USC–SIPI REPORT #206

Analysis and Design of Fuzzy Systems

by

Li-Xin Wang

June 1992

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

Acknowledgements

I would like to express my first acknowledgement to my advisor, Dr. Jerry M. Mendel, for his constant support and encouragement throughout my Ph.D. study. It was an enjoyable experience to work with Dr. Mendel. His remarkable quick response to all my work made my research more quickly. Most of all, I would like to thank Dr. Mendel for giving me the chance to study in the U.S.A..

I would like to express my special appreciation to my future postdoctoral advisor, Dr. Lotfi A. Zadeh. His invaluable encouragement and appreciation for my work on fuzzy systems is the main reason for my decision to concentrate on the research of fuzzy systems.

I am indebted to my dissertation committee members Dr. Alvin M. Despain and Dr. Robert J. Sacker for their contributions in numerous ways to this dissertation.

My extended appreciation goes to my classmates and good friends Dr. Weizheng Wang and Mr. Chiu Yeung Ngo. Their selfless help made my life easier in this new country, which in turn greatly helped my research.

Finally, I wish to express my deepest appreciation to my wife Yingbi. Her constant encouragement and support made my research so enjoyable. I would like to dedicate this dissertation to her.

Contents

List of Figures

Abstract

The goal of this thesis is to develop methods which can combine both numerical information and linguistic information into a common framework, and to apply them to a variety of control, signal processing, and communication problems. This goal is important because so much human knowledge is represented in natural language and to incorporate it into engineering systems is clearly needed. We show that fuzzy systems are powerful tools for achieving this goal. We first present a systematic description of fuzzy systems and show that the fuzzy systems are very general and include probabilistic general regression as a special case. To justify the use of fuzzy systems as basic building blocks for engineering systems (controllers, identifiers, predictors, filters, etc.), we rigorously prove, using the Stone-Weierstrass Theorem in mathematical analysis, that the fuzzy systems are capable of approximating any nonlinear function over a compact set to arbitrary accuracy. Then, three design methods for fuzzy systems are developed which determine fuzzy systems based on desired input-output pairs and fuzzy IF-THEN rules from human experts. In the first method, we show that fuzzy systems can be represented as three-layer feedforward networks, and develop a back-propagation algorithm to train the fuzzy systems to match desired input-output pairs. We use this back-propagation fuzzy system as identifiers of nonlinear dynamic systems and show, through simulations, that the performance of the fuzzy identifiers is much better than the neural network identifiers. The second method is based on the classical orthogonal least squares algorithm; the basic idea is to represent the fuzzy systems as expansions of fuzzy basis functions and use the orthogonal least squares algorithm to select the significant fuzzy basis functions. We apply this method to the control of the nonlinear ball and beam system. The third method is a simple one-pass procedure which is based on a five-step procedure of generating fuzzy rules from numerical data. We apply this method to the

truck backer-upper control and time-series prediction problems and show that the performance of this new method is better than that of the conventional fuzzy and neural approaches. Finally, we develop two nonlinear adaptive filters based on fuzzy system models, namely RLS and LMS fuzzy adaptive filters, and use them as equalizers for nonlinear communication channels. We show that the fuzzy adaptive filters are well-defined nonlinear adaptive filters, and have the unique advantage (as compared with polynomial, neural nets, and radial basis function, etc., adaptive filters) of directly incorporating linguistic information from human experts into the filters.

Chapter 1

INTRODUCTION

1.1 Combining Numerical and Linguistic Information into Engineering Systems — A Fuzzy System Approach

For most real-world engineering problems, the information concerning system design, evaluation, implementation, etc., can be classified into two categories: numerical information obtained from sensor measurements, and linguistic information obtained from human experts. Conventional engineering approaches can only process numerical information, whereas conventional expert systems can only make use of linguistic information. Therefore, their successful applications are limited to problems where either linguistic information or numerical information does not play a critical role. However, there are a large number of practical engineering problems where both numerical information and linguistic information are critical. In order to combine these two kinds of information into engineering systems, the so called intelligent approaches (intelligent control, intelligent signal processing, intelligent networks, etc.) have been developed. Most existing intelligent approaches are heuristic in nature, i.e., they combine conventional methods, perhaps after some modification, with expert systems in an ad hoc way for a specific problem; simulations are then performed to show that the proposed approaches work well for the specific problem. This kind of approach has at least

three weakpoints: 1) it is inefficient and time-consuming to use because there is no systematic way to guide the design and evaluation; 2) it is quite problem dependent, i.e., a method may work well for one problem but may not be suitable for another problem; and 3) there is no common framework to represent numerical and linguistic information, which makes theoretical analyses for these approaches very difficult.

The goal of this thesis is to develop a series of methods which combine numerical information and linguistic information into a common framework and to apply them to a variety of control, signal processing, and communication problems. Fuzzy systems are powerful tools for achieving this goal, because: on one hand, fuzzy systems are constructed from fuzzy IF-THEN rules so that linguistic information can be naturally incorporated into the systems; on the other hand, a variety of learning algorithms can be developed (as shown in this thesis) to train the systems to match numerical input-output pairs.

The idea of fuzzy systems was introduced by Zadeh [106] very early in the literature of fuzzy sets. Research on fuzzy systems has developed in two main directions. The first studies fuzzy systems in the same conceptual framework as classical systems, and has given birth to a body of abstract results concerning the stability [26], reachability [13,51], observability [13,51], etc., of fuzzy systems. This approach has not found successful applications to the study of real processes, perhaps because stability, reachability, observability, etc., are crisp concepts (e.g., a system is stable or unstable), and it is very difficult to justify their fuzzy counterparts. The second direction is the linguistic approach to fuzzy systems [41,108], in which a fuzzy system is constructed from a collection of fuzzy IF-THEN rules and a fuzzy inference engine which uses techniques in approximate reasoning [1,109,110]. This approach was used to synthesize fuzzy logic controllers [41,42] which have been successfully applied to a wide variety of practical problems during the past decade [40]. We adopt this second approach to fuzzy systems in this thesis.

The basic configuration of a fuzzy system is shown in Fig. 1.1. There are four principal components in a fuzzy system: 1) *fuzzy rule base* which comprises fuzzy rules describing how the fuzzy system performs; it is the heart of the whole system in the sense that other three components are used to interpret and combine these

rules to form the final system; 2) *fuzzy inference engine* which uses techniques in approximate reasoning to determine a mapping from the fuzzy sets in the input space $U \subset R^n$ to the fuzzy sets in the output space $V \subset R^m$; 3) *fuzzifier* which maps crisp points in the input space into fuzzy sets in the input space; and, 4) *defuzzifier* which maps fuzzy sets in the output space into crisp points in the output space. Depending upon whether there are fuzzifier and defuzzifier, we have two classes of fuzzy systems. The first class of fuzzy systems is comprised of only the fuzzy rule base and fuzzy inference engine, and therefore operates in a pure fuzzy environment, i.e., inputs and outputs to these fuzzy systems are fuzzy variables. If there is a feedback as shown in Fig. 1 by the dashed arrow line, we have the so called fuzzy dynamic systems [13,72,73]. The second class of fuzzy systems is comprised of all the four components, and performs mappings from crisp $U \subset R^n$ to crisp $V \subset R^m$. In the literature, this second class fuzzy systems are called "fuzzy logic controllers" [33,34], because their most successful applications have been to control problems. In this thesis, we use this second class of fuzzy systems.

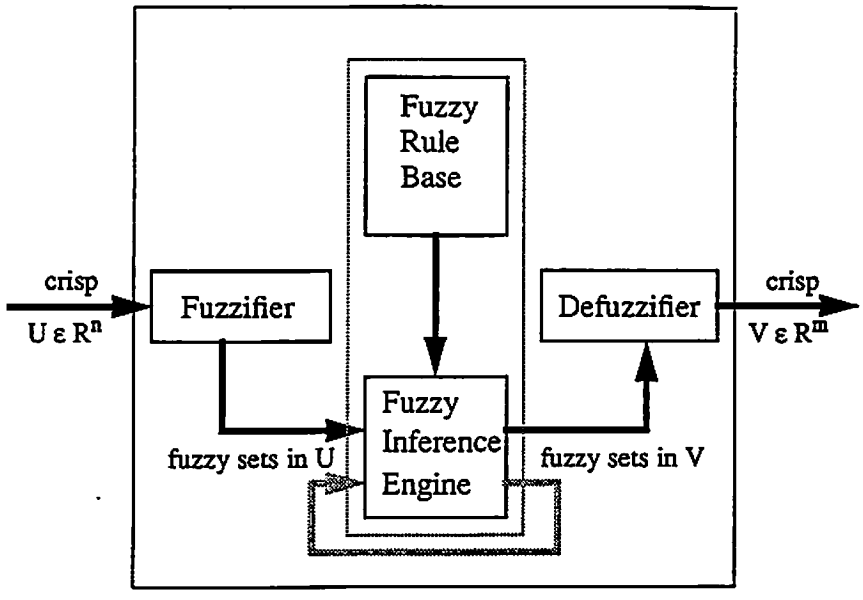


Figure 1.1: Basic configuration of fuzzy systems.

1.2 Description of Fuzzy Systems

Figure 1.1 shows the basic configuration of the fuzzy systems considered in this thesis. The fuzzy system performs a mapping from crisp $U \subset R^n$ to crisp $V \subset R^m$. We assume that $U = U_1 \times \cdots \times U_n$ and $V = V_1 \times \cdots \times V_m$, where $U_i, V_j \subset R$, $i = 1, 2, ..., n$, and $j = 1, 2, ..., m$. Before we present a detailed description for each component of the fuzzy system, we briefly review some basic concepts in fuzzy sets and systems theory which are useful for describing the fuzzy systems.

A *fuzzy set* [107] F of a universe of discourse U is characterized by a membership function $\mu_F : U \rightarrow [0, 1]$ which associates with each element u of U a number $\mu_F(u)$ in the interval $[0, 1]$ which represents the grade of membership of u in F. The label F of a fuzzy set is often some linguistic term like "small", "large", etc.. The *support* of a fuzzy set F is the crisp set of all points $u \in U$ such that $\mu_F(u) > 0$. A *fuzzy singleton* F is a fuzzy set whose support is a single point $u_0 \in U$ with $\mu_F(u_0) = 1$.

A *linguistic variable* [108] is a variable whose values are words in a natural or artificial language, and these words are labels of fuzzy sets. For example, if "slow", "medium", and "fast" are values of "speed", then "speed" is a linguistic variable. A variable can be viewed either as a linguistic variable or a numerical variable. For example, the variable "speed" can be viewed either as a linguistic variable, if it takes values of "slow", "medium", or "fast", or as a numerical variable, if it takes values in the interval $[0, V_{max}]$ where V_{max} is the maximum speed of the car.

A *fuzzy IF-THEN rule* [108] is an expression of the form "IF A THEN B," where A and B are statements about what values the linguistic variables take on. For example, A and B could be "the speed is slow" (i.e., "speed = slow"), or "the speed is fast," etc.. We often use the following fuzzy IF-THEN rules to drive a car: "IF the speed is slow, THEN apply 'more' force to the accelerator", "IF the speed is medium, THEN apply 'less' force to the accelerator," etc., where the 'more', 'less', etc., are characterized by fuzzy membership functions.

A *triangular norm* '$\star$' is defined as a mapping from $[0, 1] \times [0, 1]$ to $[0, 1]$, and the most commonly used triangular norms are [33]

$$x \star y \;=\; \begin{cases} min[x,y] & intersection \\ xy & algebraic\ product \\ max[0, x+y-1] & bounded\ product \end{cases} \tag{1.1}$$

where $x, y \in [0,1]$. A *triangular co-norm* '$\dot{+}$' is a mapping from $[0,1] \times [0,1]$ to $[0,1]$, and the most commonly used triangular co-norms are [33]

$$x \dot{+} y \;=\; \begin{cases} max[x,y] & union \\ x+y-xy & algebraic\ sum \\ min[1, x+y] & bounded\ sum. \end{cases} \tag{1.2}$$

Let $F_1, ..., F_n$ be fuzzy sets in $X_1, ..., X_n$, respectively; the *cartesian product* of $F_1, ..., F_n$, denoted by $F_1 \times \cdots \times F_n$, is a fuzzy set in the product space $X = X_1 \times \cdots \times X_n$ with membership function

$$\mu_{F_1 \times \cdots \times F_n}(x_1, ..., x_n) = \mu_{F_1}(x_1) \star \cdots \star \mu_{F_n}(x_n). \tag{1.3}$$

Let R and S be fuzzy sets in $X \times W$ and $W \times Y$, respectively; the *sup-star composition* of R and S, denoted by $R \circ S$, is a fuzzy set in $X \times Y$ with membership function

$$\mu_{R \circ S}(x,y) = sup_{w \in W}[\mu_R(x,w) \star \mu_S(w,y)]. \tag{1.4}$$

Let A and B be fuzzy sets in X and Y, respectively; the *fuzzy implication* $A \rightarrow B$, or 'IF A THEN B', is defined as a fuzzy set in $X \times Y$ with membership function

$$\mu_{A \rightarrow B}(x,y) = \mu_A(x) \star \mu_B(y). \tag{1.5}$$

Operations other than $\star$ may be used in (5) to define the fuzzy implication (see [33]).

1.2.1 Fuzzy Rule Base

A *fuzzy rule base* R is a collection of fuzzy IF-THEN rules

$$R = [R^1, R^2, ..., R^M], \tag{1.6}$$

with

$$\begin{aligned} R^l: \quad & IF \ \ (x_1 \ \ is \ \ F_1^l \ \ and \ \ \cdots \ \ and \ \ x_p \ \ is \ \ F_p^l); \\ & THEN \ \ (z_1 \ \ is \ \ G_1^l, \ \ \cdots, \ \ z_q \ \ is \ \ G_q^l), \end{aligned} \tag{1.7}$$

where $\underline{x} = (x_1, ..., x_n)^T$ and $\underline{z} = (z_1, ..., z_m)^T$ are the input and output vectors to the fuzzy system, respectively, F_i^l and G_j^l are labels of fuzzy sets in U_i and V_j, respectively, $1 \leq p \leq n$, $1 \leq q \leq m$, and $l = 1, 2, ..., M$. It is clear that R^l of (1.7) can be decomposed into a collection of q rules

$$R^l = [R_1^l, \cdots, R_q^l], \tag{1.8}$$

where

$$\begin{aligned} R_j^l: \quad & IF \ \ (x_1 \ \ is \ \ F_1^l \ \ and \ \ \cdots \ \ and \ \ x_p \ \ is \ \ F_p^l); \\ & THEN \ \ (z_j \ \ is \ \ G_j^l), \end{aligned} \tag{1.9}$$

$j = 1, 2, ..., q$.

Although the IF part of R_j^l is connected by 'and' operation, R_j^l is general enough to include rules whose IF part contains 'or' or 'not' operations. Specifically, if there is an 'or' operation, we can break the rule into two rules whose IF parts contain only 'and' operation. For example, we can decompose the rule: "IF x_1 is F_1 and x_2 is F_2 or x_3 is F_3, THEN z is G" into two rules: "IF x_1 is F_1 and x_2 is F_2, THEN z is G"; and, "IF x_3 is F_3, THEN z is G". This decomposition is clearly consistent with our intuition of the 'or' operation. If there is a 'not' operation, e.g., "x_1 is not F_1", we can define a new fuzzy set $\bar{F}_1$ with membership function $\mu_{\bar{F}_1} = 1 - \mu_{F_1}$, so that "$x_1$ is not F_1" is equivalent to "x_1 is $\bar{F}_1$". Therefore, the fuzzy rule base R is quite general and includes many linguistic descriptions in

fuzzy terms about the relationships between the input $\underline{x}$ and the output $\underline{z}$.

1.2.2 Fuzzy Inference Engine

A *fuzzy inference engine* uses the rules in the fuzzy rule base to determine a mapping from fuzzy sets in U to fuzzy sets in V based on fuzzy logic operations. In the fuzzy inference engine, the IF part of R_j^l defines a cartesian product of $F_1^l, ..., F_p^l$, and the R_j^l itself is viewed as a fuzzy implication $F_1^l \times \cdots \times F_p^l \rightarrow G_j^l$. Let A be an arbitrary fuzzy set in U, then each R_j^l of (1.9) determines a fuzzy set $A \circ R_j^l$ in V_j based on the sup-star composition

$$\begin{aligned} \mu_{A \circ R_j^l}(z_j) &= sup_{\underline{x} \in U}[\mu_A(\underline{x}) \star \mu_{F_1^l \times \cdots \times F_p^l \rightarrow G_j^l}(\underline{x}, z_j)] \\ &= sup_{\underline{x} \in U}[\mu_A(\underline{x}) \star \mu_{F_1^l}(x_1) \star \cdots \star \mu_{F_p^l}(x_p) \star \mu_{G_j^l}(z_j)], \end{aligned} \quad (1.10)$$

where $y_j \in V_j$. The final fuzzy set $A \circ R_j$ ($R_j = [R_j^1, ..., R_j^M]$) in V_j determined by the fuzzy inference engine is obtained by combining (1.10) for $l = 1, 2, ..., M$ using the triangular co-norm $\dot{+}$

$$\mu_{A \circ R_j}(z_j) = \mu_{A \circ R_j^1}(z_j) \dot{+} \cdots \dot{+} \mu_{A \circ R_j^M}(z_j). \quad (1.11)$$

In summary, the fuzzy inference engine determines a mapping from fuzzy set A in U to fuzzy set $A \circ R$ in V by

$$\mu_{A \circ R}(\underline{z}) = (\mu_{A \circ R_1}(z_1), \cdots, \mu_{A \circ R_m}(z_m))^T .^1 \quad (1.12)$$

1.2.3 Fuzzifier

A *fuzzifier* maps a crisp point $\underline{x} = (x_1, ..., x_n)^T \in U$ into a fuzzy set $A_x = [A_{x_1}, ..., A_{x_n}]$ in $U = U_1 \times \cdots \times U_n$, where A_{x_i} is a fuzzy set in U_i. There are (at least) two possible choices of this mapping:

(i) A_{x_i} is a fuzzy singleton with support x_i, i.e., $\mu_{A_{x_i}}(x_i') = 1$ for $x_i' = x_i$ and $\mu_{A_{x_i}}(x_i') = 0$ for all other $x_i' \in U_i$; we call this *a singleton fuzzifier*; and,

[1] We assume that each z_j for $1 \leq j \leq m$ appears at least once in the R^l of (1.7), therefore $\mu_{A \circ R}(\underline{z})$ is an m-dimensional vector.

(ii) $\mu_{A_{x_i}}(x_i) = 1$ and $\mu_{A_{x_i}}(x_i')$ decreases from one as x_i' moves away from x_i, e.g., $\mu_{A_{x_i}}(x_i') = exp[-\frac{(x_i'-x_i)^2}{\sigma^2}]$, where σ^2 is a parameter characterizing the shape of $\mu_{A_{x_i}}(x_i')$; we call this *a nonsingleton fuzzifer.*

The fuzzifier is needed because the inputs to the fuzzy system are crisp, whereas the inputs to the fuzzy inference engine must be fuzzy sets. In the literature, it seems that only the singleton fuzzifier has been used. We think that the nonsingleton fuzzifier may be useful if the inputs are corrupted by noise; this issue needs further studying.

1.2.4 Defuzzifier

A *defuzzifier* maps a fuzzy set in V to a crisp point in V. The defuzzifier is needed because for most practical applications, a fuzzy system is required to give a crisp output, no matter if it is used as a controller, a decision maker, etc.. We see from subsection 2.2 that the output of the fuzzy inference engine is a fuzzy set $A \circ R$ in V; therefore, the defuzzifier maps $A \circ R$ into a crisp point $\underline{z} \in V$. In the literature, there are essentially two kinds of defuzzifiers [34] ((i) and (ii) below). We propose a new defuzzifier ((iii) below) whose justification will be given after the description of the defuzzifier. The three defuzzifiers are:

(i) *maximum defuzzifier*, defined as

$$\underline{z} = argsup_{\underline{z}' \in V}(\mu_{A \circ R}(\underline{z}')), \tag{1.13}$$

where $\mu_{A \circ R}(\underline{z}')$ is given by (1.12);

(ii) *centroid defuzzifier* (which is the most commonly used defuzzifier in the literature), defined as

$$z_j = \frac{\sum_{l=1}^{M} \bar{z}_j^l(\mu_{A \circ R_j^l}(\bar{z}_j^l))}{\sum_{l=1}^{M}(\mu_{A \circ R_j^l}(\bar{z}_j^l))}, \tag{1.14}$$

where $\bar{z}_j^l$ is the point in V_j at which $\mu_{G_j^l}(z_j)$ achieves its maximum value, $\mu_{A \circ R_j^l}(z_j)$ is given by (1.10), and $j = 1, 2, ..., m$; and,

(iii) *modified centroid defuzzifier*, defined as

$$z_j = \frac{\sum_{l=1}^{M} \bar{z}_j^l (\mu_{A \circ R_j^l}(\bar{z}_j^l)/\sigma_j^{l2})}{\sum_{l=1}^{M} (\mu_{A \circ R_j^l}(\bar{z}_j^l)/\sigma_j^{l2})}, \tag{1.15}$$

where σ_j^{l2} is a parameter characterizing the shape of $\mu_{G_j^l}(z_j)$ such that the narrower the shape of $\mu_{G_j^l}(z_j)$, the smaller is σ_j^{l2}; for example, if $\mu_{G_j^l}(z_j) = exp[-(\frac{z_j - \bar{z}_j^l}{\sigma_j^l})^2]$, then σ_j^{l2} is such a parameter.

The modified centroid defuzzifier is justified as follows. Common sense indicates that the sharper the shape of $\mu_{G_j^l}(z_j)$, the stronger is our belief that the output z_j should be nearer to $\bar{z}_j^l = argsup_{z_j' \in V_j}(\mu_{G_j^l}(z_j'))$ (according to the rule R_j^l of (1.9)). The standard centroid defuzzifier, (1.14), is a weighted sum of the $\bar{z}_j^l$'s, and the weight $\mu_{A \circ R_j^l}(\bar{z}_j^l)$ determined by (1.10) does not take the shape of $\mu_{G_j^l}(z_j)$ into consideration. This is clearly not satisfactory based on our common sense. An obvious improvement is the modified centroid defuzzifier (1.15).

Note that if we use the centroid or modified centroid defuzzifiers, we do not need to calculate the $\mu_{A \circ R_j}(z_j)$ of (1.11); we only need to calculate the $\mu_{A \circ R_j^l}(z_j)$ of (1.10) in the fuzzy inference engine.

1.2.5 Some Useful Subclasses of Fuzzy Systems

From subsections 1.2.1-1.2.4 we see that the fuzzy systems of Fig. 1.1 comprise a very rich class of static systems mapping from $U \subset R^n$ to $V \subset R^m$, because within each block there are many different choices, and many combinations of these choices can result in useful subclasses of fuzzy systems. Specifically, in the defuzzifier there are three choices: maximum, centroid, and modified centroid defuzzifiers; in the fuzzifier there are two choices: singleton and nonsingleton fuzzifiers; in the fuzzy inference engine the triangular norm $\star$ in (1.10) and the triangular co-norm $\dot{+}$ in (1.11) can take any operations in (1.1) and (1.2), respectively; and, in the fuzzy rule base we can choose different membership functions for the fuzzy sets F_i^l's and G_j^l's. If we view all the fuzzy systems as a set of functions, then each combination of these choices corresponds to a subset of the fuzzy systems, as shown for some examples in Fig. 1.2. We now show the specific

functional forms of four subclasses of fuzzy systems.

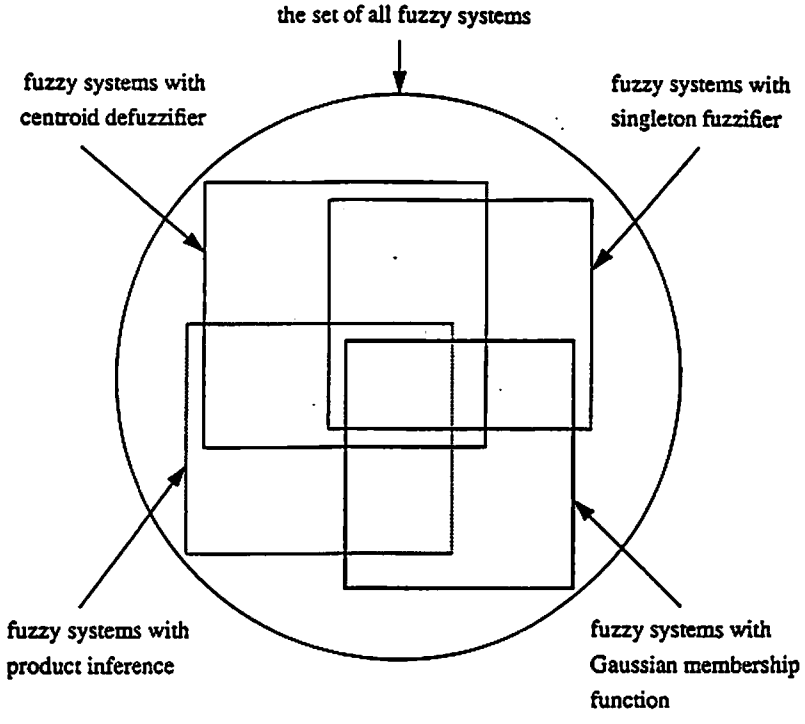


Figure 1.2: Some subclasses of fuzzy systems.

Definition 1.1: The set of fuzzy systems with *singleton fuzzifier, centroid defuzzifier, and minimum inference* is all functions $f = (f_1, ..., f_m)^T$ from $U \subset R^n$ to $V \subset R^m$ of the following form

$$f_j(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{z}_j^l min[\mu_{F_1^l}(x_1), \cdots, \mu_{F_p^l}(x_p), \mu_{G_j^l}(\bar{z}_j^l)]}{\sum_{l=1}^{M} min[\mu_{F_1^l}(x_1), \cdots, \mu_{F_p^l}(x_p), \mu_{G_j^l}(\bar{z}_j^l)]} \tag{1.16}$$

where $\underline{x} = (x_1, ..., x_n)^T \in U$, $1 \leq p \leq n$, and $\bar{z}_j^l$ is the point at which $\mu_{G_j^l}(z_j)$ achieves its maximum value. Usually, $\mu_{G_j^l}(\bar{z}_j^l) = 1$. Equation (1.16) is obtained by substituting (1.10) into (1.14), replacing $\star$ with the intersection in (1.1), and using the fact that if A is a fuzzy singleton with support $\underline{x}$ (i.e., $\mu_A(\underline{x}) = 1$ and $\mu_A(\underline{x}') = 0$ for all $\underline{x}' \neq \underline{x}$), then $\mu_{A \circ R_j^l}(z_j) = sup_{\underline{x}' \in U}[\mu_A(\underline{x}') \star \mu_{F_1^l}(x_1') \star \cdots \star \mu_{F_p^l}(x_p') \star \mu_{G_j^l}(z_j)]$ $= \mu_{F_1^l}(x_1) \star \cdots \star \mu_{F_p^l}(z_p) \star \mu_{G_j^l}(y_j)$.

Definition 1.2: The set of fuzzy systems with *singleton fuzzifier, centroid defuzzifier, and product inference* is all functions $f = (f_1, ..., f_m)^T$ from U to V of the following form

$$f_j(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{z}_j^l (\prod_{i=1}^{p} \mu_{F_i^l}(x_i)) \mu_{G_j^l}(\bar{z}_j^l)}{\sum_{l=1}^{M} (\prod_{i=1}^{p} \mu_{F_i^l}(x_i)) \mu_{G_j^l}(\bar{z}_j^l)}, \tag{1.17}$$

where the variables have the same meaning as those in (1.16). Clearly, (1.17) is obtained in the same way as (1.16), with *min* replaced by algebraic product.

Definition 1.3: The set of fuzzy systems with *singleton fuzzifier, modified centroid defuzzifier, product inference, and Gaussian membership function* is all functions $f = (f_1, ..., f_m)^T$ from U to V of the following form

$$f_j(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{z}_j^l [\prod_{i=1}^{p} a_i^l exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2)]/(\delta_j^l)^2}{\sum_{l=1}^{M} [\prod_{i=1}^{p} a_i^l exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2)]/(\delta_j^l)^2} \tag{1.18}$$

which is obtained by taking $\mu_{F_i^l}(x_i) = a_i^l exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2)$, $\mu_{G_j^l}(z_j) = exp(-\frac{1}{2}(\frac{z_j - \bar{z}_j^l}{\delta_i^l})^2)$, using the modifier centroid defuzzifier (1.15) with $\mu_{AoR_j^l}(\bar{z}_j^l)$ given by (1.10) and $\star$ = algebraic product, and noticing that $\mu_A(\underline{x}) = 1$ and $\mu_A(\underline{x}') = 0$ for all $\underline{x}' \neq \underline{x}$. In (1.18), $0 < a_i^l \leq 1$, $\sigma_i^l > 0$, $\delta_j^l > 0$, $\bar{x}_i^l \in U_i$, and $\bar{z}_j^l \in V_j$ are parameters.

Definition 1.4: The set of fuzzy systems with *singleton fuzzifier, centroid defuzzifier, product inference, and Gaussian membership function* is all functions $f = (f_1, ..., f_m)^T$ from U to V of the following form

$$f_j(\underline{x}) = \frac{\sum_{l=1}^{M} \bar{z}_j^l [\prod_{i=1}^{p} a_i^l exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2)]}{\sum_{l=1}^{M} [\prod_{i=1}^{p} a_i^l exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2)]}, \tag{1.19}$$

where the variables have the same meaning as those in (1.18).

Clearly, the set of the fuzzy systems in Definition1.4is a subset of the fuzzy systems in Definition1.3(taking all $\delta_j^l = 1$) and is also a subset of the fuzzy systems in Definition1.2(taking $\mu_{F_i^l}(x_i) = a_i^l exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2)$ and $\mu_{G_j^l}(\bar{z}_j^l) = 1$), i.e., $Y_4 \subset Y_3$, and $Y_4 \subset Y_2$, where Y_i is the set of the fuzzy systems in Definition 1.i, $i = 2, 3, 4$.

1.3 A Comparison of Fuzzy Systems with Probabilistic General Regression

Fuzziness versus probability is a well-known controversial issue. Probabilists claim that probability is the only satisfactory description of uncertainty, and all other descriptions, especially the possibility statements in fuzzy logic, are unnecessary [36,67]. Despite this criticism, fuzzy system researchers, led by Lotfi A. Zadeh, extended their studies to a formal representation of words in natural language using fuzzy sets [108], and a formal study of common sense reasoning, under the title of "fuzzy logic" [109, 110]. Because so much human knowledge is expressed in natural language, a formal study of it from an engineering point of view is clearly needed. The efforts of fuzzy system researchers were rewarded by the impressive successes of fuzzy logic controllers in a variety of industrial systems and commercial products. Indeed, fuzzy systems provide a friendly means to incorporate human knowledge into engineering systems, one that is simple to understand, easy to use, and fast to implement.

In this section, we first present a formal description of probabilistic general regression. Then, we show that probabilistic general regression is a special case of fuzzy systems, and extend this main observation by making a few remarks.

1.3.1 Probabilistic General Regression

Let $f(\underline{x}, z)$ be the joint probability density function of a vector random variable, $\underline{x} \in R^n$, and a scalar random variable, $z \in R$. The conditional mean of z given $\underline{x}$ (also called the regression of z on $\underline{x}$) is given by

$$E[z|\underline{x}] = \frac{\int_{-\infty}^{\infty} z f(\underline{x}, z) dz}{\int_{-\infty}^{\infty} f(\underline{x}, z) dz}. \tag{1.20}$$

Let $(\underline{\bar{x}}^l, \bar{z}^l)$, $l = 1, 2, ..., N$, be sample values of the random variables $\underline{x}$ and z; then, a consistent estimator of $f(\underline{x}, z)$ based upon the $(\underline{\bar{x}}^l, \bar{z}^l)$'s, which was proposed by Parzen [55] and was shown in [66] to be a good choice for estimating $f(\underline{x}, z)$, is

given by

$$\hat{f}(\underline{x},z)=\frac{1}{(2\pi)^{(n+1)/2}\sigma^{(n+1)}}\frac{1}{N}\sum_{l=1}^{N}exp[-\frac{(\underline{x}-\underline{\bar{x}}^l)^T(\underline{x}-\underline{\bar{x}}^l)}{2\sigma^2}]exp[-\frac{(z-\bar{z}^l)^2}{2\sigma^2}]. \tag{1.21}$$

Substituting (1.21) into (1.20) and performing the integration yields the following:

$$z(\underline{x})=\hat{E}(z|\underline{x})=\frac{\sum_{l=1}^{N}\bar{z}^l exp[-\frac{(\underline{x}-\underline{\bar{x}}^l)^T(\underline{x}-\underline{\bar{x}}^l)}{2\sigma^2}]}{\sum_{l=1}^{N}exp[-\frac{(\underline{x}-\underline{\bar{x}}^l)^T(\underline{x}-\underline{\bar{x}}^l)}{2\sigma^2}]}, \tag{1.22}$$

which is the probabilistic general regression. Analysis of this probabilistic general regression can be found in [65]; it has nice probabilistic properties, like consistency, etc..

1.3.2 Comparison of Fuzzy Systems with Probabilistic General Regression

Comparing (1.22) with (1.19), we see that they are almost the same. More specifically, we have the following:

Main Observation: The probabilistic general regression (1.22) is a special case of fuzzy systems. Specifically, within the subset of fuzzy systems in Definition 1.4, if we choose the parameters (in (1.19)): $M=N, a_i^l=1$, $\sigma_i^l=\sigma$ for all $i=1,2,...,n$ and $l=1,2,...,N$, $\bar{x}_i^l=$ the i'th element of the sample vector $\underline{\bar{x}}^l$, $\bar{z}^l=$ the sample $\bar{z}^l$, then the fuzzy system (1.19) becomes the probabilistic general regression (1.22) (note that $\prod_{i=1}^{n}exp[-\frac{1}{2}(\frac{x_i-\bar{x}_i^l}{\sigma})^2]=exp[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\bar{x}_i^l)^2]=exp[-\frac{(\underline{x}-\underline{\bar{x}}^l)^T(\underline{x}-\underline{\bar{x}}^l)}{2\sigma^2}]$).

