Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least-Squares Learning

Li-Xin Wang and Jerry M. Mendel

Abstract—**In this paper, fuzzy systems are represented as series expansions of fuzzy basis functions which are algebraic superpositions of fuzzy membership functions. Using the Stone–Weierstrass theorem, we prove that linear combinations of the fuzzy basis functions are capable of uniformly approximating any real continuous function on a compact set to arbitrary accuracy. Based on the fuzzy basis function representations, an orthogonal least-squares (OLS) learning algorithm is developed for designing fuzzy systems based on given input–output pairs. Specifically, an initial fuzzy system is first constructed which has as many fuzzy basis functions as input–output pairs; then, the OLS algorithm is used to select significant fuzzy basis functions which are used to construct the final fuzzy system. The most important advantage of using fuzzy basis functions is that a linguistic fuzzy IF–THEN rule from human experts is directly related to a fuzzy basis function. Therefore, a fuzzy basis function expansion provides a natural framework for combining both numerical and linguistic information in a uniform fashion. Finally, the fuzzy basis function expansion is used to approximate a controller for the nonlinear ball and beam system, and the simulation results show that the control performance is improved by incorporating some common-sense fuzzy control rules.**

I. Introduction

FUZZY systems can be represented as three-layer feedforward networks [18], [19]. Based on this representation, a back-propagation algorithm was developed in [18], [19] to train the fuzzy system to match desired input–output pairs. Because the fuzzy system is nonlinear in the parameters, the back-propagation algorithm implements a nonlinear gradient optimization procedure; it can be trapped at a local minimum and converges slowly (although much faster than a comparable back-propagation neural network, see [18] and [19]). In this paper, we fix certain parameters of the fuzzy system such that the resulting fuzzy system is equivalent to a series expansion of basis functions that are named fuzzy basis functions. This fuzzy basis function expansion is linear in the parameters; therefore, we can use the classical Gram–Schmidt orthogonal least-squares (OLS) algorithm to determine the significant fuzzy basis functions and the remaining parameters. The OLS algorithm is a one-pass regression procedure, and is therefore much faster than the back-propagation algorithm of [18] and [19]. Also, the OLS algorithm generates a robust fuzzy system which is not sensitive to noise in its inputs.

Manuscript received August 10, 1991; revised November 18, 1991. This work was supported by the Rockwell International Science Center.

J. M. Mendel is with the Department of Electrical Engineering–Systems, University of Southern California, Los Angeles, CA 90089–2564.

L.-X. Wang is with the Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

IEEE Log Number 9201453.

The most important advantage of using fuzzy basis functions, rather than polynomials [2], [12], radial basis functions [1], [13], neural networks [15], etc., is that a linguistic fuzzy IF–THEN rule is naturally related to a fuzzy basis function. Linguistic fuzzy IF–THEN rules can often be obtained from human experts who are familiar with the system under consideration. For example, pilots can describe properties of an aircraft by linguistic fuzzy IF–THEN rules [3], [8], and experienced operators of power plants can provide operational instructions in the form of linguistic fuzzy IF–THEN rules [7]. These linguistic rules are very important, and often contain information which is not contained in the input–output pairs obtained by measuring the outputs of a system for certain test inputs, because the test inputs may not be rich enough to excite all the modes of the system. Using fuzzy basis function expansions, we can easily combine two sets of fuzzy basis functions—one generated from input–output pairs using the OLS algorithm and the other obtained from linguistic fuzzy IF–THEN rules—into a single fuzzy basis function expansion, which is therefore constructed using both numerical and linguistic information in a uniform fashion.

In Section II, we define fuzzy basis functions and prove that a fuzzy basis function expansion, like a polynomial expansion or a radial basis function expansion, is capable of approximating any real continuous function on a compact set to arbitrary accuracy. In Section III, we present the OLS algorithm for designing fuzzy basis function expansions. In Section IV the OLS algorithm is used to design a fuzzy basis expansion which is used as a controller for the nonlinear ball and beam system. Section V concludes the paper.

II. Fuzzy Systems as Fuzzy Basis Function Expansions

In this paper, we consider a fuzzy system whose basic configuration is shown in Fig. 1 [9]. There are four principal elements in such a fuzzy system: fuzzifier, fuzzy rule base, fuzzy inference engine, and defuzzifier. We consider multi-input, single-output fuzzy systems: $U \subset R^n \rightarrow R$, where U is compact [14]. A multi-output system can always be separated into a group of single-output systems.

The *fuzzifier* performs a mapping from the observed crisp input space $U \subset R^n$ to the fuzzy sets defined in U, where a *fuzzy* set [21] defined in U is characterized by a membership function $\mu_F : U \rightarrow [0, 1]$, and is labeled by a linguistic term F such as "small," "medium," "large," or "very large." The most commonly used fuzzifier is the *singleton fuzzifier,* which maps $\boldsymbol{x} \in U$ into fuzzy set A_x in U with $\mu_{A_x}(\boldsymbol{x}) = 1$ and $\mu_{A_x}(\boldsymbol{x}') = 0$ for all $\boldsymbol{x}' \in U$ with $\boldsymbol{x}' \neq \boldsymbol{x}$.

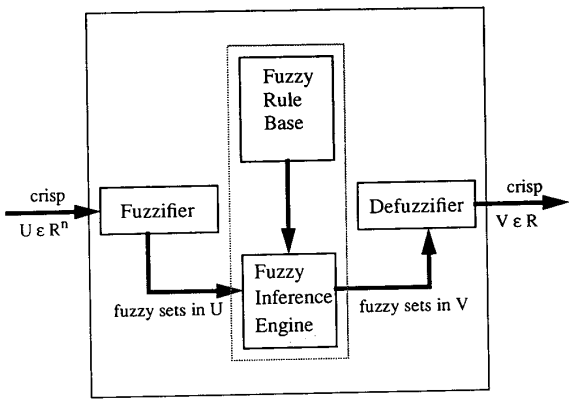

Fig. 1. Basic configuration of fuzzy systems.

