Generating Fuzzy Rules by Learning from Examples

Li-Xin Wang, *Fellow, IEEE*, and Jerry M. Mendel, *Fellow, IEEE*

***Abstract*— A general method is developed to generate fuzzy rules from numerical data. This new method consists of five steps: Step 1 divides the input and output spaces of the given numerical data into fuzzy regions; Step 2 generates fuzzy rules from the given data; Step 3 assigns a degree of each of the generated rules for the purpose of resolving conflicts among the generated rules; Step 4 creates a combined fuzzy rule base based on both the generated rules and linguistic rules of human experts; and, Step 5 determines a mapping from input space to output space based on the combined fuzzy rule base using a defuzzifying procedure. The mapping is proved to be capable of approximating any real continuous function on a compact set to arbitrary accuracy. Applications to truck backer-upper control and time series prediction problems are presented. For the truck control problem, the performance of this new method is compared with a neural network controller and a pure limited-rule fuzzy controller; the new method shows the best performance. For the time series prediction problem, results are compared by using the new method and a neural network predictor for the Mackey–Glass chaotic time series.**

I. Introduction

FOR MOST real-world control and signal processing problems, the information concerning design, evaluation, realization, etc., can be classified into two kinds: numerical information obtained from sensor measurements, and, linguistic information obtained from human experts. Most current intelligent control and signal processing approaches are heuristic in nature, i.e., they combine some standard control or signal processing methods with expert systems in an *ad hoc* way for a specific problem; simulations are then performed to show that the new approaches work well for the specific problem. This kind of approach has two weakpoints: 1) it is quite problem dependent, i.e., a method may work well for one problem but is not suited for another problem; and, 2) there is no common framework for modeling and representing different aspects of control or signal processing strategies, which makes theoretical analyses for these approaches very difficult. In this paper, we propose a general method for combining both numerical and linguistic information into a common framework—a fuzzy rule base.

Suppose we have the following problem: there is a complex control system in which a human controller is an essential part; the environment facing this human controller is so complicated that no mathematical model exists for it, or, the mathematical model is strongly nonlinear so that a design method does not exist. The task here is to design a control system to replace the human controller (see Fig. 1).

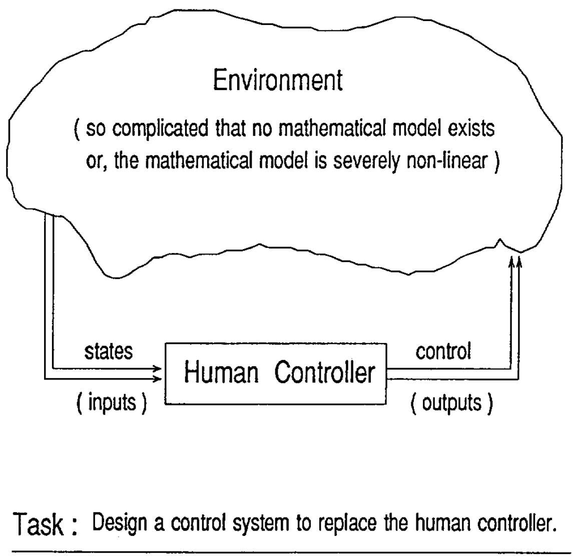

Environment
(so complicated that no mathematical model exists or, the mathematical model is severely non-linear)
states
(inputs)
Human Controller
control
(outputs)
Task : Design a control system to replace the human controller.

Fig. 1. A practical problem: Design a control system to replace the human controller.

To design such a control system, we first need to see what information is available. We assume that there is no mathematical model, i.e., we consider a model-free design problem. Since there already is a human controller who is successfully controlling the system, there are two kinds of information available to us: 1) the experience of the human controller; and, 2) sampled input–output (state-control) pairs that are recorded from successful control by the human controller. The experience of the human controller is usually expressed as some linguistic "IF-THEN" rules that state in what situation(s) which action(s) should be taken. The sampled input–output pairs are some numerical data that give the specific values of the inputs and the corresponding successful outputs.

Each of the two kinds of information alone is usually incomplete. Although the system is successfully controlled by a human controller, some information will be lost when human controllers express their experience by linguistic rules. Consequently, linguistic rules alone are usually not enough for designing a successful control system. On the other hand, the information from sampled input–output data pairs is usually also not enough for a successful design, because the past

Manuscript received March 3, 1991, revised April 20, 1992.

L. X. Wang was with the Signal and Image Processing Institute, Department of Electrical Engineering Systems, University of Southern California, Los Angeles, CA 90089-2564 and is now with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

J. M. Mendel is with the Signal and Image Processing Institute, Department of Electrical Engineering Systems, University of Southern California, Los Angeles, CA 90089-2564.

IEEE Log Number 9202126.

operations usually cannot cover all the situations the control system will face. If expert linguistic rules and numerical data pairs are the only information we can get for such a control system design, the most interesting case for us is when the combination of these two kinds of information is sufficient for a successful design.

Fuzzy control is an effective approach to utilizing linguistic rules [3], [4], whereas neural control is suited for using numerical data pairs (i.e., desired input–output pairs) [1], [4]. Present fuzzy controllers only use linguistic rules, whereas present neural controllers only use numerical data pairs. This leads to the following question: "Is it possible to develop a general approach that combines both kinds of information into a common framework, and uses both information, simultaneously and cooperatively, to solve the control design or similar problems?" In this paper, we develop such a general approach.

The key ideas of our new approach are to generate fuzzy rules from numerical data pairs, collect these fuzzy rules and the linguistic fuzzy rules into a common fuzzy rule base, and, finally, design a control or signal processing system based on this combined fuzzy rule base.

In Section II, we propose a five step procedure for generating fuzzy rules from numerical data pairs and show how to use these fuzzy rules to obtain a mapping from input space to output space. Step 1 divides the input and output spaces into fuzzy regions; Step 2 generates fuzzy rules from given desired input–output data pairs; Step 3 assigns a degree to each generated rule; Step 4 forms the combined fuzzy rule base; and, Step 5 presents a defuzzifying procedure for obtaining a mapping based on the combined fuzzy rule base. In Section III, we prove that the resulting mapping is capable of approximating any nonlinear continuous function on a compact set to arbitrary accuracy using the well-known Stone–Weierstrass theorem in analysis [5]. In Section IV, we apply our new method to a truck backer-upper control problem [1], [4]. We compare this new approach with pure neural and fuzzy approaches. The power of our new approach becomes apparent when it is used in the case where neither linguistic fuzzy rules nor input–output pairs are sufficient to successfully control the truck to a desired position, but the combination of both is sufficient. In Section V, we show that our new method can be used for time-series prediction; and, we use it to predict the Mackey–Glass chaotic time series, and compare the results with those obtained using a neural network predictor. Conclusions are given in Section VI.

II. Generating Fuzzy Rules from Numerical Data

Suppose we are given a set of desired input–output data pairs:

$$(x_1^{(1)}, x_2^{(1)}; y^{(1)}), (x_1^{(2)}, x_2^{(2)}; y^{(2)}), \cdots \qquad (1)$$

where x_1 and x_2 are inputs, and y is the output. This simple two-input one-output case is chosen in order to emphasize and to clarify the basic ideas of our new approach; extensions to general multi-input multi-output cases are straightforward and will be discussed later in this section. The task here is to generate a set of fuzzy rules from the desired input–output pairs of (1), and use these fuzzy rules to determine a mapping $f : (x_1, x_2) \rightarrow y$.

Our approach consists of the following five steps:

Step 1—Divide the Input and Output Spaces into Fuzzy Regions

Assume that the domain intervals of x_1, x_2 and y are $[x_1^-, x_1^+]$, $[x_2^-, x_2^+]$ and $[y^-, y^+]$, respectively, where "domain interval" of a variable means that most probably this variable will lie in this interval (the values of a variable are allowed to lie outside its domain interval). Divide each domain interval into $2N+1$ regions (N can be different for different variables, and the lengths of these regions can be equal or unequal), denoted by SN (Small N), $\cdots, S1$ (Small 1), CE (Center), $B1$ (Big 1), $\cdots, BN$ (Big N), and assign each region a fuzzy membership function. Fig. 2 shows an example where the domain interval of x_1 is divided into five regions $(N = 2)$, the domain region of x_2 is divided into seven regions $(N = 3)$, and the domain interval of y is divided into five regions $(N = 2)$. The shape of each membership function is triangular; one vertex lies at the center of the region and has membership value unity; the other two vertices lie at the centers of the two neighboring regions, respectively, and have membership values equal to zero. Of course, other divisions of the domain regions and other shapes of membership functions are possible.

Step 2—Generate Fuzzy Rules from Given Data Pairs

First, determine the degrees of given $x_1^{(i)}, x_2^{(i)}$ and $y^{(i)}$ in different regions. For example, $x_1^{(1)}$ in Fig. 2 has degree 0.8 in $B1$, degree 0.2 in $B2$, and zero degrees in all other regions. Similarly, $x_2^{(2)}$ in Fig. 2 has degree 1 in CE, and zero degrees in all other regions.

Second, assign a given $x_1^{(i)}, x_2^{(i)}$ or $y^{(i)}$ to the region with maximum degree. For example, $x_1^{(1)}$ in Fig. 2 is considered to be $B1$, and $x_2^{(2)}$ in Fig. 1 is considered to be CE.

