressed in Ref. 33. In this work methods to construct CMACs that guarantee an approximation for a class of functions were established. In this subsection these results are summarized.

One-Dimensional Receptive Field Functions. Given $\underline{x} = [x_1, x_2, \ldots, x_n] \in R^n$, let $[x_{i_{\min}}, x_{i_{\max}}] \in R \; \forall \; 1 \le i \le n$ be the domain of interest. For this domain, select integers N_i and strictly increasing partitions $\pi_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,N_i}\}$, $\forall 1 \le i \le n$ (e.g., $x_{i_{\min}} = x_{i,1} < x_{i,2} < \ldots < x_{i,N_i} = x_{i_{\max}}$). For each component of the input space, define the receptive field functions as

$$\begin{aligned} \mu_{i,1}(x_i) &= \Lambda(-\infty, x_{i,1}, x_{i,2})(x_i) \\ \mu_{i,j}(x_i) &= \Lambda(x_{i,j-1}, x_{i,j}, x_{i,j+1})(x_i), \qquad 1 < j < N_i \\ \mu_{i,N_i}(x_i) &= \Lambda(x_{i,N_i-1}, x_{i,N_i}, \infty)(x_i) \end{aligned} \tag{3}$$

where the triangular functions $\Lambda(.)$ are defined as

$$\Lambda(a, b, c)(y) = \begin{cases} \dfrac{y-a}{b-a}, & a \le y \le b (= 1 \text{ if } a = -\infty) \\ \dfrac{c-y}{c-b}, & b \le y \le c (= 1 \text{ if } c = \infty) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The leftmost and rightmost receptive field functions are selected such that every value of x_i corresponds to at least one receptive field function. Given the partition $\pi_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,N_i}\}$, the one-dimensional receptive field functions selected as in Eqs. (3) and (4) are shown in Fig. 4.

Multidimensional Receptive Field Functions. Given any $\underline{x} = [x_1, \ldots, x_n] \in R^n$, define multidimensional receptive field functions as

$$R_{j_1, j_2, \ldots, j_n}(x) = \frac{\mu_{1,j_1}(x_1) \cdot \mu_{2,j_2}(x_2) \ldots \mu_{n,j_n}(x_n)}{\sum_{j_n=1}^{N_n} \cdots \sum_{j_2=1}^{N_2} \sum_{j_1=1}^{N_1} \prod_{i=1}^{n} \mu_{i,j_i}(x_i)} \tag{5}$$

It is easy to see that the receptive fields so defined are normalized *n-dimensional second-order splines.*

Lemma 1 The multidimensional receptive field functions selected in Eq. (5) satisfy three significant properties:

a. Positivity, $R_{j_1,j_2,\ldots,j_n}(\underline{x}) > 0$ for all $\underline{x} \in (x_{1,j_1-1}, x_{1,j_1+1}) \times \ldots \times (x_{n,j_n-1}, x_{n,j_n+1})$

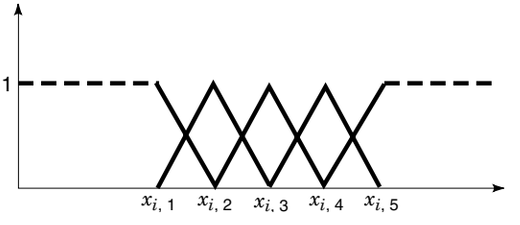

Figure 4. One-dimensional receptive field function: $N_i = 5$ spanning R^1.

b. Compact support, $R_{j_1,j_2,\ldots,j_n}(\underline{x}) = 0$ for all $\underline{x} \notin (x_{1,j_1-1}, x_{1,j_1+1}) \times \ldots \times (x_{n,j_n-1}, x_{n,j_n+1})$

c. Normalization, $\sum_{j_n=1}^{N_n} \cdots \sum_{j_2=1}^{N_2} \sum_{j_1=1}^{N_1} R_{j_1,j_2,\ldots,j_n}(\underline{x}) = 1$ for all $\underline{x}$.

According to Lemma 1(b), for any prescribed value of $\underline{x} \in R^n$, only 2^n values of $R_{j_1,j_2,\ldots,j_n}(\underline{x})$ are nonzero.

Salient Properties of the Output of CMAC. Given any element $\underline{x}$ of the input space, the receptive field values $R_{j_1,j_2,\ldots,j_n}(\underline{x})$ are elements in the association space A. The output of the CMAC neural network is now computed by projecting this association vector onto a vector of adjustable weights w. Let $w_{(j_1,\ldots,j_n)}$ be the weight associated with the index $j_1, \ldots, j_n$. Then the function manufactured by a single-output CMAC can be expressed as

$$g(x) = \sum_{j_n=1}^{N_n} \cdots \sum_{j_1=1}^{N_1} w_{(j_1,\ldots,j_n)} R_{j_1,\ldots,j_n}(x) : R^n \to R \tag{6}$$

A general CMAC is easily constructed by using this framework as follows.

Lemma 2 A multi-input multi-output CMAC with output $g(\underline{x}) : R^n \to R^m$ is a nonlinear mapping defined as

$$g(x) = [g_1(x), g_2(x), \ldots, g_m(x)]^T \tag{7}$$

where

$$g_k(x) = \sum_{j_n=1}^{N_n} \cdots \sum_{j_1=1}^{N_1} w_{k,(j_1,\ldots,j_n)} R_{j_1,\ldots,j_n}(x) : R^n \to R \tag{8}$$

The function $g(\underline{x})$ in Eq. (7) is Lipschitz continuous.

In fact, according to the normalization property of Lemma 1(c), Eq. (8) is a *convex combination* of the weights w.

Function Approximation Properties of CMAC Neural Networks. In recent years neural networks have been used in the control of systems with unknown dynamics. In the early applications NNs were used as direct adaptive controllers, where the NN was used to identify the system off-line, and the controllers were developed using the identified model. In later applications on-line learning laws were developed, and the NNs were used as indirect adaptive controllers (see NEURAL NETWORKS FOR FEEDBACK CONTROL). In all of these approaches the approximation property of fully connected NNs is basic to their application in control of complex dynamical systems. However, the effectiveness of a general multilayer NN is limited in problems requiring on-line learning. Since in a CMAC only a finite number of receptive fields are active for any given input, an efficient controller for systems with unknown dynamics can be implemented using CMAC NNs.

In the early approaches learning in CMAC was first accomplished off-line. The CMAC was presented with training samples, and the corresponding weights were updated until the network could reconstruct the unknown function with reasonable accuracy over the domain of interest. In these works the CMAC weight update rules were similar to the least mean

squares (LMS) algorithm. This way they ensured convergence of CMAC learning to some local minima. The convergence properties of CMAC were also studied by Wong and Sideris (40). In this work the CMAC learning is essentially solving a linear system with methods similar to the Gauss-Seidel method. This results in a highly accurate learning algorithm that converges exponentially fast. Therein the following result was also established.

Theorem 3 (40) Given a set of training samples composed of input-output pairs from $R^n \rightarrow R^m$, CMAC always learns the training set with arbitrary accuracy if the input space is discretized such that no two training samples excite the same set of association cells.

Recently it has been shown that CMACs can be constructed to approximate nonlinear function's with arbitrary accuracy. Consider the partition π_i, $1 \leq i \leq n$, given earlier. Then the following theorem can be proved (33).