We now extend this Main Observation by making a few remarks.

Remark 1.1: Fuzzy systems are constructed and justified based on fuzzy logic; very little is known about their statistical properties when they are used in a random environment. From this Main Observation we see that although there is no statistical consideration when constructing such a fuzzy system, the resulting fuzzy system turns out to be "optimal" from a statistical point of view.

Remark 1.2: From Section 1.2 we see that the fuzzy systems in Definition 1.4 are only a very small subset of fuzzy systems; and, from the Main Observation we

see that the probabilistic general regression is a special case in this small subset. Therefore, there remains a huge number of fuzzy systems for which a statistical study is needed. Up to now, it seems that the most successful application of fuzzy systems is to the control of industrial processes where random noise always exists. Therefore, knowing the statistical properties of various kinds of fuzzy systems is very important. There are many fuzzy systems that are quite different from the probabilistic general regression, e.g., the fuzzy systems with min inference and triangular membership function, and that have found successful practical applications. We hope that these fuzzy systems also have nice statistical properties.

Remark 1.3: Although the Main Observation shows the similarity between fuzzy systems and the probabilistic general regression, it should be emphasized that they are quite different from many fundamental points of view. For example, fuzzy systems are constructed from fuzzy IF-THEN rules, whereas the probabilistic general regression is constructed from sample data pairs $(\bar{x}^l, \bar{y}^l)$. It is not difficult to generate reasonable fuzzy IF-THEN rules based on the sample pairs, e.g., the method in the Main Observation, and some other methods in [82-86]; however, it is very difficult to generate reasonable data pairs based on fuzzy IF-THEN rules, because fuzzy IF-THEN rules characterize fuzzy relationships between $\underline{x}$ and y so that if we want to generate data pairs based on them, we have to use some fuzzification, defuzzification, and inference strategies which will have the same influence on the generated data pairs as the original fuzzy IF-THEN rules.

Remark 1.4: Perhaps the most fundamental difference between fuzzy systems and the probabilistic general regression is that: fuzzy systems provide a very good framework to combine linguistic information in natural language (in the form of fuzzy IF-THEN rules) and measured numerical information (in the form of sample data pairs) in a uniform fashion, whereas the probabilistic general regression can only make use of the numerical information. More specifically, fuzzy IF-THEN rules can be generated based on the sample pairs as in the Main Observation, or using the methods in [82-86], and these rules can be combined with the linguistic rules to form the final fuzzy system, which is therefore constructed based upon both linguistic information and numerical information in a uniform fashion.

1.3.3 Concluding Remarks

In this section, we showed that probabilistic general regression is a special case of fuzzy systems, i.e., given the same set of information (sample pairs), we can construct a fuzzy system (using fuzzy logic principles) which is exactly the same as the probabilistic general regression. We showed that fuzzy systems are more general than the probabilistic general regression not only in the functional form, but also from a more fundamental capability point of view — fuzzy systems can effectively combine both linguistic information in natural language and measured numerical information in a uniform fashion, whereas probabilistic general regression can only make use of measured numerical information.

Bayesian statisticians believe that probability is sufficient to represent any kind of uncertainty; therefore, human experts should express their knowledge about uncertainty events in terms of (conditional) probabilities [36]. However, there is so much human knowledge about uncertain events that is expressed in natural language, which is far from easy to "translate" into probabilistic terms. Fuzzy system researchers have found fuzzy systems a very *friendly* tool to represent linguistic information in natural language, and have proven the usefulness of fuzzy systems by successfully applying them to a variety of practical problems. From this section we see that the superior performance of fuzzy systems is not a surprise, because fuzzy systems include the "optimal" probabilistic general regression as a special case; therefore, by carefully searching the whole space of fuzzy systems, the performance of the resulting fuzzy system should be no worse than the probabilistic general regression. A very interesting research topic is to rigorously study various kinds of fuzzy systems from a probabilistic point of view, so that fuzzy system researchers can make use of the wonderful treasure created by probabilists during the last four hundred years.

We conclude this section by quoting what L. Ljung wrote in [39]:

"$\cdots$ An issue in system identification is to have an open mind about nonlinear black-box structures; to try out the above [Volterra series, neural nets, general regressions, etc.] along with many other ideas, like $\cdots$"

1.4 Outline of the Thesis

In Chapter 2, we prove, using the Stone-Weierstrass Theorem, that the fuzzy systems described in Section 1.2 are universal approximators, i.e, they are capable of approximating any nonlinear function over a compact set to arbitrary accuracy. We then extend this result to a dynamic system, i.e., we prove that dynamic systems based on fuzzy systems can follow the output of a very general nonlinear dynamic system to arbitrary accuracy within any finite time interval.

In Chapter 3, we show that fuzzy systems can be represented as three-layer feedforward networks, and develop a back-propagation algorithm to train the fuzzy systems to match desired input-output pairs. We then use this back-propagation fuzzy system as identifiers for nonlinear dynamic systems.

A weakpoint of the back-propagation algorithm developed in Chapter 2 is that it performs a nonlinear search procedure and therefore may converge slowly or be trapped at a local minimum. To overcome this weakpoint, we develop another method for fuzzy system design, in Chapter 4, that uses the classical orthogonal least squares algorithm. We apply this method to the control of the nonlinear ball and beam system, and to predict a chaotic time series.

The methods in Chapters 3 and 4 are not simple, in the sense that they require iterative computations. We therefore develop a third method, in Chapter 5, that performs a one-pass operation on the data pairs and does not require iterative computation. We apply this method to the truck backer-upper control and chaotic time-series prediction problems, and compare the results with those using conventional fuzzy and neural approaches.

In Chapter 6, we develop two nonlinear adaptive filters based on fuzzy systems: RLS and LMS fuzzy adaptive filters, and use them as nonlinear channel equalizers.

Chapter 7 concludes the thesis and points out some future work.

Chapter 2

FUZZY SYSTEMS AS UNIVERSAL APPROXIMATORS TO STATIC AND DYNAMIC SYSTEMS

2.1 Introduction

Fuzzy systems have been successfully applied to a wide variety of practical problems. Notable applications of fuzzy systems include the control of: warm water [24], robot [21,71], heat exchanger [53], traffic junction [54], cement kiln [32], activated sludge [22], automobile speed [46], automatic train operation systems [103], model-car parking and turning [69], turning [61], aircraft [9], water purification [100], automatic container crane operation systems [105], elevator [15], automobile transmission [23], and power systems and nuclear reactor [2]. Recent advances of fuzzy memory devices and fuzzy chips [75,97] make fuzzy systems especially suitable for industrial applications.

A very fundamental theoretical question about fuzzy systems remains unanswered, namely: "Why does a fuzzy system have such excellent performance for

such a wide variety of applications ?" Existing explanations are qualitative, e.g., "fuzzy systems can utilize linguistic information from human experts," "fuzzy systems can simulate human thinking procedure," "fuzzy systems capture the approximate and inexact nature of the real world," etc.. In this chapter, we try to answer this fundamental question by proving that fuzzy systems are universal approximators, i.e., they are capable of approximating any real continuous function on a compact set to arbitrary accuracy. We use the famous Stone-Weierstrass Theorem [59] to prove this fundamental result. This result can be viewed as an existence theorem of an optimal fuzzy system for a wide variety of problems.

2.2 Fuzzy Systems as Universal Approximators of Static Systems

We have the following main result:

THEOREM 2.1: For any given real continuous function g on the compact set $U \subset R^n$ and arbitrary $\epsilon > 0$, there exists $f \in Y_4$ (Y_4 is defined in Definition 1.4) such that

$$sup_{\underline{x} \in U} |g(\underline{x}) - f(\underline{x})| < \epsilon. \tag{2.1}$$

We use the following Stone-Weierstrass Theorem to prove Theorem 2.1.

Stone-Weierstrass Theorem [59]: 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 $\underline{x}, \underline{y} \in U, \underline{x} \neq \underline{y}$, there exists $f \in Z$ such that $f(\underline{x}) \neq f(\underline{y})$; and, 3) Z *vanishes at no point of* U, i.e., for each $\underline{x} \in U$ there exists $f \in Z$ such that $f(\underline{x}) \neq 0$; then, the uniform closure of Z consists of all real continuous functions on U, i.e., (Z, d_∞) is dense in $(C[U], d_\infty)$.

In order to use the Stone-Weierstrass Theorem to prove Theorem 2.1, we need to show that Y_4 is an algebra, Y_4 separates points on U, and, Y_4 vanishes at no point of U. The following Lemmas 2.1-2.3 prove that Y_4 has these properties.

LEMMA 2.1: (Y_4, d_∞) is an algebra.

Proof: Let $f_1, f_2 \in Y_4$, so that we can write them as

$$f_1(\underline{x}) = \frac{\sum_{j=1}^{K1}(\bar{z1}^j \Pi_{i=1}^n \mu_{A1_i^j}(x_i))}{\sum_{j=1}^{K1}(\Pi_{i=1}^n \mu_{A1_i^j}(x_i))}, \tag{2.2}$$

$$f_2(\underline{x}) = \frac{\sum_{j=1}^{K2}(\bar{z2}^j \Pi_{i=1}^n \mu_{A2_i^j}(x_i))}{\sum_{j=1}^{K2}(\Pi_{i=1}^n \mu_{A2_i^j}(x_i))}; \tag{2.3}$$

hence,

$$f_1(\underline{x}) + f_2(\underline{x}) = \frac{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\bar{z1}^{j1} + \bar{z2}^{j2})(\Pi_{i=1}^n \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\Pi_{i=1}^n \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}. \tag{2.4}$$

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, (2.4) is the same form as (1.19), so that $f_1 + f_2 \in Y_4$. Similarly,

$$f_1(\underline{x})f_2(\underline{x}) = \frac{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\bar{z1}^{j1}\bar{z2}^{j2})(\Pi_{i=1}^n \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\Pi_{i=1}^n \mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}, \tag{2.5}$$

which is also in the same form of (1.19); hence, $f_1 f_2 \in Y$. Finally, for arbitrary $c \in R$,

$$cf_1(\underline{x}) = \frac{\sum_{j=1}^{K1}(c\bar{z1}^j)(\Pi_{i=1}^n \mu_{A1_i^j}(x_i))}{\sum_{j=1}^{K1}(\Pi_{i=1}^n \mu_{A1_i^j}(x_i))}, \tag{2.6}$$

which is again in the form of (1.19); hence, $cf_1 \in Y$. Q.E.D..

LEMMA 2.2: (Y_4, d_∞) separates points on U.

Proof: We prove this by constructing a required f, i.e, we specify the number of fuzzy sets defined in U and R, the parameters of the Gaussian membership functions, the number of fuzzy rules, and the statements of fuzzy rules, such that the resulting f (in the form of (1.19)) has the property that $f(\underline{x}^0) \neq f(\underline{y}^0)$ for arbitrarily given $\underline{x}^0, \underline{y}^0 \in U$ with $\underline{x}^0 \neq \underline{y}^0$. Let $\underline{x}^0 = (x_1^0, x_2^0, ..., x_n^0)$ and $\underline{y}^0 = (y_1^0, y_2^0, ..., y_n^0)$. If $x_i^0 \neq y_i^0$, we define two fuzzy sets, $(A_i^1, \mu_{A_i^1})$ and $(A_i^2, \mu_{A_i^2})$, in the

$i'th$ subspace of U, with

$$\mu_{A_i^1}(x_i) = exp[-\frac{(x_i - x_i^0)^2}{2}], \tag{2.7}$$

$$\mu_{A_i^2}(x_i) = exp[-\frac{(x_i - y_i^0)^2}{2}]. \tag{2.8}$$

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 in the $i'th$ subspace of U. We define two fuzzy sets, (B^1, μ_{B^1}) and (B^2, μ_{B^2}), in the output universe of discourse R, with

$$\mu_{B^j}(z) = exp[-\frac{(z - \bar{z}^j)^2}{2}], \tag{2.9}$$

where $j = 1, 2$, and $\bar{z}^j$ will be specified later. We choose two fuzzy rules for the fuzzy rule base (i.e., M=2). 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 (1.19) with $M = 2$. With this f, we have

$$f(\underline{x}^0) = \frac{\bar{z}^1 + \bar{z}^2 \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}{1 + \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]} = \alpha \bar{z}^1 + (1 - \alpha)\bar{z}^2, \tag{2.10}$$

$$f(\underline{y}^0) = \frac{\bar{z}^2 + \bar{z}^1 \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}{1 + \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]} = \alpha \bar{z}^2 + (1 - \alpha)\bar{z}^1, \tag{2.11}$$

where

$$\alpha = \frac{1}{1 + \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}. \tag{2.12}$$

Since $\underline{x}^0 \neq \underline{y}^0$, there must be some i such that $x_i^0 \neq y_i^0$; hence, we have $\Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2] \neq 1$, or, $\alpha \neq 1 - \alpha$. If we choose $\bar{z}^1 = 0$ and $\bar{z}^2 = 1$, then $f(\underline{x}^0) = 1 - \alpha \neq \alpha = f(\underline{y}^0)$. Q.E.D..

LEMMA 2.3: (Y_4, d_∞) vanishes at no point of U.

Proof: By observing (3) and (2), we simply choose all $\bar{z}^j > 0$ $(j = 1, 2, ..., M)$, i.e., any $f \in Y_4$ with $\bar{z}^j > 0$ serves as the required f. Q.E.D..

Proof of Theorem 2.1: From (1.19), it is obvious that Y_4 is a set of real continuous functions on U. Theorem 2.1 is therefore a direct consequence of the Stone-Weierstrass Theorem and Lemmas 2.1-2.3. Q.E.D..

Theorem 2.1 shows that the fuzzy systems in Y_4 can approximate continuous functions. The following corollary generalizes the result of Theorem 2.1 to discrete functions.

COROLLARY 2.1: For any $g \in L_2(U)$ and arbitrary $\epsilon > 0$, there exists $f \in Y_4$ such that

$$(\int_U |f(\underline{x}) - g(\underline{x})|^2 d\underline{x})^{1/2} < \epsilon, \tag{2.13}$$

where $U \subset R^n$ is compact, $L_2(U) = [g : U \to R | \int_U |g(\underline{x})|^2 d\underline{x} < \infty]$, and the integrals are in the Lebesgue sense.

Proof: Since U is compact, $\int_U d\underline{x} = V < \infty$. Since continuous functions on U form a dense subset of $L_2(U)$ [59], for any $g \in L_2(U)$ there exists a continuous function $\bar{g}$ on U such that $(\int_U |g(\underline{x}) - \bar{g}(\underline{x})|^2 d\underline{x})^{1/2} < \epsilon/2$. By Theorem 1, there exists $f \in Y_4$ such that $sup_{\underline{x} \in U} |f(\underline{x}) - \bar{g}(\underline{x})| < \epsilon/(2V^{1/2})$; hence, we have

$$\begin{aligned}
(\int_U |f(\underline{x}) - g(\underline{x})|^2 d\underline{x})^{1/2} &\leq (\int_U |f(\underline{x}) - \bar{g}(\underline{x})|^2 d\underline{x})^{1/2} + (\int_U |\bar{g}(\underline{x}) - g(\underline{x})|^2 d\underline{x})^{1/2} \\
&< (\int_U (sup_{\underline{x} \in U} |f(\underline{x}) - \bar{g}(\underline{x})|)^2 d\underline{x})^{1/2} + \epsilon/2 \\
&< (\frac{\epsilon^2}{2^2 V} V)^{1/2} + \epsilon/2 = \epsilon.
\end{aligned} \tag{2.14}$$

Q.E.D..

2.3 Fuzzy Systems as Universal Approximators of Dynamic Systems

In this section we study the capability of using the fuzzy systems in Definition 1.4 to approximate dynamic systems over a finite time interval. Consider the dynamic system

$$\dot{\underline{x}} = A\underline{x} + g(\underline{x}, \underline{u}) \tag{2.15}$$

$$\underline{y} = h(\underline{x}, \underline{u}), \tag{2.16}$$

where $\underline{u} \in U$ is the input, $\underline{y} \in V$ is the output, $\underline{x} \in W \subset R^r$ is the state, A is a known Hurwitz matrix, and g, h are unknown continuous functions. We make

the following assumption.

Assumption 2.1: For any $\underline{u}$ in compact U and any finite initial condition $\underline{x}(0) = \underline{x}^0$, the solution $\underline{x}(t)$ of (2.15) satisfies $|\underline{x}(t) - \underline{x}^0| \leq b$ for some positive constant b and all $t \in [0, T]$, where T is an arbitrary positive constant.

Define K to be the set

$$K = [(\underline{x}, \underline{u}) \in R^{r+n} : |\underline{x} - \underline{x}^0| \leq b + \epsilon, \underline{u} \in U], \tag{2.17}$$

where $\epsilon > 0$ is an arbitrary constant, and $\underline{x}^0$ is any finite initial condition. Because U is compact and $|\underline{x}| \leq |\underline{x}^0| + b + \epsilon < \infty$, K is compact. Assumption 2.1 assures us that for $\underline{u} \in U$ and finite $\underline{x}^0$, (2.15) generates $(\underline{x}(t), \underline{u}(t)) \in K$ for $t \in [0, T]$. Let $\hat{g}(\underline{x}, \underline{u}|\Theta_g) = (\hat{g}_1(\underline{x}, \underline{u}|\Theta_g), ..., \hat{g}_r(\underline{x}, \underline{u}|\Theta_g))^T$ and $\hat{h}(\underline{x}, \underline{u}|\Theta_h) = (\hat{h}_1(\underline{x}, \underline{u}|\Theta_h), ..., \hat{h}_m(\underline{x}, \underline{u}|\Theta_h))^T$ be the fuzzy systems in Definition 1.4, where Θ_g, Θ_h are collections of the parameters a_i^l, $\bar{x}_i^l$, σ_i^l and $\bar{z}_j^l$. Define Θ_g^* and Θ_h^* such that

$$sup_{\underline{x},\underline{u} \in K} |\hat{g}(\underline{x}, \underline{u}|\Theta_g^*) - g(\underline{x}, \underline{u})| < \epsilon_g \tag{2.18}$$

$$sup_{\underline{x},\underline{u} \in K} |\hat{h}(\underline{x}, \underline{u}|\Theta_h^*) - h(\underline{x}, \underline{u})| < \epsilon_h \tag{2.19}$$

for some arbitrary $\epsilon_g > 0$ and $\epsilon_h > 0$. Theorem 2.1 assures the existence of the Θ_g^* and Θ_h^*. We now show that by replacing g and h in (2.15) and (2.16) with $\hat{g}$ and $\hat{h}$, we obtain a dynamic system whose output can approximate the output of (2.15) and (2.16) to arbitrary accuracy over any finite interval of time.

THEOREM 2.2: Consider the dynamic system

$$\dot{\underline{\hat{x}}} = A\underline{\hat{x}} + \hat{g}(\underline{\hat{x}}, \underline{u}|\Theta_g^*) \tag{2.20}$$

$$\underline{\hat{y}} = \hat{h}(\underline{\hat{x}}, \underline{u}|\Theta_h^*) \tag{2.21}$$

where $\underline{\hat{x}}(0) = \underline{x}(0) = \underline{x}^0$, $\underline{u} \in U$, $\hat{g}$ and $\hat{h}$ are the fuzzy systems in Definition 1.4, and the parameters Θ_g^* and Θ_h^* are defined in (2.18) and (2.19). Then, for any $\epsilon > 0$, finite $T > 0$, and properly chosen ϵ_g and ϵ_h, we have that

$$sup_{t \in [0,T]} |\underline{\hat{y}}(t) - \underline{y}(t)| < \epsilon. \tag{2.22}$$

Proof: From (1.19) we see that $\hat{g}$ and $\hat{h}$ are continuous functions and therefore satisfy the Lipschitz condition in the compact set K of (2.17), i.e., there exist constants b_g and b_h such that for all $(\underline{x}^{(1)}, \underline{u}), (\underline{x}^{(2)}, \underline{u}) \in K$

$$|\hat{g}(\underline{x}^{(1)}, \underline{u}|\Theta_g) - \hat{g}(\underline{x}^{(2)}, \underline{u}|\Theta_g)| \leq b_g|\underline{x}^{(1)} - \underline{x}^{(2)}|, \tag{2.23}$$

$$|\hat{h}(\underline{x}^{(1)}, \underline{u}|\Theta_h) - \hat{h}(\underline{x}^{(2)}, \underline{u}|\Theta_h)| \leq b_h|\underline{x}^{(1)} - \underline{x}^{(2)}|. \tag{2.24}$$

Define $\underline{e}_x \equiv \hat{\underline{x}} - \underline{x}$; then, from (2.15) and (2.20) we have

$$\dot{\underline{e}}_x = A\underline{e}_x + \hat{g}(\hat{\underline{x}}, \underline{u}|\Theta_g^*) - g(\underline{x}, \underline{u}), \tag{2.25}$$

whose solution can be expressed as

$$\underline{e}_x(t) = \int_0^t e^{A(t-\tau)}[\hat{g}(\hat{\underline{x}}(\tau), \underline{u}(\tau)|\Theta_g^*) - g(\underline{x}(\tau), \underline{u}(\tau))]d\tau. \tag{2.26}$$

Since A is a Hurwitz matrix, there exist positive constants c and α such that $||e^{At}|| \leq ce^{-\alpha t}$ for all $t \geq 0$. Let $\epsilon_g = \frac{c\alpha}{2cb_h}e^{-\frac{cb_g}{\alpha}}$, then from (2.26), (2.23) and (2.18) we have that

$$\begin{aligned}
|\underline{e}_x(t)| &\leq \int_0^t ||e^{A(t-\tau)}|||\hat{g}(\hat{\underline{x}}(\tau), \underline{u}(\tau)|\Theta_g^*) - \hat{g}(\underline{x}(\tau), \underline{u}(\tau)|\Theta_g^*)|d\tau \\
&\quad + \int_0^t ||e^{A(t-\tau)}|||\hat{g}(\underline{x}(\tau), \underline{u}(\tau)|\Theta_g^*) - g(\underline{x}(\tau), \underline{u}(\tau))|d\tau \\
&\leq \int_0^t ce^{-\alpha(t-\tau)}b_g|\underline{e}_x(\tau)|d\tau + \int_0^t ce^{-\alpha(t-\tau)}\frac{\epsilon\alpha}{2cb_h}e^{-\frac{cb_g}{\alpha}}d\tau \\
&\leq cb_g\int_0^t e^{-\alpha(t-\tau)}|\underline{e}_x(\tau)|d\tau + \frac{\epsilon}{2b_h}e^{-\frac{cb_g}{\alpha}}.
\end{aligned} \tag{2.27}$$

Using the Bellman-Gronwall Lemma [17], we obtain

$$\begin{aligned}
|\underline{e}_x(t)| &\leq \frac{\epsilon}{2b_h}e^{-\frac{cb_g}{\alpha}}e^{cb_g\int_0^t e^{-\alpha(t-\tau)}d\tau} \\
&\leq \frac{\epsilon}{2b_h}e^{-\frac{cb_g}{\alpha}[1+\int_0^t e^{-\alpha(t-\tau)}d\alpha(t-\tau)]} \leq \frac{\epsilon}{2b_h}.
\end{aligned} \tag{2.28}$$

Therefore, $|\hat{\underline{x}} - \underline{x}^0| \leq |\hat{\underline{x}} - \underline{x}| + |\underline{x} - \underline{x}^0| \leq \frac{\epsilon}{2b_h} + b$ from Assumption 2.1. Without loss of generality, we assume that $b_h \geq 1$; therefore, $(\hat{\underline{x}}, \underline{u}) \in K$. Hence, letting $\epsilon_h = \epsilon/2$, using (2.19), (2.24) and (2.28), and considering the fact that $(\underline{x}, \underline{u}) \in K$

and $(\underline{\hat{x}}, \underline{u}) \in K$, we obtain

$$\begin{aligned} |\underline{\hat{y}}(t) - \underline{y}(t)| &\leq |\hat{h}(\underline{\hat{x}}, \underline{u}|\Theta_h^*) - \hat{h}(\underline{x}, \underline{u}|\Theta_h^*)| + |\hat{h}(\underline{x}, \underline{u}|\Theta_h^*) - h(\underline{x}, \underline{u})| \\ &\leq b_h|\underline{e}_x(t)| + \epsilon/2 \leq \epsilon \end{aligned} \tag{2.29}$$

for all $t \in [0, T]$. Q.E.D..

Chapter 3

DESIGN OF FUZZY SYSTEMS USING BACK-PROPAGATION TRAINING

3.1 Introduction

Feedforward neural networks were successfully used in [49] as identifiers for nonlinear components in dynamic systems. Theoretical justification of this approach is that feedforward neural networks can approximate any real continuous function on a compact set to arbitrary accuracy [12,20]. The back-propagation training algorithm [60,95] makes it possible to train the neural network identifiers on-line to match unknown nonlinear mappings.

In Chapter 2 we proved that fuzzy systems are also universal approximators. If we can develop a back-propagation algorithm for fuzzy systems, similar to the back-propagation algorithm for neural networks, then we can use the back-propagation fuzzy systems as identifiers for nonlinear dynamic systems. In this chapter, we develop such a back-propagation algorithm for fuzzy systems. The key ideas in developing this training algorithm are to view a fuzzy system as a three-layer feedforward network, and to use the chain rule to determine gradients of the output errors of the fuzzy system with respect to its design parameters. We show that this training algorithm performs an error back-propagation procedure; hence,

we call the fuzzy system equipped with the back-propagation training algorithm a "Back-Propagation Fuzzy System (BP FS)."

How does the BP FS compare with a back-propagation feedforward neural network (BP FNN) when they are used as identifiers for nonlinear dynamic systems? They are similar in that both: (1) are universal approximators, and therefore qualify as identifiers for nonlinear systems; and, (2) use back-propagation training algorithms to adjust their parameters for the purpose of matching desired input-output pairs.

There are two important advantages of a BP FS over a BP FNN. First, the parameters of a BP FS have clear physical meanings; hence (as we show), it is possible to develop a very good method for choosing its initial parameters. On the other hand, the parameters of a BP FNN have no clear relationships with input-output data, and therefore their initial values are usually chosen randomly. Because both BP training algorithms are gradient algorithms, good initial parameters dramatically speeds convergence. The second advantage of a BP FS over a BP FNN, which may be more essential than the first one, is that a BP FS provides a natural framework in which to incorporate human linguistic descriptions (in the form of IF-THEN rules) about the unknown nonlinear system. Frequently, linguistic (subjective) information is difficult to quantify, and is ignored at the front-end of identifier designs [16,38]. It is used to help evaluate such designs. Using the results of this paper, the identifier designer will be able to include subjective information at the front-end of the design, where we believe it will do the most good.

3.2 Back-Propagation Training for Fuzzy Systems

Consider the fuzzy systems in Definition 1.4, i.e.,

$$f(\underline{x}) = \frac{\sum_{j=1}^{M} \bar{z}^j (\prod_{i=1}^{n} a_i^j exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^j}{\sigma_i^j})^2))}{\sum_{j=1}^{M} [\prod_{i=1}^{n} a_i^j exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^j}{\sigma_i^j})^2)]} \tag{3.1}$$

By analysing (3.1), we observe a very important fact: the mapping $f : U \rightarrow R$ determined by (3.1) can be represented as a three-layer feedforward network, as shown in Fig. 3.1. Now we consider the following problem: Given an input-output pair $(\underline{x}^p, d^p)$, $\underline{x}^p \in U \subset R^n$ and $d^p \in R$, design a fuzzy system f in the form of (3.1) such that

$$e^p = \frac{1}{2}[f(\underline{x}^p) - d^p]^2 \tag{3.2}$$

is minimized, where by "designing a f" we mean to specify the parameters of f, i.e., to specify $M, \bar{z}^j, a_i^j$, $\bar{x}_i^j$ and σ_i^j for $i = 1, 2, ..., n$ and $j = 1, 2, ..., M$. In this chapter, we consider the case where M and a_i^j are fixed. In fact, choosing M can be viewed as an order determination problem [16], and, choosing $\bar{z}^j, a_i^j$, $\bar{x}_i^j$ and σ_i^j can be viewed as a parameter estimation problem. We only consider the parameter estimation problem. Additionally, we fix $a_i^j \equiv 1$ because for practical problems it is reasonable to assume that every fuzzy membership function achieves unity membership value at some point. We now develop an error back-propagation training algorithm to determine the remaining parameters $\bar{z}^j, \bar{x}_i^j$ and σ_i^j. We will use e, f and d to denote $e^p, f(\underline{x}^p)$ and d^p, respectively.

To train $\bar{z}^j$, we use

$$\bar{z}^j(k+1) = \bar{z}^j(k) - \alpha \frac{\partial e}{\partial \bar{z}^j}|_k, \tag{3.3}$$

where $j = 1, 2, ..., M$, $k = 0, 1, 2, ...$, and α is a constant stepsize. From Fig. 3.1 we see that f (and hence e) depends on $\bar{z}^j$ only through a, where $f = a/b$, $a = \sum_{j=1}^{M} (\bar{z}^j y^j)$, $b = \sum_{j=1}^{M} y^j$, and $y^j = \prod_{i=1}^{M} exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^j}{\sigma_i^j})^2)$; hence, using the chain rule, we have

$$\frac{\partial e}{\partial \bar{z}^j} = (f - d)\frac{\partial f}{\partial a}\frac{\partial a}{\partial \bar{z}^j} = (f - d)\frac{1}{b}y^j. \tag{3.4}$$

Substituting (3.4) into (3.3), we obtain the training algorithm for $\bar{z}^j$:

$$\bar{z}^j(k+1) = \bar{z}^j(k) - \alpha \frac{f - d}{b} y^j, \tag{3.5}$$

where $j = 1, 2, ..., M$, and $k = 0, 1, 2, ...$

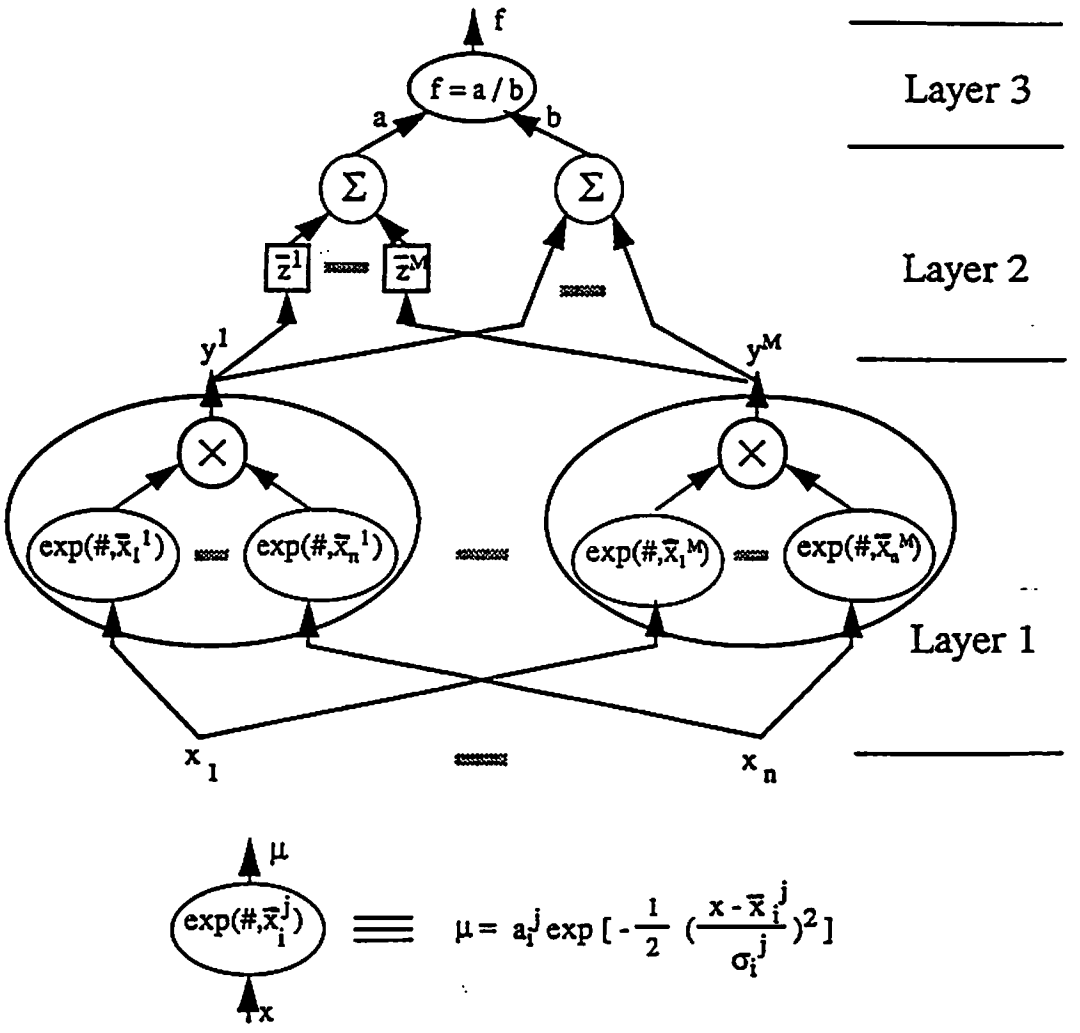


Figure 3.1: Network representation of the fuzzy systems.