Finally, obtain one rule from one pair of desired input–output data, e.g.,

$$(x_1^{(1)}, x_2^{(1)}; y^{(1)}) \Rightarrow [x_1^{(1)}(0.8 \,\text{in}\, B1, \max), x_2^{(1)}(0.7 \,\text{in}\, S1, \max); \\ y^{(1)}(0.9 \,\text{in}\, CE, \max)] \Rightarrow \text{Rule 1:}$$

IF x_1 is $B1$ and x_2 is $S1$, THEN y is CE;

$$(x_1^{(2)}, x_2^{(2)}; y^{(2)}) \Rightarrow [x_1^{(2)}(0.6 \,\text{in}\, B1, \max), x_2^{(2)}(1 \,\text{in}\, CE, \max); \\ y^{(2)}(0.7 \,\text{in} B1, \max)] \Rightarrow \text{Rule 2:}$$

IF x_1 is $B1$ and x_2 is CE, THEN y is $B1$.

The rules generated in this way are "and" rules, i.e., rules in which the conditions of the IF part must be met simultaneously in order for the result of the THEN part to occur. For the problems considered in this paper, i.e., generating fuzzy rules from numerical data, only "and" rules are required since the antecedents are different components of a single input vector.

Step 3—Assign a Degree to Each Rule

Since there are usually lots of data pairs, and each data pair generates one rule, it is highly probable that there will be some conflicting rules, i.e., rules that have the same IF part but a

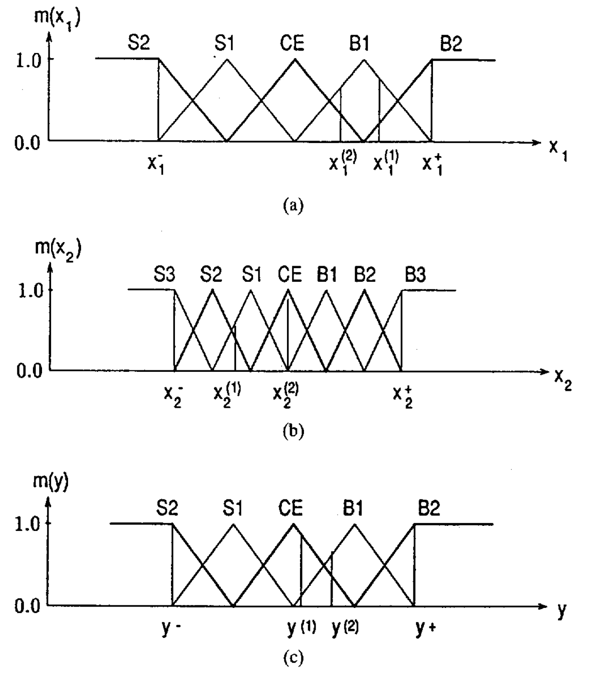


Fig. 2. Divisions of the input and output spaces into fuzzy regions and the corresponding membership functions. (a) $m(x_1)$. (b) $m(x_2)$. (c) $m(y)$.

different THEN part. One way to resolve this conflict is to assign a degree to each rule generated from data pairs, and accept only the rule from a conflict group that has maximum degree. In this way not only is the conflict problem resolved, but also the number of rules is greatly reduced.

We use the following product strategy to assign a degree to each rule: for the rule: "IF x_1 is A and x_2 is B, THEN y is C," the degree of this rule, denoted by D(Rule), is defined as

$$D(\text{Rule}) = m_A(x_1)m_B(x_2)m_C(y). \quad (2)$$

As examples, Rule 1 has degree

$$\begin{aligned} D(\text{Rule1}) &= m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y) \\ &= 0.8 \times 0.7 \times 0.9 = 0.504 \end{aligned} \quad (3)$$

(see Fig. 2) and Rule 2 has degree

$$\begin{aligned} D(\text{Rule2}) &= m_{B1}(x_1)m_{CE}(x_2)m_{B1}(y) \\ &= 0.6 \times 1 \times 0.7 = 0.42. \end{aligned} \quad (4)$$

In practice, we often have some a priori information about the data pairs. For example, if we let an expert check given data pairs, the expert may suggest that some are very useful and crucial, but others are very unlikely and may be caused just by measurement errors. We can therefore assign a degree to each data pair that represents our belief of its usefulness. In this sense, the data pairs constitute a fuzzy set, i.e., the fuzzy set is defined as the useful measurements; a data pair belongs to this set to a degree assigned by a human expert.

Suppose the data pair $(x_1^{(1)}, x_2^{(1)}; y^{(1)})$ has degree $m^{(1)}$, then we redefine the degree of Rule 1 as

$$D(\text{Rule1}) = m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y)m^{(1)} \quad (5)$$

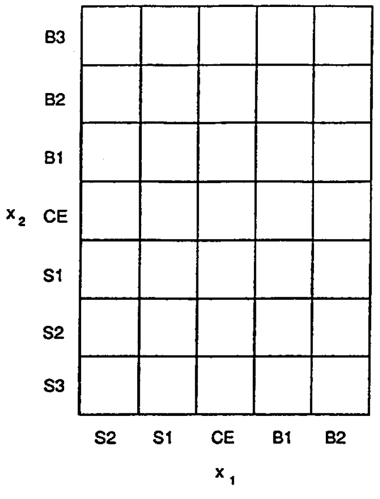


Fig. 3. The form of a fuzzy rule base.

i.e., the degree of a rule is defined as the product of the degrees of its components and the degree of the data pair that generates this rule. This is important in practical applications, because real numerical data have different reliabilities, e.g, some real data can be very bad ("wild data"). For good data we assign higher degrees, and for bad data we assign lower degrees. In this way, human experience about the data is used in a common base as other information. If one emphasizes objectivity and does not want a human to judge the numerical data, our strategy still works by setting all the degrees of the data pairs equal to unity.

Step 4—Create a Combined Fuzzy Rule Base

The form of a fuzzy rule base is illustrated in Fig. 3. We fill the boxes of the base with fuzzy rules according to the following strategy: a combined fuzzy rule base is assigned rules from either those generated from numerical data or linguistic rules (we assume that a linguistic rule also has a degree that is assigned by the human expert and reflects the expert's belief of the importance of the rule); if there is more than one rule in one box of the fuzzy rule base, use the rule that has maximum degree. In this way, both numerical and linguistic information are codified into a common framework—the combined fuzzy rule base. If a linguistic rule is an "and" rule, it fills only one box of the fuzzy rule base; but, if a linguistic rule is an "or" rule (i.e., a rule for which the THEN part follows if any condition of the IF part is satisfied), it fills all the boxes in the rows or columns corresponding to the regions of the IF part. For example, suppose we have the linguistic rule: "IF x_1 is $S1$ or x_2 is CE, THEN y is $B2$" for the fuzzy rule base of Fig. 3; then we fill the seven boxes in the column of $S1$ and the five boxes in the row of CE with $B2$. The degrees of all the $B2$'s in these boxes equal the degree of this "or" rule.

Step 5—Determine a Mapping Based on the Combined Fuzzy Rule Base

We use the following defuzzification strategy to determine the output control y for given inputs (x_1, x_2): first, for given inputs (x_1, x_2), we combine the antecedents of the ith fuzzy rule using product operation to determine the degree, m^i_{Oi}, of the output control corresponding to (x_1, x_2), i.e.,

$$m^i_{O^i} = m_{I^i_1}(x_1) m_{I^i_2}(x_2), \tag{6}$$

where O^i denotes the output region of Rule i, and I^i_j denotes the input region of Rule i for the jth component, e.g., Rule 1 gives

$$m^1_{CE} = m_{B1}(x_1) m_{S1}(x_2) \tag{7}$$

then, we use the following centroid defuzzification formula to determine the output

$$y = \frac{\sum_{i=1}^{K} m^i_{O^i} \bar{y}^i}{\sum_{i=1}^{K} m^i_{O^i}} \tag{8}$$

where $\bar{y}^i$ denotes the *center* value of region O^i (the center of a fuzzy region is defined as the point that has the smallest absolute value among all the points at which the membership function for this region has membership value equal to one), and K is the number of fuzzy rules in the combined fuzzy rule base.

From Steps 1 to 5 we see that our new method is simple and straightforward in the sense that it is a one-pass build-up procedure that does not require time-consuming training; hence, it has the same advantage that the fuzzy approach has over the neural approach, namely, it is simple and quick to construct.

This five step procedure can easily be extended to general multi-input multi-output cases. Steps 1 to 4 are independent of how many inputs and how many outputs there are. In Step 5, we only need to replace $m^i_{O^i}$ in (6) with $m^i_{O^i_j}$, where j denotes the jth component of the output vector (O^i_j is the region of Rule i for the jth output component; $m^i_{O^i_j}$ is the same for all j), and change (8) to

$$y_j = \frac{\sum_{i=1}^{K} m^i_{O^i_j} \bar{y}^i_j}{\sum_{i=1}^{K} m^i_{O^i_j}} \tag{9}$$

where $\bar{y}^i_j$ denotes the center of region O^i_j.