The *fuzzy rule base* consists of a set of linguistic rules in the form of "IF a set of conditions are satisfied, THEN a set of consequences are inferred." In this paper, we consider the case where the fuzzy rule base consists of M rules in the following form:

$$R_j: \text{IF } x_1 \text{ is } A_1^j \text{ and } x_2 \text{ is } A_2^j \text{ and } \cdots \text{ and } x_n \text{ is } A_n^j, \text{ THEN } z \text{ is } B^j, \quad (1)$$

where $j = 1, 2, \cdots, M$, x_i $(i = 1, 2, \cdots, n)$ are the input variables to the fuzzy system, z is the output variable of the fuzzy system, and A_i^j and B^j are linguistic terms characterized by fuzzy membership functions $\mu_{A_i^j}(x_i)$ and $\mu_{B^j}(z)$, respectively. Each R_j can be viewed as a *fuzzy implication* $A_1^j \times \cdots \times A_n^j \rightarrow B^j$, which is a fuzzy set in $U \times R$ with $\mu_{A_1^j \times \cdots \times A_n^j \rightarrow B^j}(x_1', \cdots, x_n', z) = \mu_{A_1^j}(x_1') \star \cdots \star \mu_{A_n^j}(x_n') \star \mu_{B^j}(z)$ (other operations are possible, see [9] and [10]), where $\boldsymbol{x}' = (x_1', \cdots, x_n')^T \in U$, $z \in R$, and the most commonly used operations for "$\star$" are "product" and "min" [9].

The *fuzzy inference engine* is decision making logic which employs fuzzy rules from the fuzzy rule base to determine a mapping from the fuzzy sets in the input space U to the fuzzy sets in the output space R. Let A_x be an arbitrary fuzzy set in U; then each R_j of (1) determines a fuzzy set $A_x \circ R_j$ in R based on the *sup-star composition* [9]:

$$\begin{aligned}\mu_{A_x \circ R_j}(z) &= sup_{\boldsymbol{x}' \in U}[\mu_{A_x}(\boldsymbol{x}') \\ &\qquad \star \mu_{A_1^j \times \cdots \times A_n^j \rightarrow B^j}(x_1', \cdots, x_n', z)] \\ &= sup_{\boldsymbol{x}' \in U}[\mu_{A_x}(\boldsymbol{x}') \star \mu_{A_1^j}(x_1') \star \cdots \star \\ &\qquad \mu_{A_n^j}(x_n') \star \mu_{B^j}(z)]. \end{aligned} \quad (2)$$

The *defuzzifier* performs a mapping from the fuzzy sets in R to crisp points in R. The following *centroid defuzzifier* [10], which performs a mapping from the fuzzy sets $A_x \circ R_j (j = 1, 2, \cdots, M)$ in R to a crisp point $z \in R$, is the most commonly used method:

$$z = \frac{\sum_{j=1}^M \overline{z}^j \mu_{A_x \circ R_j}(\overline{z}^j)}{\sum_{j=1}^M \mu_{A_x \circ R_j}(\overline{z}^j)}, \quad (3)$$

where $\overline{z}^j$ is the point in R at which $\mu_{B^j}(z)$ achieves its maximum value (usually, we assume that $\mu_{B^j}(\overline{z}^j) = 1$).

We now consider a subset of the fuzzy systems of Fig. 1.

Definition 1: The set of fuzzy systems with *singleton fuzzifier, product inference, centroid defuzzifier,* and *Gaussian membership function* consists of all functions of the form

$$f(\boldsymbol{x}) = \frac{\sum_{j=1}^M \overline{z}^j \left(\prod_{i=1}^n \mu_{A_i^j}(x_i)\right)}{\sum_{j=1}^M \left(\prod_{i=1}^n \mu_{A_i^j}(x_i)\right)}, \quad (4)$$

where $f: U \subset R^n \rightarrow R$, $\boldsymbol{x} = (x_1, x_2, \cdots, x_n) \in U$; $\mu_{A_i^j}(x_i)$ is the *Gaussian membership function,* defined by

$$\mu_{A_i^j}(x_i) = a_i^j \exp\left[-\frac{1}{2}\left(\frac{x_i - \overline{x}_i^j}{\sigma_i^j}\right)^2\right], \quad (5)$$

where a_i^j, $\overline{x}_i^j$, and σ_i^j are real-valued parameters with $0 < a_i^j \leq 1$, and $\overline{z}^j$ is the point in the output space R at which $\mu_{B^j}(z)$ achieve its maximum value.

Clearly, (4) is obtained by substituting (2) into (3) (centroid defuzzifier), replacing the "$\star$" with "product" (product inference), and considering the fact that by using the singleton fuzzifier and assuming that $\mu_{B^j}(\overline{z}^j) = 1$, we have $\mu_{A_x \circ R_j}(\overline{z}^j) = sup_{\boldsymbol{x}' \in U}[\mu_{A_x}(\boldsymbol{x}')\mu_{A_1^j}(\boldsymbol{x}_1') \cdots \mu_{A_n^j}(\boldsymbol{x}_n')\mu_{B^j}(\overline{z}^j)] = \prod_{i=1}^n \mu_{A_i^j}(x_i)$ (because $\mu_{A_x}(\boldsymbol{x}) = 1$ and $\mu_{A_x}(\boldsymbol{x}') = 0$ for all $\boldsymbol{x}' \in U$ with $\boldsymbol{x}' \neq \boldsymbol{x}$).

If we view $\left(\prod_{i=1}^n \mu_{A_i^j}(x_i) / \sum_{j=1}^M \prod_{i=1}^n \mu_{A_i^j}(x_i)\right)$ as basis functions and $\overline{z}^j$ as constants, then $f(\boldsymbol{x})$ of (4) can be viewed as a linear combination of the basis functions.

Definition 2: Define *fuzzy basis functions* (FBF's) as

$$p_j(\boldsymbol{x}) = \frac{\prod_{i=1}^n \mu_{A_i^j}(x_i)}{\sum_{j=1}^M \prod_{i=1}^n \mu_{A_i^j}(x_i)}, \qquad j = 1, 2, \cdots, M, \quad (6)$$

where $\mu_{A_i^j}(x_i)$ are the Gaussian membership functions (5). Then, the fuzzy system (4) is equivalent to an *FBF expansion:*

$$f(\boldsymbol{x}) = \sum_{j=1}^M p_j(\boldsymbol{x})\theta_j, \quad (7)$$

where $\theta_j \in R$ are constants.

From (6) and (1) we see that an FBF corresponds to a fuzzy IF–THEN rule. Specifically, an FBF for R_j can be determined as follow. First, calculate the product of all membership functions for the linguistic terms in the IF part of R_j, and call it a *pseudo-*FBF for R_j; then, after calculating the pseudo-FBF's for all the M fuzzy IF–THEN rules, the FBF for R_j is determined by dividing the pseudo-FBF for R_j by the sum of all the M pseudo-FBF's. An FBF can either be determined based on a given linguistic rule as above or generated based on a numerical input–output pair (as shown later in the initial FBF determination, in Section III).