To train $\bar{x}_i^j$, we use

$$\bar{x}_i^j(k+1) = \bar{x}_i^j(k) - \alpha \frac{\partial e}{\partial \bar{x}_i^j}|_k, \tag{3.6}$$

where $i = 1, 2, ..., n, j = 1, 2, ..., M$, and $k = 0, 1, 2, ...$ We see from Fig. 3.1 that f (and hence e) depends on $\bar{x}_i^j$ only through y^j; hence, using the chain rule, we have

$$\frac{\partial e}{\partial \bar{x}_i^j} = (f-d)\frac{\partial f}{\partial y^j}\frac{\partial y^j}{\partial \bar{x}_i^j} = (f-d)\frac{\bar{z}^j - f}{b} y^j \frac{x_i^p - \bar{x}_i^j}{\sigma_i^{j2}}. \tag{3.7}$$

Substituting (3.7) into (3.6), we obtain the training algorithm for $\bar{x}_i^j$:

$$\bar{x}_i^j(k+1) = \bar{x}_i^j(k) - \alpha \frac{f-d}{b}(\bar{z}^j - f) y^j \frac{x_i^p - \bar{x}_i^j(k)}{\sigma_i^{j2}(k)}, \tag{3.8}$$

where $i = 1, 2, ..., n, j = 1, 2, ..., M$, and $k = 0, 1, 2, ...$

Using the same method as above, we obtain the following training algorithm for σ_i^j:

$$\begin{aligned} \sigma_i^j(k+1) &= \sigma_i^j(k) - \alpha \frac{\partial e}{\partial \sigma_i^j}|_k \\ &= \sigma_i^j(k) - \alpha \frac{f-d}{b}(\bar{z}^j - f) y^j \frac{(x_i^p - \bar{x}_i^j(k))^2}{\sigma_i^{j3}(k)}, \end{aligned} \tag{3.9}$$

where $i = 1, 2, ..., n, j = 1, 2, ..., M$, and $k = 0, 1, 2, ...$

The training algorithm (3.5), (3.8) and (3.9) performs an error back-propagation procedure: to train $\bar{z}^j$, the "normalized" error $(f-d)/b$ is back-propagated to the layer of $\bar{z}^j$, then $\bar{z}^j$ is updated using (3.5) in which y^j is the input to $\bar{z}^j$ (see Fig. 3.1); to train $\bar{x}_i^j$ and σ_i^j, the "normalized" error $(f-d)/b$ times $(\bar{z}^j - f)$ and y^j is back-propagated to the processing unit of Layer 1 whose output is y^j; then, $\bar{x}_i^j$ and σ_i^j are updated using (3.8) and (3.9) respectively in which the remaining variables $\bar{x}_i^j, x_i^p$ and σ_i^j (i.e., the variables on the right-hand sides of (3.8) and (3.9), except the back-propagated error $\frac{f-\hat{f}}{b}(\bar{z}^j - f)y^j$) can be obtained locally.

The training procedure for the fuzzy system of Fig. 3.1 is a two-pass procedure: first, for a given input $\underline{x}^p$, compute forward along the network (i.e., the fuzzy system) to obtain y^j $(j = 1, 2, ..., M)$, a, b and f; then, train the network

parameters $\bar{z}^j, \bar{x}_i^j$ and σ_i^j $(i = 1, 2, ..., n, j = 1, 2, ..., M)$ backwards using (3.5), (3.8) and (3.9), respectively.

3.3 Identification of Nonlinear Dynamic Systems Using the Back-Propagation Fuzzy Systems

Theorem 2.1 assures us that the fuzzy systems with centroid defuzzification can approximate any nonlinear real continuous function on the compact set $U \subset R^n$ to arbitrary accuracy; and, the back-propagation training algorithm developed in the last section gives us a practical way to train the fuzzy system to match desired input-output pairs. These provide the justification to use back-propagation fuzzy systems as identifiers for nonlinear dynamic systems. Additionally, the fuzzy systems have two important features which make them even more attactive as identifiers for nonlinear systems.

First, the parameters of the fuzzy systems, $\bar{z}^j, \bar{x}_i^j$ and σ_i^j, have clear physical meanings. Specifically, $\bar{x}_i^j$ and $\bar{z}^j$ are points at which the membership functions of the fuzzy sets defined in the input and output spaces achieve their maximum values, respectively, and, σ_i^j characterize the shape of the input membership functions. Based on these physical meanings, we can develop a very good method for choosing their initial values, as described later in this section. Because the back-propagation algorithm is a gradient algorithm, a good choice of initial parameters speeds up convergence dramatically.

The second attractive feature of fuzzy systems, which may be more important and essential than the first one, is that the fuzzy systems provide a natural framework to incorporate linguistic descriptions about the unknown nonlinear systems into the identifiers. Specifically, *we construct the initial identifier based on linguistic rules, and then update the identifier based on numerical data pairs*; in this way, we combine linguistic and numerical information into the fuzzy identifier in a uniform fashion. This feature has practical importance because many real world nonlinear systems are controlled by human experts (e.g., aircraft, power systems,

economic systems, etc.), and, these experts can provide linguistic descriptions about the nonlinear systems. These linguistic descriptions are vague and fuzzy, so that traditional identifiers [16, 38] and neural identifiers [49] cannot make use of them at the front-end of their designs. They are used only to help evaluate such designs. The BP FS provides an identifier which can make use of both numerical information (in the form of input-output pairs) and linguistic information (in the form of IF-THEN rules) at the front-end of its design in an uniform manner.

Based on the discussion in [49], we use the series-parallel identification model in which the output of the nonlinear plant (rather than the identification model) is fed back into the identification model, as shown in Fig. 3.2 for example. We can directly use the back-propagation algorithm to train the BP FS in Fig. 3.2, because the error term between the outputs of g and f, which is needed in the BP training algorithm, equals the system output error $e(k+1)$, which is available. Because we only use the series-parallel identification model, the static BP algorithm developed in the last section is sufficient to train the identifiers. In fact, using ideas in [49], we can develop a dynamic back-propagation algorithm for fuzzy systems; we leave the development and applications of this to the future work.

We now propose the method for on-line initial parameter choosing, and provide a theoretical justification (Theorem 3.1) for why this is a good method.

An On-Line Initial Parameter Choosing Method for BP FS: Suppose the nonlinear plant to be identified starts operation from $k = 0$. Do not start the BP training algorithm (3.5), (3.8) and (3.9) for the first M time points. Set the parameters $\bar{x}_i^j(M) = u_i(j)$ and $\bar{z}^j(M) = g(\underline{u}(j))$, where $\underline{u}(j) = (u_1(j), ..., u_n(j))$ is the input to both the plant and the identification model, and $g(\underline{u}(j))$ is the desired output of the BP FS for input $\underline{u}(j)$; and, set $\sigma_i^j(M)$ equal to some small numbers (see Theorem 3.1), or set $\sigma_i^j(M) = [max(u_i(j) : j = 1, 2, ..., M) - min(u_i(j) : j = 1, 2, ..., M)]/2M$ (this choice makes the input membership functions "uniformly" cover the range of $u_i(j)$ from $j = 1$ to $j = M$; in all the simulations in Section 3.4, we use this choice), where $j = 1, 2, ..., M$, $i = 1, 2, ..., n$. Start the on-line training for the BP FS identifier from time point $M + 1$.

We now show that by choosing the σ_i^j sufficiently small, the fuzzy system with

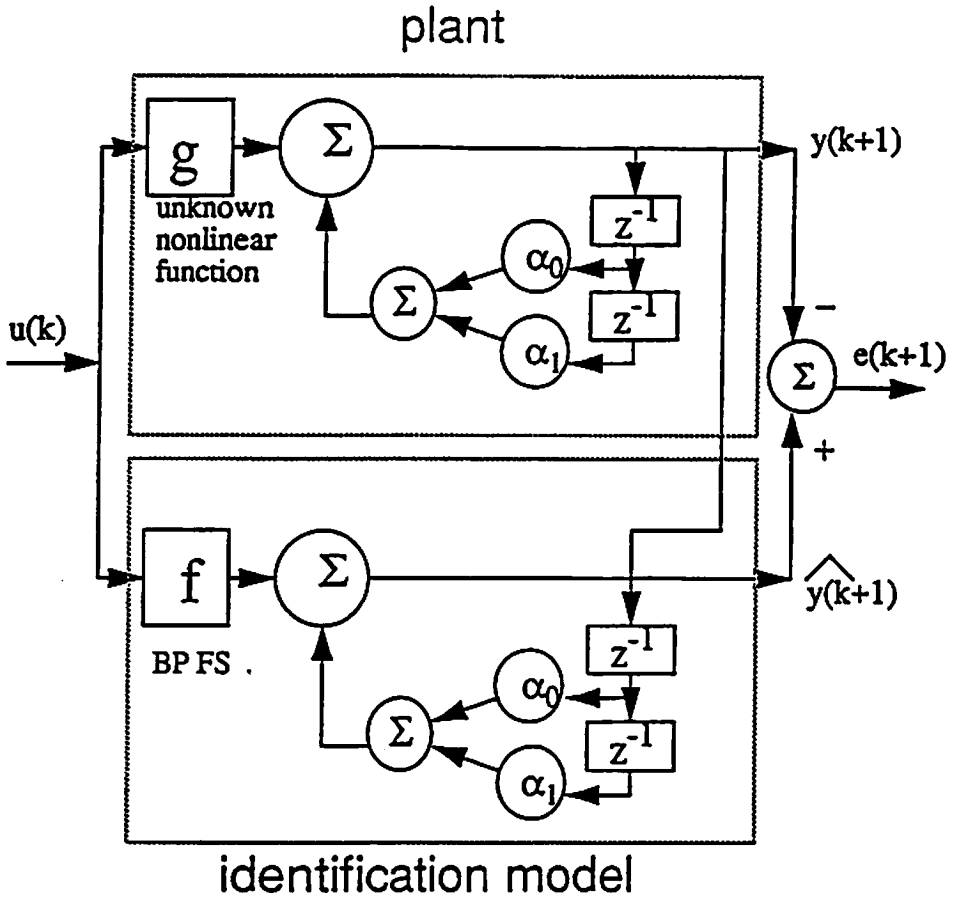


Figure 3.2: An example of series-parallel identification model, where g is an unknown nonlinear function and f is a BP FS.

the above initial parameters can match all the M input-output pairs $(\underline{u}(j), g(\underline{u}(j)))$, $j = 1, 2, ..., M$, to arbitrary accuracy.

THEOREM 3.1: For arbitrary $\epsilon > 0$, there exists $\sigma^* > 0$ such that the fuzzy system f of (3.1) with the above initial $\bar{x}_i^j$ and $\bar{z}^j$ and $\sigma_i^j = \sigma^*$ has the property that

$$|f(\underline{u}(j)) - g(\underline{u}(j))| < \epsilon, \tag{3.10}$$

for all $j = 1, 2, ..., M$.

Proof: From the initial parameter choosing procedure and (3.1) (with $a_i^l = 1$), we have that the fuzzy system with the initial parameters $\bar{x}_i^j(M)$ and $\bar{z}^j(M)$ and $\sigma_i^j = \sigma^*$ is

$$\begin{aligned} f^M(\underline{u}(j)) &= \frac{\sum_{k=1}^M g(\underline{u}(k))(\Pi_{i=1}^n exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}])}{\sum_{k=1}^M (\Pi_{i=1}^n exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}])} \\ &= \frac{g(\underline{u}(j)) + \sum_{j \neq k=1}^M g(\underline{u}(k))(\Pi_{i=1}^n exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}])}{1 + \sum_{j \neq k=1}^M (\Pi_{i=1}^n exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}])} \end{aligned} \tag{3.11}$$

where $j = 1, 2, ..., M$. First, assume that $\underline{u}(j) \neq \underline{u}(k)$ for $j \neq k$; thus, there exist some i such that $u_i(j) \neq u_i(k)$; hence, for arbitrary $\epsilon_1 > 0$ and any $j, k = 1, 2, ..., M$, $j \neq k$, we can make $\Pi_{i=1}^n exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}] < \epsilon_1$ by properly choosing σ^*, because $exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}] \to 0$ as $\sigma^* \to 0$ if $u_i(j) \neq u_i(k)$. From this result and (3.11) we conclude that there exists $\sigma^* > 0$ such that $|f^M(\underline{u}(j)) - g(\underline{u}(j))| < \epsilon$ for all $j = 1, 2, ..., M$..

If $\underline{u}(j) = \underline{u}(k_0)$ for some $k_0 \neq j$, and there are $r - 1$ such k_0; then (3.11) can be written as

$$f^M(\underline{u}(j)) = \frac{r g(\underline{u}(j)) + \sum_k g(\underline{u}(k))(\Pi_{i=1}^n exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}])}{r + \sum_k (\Pi_{i=1}^n exp[-(u_i(j) - u_i(k))^2/2\sigma^{*2}])}, \tag{3.12}$$

where the $\sum_k$ is over all k's in $[1, 2, ..., M]$ except j and the k_0's. Using the same arguments as above, we can prove the truth of (3.10). Q.E.D..

Based on Theorem 3.1 we can say that the initial parameter choosing method is a good one because the fuzzy system with these initial parameters can at least match the first M input-output pairs arbitrarily well. If these first M input-output pairs contain some important features of the unknown nonlinear mapping, we may

hope that after the back-propagation training starts from time point $M+1$, the BP FS identifier will converge to the unknown nonlinear mapping very quickly. In fact, based on our simulation results in Section 3.4, this is indeed true. However, we cannot choose σ_i^j to be too small, because although a fuzzy system with small σ_i^j matches the first M pairs quite well, it will have large approximation errors for other input-output pairs. Therefore, in our simulations, we use the second choice of σ_i^j described in the on-line initial parameter choosing method.

3.4 Simulations

We used the same examples as in [49] to simulate our BP FS identifiers, because we want to compare the BP FS identifiers with neural network identifiers. We simulated the BP FS identifier for four examples, with each example emphasizing a specific point. Example 3.1 emphasizes the detailed procedure of how the BP FS learns to match the unknown nonlinear mapping as training progresses. Example 3.2 shows how performance is improved by incorporating linguistic rules. Example 3.3 shows how the identifier works when only the initial parameters are used. Finally, Example 3.4 shows how the BP FS identifier work for a multi-input-multi-output system. We chose $M = 40$ for all four examples. The BP FS of (3.1) with $M = 40$ has $40 \times 3 = 120$ free parameters (corresponding to each rule there are three free parameters: $\bar{z}^j, \bar{x}_i^j$ and σ_i^j). In [49] the neural network identifiers had two hidden-layers with 10 and 20 neurons in each layer respectively; hence, these neural identifiers had $10 \times 20 = 200$ free parameters. Consequently, from a system complexity point of view (in the sense of number of free parameters), the BP FS identifiers used for the next four examples are simpler than the neural network identifiers used in [49] for the same examples.

Example 3.1: The plant to be identified is governed by the difference equation

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + g[u(k)], \tag{3.13}$$

where the unknown function has the form $g(u) = 0.6sin(\pi u) + 0.3sin(3\pi u) + 0.1sin(5\pi u)$. In order to identify the plant, a series-parallel model governed by

the difference equation

$$\hat{y}(k+1) = 0.3y(k) + 0.6y(k-1) + f[u(k)] \tag{3.14}$$

was used, where $f(*)$ is of the form (3.1) with $M = 40$ and $a_i^l = 1$. We chose $\alpha = 0.5$ in the BP training algorithm (3.5), (3.8) and (3.9), and, we used the on-line initial parameter choosing method in Section 3.3. We started the training from time point $k = 40$, and trained the parameters $\bar{z}^l, \bar{x}_i^j$ and σ_i^j for one cycle at each time point, i.e., we used (3.5), (3.8) and (3.9) once at each time point (in this case the "k" in (3.5), (3.8) and (3.9) agrees with the "k" in (3.13) and (3.14)). Figures 3.3-3.5 show the outputs of the plant (solid-line) and the identification model (dashed-line) when the training was stopped at $k = 100, 200$ and 400, respectively, where the input $u(k) = sin(2\pi k/250)$. We see from Figs. 3.3-3.5 that: 1) the output of the identification model follows the output of the plant almost immediately, and still does so when the training was stopped at $k = 100, 200, 300$ and 400; and, 2) the identification model approximates the plant more and more accurately as more and more training is performed. In [49] the same plant was identified using a neural network identifier which failed to follow the plant when the training was stopped at $k = 500$; Fig. 3.6 shows this result. We see from Fig. 3.3 that our BP FS identifier follows the plant without large errors even when the training was stopped as early as $k = 100$. We think that the main reason for the superior performance of the BP FS identifier is that we have a very good initial parameter choosing method. We further test the initial parameter choosing method in Example 3.3.

If we accept the initial parameter choosing method as a good one, how about the BP training algorithm itself ? Can the latter make the identification model converge to the plant when the initial parameters are chosen randomly ? Figure 3.7 shows the outputs of the identification model and the plant for the input $u(k) = sin(2\pi k/250)$ for $1 \leq k \leq 250$ and $501 \leq k \leq 700$ and $u(k) = 0.5sin(2\pi k/250) + 0.5sin(2\pi k/25)$ for $251 \leq k \leq 500$ after the identification model was trained for 5000 time steps using a random input whose amplitude was uniformly distributed over the interval [-1, 1], where the initial $\bar{z}^j(0), \bar{x}_i^j(0)$ and $\sigma_i^j(0)$ for the training phase were random and uniformly distributed over $[-5, 5]$, $[-1, 1]$ and $[0, 0.3]$,

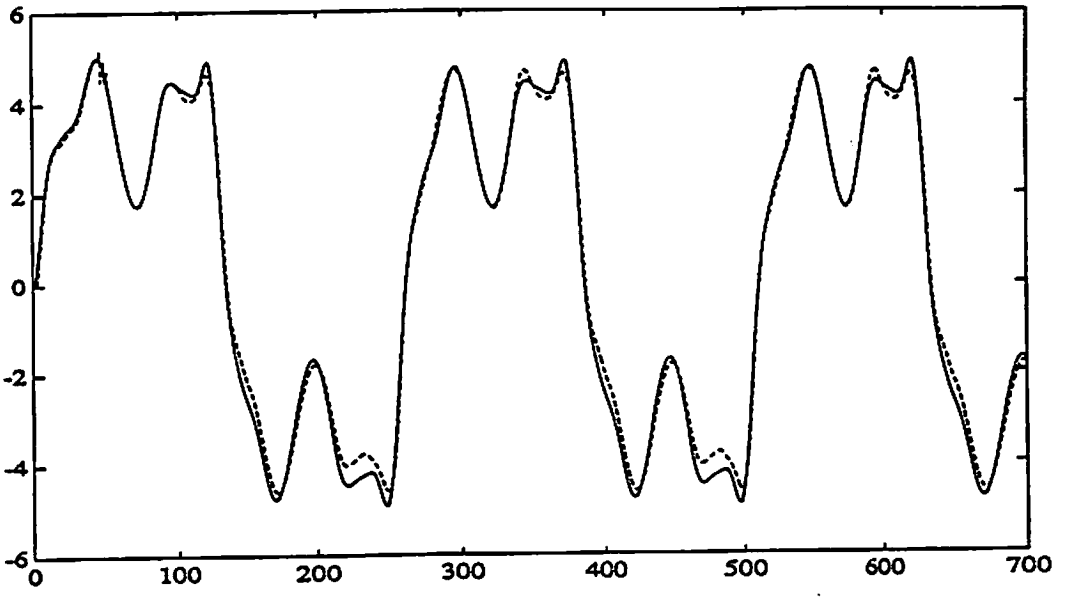


Figure 3.3: Outputs of the plant (solid-line) and the identification model (dashed-line) for Example 3.1 when the training stops at $k = 100$.

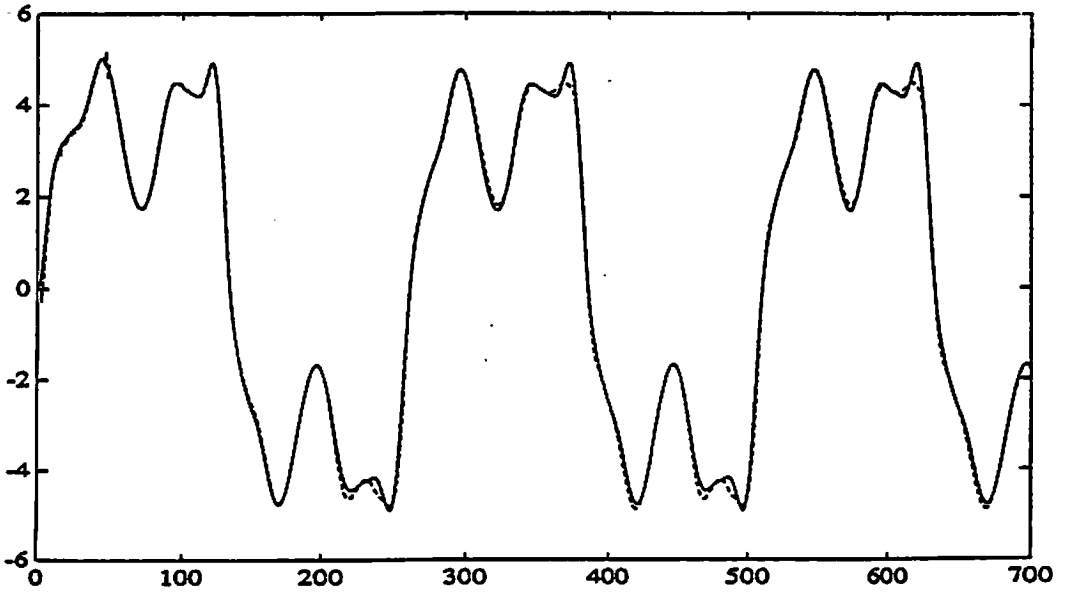


Figure 3.4: Outputs of the plant (solid-line) and the identification model (dashed-line) for Example 3.1 when the training stops at $k = 200$.

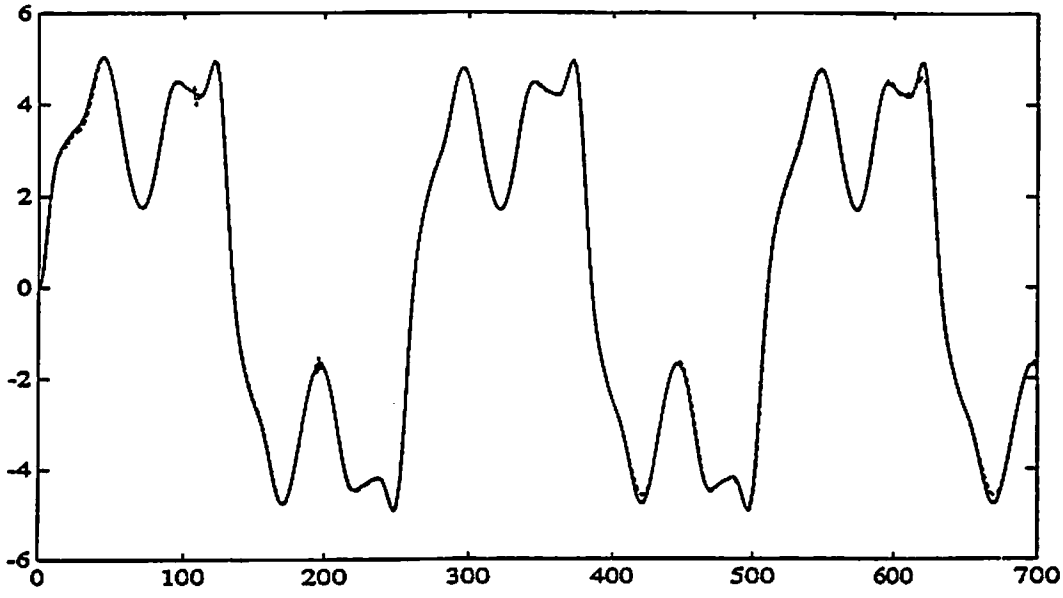


Figure 3.5: Outputs of the plant (solid-line) and the identification model (dashed-line) for Example 3.1 when the training stops at $k = 400$.

respectively. We performed 50 Monte Carlo simulations for the training phase, and all the trained fuzzy identifiers show indistinguishable responses from those shown in Fig. 3.7. We see from Fig. 3.7 that the trained identification model approximates the plant quite well.

Example 3.2: The plant to be identified is described by the second-order difference equation

$$y(k+1) = g[y(k), y(k-1)] + u(k), \tag{3.15}$$

where

$$g[y(k), y(k-1)] = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)}, \tag{3.16}$$

and $u(k) = sin(2\pi k/25)$. A series-parallel identifier described by the equation

$$\hat{y}(k+1) = f[y(k), y(k-1)] + u(k) \tag{3.17}$$

was used, where $f[y(k), y(k-1)]$ is in the form of (3.1) with $M = 40$ and $a_i^l = 1$. We chose $\alpha = 0.5$ in the BP training algorithm. Figure 3.8 shows the outputs of

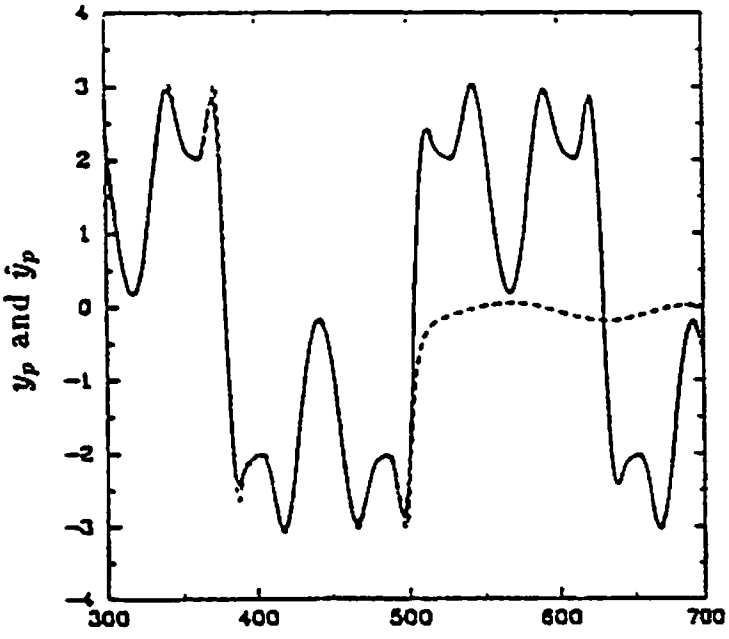


Figure 3.6: Outputs of the plant (solid-line) and the neural network identification model of [49] (dashed-line) for Example 3.1 when the training stops at $k = 500$. (Narendra et al., 1990. © 1990 IEEE.)

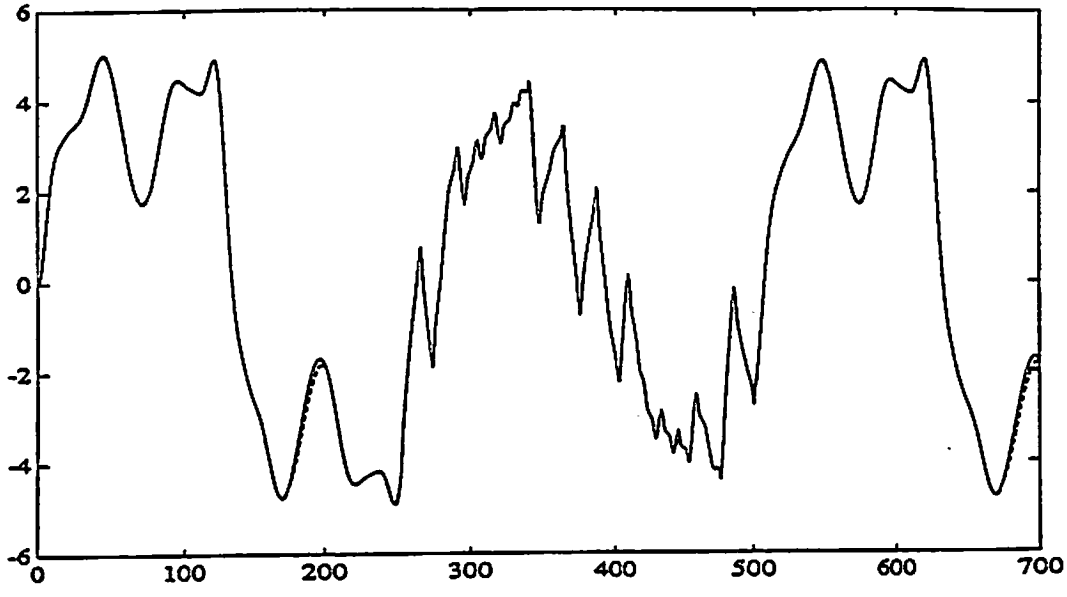


Figure 3.7: Outputs of the identification model (dashed-line) and the plant (solid-line) for Example 3.1 for the input $u(k) = sin(2\pi k/250)$ for $1 \leq k \leq 250$ and $501 \leq k \leq 700$ and $u(k) = 0.5sin(2\pi k/250) + 0.5sin(2\pi k/25)$ for $251 \leq k \leq 500$ after the identification model was trained for 5000 time steps.

the plant and the identification model for arbitrary $\bar{z}^j(0), \bar{x}_i^j(0)$ and $\sigma_i^j(0)$ which were chosen from the intervals [-2,2], [-1,1] and [0, 0.3], respectively, where the BP FS identifier was trained for one cycle at each time point starting from $k = 0$. Now suppose that we have the following linguistic rule describing the unknown nonlinear function $g[y(k), y(k-1)]$ of (3.16):

$$R: \ IF \quad y(k) \ is \ near \ zero \ or \ y(k-1) \ is \ near \ zero \ or \ y(k) \ is \ near \ -2.5, \ THEN \ g[y(k), y(k-1)] \ is \ near \ zero, \qquad (3.18)$$

where "y is near zero" (y can be y(k), y(k-1) or g[y(k), y(k-1)]) is characterized by the Gaussian membership function $exp[-\frac{1}{2}(\frac{y}{0.3})^2]$, and "y is near 0.5" is characterized by the Gaussian membership function $exp[-\frac{1}{2}(\frac{y-0.5}{0.3})^2]$. Figure 3.9 shows the outputs of the plant and the identification model after the linguistic rule (3.18) is incorporated, where the initial $\bar{z}^j(0), \bar{x}_i^j(0)$ and $\sigma_i^j(0)$ were the same as those used in the simulation of Fig. 3.8, and the identifier was trained for one cycle at each time point starting from $k = 0$. Comparing Figs. 3.9 and 3.8 we see an improvement in adaptation speed after the linguistic rule was incorporated.

Example 3.3: The plant to be identified is of the form

$$y(k+1) = g[y(k), y(k-1), y(k-2), u(k), u(k-1)], \qquad (3.19)$$

where the unknown function g has the form

$$g(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2}, \qquad (3.20)$$

and $u(k) = sin(2\pi k/250)$ for $k \leq 500$ and $u(k) = 0.8sin(2\pi k/250) + 0.2sin(2\pi k/25)$ for $k > 500$. The identification model is

$$\hat{y}(k+1) = f[y(k), y(k-1), y(k-2), u(k), u(k-1)], \qquad (3.21)$$

where f is of the form (3.1) with $M = 40$ and $a_i^l = 1$. One purpose for this example is to test the initial parameter choosing method described in Section 3.3. For this purpose, we used the on-line initial parameter choosing method, but

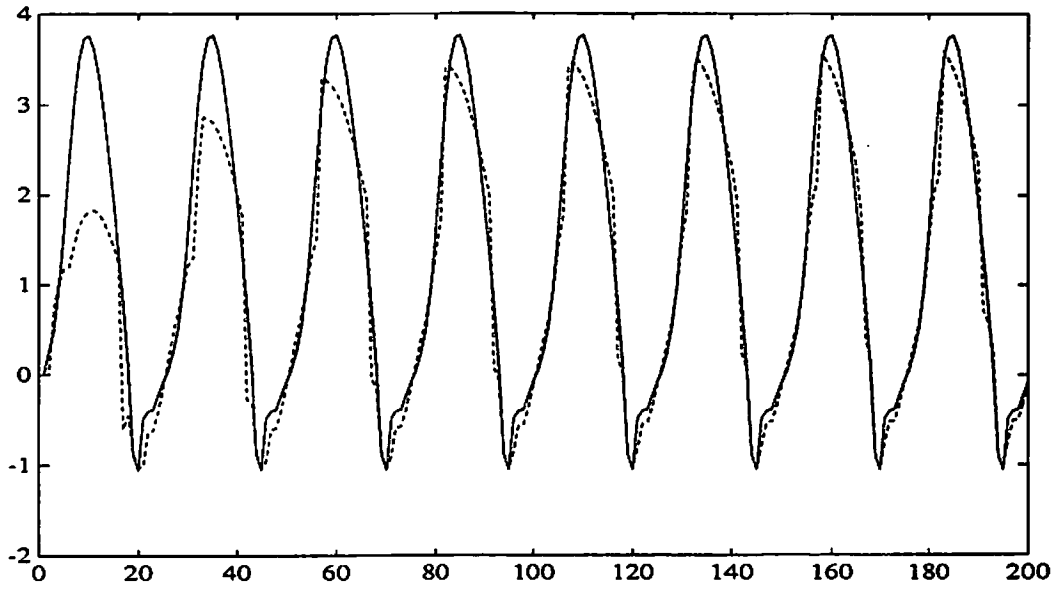


Figure 3.8: Outputs of the plant (solid-line) and the identification model (dashed-line) for Example 3.2 without using the linguistic rule.