If we view this five step procedure as a block, then the inputs to this block are "examples" (desired input–output data pairs) and expert rules (linguistic IF-THEN statements), and the output is a mapping from input space to output space. For control problems, the input space is the state of the plant to be controlled, and the output space is the control applied to the plant. For time-series prediction problems, the input and output spaces are subsequences of the time series such that the input subsequence precedes the output subsequence (details are given in Section V). Our new method essentially "learns" from the "examples" and expert rules to obtain a mapping that, hopefully, has the "generalization" property that when new inputs are presented the mapping continues to give desired or successful outputs. Hence, our new method can be viewed as a very general *model-free trainable fuzzy system* for a wide range of control and signal processing problems, where: "Model-Free" means no mathematical model is required for the problem; "Trainable" means the system learns from "examples" and expert rules, and can adaptively change the mapping when new "examples" and expert rules are available; and, "Fuzzy" denotes the fuzziness introduced into the system by linguistic fuzzy rules, fuzziness of data, etc.

III. Fuzzy System as a Universal Approximator

The five step procedure of the last section generates a fuzzy system, i.e., a mapping from input space to output space. Specifically, this mapping is represented by (6) and (8) for the two-input one-output case. Using simplified notations, we rewrite (6) and (8), for the general n-input one-output case, as

$$m^i = \Pi_{1 \le j \le n}[m^i_j(x_j)] \tag{10}$$

$$f(\underline{x}) = \frac{\sum_{i=1}^{K} \bar{y}^i m^i}{\sum_{i=1}^{K} m^i} = \frac{\sum_{i=1}^{K} \bar{y}^i \Pi_{1 \le j \le n}[m^i_j(x_j)]}{\sum_{i=1}^{K} \Pi_{1 \le j \le n}[m^i_j(x_j)]} \tag{11}$$

where m^i_j is the membership function of the ith rule for the jth component of the input vector, and $\bar{y}^i$ is the center value of the output region of the ith rule. We will prove that this generated fuzzy system, i.e., (11), is a universal approximator from a compact set $Q \subset R^n$ to R, i.e., it can approximate any real continuous function defined on Q to any accuracy, where the compact set Q is defined as

$$Q = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n] \subset R^n. \tag{12}$$

For notational convenience, we represent Rule $i (i = 1, 2, \cdots, K)$ in the fuzzy rule base as: "IF x_1 is RG^i_1, x_2 is $RG^i_2, \cdots, x_n$ is RG^i_n, THEN y is RG^i_0," where $RG^i_j (j = 1, 2, \cdots, n)$ denotes the region for the j'th input antecedent of Rule i (e.g., it can be $S2, CE, B1$, etc.), and RG^i_0 denotes the outupt region of Rule i.

Let F be the family of functions of the form of (11) on the compact set Q. There are three factors that determine a member of F: 1) the definition of fuzzy regions, i.e., how to define and divide the domain intervals; 2) the specific form of membership functions m^i_j; and, 3) the specific statements of fuzzy rules in the fuzzy rule base. By fixing fuzzy regions, membership functions, and fuzzy rules, we obtain an element of F. If f_1 and f_2 are different elements of F, then at least one of the three factors for f_1 and f_2 must be different. In order to analyses the family F, we make the following assumptions for these three factors:

AS.1: The fuzzy regions for the input and output spaces can be arbitrarily defined.

AS.2: The membership functions m_j^i can be any continuous functions from $[a_j, b_j]$ to [0, 1] for $j = 1, 2, \cdots, n$ (i.e., for inputs) and from $(-\infty, \infty)$ to [0, 1] for $j = 0$ (i.e., for output); however, m_j^i must satisfy the following constraint: $m_j^i(x_j) \neq 0$ for $x_j \in RG_j^i, i = 1, 2, \cdots, K, j = 0, 1, \cdots, n$, with $x_0 = y$. This constraint means that the membership value of an antecedent for a rule cannot equal zero if the actual input value of this antecedent falls into the required region of the rule.

AS.3: Any rule can be assigned to any box of the fuzzy rule base.

These assumptions are usually satisfied in practice. Specifically, we have total freedom in defining fuzzy regions; we can choose any membership functions subject to the constraint of AS.2; and, we can assign any rule to any box of the fuzzy rule base.

To analyses the properties of the function family F, we must first establish that the mapping defined by (11) is well-defined, i.e., for any input $\boldsymbol{x} \in Q$, (11) will generate an output $f(\boldsymbol{x}) \in R$. The following two lemmas give sufficient conditions for (11) to be well-defined.

Lemma 1: If all the membership functions m_j^i are nonzero, and there is at least one rule in the fuzzy rule base, then the mapping defined by (11) from Q to R is well-defined.

Proofs of lemmas and theorems are given in Appendix I.

Lemma 2: If every box in the fuzzy rule base has a rule associated with it, i.e., there are no empty boxes in the fuzzy rule base, then the mapping defined by (11) from Q to R is well-defined under AS.2.

In practice, the input space is usually high dimensional, whereas the given successful data pairs and expert rules are often quite limited; as a result, many boxes of the fuzzy rule base may be empty. However, it is possible to fill up these empty boxes based on the limited given rules using the method of Section II. Specifically, Steps 1–4 are first used to generate a fuzzy rule base based on the limited data pairs and linguistic rules; then, the output for some typical input for which the box in the fuzzy rule base is empty can be determined based on the limited fuzzy rule base; finally, the range in which the output has the maximum degree is assigned to the empty box as a new rule. This can be an iterative procedure, i.e., when a new rule is generated, this new rule and the existing rules are combined into a fuzzy rule base that is used to generate the next new rule. We can start the procedure from the empty boxes that are the nearest neighbors of the full boxes; in this way, the fuzzy rule base expands from existing rules until all the boxes are filled up. This procedure always works if we choose the nonzero regions of the membership functions to be large enough such that the values of the membership functions will not be zero for some points of their nearest neighbors. We will not study this procedure in detail in this paper; we gave the basic ideas of the procedure in order to show that the conditions of Lemma 2 can be satisfied.

Now we state the main result of this section.

Theorem 1: If the mapping defined by (11) is well-defined, and if AS.1–AS.3 are true, then the mapping defined by (11) is capable of approximating any real continuous function over the compact set Q to arbitrary accuracy. (The proof of Theorem 1 uses AS.4 and the definition of "active rule" that will be given later in this section; hence, we suggest the reader read the rest of this section before going to the proof of Theorem 1. The rest of this section will not use the result of Theorem 1.)

Theorem 1 is an existence theorem showing that there exists a way of defining fuzzy regions, a way of choosing membership functions, and a way of assigning fuzzy rules to the boxes of the fuzzy rule base, such that the resulting mapping, (11), approximates an arbitrary nonlinear continuous mapping from Q to R to any accuracy. This Theorem is similar to the results of [6] and [7], which showed that a three-layer feedforward neural network is a universal approximator provided that there are sufficiently large numbers of hidden-layer neurons. Theorem 1 provides the theoretical basis for successful applications of our new method to many different practical problems.

In many applications of fuzzy systems (e.g., [3], [4]), the membership functions are triangular. We now study some properties of the fuzzy systems that use the specific form of membership functions that are defined as follows.

AS.4: The membership function for any intermediate fuzzy region (i.e., not the smallest or the largest region) is a triangle whose vertices are at $(x, m) = (x_{-1}, 0), (x_0, 1)$, and $(x_1, 0)$, where the x-axis denotes a coordinate of the input or output space, the m-axis denotes the corresponding membership value, x_0 denotes the center of the region, and $x_{-1}(x_1)$ denotes the center of the left (right) region. See Fig. 11 for an example. The membership functions for the smallest and largest regions are determined by the way shown in Fig. 11.

If every box of the fuzzy rule base has a rule and $[a_j, b_j]$ is divided into r_j fuzzy regions $(j = 1, 2, \cdots, n)$, then there are $N = r_1 \times r_2 \times \cdots \times r_n$ rules in the fuzzy rule base. N can be a huge number if the $r_j's$ and n are large. However, under the situation of AS.4, there are only a small fraction of these rules that are really used in (11) for any given $\boldsymbol{x} \in Q$.

Definition: The ith fuzzy rule in the fuzzy rule base is *active* for $\boldsymbol{x} \in Q$ if $m_j^i(x_j) \neq 0$ for all $j = 1, 2, \cdots, n$. Referring to (11), we see that a rule is active means that it will be used in (11).

Lemma 3: Under AS.4, the following is true:

1) There are at most 2^n active rules for any $\boldsymbol{x} \in Q$.
2) If r components of $\boldsymbol{x} \in Q$ are at the centers of some fuzy regions $(r = 0, 1, 2, \cdots, n)$, there are at most 2^{n-r} active rules at the $\boldsymbol{x}$ (the center of a fuzzy region is defined in Step 5 of Section II).
3) If r components of $\boldsymbol{x} \in Q$ are at the centers of some fuzzy regions, and if q components of the $\boldsymbol{x}$ are smaller (greater) than the center values of the smallest (largest) regions of the corresponding components, then there are at most 2^{n-r-q} active rules at the $\boldsymbol{x}$.

Lemma 3 is useful in practice. Although we may need a huge memory to store the fuzzy rule base, when we use the fuzzy rule base for a given input $\boldsymbol{x} \in Q$, only a relatively small number of rules are used. In practice, we may store the fuzzy rule base in a cheap external memory; when we have an input, we only take the active rules from the fuzzy rule base into the host computer.