How does the FBF of (6) look when plotted as a function of $\boldsymbol{x}$? We now consider a simple one-dimensional example (i.e., $n = 1$). Suppose that we have four fuzzy rules in the form of (1) with $\mu_{A^j}(x) = \exp\left[-\frac{1}{2}(x - \overline{x}^j)^2\right]$, where $\overline{x}^j = -3, 1, 1, 3$ for $j = 1, 2, 3, 4$ respectively (note that the FBF's are determined based only on the IF parts of the

rules, so we do not need the $\mu_{B^j}(z)$). Therefore, $p_j(x) = \exp\left[-\frac{1}{2}\left(x-\bar{x}^j\right)^2\right] / \sum_{i=1}^{4} \exp\left[-\frac{1}{2}\left(x-\bar{x}^i\right)^2\right]$, which are plotted in Fig. 2 from left to right for $j = 1, 2, 3, 4$, respectively. From Fig. 2 we see a very interesting property of the FBF's: the $p_j(x)$'s whose centers $\bar{x}^j$ are inside the interval $[-3, 3]$ (which contains all the centers) look like Gaussian functions, whereas the $p_j(x)$'s whose centers $\bar{x}^j$ are on the boundaries of the interval $[-3, 3]$ look like sigmoidal functions [4]. It is known in the neural network literature that Gaussian radial basis functions are good at characterizing local properties, whereas neural networks with sigmoidal nonlinearities are good at characterizing global properties [11]. Our FBF's seem to combine the advantages of both the Gaussian radial basis functions and the sigmoidal neural networks. Specifically, for regions in the input space U which have sampling points (we often use the sampling points as centers of the FBF's; see Section III), the FBF's cover them with Gaussian-like functions so that higher resolution can be obtained for the FBF expansion over these regions. On the other hand, for regions in U which have no sampling points, the FBF's cover them with sigmoidal-like functions which have shown themselves to have good global properties [4], [11]. Of course, all the above are empirical observations; it seems to be a very interesting research topic to study the properties of the FBF's from a rigorous mathematical point of view.

Equation (6) defines only one kind of FBF; i.e., it defines the FBF for fuzzy systems with singleton fuzzifier, product inference, centroid defuzzifier, and Gaussian membership function. Other fuzzy systems can have other forms of FBF's; e.g., the fuzzy systems with minimum inference have an FBF in the form of (6) with product operation replaced by minimum operation. However, the basic idea remains the same, i.e., to view a fuzzy system as a linear combination of functions which are defined as FBF's. Different FBF's have different properties. Next, we show an important property of the FBF of (6).

Let Y be the set of all the FBF expansions (7) with $p_j(\boldsymbol{x})$ given by (6), and $d_\infty(f_1, f_2) = sup_{\boldsymbol{x}\in U}(|f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})|)$ be the sup-metric; then, (Y, d_∞) is a metric space [14]. The following theorem shows that (Y, d_∞) is dense in $(C[U], d_\infty)$, where $C[U]$ is the set of all real continuous functions defined on U.

Theorem: For any given real continuous function g on the compact set $U \subset R^n$ and arbitrary $\epsilon > 0$, there exists $f \in Y$ such that

$$sup_{\boldsymbol{x}\in U}|g(\boldsymbol{x}) - f(\boldsymbol{x})| < \epsilon. \tag{8}$$

A proof of this theorem is given in the Appendix. This theorem shows that the FBF expansions (7) are "universal approximators."

We can analyze (7) from two points of view. First, if we view all the parameters a_i^j, $\bar{x}_i^j$, and σ_i^j in $p_j(\boldsymbol{x})$ as free design parameters, then the FBF expansion (7) is nonlinear in the parameters. In order to specify such an FBF expansion, we must use nonlinear optimization techniques, e.g., use the back-propagation algorithm of [18] and [19]. On the other hand, we can fix all the parameters in $p_j(\boldsymbol{x})$ at the very beginning of the FBF expansion design procedure, so that the only free design parameters are θ_j; in this case, $f(\boldsymbol{x})$ of (7) is linear in the parameters. We adopt this second point of view in this paper. The advantage of this point of view is that we are now able to use some very efficient linear parameter estimation methods, e.g., the Gram–Schmidt orthogonal least squares algorithm [1], [2], to design the FBF expansions.

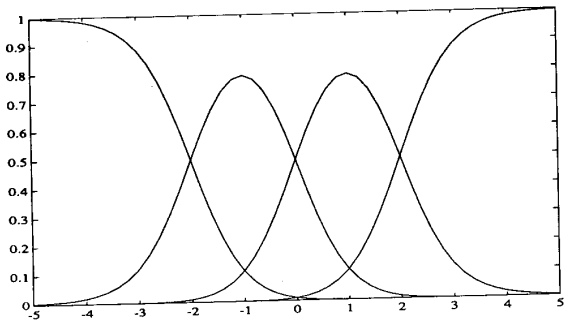

Fig. 2. An example of the fuzzy basis functions.

III. Orthogonal Least-Squares Learning Algorithm

In order to describe how the orthogonal least-squares (OLS) learning algorithm works, it is essential to view the fuzzy basis function expansion (7) as a special case of the linear regression model

$$d(t) = \sum_{j=1}^{M} p_j(t)\theta_j + e(t), \tag{9}$$

where $d(t)$ is system output, θ_j are real parameters, $p_j(t)$ are known as regressors which are fixed functions of system inputs $\boldsymbol{x}(t)$, i.e.,

$$p_j(t) = p_j(\boldsymbol{x}(t)), \tag{10}$$

and, $e(t)$ is an error signal which is assumed to be uncorrelated with the regressors. Suppose that we are given N input–output pairs: $(\boldsymbol{x}^0(t), d^0(t))$, $t = 1, 2, \cdots, N$. Our task is to design an FBF expansion $f(\boldsymbol{x})$ such that some error function between $f(\boldsymbol{x}^0(t))$ and $d^0(t)$ is minimized.