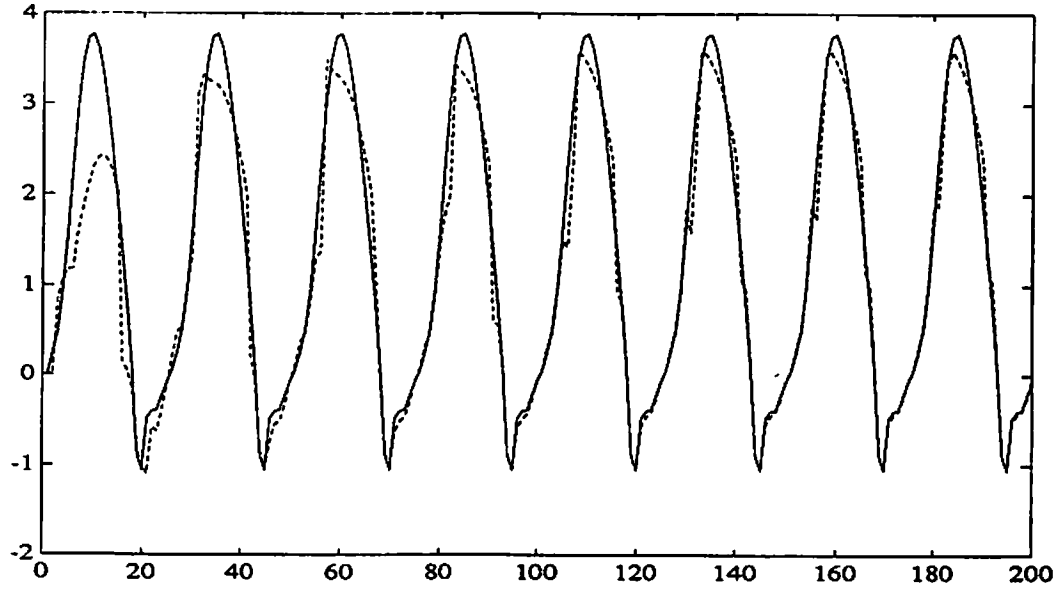


Figure 3.9: Outputs of the plant (solid-line) and the identification model (dashed-line) for Example 3.2 after the linguistic rule was incorporated.

after $k = M$ no back-propagation was performed. Figure 3.10 shows the outputs of the plant and the identification model whose parameters were determined only based on the on-line initial parameter choosing method. We see from Fig. 3.10 that the identification model could track the plant but with large error. Next, we trained the BP FS identifier for 5000 time steps using a random input $u(k)$ whose magnitude was uniformly distributed over [-1,1], where the parameters were trained for one cycle at each time point, and we used the on-line initial parameter choosing method. We chose $\alpha = 0.5$ in the training phase. The outputs of the plant and the trained BP FS identifier are shown in Fig. 3.11. In [49], a neural network identifier was used for this plant; Fig. 3.12 shows the outputs of the plant and the neural identifier after the neural identifier was trained for 10^5 steps using a random input uniformly distributed in the interval $[-1, 1]$. Comparing Figs. 3.11 and 3.12, we see that although the neural identifier was trained for 100,000 steps, its performance is worse than that of our BP FS identifier, which was trained for only 5000 steps. This suggests that, even without using the on-line initial parameter choosing method proposed in Section 3.3, the BP FS identifier still shows superior performance over the BP neural network identifier.

Example 3.4: In this example, we show how the BP FS identifier works for a multi-input-multi-output plant which is described by the equations

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} \frac{y_1(k)}{1+y_2^2(k)} \\ \frac{y_1(k)y_2(k)}{1+y_2^2(k)} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \tag{3.22}$$

The series-parallel identification model consists of two BP FS's f^1 and f^2 and is described by the equations

$$\begin{bmatrix} \hat{y}_1(k+1) \\ \hat{y}_2(k+1) \end{bmatrix} = \begin{bmatrix} f^1(y_1(k), y_2(k)) \\ f^2(y_1(k), y_2(k)) \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \tag{3.23}$$

Both f^1 and f^2 are in the form of (3.1) with $M = 40$ and $a_i^l = 1$. The identification procedure was carried out for 5000 time steps using random inputs $u_1(k)$ and $u_2(k)$ whose magnitudes were uniformly distributed over [-1,1], where we chose $\alpha = 0.5$, used the on-line initial parameter choosing method proposed in Section

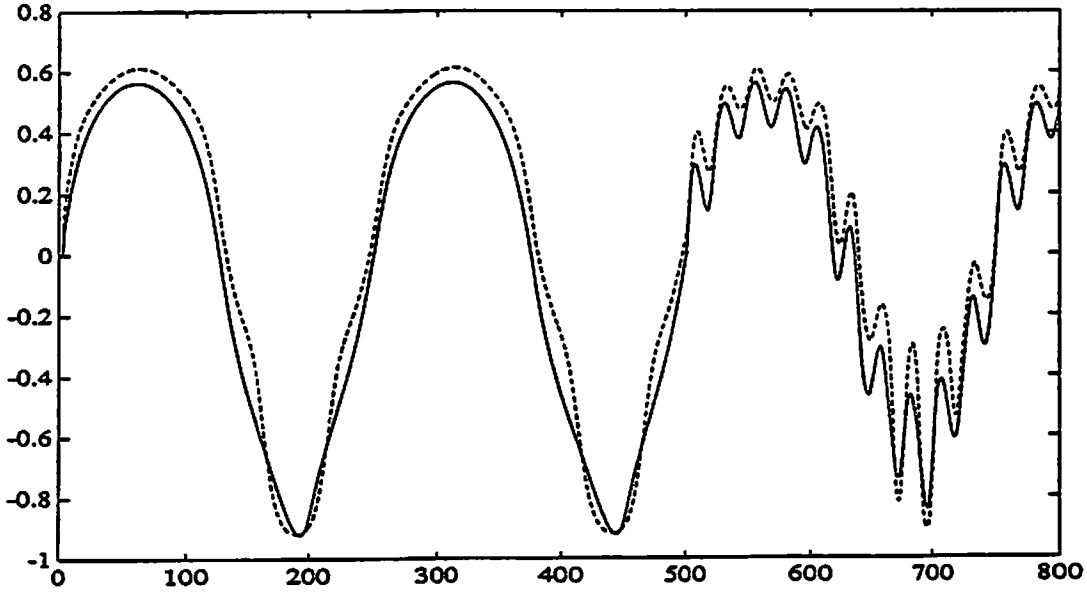


Figure 3.10: Outputs of the plant (solid-line) and the identification model (dashed-line) for Example 3.3 when the parameters of the identifier were determined based only on the initial parameter choosing method and there was no BP training.

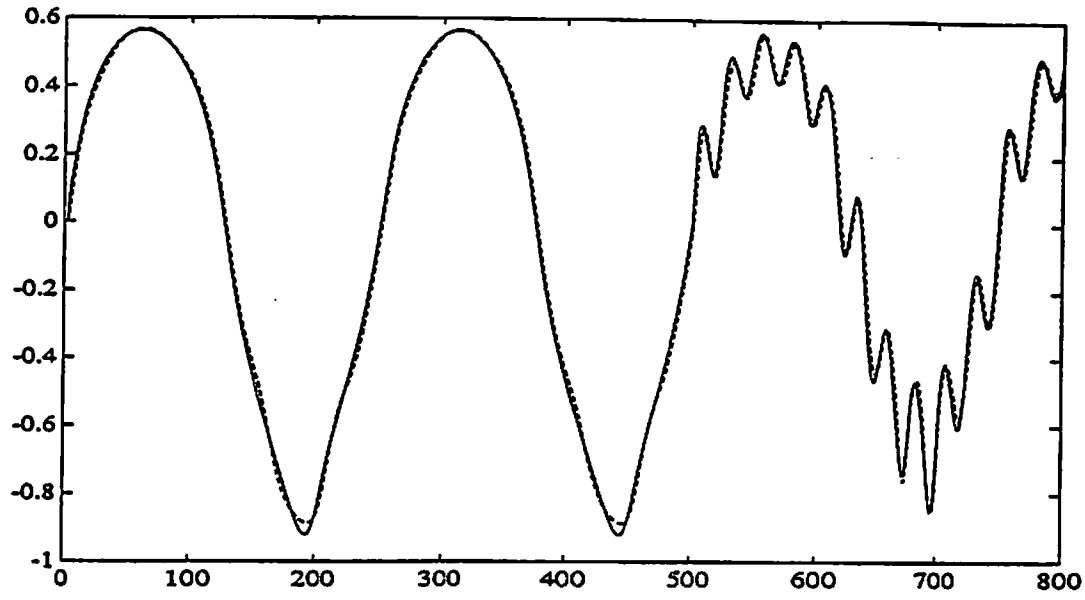


Figure 3.11: Outputs of the plant (solid-line) and the identification model (dashed-line) for Example 3.3 after 5000 step training.

3.3, and trained the parameters for one cycle at each time point. The responses of the plant and the trained identification model for a vector input $[u_1(k), u_2(k)] = [sin(2\pi k/25), cos(2\pi k/25)]$ are shown in Figs. 3.12 and 3.13 for $y_1(k)$ and $\hat{y}_1(k)$ and $y_2(k)$ and $\hat{y}_2(k)$, respectively. In [49], a neural network identifier was used for this plant; Figs. 3.15 and 3.16 show the outputs of the plant and the neural identifier after the neural identifier was trained for 10^5 steps with inputs u_1 and u_2 uniformly distributed in $[-1, 1]$. Comparing Figs. 3.13 and 3.14 with Figs. 3.15 and 3.16, we see that the performance of the fuzzy and neural identifiers is similar, although the BP FS identifier was trained for only 5000 steps, whereas the neural identifier was trained for 10^5 steps.

3.5 Conclusions

In this chapter, a trainable fuzzy system which we call a "Back-Propagation Fuzzy System (BP FS)," was used as an identifier for nonlinear dynamic systems. The BP FS was proven to be capable of approximating any nonlinear real continuous

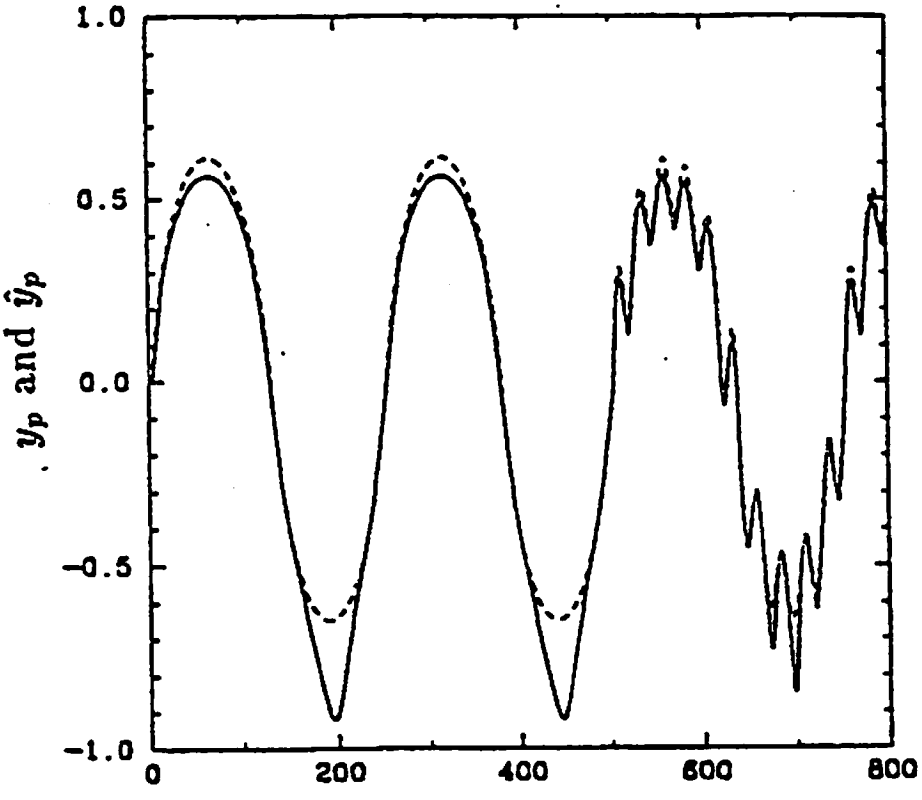


Figure 3.12: Outputs of the plant (solid-line) and the neural network identification model of [49] (dashed-line) for Example 3.3 after 10^5 step training. (Narendra et al., 1990. © 1990 IEEE.)

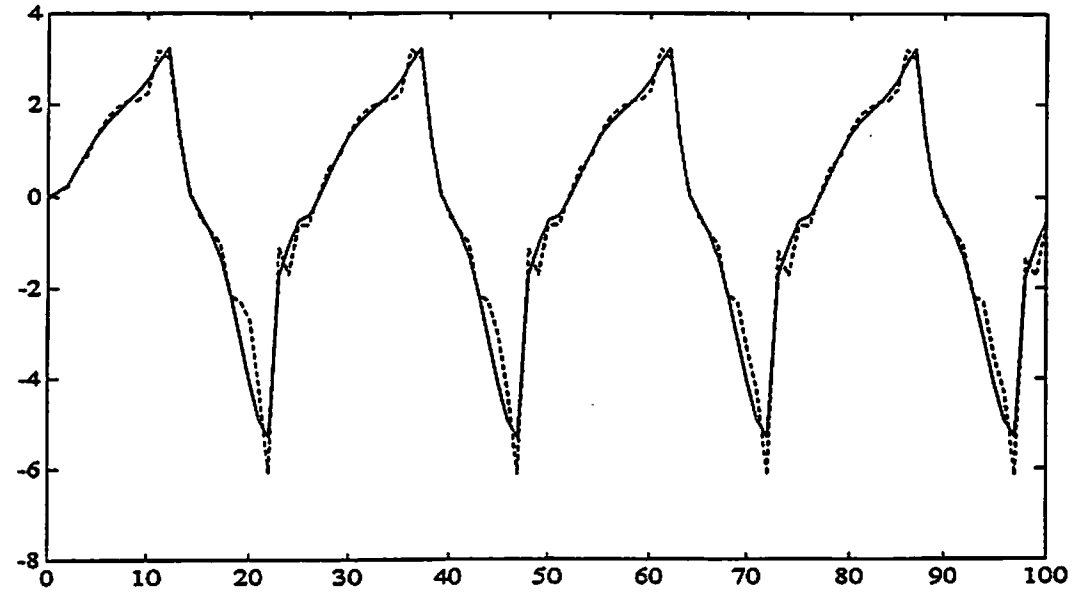


Figure 3.13: Outputs of the plant ($y_1(k)$, solid-line) and the identification model ($\hat{y}_1(k)$, dashed-line) for Example 3.4 after 5000 steps of training.

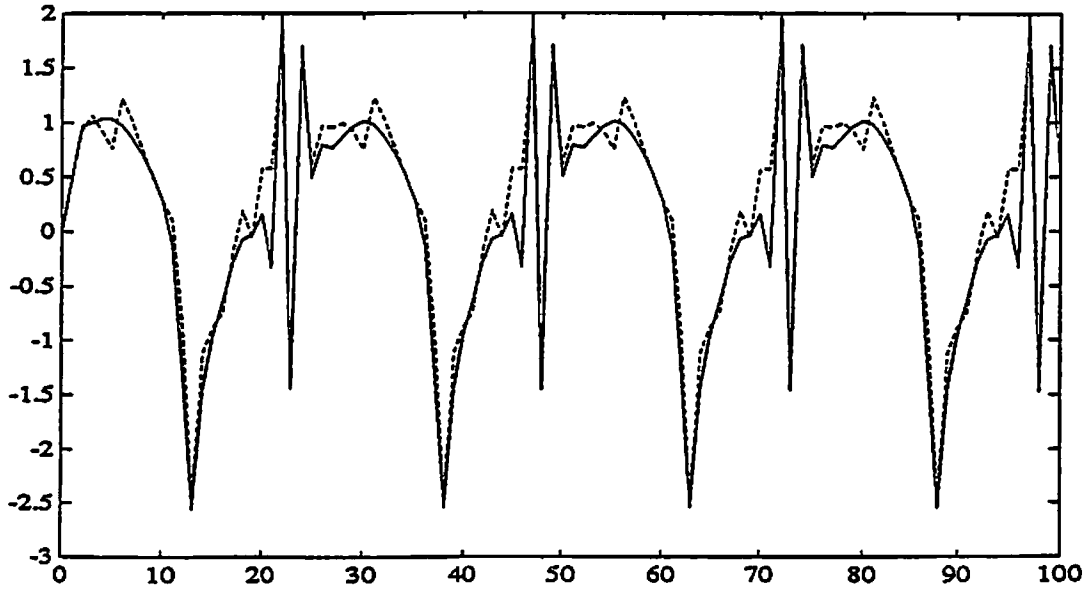


Figure 3.14: Outputs of the plant ($y_2(k)$, solid-line) and the identification model ($\hat{y}_2(k)$, dashed-line) for Example 3.4 after 5000 steps of training.

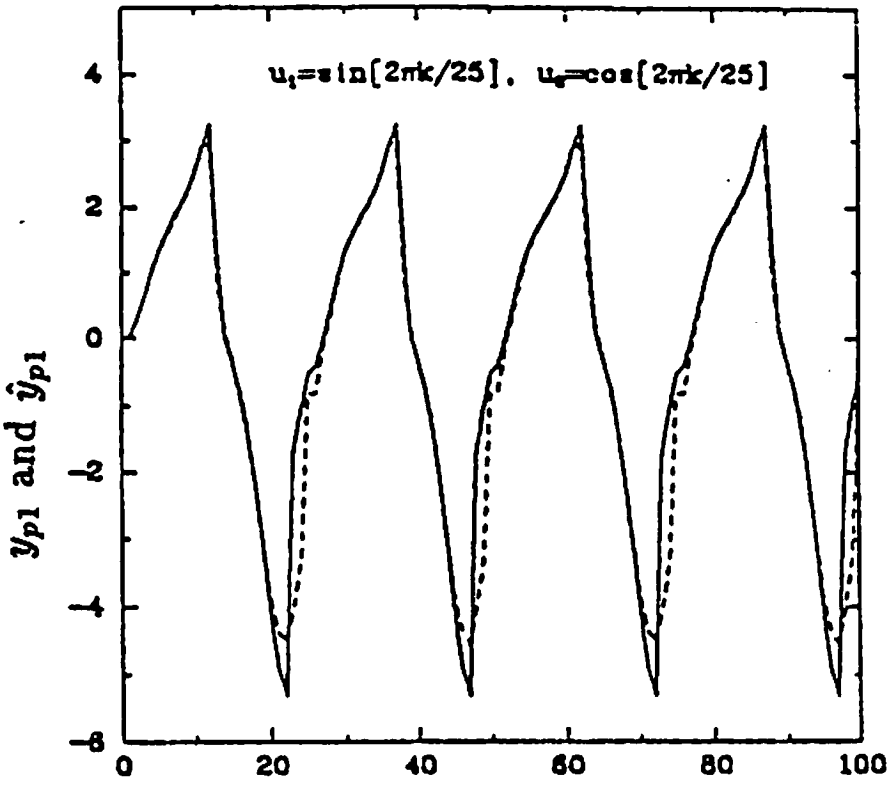


Figure 3.15: Outputs of the plant ($y_1(k)$, solid-line) and the neural network identification model of [49] ($\hat{y}_1(k)$, dashed-line) for Example 3.4 after 10^5 steps of training. (Narendra et al., 1990. © 1990 IEEE.)

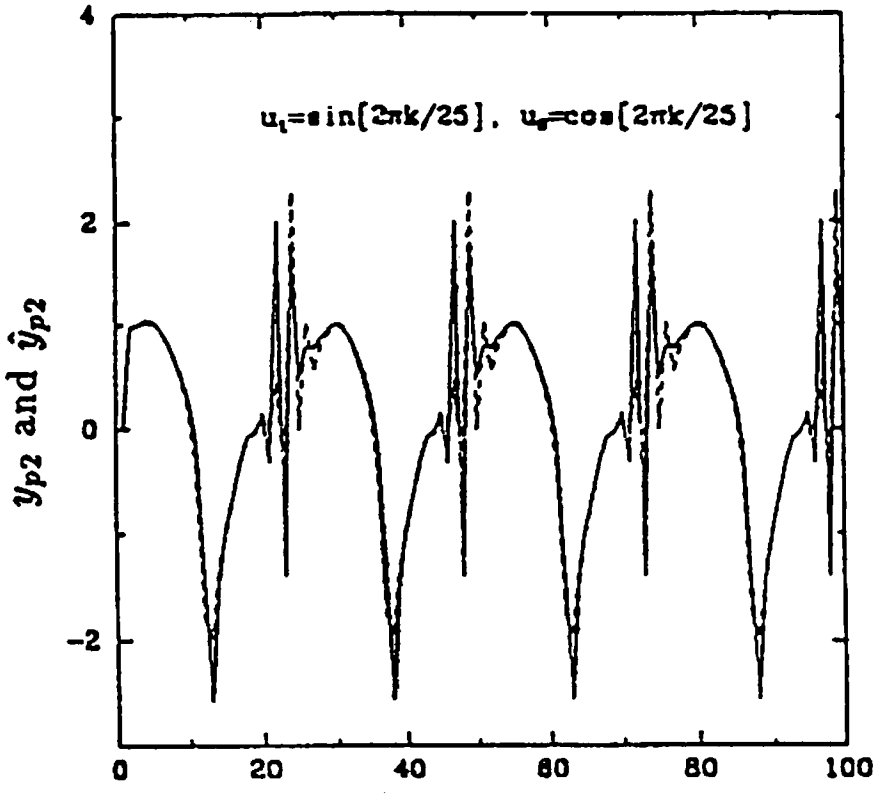


Figure 3.16: Outputs of the plant ($y_2(k)$, solid-line) and the neural network identification model of [49] ($\hat{y}_2(k)$, dashed-line) for Example 3.4 after 10^5 steps of training. (Narendra et al., 1990. © 1990 IEEE.)

function on a compact set to arbitrary accuracy. By using the fact that fuzzy systems can be represented as a three-layer feedforward network, a back-propagation algorithm was developed to train the fuzzy system to match desired input-output pairs. We proposed an on-line initial parameter choosing method for the BP FS, and, we showed that it is straightforward to incorporate linguistic IF-THEN rules into the BP FS.

We simulated the BP FS identifier for the same examples as in [49] (for testing neural network identifiers) and observed that: (1) convergence of the BP training algorithm for the fuzzy system was much faster than that of the BP training algorithm for the neural network; (2) the BP FS identifiers achieved similar or better performance than the neural network identifiers, using simpler systems (in the sense of fewer free parameters which need to be trained by the BP algorithms); and, (3) the BP FS identifier can effectively incorporate linguistic IF-THEN rules into the identifers, whereas the neural network identifiers cannot.

Chapter 4

DESIGN OF FUZZY SYSTEMS USING ORTHOGONAL LEAST SQUARES LEARNING

4.1 Introduction

In Chapter 3 we represented fuzzy systems as three-layer feedforward networks, and based on this representation, the back-propagation algorithm was developed 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, and can be trapped at a local minimum and converges slowly [although much faster than a comparable back-propagation neural network (see Chapter 3)]. In this chapter, we fix some parameters of the fuzzy system such that the resulting fuzzy system is equivalent to a series expansion of some basis functions which 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. Also, the OLS algorithm generates a robust fuzzy system which is not sensitive to noise in its inputs.

The most important advantage of using fuzzy basis functions, rather than polynomials [6,57], radial basis functions [5,58], neural networks [60], 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 [9,31], and, experienced operators of power plants can provide operational instructions in the form of linguistic fuzzy IF-THEN rules [25], etc.. 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 some 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.

4.2 Fuzzy Systems as Fuzzy Basis Function Expansions

Definition 4.1: Consider the fuzzy systems in Definition 1.4. Define *fuzzy basis functions* (FBF) as

$$p_j(\underline{x}) = \frac{\prod_{i=1}^{n} \mu_{F_i^j}(x_i)}{\sum_{j=1}^{M} \prod_{i=1}^{n} \mu_{F_i^j}(x_i)}, \quad j = 1, 2, ..., M, \tag{4.1}$$

where $\mu_{F_i^j}(x_i) = a_i^j exp(-\frac{1}{2}(\frac{x_i - \bar{x}_i^j}{\sigma_i^j})^2$ are Gaussian membership functions. Then, the fuzzy system (1.19) is equivalent to an *FBF expansion*

$$f(\underline{x}) = \sum_{j=1}^{M} p_j(\underline{x}) \theta_j, \tag{4.2}$$

where $\theta_j = \bar{z}^j \in R$ are constants.

From (4.1) and (1.9) we see that an FBF corresponds to a fuzzy IF-THEN rule. Specifically, an FBF for a rule can be determined as follows: first, calculate the product of all membership functions for the linguistic terms in the IF part of the rule, and call it a *pseudo-FBF* for the rule; then, after calculating the pseudo-FBF's for all the M rules, the FBF for the j'th rule is determined by dividing the pseudo-FBF for the j'th rule 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 method described in Section 4.3).

What does the FBF of (4.1) look like when plotted as a function of $\underline{x}$? We now consider a simple one-dimensional example (i.e., $n = 1$). Suppose that we have four fuzzy rules in the form of (1.9) with $\mu_{Fj}(x) = exp[-\frac{1}{2}(x - \bar{x}^j)^2]$, where $\bar{x}^j = -3, 1, 1, 3$ for $j = 1, 2, 3, 4$, respectively (note that the FBF's are determined only based on the IF parts of the rules, so we do not need the $\mu_{Gj}(z)$). Therefore, $p_j(x) = exp[-\frac{1}{2}(x-\bar{x}^j)^2] / \sum_{i=1}^{4} exp[-\frac{1}{2}(x-\bar{x}^i)^2]$, which are plotted in Fig. 4.1 from left to right for $j = 1, 2, 3, 4$, respectively. From Fig. 4.1 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 [12]. 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 [37]. 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 3), 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 to have good global properties [12,37]. 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.

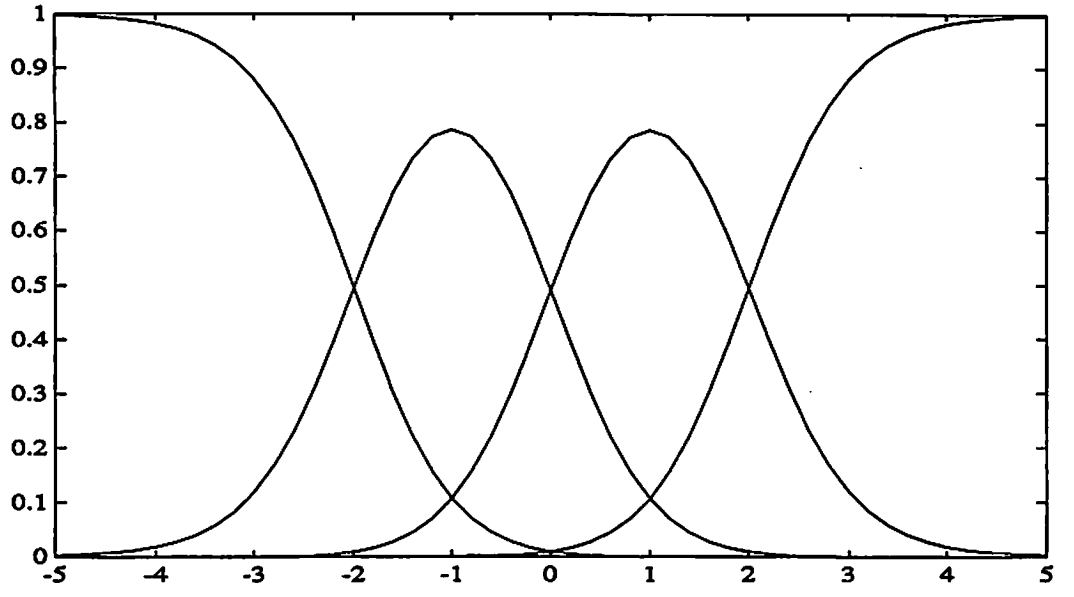


Figure 4.1: An example of the fuzzy basis functions.

Equation (4.1) 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 (3.1) 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 some functions which are defined as FBF's. Different FBF's have different properties.

We can analyse (4.2) from two points of view. First, if we view all the parameters $a_i^j, \bar{x}_i^j$ and σ_i^j in $p_j(\underline{x})$ as free design parameters, then the FBF expansion (4.2) 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 Chapter 3. On the other hand, we can fix all the parameters in $p_j(\underline{x})$ at the very begining of the FBF expansion design procedure, so that the only free design parameters are θ_j; in this case, $f(\underline{x})$ of (4.2) is linear in the parameters. We adopt this second point of view in this chapter. 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 [5,6], to design the FBF expansions.

4.3 Orthogonal Least Squares Learning

In order to describe how the orthogonal least squares (OLS) learning algorithm works, it is essential to view the fuzzy basis function expansion (4.2) as a special case of the linear regression model

$$d(t) = \sum_{j=1}^{M} p_j(t)\theta_j + e(t), \tag{4.3}$$

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

$$p_j(t) = p_j(\underline{x}(t)), \tag{4.4}$$

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: $(\underline{x}^0(t), d^0(t)), t = 1, 2, ..., N$. Our task is to design an FBF expansion $f(\underline{x})$ such that some error function between $f(\underline{x}^0(t))$ and $d^0(t)$ is minimized.

In order to present the OLS algorithm, we arrange (4.3) from $t = 1$ to N in the following matrix form:

$$\underline{d} = P\underline{\theta} + \underline{e}, \tag{4.5}$$

where $\underline{d} = [d(1), ..., d(N)]^T$, $P = [\underline{p}_1, ..., \underline{p}_M]$ with $\underline{p}_i = [p_i(1), ..., p_i(N)]^T$, $\underline{\theta} = [\theta_1, ..., \theta_M]^T$, and $\underline{e} = [e(1), ..., e(N)]^T$. The OLS algorithm transforms the set of $\underline{p}_i$ 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 $p_j(\underline{x})$ based on the input-output pairs. We propose the following scheme:

Initial FBF Determination: Choose N initial $p_j(\underline{x})$'s in the form of (4.1) (for this case, M in (4.1) equals N), with the parameters determined as follows:

$a_i^j = 1, \bar{x}_i^j = x_i^0(j)$, and $\sigma_i^j = [max(x_i^0(j), j = 1,2,...,N) - min(x_i^0(j), j = 1,2,...,N)]/M_s$, where $i = 1,2,...,n$, $j = 1,2,...,N$, 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, $M_s << N$.

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 [5,6] (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 $1 \leq i \leq N$, compute

$$\underline{w}_1^{(i)} = \underline{p}_i, \qquad g_1^{(i)} = (\underline{w}_1^{(i)})^T \underline{d}^0 / ((\underline{w}_1^{(i)})^T \underline{w}_1^{(i)}), \tag{4.6}$$

$$[err]_1^{(i)} = (g_1^{(i)})^2 (\underline{w}_1^{(i)})^T \underline{w}_1^{(i)} / (\underline{d}^{0T} \underline{d}^0), \tag{4.7}$$

where $\underline{p}_i = [p_i(\underline{x}^0(1)), ..., p_i(\underline{x}^0(N))]^T$, and $p_i(\underline{x}^0(t))$ are given by the Initial FBF Determination method. Find

$$[err]_1^{(i_1)} = max([err]_1^{(i)}, 1 \leq i \leq N), \tag{4.8}$$

and select

$$\underline{w}_1 = \underline{w}_1^{(i_1)} = \underline{p}_{i_1}, \quad g_1 = g_1^{(i_1)}. \tag{4.9}$$

□ At the k'th step where $2 \leq k \leq M_s$, for $1 \leq i \leq N$, $i \neq i_1, ..., i \neq i_{k-1}$, compute

$$\alpha_{jk}^{(i)} = \underline{w}_j^T \underline{p}_i / (\underline{w}_j^T \underline{w}_j), \quad 1 \leq j < k \tag{4.10}$$

$$\underline{w}_k^{(i)} = \underline{p}_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} \underline{w}_j, \qquad g_k^{(i)} = (\underline{w}_k^{(i)})^T \underline{d}^0 / ((\underline{w}_k^{(i)})^T \underline{w}_k^{(i)}), \tag{4.11}$$

$$[err]_k^{(i)} = (g_k^{(i)})^2 (\underline{w}_k^{(i)})^T \underline{w}_k^{(i)} / (\underline{d}^{0T} \underline{d}^0). \tag{4.12}$$

Find

$$[err]_k^{(i_k)} = max([err]_k^{(i)}, 1 \leq i \leq N, i \neq i_1, ..., i \neq i_{k-1}), \tag{4.13}$$

and select

$$\underline{w}_k = \underline{w}_k^{(i_k)}, \quad g_k = g_k^{(i_k)}. \tag{4.14}$$

□ Solve the triangular system

$$A^{(M_s)}\underline{\theta}^{(M_s)} = \underline{g}^{(M_s)}, \tag{4.15}$$

where

$$A^{(M_s)} = \begin{bmatrix} 1 & \alpha_{12}^{(i_2)} & \alpha_{13}^{(i_3)} & ... & \alpha_{1M_s}^{(i_{M_s})} \\ 0 & 1 & \alpha_{23}^{(i_3)} & ... & \alpha_{2M_s}^{(i_{M_s})} \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & 1 & \alpha_{M_s-1,M_s}^{(i_{M_s})} \\ 0 & 0 & 0 & ... & 1 \end{bmatrix}, \tag{4.16}$$

$$\underline{g}^{(M_s)} = [g_1, ..., g_{M_s}]^T, \qquad \underline{\theta}^{(M_s)} = [\theta_1^{(M_s)}, ..., \theta_{M_s}^{(M_s)}]^T. \tag{4.17}$$

The final FBF expansion is

$$f(\underline{x}) = \sum_{j=1}^{M_s} p_{i_j}(\underline{x})\theta_j^{(M_s)}, \tag{4.18}$$

where $p_{i_j}(\underline{x})$ are 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 [5,6].

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 (4.5), we have that $\underline{d} = WA\underline{\theta} + \underline{e} = W\underline{g} + \underline{e}$, where $\underline{g} = A\underline{\theta}$ 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 $[err]_k^{(i)} = (g_k^{(i)})^2(\underline{w}_k^{(i)})^T\underline{w}_k^{(i)}/(\underline{d}^{0T}\underline{d}^0)$ represents the error reduction ratio due to $\underline{w}_k^{(i)}$ [5,6]; 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.

4.4 Control of the Nonlinear Ball and Beam System Using FBF Expansions

In this section, we use the OLS algorithm to design an FBF expansion to approximate a controller for a nonlinear ball and beam system [18]. 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 [18] to generate a set of state-control pairs for some randomly sampled points in a certain region of the state space, and then view these state-control pairs as the input-output pairs in Section 4.3 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 [18] to generate a look-up 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 look-up 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. 4.2. 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 $\underline{x} = (r, \dot{r}, \theta, \dot{\theta})^T$ be the state vector of the system, and $y = r$ be the output of the

system. Then, from [18], the system can be represented by the state-space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - Gsinx_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u \tag{4.19}$$

$$y = x_1, \tag{4.20}$$

where the control u is the acceleration of θ, and the parameters B and G are defined in [18]. The purpose of control is to determine $u(\underline{x})$ such that the closed-loop system output y will converge to zero from arbitrary initial conditions in a certain region.