Theorem 4 The function estimate $g(\underline{x})$ defined in Eq. (6) uniformly approximates any C^1-continuous function $f(\underline{x})$: $R^n \rightarrow R^m$ on $\Omega \subset R^n$. Specifically, given any $\epsilon > 0$ and L, the Lipschitz constant of $f(.)$ on Ω, the maximum partition size δ can be chosen such that

$$\|f(x) - g(x)\| \leq \epsilon \tag{9}$$

where

$$\delta \leq \frac{\epsilon}{mL} \tag{10}$$

and

$$\delta = \max(\|x - y|) \quad \forall x, y \in [x_{1,j_1-1}, x_{1,j_1}) \times \cdots \times [x_{n,j_n-1}, x_{n,j_n}), \forall j_i \tag{11}$$

According to the theorem, an estimate to a given function $f(\underline{x})$ is given by $g(.) = [g_1, g_2, \ldots, g_m]^T$ with

$$g_k(x) = \sum_{j_n=1}^{N_n} \cdots \sum_{j_1=1}^{N_1} w_{k,(j_1,\ldots,j_n)} R_{j_1,\ldots,j_n}(x) \tag{12}$$

for some weights w. In fact, the weights can be shown to be the samples of the function components to be approximated at each of the knot points of the partition.

Implementation Properties of CMAC Neural Networks

The output in Eqs. (7) and (8) of the CMAC can be represented as a function from R^n to R^m and expressed in vector notation as

$$g(\underline{x}) = w^T \Gamma(\underline{x}) \tag{13}$$

where w is a matrix containing the set of weights, and Γ is a vector of the receptive field activation values. The definition of w and Γ is not unique, though $w^T\Gamma$ is equal to the right-hand side of Eqs. (7) and (8). In the implementation of CMAC neural networks, it is customary to employ the following submappings (18,22,26):

$$\begin{aligned} &R : X \Rightarrow M \\ &Q : M \Rightarrow I \\ &\Gamma : I \times M \Rightarrow A \end{aligned} \tag{14}$$

where $R(\underline{x})$ is the receptive field function described in Eq. (5), Q is a quantization function, M is a matrix of receptive field activation values, and I is an array of column vectors used to identify the locations of the active receptive fields along each input dimension.

Let the receptive field functions along each dimension be chosen to have an overlap of two. Then, in all, only 2^n receptive fields will be active for a given input $\underline{x}$. These active receptive fields can be located by constructing a set of active indices of α. Given the partition on the input space, for any $\underline{x} \in R^n$ there exists a unique *n-tuple* $(j_1, j_2, \ldots, j_n)$ such that $\underline{x} \in \Omega_{j_1,j_2,\ldots,j_n}$. Let $k_1, k_2, \ldots, k_n$ be positive integers such that $(x_{j_1}, x_{j_2}, \ldots, x_{j_n}) \in \pi_{k_1,k_2,\ldots,k_n}$. Given this index set $(k_1, k_2, \ldots, k_n)$, after selecting *left-hand odometer* ordering, the indicator function is constructed as

$$I = k_1 + (k_2 - 1)N_1 + (k_3 - 1)N_1N_2 + \cdots \tag{15}$$

By Lemma 1, the elements of a not addressed by I are equal to zero. Thus Q is a map from $N_1 \times N_2 \times \ldots \times N_n$ space composed of the tensor products of the receptive field functions to a $(N_1N_2 \ldots N_n) \times$ one-dimensional space I. The map Γ is now defined by I and M. Specifically the 2^n nonzero values of $R(\underline{x})$ are placed into the matrix $\Gamma(\underline{x})$ at the locations specified by $I(\underline{x})$. This *ordering* of the indices uniquely determines w and Γ in Eq. (13).

Corollary 1 Given any C^1-function $f(.)$, ideal weights w can be found such that

$$f(\underline{x}) = w^T \Gamma(\underline{x}) + \epsilon \tag{16}$$

where ϵ is the function estimation error and $\|\epsilon\| \leq \epsilon_N$, with ϵ_N a given bound.

BACKGROUND ON NONLINEAR DYNAMICAL SYSTEMS

The earliest use of CMACs in control applications was in the control of robot manipulators (22,26,40). In these application's, the CMAC was first trained to learn the inverse dynamics of the system to be controlled (41,42). The training law used in these applications is similar to the Widrow-Hoff training procedure for linear adaptive elements (43,44),

$$dw = \beta^*(V_o - f(s_o))$$

where

- dw is the weight vector adjustment.
- β is the learning gain between 0 and 1.
- V_o is the applied control command vector during the previous control cycle.

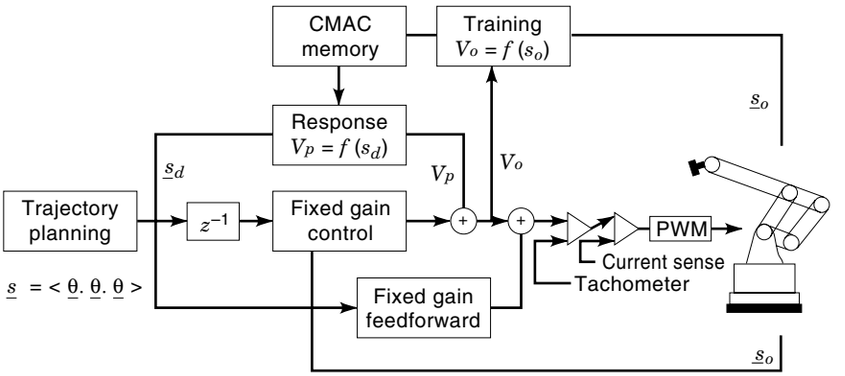

Figure 5. Block diagram of learning controller (32) for robot control. The output of the controller has two components: a fixed part and a variable part that depends on the response determined by the CMAC memory.

s_o is the observed state of the system in the previous control cycle.

$f(s_o)$ is the predicted drive value.

When the system is initialized, the weights contains all zeros. Therefore the output of the CMAC is zero. As the CMAC learns the inverse dynamics of the system, the CMAC network output will be similar to the actual control values required and the CMAC will take over from the fixed gain controller (see Fig. 5).

To illustrate the application of CMAC NNs in the control of nonlinear systems with unknown dynamics, three classes of systems from literature are presented. The systems represented by the dynamical equations in the following subsections are important from the standpoint of control, since most physical systems to be controlled can be expressed in the form of these equations. Here the dynamical representation is given followed by the CMAC formulation of the controller.

Discrete-Time Representation of a Nonlinear System in Brunowskii Canonical Form

The description of a nonlinear system in Brunowskii canonical form is given as

$$\begin{aligned}
&x_1(k+1) = x_2(k)\\
&x_2(k+1) = x_3(k)\\
&\cdot\\
&x_{n_1}(k+1) = f_1(\underline{x}(k)) + b_1u_1(k) + d_1(k)\\
&x_{n_1+1}(k+1) = x_{n_1+2}(k)\\
&x_{n_1+2}(k+1) = x_{n_1+3}(k)\\
&\cdot\\
&x_{n_1+n_2}(k+1) = f_2(\underline{x}(k)) + b_2u_2(k) + d_2(k)\\
&\cdot\\
&x_{n_1+n_2+\ldots+n_{m-1}+1}(k+1) = x_{n_1+n_2+\cdots+n_{m-1}+2}(k)\\
&x_{n_1+n_2+\ldots+n_{m-1}+2}(k+1) = x_{n_1+n_2+\cdots+n_{m-1}+3}(k)\\
&\cdot\\
&x_n(k+1) = f_m(x(k)) + b_mu_m(k) + d_m(k)
\end{aligned} \tag{17}$$

with the output equation given as

$$y(k) = \begin{bmatrix} x_1(k) \\ x_{n_1+1}(k) \\ \cdot \\ x_{n_1+n_2+\cdots+n_{m-1}+1}(k) \end{bmatrix} \tag{18}$$

where $y(k)$ denotes the sampled value of $y(t)$ at $t = kT$, and T is the sampling period. It is assumed that the coefficients b_i, $1 \le I \le m$ are known. $d(k) = [d_1(k), d_2(k), \ldots, d_m(k)]^T$ is an unknown disturbance with known upper bound so that $\|d\| < b_d$, $\underline{x}(k) = [x_1(k), x_2(k), \ldots, x_n(k)]^T \in R^n$, and $f = [f_1, f_2, \ldots, f_m]^T : R^n \to R^m$ is a smooth vector function.