In order to present the OLS algorithm, we arrange (9) from $t = 1$ to N in the following matrix form:

$$\boldsymbol{d} = P\boldsymbol{\theta} + \boldsymbol{e}, \tag{11}$$

where $\boldsymbol{d} = [d(1), \cdots, d(N)]^T$, $P = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_M]$ with $\boldsymbol{p}_i = [p_i(1), \cdots, p_i(N)]^T$, $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_M]^T$, and $\boldsymbol{e} = [e(1), \cdots, e(N)]^T$. The OLS algorithm transforms the set of $\boldsymbol{p}_i$ into a set of orthogonal basis vectors and uses only the significant basis vectors to form the final FBF expansion. In order to perform the OLS procedure, we first need to fix the parameters a_i^j, $\bar{x}_i^j$, and σ_i^j in the FBF $p_j(\boldsymbol{x})$ based on the input–output pairs. We propose the following scheme:

Initial FBF Determination

Choose N initial $p_j(\boldsymbol{x})$'s in the form of (6) (for this case, the M in (6) equals N), with the parameters determined as follows: $a_i^j = 1$, $\bar{x}_i^j = x_i^0(j)$, and $\sigma_i^j = [\max(x_i^0(j), j = 1, 2, \cdots, N) - \min(x_i^0(j), j = 1, 2, \cdots, N)]/M_s$, where $i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, N$, and M_s is the number of FBF's in the final FBF expansion. We assume that M_s is given based on practical constraints; in general, $M_s \ll N$.

We choose $a_i^j = 1$ because $\mu_{A_i^j}(x_i)$ are fuzzy membership functions which can be assumed to achieve unity membership value at some center $\bar{x}_i^j$. We choose the centers $\bar{x}_i^j$ to be the input points in the given input–output pairs. Finally, the above choice of σ_i^j should make the final FBF's "uniformly" cover the input region spanned by the input points in the given input–output pairs.

Next, we use the OLS algorithm, similar to that in [1] and [2] (it is based on the classical Gram–Schmidt orthogonalization procedure), to select the significant FBF's from the N FBF's determined by the initial FBF determination method:

At the first step, for $1 \le i \le N$, compute

$$\boldsymbol{w}_1^{(i)} = \boldsymbol{p}_i \qquad g_1^{(i)} = \left(\boldsymbol{w}_1^{(i)}\right)^T \boldsymbol{d}^0 \Big/ \left(\left(\boldsymbol{w}_1^{(i)}\right)^T \boldsymbol{w}_1^{(i)}\right) \tag{12}$$

$$[\mathrm{err}]_1^{(i)} = \left(g_1^{(i)}\right)^2 \left(\boldsymbol{w}_1^{(i)}\right)^T \boldsymbol{w}_1^{(i)} \Big/ (\boldsymbol{d}^{0T}\boldsymbol{d}^0), \tag{13}$$

where $\boldsymbol{p}_i = \left[p_i(\boldsymbol{x}^0(1)), \cdots, p_i(\boldsymbol{x}^0(N))\right]^T$, and $p_i(\boldsymbol{x}^0(t))$ are given by the initial FBF determination method. Find

$$[\mathrm{err}]_1^{(i_1)} = \max\left([\mathrm{err}]_1^{(i)}, 1 \le i \le N\right) \tag{14}$$

and select

$$\boldsymbol{w}_1 = \boldsymbol{w}_1^{(i_1)} = \boldsymbol{p}_{i_1} \qquad g_1 = g_1^{(i_1)}. \tag{15}$$

At the kth step where $2 \le k \le M_s$, for $1 \le i \le N$, $i \ne i_1, \cdots, i \ne i_{k-1}$, compute

$$\alpha_{jk}^{(i)} = \boldsymbol{w}_j^T \boldsymbol{p}_i / (\boldsymbol{w}_j^T \boldsymbol{w}_j), \qquad 1 \le j < k \tag{16}$$

$$\boldsymbol{w}_k^{(i)} = \boldsymbol{p}_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} \boldsymbol{w}_j$$

$$g_k^{(i)} = \left(\boldsymbol{w}_k^{(i)}\right)^T \boldsymbol{d}^0 \Big/ \left(\left(\boldsymbol{w}_k^{(i)}\right)^T \boldsymbol{w}_k^{(i)}\right) \tag{17}$$

$$[\mathrm{err}]_k^{(i)} = \left(g_k^{(i)}\right)^2 \left(\boldsymbol{w}_k^{(i)}\right)^T \boldsymbol{w}_k^{(i)} / (\boldsymbol{d}^{0T}\boldsymbol{d}^0). \tag{18}$$

Find

$$[\mathrm{err}]_k^{(i_k)} = \max\left([\mathrm{err}]_k^{(i)}, 1 \le i \le N, i \ne i_1, \cdots, i \ne i_{k-1}\right) \tag{19}$$

and select

$$\boldsymbol{w}_k = \boldsymbol{w}_k^{(i_k)} \qquad g_k = g_k^{(i_k)}. \tag{20}$$

Solve the triangular system

$$A^{(M_s)}\boldsymbol{\theta}^{(M_s)} = \boldsymbol{g}^{(M_s)}, \tag{21}$$

where

$$A^{(M_s)} = \begin{bmatrix} 1 & \alpha_{12}^{(i_2)} & \alpha_{13}^{(i_3)} & \cdots & \alpha_{1M_s}^{(i_{M_s})} \\ 0 & 1 & \alpha_{23}^{(i_3)} & \cdots & \alpha_{2M_s}^{(i_{M_s})} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & \alpha_{M_s-1,M_s}^{(i_{M_s})} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{22}$$

$$\boldsymbol{g}^{(M_s)} = [g_1, \cdots, g_{M_s}]^T \qquad \boldsymbol{\theta}^{(M_s)} = \left[\theta_1^{(M_s)}, \cdots, \theta_{M_s}^{(M_s)}\right]^T. \tag{23}$$

The final FBF expansion is

$$f(\boldsymbol{x}) = \sum_{j=1}^{M_s} p_{i_j}(\boldsymbol{x}) \theta_j^{(M_s)}, \tag{24}$$

where $p_{i_j}(\boldsymbol{x})$ make up the subset of the FBF's determined by the initial FBF determination method with i_j determined by the above steps.

Some comments on this OLS algorithm are now in order. For in-depth discussions on the OLS algorithm, see [1] and [2].

1) The purpose of the original Gram–Schmidt OLS algorithm is to perform an orthogonal decomposition for P, i.e., $P = WA$, where W is an orthogonal matrix and A is an upper-triangular matrix with unity diagonal elements. Substituting $P = WA$ into (11), we have that $\boldsymbol{d} = WA\boldsymbol{\theta} + \boldsymbol{e} = W\boldsymbol{g} + \boldsymbol{e}$ where $\boldsymbol{g} = A\boldsymbol{\theta}$ has the same meaning as used in our OLS algorithm, and the $\alpha_{jk}^{(i)}$ in our OLS algorithm correspond to the elements of A. Our OLS algorithm does not complete the decomposition of $P = WA$, but only selects some domain columns from P.