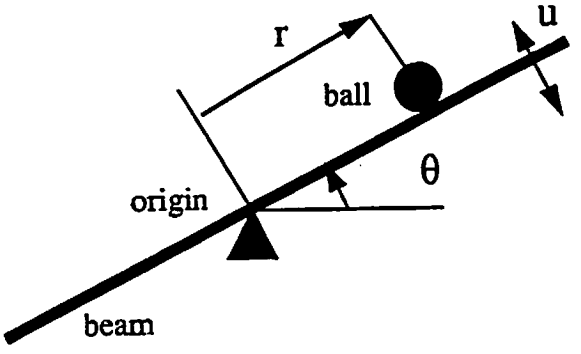


Figure 4.2: The ball and beam system.

The input-output linearization algorithm of [18] determines the control law $u(\underline{x})$ as follows: for state $\underline{x}$, compute $v(\underline{x}) = -\alpha_3\phi_4(\underline{x}) - \alpha_2\phi_3(\underline{x}) - \alpha_1\phi_2(\underline{x}) - \alpha_0\phi_1(\underline{x})$, where $\phi_1(\underline{x}) = x_1$, $\phi_2(\underline{x}) = x_2$, $\phi_3(\underline{x}) = -BGsinx_3$, $\phi_4(\underline{x}) = -BGx_4cosx_3$, 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 $a(\underline{x}) = -BGcosx_3$ and $b(\underline{x}) = BGx_4^2 sinx_3$; then, $u(\underline{x}) =$

$(v(\underline{x}) - b(\underline{x}))/a(\underline{x})$.

In our simulations, we used the $u(\underline{x}) = (v(\underline{x}) - b(\underline{x}))/a(\underline{x})$ to generate N $(\underline{x}, u)$ pairs with $\underline{x}$ 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: Case 1: $N = 200, M_s = 20$, and the final FBF expansion $f(\underline{x})$ of (4.18) was used as the control u in (4.19); Case 2: $N = 40, M_s = 20$, and the final FBF expansion $f(\underline{x})$ of (4.18) was used as the control u in (4.19); and, Case 3: $N = 40, M_s = 20$, and the control

$$u(\underline{x}) = \frac{1}{2}[f(\underline{x}) + f^L(\underline{x})], \tag{4.21}$$

where $f(\underline{x})$ is given by (4.18), and $f^L(\underline{x})$ is a *linguistic controller* which is in the form of (1.19) 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',* (4.22)

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,'* (4.23)

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,'* (4.24)

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,'* (4.25)

where the 'positive' for x_1 is a fuzzy set $P1$ with membership function $\mu_{P1}(x_1) = exp[-\frac{1}{2}(\frac{min(x_1-4,0)}{4})^2]$, the 'negative' for x_1 is a fuzzy set $N1$ with membership function $\mu_{N1}(x_1) = exp[-\frac{1}{2}(\frac{max(x_1+4,0)}{4})^2]$, the 'near zero' for both x_2 and x_4 is a fuzzy set ZO with $\mu_{NO}(x) = exp[-\frac{1}{2}x^2]$, the 'positive' for x_3 is a fuzzy set $P3$ with $\mu_{P3}(x_3) = exp[-\frac{1}{2}(\frac{min(x_3-\pi/4,0)}{\pi/4})^2]$, the 'negative' for x_3 is a fuzzy set $N3$ with $\mu_{N3}(x_3) = exp[-\frac{1}{2}(\frac{max(x_3+\pi/4,0)}{\pi/4})^2]$, the 'positive' for u is a fuzzy set Pu with $\mu_{Pu}(u) = exp[-\frac{1}{2}(u - 0.1)^2]$, the 'negative' for u is a fuzzy set Nu with $\mu_{Nu}(u) = exp[-\frac{1}{2}(u + 0.1)^2]$, the 'positive big' for u is a fuzzy set PBu with $\mu_{PBu}(u) = exp[-\frac{1}{2}(u - 0.4)^2]$, and the 'negative big' for u is a fuzzy set NBu

with $\mu_{NBu}(u) = exp[-\frac{1}{2}(u+0.4)^2]$. 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 parts membership functions were determined by common sense and trial and error. The detailed formula of $f^L(\underline{x})$ can be easily obtained based on (1.19) and the 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. 4.2 (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 (4.19)). 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 (4.21).

We simulated each of the three cases for four initial conditions, $\underline{x}(0) = [2.4, -0.1, 0.6, 0.1]^T$, $[1.6, 0.05, -0.6, -0.05]^T$, $[-1.6, -0.05, 0.6, 0.05]^T$, and $[-2.4, 0.1, -0.6, -0.1]^T$, which were arbitrarily chosen in $U = [-5, 5] \times [-2, 2] \times [-\pi/4, \pi/4] \times [-0.8, 0.8]$. Figures 4.3-4.5 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 2nd/3rd 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 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 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; and, c) using the same small number of state-control pairs but adding the fuzzy control rules (4.22)-(4.25), 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 (4.21)) the linguistic fuzzy control rules into the controller.

We also simulated two extreme cases: (i) using the original controller of [18],

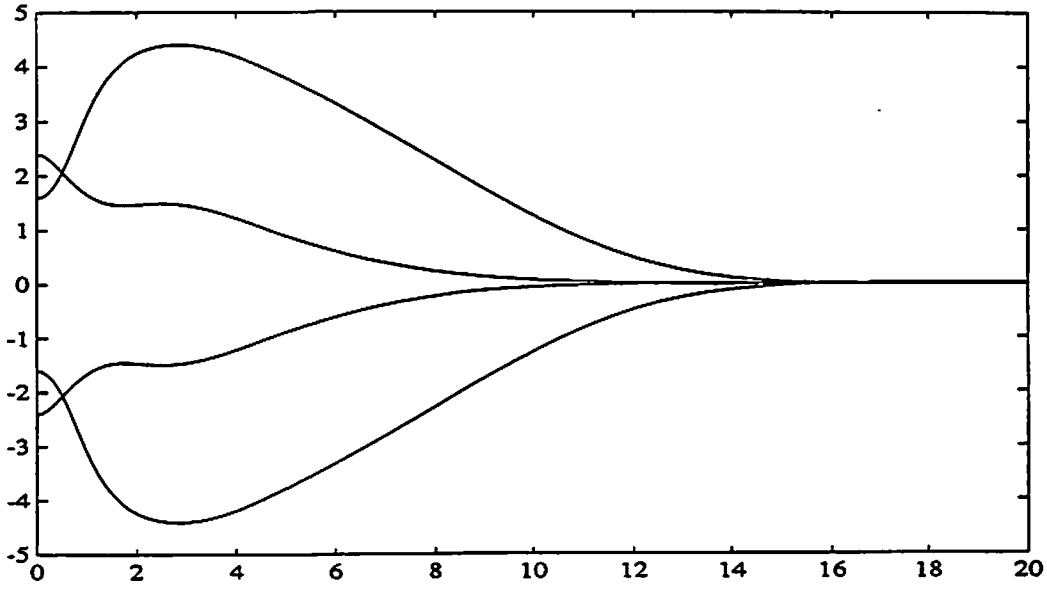


Figure 4.3: Outputs of the closed-loop ball and beam system for Case 1 and four initial conditions.

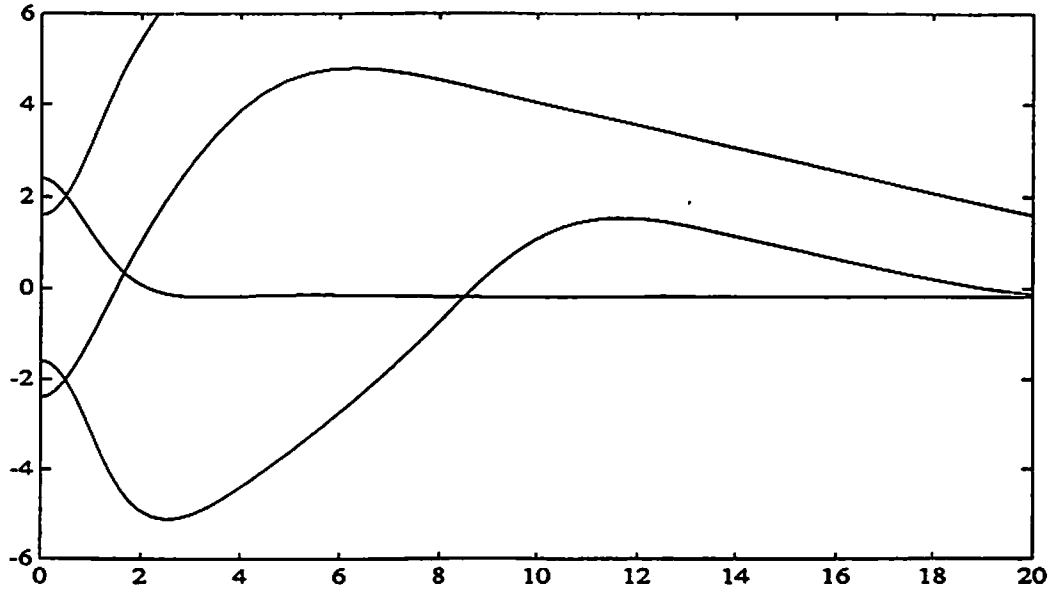


Figure 4.4: Outputs of the closed-loop ball and beam system for Case 2 and four initial conditions.

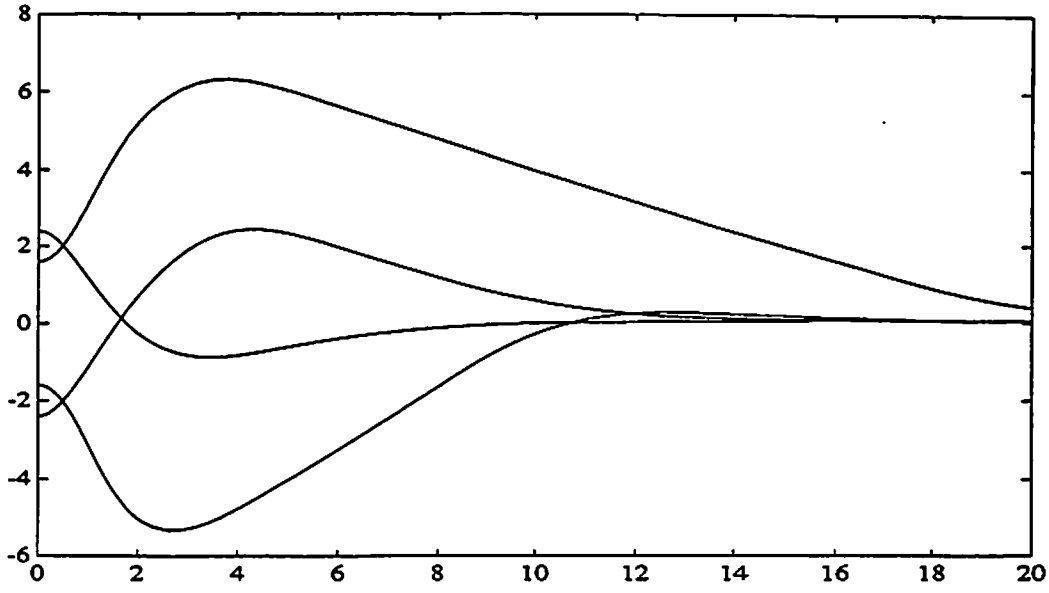


Figure 4.5: Outputs of the closed-loop ball and beam system for Case 3 and four initial conditions.

and (ii) using only the pure linguistic controller $f^L(\underline{x})$ based on $R_1^L - R_4^L$, for the same initial conditions as in Cases 1-3. Figures 4.6 and 4.7 show the output of the closed-loop system for the two cases (i) and (ii), respectively. Comparing Fig. 4.6 with Figs. 4.3-4.5 we see that the original controller of [18] gave the best performance; this is to be expected because we used the FBF expansions to approximate this controller. Figure 4.7 shows that if we use the FBF expansion controller based only on the four linguistic rules (4.22)-(4.25), the closed-loop system is unstable, i.e., pure fuzzy logic controller with only four linguistic rules is not sufficient to control the system.

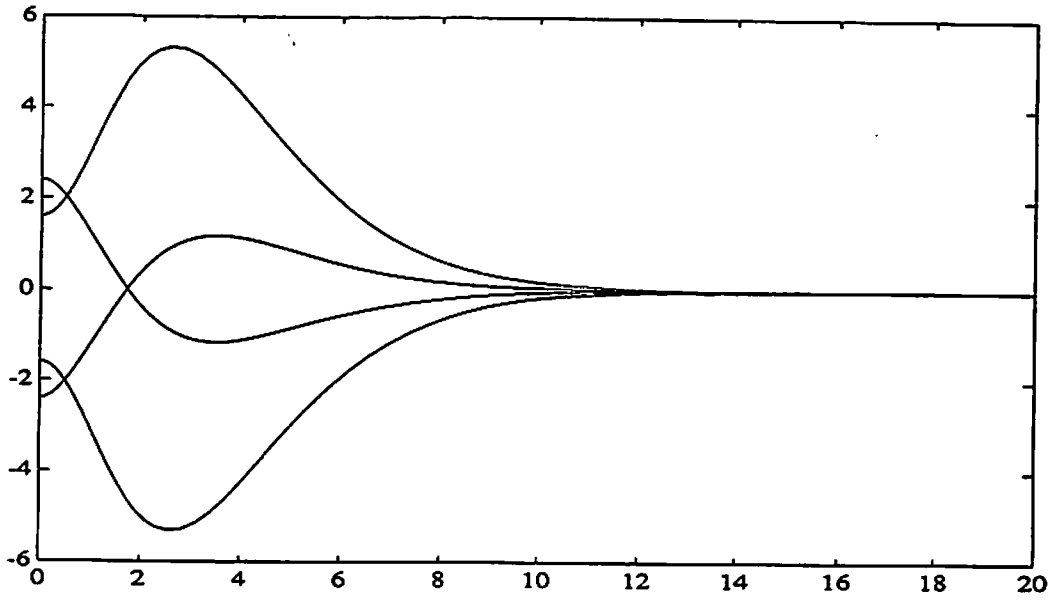


Figure 4.6: Outputs of the closed-loop ball and beam system using the input-output linearization algorithm of [18] and four initial conditions.

4.5 Modeling the Mackey-Glass Chaotic Time Series by FBF Expansion

Figure 4.8 shows a section of 1000 points (from $t = 1001$ to 2000) of the Mackey-Glass chaotic time series generated by the differential equation [30]

$$\frac{dx(t)}{dt} = \frac{0.2x(t-30)}{1+x^{10}(t-30)} - 0.1x(t). \tag{4.26}$$

Our task is to determine a FBF expansion as a one-step-ahead predictor for the chaotic time series based on the data section of Fig. 4.8; i.e., we shall determine

$$\hat{x}(t) = f(\underline{z}(t)), \tag{4.27}$$

where $f(*)$ is in the form of (5), and $\underline{z}(t) = [x(t-1), ..., x(t-n)]^T$ are past observations of the series.

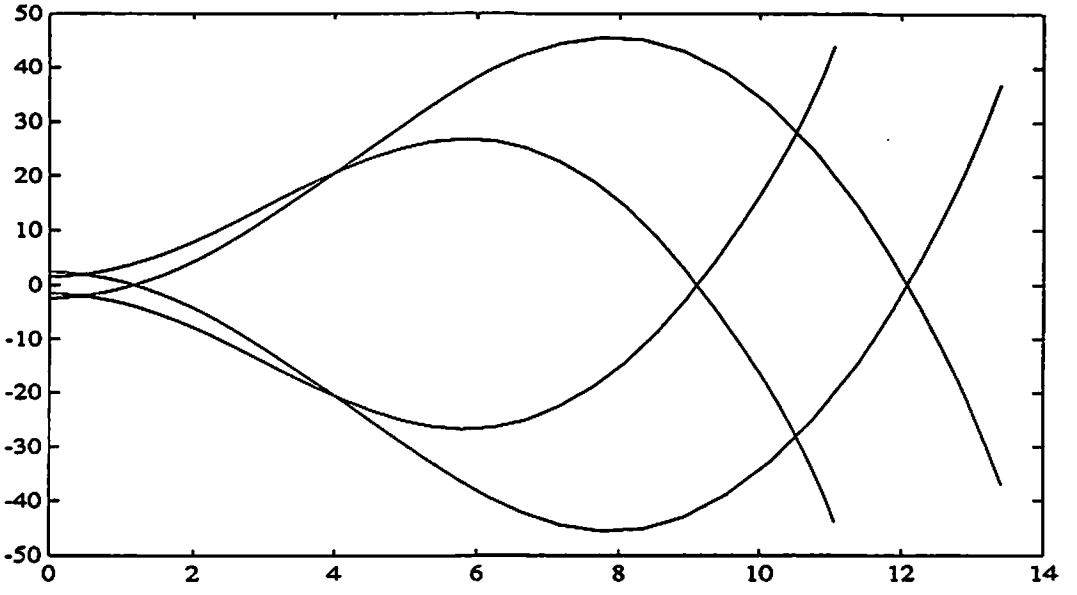


Figure 4.7: Outputs of the closed-loop ball and beam system using the pure fuzzy logic controller based on the four linguistic rules (4.22)-(4.25) and four initial conditions.

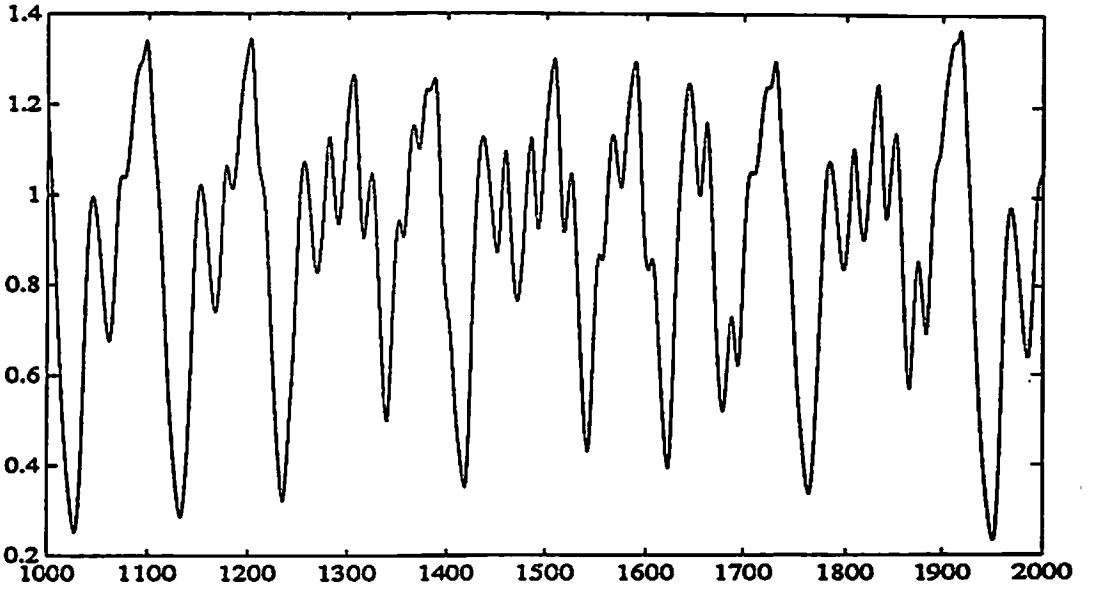


Figure 4.8: A section of the Mackey-Glass chaotic time series.

We chose $M_s = 80$ and $n = 10$. The residuals determined by

$$e(t) = x(t) - \hat{x}(t) \tag{4.28}$$

are shown in Fig. 4.9 for $t = 1001$ to 2000. The series $x(t)$ and $\hat{x}(t)$ look almost indistinguishable if we depict them on the same plot.

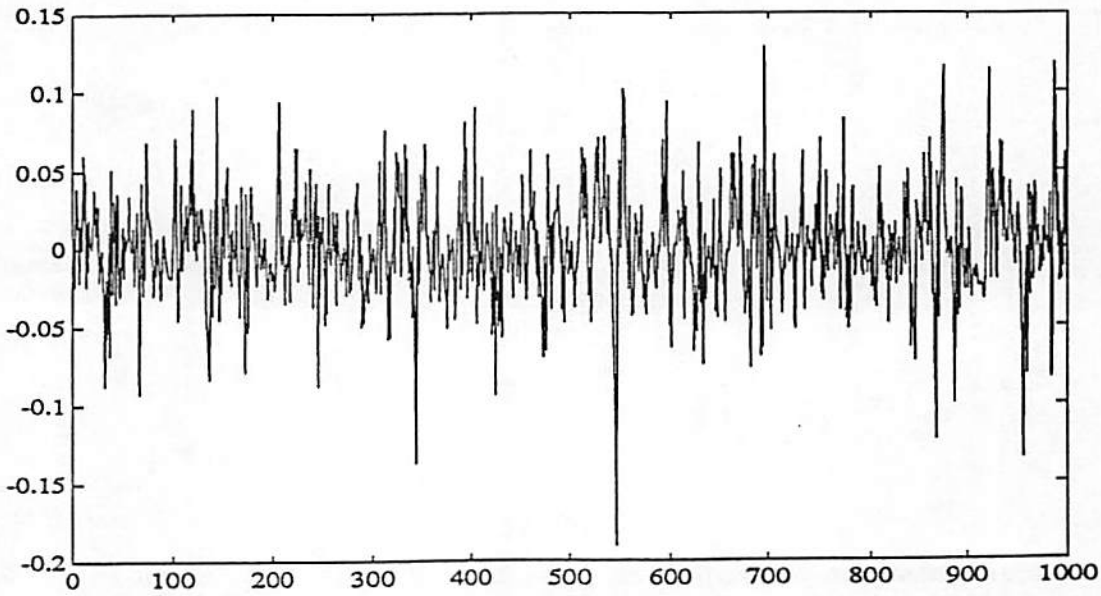


Figure 4.9: Residual sequence using the FBF expansion predictor for the chaotic time series of Fig. 4.8.

We performed two kinds of statistical tests — autocorrelation test and Chi-squared test — in order to determine whether the final FBF expansion is a valid model for the chaotic time series. The autocorrelations of $e(t)$ are plotted in Fig. 4.10. To perform the Chi-squared test, define

$$\underline{r}(t) = [w(t), ..., w(t - \eta + 1)]^T, \tag{4.29}$$

where $w(t)$ is some function of the past observations and residuals, and let

$$\Gamma^T \Gamma = N^{-1} \sum_{t=1001}^{N+1000} \underline{r}(t) \underline{r}^T(t), \tag{4.30}$$

where $N = 1000$. The Chi-squared statistics are calculated according to [5] as

$$\zeta(\eta) = N\underline{\mu}^T(\Gamma^T\Gamma)^{-1}\underline{\mu}, \tag{4.31}$$

where

$$\underline{\mu} = N^{-1}\sum_{t=1001}^{N+1000}\underline{r}(t)e(t)/\sigma_e, \tag{4.32}$$

and σ_e^2 is the variance of the residuals $e(t)$. Figures 4.11 and 4.12 show the Chi-squared statistics $\zeta(\eta)$ for $w(t) = e^2(t-1)$ and $w(t) = e^2(t-1)y^2(t-1)$, respectively; we see that they are all within the 95% confidence band. Consequently, the model validity tests confirm that this FBF expansion is an adequate model for the chaotic time series.

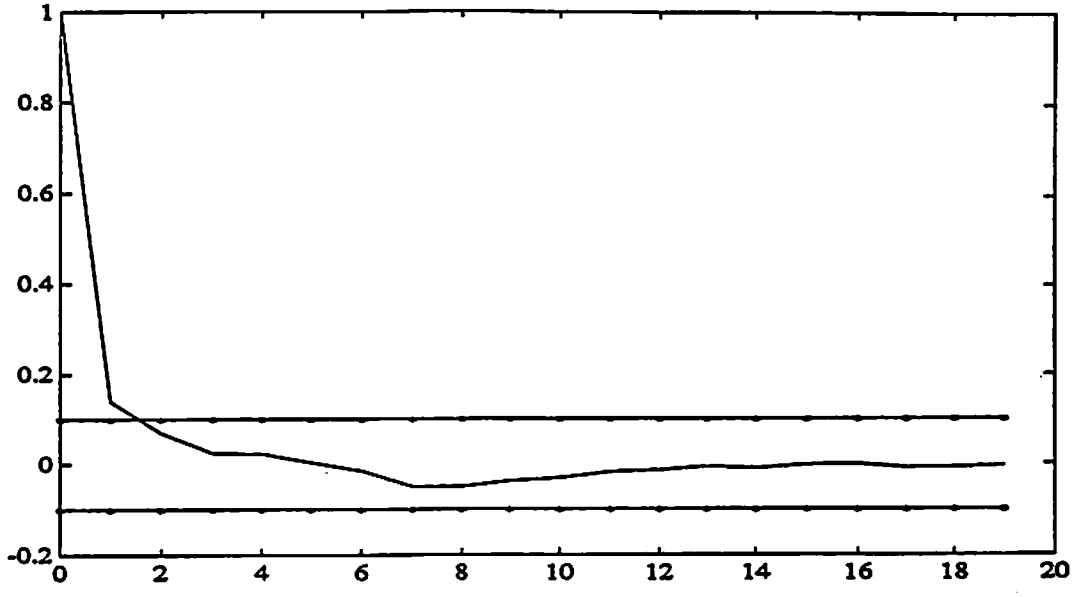


Figure 4.10: Autocorrelations of the residuals of Fig. 4.9. -*- represents 95% confidence band.

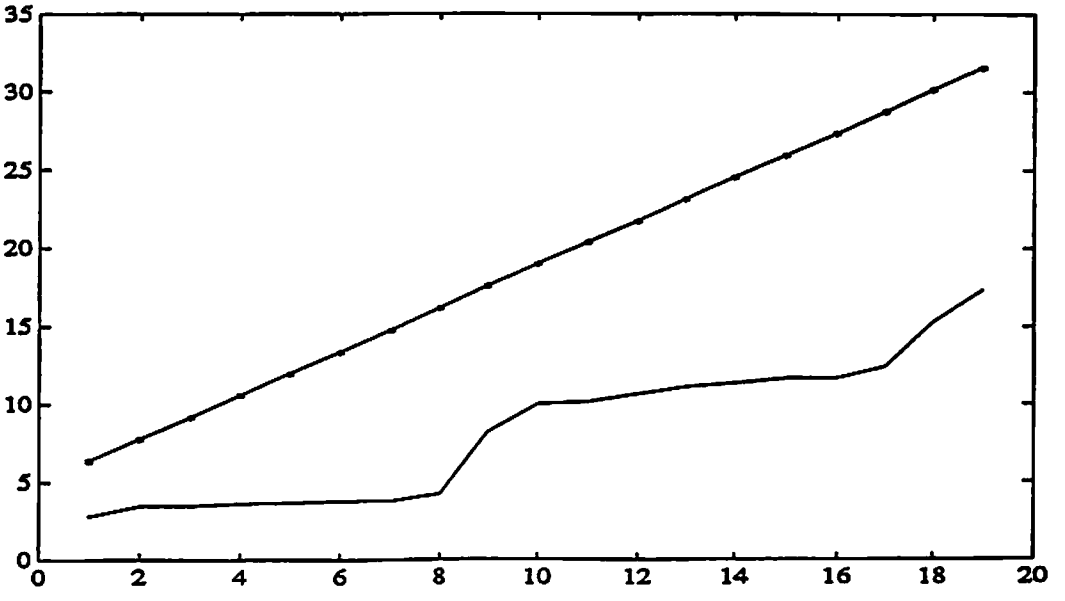


Figure 4.11: Chi-squared statistics $\zeta(\eta)$ for $w(t) = e^2(t-1)$. -*- represents 95% confidence band.

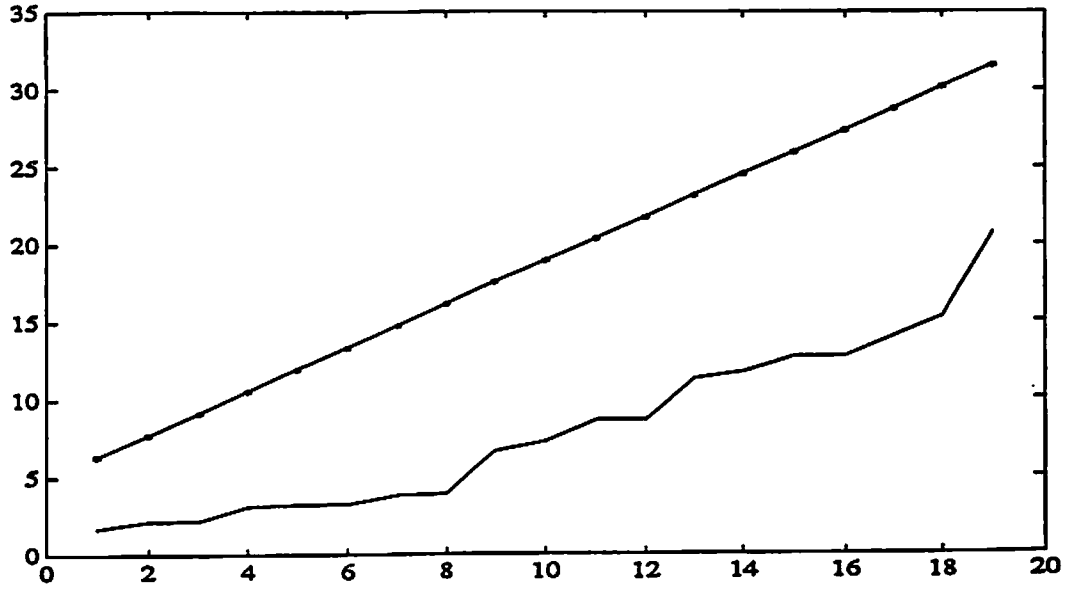


Figure 4.12: Chi-squared statistics $\zeta(\eta)$ for $w(t) = e^2(t-1)y^2(t-1)$. -*- represents 95% confidence band.

4.6 Conclusions

In this chapter, we: 1) showed that fuzzy systems can be represented as linear combinations of fuzzy basis functions; 2) developed an orthogonal least squares algorithm to select the significant fuzzy basis functions; and, 3) used the fuzzy basis function expansions as controllers for the ball and beam system and as predictors for the Mackey-Glass chaotic time series. 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 greatly improved by incorporating these linguistic fuzzy control rules.

Chapter 5

DESIGN OF FUZZY SYSTEMS USING A SIMPLE ONE-PASS METHOD

5.1 Introduction

The two design methods in Chapters 3 and 4 are not simple, in the sense that they may require intensive computations, since the back-propagation algorithm performs a nonlinear search procedure and the orthogonal least squares algorithm needs iterative operations. In this Chapter, we develop a very simple method for fuzzy system design which performs a one-pass operation on the numerical input-output pairs and linguistic fuzzy IF-THEN rules. The key idea of this method is to generate fuzzy rules from input-output pairs, collect the generated rules and linguistic rules from human experts into a common fuzzy rule base, and construct a final fuzzy system based on the combined fuzzy rule base.

5.2 Generating Fuzzy Rules from Numerical Data

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

$$(x_1^{(1)}, x_2^{(1)}; y^{(1)}), (x_1^{(2)}, x_2^{(2)}; y^{(2)}), \cdots \tag{5.1}$$

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 (5.1), and use these fuzzy rules to determine a mapping $f : (x_1, x_2) \rightarrow y$.

Our new 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), ..., S1 (Small 1), CE (Center), B1 (Big 1), ..., BN (Big N), and assign each region a fuzzy membership function. Figure 5.1 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 neighbouring 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. 5.1 has degree 0.8 in B1, degree 0.2 in B2, and zero degrees in all other regions. Similarly, $x_2^{(2)}$ in Fig. 5.1 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. 5.1 is considered to be B1, and $x_2^{(2)}$ in Fig. 5.1 is considered to be CE.

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

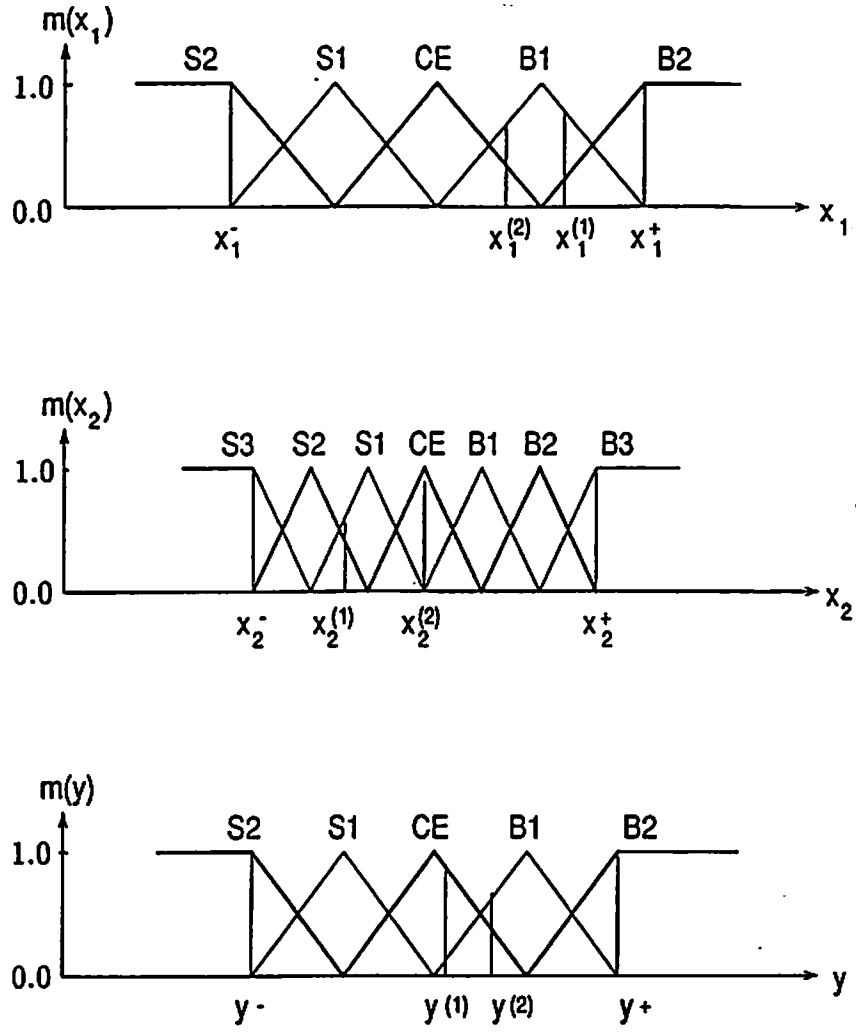


Figure 5.1: Divisions of the input and output spaces into fuzzy regions and the corresponding membership functions.