Output Tracking Problem. Given the system in Eqs. (17) and (18), it is required to manufacture a bounded control input $u(k) = [u_1(k), u_2(k), \ldots, u_m(k)]^T$ such that the output $y(k)$ of the system tracks a specified desired output $y_d(k) = [y_{d_1}(k), y_{d_2}(k), \ldots, y_{d_m}(k)]^T$ while ensuring that the states $\underline{x}(k)$ are bounded. It is assumed that the desired output satisfies

$$\left[\begin{Vmatrix} y_d(k) \\ y_d(k+1) \\ \cdot \\ y_d(k+n) \end{Vmatrix}\right] \le \gamma, \qquad k = 0, 1, 2, \ldots, N-1 \tag{19}$$

Feedback Linearizing Controller. The tracking problem above can be solved using a feedback linearizing controller if the complete dynamics in Eq. (20) are known. In this implementation the system is first expressed in terms of the filtered error system and the filter gains selected to make the error dynamics Hurwitz (Table 1). The control input is then

Table 1

Tracking error	$e_i(k) = y_i(k) - y_{d_i}(k)$
Filtered tracking error	$r_i(k) = e_i(k) + \lambda_{i,n_i-1}e_i(k-1) + \ldots + \lambda_{i,1}e_i(k-n_i+1)$ $1 \le i \le n$
Control input	$u_i(k) = \{-f_i(\underline{x}(k)) - K_{v_i}r_i(k) - [\lambda_{i,n_i-1}e_i(k) + \lambda_{i,n_i-2}e_i(k-1) + \ldots + \lambda_{i,1}e_i(k-n_i+2)] + y_{d_i}(k+1)\}/b_i$ $1 \le i \le m$
Filtered tracking error system	$r_i(k+1) = K_{v_i}r_k(k) + d_i(k)$

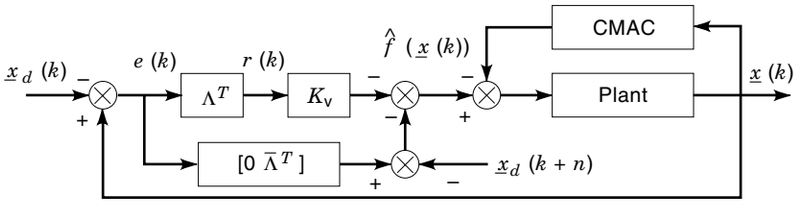

Figure 6. Control of an unknown nonlinear system using CMAC neural network. The controller includes an inner feedback linearizing loop and an outer tracking loop.

computed to force the filtered tracking error to be bounded, which in turn guarantees that the error and all its derivatives are bounded (39,45).

Adaptive CMAC Control. In the implementation of the controller in Table 1, it is assumed that the function $f(.)$ is known. However, in practice, $f(.)$ is unknown, since the information on the dynamics of the system is only partially known. The approach of the preceding section can still be used if an estimate $\hat{f}(x)$ of $\hat{f}(\underline{x})$ of $f(.)$ is available. According to Corollary 1, any nonlinear function can be approximated to any required degree of accuracy using a CMAC neural network. The output of the CMAC is then given as

$$f(\underline{x}(k)) = w^T(k)\Gamma(\underline{x}(k)) \tag{20}$$

where w is a matrix of weights and $\Gamma(\underline{x})$ is the vector of receptive field activation values based on n-dimensional *second-order* splines. However, for such a network to ensure small tracking error in closed-loop control, the weights (e.g., sample values of $f(.)$) associated with the network must be known. Since $f(.)$ is unknown in control applications, it is necessary to learn the weights on-line. In Refs. 39 and 45, a learning law was derived that ensured the stability of the overall filtered tracking error system (Table 1).

Theorem 5 For the system in Eqs. (17) and (18) let the inputs be selected as in (Table 1) (39,45). Further let the estimate of the nonlinearity $\hat{f}(.)$ be manufactured by a CMAC NN in Eq. (20). Let the CMAC weights be tuned on-line by

$$\hat{w}_{k+1} = \alpha\hat{w}_k - \beta R_k r_{k+1}^T \tag{21}$$

with α, $\beta > 0$ design parameters. Then for small enough outer-loop gains K_{v_i} (as specified in the proof), the filtered tracking error $r(k)$ and the weight estimates $\hat{w}(k)$ are Uniformly Ultimately Bounded (UUB). Further, the filtered tracking error can be made arbitrarily small by proper selection of the feedback gains K_{vi}.

The proposed control scheme (Table 1) is shown in Fig. 6. Note that the structure has a nonlinear CMAC inner loop plus a linear outer tracking loop. The CMAC inner loop learns the unknown nonlinear dynamics, while the outer tracking loop ensures stability of the closed loop system. As the CMAC learns, more of the stabilization role is assumed by the CMAC, which cancels out the nonlinear terms in the dynamics.

Remark 1 The first term in Eq. (21) is a gradient term that ensures stability of the weight update algorithm. The second term is necessary to overcome the requirement of *persistency of excitation* condition (46) for the convergence of the weights and ensures robustness in the closed-loop.

Remark 2 (39,45) $\Gamma^T(\underline{x}(k))\Gamma(\underline{x}(k)) < 1$.

Remark 2 explains how the CMACs overcome one of the serious difficulties in the implementation of fully connected NNs. In the fully connected NNs, the adaptation rate a must satisfy the condition a must satisfy the condition $\alpha\|\phi^T(\underline{x}(k))\phi(\underline{x}(k))\| < 1$, where $\phi(.)$ is the vector of the activation function of each node. Therefore, as the number of nodes increase, a must decrease thereby slowing the rate of adaptation (15). In the case of CMAC, however, since $\Gamma^T(\underline{x}(k))\Gamma(\underline{x}(k)) < 1$, the rate of adaptation can be chosen independent of the partitioning of the input space. This, together with the localized learning in CMAC, ensures quick convergence of the weights of the CMAC and better tracking performance.

Numerical Example. As an example (45), the controller proposed in the preceding sections is tested on the system given by the following set of equations:

$$\begin{aligned} \dot{x}_1 &= x_2 + u_1 \\ \dot{x}_2 &= x_1 + 2e^{-(x_1^2+x_2^2)}x_2 - 0.1x_2 + u_2 \end{aligned} \tag{22}$$

The system outputs are

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_2 \end{aligned} \tag{23}$$

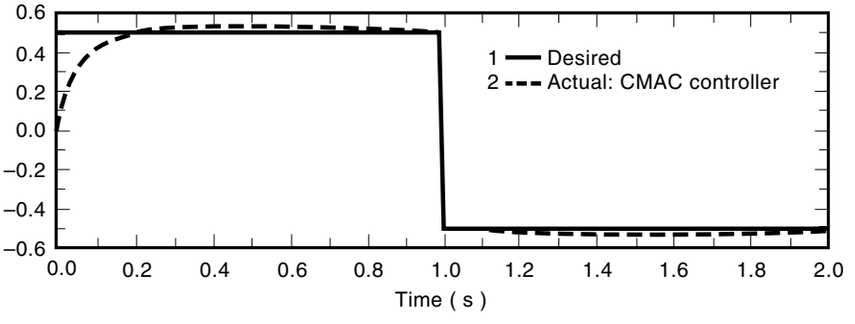

Figure 7. Actual and desired output y_1 with the discrete-time CMAC controller.