2) The $[\mathrm{err}]_k^{(i)} = \left(g_k^{(i)}\right)^2 \left(\boldsymbol{w}_k^{(i)}\right)^T \boldsymbol{w}_k^{(i)} / \left(\boldsymbol{d}^{0T}\boldsymbol{d}^0\right)$ represents the error reduction ratio caused by $\boldsymbol{w}_k^{(i)}$ [1], [2]. Hence our OLS algorithm selects significant FBF's based on their error reduction ratio; i.e., the FBF's with largest error reduction ratios are retained in the final FBF expansion.

IV. Control of a Nonlinear Ball and Beam System Using FBF Expansion

In this section, we use the OLS algorithm to design an FBF expansion to approximate a controller for a nonlinear ball and beam system [5]. Our purpose is to use the FBF expansion as a controller to regulate the system to the origin from a certain range of initial conditions. We first use the input–output linearization algorithm of [5] to generate a set of state-control pairs for randomly sampled points in a certain region of the state space. Then we view these state-control pairs as the input–output pairs in Section III and use the OLS algorithm to determine an FBF expansion which is used as the controller for the ball and beam system with initial conditions arbitrarily chosen in the sampled state space. In other words, we use the controller of [5] to generate a lookup table of state-control pairs, and then use the FBF expansion to interpolate these pairs to form the final controller. For many practical

problems, this kind of lookup table of state-control pairs can be provided by human experts or collected from past successful control executions.

The ball and beam system, which can be found in many undergraduate control laboratories, is shown in Fig. 3. The beam is made to rotate in a vertical plane by applying a torque at the center of rotation and the ball is free to roll along the beam. We require that the ball remain in contact with the beam. Let $\boldsymbol{x} = \left(r, \dot{r}, \theta, \dot{\theta}\right)^T$ be the state of the system, and $y = r$ be the output of the system. Then, from [5], the system can be represented by the state-space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - G \sin x_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u \tag{25}$$

$$y = x_1, \tag{26}$$

where the control u is the acceleration of θ, and the parameters B and G are defined in [5]. The purpose of control is to determine $u(\boldsymbol{x})$ such that the closed-loop system output y will converge to zero from certain initial conditions.

The input–output linearization algorithm of [5] determines the control law $u(\boldsymbol{x})$ as follows: for state $\boldsymbol{x}$, compute $v(\boldsymbol{x}) = -\alpha_3\phi_4(\boldsymbol{x}) - \alpha_2\phi_3(\boldsymbol{x}) - \alpha_1\phi_2(\boldsymbol{x}) - \alpha_0\phi_1(\boldsymbol{x})$, where $\phi_1(\boldsymbol{x}) = x_1$, $\phi_2(\boldsymbol{x}) = x_2$, $\phi_3(\boldsymbol{x}) = -BG \sin x_3$, $\phi_4(\boldsymbol{x}) = -BGx_4 \cos x_3$, and the α_i are chosen so that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is a Hurwitz polynomial. Compute $a(\boldsymbol{x}) = -BG \cos x_3$ and $b(\boldsymbol{x}) = BGx_4^2 \sin x_3$; then $u(\boldsymbol{x}) = (v(\boldsymbol{x}) - b(\boldsymbol{x}))/a(\boldsymbol{x})$.

In our simulations, we used the $u(\boldsymbol{x}) = (v(\boldsymbol{x}) - b(\boldsymbol{x}))/a(\boldsymbol{x})$ to generate $N(\boldsymbol{x}, u)$ pairs with $\boldsymbol{x}$ randomly sampled in the region $U = [-5, 5] \times [-2, 2] \times [-\pi/4, \pi/4] \times [-0.8, 0.8]$. We simulated three cases. For case 1, $N = 200$, $M_s = 20$, and the final FBF expansion $f(\boldsymbol{x})$ of (24) was used as the control u in (25). In case 2, $N = 40$, $M_s = 20$, and the final FBF expansion $f(\boldsymbol{x})$ of (24) was used as the control u in (25). In case 3, $N = 40$, $M_s = 20$, and the control

$$u(\boldsymbol{x}) = \frac{1}{2}\left[f(\boldsymbol{x}) + f^L(\boldsymbol{x})\right], \tag{27}$$

where $f(\boldsymbol{x})$ is given by (24), and $f^L(\boldsymbol{x})$ is a *linguistic controller*. This controller is in the form of (4) and is determined based on the following four common-sense linguistic control rules:

R_1^L: IF x_1 is "positive" and x_2 is "near zero" and x_3 is "positive" and x_4 is "near zero," THEN u is "negative." (28)

R_2^L: IF x_1 is "positive" and x_2 is "near zero" and x_3 is "negative" and x_4 is "near zero," THEN u is "positive big." (29)

R_3^L: IF x_1 is "negative" and x_2 is "near zero" and x_3 is "positive" and x_4 is "near zero," THEN u is "negative big." (30)

R_4^L: IF x_1 is "negative" and x_2 is "near zero" and x_3 is "negative" and x_4 is "near zero," THEN u is "positive." (31)

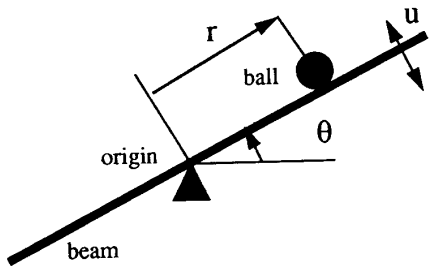

Fig. 3. The ball and beam system.

Here the "positive" for x_1 is a fuzzy set $P1$ with membership function $\mu_{P1}(x_1) = \exp\left[-\frac{1}{2}\left(\frac{\min(x_1-4,0)}{4}\right)^2\right]$; the "negative" for x_1 is a fuzzy set $N1$ with membership function $\mu_{N1}(x_1) = \exp\left[-\frac{1}{2}\left(\frac{\max(x_1+4,0)}{4}\right)^2\right]$; the "near zero" for both x_2 and x_4 is a fuzzy set ZO with $\mu_{NO}(x) = \exp\left[-\frac{1}{2}x^2\right]$; the "positive" for x_3 is a fuzzy set $P3$ with $\mu_{P3}(x_3) = \exp\left[-\frac{1}{2}\left(\frac{\min(x_3-\pi/4,0)}{\pi/4}\right)^2\right]$; the "negative" for x_3 is a fuzzy net $N3$ with $\mu_{N3}(x_3) = \exp\left[-\frac{1}{2}\left(\frac{\max(x_3+\pi/4,0)}{\pi/4}\right)^2\right]$; the "positive" for u is a fuzzy set Pu with $\mu_{Pu}(u) = \exp\left[-\frac{1}{2}(u-0.1)^2\right]$; the "negative" for u is a fuzzy set Nu with $\mu_{Nu}(u) = \exp\left[-\frac{1}{2}(u+0.1)^2\right]$; the "positive big" for u is a fuzzy set PBu with $\mu_{PBu}(u) = \exp\left[-\frac{1}{2}(u-0.4)^2\right]$; and the "negative big" for u is a fuzzy set NBu with $\mu_{NBu}(u) = \exp\left[-\frac{1}{2}(u+0.4)^2\right]$. The above membership functions for the IF parts of R_1^L–R_4^L were determined based on the meaning of the linguistic terms; the parameters of the THEN part membership functions were determined by common sense and trial and error. The detailed formula of $f^L(\boldsymbol{x})$ can be easily obtained based on (4) and above membership functions.