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

$(x_1^{(2)}, x_2^{(2)}; y^{(2)}) \Rightarrow [x_1^{(2)}(0.6 \text{ in B1, max}), x_2^{(2)}(1 \text{ in CE, max}) ; y^{(2)}(0.7 \text{ in B1, max})] \Rightarrow$ 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 here, 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 which have the same IF part but a 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

$$D(Rule) = m_A(x_1)m_B(x_2)m_C(y). \tag{5.2}$$

As examples, Rule 1 has degree

$$\begin{aligned} D(Rule1) &= m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y) \\ &= 0.8 \times 0.7 \times 0.9 = 0.504; \end{aligned} \tag{5.3}$$

(see Fig. 5.1) and Rule 2 has degree

$$\begin{aligned} D(Rule2) &= m_{B1}(x_1)m_{CE}(x_2)m_{B1}(y) \\ &= 0.6 \times 1 \times 0.7 = 0.42. \end{aligned} \tag{5.4}$$

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 which 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

$$D(Rule1) = m_{B1}(x_1) m_{S1}(x_2) m_{CE}(y) m^{(1)}, \tag{5.5}$$

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 which 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 the Fuzzy Rule Base is illustrated in Fig. 5.2. 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 which 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. 5.2; 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.

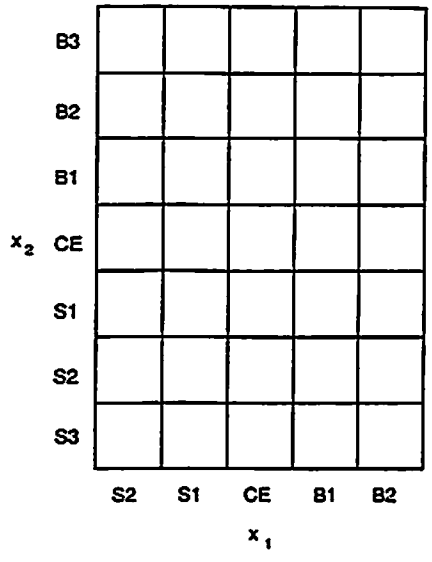


Figure 5.2: Illustration of a fuzzy rule base.

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 i'th fuzzy rule using *product* operations to determine the degree, $m^i_{O^i}$, of the output control corresponding to (x_1, x_2), i.e.,

$$m^i_{O^i} = m_{I^i_1}(x_1) m_{I^i_2}(x_2), \tag{5.6}$$

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

$$m^1_{CE} = m_{B1}(x_1) m_{S1}(x_2); \tag{5.7}$$

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

$$y = \frac{\sum_{i=1}^{K} m_{O^i}^i \bar{y}^i}{\sum_{i=1}^{K} m_{O^i}^i} \tag{5.8}$$

where $\bar{y}^i$ denotes the center value of region O^i (the *center* of a fuzzy region is defined as the point which 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 $m_{O^i}^i$ in Eq. (5.6) with $m_{O_j^i}^i$, where j denotes the j^{th} component of the output vector (O_j^i is the region of Rule i for the j^{th} output component; $m_{O_j^i}^i$ is the same for all j), and change Eq. (5.8) to

$$y_j = \frac{\sum_{i=1}^{K} m_{O_j^i}^i \bar{y}_j^i}{\sum_{i=1}^{K} m_{O_j^i}^i} \tag{5.9}$$

where $\bar{y}_j^i$ denotes the center of region O_j^i.

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 5.4). Our new method essentially "learns" from the "examples" and expert rules to obtain a mapping which, 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..

5.3 Application to Truck Backer-Upper Control

Backing a truck to a loading dock is a difficult exercise. It is a non-linear control problem for which no traditional control system design methods exist. A neural network controller for the truck backer-upper problem was developed in [52], whereas a fuzzy control strategy for the same problem was proposed in [28]. The neural network controller of [52] only uses numerical data, and cannot utilize linguistic rules determined from expert drivers; on the other hand, the fuzzy controller of [28] only uses linguistic rules, and cannot utilize sampled data. Since the truck backer-upper control problem is a good example of the kind of control system design problems that replace a human controller by a machine, it is interesting to apply the approach developed in Section 5.2 to this problem. In order to distinguish these methods, we call the method of [28] the "fuzzy approach", the method of [52] the "neural approach", and our new method the "numerical-fuzzy approach".

The results of [28] 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 [28] 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 5.1: 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 performances.

Statement of the Truck Backer-Upper Control Problem

The simulated truck and loading zone are shown in Fig. 5.3. The truck corresponds to the cab part of the neural truck in the Nguyen-Widrow [52] 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. 5.3. 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°) = (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,

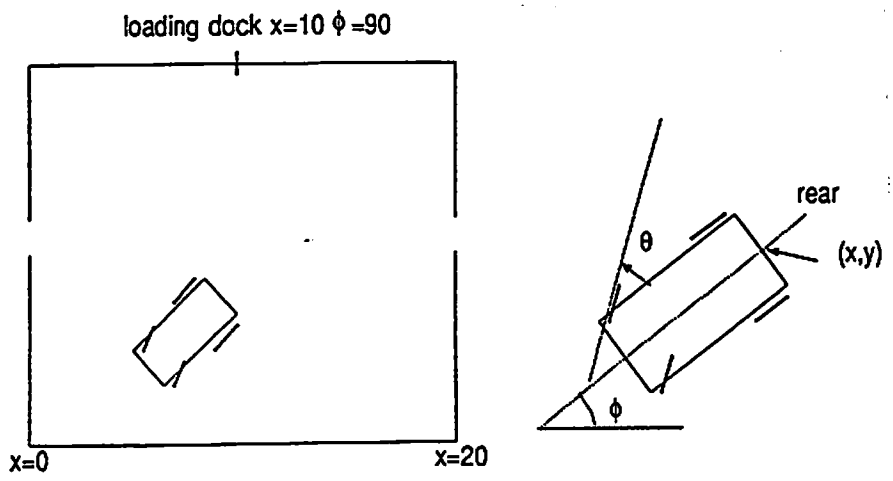


Figure 5.3: Diagram of simulated truck and loading zone.

we needed to know the dynamics of the truck backer-upper procedure. We used the following approximate kinematics (see [83] for details) :

$$x(t+1) = x(t) + cos[\phi(t) + \theta(t)] + sin[\theta(t)]sin[\phi(t)] \tag{5.10}$$

$$y(t+1) = y(t) + sin[\phi(t) + \theta(t)] - sin[\theta(t)]cos[\phi(t)] \tag{5.11}$$

$$\phi(t+1) = \phi(t) - sin^{-1}[\frac{2sin(\theta(t))}{b}] \tag{5.12}$$

where b is the length of the truck. We assumed $b = 4$ in the simulations of this section. Equations (5.10) to (5.12) were used to obtain the next state when the present state and control are given. Since y is not considered a state, only Eqs. (5.10) and (5.12) were used in the simulations. We wrote Eq. (5.11) here for the purpose of showing the complete dynamics of the truck. Observe, from Eqs. (5.10)-(5.12), that even this simplified dynamic model of the truck is nonlinear. The 14 sequences of desired $(x, \phi; \theta)$ pairs are given in [83]; we only include one such sequence in Table 5.1.

Table 5.1

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			

Table 5.2

Fuzzy rules generated from the desired input-output pairs of Table 1, 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				

Neural Control and Simulation Results

We used a two-input single-output three-layer back-propagation neural network [60] for our control task. 20 hidden neurons were used, and a sigmoid non-linear 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 $[-40^o, 40^o]$, i.e., if the neuron output is $g(t)$, the corresponding output $\theta(t)$ is

$$\theta(t) = 80g(t) - 40. \tag{5.13}$$

In the simulations, we normalized $[-40^o, 40^o]$ 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 [52]. First, we have only one neural network which 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 [52] connect their neural network stage by stage and train these 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 [52]). 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 [60] 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 Eqs. (5.10)-(5.12). Three arbitrarily chosen initial states, (x_0, ϕ_0^o) = (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. 5.4. 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 5.2 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. 5.5, 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 [83]; we only show the generated rules for the data pairs of Table 5.1 in Table 5.2. The final Fuzzy Rule Base is shown in Fig. 5.6 (this is the result of Step 4 of our method in Section 2; here we assume that no linguistic rules are available). We see from Fig. 5.6 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. 5.6 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, ϕ_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. 5.4; hence, Fig. 5.4 also shows the truck trajectories using the numerical-fuzzy controller.

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

Example 5.2: 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

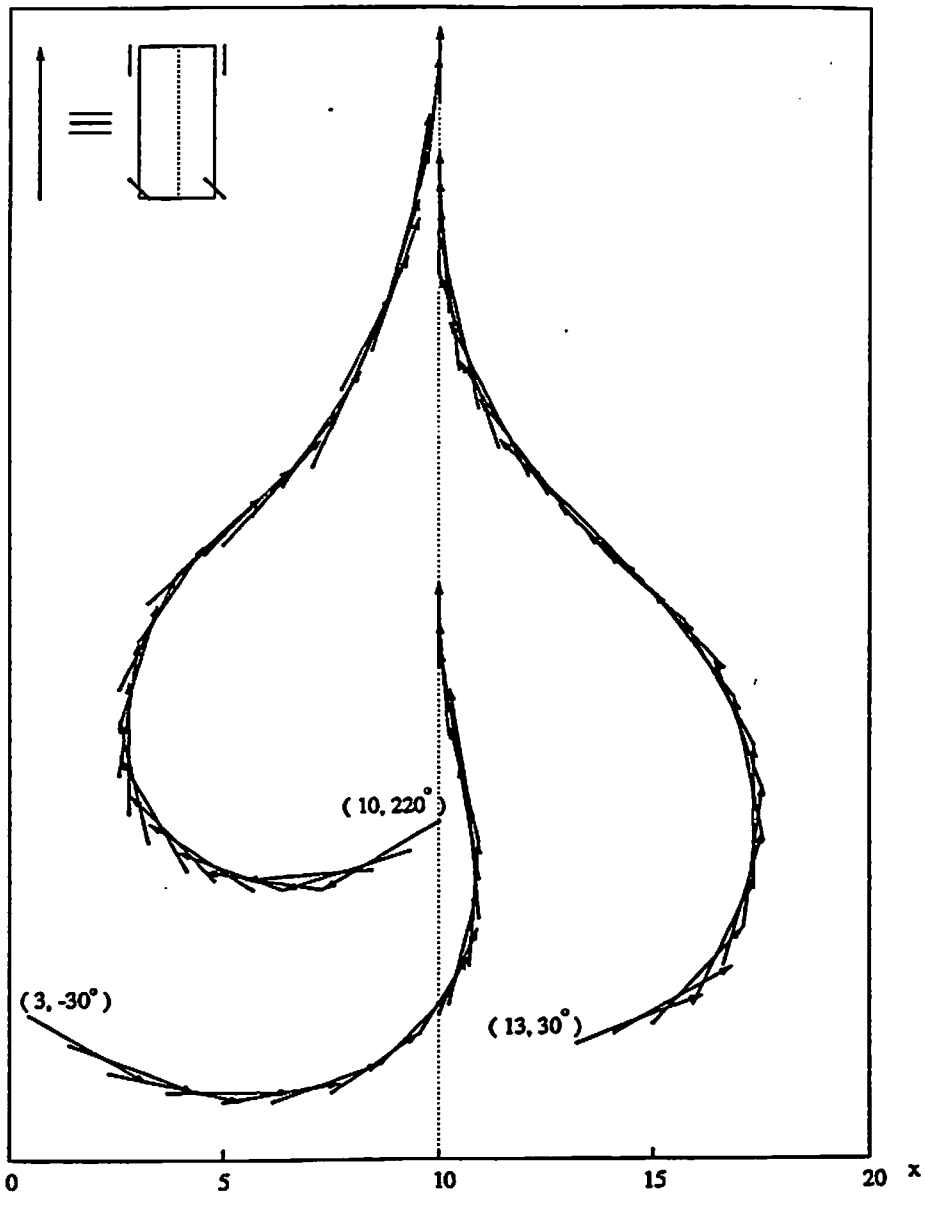
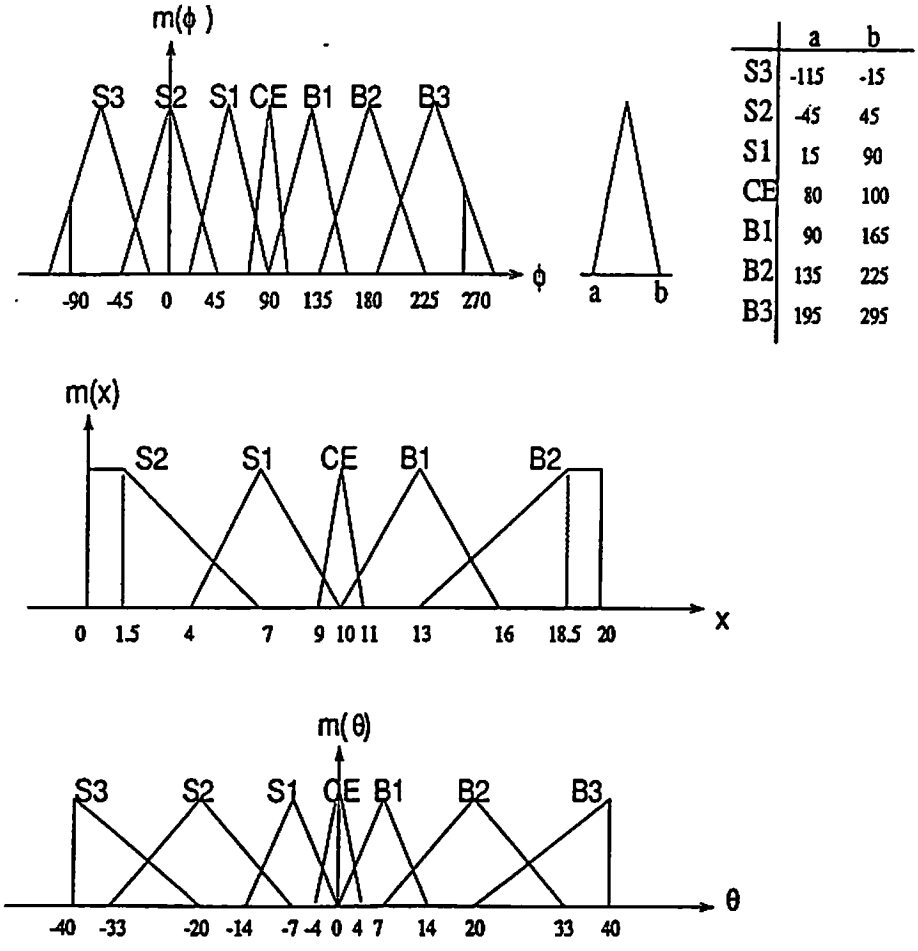


Figure 5.4: Truck trajectories using the neural controller and the numerical-fuzzy controller.



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

Figure 5.5: Fuzzy membership functions for the truck backer-upper control problem.

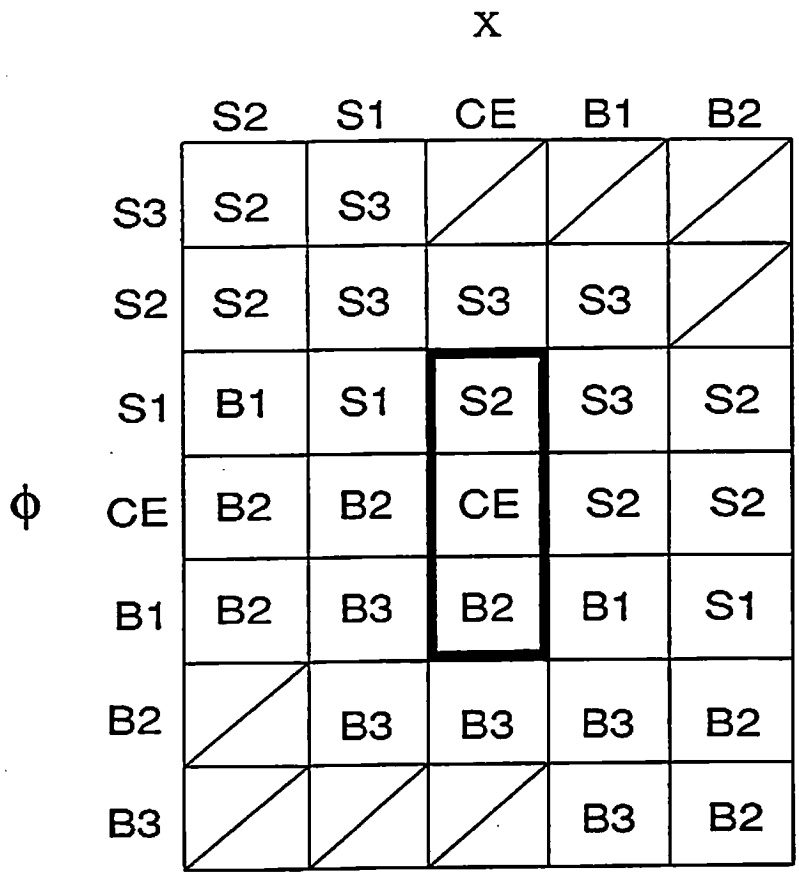


Figure 5.6: The final fuzzy rule base generated from the numerical data for the truck backer-upper control problem.

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. 5.6 except that 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 which are the same as the three center rules of Fig. 5.6.

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. 5.6; hence, the truck trajectories for this case must be the same as those using the Fuzzy Rule Base of Fig. 5.6. For each of the cases, we simulated the system starting from the following three initial states: (x_0, ϕ_0°) = (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. 5.7, 5.8 and 5.4, 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.

5.4 Application to Time-Series Prediction

Time-series prediction is a very important practical problem [4]. 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 $z(k)$ $(k = 1, 2, 3, ...)$ be a time series. The problem of time-series prediction can be formulated as: given $z(k-m+1), z(k-m+2), ..., z(k)$, determine $z(k + l)$, where m and l are fixed positive integers; i.e., determine a mapping from $[z(k - m + 1), z(k - m + 2), ..., z(k)] \in R^m$ to $[z(k + l)] \in R$.

A feedforward neural network can also be used for this problem [30]. 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 $[z(k-m+1), z(k-$

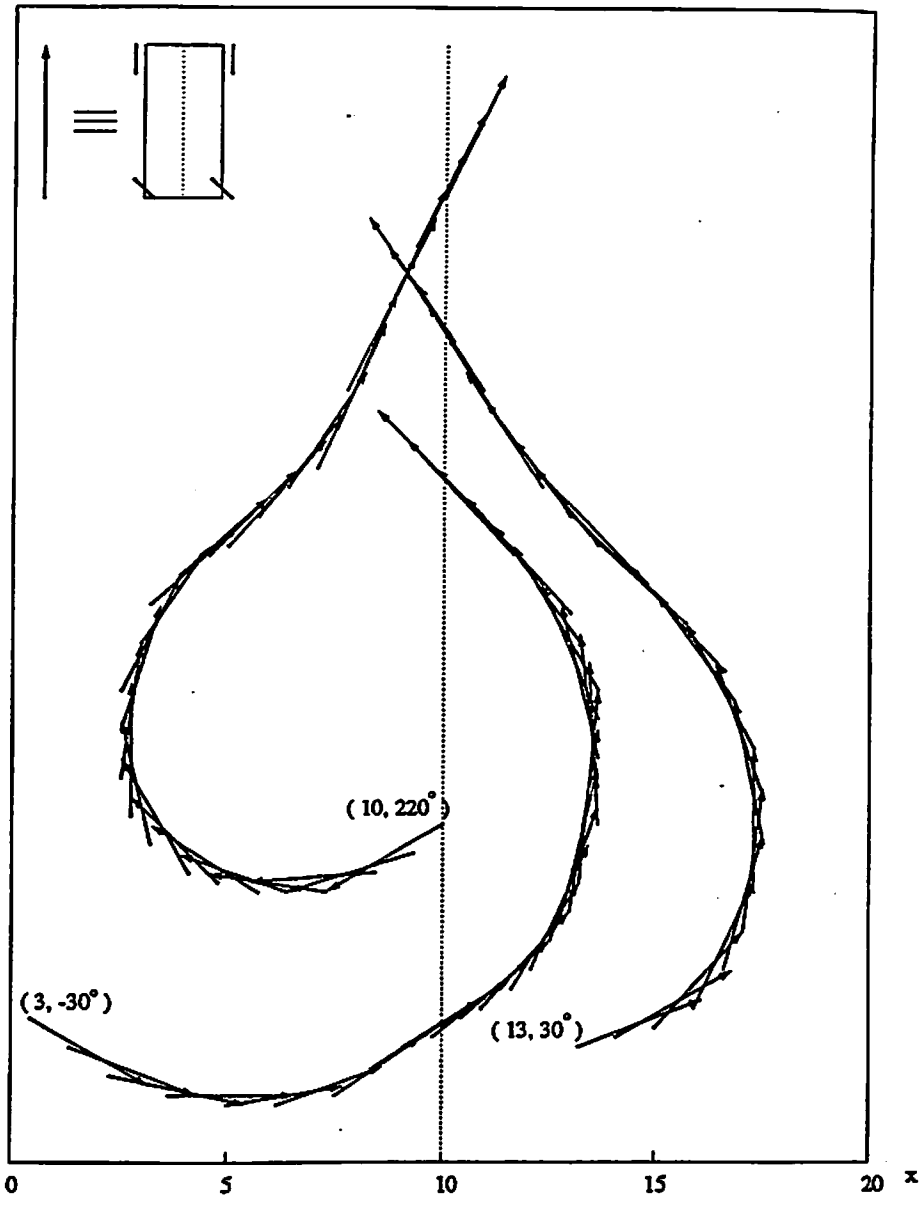


Figure 5.7: Truck trajectories using the fuzzy rules from the truncated data pairs only.

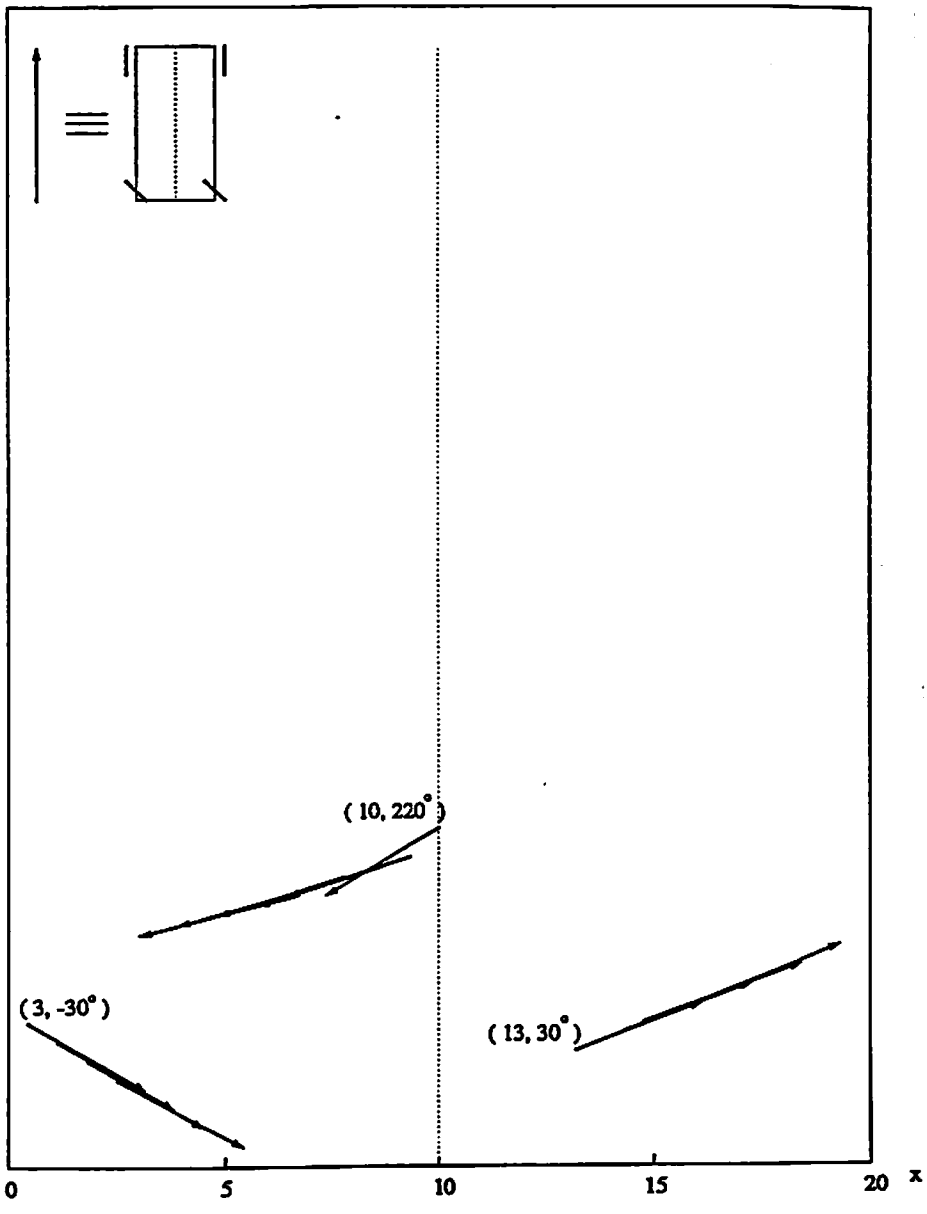


Figure 5.8: Truck trajectories using the selected linguistic rules only.

$m+2),...,z(k)]$ 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 $z(1), z(2), ..., z(M)$ are given; then we form $M-m$ desired input-output pairs:

$$\begin{aligned}
&[z(M-m),...,z(M-1);z(M)]\\
&[z(M-m-1),...,z(M-2);z(M-1)]\\
&...\\
&[z(1),...,z(m);z(m+1)].
\end{aligned} \tag{5.14}$$

We train the neural network to match these $M-m$ pattern pairs using the error back-propagation algorithm [60].

Our numerical-fuzzy method in Section 5.2 can also be used for this time series prediction problem. Similar to the neural network approach, we assume that $z(1), z(2), ..., z(M)$ are given, and we form the $M-m$ desired input-output pattern pairs in (5.14). Steps 1-4 of our numerical-fuzzy approach are used to generate a Fuzzy Rule Base based on the pattern pairs (5.14); then this Fuzzy Rule Base is used to forecast $z(M+p)$ for $p=1,2,...$ using the defuzzifying procedure of Step 5 of our numerical-fuzzy method, where the inputs to the network are $z(M+p-m), z(M+p-m+1), ..., z(M+p-1)$.

Example 5.3: Here we apply our numerical-fuzzy approach to prediction of the Mackey-Glass chaotic time-series [30]. 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 [19]. In [30], 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 applied to the same Mackey-Glass chaotic time series in [30], 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:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{5.15}$$

When $\tau > 17$, Eq. (5.15) shows chaotic behavior. Higher values of τ yield higher dimensional chaos. In our simulation, we chose the series with $\tau = 30$. Figure 5.9 shows 1000 points of this chaotic series which we used to test both the numerical-fuzzy and neural approaches.

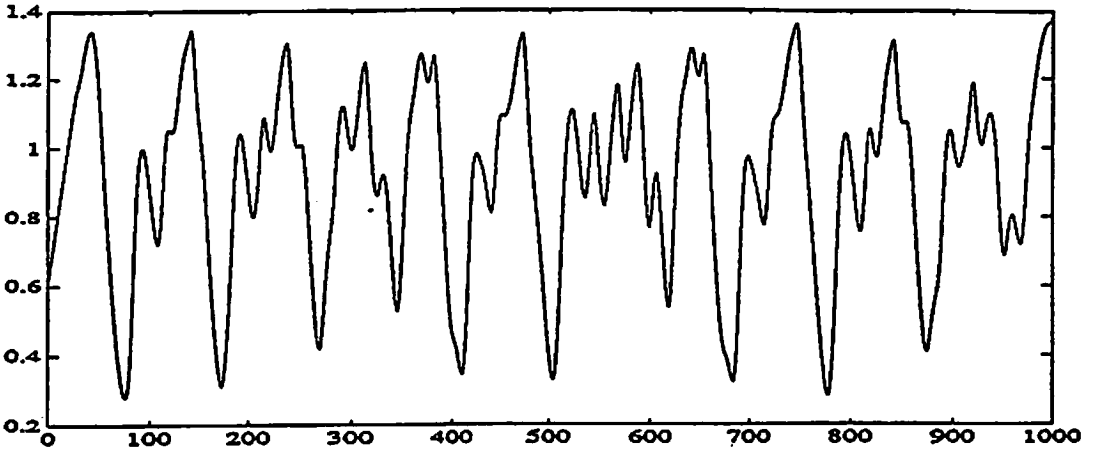


Figure 5.9: A section of the Mackey-Glass chaotic time series.

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. 5.10 for the numerical-fuzzy predictor (later, we 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 points were used as test data (for additional cases, see [83]). 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. Figures 5.11 and 5.12 show the results of the numerical-fuzzy and neural predictors respectively for case (1); and, Figs. 5.13 and 5.14 show similar results for case (2). As in [30], 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. 5.11 to 5.14 that our new numerical-fuzzy predictor gave about the same results as the neural network predictor.

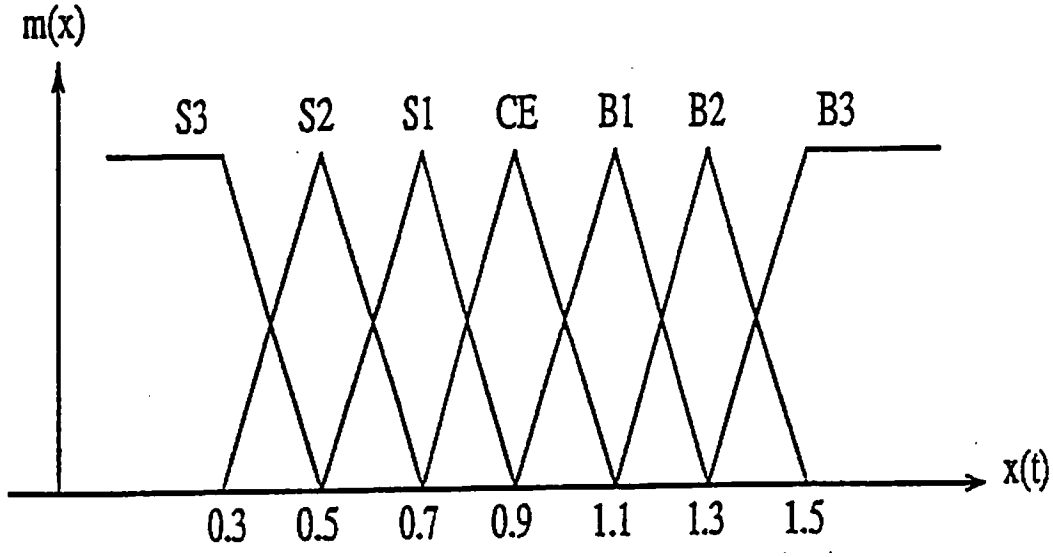


Figure 5.10: The first choice of membership functions for the chaotic time series prediction problem.

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 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. 5.9: 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

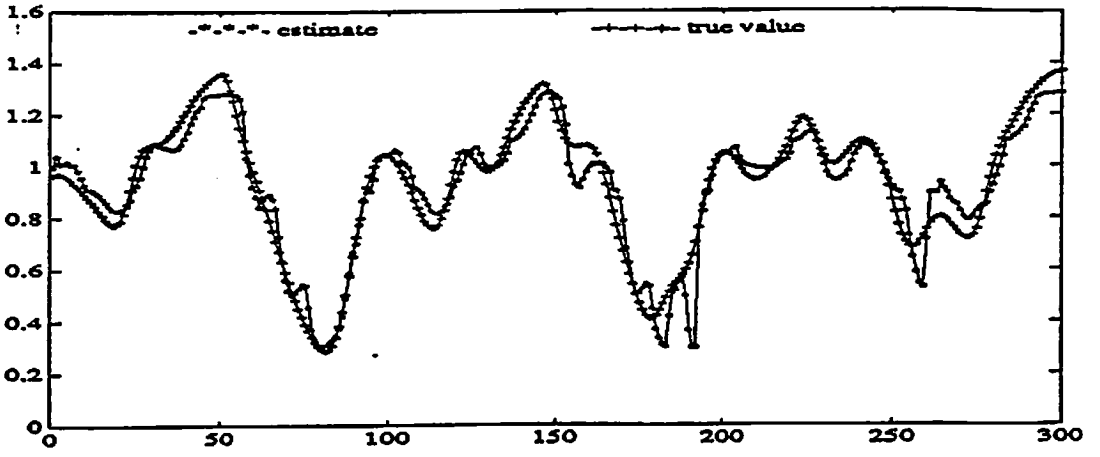


Figure 5.11: Prediction of the chaotic time series from x(701) to x(1000) using the numerical-fuzzy predictor when 200 training data (from x(501) to x(700)) are used.

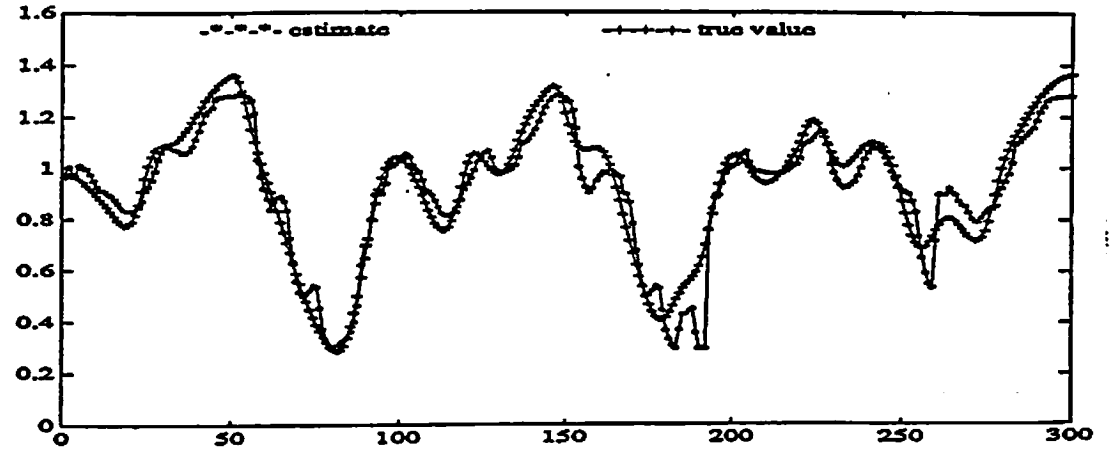


Figure 5.12: Prediction of the chaotic time series from x(701) to x(1000) using the neural predictor when 200 training data (from x(501) to x(700)) are used.