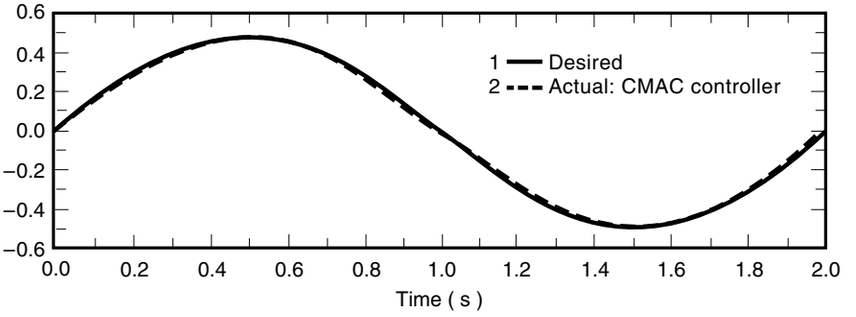

Figure 8. Actual and desired output y_2 with the discrete-time CMAC controller.

The control inputs u_1 and u_2 are to be selected so that y_1 tracks a square signal and y_2 tracks a sinusoidal signal of period 2 seconds.

In the implementation of the CMAC controller for the system in Eqs. (22) and (23), the system is first discretized for a sample period of 10 milliseconds. The CMAC is then required to manufacture the nonlinearities in the system dynamics. In order to achieve this, the receptive fields for the CMAC NN are selected to cover the input space $\{[-2, 2] \times [-2, 2]\}$ with knot points at intervals of 0.25 along each input dimension. The initial conditions for both the states x_1 and x_2 are taken to be zero. Figures 7 and 8 show the desired and actual outputs for the MIMO system in Eqs. (22) and (23) using the CMAC NN controller (Table 1). It is seen that although *578 weights are needed to define the output in Eq. (22), only 8 (2×2^2) weights are updated at any given instant.* In other words, the performance of the CMAC controller is good even though the CMAC controller knows none of the dynamics a priori.

Class of State-Feedback Linearizable Nonlinear Systems

A class of mnth order multi-input multi-output (MIMO) state-feedback linearizable system in the controllability canonical form is given as

$$\begin{aligned} \dot{x}^1 &= x^2 \\ \dot{x}^2 &= x^3 \\ &\vdots \\ \dot{x}^n &= f(\mathbf{x}) + g(x)u + d \\ y &= x^1 \end{aligned} \tag{24}$$

with state $x_i = [x^1 x^2 \ldots x^n]^T \in \Re^n$ for $i = 1, \ldots, m$, output $y_i(t) \in \Re^m$ and control u. It is assumed that the unknown disturbance $d(t) \in \Re^m$ has a constant known upper bound so that $|d(t)| < b_d$, and that $f, g : \Re^{mn} \to \Re^m$ are smooth unknown functions with $|g(x)| \geq g > 0$ for all x, where g is a known lower bound.

Tracking Problem. The output tracking problem for this class of systems can be handled using the same design procedure as in the preceding section. The chief difference is that for this class of systems, the control coefficient $g(x)$ is not constant but a function of the states of the system. Let

$$\mathbf{x}_d(t) \equiv [y_d \dot{y}_d \ldots y_d^{(n-1)}]^T \tag{25}$$

be the desired output vector or the trajectory to be tracked. Here the superscript in parenthesis indicates the order of the operator d/dt. It is assumed that the desired trajectory vector y_d is continuous and bounded and that the sign of $g(x)$ is known.

The state-feedback linearizing controller is implemented as shown in Table 2. The system is first expressed in terms of the filtered error system and the filter gains selected to make the error dynamics Hurwitz (Table 2). The control input is then computed to force the filtered tracking error small, which in turn guarantees that the error and all its derivatives are bounded (34).

CMAC NN Controller. The controller in Table 2 cannot be implemented in practice as the functions $f(.)$ and $g(.)$ are unknown. As seen earlier, the controller can be implemented using estimates of $f(.)$ and $g(.)$. In order to approximate $f(.)$ and $g(.)$, two CMAC NN systems are employed. Using the approximation property of the CMAC, $f(.)$ and $g(.)$ can be written as

$$f(x) = W_f^T \Gamma_f(x) + \epsilon_f \tag{26}$$

$$g(x) = W_g^T \Gamma_g(x) + \epsilon_g \tag{27}$$

where W_f, W_g are vectors and ϵ_f, ϵ_g are the maximal function reconstruction errors for $f(.)$ and $g(.)$, respectively. Let $\hat{f}(x)$ and $\hat{g}(x)$ be the estimates of $f(.)$ and $g(.)$ generated by the CMACs. The controller can then be implemented as in Table 3 (34). The closed-loop implementation is as shown in Fig. 9.

Robot Arm Control

The dynamics of an n-link robot manipulator may be expressed in the Lagrange form as (47)

$$M(q)\ddot{q} + V_m(q, \dot{q})q + G(q) + F(\dot{q}) + \tau_d = \tau \tag{28}$$

Table 2

Tracking error	$e = x - x_d$
	$e_{i+1} \equiv y^{(i)}(t) - y_d^{(i)}(t), i = 1, 2, \ldots, n-1$
Filtered error	$r = \Lambda^T e$, where $\Lambda = [\underline{\Lambda}\ 1] = [\lambda_1 \lambda_2 \ldots \lambda_{n-1} 1]^T$
	$s^{n-1} + \lambda_{n-1} s^{n-2} + \ldots + \lambda_1$ is Hurwitz.
Filtered tracking error dynamics	$\dot{r} = f(\mathbf{x}) + g(x)u + d + Y_d$
	where $Y_d \equiv -y_d^{(n)} + \sum_{i=1}^{n-1} \lambda_i e_{i+1}$
Control input	$U = \frac{1}{g(x)}[-f(x) - y_d - \Lambda r]$
Closed-loop dynamics	$\dot{r} = \Lambda r + d$

Table 3

Auxiliary control input	$u_c = \begin{cases} \frac{1}{\hat{g}(x)}[-\hat{f}(x)+v] \\ v = -K_v r - Y_d, K_v > 0. \end{cases}$
Robustifying control input	$u_r = -\mu \frac{\lvert u_c \rvert}{\underline{g}} \operatorname{sgn}(r)$
Control input	$u = \begin{cases} u_c + \frac{u_r - u_c}{2} e^{\gamma \lvert u_c \rvert - s)} & \text{if } I = 1 \\ u_r - \frac{u_r - u_c}{2} e^{-\gamma \lvert u_c \rvert - s)} & \text{if } I = 0 \end{cases}$
Indicator	$I = \begin{cases} 1 & \text{if } \hat{g} \geq \underline{g} \text{ and } \lvert u_c \rvert \leq s \\ 0 & \text{otherwise} \end{cases}$
Design parameters	$\gamma < \ln 2/s, \mu > 0, M_f, M_g > 0$, and $s > 0$
Weight update for $\hat{f}$	$\dot{\hat{W}}_f = M_f \Gamma_f(x) r - \kappa M_f \lVert r \rVert \hat{W}_f$
Weight update for $\hat{g}$	$\dot{\hat{U}}_g = \begin{cases} M_g \Gamma_g(x) r - \kappa M_g \lVert r \rVert \hat{W}_g & \text{if } I = 1 \\ 0 & \text{otherwise} \end{cases}$

with $q(t) \in R^n$ the joint variable vector, $M(q)$ the inertia matrix, $V_m(q, \dot{q})$ the coriolis/centripetal vector, and $F(\dot{q})$ the friction component. Bounded unknown disturbances are denoted by τ_d, and τ is the control torque. It is assumed that τ_d is an unknown disturbance with a known upper bound b_d so that $\lVert \tau_d \rVert \leq b_d$. The control problem is then to design a control input τ such that the joint angles $q(t)$ track a desired trajectory $q_d(t)$.