Clearly, R_1^L–R_4^L are determined based on our common sense of how to control the ball to stay at the origin when the ball is in certain regions. Take R_1^L as an example. If the ball stays at its position depicted in Fig. 3 (which just corresponds to the IF part of R_1^L), then we should move the beam downwards to reduce θ (but not a lot), which is equivalent to saying "u is 'negative'," because the control u equals the acceleration of θ (see (25)). Although these common-sense control rules are not precise, the control performance will, as we show below, be greatly improved by incorporating them into the controller (27).

We simulated each of the three cases for four initial conditions: $\boldsymbol{x}(0) = [2.4, -0.1, 0.6, 0.1]^T$, $[1.6, 0.05, -0.6, -0.05]^T$, $[-1.6, -0.05, 0.6, 0.05]^T$, and $[-2.4, 0.1, -0.6, -0.1]^T$, which were arbitrarily chosen in $U = [-5, 5] \times [-2, 2] \times [-\pi/4, \pi/4] \times [-0.8, 0.8]$. Figs. 4–6 depict the output y of the closed-loop system for cases 1–3, respectively. In the simulations, we solved the differential equations using the MATLAB command "ode23," which uses the second/third order Runge–Kutta method.

Some comments on these simulation results are now in order:

a) The fuzzy controller in case 1 gave the best overall performance; this suggests that given a sufficient number of state-control pairs, the OLS algorithm can determine a successful FBF expansion controller.

b) The fuzzy controller in case 2 could regulate the ball

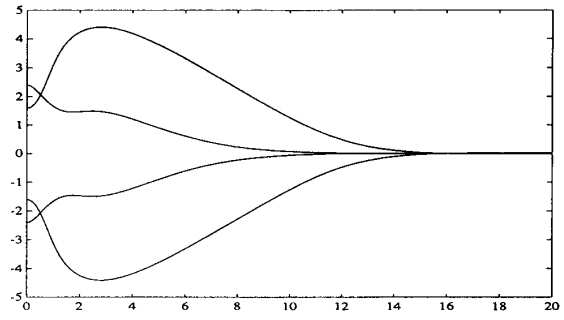

Fig. 4. Outputs of the closed-loop ball and beam system for case 1 and four initial conditions.

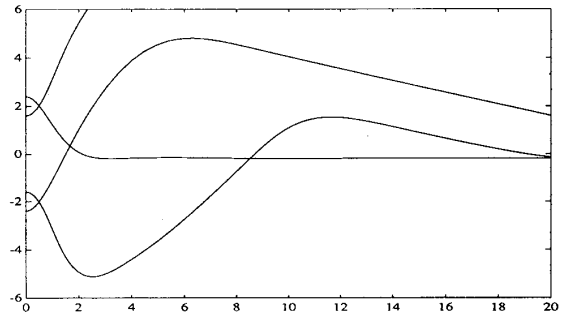

Fig. 5. Outputs of the closed-loop ball and beam system for case 2 and four initial conditions.

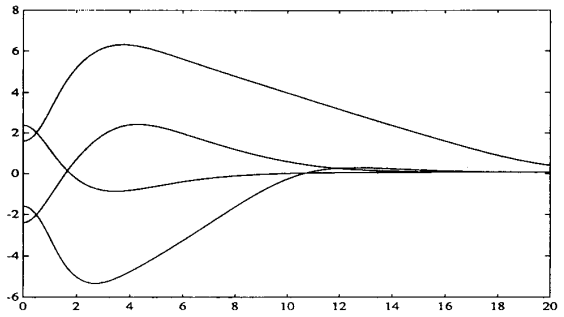

Fig. 6. Outputs of the closed-loop ball and beam system for case 3 and four initial conditions.

to the origin for some initial conditions, but the closed-loop system was unstable for some initial condition; this suggests that sufficient sampling of the state space is important for the "pure numerical" fuzzy controller to be successful.

c) Using the same small number of state-control pairs but adding the fuzzy control rules (28)–(31), the fuzzy controller in case 3 showed much better performance than the fuzzy controller in case 2; i.e., control performance was greatly improved by incorporating (in the sense of (27)) the linguistic fuzzy control rules into the controller.

We also simulated two extreme cases: (i) using the original controller of [5] and (ii) using only the pure linguistic controller $f^L(\boldsymbol{x})$ based on R_1^L–R_4^L, for the same initial conditions as in cases 1–3. Figs. 7 and 8 show the output of the closed-loop system for the two cases (i) and (ii), respectively. Comparing Fig. 7 with Figs. 4–6, we see that the original controller of [5] gave the best performance; this is to be expected because we used the FBF expansions to approximate this controller. But the controller of [5] requires the mathematical model of the system, whereas our method does not. Fig. 8 shows that if we use the FBF expansion controller based only on the four linguistic rules (28)–(31), the closed-loop system is unstable; i.e., a pure fuzzy logic controller with only these four linguistic rules is not sufficient to control the system.

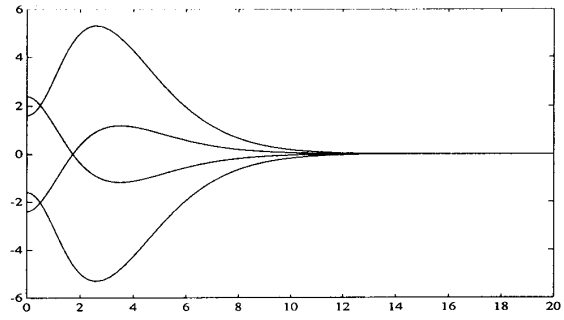

Fig. 7. Outputs of the closed-loop ball and beam system using the input–output linearization algorithm of [5] and four initial conditions.