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. 5.15. Comparing Figs. 5.15 and 5.13 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. 5.16, and the other with the membership function shown in Fig. 5.17. We used the adaptive-Fuzzy-Rule-Base procedure for both simulations. The results are shown in Figs. 5.18 and 5.19, where Fig. 5.18 (5.19) shows the result corresponding to the membership function of Fig. 5.16 (5.17). Comparing Figs. 5.15, 5.18 and 5.19 we see very clearly that we obtain better and better results as the "domain interval" is divided finer and finer. Figure 5.19 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.

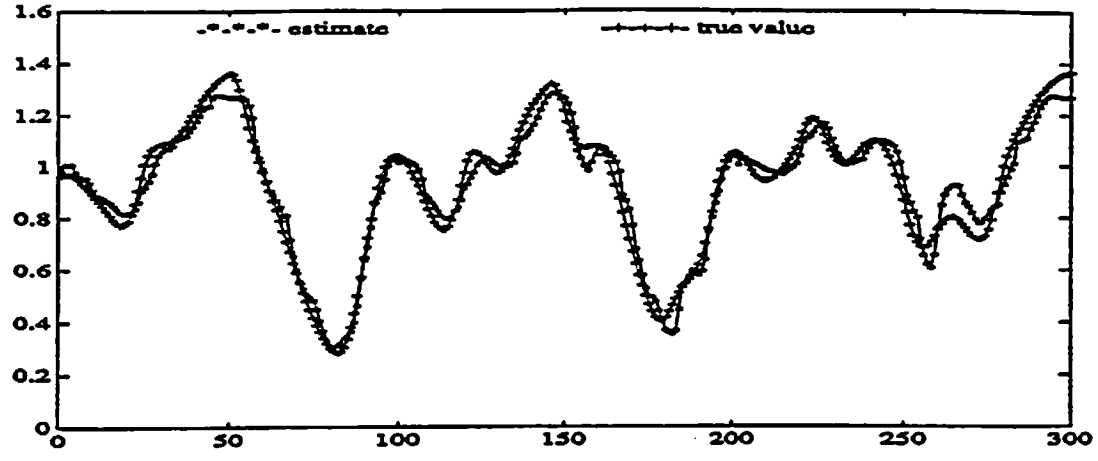


Figure 5.13: Prediction of the chaotic time series from x(701) to x(1000) using the numerical-fuzzy predictor when 700 training data (from x(1) to x(700)) are used.

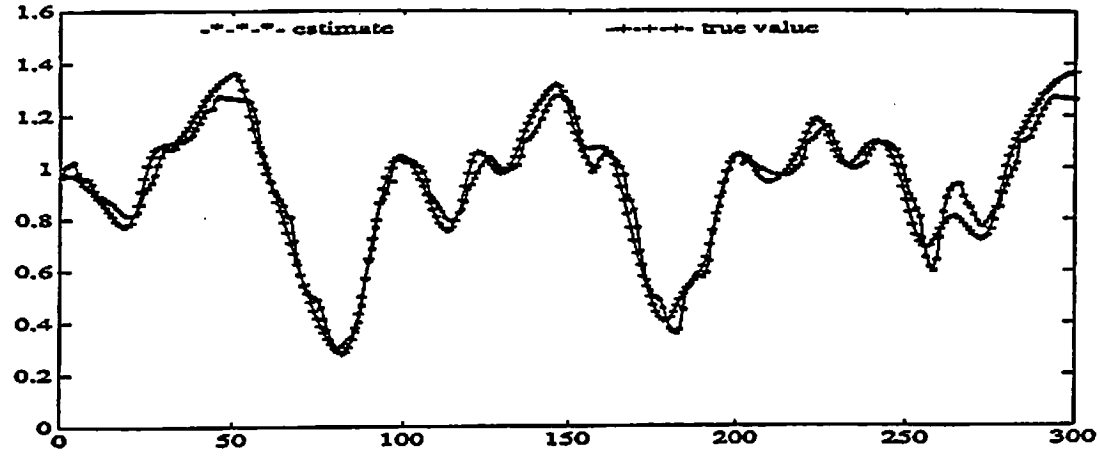


Figure 5.14: Prediction of the chaotic time series from x(701) to x(1000) using the neural predictor when 700 training data (from x(1) to x(700)) are used.

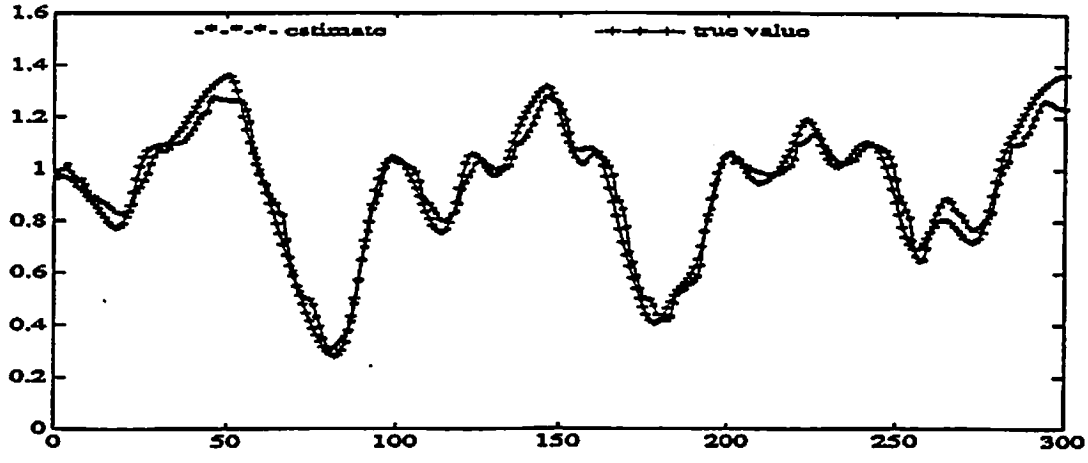


Figure 5.15: Prediction of the chaotic time series from x(701) to x(1000) using the updating fuzzy rule base procedure.

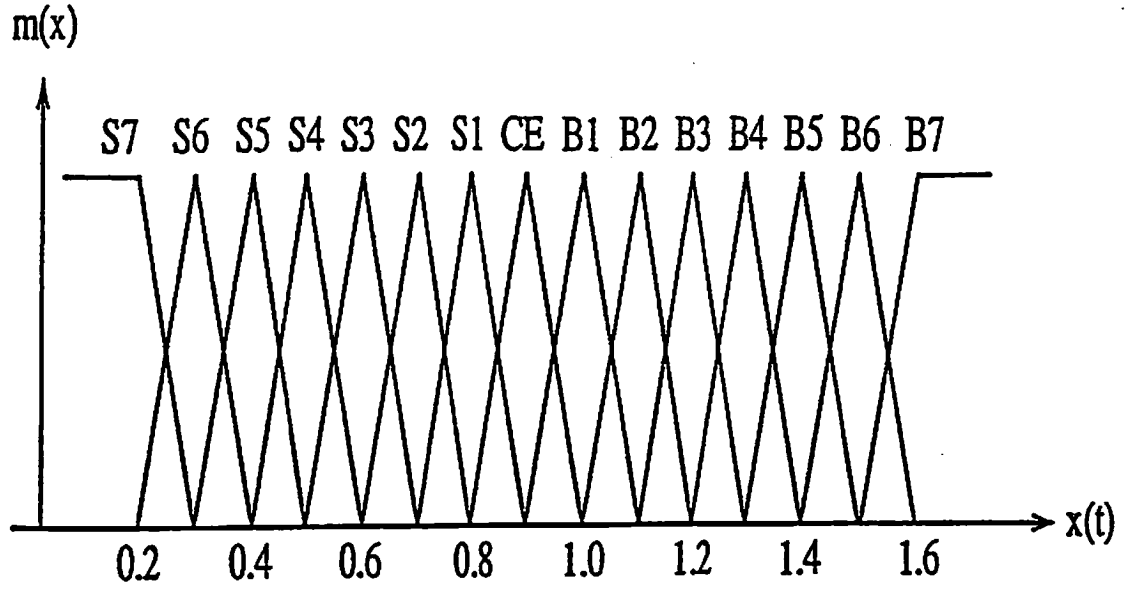


Figure 5.16: The second choice of membership functions for the chaotic time series prediction problem.

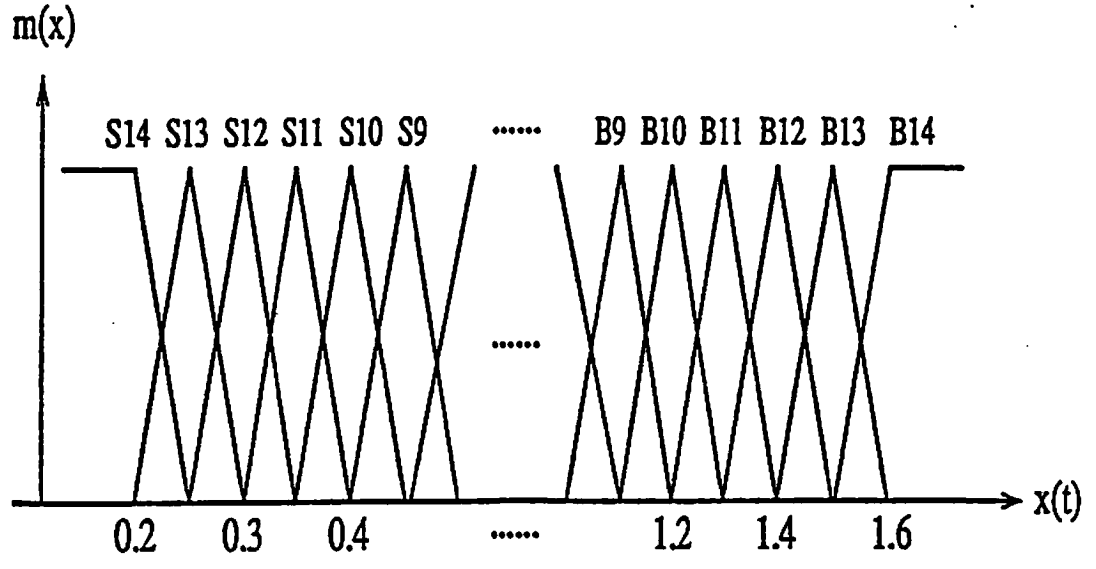


Figure 5.17: The third choice of membership functions for the chaotic time series prediction problem.

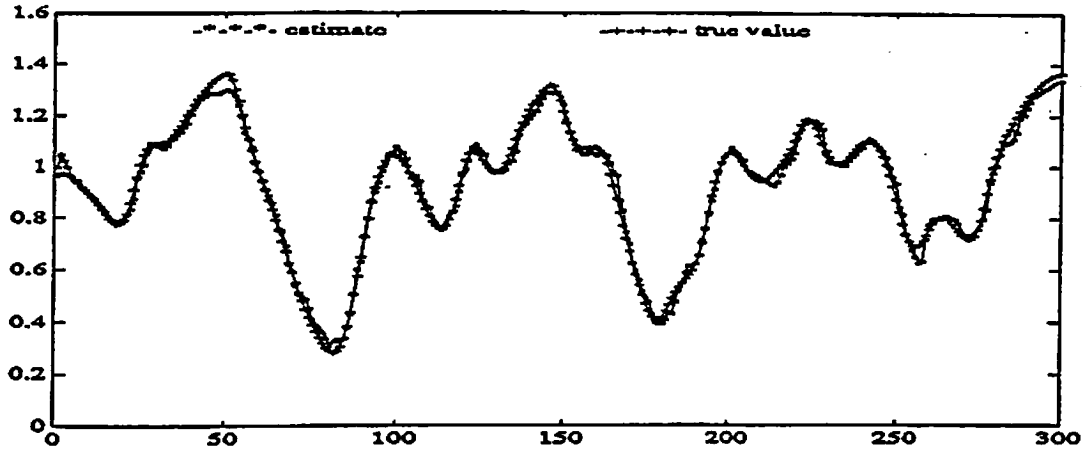


Figure 5.18: Prediction of the chaotic time series from x(701) to x(1000) using the updating fuzzy rule base procedure with the second choice of membership function.

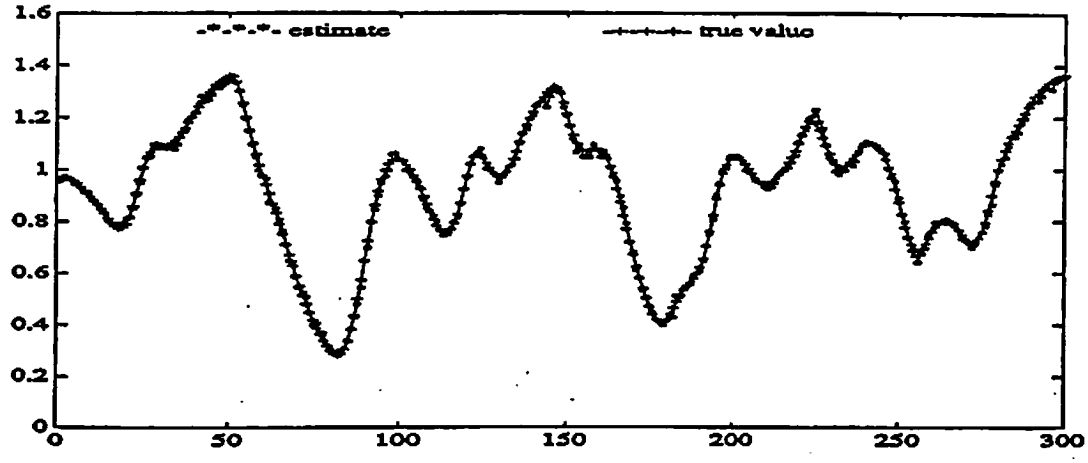


Figure 5.19: Prediction of the chaotic time series from x(701) to x(1000) using the updating fuzzy rule base procedure with the third choice of membership function.

5.5 Conclusions

In this chapter, 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 applied our new method to a truck backer-upper control problem, 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.

Chapter 6

FUZZY ADAPTIVE FILTERS, WITH APPLICATION TO NONLINEAR CHANNEL EQUALIZATION

6.1 Introduction

Filters are information processors. In practice, information usually comes from two sources: sensors which provide numerical data associated with a problem, and human experts who provide linguistic descriptions (often in the form of fuzzy IF-THEN rules) about the problem. Existing filters can only process numerical data, whereas existing expert systems can only make use of linguistic information; therefore, their successful applications are limited to problems (or portions of problems) where either linguistic rules or numerical data do not play a critical role. There are, however, a large number of practical problems in economics, seismology, management, etc., where both linguistic and numerical information are critical. At present, when we are faced with such problems, we use linguistic information, consciously or unconsciously, in the: choice among different filters, evaluation of filter performance, choice of filter orders, interpretation of filtering results, etc.. There are serious limitations to using linguistic information in this

way, because for most practical problems the linguistic information (in its natural form) is not about which kind of filter should be chosen or what the order of the filter should be, etc., but is in the form of IF-THEN rules concerning fuzzy concepts like "small", "hot", "not very fast", "very large but not very very large", etc.. The purpose of this chapter is to develop new kind of nonlinear adaptive filters, which we refer to as *fuzzy adaptive filters*, that make use of both linguistic and numerical information in their natural form, i.e., as fuzzy IF-THEN rules and input-output data pairs.

A fuzzy adaptive filter is constructed from a set of changeable fuzzy IF-THEN rules. These fuzzy rules come either from human experts or by matching input-output pairs through an adaptation procedure. The adaptive algorithms update the parameters of the membership functions which characterize the fuzzy concepts in the IF-THEN rules, by minimizing some criterion functions. Two fuzzy adaptive filters are developed in this chapter which use recursive least squares (RLS) and least mean squares (LMS) algorithms, respectively.

6.2 RLS Fuzzy Adaptive Filter

Our RLS fuzzy adaptive filter solves the following problem.

Problem 1: Consider a real-valued vector sequence $[\underline{x}(k)]$ and a real-valued scalar sequence $[d(k)]$, where $k = 0, 1, 2, ...$ is the time index, and $\underline{x}(k) \in U \equiv [C_1^-, C_1^+] \times [C_2^-, C_2^+] \times \cdots \times [C_n^-, C_n^+] \subset R^n$ (we call U and R the input and output spaces of the filter, respectively). At each time point k, we are given the values of $\underline{x}(k)$ and $d(k)$. The problem is: at each time point $k = 0, 1, 2, ...$, determine an adaptive filter $f_k : U \subset R^n \rightarrow R$ such that

$$J(k) = \sum_{i=0}^{k} \lambda^{k-i} [d(i) - f_k(\underline{x}(i))]^2 \tag{6.1}$$

is minimized, where $\lambda \in (0, 1]$ is a forgetting factor.

The above problem is quite general. As a particular example, consider the following time-series prediction problem: we measure the values of a bounded time-series $[y(k)]$ at each time point $k = 0, 1, 2, ...$; at time point $k - 1$, we want

to determine a filter $f_{k-1} : U \rightarrow R$ such that $f_{k-1}[y(k-1),...,y(k-n)]$ is an optimal prediction of $y(k)$ in some sense. For this problem, we have $\underline{x}(k) = [y(k-1),...,y(k-n)]^T$, $d(k) = y(k)$, and the "in some sense" means to minimize the $J(k)$ of (6.1). If we constrain the f_k's to be linear functions, the problem becomes an FIR adaptive filter design problem [11,96]. If the f_k's are Volterra series expansions, we have an adaptive polynomial filter design problem [43,63]. If the f_k's are multi-layer perceptrons or radial basis function expansions, the problem becomes the neural nets adaptive filter design problem [7,8].

Design Procedure of the RLS Fuzzy Adaptive Filter:

Step 1: Define m_i fuzzy sets in each interval $[C_i^-, C_i^+]$ of the input space U, which are labeled as F_i^{ji} $(i = 1,2,...,n,\ ji = 1,2,...,m_i)$, in the following way: the m_i membership functions $\mu_{F_i^{ji}}$ cover the interval $[C_i^-, C_i^+]$ in the sense that for each $x_i \in [C_i^-, C_i^+]$ there exists at least one $\mu_{F_i^{ji}}(x_i) \neq 0$.

Step 2: Construct a set of $\prod_{i=1}^n m_i$ fuzzy IF-THEN rules in the following form:

$$R^{(j1,...,jn)}: \ IF\ x_1\ is\ F_1^{j1}\ and\ \cdots\ and\ x_n\ is\ F_n^{jn},\ THEN\ d\ is\ G^{(j1,...,jn)}, \tag{6.2}$$

where $\underline{x} = (x_1,...,x_n)^T \in U$ (the filter input), $d \in R$ (the filter output), $ji = 1,2,...,m_i$ with $i = 1,2,...,n$, F_i^{ji}'s are the same labels of the fuzzy sets defined in Step 1, and the $G^{(j1,...,jn)}$'s are labels of fuzzy sets defined in the output space which are determined in the following way: if there are linguistic rules from human experts in the form of (6.2), set $G^{(j1,...,jn)}$ to be the corresponding linguistic terms of these rules; otherwise, set $\mu_{G^{(j1,...,jn)}}$ to be an arbitrary membership function over the output space R. *It is in this way that we incorporate linguistic rules into the fuzzy adaptive filter, i.e., we use linguistic rules to construct the initial filter.*

Step 3: Construct the filter f_k based on the $\prod_{i=1}^n m_i$ rules in Step 2 as follow:

$$f_k(\underline{x}) = \frac{\sum_{j1=1}^{m_1} \cdots \sum_{jn=1}^{m_n} \theta^{(j1,...,jn)}(\mu_{F_1^{j1}}(x_1) \cdots \mu_{F_n^{jn}}(x_n))}{\sum_{j1=1}^{m_1} \cdots \sum_{jn=1}^{m_n} (\mu_{F_1^{j1}}(x_1) \cdots \mu_{F_n^{jn}}(x_n))}, \tag{6.3}$$

where $\underline{x} = (x_1,...,x_n)^T \in U$, $\mu_{F_i^{ji}}$'s are membership functions defined in Step 1, and $\theta^{(j1,...,jn)} \in R$ is the point at which $\mu_{G^{(j1,...,jn)}}$ achieves its maximum value. Due to the way in which we defined the $\mu_{F_i^{ji}}$'s in Step 1, the denominator of

(6.3) is nonzero for all the points of U, therefore the filter f_k of (6.3) is well-defined. Equation (6.3) is obtained by combining the $\prod_{i=1}^{n} m_i$ rules of Step 2 using product inference and centroid defuzzification (see Section 1.2). Another way of interpreting (6.3) is as follows. For a given input $\underline{x} \in U$, we determine the filter output $f_k(\underline{x})$ as a weighted sum of the $\prod_{i=1}^{n} m_i$ points $\theta^{(j1,\ldots,jn)}$ in the output space at which the fuzzy sets $G^{(j1,\ldots,jn)}$ of the THEN parts of the $\prod_{i=1}^{n} m_i$ rules have maximum membership values; and, the weight $\mu_{F_1^{j1}}(x_1)\cdots\mu_{F_n^{jn}}(x_n)$ for $\theta^{(j1,\ldots,jn)}$ is proportional to the membership values for which $\underline{x}$ satisfies the IF part of $R^{(j1,\ldots,jn)}$. This is a reasonable filter because $\theta^{(j1,\ldots,jn)}$'s are the "most likely" points in the output space based on the $\prod_{i=1}^{n} m_i$ rules, and the point $\theta^{(j1,\ldots,jn)}$ should be given more weight if the given input point $\underline{x}$ satisfies the corresponding IF part "more likely" (in the sense of larger membership value).

In (6.3), the weights $\mu_{F_1^{j1}}(x_1)\cdots\mu_{F_n^{jn}}(x_n)$ are fixed functions of $\underline{x}$; therefore, the free design parameters of the fuzzy adaptive filter are the $\theta^{(j1,\ldots,jn)}$'s which are now collected as a $\prod_{i=1}^{n} m_i$-dimensional vector

$$\begin{aligned}\underline{\theta} \equiv\ & (\theta^{(1,1,\ldots,1)},\ldots,\theta^{(m_1,1,\ldots,1)};\theta^{(1,2,1,\ldots,1)},\ldots,\theta^{(m_1,2,1,\ldots,1)};\ldots;\\ & \theta^{(1,m_2,1,\ldots,1)},\ldots,\theta^{(m_1,m_2,1,\ldots,1)};\ldots;\theta^{(1,m_2,\ldots,m_n)},\ldots,\theta^{(m_1,m_2,\ldots,m_n)})^T. \quad (6.4)\end{aligned}$$

Define the *fuzzy basis functions* (chapter 4)

$$p^{(j1,\ldots,jn)}(\underline{x}) = \frac{\mu_{F_1^{j1}}(x_1)\cdots\mu_{F_n^{jn}}(x_n)}{\sum_{j1=1}^{m_1}\cdots\sum_{jn=1}^{m_n}(\mu_{F_1^{j1}}(x_1)\cdots\mu_{F_n^{jn}}(x_n))}, \quad (6.5)$$

and collect them as a $\prod_{i=1}^{n} m_i$-dimensional vector $\underline{p}(\underline{x})$ in the same ordering as the $\underline{\theta}$ of (6.4)

$$\begin{aligned}\underline{p}(\underline{x}) \equiv\ & (p^{(1,1,\ldots,1)}(\underline{x}),\ldots,p^{(m_1,1,\ldots,1)}(\underline{x});p^{(1,2,1,\ldots,1)}(\underline{x}),\ldots,p^{(m_1,2,1,\ldots,1)}(\underline{x});\ldots;p^{(1,m_2,1,\ldots,1)}(\underline{x}),\\ & \ldots,p^{(m_1,m_2,1,\ldots,1)}(\underline{x});\ldots;p^{(1,m_2,\ldots,m_n)}(\underline{x}),\ldots,p^{(m_1,m_2,\ldots,m_n)}(\underline{x}))^T. \quad (6.6)\end{aligned}$$

Based on (6.4) and (6.6) we can now rewrite (6.3) as

$$f_k(\underline{x}) = \underline{p}^T(\underline{x})\underline{\theta}. \quad (6.7)$$

We see from (6.7) that f_k is linear in the parameter; therefore, we can use the fast-convergent RLS algorithm to update the parameters $\underline{\theta}$.

Step 4: Use the following RLS algorithm [11] to update $\underline{\theta}$: let the initial estimate of $\underline{\theta}$, $\underline{\theta}(0)$, be determined as in Step 2, and $P(0) = \sigma I$, where σ is a small positive constant, and I is the $\prod_{i=1}^{n} m_i - by - \prod_{i=1}^{n} m_i$ identity matrix; at each time point $k = 1, 2, ...$, do the following:

$$\underline{\phi}(k) = \underline{p}(\underline{x}(k)), \tag{6.8}$$

$$P(k) = \frac{1}{\lambda}[P(k-1) - P(k-1)\underline{\phi}(k)(\lambda + \underline{\phi}^T(k)P(k-1)\underline{\phi}(k))^{-1}\underline{\phi}^T(k)P(k-1)], \tag{6.9}$$

$$K(k) = P(k-1)\underline{\phi}(k)[\lambda + \underline{\phi}^T(k)P(k-1)\underline{\phi}(k)]^{-1}, \tag{6.10}$$

$$\underline{\theta}(k) = \underline{\theta}(k-1) + K(k)(d(k) - \underline{\phi}^T(k)\underline{\theta}(k-1)), \tag{6.11}$$

where $[\underline{x}(k)]$ and $[d(k)]$ are the sequences defined above in Problem 1, $\underline{p}(*)$ is defined in (6.6), and λ is the forgetting factor in (6.1).

Some comments on this RLS fuzzy adaptive filter are now in order.

Remark 6.1: The RLS algorithm (6.9)-(6.11) is obtained by minimizing $J(k)$ of (6.1) with f_k constrained to be the form of (6.7). Because f_k of (6.7) is linear in the parameter, the derivation of (6.9)-(6.11) is the same as that of the FIR linear adaptive filter [11]; therefore, we omit the details.

Remark 6.2: The RLS algorithm (6.9)-(6.11) can be viewed as updating the $\prod_{i=1}^{n} m_i$ rules in the form of (6.2) by changing the "centers" $\theta^{(j1,...,jn)}$ of the THEN parts of these rules in the direction of minimizing the criterion function (6.1). We are allowed only to change these "centers." The membership functions $\mu_{F_i^{ji}}$ of the IF parts of the rules are fixed at the very beginning and are not allowed to change; therefore, a good choice of $\mu_{F_i^{ji}}$'s is important to the success of the entire filter. In the next section, we will allow the $\mu_{F_i^{ji}}$'s also to change during the adaptation procedure.

Remark 6.3: It was proven in Chapter 2 that functions in the form of (6.3) are universal approximators, i.e., for any real continuous function g on the compact set U, there exists a function in the form of (6.3) such that it can uniformly approximate g over U to arbitrary accuracy. Consequently, our fuzzy adaptive

filter is a powerful nonlinear adaptive filter in the sense that it has the capability of performing difficult nonlinear filtering operations.

Remark 6.4: The fuzzy adaptive filter (6.7) performs a two stage operation on the input vector $\underline{x}$: first, it performs a nonlinear transformation $\underline{p}(*)$ on $\underline{x}$; then, the filter output is obtained as a linear combination of these transformed signals. In this sense, our fuzzy adaptive filter is similar to the radial basis function [8,58] and potential function [44] approaches. The unique feature of our fuzzy adaptive filter, which is not shared by other nonlinear adaptive filters, is that linguistic rules can be incorporated into the filter, as discussed next.

Remark 6.5: Linguistic information (in the form of the fuzzy IF-THEN rules of (2)) and numerical information (in the form of desired input-output pairs $(\underline{x}(k), d(k))$) are combined into the filter in the following way: due to Steps 2-4, linguistic IF-THEN rules are directly incorporated into the filter (6.3) by constructing the initial filter based on the linguistic rules; and, due to the adaptation of Step 4, numerical pairs $(\underline{x}(k), d(k))$ are incorporated into the filter by updating the filter parameters such that the filter output "matches" the pairs in the sense of minimizing (6.1). It is natural and reasonable to assume that linguistic information from human experts is provided in the form of (6.2) because the rules of (6.2) state what the filter outputs should be in some input situations, where "what should be" and "some situations" are represented by linguistic terms which are characterized by fuzzy membership functions. On the other hand, it is obvious that the most natural form of numerical information is provided in the form of input-output pairs $(\underline{x}(k), d(k))$.

Remark 6.6: By fixing the fuzzy membership functions on the input space U at the very begining, we obtained a nonlinear filter which is linear in the parameter; therefore, we could use the fast-convergent RLS algorithm in the adaptation procedure. The price paid is that we had to include all the $\prod_{i=1}^{n} m_i$ possible rules in the filter, because if a region of U is not covered by any rules and an input $\underline{x}$ to the filter happens to be in this region, then the filter response will be very poor. As a result, for problems of high dimension n and large m_i the computations involved in this fuzzy adaptive filter are intense, because at each time point k we need to perform the $\prod_{i=1}^{n} m_i$-dimensional matrix-to-vector multiplications of (6.9)-(6.11)

and to evaluate the values of the $\prod_{i=1}^{n} m_i$ fuzzy basis functions of (6.8) (see also (6.6) and (6.5)). Although these computations are highly parallelizable, we may not be able to use the filter in some practical situations where computing power is limited; therefore, we will develop another fuzzy adaptive filter which involves much less computations, next.

6.3 LMS Fuzzy Adaptive Filter

Our LMS fuzzy adaptive filter solves the following problem.

Problem 2: Consider the same input sequence $[\underline{x}(k)]$ and output sequence $[d(k)]$ as in Problem 1. The problem is: at each time point $k = 1, 2, ...$, determine an adaptive filter $f_k : U \rightarrow R$ such that

$$L = E[(d(k) - f_k(\underline{x}(k)))^2] \tag{6.12}$$

is minimized.

Design Procedure of the LMS Fuzzy Adaptive Filter:

Step 1: Define M fuzzy sets F_i^l in each interval $[C_i^-, C_i^+]$ of U with the following *Gaussian* membership functions

$$\mu_{F_i^l}(x_i) = exp[-\frac{1}{2}(\frac{x_i - \bar{x}_i^l}{\sigma_i^l})^2], \tag{6.13}$$

where $l = 1, 2, ..., M$, $i = 1, 2, ..., n$, $x_i \in [C_i^-, C_i^+]$, and $\bar{x}_i^l$ and σ_i^l are free parameters which will be updated in the LMS adaptation procedure of Step 4.

Step 2: Construct a set of M (in general, $M << \prod_{i=1}^{n} m_i$) fuzzy IF-THEN rules in the following form:

$$R^l: \ IF \ x_1 \ is \ F_1^l \ and \ \cdots \ and \ x_n \ is \ F_n^l, \ THEN \ d \ is \ G^l, \tag{6.14}$$

where $\underline{x} = (x_1, ..., x_n)^T \in U$, $d \in R$, F_i^l's are defined in Step 1, and G^l's are fuzzy sets defined in R which are determined as follows: *if there are linguistic rules in the form of (6.14), set F_i^l's and G^l to be the labels of these linguistic rules; otherwise, choose μ_{G^l} and the parameters $\bar{x}_i^l$ and σ_i^l arbitrarily.* The (parameters

of) membership functions $\mu_{F_i^l}$ and μ_{G^l} in these rules will change during the LMS adaptation procedure of Step 4; therefore, the rules constructed in this step are initial rules of the fuzzy adaptive filter. As in the RLS fuzzy adaptive filter, we incorporate linguistic rules into the LMS fuzzy adaptive filter by constructing the initial filter based on these rules.

Step 3: Construct the filter $f_k : U \rightarrow R$ based on the M rules of Step 2 as follows:

$$f_k(\underline{x}) = \frac{\sum_{l=1}^{M} \theta^l (\prod_{i=1}^{n} \mu_{F_i^l}(x_i))}{\sum_{l=1}^{M} (\prod_{i=1}^{n} \mu_{F_i^l}(x_i))}, \tag{6.15}$$

where $\underline{x} = (x_1, ..., x_n)^T \in U$, $\mu_{F_i^l}$'s are the Gaussian membership functions of (6.13), and $\theta^l \in R$ is any point at which μ_{G^l} achieves its maximum value. The filter (6.15) is constructed in the same way as (6.3), and shares the same interpretation. Because we chose the membership functions $\mu_{F_i^l}(x_i)$ to be Gaussian functions which are nonzero for any $x_i \in [C_i^-, C_i^+]$, the denominator of (6.15) is nonzero for any $\underline{x} \in U$; therefore, the filter f_k of (6.15) is well-defined. Because the θ^l as well as $\bar{x}_i^l$ and σ_i^l are free parameters, the filter (6.15) is nonlinear in the parameters.