Conventional Controller Design. Traditionally the controller problem has been attempted by linearizing the robot system in some region of operation and then designing a linear proportional-derivative (P-D) or proportional-integral-derivative (PID) controller for the system. That is, the system in Eq. (28) is first expressed as

$$\ddot{q}(t) = M^{-1}(q)\{-V_m(q, \dot{q})\dot{q} - G(q) - F(\dot{q}) - \tau_d\} + M^{-1}(q)\tau \tag{29}$$

In practice, it is known that $M^{-1}(.)$ exists, and hence the linear equivalent of Eq. (28) can be found about any operating point. Thus, given any smooth desired trajectory $q_d(t)$, neglecting the coriolis, gravity, and the friction terms, the control input can be designed as

$$\tau = -\underline{M}\{K_v \dot{e} + K_p e - \ddot{q}_d\} \tag{30}$$

where the tracking error is defined as $e(t) \equiv q(t) - q_d(t)$, $\underline{M}$ is a constant diagonal matrix approximation of the inertia matrix, and K_v, K_p are constant diagonal matrices of the derivative and proportional gains.

With this control, Eq. (29) can be rewritten as

$$\begin{aligned} \ddot{q}(t) = {} & M^{-1}(q)\{-V_m(q, \dot{q})\dot{q} - G(q) - F(\dot{q}) - \tau_d\} \\ & - M^{-1}(q)\underline{M}\{K_v \dot{e} + K_p e - \ddot{q}_d\} \end{aligned} \tag{31}$$

Simplifying and rearranging, we get

$$\begin{aligned} \ddot{e}(t) + K_v \dot{e}(t) + K_p e = {} & M^{-1}(q)\{-V_m(q, \dot{q})\dot{q} - G(q) - F(\dot{q}) - \tau_d\} \\ & + (I - M^{-1}(q)\underline{M})\{K_v \dot{e} + K_p e - \ddot{q}_d\} \end{aligned}$$

Defining

$$\begin{aligned} f(q, \dot{q}) = {} & M^{-1}(q)\{-V_m(q, \dot{q})\dot{q} - G(g) - F(\dot{q})\} \\ & + (I - M^{-1}(q)\underline{M})\{K_v \dot{e} + K_p e - \ddot{q}_d\} \end{aligned} \tag{32}$$

$$\ddot{e}(t) + K_v \dot{e}(t) + K_p e = f(q, \dot{q}) - M^{-1}(q)\tau_d \tag{33}$$

In conventional controller design, it is standard practice to design $\underline{M}$ such that $\lVert I - M^{-1}(q)\underline{M} \rVert$ is small. Also for nominal trajectories, the effects of the centripetal, coriolis, and the friction terms on Eq. (31) are small. Therefore $f(q, \dot{q})$ is small and can be neglected. This design guarantees adequate performance in the designed region of operation, but the tracking response degenerates rapidly if the region of operation is enlarged. Moreover, even for a given region of operation, the effect of $f(q, \dot{q})$ cannot be neglected if the robot is required to operate at high speeds. This in essence becomes a serious bottleneck to enlarging the envelope of the robot performance (47).

Robot Control Using State-Feedback Linearization Approach. The use of CMAC NN in designing feedback linearizing controllers can be extended to control the robotic system in Eq. (29) (34). Consider a two-link robot arm (47) where the first link is 1 m long and weighs 1 kg and the second link is 1 m long and weighs 2.3 kg. The first joint is required to track a trajectory $q_{d_1} = 0.3 \sin(t)$ and the second joint is required to track a trajectory $q_{d_2} = 0.3 \cos(t)$. The controller parameters were selected as $k_v = \text{diag}\{5,5\}$, $\Lambda = \text{diag}\{5,5\}$, and the diagonal elements of the design matrix F are taken to be 10 with $\kappa = -2$. The response of the system with the CMAC controller

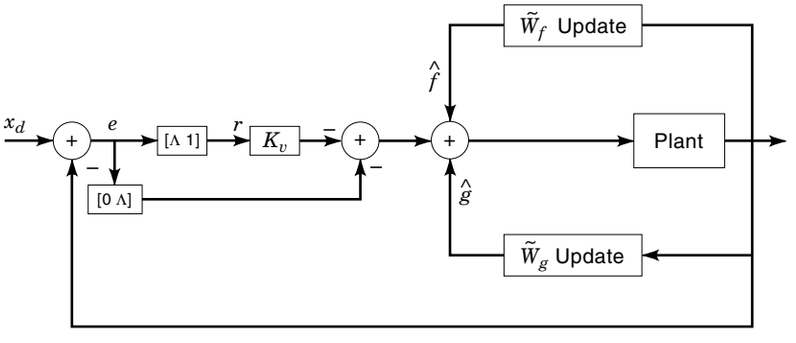

Figure 9. Structure of the feedback linearizing CMAC controller. The controller has two adaptive loops: one for generating the estimate of the unknown function $f(.)$ and the other for generating the estimate of the unknown function $g(.)$.

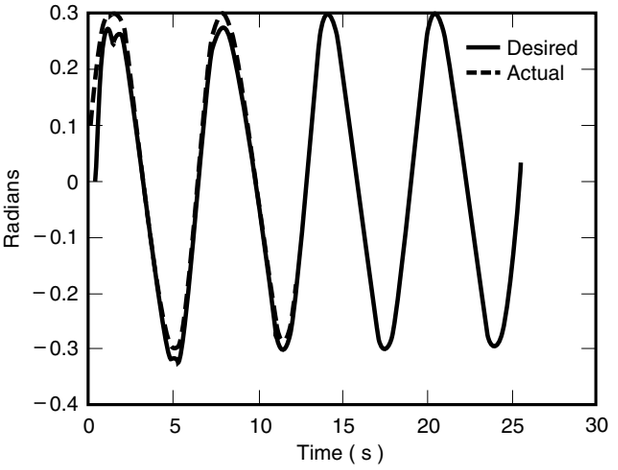

Figure 10. Robot control—Joint 1 response with CMAC controller.

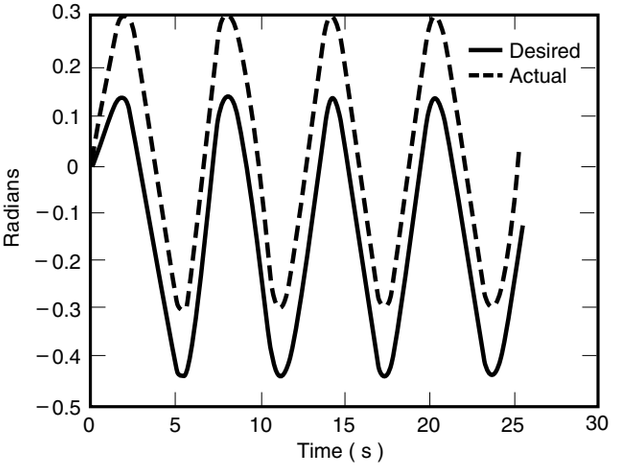

Figure 12. Robot control—Joint 1 response without CMAC controller.

is shown in Figs. 10 and 11. From these figures it is evident that after a short learning phase, the system is able to track the desired trajectories effectively. Figures 12 and 13 show the response of the system without the CMAC NN in the feedback loop. From these results it is clear that the CMAC NN does improve on the linear design.