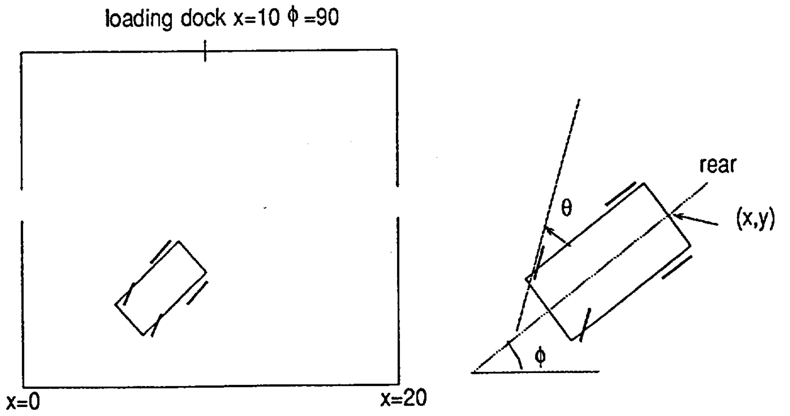


Fig. 4. Diagram of simulated truck and loading zone.

IV. Application to Truck Backer-Upper Control

Backing a truck to a loading dock is a difficult exercise. It is a nonlinear control problem for which no traditional control system design methods exist. In [1], Nguyen and Widrow develop a neural network controller for the truck backer-upper problem; and, in [4], Kong and Kosko propose a fuzzy control strategy for the same problem. The neural network controller [1] only uses numerical data, and cannot utilize linguistic rules determined from expert drivers; on the other hand, the fuzzy controller of [4] only uses linguistic rules, and cannot utilize sampled data. Since the truck backer-upper control problem is a good example of the control system design problem discussed in the Introduction of this paper (i.e., replace a human controller by a machine), it is interesting to apply the approach developed in Section II to this problem. In order to distinguish these methods, we call the method of [4] the "fuzzy approach," the method of [1] the "neural approach," and our new method the "numerical-fuzzy approach."

The results of [4] demonstrated superior performance of the fuzzy controller over the neural controller; however, the fuzzy and neural controllers use different information to construct the control strategies. It is possible that the fuzzy rules used in [4] to construct the controller are more complete and contain more information than the numerical data used to construct the neural controller; hence, the comparison between the fuzzy and neural controllers, from a final control performance point of view, is somewhat unfair. If the linguistic fuzzy rules were incomplete, whereas the numerical information contained lots of very good data pairs, it is highly possible that the neural controller would outperform the fuzzy controller.

Our new numerical-fuzzy approach provides a fair basis for comparing fuzzy and neural controllers (the numerical-fuzzy approach can be viewed as a fuzzy approach in the sense that it differs from the pure fuzzy approach only in the way it obtains fuzzy rules). We can provide the same desired input–output pairs to both the neural and numerical-fuzzy approaches; consequently, we can compare the final control performances of both controllers fairly since they both use the same information.

Example 1: In this example, we use the same set of desired input–output pairs to simulate neural and numerical-fuzzy controllers, and compare their final control performance.

Statement of the Truck Backer-Upper Control Problem: The simulated truck and loading zone are shown in Fig. 4 [1], [4]. The truck corresponds to the cab part of the neural truck in the Nguyen–Widrow [1] neural truck backer-upper system. The truck position is exactly determined by the three state variables ϕ, x, and y, where ϕ is the angle of the truck with the horizontal as shown in Fig. 4. Control to the truck is the angle θ. Only backing up is considered. The truck moves backward by a fixed unit distance every stage. For simplicity, we assume enough clearance between the truck and the loading dock such that y does not have to be considered as an input. The task here is to design a control system, whose inputs are $\phi \in [-90°, 270°]$ and $x \in [0, 20]$, and whose output is $\theta \in [-40°, 40°]$, such that the final states will be $(x_f, \phi_f) = (10, 90°)$.

Generating Desired Input–Output Pairs$(x, \phi; \theta)$: We do this by trial and error: at every stage (given ϕ and x) starting from an initial state, we determined a control θ based on common sense (i.e., our own experience of how to control the steering angle in the situation); after some trials, we chose the desired input–output pairs corresponding to the smoothest successful trajectory.

The following 14 initial states were used to generate desired input–output pairs: (x_0, ϕ_0^o) = (1, 0), (1, 90), (1, 270); (7, 0), (7, 90), (7, 180), (7, 270); (13, 0), (13, 90), (13, 180), (13, 270); (19, 90), (19, 180), (19, 270). Since we performed simulations, we needed to know the dynamics of the truck backer-upper procedure. We used the following approximate kinematics (see [8] for details):

$$x(t+1) = x(t) + \cos[\phi(t) + \theta(t)] + \sin[\theta(t)]\sin[\phi(t)] \quad (13)$$

$$y(t+1) = y(t) + \sin[\phi(t) + \theta(t)] - \sin[\theta(t)]\cos[\phi(t)] \quad (14)$$

$$\phi(t+1) = \phi(t) - \sin^{-1}\left[\frac{2\sin(\theta(t))}{b}\right] \quad (15)$$

TABLE I
DESIRED TRAJECTORY STARTING FROM $(x_0\phi_0) = (1, 0°)$

t	x	$\phi°$	$\theta°$
0	1.00	0.00	-19.00
1	1.95	9.37	-17.95
2	2.88	18.23	-16.90
3	3.79	26.59	-15.85
4	4.65	34.44	-14.80
5	5.45	41.78	-13.75
6	6.18	48.60	-12.70
7	7.48	54.91	-11.65
8	7.99	60.71	-10.60
9	8.72	65.99	-9.55
10	9.01	70.75	-8.50
11	9.28	74.98	-7.45
12	9.46	78.70	-6.40
13	9.59	81.90	-5.34
14	9.72	84.57	-4.30
15	9.81	86.72	-3.25
16	9.88	88.34	-2.20
17	9.91	89.44	0.00
18			
19			
20			

where b is the length of the truck. We assumed $b = 4$ in the simulations of this paper. Equations (13)–(15) were used to obtain the next state when the present state and control are given. Since y is not considered a state, only (13) and (15) were used in the simulations. We wrote (14) here for the purpose of showing the complete dynamics of the truck. Observe, from (13)–(15), that even this simplified dynamic model of the truck is nonlinear. The 14 sequences of desired $(x, \phi; \theta)$ pairs are given in [8]; we only include one such sequence in this paper (Table I).

Neural Control and Simulation Results: We used a two-input single-output three-layer back-propagation neural network [9, 10] for our control task. Twenty hidden neurons were used, and a sigmoid nonlinear function was used for each neuron. The output of the third-layer neuron represents the steering angle θ according to a uniform mapping from [0, 1] to $[-40°, 40°]$, i.e., if the neuron output is $g(t)$, the corresponding output $\theta(t)$ is

$$\theta(t) = 80g(t) - 40. \qquad (16)$$

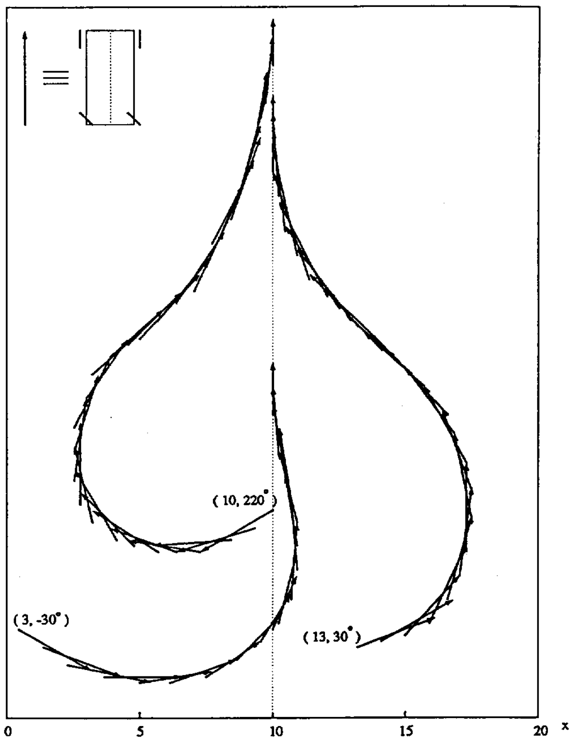
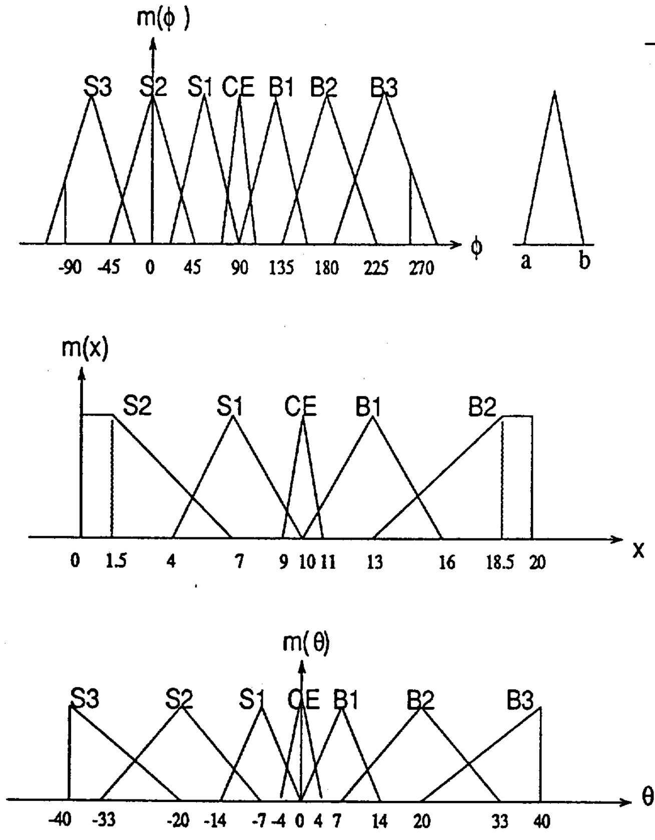


Fig. 5. Truck trajectories using the neural controller and the numerical-fuzzy controller.