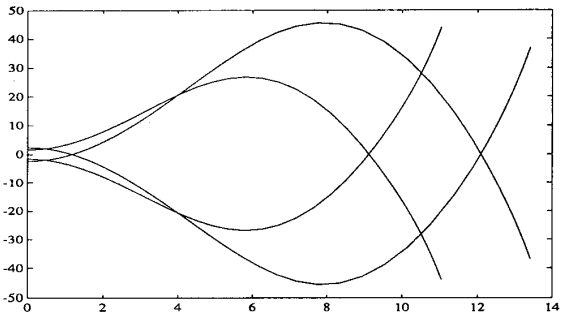

Fig. 8. Outputs of the closed-loop ball and beam system using the pure fuzzy logic controller based on the four linguistic rules (28)–(31) and four initial conditions.

V. Conclusions

In this paper we have 1) showed that fuzzy systems can be represented as linear combinations of fuzzy basis functions; 2) proved that linear combinations of the fuzzy basis functions are capable of uniformly approximating any real continuous function on a compact set to arbitrary accuracy, i.e., they are universal approximators; and 3) developed an orthogonal least-squares algorithm to select the significant fuzzy basis functions. Through a simple example we illustrated that the fuzzy basis functions whose centers are inside the sampling region look Gaussian, whereas the fuzzy basis functions whose centers are on the boundaries of the sampling regions look sigmoidal. The most important advantage of the fuzzy basis functions is that a linguistic fuzzy IF–THEN rule is directly related to a fuzzy basis function, so that the fuzzy basis

function expansion provides a natural framework to combine both numerical information (in the form of input–output pairs) and linguistic information (in the form of fuzzy IF–THEN rules) in a uniform fashion. We showed an example of how to combine the fuzzy basis functions generated from a numerical state-control table and the fuzzy basis functions generated from some common-sense linguistic fuzzy control rules, to form a controller for the nonlinear ball and beam system. The simulation results showed that the control performance was improved by incorporating these linguistic fuzzy control rules.

APPENDIX

We use the following Stone–Weierstrass theorem to prove the theorem.

Stone–Weierstrass Theorem [14]: Let Z be a set of real continuous functions on a compact set U. If 1) Z is an *algebra,* i.e., the set Z is closed under addition, multiplication, and scalar multiplication, 2) Z *separates points on* U, i.e., for every $\boldsymbol{x}, \boldsymbol{y} \in U$, $\boldsymbol{x} \neq \boldsymbol{y}$, there exists $f \in Z$ such that $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$, and 3) Z *vanishes at no point of* U, i.e., for each $\boldsymbol{x} \in U$ there exists $f \in Z$ such that $f(\boldsymbol{x}) \neq 0$, then the uniform closure of Z consists of all real continuous functions on U; i.e., (Z, d_∞) is dense in $(C[U], d_\infty)$.

Proof: First, we prove that (Y, d_∞) is an algebra. Let $f_1, f_2 \in Y$, so that we can write them as

$$f_1(\boldsymbol{x}) = \frac{\sum_{j=1}^{K1} \left(\bar{z}1^j \prod_{i=1}^{n} \mu_{A1_i^j}(x_i)\right)}{\sum_{j=1}^{K1} \left(\prod_{i=1}^{n} \mu_{A1_i^j}(x_i)\right)}, \tag{A1}$$

$$f_2(\boldsymbol{x}) = \frac{\sum_{j=1}^{K2} \left(\bar{z}2^j \prod_{i=1}^{n} \mu_{A2_i^j}(x_i)\right)}{\sum_{j=1}^{K2} \left(\prod_{i=1}^{n} \mu_{A2_i^j}(x_i)\right)}; \tag{A2}$$

we therefore have (A3), shown at the bottom of the page. Since $\mu_{A1_i^{j1}}$ and $\mu_{A2_i^{j2}}$ are Gaussian in form, their product $\mu_{A1_i^{j1}}\mu_{A2_i^{j2}}$ is also Gaussian in form (this can be verified by straightforward algebraic operations); hence, (A3) is in the same form as (4), so that $f_1 + f_2 \in Y$. Similarly, we have (A4), also shown at the bottom of the page, which is also in the same form of (4); hence, $f_1 f_2 \in Y$. Finally, for arbitrary $c \in R$,

$$cf_1(\boldsymbol{x}) = \frac{\sum_{j=1}^{K1} \left(c\bar{z}1^j\right)\left(\prod_{i=1}^{n} \mu_{A1_i^j}(x_i)\right)}{\sum_{j=1}^{K1} \left(\prod_{i=1}^{n} \mu_{A1_i^j}(x_i)\right)}, \tag{A5}$$

which is again in the form of (4); hence, $cf_1 \in Y$. Therefore, (Y, d_∞) is an algebra.

Next, we prove that (Y, d_∞) separates points on U. We prove this by constructing a required f; i.e., we specify $f \in Y$ such that $f(\boldsymbol{x}^0) \neq f(\boldsymbol{y}^0)$ for arbitrarily given $\boldsymbol{x}^0, \boldsymbol{y}^0 \in U$ with $\boldsymbol{x}^0 \neq \boldsymbol{y}^0$. We choose two fuzzy rules in the form of (1) for the fuzzy rule base (i.e., $M = 2$). Let $\boldsymbol{x}^0 = \left(x_1^0, x_2^0, \cdots, x_n^0\right)$ and $\boldsymbol{y}^0 = \left(y_1^0, y_2^0, \cdots, y_n^0\right)$. If $x_i^0 \neq y_i^0$, we define two fuzzy sets $(A_i^1, \mu_{A_i^1})$ and $(A_i^2, \mu_{A_i^2})$ with

$$\mu_{A_i^1}(x_i) = \exp\left[-\frac{\left(x_i - x_i^0\right)^2}{2}\right] \tag{A6}$$

$$\mu_{A_i^2}(x_i) = \exp\left[-\frac{\left(x_i - y_i^0\right)^2}{2}\right]. \tag{A7}$$