Intelligent Control Formulation of the Robot Control Problem. While it is well known that the robot control problem can be satisfactorily addressed using the *filtered tracking error* formulation of the control problem (16,17,47), this approach would entail a complete redesign of the controller. Here we propose an alternative strategy based on techniques rooted in intelligent control literature. It will be shown that the thorny problem of the neglected dynamics can be easily handled by adding a feedforward component to the controller designed in Eq. (31). The feedforward component is adaptive in nature and can be manufactured using neural networks, adaptive controllers, or fuzzy logic networks (39,45). Here we restrict the presentation to CMAC neural networks.

Let the modified control be defined as

$$\tau = -\underline{M}\{K_v\dot{e} + K_p e - \ddot{q}_d + \hat{f}(q, \dot{q})\} \tag{34}$$

where $\hat{f}(q, \dot{q})$ is the output generated by a CMAC NN. The error dynamics of the system in Eq. (29) under this new control can be written in the form

$$\ddot{e}(t) + K_v\dot{e}(t) + K_p e = f(q, \dot{q}) - M^{-1}(q)\tau_d - M^{-1}(q)\underline{M}\hat{f}(q, \dot{q}) \tag{35}$$

Defining

$$N = f(q, \dot{q}) - M^{-1}(q)\underline{M}\hat{f}(q, \dot{q}) - M^{-1}(q)\tau_d \tag{36}$$

and the state $\underline{e} \equiv [e^T \quad \dot{E}^T]^T$, the error equation in Eq. (36) can be put in the state-space form as

$$\dot{\underline{e}} = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix} \underline{e} + \begin{bmatrix} 0 \\ N \end{bmatrix} \tag{37}$$

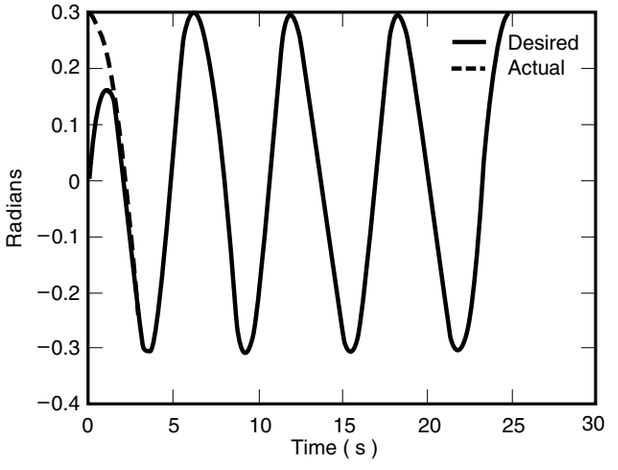

Figure 11. Robot control—Joint 2 response with CMAC controller.

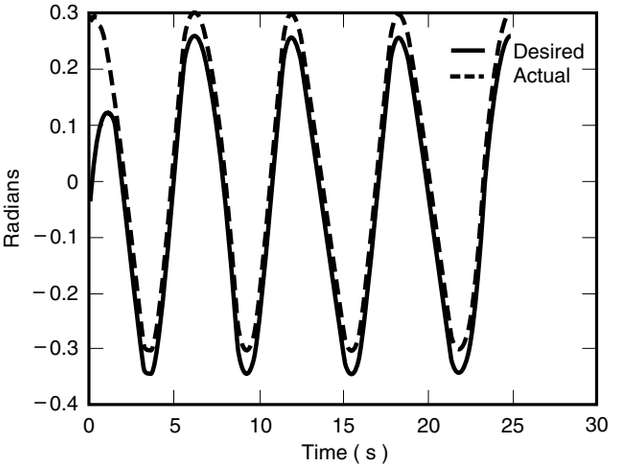

Figure 13. Robot control—Joint 2 response without CMAC controller.

Now, if the CMAC NN is designed such that

$$\hat{f}(q, \dot{q}) = \underline{M}^{-1} M(q) f(q, \dot{q}) \tag{38}$$

then

$$f(q, \dot{q}) - M^{-1}(q)\underline{M}\hat{f}(q, \dot{q}) \equiv 0 \tag{39}$$

Then in the absence of disturbances, perfect tracking can be achieved. However, since $f(q, \dot{q})$ and $M(q)$ are not known in practice, the CMAC NN can be designed to learn the dynamics online and ensure that $\|N\|$ in Eq. (36) is small. In fact this bound on $\|N\|$ influences the overall closed-loop performance and can be made as small as desired by proper choice of the learning laws for the CMAC NN.

Theorem 6 For the system in Eq. (28) let the inputs be selected as in Eq. (36) (36). Let k_1 be a positive constant such that

$$\underline{e}^T \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix} \underline{e} \leq -k_1 \|e\|^2$$

Further let the estimate of the nonlinearity $\hat{f}(.)$ be manufactured by a CMAC NN in Eq. (41). Let the weights of the CMAC NN be tuned on-line by the following update laws:

$$\dot{\hat{w}} = \hat{\Phi}(x) r^T - k_1 \|r\| \hat{w} \tag{40}$$

with k_2 a positive design parameter. Then for large enough outer-loop gain k_1, the tracking error $e(t)$, and the weight estimates are UUB. Further the tracking error can be made arbitrarily small by proper selection of the feedback gains K_p and K_v.

PASSIVITY-BASED DESIGN

Earlier CMAC controllers were presented that guarantee closed-loop tracking performance for nonlinear systems with unknown dynamics. The stability of these controllers was proved using Lyapunov stability analysis. While this technique guarantees closed-loop stability of the overall system, it does not give insight into the actual selection of CMAC learning laws for a particular application. In recent work (37,38), the CMAC design was studied from an input-output point of view, and conditions that guarantee closed-loop stability were derived. These results give insight into the selection of learning laws for a given class of systems and are presented in the following subsection.

Background on Passivity

The relationship between the input-output properties of a system and its stability have been extensively studied using the theory of *dissipative systems* (48–53). These results were later extended to derive conditions for nonlinear systems subjected to bounded disturbances (37,38). It can be shown that the CMAC neural network used for control purposes can be constructed to have an important *dissipativity* property that makes it *robust* to disturbances and unmodeled dynamics.

Assumption 1 Let the system in Eqs. (17) and (18) satisfy the following conditions:

a. $f(0) = y(0) = 0$.
b. The system is completely reachable; that is, for a given $\underline{x}(t_f)$ there exists a constant N, and bounded controls $u(k)$, $k = 0,1,2, \ldots, N-1$ such that the state can be driven from $\underline{x}(0) = 0$ to $\underline{x}(t_f = NT)$.
c. $\sigma(u, y, T_c)$ is an energy supply rate associated with this system such that

$$\sigma(u, y, T_c) = \langle y, Qy \rangle_{T_c} + 2\langle y, Su \rangle_{T_c} + \langle u, Ru \rangle_{T_c} \tag{41}$$

where Q, R, S are constant matrices with Q and $R > \phi$ symmetric and $\langle .,. \rangle$ is the inner product.