In the simulations, we normalized $[-40°, 40°]$ into $[-1, 1]$. Similarly, the inputs to the neurons were also normalized into $[-1, 1]$.

Our neural network controller is different from the Nguyen–Widrow neural controller [1]. First, we have only one neural network that does the same work as the Truck Controller of the Nguyen–Widrow network; the truck emulator of the Nguyen–Widrow network is not needed in our task. Second, and more fundamentally, we train our neural network using desired input–output (state-control) pairs, which are obtained from the past successful control history of the truck, whereas Nguyen and Widrow [1] connect their neural network stage by stage and train the concatenated neural networks by back-propagating the error at the final state through this long network chain (the detailed algorithm is different from the standard error back-propagation algorithm in order to meet the constraint that the neural networks at each stage perform the same transformation; for details see [1]). Hence, the training of our neural network is simpler than that of the Nguyen–Widrow network. Of course, we need to know some successful control trajectories (state-control pairs) starting from some typical initial states; this is not required in the Nguyen–Widrow neural network controller.

We trained the neural network using the standard error back-propagation algorithm [9], [10] for the generated 14 sequences of desired $(x, \phi; \theta)$ pairs. We used the converged network to control the truck whose dynamics are approximately given by (13)–(15). Three arbitrarily chosen initial states, $(x_0, \phi_0^o) =$



	a	b
S3	-115	-15
S2	-45	45
S1	15	90
CE	80	100
B1	90	165
B2	135	225
B3	195	295

Fig. 6. Fuzzy membership functions for the truck backer-upper control problem.

(3, −30), (10, 220), and (13, 30), were used to test the neural controller. The truck trajectories from the three initial states are shown in Fig. 5. We see that the neural controller successfully controls the truck to the desired position starting from all three initial states.

Numerical-Fuzzy Control and Simulation Results: We used the five-step procedure of Section II to determine the control law $f : (x, \phi) \rightarrow \theta$, based on the 14 generated sequences of successful $(x, \phi; \theta)$ pairs. For this specific problem, we used membership functions shown in Fig. 6, which are similar to those used in [4] for fuzzy control of the problem based only on linguistic rules. The fuzzy rules generated from the desired input–output pairs and their corresponding degrees are given in [8]; we show only the generated rules for the data pairs of Table I (in this paper) in Table II. The final fuzzy rule base is shown in Fig. 7 (this is the result of Step 4 of our method in Section II; here we assume that no linguistic rules are available). We see from Fig. 7 that there are no generated rules for some ranges of x and ϕ. This shows that the desired trajectories from the 14 initial states do not cover all the possible cases; however, we will see that the rules in Fig. 7 are sufficient for controlling the truck to the desired state starting from some given initial states.

Finally, Step 5 of our numerical-fuzzy method was used to control the truck from the three initial states, $(x_0, \phi_0^o) =$ (3, −30), (10, 220), and (13, 30), which are the same states used in the simulations of the neural controller. The final trajectories of the truck have no visible difference from Fig. 5; hence, Fig. 5 also shows the track trajectories using the numerical-fuzzy controller.

		x				
		S2	S1	CE	B1	B2
	S3	S2	S3			
	S2	S2	S3	S3	S3	
	S1	B1	S1	S2	S3	S2
ϕ	CE	B2	B2	CE	S2	S2
	B1	B2	B3	B2	B1	S1
	B2		B3	B3	B3	B2
	B3				B3	B2

Fig. 7. The final fuzzy rule base generated from the numerical data for the truck backer-upper control problem.

We simulated the neural and numerical-fuzzy controllers for other initial truck positions, and observed that the truck trajectories using these two controllers were also almost the

TABLE II
Fuzzy Rules Generated from the Desired Input–Output Pairs of Table I and the Degrees of These Rules

Fuzzy rules for t=	IF		THEN	Degree
	x is	ϕ is	θ is	
0	S2	S2	S2	1.00
1	S2	S2	S2	0.92
2	S2	S2	S2	0.35
3	S2	S2	S2	0.12
4	S2	S2	S2	0.07
5	S1	S2	S1	0.08
6	S1	S1	S1	0.18
7	S1	S1	S1	0.52
8	S1	S1	S1	0.56
9	S1	S1	S1	0.60
10	CE	S1	S1	0.35
11	CE	S1	S1	0.21
12	CE	S1	CE	0.16
13	CE	CE	CE	0.32
14	CE	CE	CE	0.45
15	CE	CE	CE	0.54
16	CE	CE	CE	0.88
17	CE	CE	CE	0.92
18				
19				
20				

same. This is not surprising because both controllers used the same information to construct their control laws.

Example 2: In this example we consider the situation where neither linguistic fuzzy rules alone nor desired input–output pairs alone are sufficient to successfully control the truck to the desired position, i.e., neither the usual fuzzy controller with limited fuzzy rules nor the usual neural controller can control the truck to the desired position, but a combination of linguistic fuzzy rules and fuzzy rules generated from the desired input–output data pairs is sufficient to successfully control the truck to the desired position.

We consider the case where the beginning part of the information comes from desired input–output pairs whereas the ending part of the information comes from linguistic rules. To do this we used only the first three pairs of each of the 14 desired sequences, and generated fuzzy rules based only on these truncated pairs. The fuzzy rule base generated from these truncated data pairs is the same as Fig. 7 except that

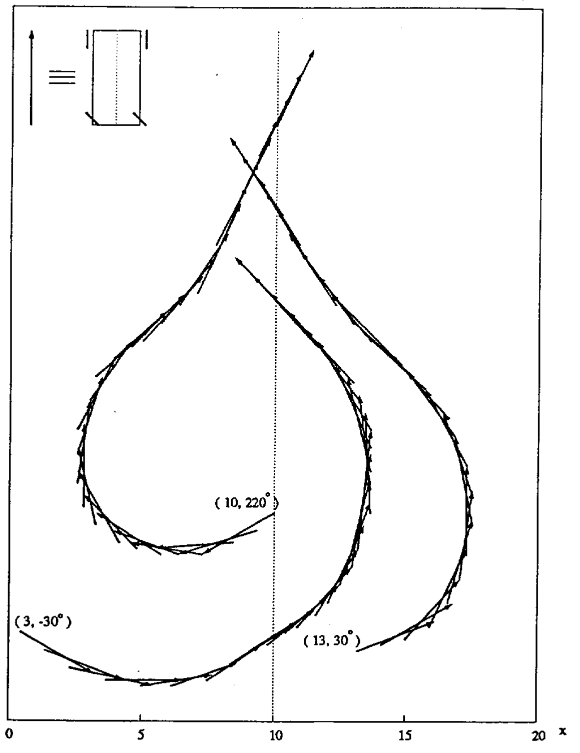


Fig. 8. Truck trajectories using the fuzzy rules from the truncated data pairs only.

there are no rules in the three center boxes outlined by the heavy lines. The fuzzy rule base of linguistic rules for the ending part was chosen to have only three rules that are the same as the three center rules of Fig. 7.

We simulated the following three cases in which we used the: 1) fuzzy rule base generated from only the truncated data pairs; 2) fuzzy rule base of selected linguistic rules; and, 3) fuzzy rule base which combined the fuzzy rule bases of 1) and 2). We see that for Case 3 the fuzzy rule base is the same as in Fig. 7; hence, the truck trajectories for this case must be the same as those using the fuzzy rule base of Fig. 7. For each of the cases, we simulated the system starting from the following three initial states: $(x_0, \phi_0^o) = (3, -30)$, (10, 220), and (13, 30). The resulting trajectories for cases 1), 2), and 3) for the three initial states are shown in Figs. 8, 9 and 5, respectively.

We see very clearly from these figures that, for cases (1) and (2) the truck cannot be controlled to the desired position, whereas for case (3) we successfully controlled the truck to the desired position.

V. Application to Time-Series Prediction

Time-series prediction is a very important practical problem [2]. Applications of time-series prediction can be found in the areas of economic and business planning, inventory and production control, weather forecasting, signal processing, control, and lots of other fields. Let $z(k)(k = 1, 2, 3, \cdots)$ be a time series. The problem of time-series prediction can be formulated as: given $z(k - m + 1), z(k - m + 2), \cdots, z(k)$,

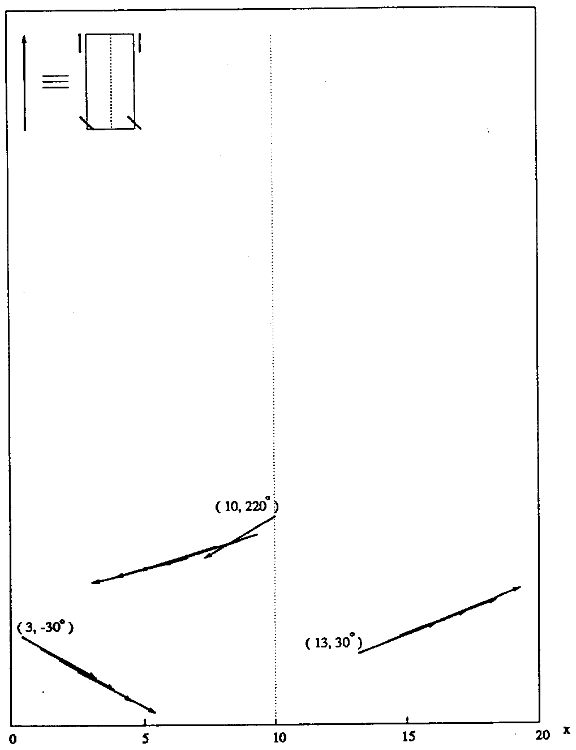


Fig. 9. Truck trajectories using the selected linguistic rules only.

determine $z(k+l)$, where m and l are fixed positive integers; i.e., determine a mapping from $[z(k-m+1), z(k-m+2), \cdots, z(k)] \in R^m$ to $[z(k+l)] \in R$.