If $x_i^0 = y_i^0$, then $A_i^1 = A_i^2$ and $\mu_{A_i^1} = \mu_{A_i^2}$; i.e., only one fuzzy set is defined. We define two fuzzy sets (B^1, μ_{B^1}) and (B^2, μ_{B^2}) with

$$\mu_{B^j}(z) = \exp\left[-\frac{\left(z - \bar{z}^j\right)^2}{2}\right], \tag{A8}$$

where $j = 1, 2$, and $\bar{z}^j$ will be specified later. Now we have specified all the design parameters except $\bar{z}^j (j = 1, 2)$, i.e., we have already obtained a function f which is in the form of (4) with $M = 2$ and $\mu_{A_i^j}$ given by (A6) and (A7). With this f, we have

$$\begin{aligned} f(\boldsymbol{x}^0) &= \frac{\bar{z}^1 + \bar{z}^2 \prod_{i=1}^{n} \exp\left[-\left(x_i^0 - y_i^0\right)^2/2\right]}{1 + \prod_{i=1}^{n} \exp\left[-\left(x_i^0 - y_i^0\right)^2/2\right]} \\ &= \alpha\bar{z}^1 + (1-\alpha)\bar{z}^2 \end{aligned} \tag{A9}$$

$$\begin{aligned} f(\boldsymbol{y}^0) &= \frac{\bar{z}^2 + \bar{z}^1 \prod_{i=1}^{n} \exp\left[-\left(x_i^0 - y_i^0\right)^2/2\right]}{1 + \prod_{i=1}^{n} \exp\left[-\left(x_i^0 - y_i^0\right)^2/2\right]} \\ &= \alpha\bar{z}^2 + (1-\alpha)\bar{z}^1 \end{aligned} \tag{A10}$$

where

$$\alpha = \frac{1}{1 + \prod_{i=1}^{n} \exp\left[-\left(x_i^0 - y_i^0\right)^2/2\right]}. \tag{A11}$$

$$f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) = \frac{\sum_{j1=1}^{K1} \sum_{j2=1}^{K2} \left(\bar{z}1^{j1} + \bar{z}2^{j2}\right)\left(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i)\right)}{\sum_{j1=1}^{K1} \sum_{j2=1}^{K2} \left(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i)\right)}. \tag{A3}$$

$$f_1(\boldsymbol{x}) f_2(\boldsymbol{x}) = \frac{\sum_{j1=1}^{K1} \sum_{j2=1}^{K2} \left(\bar{z}1^{j1}\bar{z}2^{j2}\right)\left(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i)\right)}{\sum_{j1=1}^{K1} \sum_{j2=1}^{K2} \left(\prod_{i=1}^{n} \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i)\right)}, \tag{A4}$$

Since $\boldsymbol{x}^0 \neq \boldsymbol{y}^0$, there must be some i such that $\boldsymbol{x}_i^0 \neq \boldsymbol{y}_i^0$, hence, we have $\prod_{i=1}^{n} \exp[-(x_i^0 - y_i^0)^2/2] \neq 1$, or, $\alpha \neq 1-\alpha$. If we choose $\overline{z}^1 = 0$ and $\overline{z}^2 = 1$, then $f(\boldsymbol{x}^0) = 1-\alpha \neq \alpha = f(\boldsymbol{y}^0)$. Therefore, (Y, d_∞) separates points on U.

Finally, we prove that (Y, d_∞) vanishes at no point of U. By observing (4) and (5), we simply choose all $\overline{z}^j > 0 (j = 1, 2, \cdots, M)$; i.e., any $f \in Y$ with $\overline{z}^j > 0$ serves as the required f.

In summary, using the Stone–Weierstrass theorem and the fact that Y is a set of real continuous functions on U (see (4) and (5)), we have proven the theorem. Q.E.D.

ACKNOWLEDGMENT

The authors would like to thank Dr. J. Hauser for furnishing the ball and beam example, and the reviewers for constructive comments.

REFERENCES

[1] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks,* vol. 2, pp. 302–309, Mar. 1991.

[2] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *Int. J. Contr.*, vol. 50, no. 5, pp. 1873–1896, 1989.

[3] S. Chiu, S. Chand, D. Moore, and A. Chaudhary, "Fuzzy logic for control of roll and moment for a flexible wing aircraft," *IEEE Control Systems Magazine,* vol. 11, no. 4, pp. 42–48, 1991.

[4] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, and Systems,* vol. 2, pp. 303–314, 1989.

[5] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear control via approximate input–output linearization: The ball and beam example," *IEEE Trans. Automat. Contr.*, to be published.

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

[7] M. Kinoshita, T. Fukuzaki, T. Satoh, and M. Miyake, "An automatic operation method for control rods in BWR plants," in *Proc. Specialists' Meeting on In-Core Instrumentation and Reactor Core Assessment* (Cadarache, France), 1988.

[8] L. I. Larkin, "A fuzzy logic controller for aircraft flight control," in *Industrial Applications of Fuzzy Control,* M. Sugeno, Ed. Amsterdam: North-Holland, 1985, pp. 87–104.

[9] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller, part I," *IEEE Trans. Syst., Man, Cybern.,* vol. 20, no. 2, pp. 404–418, 1990.

[10] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller, part II," *IEEE Trans. Syst., Man, Cybern.,* vol. 20, no. 2, pp. 419–435, 1990.

[11] R. Lippmann, "A critical overview of neural network pattern classifiers," in *Proc. 1991 IEEE Workshop on Neural Networks for Signal Processing* (Princeton, NJ), 1991, pp. 266–275.

[12] M. J. D. Powell, *Approximation Theory and Methods.* Cambridge, U.K.: Cambridge University Press, 1981.

[13] M. J. D. Powell, "Radial basis functions for multivariable interpolation: A review," in *Algorithms for Approximation,* J. C. Mason and M. G. Cox, Eds., Oxford, 1987, pp. 143–167.

[14] W. Rudin, *Principles of Mathematical Analysis.* New York: McGraw-Hill, 1976.

[15] D. E. Rumelhart and J. L. McCleland, Eds., *Parallel Distributed Processing I, II.* Cambridge, MA: MIT Press, 1986.

[16] L. X. Wang, "Fuzzy systems are universal approximators," in *Proc. IEEE 1992 Int. Conf. Fuzzy Systems* (San Diego, CA), Mar. 1992, pp. 1163–1170.

[17] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," in *Proc. 6th Int. Symp. Intelligent Control* (Washington, DC), 1991, pp. 263–268; also *IEEE Trans. Syst., Man, Cybern.,* to be published.

[18] L. X. Wang and J. M. Mendel, "Analysis and design of fuzzy logic controller," USC SIPI Rep. 184, 1991.

[19] L. X. Wang and J. M. Mendel, "Back-propagation fuzzy systems as nonlinear dynamic system identifiers," in *Proc. IEEE 1992 Int. Conf. Fuzzy Systems* (San Diego, CA) Mar. 1992, pp. 1409–1418.

[20] L. X. Wang and J. M. Mendel, "Fuzzy adaptive filters, with application to nonlinear channel equalization," submitted to *IEEE Trans. Fuzzy Systems,* 1992.

[21] L. A. Zadeh, "Fuzzy sets," *Informat. Control,* vol. 8, pp. 338–353, 1965.