Definition 1 A system is *state-strict passive* if it is (a) passive (48–53) and (b) there exists a real function $\Psi(.)$ satisfying $\Psi(\underline{x}(k)) > 0$ $\forall \underline{x}(k) \neq 0$, $\Psi(0) = 0$, and

$$\Psi(\underline{x}(k+1)) - \Psi(\underline{x}(k)) \leq y'(k)u(k) - \epsilon \underline{x}^T(k)\underline{x}(k) \tag{42}$$

where $\underline{x}$ is the state of the system. Equation (42) is referred to in literature as the *power form*.

Theorem 7 Consider the system of the form shown in Fig. 14. Suppose that the subsystems H_1 and H_2 are *state-strict passive* with the supply rates $\sigma_1(u, y, T_c)$ and $\sigma_2(u, y, T_c)$. Further let H_1 satisfy

$$\|y_1(k)\| \leq \alpha \|\underline{x}(k)\|, \qquad \alpha < 0 \tag{43}$$

Then the feedback system is UUB for all bounded inputs $e_1(k)$.

Theorem 7 is a powerful tool for analyzing the internal stability of interconnected systems. In fact this result is instrumental in developing a new approach to designing adaptive controllers (54). The use of this result is demonstrated in the following lemmas.

Define $\xi(k) = \tilde{f}(\underline{x}(k)) + d(k)$. Then the filtered error system (Table 1) can be expressed in vector notation as

$$r(k+1) = -K_v r(k) + \xi(k) \tag{44}$$

Lemma 8 The dynamics in Eq. (44) from $\xi(k)$ to $r(k)$ are a *state-strict passive system.*

Lemma 9 The weight update law in Eq. (21) guarantees the CMAC neural network to be state-strict passive from input $r(k)$ to $\tilde{w}^T(k)R(k)$ (Fig. 15).

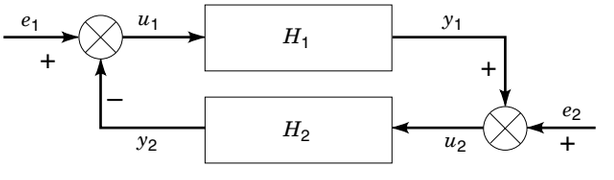

Figure 14. Interconnection of two subsystems in feedback configuration.

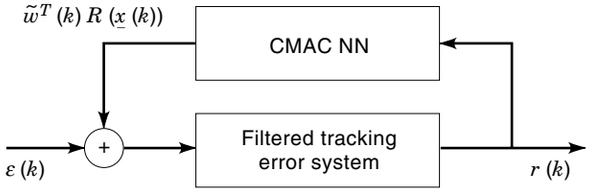

Figure 15. Interconnected feedback structure of the CMAC controller.

Lemma 8 and Lemma 9 together guarantee the closed-loop system to be state-strict passive. Further, by Theorem 7, the closed-loop system is UUB. It is seen that state-strict passivity is needed to ensure boundedness of all states when the closed-loop system is subjected to bounded disturbances. While it is well-known in literature that passivity of each subsystem is sufficient to prove that the closed-loop system is Lyapunov stable (52), this result shows that state-strict passivity is needed to guarantee UUB for passive systems subjected to bounded exogenous inputs (54). The choice of the tuning laws for the CMAC NN is crucial, since this guarantees the boundedness of weights without the *persistency of excitation* (PE) condition required in most adaptive control techniques. Therefore for constructing stable CMAC NN controllers, it is sufficient to ensure that the approximation property (Corollary 1) holds and that the learning laws are selected to make the CMAC state-strict passive.

EVALUATION

In this article the application of CMAC NNs in the discrete-time control of nonlinear systems was presented. The structure of the CMAC NNs and the properties of the output generated were studied. It can be seen that the structure of the CMAC results in localized learning. This property can be exploited in implementing efficient controllers. CMAC controllers for three standard nonlinear systems were examined. These controllers have two components: an outer tracking-loop and an inner-loop comprising of CMAC NN for manufacturing the nonlinear elements in the dynamics. While the weight-update laws can be derived using Lyapunov stability analysis, closed-loop stability can also be guaranteed using a passivity-based design. The chief advantage of these controllers is that they are model free, and the localized-learning enables quick learning of the nonlinear dynamics. Further, from a practical standpoint, the controller structure leads to efficient implementation.

While CMAC NNs are extremely attractive from a practical standpoint, the main factors that make them attractive must be kept in mind. It is seen that the approximation property is very important for closed-loop control. Therefore in the implementation of the CMAC, care must be taken to ensure that the CMAC can indeed generate the function with the desired degree of accuracy. In doing so, the effect of hashing must be kept in mind. Another important factor in the controller design is the selection of the closed-loop gains for the outer tracking loop. These gains ensure that the closed-loop system is stable while the CMAC NN learns the dynamics of the system. The tradeoff between the closed-loop gains and the rate of learning must be understood for effective controller design.

BIBLIOGRAPHY

1. K. S. Narendra and R. V. Monopoli, *Applications of Adaptive Control,* New York: Academic Press, 1980.
2. Y. D. Landau, *Adaptive Control: The Model Reference Approach,* New York: Dekker, 1979.
3. K. J. Åström and B. Wittenmark, On self tuning regulators, *Automatica,* **9**: 185–199, 1973.
4. K. J. Åström and B. Wittenmark, *Adaptive Control,* Reading, MA: Addison-Wesley, 1989.
5. K. S. Narendra, Adaptive control using neural networks, in W. T. Miller, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control,* Cambridge, MA: MIT Press, 1991, pp. 115–142.
6. K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.,* **1**: 4–27, 1990.
7. P. J. Werbos, Back propagation: Past and future, *Proc. 1988 Int. Conf. Neural Netw.,* Vol. 1, pp. 1343–1353, 1989.
8. Y. Iiguni, H. Sakai, and H. Tokumaru, A nonlinear regulator design in the presence of system uncertainties using multilayer neural networks, *IEEE Trans. Neural Netw.,* **2**: 410–417, 1991.
9. K. S. Narendra and K. Parthasarathy, Gradient methods for the optimization of dynamical systems containing neural networks, *IEEE Trans. Neural Netw.,* **2**: 252–262, 1991.
10. T. Yabuta and T. Yamada, Neural network controller characteristics with regard to adaptive control, *IEEE Trans. Syst. Man Cybern.,* **22**: 170–176, 1992.
11. N. Sadegh, A perceptron network for functional identification and control of nonlinear systems, *IEEE Trans. Neural Netw.,* **4**: 982–988, 1993.
12. G. A. Rovithakis and M. A. Cristodoulou, Adaptive control of unknown plants using dynamical neural networks, *IEEE Trans. Syst., Man Cybern.,* **24**: 971–981, 1994.
13. M. M. Polycarpou and P. A. Ioannou, Neural networks as on-line approximators of nonlinear systems, *Proc. IEEE Conf. Decision Control,* Tucson, AZ, pp. 7–12, 1992.
14. R. M. Sanner and J.-J. E. Slotine, Stable adaptive control and recursive identification using radial gaussian networks, *Proc. IEEE Conf. Decision Control,* Brighton, 1991.
15. S. Jagannathan and F. L. Lewis, Multilayer discrete-time neural net controller with guaranteed performance, *IEEE Trans. Neural Netw.,* **7**: 107–130, 1996.
16. F. L. Lewis, A. Yesildirek, and K. Liu, Neural net robot controller: Structure and stability proofs, *J. Intell. Robot. Syst.,* **12**: 277–299, 1995.
17. F. L. Lewis, K. Liu, and A. Yesildirek, Neural net robot controller with guaranteed tracking performance, *IEEE Trans. Neural Netw.,* **6**: 703–715, 1995.
18. S. H. Lane, D. A. Handelman, and J. J. Gelfand, Theory and development of higher-order CMAC neural networks, *IEEE Control Syst. Technol.,* **12**: 23–30, 1992.
19. L. G. Kraft, W. T. Miller, and D. Dietz, Development and application of CMAC neural network-based control, in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches,* New York: Van Nostrand-Reinhold, 1992, pp. 215–232.
20. L. G. Kraft and D. P. Campagna, A comparison between CMAC neural network control and two traditional adaptive control systems, *IEEE Control Syst. Technol.,* **10**: 36–43, 1990.
21. L. G. Kraft and D. P. Campagna, A summary comparison of CMAC neural network and traditional adaptive control systems,

in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches,* New York: Van Nostrand-Reinhold, 1992, pp. 143–169.