A feedforward neural network can also be used for this problem [11]. For example, we can use a three-layer feedforward neural network, which has m input neurons and one output neuron, to represent the mapping from $[z(k-m+1), z(k-m+2), \cdots, z(k)]$ to $[z(k+l)]$. The network is trained for the known $z(k)$'s, and then the converged network is used for the prediction. Specifically, assume that $z(1), z(2), \cdots, z(M)$ are given; then we form $M-m$ desired input–output pairs:

$$\begin{aligned}&[z(M-m), \cdots, z(M-1); z(M)]\\ &[z(M-m-1), \cdots, z(M-2); z(M-1)]\\ &\cdots\\ &[z(1), \cdots, z(m); z(m+1)]. \qquad (17)\end{aligned}$$

We train the neural network to match these $M-m$ pattern pairs using the error back-propagation algorithm [9], [10].

Our numerical-fuzzy method in Section II can also be used for this time series prediction problem. Similar to the neural network approach, we assume that $z(1), z(2), \cdots, z(M)$ are given, and we form the $M-m$ desired input–output pattern pairs in (17). Steps 1–4 of our numerical-fuzzy approach are used to generate a fuzzy rule base based on the pattern pairs (17); then this fuzzy rule base is used to forecast $z(M+p)$ for $p = 1, 2, \cdots$ using the defuzzifying procedure of Step 5 of our numerical-fuzzy method, where the inputs to the network are $z(M+p-m), z(M+p-m+1), \cdots, z(M+p-1)$.

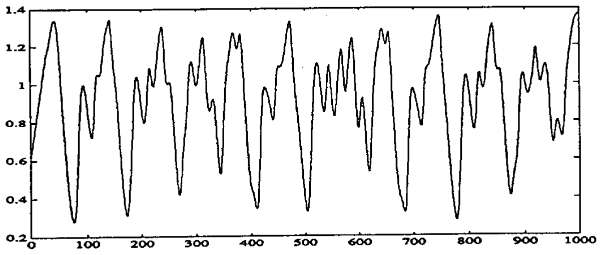


Fig. 10. A section of the Mackey-Glass chaotic time series.

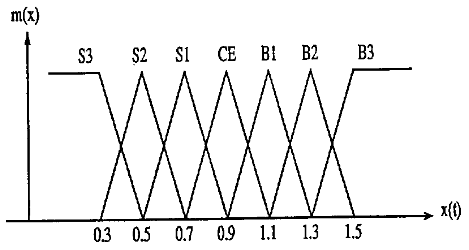


Fig. 11. The first choice of membership functions for the chaotic time series prediction problem.

Example 3: Now we apply our numerical-fuzzy approach to predict the Mackey–Glass chaotic time-series [11]. Chaotic time series are generated from deterministic nonlinear systems and are sufficiently complicated that they appear to be "random" time series; however, because there are underlying deterministic maps that generate the series, chaotic time series are not random time series. In [11], feedforward neural networks were used for chaotic time-series prediction, and were compared with conventional approaches, like linear predictive method, Gabor polynomial method, etc. The results showed that the neural network approach gave the best prediction, and the accuracy obtained using the neural network approach was orders of magnitude higher than that obtained using the conventional approaches. Here we use our numerical-fuzzy approach to the same Mackey–Glass chaotic time series in [11], and compare the results obtained with those obtained using the neural network approach.

The Mackey–Glass chaotic time series is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \qquad (18)$$

When $\tau > 17$, (18) shows chaotic behavior. Higher values of τ yield higher dimensional chaos. In our simulation, we chose the series with $\tau = 30$. Fig. 10 shows 1000 points of this chaotic series that we used to test both the numerical-fuzzy and neural approaches.

We chose $m = 9$ and $l = 1$ in our simulation, i.e., nine point values in the series were used to predict the value of the next time point. The membership functions for any point are shown in Fig. 11 for the numerical-fuzzy predictor (later, we will use other membership functions). 40 hidden-layer neurons were used for the neural network predictor. The first 700 points of the series were used as training data, and the final 300

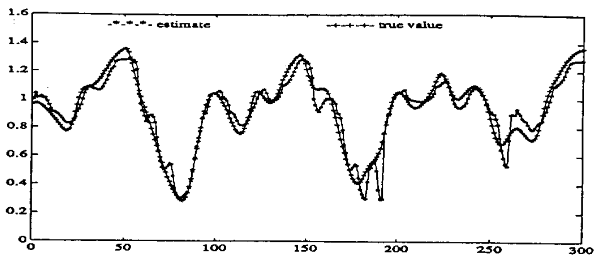


Fig. 12. Prediction of the chaotic time series from $x(701)$ to $x(1000)$ using the numerical-fuzzy predictor when 200 training data (from $x(501)$ to $x(700)$ are used.

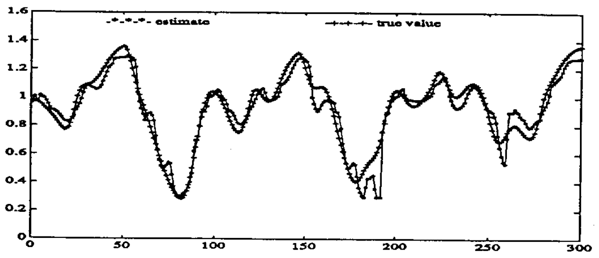


Fig. 13. Prediction of the chaotic time series from $x(701)$ to $x(1000)$ using the neural predictor when 200 training data (from $x(501)$ to $x(700)$) are used.

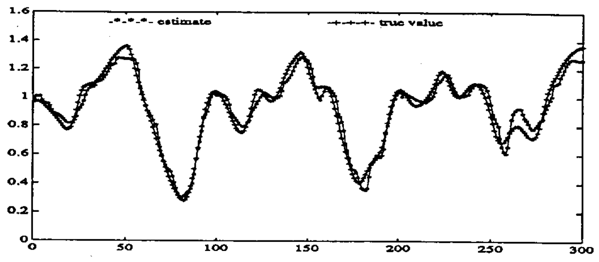


Fig. 14. Prediction of the chaotic time series from $x(701)$ to $x(1000)$ using the numerical-fuzzy predictor when 700 training data (from $x(1)$ to $x(700)$) are used.

points were used as test data (for additional cases, see [8]). We simulated two cases: 1) 200 training data (from 501 to 700) were used to construct the fuzzy rule base and to train the neural network; and, 2) 700 training data (from 1 to 700) were used. Figs. 12 and 13 show the results of the numerical-fuzzy and neural predictors respectively for case 1); and, Figs. 14 and 15 show similar results for case 2). As in [11], the "past" data needed to perform prediction is obtained from observing the actual time series; thus, one makes a prediction and uses the actual values to make the next prediction. We see from Figs. 12 to 15 that our new numerical-fuzzy predictor gave about the same results as the neural network predictor.

One advantage of the numerical-fuzzy approach is that it is very easy to modify the fuzzy rule base as new data become available. Specifically, when a new data pair becomes available, we create a rule for this data pair and add the new rule to the fuzzy rule base; then, the updated (i.e., adapted) fuzzy rule base is used to predict the future values. By using

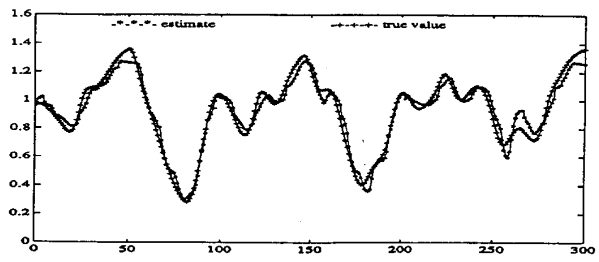


Fig. 15. Prediction of the chaotic time series from $x(701)$ to $x(1000)$ using the neural predictor when 700 training data (from $x(1)$ to $x(700)$) are used.

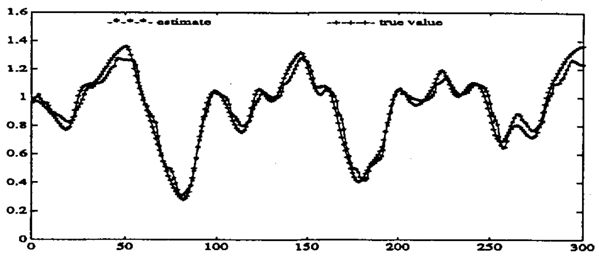


Fig. 16. Prediction of the chaotic time series from $x(701)$ to $x(1000)$ using the updating fuzzy rule base procedure.

this "adaptive" procedure we use all the available information to predict the next value of the series. We simulated this adaptive procedure for the chaotic series of Fig. 10: we started with the fuzzy rule base generated by the data $x(1)$ to $x(700)$, made a prediction of $x(701)$, then used the true value of $x(701)$ to update the fuzzy rule base, and this updated fuzzy rule base was then used to predict $x(702)$. This adaptive procedure continued until $x(1000)$. Its results are shown in Fig. 16. Comparing Figs. 16 and 14 we see that we obtain only a slightly improved prediction.