22. J. S. Albus, A new approach to manipulator control: The cerebellar model articulation controller (CMAC), *Trans. ASME J. Dyn. Syst. Meas. Control,* **97**: 220–227, 1975.
23. R. M. Sanner and J.-J. E. Slotine, Function approximation, neural networks, and adaptive control, pp. 1225–1232, 1994.
24. P. K. Simpson, Foundations of neural networks, in E. Sánchez-Sinencio and C. Lau (eds.), *Artificial Neural Networks: Paradigms, Applications, and Hardware Implications,* New York: IEEE Press, 1992, pp. 3–24.
25. D. Campagna and L. G. Kraft, Stability analysis of open-loop learning in CMAC neural networks, preprint, 1998.
26. J. S. Albus, Data storage in the cerebellar model articulation controller, *Trans. ASME J. Dyn. Syst. Meas. Control,* **97**: 228–233, 1975.
27. H. Tolle and E. Ersu, Lecture notes in control and information sciences, in M. Thoma and A. Wyner (eds.), *NeuroControl,* Berlin: Springer-Verlag, 1992, Vol. 172.
28. C. J. Harris, C. G. Moore, and M. Brown, Aspects of fuzzy logic and neural nets, *World Scientific Series in Robotics and Automated Systems,* Singapore: World Scientific, 1993, Vol. 6.
29. H. Wang et al., *Advanced Adaptive Control,* Oxford: Pergamon and Elsevier, 1995.
30. P. C. Parks and J. Militzer, Convergence properties of associative memory storage for learning control systems, *Autom. Remote Control,* **50** (P. II): 254–286, 1989.
31. P. C. Parks and J. Militzer, Convergence properties of associative memory storage for learning control systems, *Proc. IFAC Symp. Adapt. Syst. Control Signal Process.,* Vol. 2, 565–572, 1989.
32. D. Ellison, On the convergence of the multi-dimensional Albus perceptron, *Int. J. Robot. Res.,* **10**: 338–357, 1991.
33. S. Commuri and F. L. Lewis, CMAC neural networks for control of nonlinear dynamical systems: Structure, stability, and passivity, *Automatica,* **33**: 635–641, 1997
34. S. Commuri, S. Jagannathan, and F. L. Lewis, CMAC neural network control of robot manipulators, *J. Robot. Syst.,* **14**: 465–482, 1997.
35. S. Jagannathan, S. Commuri, and F. L. Lewis, Feedback linearization using CMAC neural networks, *Proc. IEEE International Conf. Decision Control,* Vol. 3, pp. 3304–3309, 1996.
36. S. Commuri and S. Jagannathan, Modular controls design for robot manipulators using CMAC neural networks, *Proc. IEEE Int. Conf. Robot. Autom.,* pp. 1725–1730, 1997.
37. S. Commuri and F. L. Lewis, A new methodology for the design of adaptive controllers using "state-strict passivity": Application to neural network controllers, in Special Issue: New Directions in Control and Automation II, *Kybernetica,* **33**: 1–16, 1997.
38. S. Commuri and F. L. Lewis, A framework for the discrete time design of intelligent controllers using "state-strict passivity," *Proc. ISAI/IFIS,* Cancun, Mexico, pp. 386–393, 1996.
39. S. Commuri, A framework for intelligent control of nonlinear systems, Ph.D. thesis, Dept. of Electrical Engineering, University of Texas at Arlington, 1996.
40. Y. Wong and A. Sideris, Learning convergence in the cerebellar model articulation controller, *IEEE Trans. Neural Netw.,* **3**: 115–121, 1992.
41. W. T. Miller et al., Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller, *IEEE Trans. Robot. Autom.,* **6**: 1–9, 1990.
42. H. Shiraish, S. L. Ipri, and D. D. Cho, CMAC neural network controller for fuel injection systems, *IEEE Trans. Control Syst. Technol.,* **3**: 32–38, 1995.
43. B. Widrow and M. E. Hoff, Adaptive switching circuits, *Wescon Conv. Rec.,* pp. 96–104, 1960.
44. B. Widrow and S. Stearns, *Adaptive Signal Processing,* Englewood Cliffs, NJ: Prentice-Hall, 1985.
45. S. Commuri and F. L. Lewis, Discrete-time CMAC neural networks for control applications, *Proc. IEEE Conf. Decision Control,* 1995.
46. K. S. Narendra and A. M. Annaswamy, A new adaptive law for robust adaptation without persistent adaptation, *IEEE Trans. Autom. Control,* **32**: 134–145, 1987.
47. F. L. Lewis, C. T. Abdallah, and D. M. Dawson, *Control of Robot Manipulators,* New York: Macmillan, 1993.
48. P. J. Moylan, Implications of passivity in a class of nonlinear systems, *IEEE Trans. Autom. Control,* **19**: 373–381, 1974.
49. D. J. Hill and P. J. Moylan, Dissipative dynamical systems: basic input-output and state properties, *J. Franklin Inst.,* **309**: 328–357, 1980.
50. G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction, and Control,* Englewood Cliffs, NJ: Prentice-Hall, 1984.
51. D. J. Hill and P. J. Moylan, The stability of nonlinear dissipative systems, *IEEE Trans. Autom. Control,* **21**: 708–711, 1976.
52. D. J. Hill and P. J. Moylan, Stability results for nonlinear feedback systems, *Automatica,* **13**: 377–382, 1977.
53. P. J. Moylan and D. J. Hill, Stability criteria for large-scale systems, *IEEE Trans. Autom. Control,* **23**: 143–149, 1978.
54. S. Commuri and F. L. Lewis, A new methodology for the design of adaptive controllers using "state-strict passivity," *Proc. 3rd IEEE Mediterranean Symp. New Directions Control Autom.,* pp. 265–271, 1995.

Reading List

A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inf. Theory,* **9** (3): 930–945, 1993.

P. M. Prenter, *Splines and Variational Methods,* New York: Wiley, 1975.

R. M. Sanner and J.-J. E. Slotine, Function approximation, "Neural" networks, and adaptive nonlinear control, *Proc. IEEE Conf. Control Appl.,* Glasgow, Vol. 2, pp. 1225–1232, 1994.

S. COMMURI
CGN & Associates, Inc.
F. L. LEWIS
The University of Texas at Arlington

CHANGE MANAGEMENT.

See MANAGEMENT OF CHANGE.

CHANGE MANAGEMENT FOR SOFTWARE.

See SOFTWARE MAINTENANCE.