Finally, we show that prediction can be greatly improved by dividing the "domain interval" into finer regions. We performed two simulations: one with the membership function shown in Fig. 17, and the other with the membership function shown in Fig. 18. We used the adaptive fuzzy rule base procedure for both simulations. The results are shown in Figs. 19 and 20, for the membership functions of Figs. 17 and 18, respectively. Comparing Figs. 16, 19 and 20 we see very clearly that we obtain better and better results as the "domain interval" is divided finer and finer. Fig. 20 shows that we obtained an almost perfect prediction when we divided the "domain interval" into 29 regions. Of course, the price paid for doing this is a larger fuzzy rule base.

VI. Conclusion

In this paper, we developed a general method to generate fuzzy rules from numerical data. This method can be used as a general way to combine both numerical and linguistic information into a common framework—a fuzzy rule base. This fuzzy rule base consists of two kinds of fuzzy rules: some obtained from experts, and others generated from measured numerical data using the method of this paper. We proved that

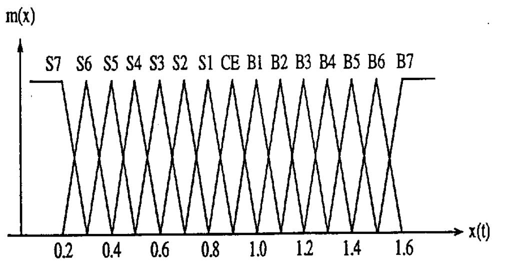


Fig. 17. The second choice of membership functions for the chaotic time series prediction problem.

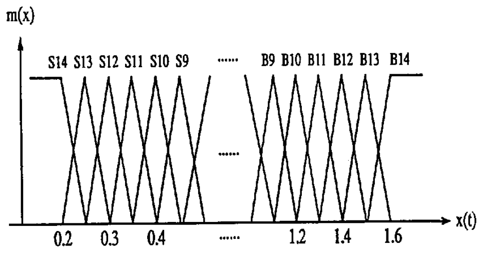


Fig. 18. The third choice of membership functions for the chaotic time series prediction problem.

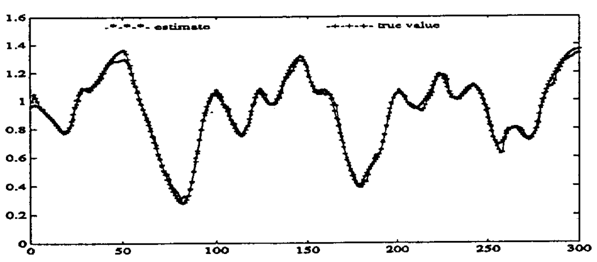


Fig. 19. Prediction of the chaotic time series from $x(701)$ to $x(1000)$ using the updating fuzzy rule base procedure with the membership functions of Fig. 18.

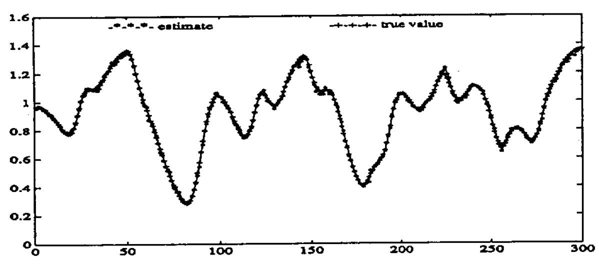


Fig. 20. Prediction of the chaotic time series from $x(701)$ to $x(1000)$ using the updating fuzzy rule base procedure with the membership functions of Fig. 19.

the generated fuzzy system is capable of approximating any nonlinear continuous function on a compact set to arbitrary accuracy. We applied our new method to a truck backer-upper control problem [1], [4], and observed that: 1) for the same training set (i.e., the same given input–output pairs), the final control performance of our new method is indistinguishable from that of the pure neural network controller; and, 2) in the case where neither numerical data nor linguistic rules contain enough information, both the pure neural and pure fuzzy methods failed to control the truck to the desired position, but our new method succeeded. We also applied our new method to a chaotic time-series prediction problem, and the results showed that our new method worked quite well.

The main features and advantages of the new method developed in this paper are: 1) it provides us with a general method to combine measured numerical information and human linguistic information into a common framework—a combined fuzzy rule base; this could be viewed as a first step to develop some theoretically analyzable control algorithms that use both numerical and linguistic information; 2) it is a simple and straightforward one-pass build-up procedure; hence, no time-consuming iterative training is required, so that it requires much less construction time than a comparable neural network; 3) there is lots of freedom in choosing the membership functions; this provides us with lots of flexibilities to design systems according to different requirements; and, 4) it can perform successful control for some cases where neither a pure neural network control nor a pure fuzzy control can.

Appendix I

Proof of Lemma 1: Since $0 \leq m_j^i(x_j) \leq 1$, it is sufficient to prove that for any $\boldsymbol{x} \in Q$ there exist some i such that $\Pi_{1 \leq j \leq n}[m_j^i(x_j)] \neq 0$. Under the condition of this lemma, there exists at least one i such that $\Pi_{1 \leq j \leq n}[m_j^i(x_j)] \neq 0$ for any $\boldsymbol{x} \in Q$, hence (11) is well-defined. Q.E.D.

Proof of Lemma 2: Since every box in the fuzzy rule base has a rule, for any $\boldsymbol{x} \in Q$ there must be a rule, say Rule i, such that $x_j \in RG_j^i$ for $j = 1, 2, \cdots, n$. By AS.2, $m_j^i(x_j) \neq 0$ for all $j = 1, 2, \cdots, n$, hence $\Pi_{1 \leq j \leq n}[m_j^i(x)_j)] \neq 0$, i.e., (11) is well-defined. Q.E.D.

In order to prove Theorem 1 we need some definitions. A family F of real functions defined on a set E is an *algebra* if F is closed under addition, multiplication, and scalar multiplication. The family F *separates points on* E if for every $x, y \in E, x \neq y$, there exists a function $f \in F$ such that $f(x) \neq f(y)$. The family F *vanishes at no point of* E if for each $x \in E$ there exists $f \in F$ such that $f(x) \neq 0$. Our proof of Theorem 1 is based on the Stone-Weierstrass Theorem [5], which we state here for convenience of the reader.

Stone–Weierstrass Theorem: Let F be an algebra of real continuous functions on a compact set K. If F separates points on K and if F vanishes at no point on K, then the uniform closure B of F consists of all real continuous functions on K.

The uniform closure B of F is the union of F and its limit points; hence, if B consists of all real continuous functions on K, then F is capable of approximating any real continuous function on K to arbitrary accuracy.

Proof of Theorem 1: Let F be the family of well-defined functions of the form of (11) on the compact set Q under AS.1, AS.2, and AS.3. If we prove that F is an algebra of real continuous functions, F separates points on Q, and F vanishes at no point of Q, then the Stone–Weierstrass Theorem guarantees the conclusion of Theorem 1.

By AS.2, the $m_j^i(x_j)$'s are assumed to be real continuous functions; hence, F is a family of real continuous functions. Let $f_1, f_2 \in F$, so that we can write them as

$$f_1(\boldsymbol{x}) = \frac{\sum_{i=1}^{K1} \overline{y1}^i \Pi_{1\le j\le n}[m1_j^i(x_j)]}{\sum_{i=1}^{K1} \Pi_{1\le j\le n}[m1_j^i(x_j)]} \tag{19}$$

$$f_2(\boldsymbol{x}) = \frac{\sum_{i=1}^{K2} \overline{y2}^i \Pi_{1\le j\le n}[m2_j^i(x_j)]}{\sum_{i=1}^{K2} \Pi_{1\le j\le n}[m2_j^i(x_j)]}. \tag{20}$$

Thus we have (21), shown at the bottom of the next page. Now Define $m1_j^{i1}(x_j)m2_j^{i2}(x_j)$ as a new membership function of x_j, say $m_j^{i1,i2}(x_j)$, and define $\overline{y1}^{i1} + \overline{y2}^{i2}$ as the output center of a new rule, say $\overline{y}^{i1,i2}$; then, (21) is of the form of (11); hence, $f_1 + f_2 \in F$. Similarly, $f_1(\boldsymbol{x})f_2(\boldsymbol{x})$ can be written as

$$f_1(\boldsymbol{x})f_2(\boldsymbol{x}) = \frac{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \overline{y1}^{i1}\overline{y2}^{i2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]} \tag{22}$$

which is of the form (11); hence, $f_1 f_2 \in F$. Finally, for any $c \in R$:

$$cf_1(\boldsymbol{x}) = \frac{\sum_{i=1}^{K1} c\overline{y1}^i \Pi_{1\le j\le n}[m1_j^i(x_j)]}{\sum_{i=1}^{K1} \Pi_{1\le j\le n}[m1_j^i(x_j)]} \tag{23}$$

which is also of the form of (11); hence $cf_1(\boldsymbol{x}) \in F$. In summary, F is an algebra of real continuous functions.

Next, we prove that F separates points on Q. Let $\boldsymbol{x}, \boldsymbol{z} \in Q$ and $\boldsymbol{x} \neq \boldsymbol{z}$. We now construct $f \in F$ with $f(\boldsymbol{x}) \neq f(\boldsymbol{z})$. First, we define the fuzzy regions of the input space Q such that each element of $\boldsymbol{x}$ and $\boldsymbol{z}$ is at the center of a fuzzy region (recall that the center of a fuzzy region is defined as the point that has the smallest absolute value among all the points at which the membership function for this region has membership value equal to one; see Section II, Step 5). Then, we choose the membership functions for the input space Q to be of the specific triangular form defined by AS.4 of Section III. By such a choice of fuzzy regions and membership functions, we have $m_j^i(x_j) = m_j^l(z_j) = 1$ for each active Rule i at $\boldsymbol{x}$, and each active Rule l at $\boldsymbol{z}$, and all $j = 1, 2, \cdots, n$ (the definition of *active rule* is given in Section III); additionally, there is one and only one active rule for $\boldsymbol{x}$ and one and only one active rule for $\boldsymbol{z}$, because (11) is well-defined (which guarantees that there is at least one active rule for $\boldsymbol{x}$ and at least one active rule for $\boldsymbol{z}$), and since only the membership functions for the regions with centers at the components of $\boldsymbol{x}$ or $\boldsymbol{z}$ are nonzero at $\boldsymbol{x}$ or $\boldsymbol{z}$, whereas all other membership functions are zero at $\boldsymbol{x}$ and $\boldsymbol{z}$ (which guarantees that there is at most one active rule for $\boldsymbol{x}$ and at most one active rule for $\boldsymbol{z}$). Since $\boldsymbol{x} \neq \boldsymbol{z}$, there must be at least one j such that $x_j \neq z_j$, hence, the only active rule for $\boldsymbol{x}$ and the only active rule for $\boldsymbol{z}$ are at two different boxes of the fuzzy rule base. Since we are free to assign any rules to the boxes of the fuzzy rule base (AS.3), we just assign two different rules to these two boxes, and obtain the required $f \in F$ with $f(\boldsymbol{x}) = \overline{y}^i \neq \overline{y}^l = f(\boldsymbol{z})$ (see (11)), where $\overline{y}^i(\overline{y}^l)$ is the center of the output region of the active rule for $\boldsymbol{x}(\boldsymbol{z})$.

Finally, we prove that F vanishes at no point of Q. By AS.1 and AS.2, we can make all the $\overline{y}^i > 0$. Since (11) is well-defined, there exists at least one i such that $\Pi_{1\le j\le n}[m_j^i(x_j)] \neq 0$ for any $\boldsymbol{x} \in Q$. Since (11) is a weighted average of positive $\overline{y}^i$'s with some nonzero weights, the result is also positive, i.e., we obtain $f \in F$ such that $f(\boldsymbol{x}) \neq 0$ (in fact, $f(\boldsymbol{x}) > 0$) for any $\boldsymbol{x} \in Q$. Q.E.D.

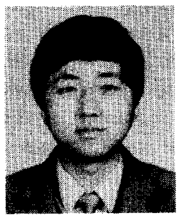
Proof of Lemma 3: For arbitrary $\boldsymbol{x} \in Q$ and fixed j, there are at most two m_j^i's which are nonzero at x_j under AS.4. Since a rule, say Rule i, is active only when $m_j^i(x_j) \neq 0$ for all $j = 1, 2, \cdots, n$, there are at most 2^n active rules for any $\boldsymbol{x} \in Q$; this proves (1). If r components of $\boldsymbol{x} \in Q$ are at the centers of some fuzzy regions, there is only one m_j^i, which is nonzero at each of these r components (in fact, these m_j^is are equal to unity at these r components), and for each of the other $n - r$ components there are at most two nonzero m_j^i's, hence the total number of active rules is at most 2^{n-r}; this proves (2). If r components of $\boldsymbol{x} \in Q$ are at the centers of

$$\begin{aligned} f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) &= \frac{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \overline{y1}^{i1} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)] + \sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \overline{y2}^{i2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]} \\ &= \frac{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} (\overline{y1}^{i1} + \overline{y2}^{i2}) \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}{\sum_{i1=1}^{K1}\sum_{i2=1}^{K2} \Pi_{1\le j\le n}[m1_j^{i1}(x_j)m2_j^{i2}(x_j)]}. \end{aligned} \tag{21}$$

some fuzzy regions and q components of the $\boldsymbol{x}$ are smaller (or greater) than the center values of the corresponding smallest (or the corresponding largest) fuzzy regions, then there is only one nonzero m_j^i for each of these $r+q$ components, and the other $n-r-q$ components have two nonzero m_j^i's associated with each of them; hence, the total number of active rules is at most 2^{n-r-q}; this proves 3). Q.E.D.

REFERENCES

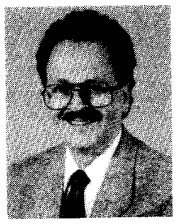
[1] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural network," *IEEE Contr. Syst. Mag.*, vol. 10, no. 3, pp. 18–23, 1990.
[2] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control.* Oakland, CA: Holden-Day, 1976.
[3] Y. F. Li and C. C. Lan, "Development of fuzzy algorithms for servo systems," *IEEE Contr. Syst. Mag.*, vol. 9, no. 3, pp. 65–72, 1989.
[4] S. G. Kong and B. Kosko, "Comparison of fuzzy and neural truck backer upper control systems," in *Proc. IJCNN-90,* vol. 3, June 1990, pp. 349–358.
[5] W. Rudin, *Principles of Mathematical Analysis.* New York: McGraw-Hill, 1964.
[6] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, pp. 359-366, 1989.
[7] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems,* 1989.
[8] L. X. Wang and J. M. Mendel, "Generating fuzzy rules from numerical data, with applications," USC SIPI Rep. No. 169, Univ. Southern Calif., Los Angeles, 1991.
[9] P. Werbos, "New tools for predictions and analysis in the behavioral science," Ph.D. dissertation, Harvard Univ. Comm. Appl. Math., 1974.
[10] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing,* Vol. 1. Cambridge, MA: MIT Press, 1986.
[11] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modeling," LA-UR-87-2662, 1987.



Li-Xin Wang received the B.S. and M.S. degrees from the Northwestern Polytechnical University, Xian, People's Republic of China, in 1984 and 1987, respectively, and the Ph.D. degree from the University of Southern California, Los Angeles in 1992, all in electrical engineering.

From 1987 to 1989, he was with the Department of Computer Science and Engineering, Northwestern Polytechnical University, Xian, People's Republic of China. From Fall 1989 to Spring 1992, he was a Research/Teaching Assistant in the Department of Electrical Engineering-Systems at the University of Southern California, Los Angeles. He is now a Postdoctoral Fellow in the Department of Electrical Engineering and Computer Science, University of California at Berkeley. His research interests are fuzzy systems and neural computing.

Dr. Wang received the Phi Kappa Phi Student Recognition Award for his work on fuzzy systems.



Jerry M. Mendel (S'59–A'61–M'72–F'78) received the B.S. degree in mechanical engineering and the M.S. and Ph.D. degrees in electrical engineering from the Polytechnic Institute of Brooklyn, Brooklyn, NY, in 1959, 1960, and 1963, respectively.

His experience has included teaching courses in Electrical Engineering at the Polytechnic Institute of Brooklyn, from 1960 to 1963, and has also included various consulting positions. From July 1963 to January 1974 he was with McDonnell Douglas Astronautics Company. Currently he is Professor of Electrical Engineering-Systems at the University of Southern California in Los Angeles, and is Director of the Signal & Image Processing Institute. He was Chairman of the EE-Systems Department from March 1984 to August 1991.

He teaches courses in estimation theory, deconvolution, and higher-order statistics. He has published more than 240 technical papers and is author of the monographs *Maximum-Likelihood Deconvolution: a Journey into Model-Based Signal Processing* (Springer-Verlag, 1990) and *Optimal Seismic Deconvolution: An Estimation-Based Approach* (Academic Press, 1983), the texts *Lessons in Digital Estimation Theory* (Prentice-Hall, 1987), and *Discrete Techniques of Parameter Estimation: The Equation Error Formulation* (Dekker, 1973), and, co-editor (with K. S. Fu (deceased)) of *Adaptive, Learning and Pattern Recognition Systems* (Academic Press, 1970). He is also author of the IEEE Individual Learning Program, *Kalman Filtering, and Other Digital Estimation Techniques.* He served as Editor of the IEEE Control Systems Society's IEEE TRANSACTIONS ON AUTOMATIC CONTROL, and is on the Editorial Board of the IEEE PROCEEDINGS.

Dr. Mendel is a Fellow of the IEEE, Distinguished Member of the IEEE Control Systems Society, member of the IEEE Geoscience and Remote Sensing Society, the IEEE Signal Processing Society, the Society of Exploration Geophysicists, the European Association for Signal Processing, Tau Beta Pi and Pi Tau Sigma, and a registered Professional Control Systems Engineer in California. He was President of the IEEE Control Systems Society in 1986. He received the SEG 1976 Outstanding Presentation Award for a paper on the application of Kalman Filtering to deconvolution; the 1983 Best Transactions Paper Award for a paper on maximum-likelihood deconvolution in the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING; a Phi Kappa Phi book award for his 1983 research monograph on seismic deconvolution; a 1985 Burlington Northern Faculty Achievement Award; and a 1984 IEEE Centennial Medal.