Step 4: Use the following LMS algorithm [96] to update the filter parameters $\theta^l, \bar{x}_i^l$ and σ_i^l: let the initial $\theta^l(0), \bar{x}_i^l(0)$ and $\sigma_i^l(0)$ be as determined in Step 2; at each time point $k = 1, 2, ...$, do the following:

$$\theta^l(k) = \theta^l(k-1) + \alpha[d(k) - f_k]\frac{a^l(k-1)}{b(k-1)}, \tag{6.16}$$

$$\bar{x}_i^l(k) = \bar{x}_i^l(k-1) + \alpha[d(k) - f_k]\frac{\theta^l(k-1) - f_k}{b(k-1)} a^l(k-1)\frac{x_i(k) - \bar{x}_i^l(k-1)}{(\sigma_i^l(k-1))^2}, \tag{6.17}$$

$$\sigma_i^l(k) = \sigma_i^l(k-1) + \alpha[d(k) - f_k]\frac{\theta^l(k-1) - f_k}{b(k-1)} a^l(k-1)\frac{(x_i(k) - \bar{x}_i^l(k-1))^2}{(\sigma_i^l(k-1))^3}, \tag{6.18}$$

where $a^l(k-1) = \prod_{i=1}^{n} exp[-\frac{1}{2}(\frac{x_i(k) - \bar{x}_i^l(k-1)}{\sigma_i^l(k-1)})^2]$, $b(k-1) = \sum_{l=1}^{M} a^l(k-1)$, $f_k = \frac{\sum_{l=1}^{M} \theta^l a^l(k-1)}{b(k-1)}$, α is a small positive stepsize, $l = 1, 2, ..., M$, and $i = 1, 2, ..., n$. Equations (6.16)-(6.18) are obtained by taking the gradient of L (6.12) (ignore the expectation E) with respect to the parameters and using the specific formula of (6.15) and (6.13).

Some comments on this LMS fuzzy adaptive filter are now in order.

Remark 6.7: From Steps 2-4 we see that the initial LMS fuzzy adaptive filter is constructed based on linguistic rules from human experts and some arbitrary rules (in the sense that the parameters of membership functions $\mu_{F_i^l}$ and μ_{G^l} which characterize these rules are chosen arbitrarily). Both sets of rules are updated during the LMS adaptation procedure of Step 4 by changing the parameters in the direction of minimizing the L of (6.12). Because minimizing (6.12) can be viewed as matching the input-output pairs $[\underline{x}(k); d(k)]$, our LMS fuzzy adaptive filter combines both linguistic and numerical information in its design.

Remark 6.8: Because the LMS algorithm is a gradient algorithm, a good choice of initial parameters is very important to its convergence. Because we use linguistic information to choose the initial parameters, the adaptation procedure should converge quickly if the linguistic rules provide good instructions for how the filter should perform, i.e, good descriptions of the input-ouput pairs $[\underline{x}(k); d(k)]$. Therefore, although LMS algorithms in general are slow to converge, our LMS algorithm in particular may converge fast, provided that there are sufficient linguistic rules.

Remark 6.9: The filter f_k of (6.15) can match any input-output pair $[\underline{x}(k); d(k)]$ to arbitrary accuracy by properly choosing the parameters θ^l, $\bar{x}_i^l$ and σ_i^l, as we show next. For given $[\underline{x}(k); d(k)]$, let $\bar{x}_i^1 = x_i(k)$, $\bar{x}_i^l \neq x_i(k)$ for $l \neq 1$, and $\theta^1 = d(k)$; therefore, for $\underline{x} = \underline{x}(k)$ in (6.15), the weight $(\prod_{i=1}^n \mu_{F_i^1}(x_i(k)))$ for $\theta^1 = d(k)$ equals one for any choice of σ_i^1, and the other $M-1$ weights $(\prod_{i=1}^n \mu_{F_i^l}(x_i(k)))$ for θ^l with $l \neq 1$ can be arbitrarily close to zero if we choose all σ_i^l to be sufficiently small (see (6.13) and notice that $x_i(k) \neq \bar{x}_i^l$ for $l \neq 1$). As a result, $|d(k) - f_k(\underline{x}(k))|$ can be arbitrarily small. Because of this property and the freedom to update the parameters during the adaptation procedure, we can hope that we have a well-performing filter using only a small number of rules (i.e., we may choose $M << \prod_{i=1}^n m_i$).

6.4 Application to Nonlinear Channel Equalization

Nonlinear distortion over a communication channel is now a significant factor hindering further increase in the attainable data rate in high-speed data transmission

[3,14]. Because the received signal over a nonlinear channel is a nonlinear function of the past values of the transmitted symbols, and the nonlinear distortion varies with time and from place to place, effective equalizers for nonlinear channels should be nonlinear and adaptive.

In [3,14], ploynomial adaptive filters were developed for nonlinear channel equalization. In [7,8], multi-layer perceptrons and radial basis function expansions were used as adaptive equalizers for nonlinear channels. Because nonlinear channels include a very broad spectrum of nonlinear distortion, it is very difficult to say which nonlinear adaptive filter is dominantly better than the others. Therefore, it is worth trying other new nonlinear structures as pototypes of nonlinear adaptive filters in addition to the existing Volterra series, multi-layer perceptron, radial basis function expansions, etc.. The RLS and LMS adaptive filters are such new nonlinear adaptive filters. In this section, we use them as equalizers for nonlinear channels.

The digital communication system considered in this paper is shown in Fig. 6.1, where the "channel" includes the effects of the transmitter filter, the transmission medium, the receiver matched filter, and other components. The transmitted data sequence $s(k)$ is assumed to be an independent sequence taking values from $\{-1,1\}$ with equal probability. The inputs to the equalizer, $x(k), x(k-1)$, $\cdots, x(k-n+1)$, are the channel outputs corrupted by an additive noise $e(k)$. The task of the equalizer at the sampling instant k is to produce an estimate of the transmitted symbol $s(k-d)$ using the information contained in $x(k), x(k-1)$, $\cdots, x(k-n+1)$, where the integers n and d are known as the order and the lag of the equalizer, respectively.

We use the geometric formulation of the equalization problem due to [7,8]. Using similar notation to that in [7,8], define

$$P_{n,d}(1) = \{\underline{\hat{x}}(k) \in R^n | s(k-d) = 1\}, \tag{6.19}$$

$$P_{n,d}(-1) = \{\underline{\hat{x}}(k) \in R^n | s(k-d) = -1\}, \tag{6.20}$$

where

$$\underline{\hat{x}}(k) = [\hat{x}(k), \hat{x}(k-1), \cdots, \hat{x}(k-n+1)]^T, \tag{6.21}$$

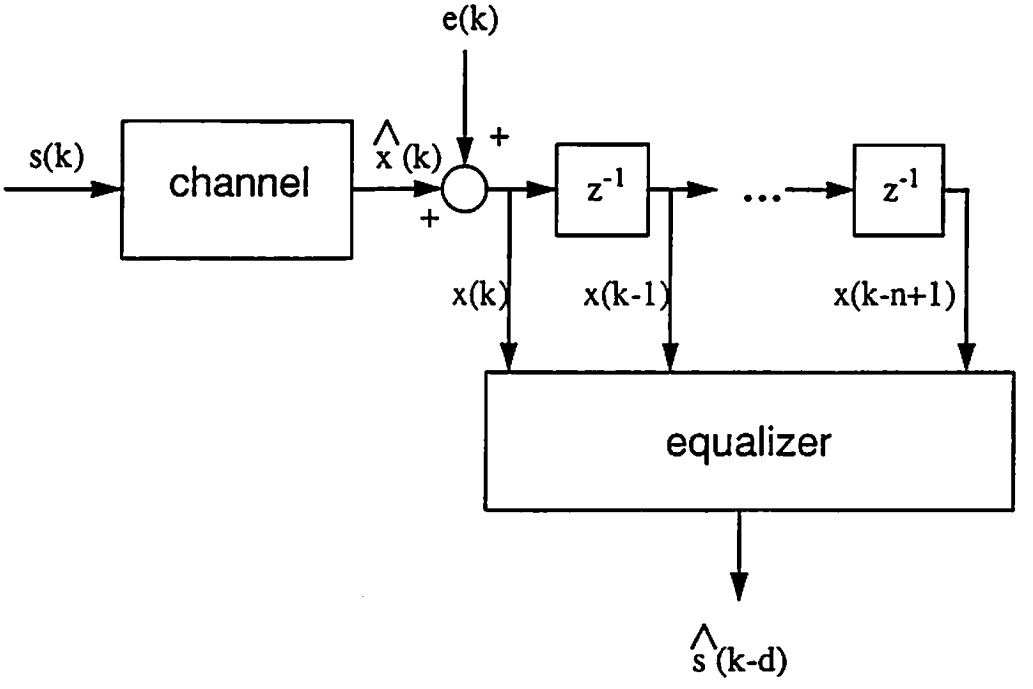


Figure 6.1: Schematic of data transmission system.

$\hat{x}(k)$ is the noise-free output of the channel (see Fig. 1), and $P_{n,d}(1)$ and $P_{n,d}(-1)$ represent the two sets of possible channel noise-free output vectors $\underline{\hat{x}}(k)$ that can be produced from sequences of the channel inputs containing $s(k-d)=1$ and $s(k-d)=-1$, respectively. The equalizer can be characterized by the function

$$g_k : R^n \rightarrow \{-1, 1\} \tag{6.22}$$

with

$$\hat{s}(k-d) = g_k(\underline{x}(k)), \tag{6.23}$$

where

$$\underline{x}(k) = [x(k), x(k-1), \cdots, x(k-n+1)]^T \tag{6.24}$$

is the observed channel output vector. Let $p_1[\underline{x}(k)|\underline{\hat{x}}(k) \in P_{n,d}(1)]$ and $p_{-1}[\underline{x}(k)|\underline{\hat{x}}(k) \in P_{n,d}(-1)]$ be the conditional probability density functions of $\underline{x}(k)$ given $\underline{\hat{x}}(k) \in P_{n,d}(1)$ and $\underline{\hat{x}}(k) \in P_{n,d}(-1)$, respectively. It was shown in [2,3] that the equalizer which is defined by

$$f_{opt}(\underline{x}(k)) = sgn[p_1(\underline{x}(k)|\underline{\hat{x}}(k) \in P_{n,d}(1)) - p_{-1}(\underline{x}(k)|\underline{\hat{x}}(k) \in P_{n,d}(-1))] \tag{6.25}$$

achieves the minimum bit error rate for the given order n and lag d, where $sgn(y) = 1(-1)$ if $y \geq 0$ ($y < 0$). If the noise $e(k)$ is zero-mean and Gaussian with covariance matrix

$$Q = E[(e(k), ..., e(k-n+1))(e(k), ..., e(k-n+1))^T], \tag{6.26}$$

then from $x(k) = \hat{x}(k) + e(k)$ we have that

$$\begin{aligned} & p_1[\underline{x}(k)|\underline{\hat{x}}(k) \in P_{n,d}(1)] - p_{-1}[\underline{x}(k)|\underline{\hat{x}}(k) \in P_{n,d}(-1)] \\ = & \sum exp[-\frac{1}{2}(\underline{x}(k) - \underline{\hat{x}}_+)^T Q^{-1}(\underline{x}(k) - \underline{\hat{x}}_+)] \\ & - \sum exp[-\frac{1}{2}(\underline{x}(k) - \underline{\hat{x}}_-)^T Q^{-1}(\underline{x}(k) - \underline{\hat{x}}_-)], \end{aligned} \tag{6.27}$$

where the first (second) sum is over all the points $\underline{\hat{x}}_+ \in P_{n,d}(1)$ ($\underline{\hat{x}}_- \in P_{n,d}(-1)$).

Now consider the nonlinear channel

$$\hat{x}(k) = s(k) + 0.5s(k-1) - 0.9[s(k) + 0.5s(k-1)]^3, \tag{6.28}$$

and white Gaussian noise $e(k)$ with $E[e^2(k)] = 0.2$. For this case, the optimal decision region for $n = 2$ and $d = 0$,

$$[\underline{x}(k) \in R^2 | p_1[\underline{x}(k)|\hat{\underline{x}}(k) \in P_{2,0}(1)] - p_{-1}[\underline{x}(k)|\hat{\underline{x}}(k) \in P_{2,0}(-1)] \geq 0], \tag{6.29}$$

is shown in Fig. 6.2 as the shaded area. The elements of the sets $P_{2,0}(1)$ and $P_{2,0}(-1)$ are illustrated in Fig. 6.2 by the "o" and "*", respectively. From Fig. 6.2 we see that the optimal decision boundary for this case is severely nonlinear. We now use the RLS and LMS fuzzy adaptive filters to solve this specific equalization problem (channel (6.28), $e(k)$ white Gaussian with variance 0.2, equalizer order $n = 2$ and lag $d = 0$) under various conditions (Examples 6.1-6.4).

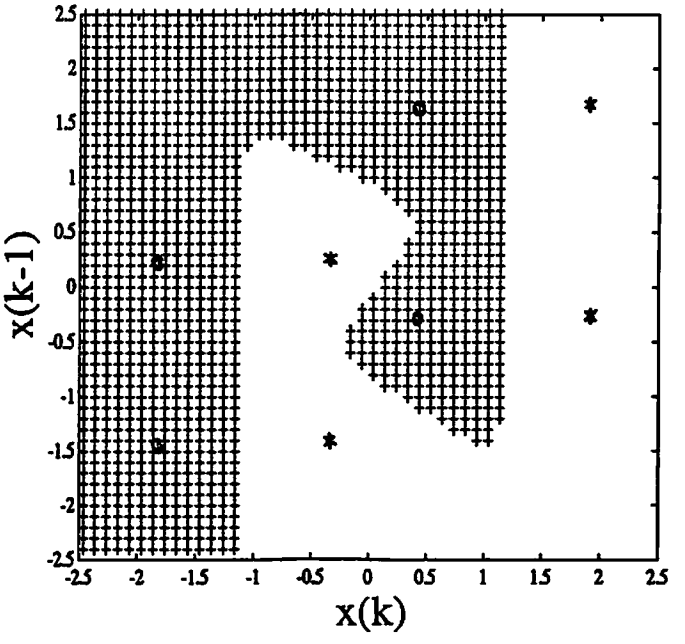


Figure 6.2: Optimal decision region for the channel (6.28), Gaussian white noise with variance $\sigma_e^2 = 0.2$, and equalizer order $n = 2$ and lag $d = 0$.

Example 6.1: Here, we used the RLS fuzzy adaptive filter without any linguistic information. We chose $\lambda = 0.999, \sigma = 0.1$, $m_1 = m_2 = 9$, and $\mu_{F_i^j}(x_i) = exp[-\frac{1}{2}(\frac{x_i - \bar{x}_i^j}{0.3})^2]$ with $\bar{x}_i^j = -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2$ for $j = 1, 2, ..., 9$, respectively, where $i = 1, 2$, $x_1 = x(k)$ and $x_2 = x(k-1)$. For the same realization of the sequence $s(k)$ and the same randomly chosen initial parameters $\underline{\theta}(0)$ (within [-0.3, 0.3]), we simulated three cases: the adaptation (6.9)-(6.11) stopped at (i) $k = 30$, (ii) $k = 50$, and (iii) $k = 100$. The final decision regions, $[\underline{x}(k) \in R^2 | f_k(\underline{x}(k)) \geq 0]$, for the three cases are shown in Figs. 6.3-6.5, respectively. From Figs. 6.3-6.5 we see that the decision regions obtained from the RLS fuzzy adaptive filter tended to converge towards the optimal decision region.

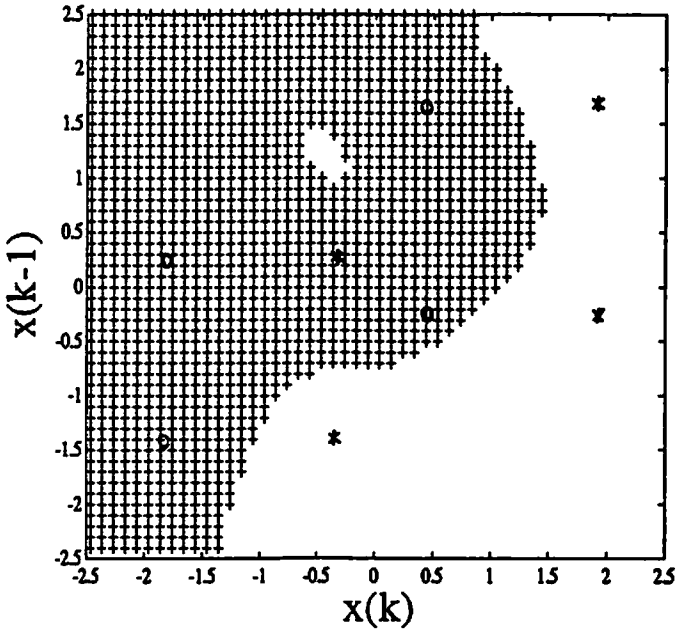


Figure 6.3: Decision region of the RLS fuzzy adaptive filter without using any linguistic information and when the adaptation stopped at $k = 30$.

Example 6.2: Next, we used the RLS fuzzy adaptive filter incorporating the following linguistic information about the decision region. From the geometric formulation we see that the equalization problem is equivalent to determining a decision boundary in the input space of the equalizer. Suppose that there are human experts who are very familiar with the specific situation, such that

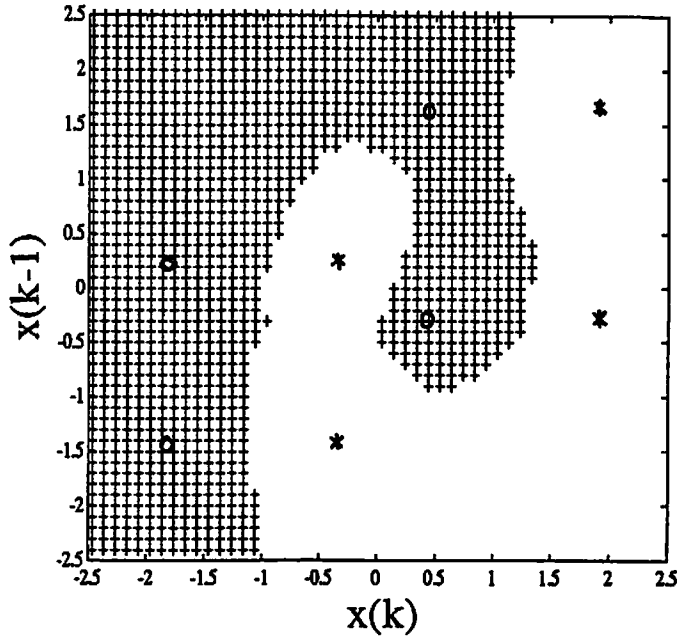


Figure 6.4: Decision region of the RLS fuzzy adaptive filter without using any linguistic information and when the adaptation stopped at $k = 50$.

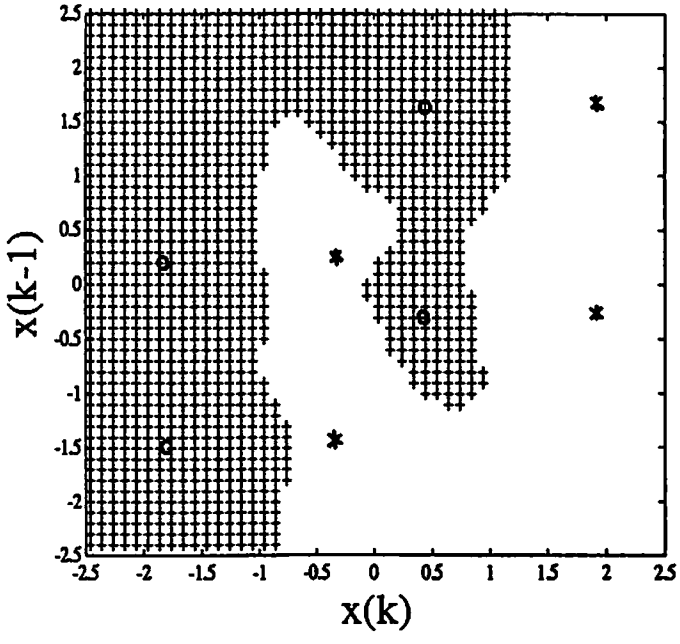


Figure 6.5: Decision region of the RLS fuzzy adaptive filter without using any linguistic information and when the adaptation stopped at $k = 100$.

although they cannot draw the specific decision boundary in the input space of the equalizer, they can assign degrees to different regions in the input space which reflect their belief that the regions should belong to 1-catalog or -1-catalog. Take Fig. 6.2 as an example. We see from Fig. 6.2 that the difficult is to determine which catalog the middle portion should belong to; in other words, as we move away from the middle portion, we have less and less uncertainty about which catalog the region should belong to. For example, for the left-most region in Fig. 6.2, we have more confidence that it should belong to the 1-catalog rather than the -1-catalog. Similarly, for the right-most region in Fig. 6.2, we have more confidence that it should belong to the -1-catalog rather than the 1-catalog. Also, we assume that the human experts know that a portion of the boundary is somewhere around $x(k) = -1.2$ for $x(k-1)$ less than 1 and around $x(k) = 1.2$ for $x(k-1)$ greater than -1. To make these observations specific, we have the fuzzy rules shown in Fig. 6.6 where the membership functions N3, N2, etc. are the $\mu_{F_i^j}$'s defined in Example 6.1. We have 48 rules in Fig. 6.6, corresponding to the boxes with numbers; for example, the bottom-left box corresponds to the rule: "IF $x(k)$ is N4 and $x(k-1)$ is N4, THEN f_k is G," where f_k is the filter output, and the center of μ_G is 0.6. Because the filter output f_k is a weighted sum of these centers (see (6.3)), the numbers 0.6, 0.4, -0.4, -0.6 in Fig. 6.6 reflect our belief that the regions should correspond to the 1-catalog or the -1-catalog. For example, if the input point $[x(k), x(k-1)]$ falls in the left-most region of Fig. 6.6, then we have more confidence that the transmitted $s(k)$ should be 1 rather than -1, and, we represent this confidence by assigning the center of the fuzzy term in the corresponding THEN part to be 0.6.

It should be emphasized that the rules in Fig. 6.6 provide very fuzzy information about the decision region, because: 1) the regions are fuzzy, i.e., there are no clear boundaries between the regions, and 2) the numbers 0.6, 0.4, -0.4, -0.6 are conservative, i.e., they are away from the real transmitted values 1 or -1. We now show that although these rules are fuzzy, the adaptation speed is greatly improved by incorporating them into the RLS fuzzy adaptive equalizer (filter). Figure 6.7 shows the final decision region determined by the RLS fuzzy adaptive filter, $[\underline{x}(k) \in R^2 | f_k(\underline{x}(k)) \geq 0]$ (shaded area), when the adaptation stopped at

$k = 30$ after the rules in Fig. 6.6 were incorporated, where the $\mu_{F_i^j}$'s and the sequence $s(k)$ were the same as those in Example 6.1. Comparing Figs. 6.7 and 6.3 we see that the adaptation speed was greatly improved by incorporating these fuzzy rules.

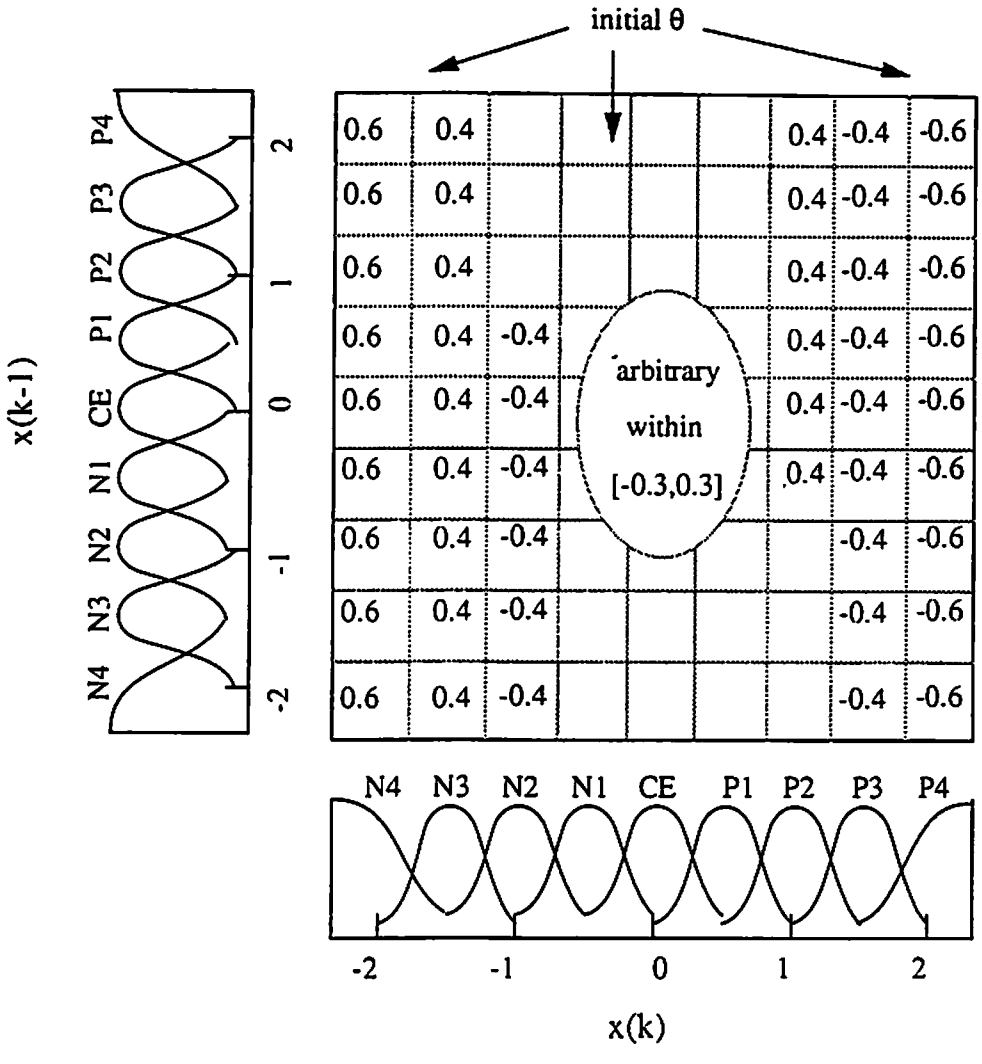


Figure 6.6: Illustration of some fuzzy rules about the decision region.

Example 6.3: Here we used the LMS fuzzy adaptive filter without any linguistic information. We chose: $M = 20, \alpha = 0.05$, the initial $\theta^l(0)$ randomly in $[-0.3, 0.3]$, $\bar{x}_i^j(0)$'s randomly in $[-2, 2]$, and $\bar{\sigma}_i^j(0)$'s randomly in $[0.1, 0.3]$. For

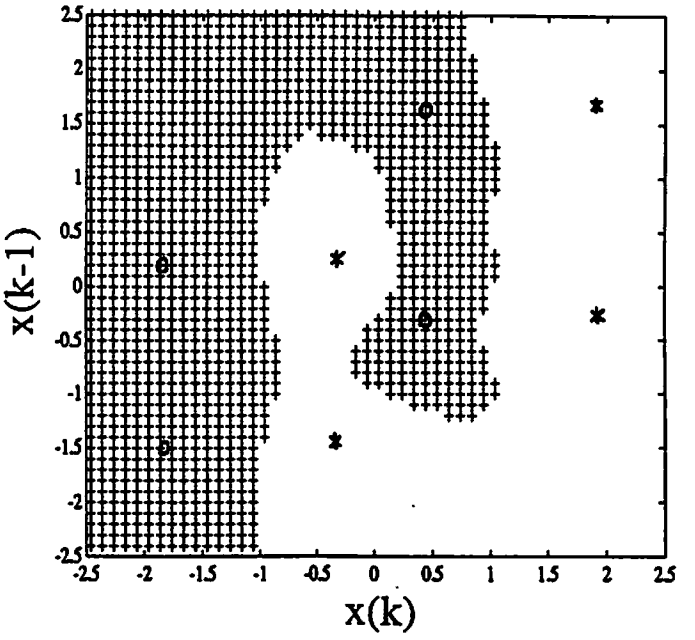


Figure 6.7: Decision region of the RLS fuzzy adaptive filter after incorporating the fuzzy rules illustrated in Fig. 6.6 and when the adaptation stopped at $k = 30$.

the same sequence $s(k)$ (in Example 6.1) and the same initial parameters, we simulated the cases when the adaptation algorithm (6.16)-(6.18) stopped at: (i) $k = 100$, (ii) $k = 200$, and (iii) $k = 500$. The decision regions for the three cases are shown in Figs. 6.8-6.10, respectively.

Example 6.4: Next, we used the LMS fuzzy adaptive filter and incorporated some of the fuzzy rules in Fig. 6.6. We still chose $M = 20$ and $\alpha = 0.05$ and used the same $s(k)$ sequence. Since the filter is constructed from 20 rules, whereas Fig. 6.6 contains 48 rules, we can only choose a portion of the rules in Fig. 6.6 to construct the initial LMS fuzzy adaptive filter. We chose 20 rules arbitrarily from the boxes labeled 0.4 and -0.4. The final decision region for this case, when the adaptation stopped at $k = 100$, is shown in Fig. 6.11. Comparing Figs. 6.11 and 6.8 we see that the adaptation speed was improved by incorporating these fuzzy rules.

Example 6.5: Here, we considered the same situation as in Example 6.1, except that we chose $d = 1$ rather than $d = 0$. The optimal decision region for this case is shown in Fig. 6.12. Figures 6.13 and 6.14 show the final decision regions

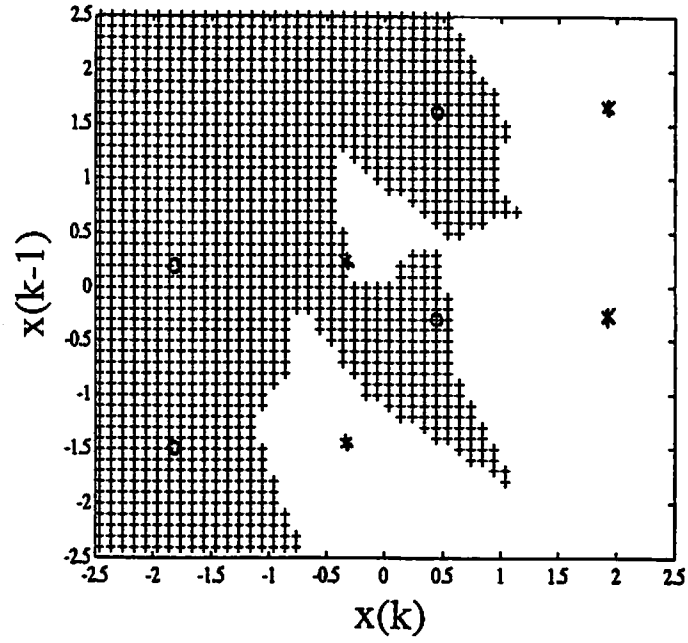


Figure 6.8: Decision region of the LMS fuzzy adaptive filter without using any linguistic information and when the adaptation stopped at $k = 100$.

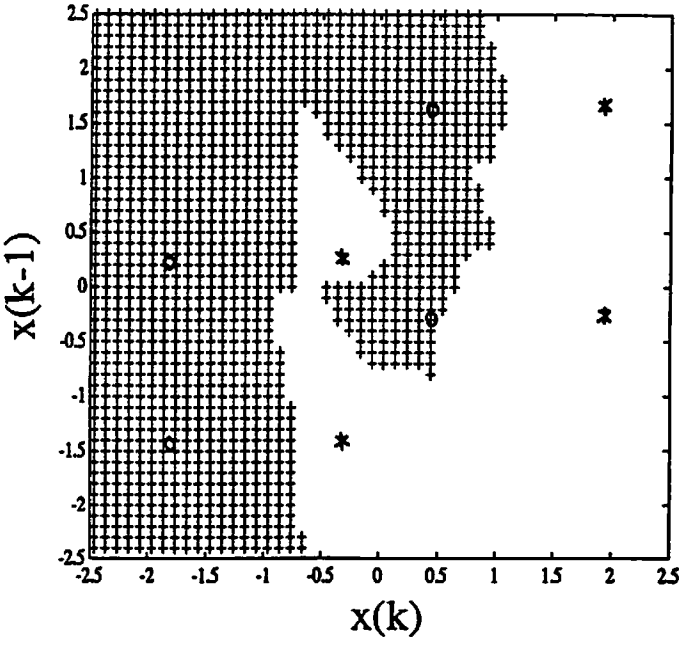


Figure 6.9: Decision region of the LMS fuzzy adaptive filter without using any linguistic information and when the adaptation stopped at $k = 200$.

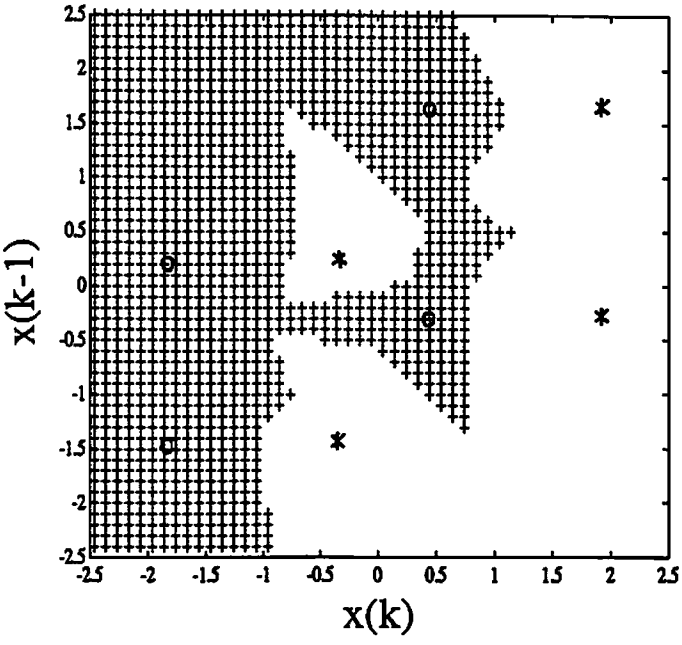


Figure 6.10: Decision region of the LMS fuzzy adaptive filter without using any linguistic information and when the adaptation stopped at $k = 500$.

determined by the RLS fuzzy adaptive filter when the adaptation stoped at $k = 20$ and $k = 50$, respectively. We also simulated the LMS fuzzy adaptive filter for this case, and the final decision region was similar to Fig. 6.14 when the adaptation stoped at $k = 300$. Since we showed this kind of comparisons in Examples 6.1-6.4, we omit the details.

Example 6.6: In this final example, we compare the bit error rates achieved by the optimal equalizer (6.25) and the fuzzy adaptive equalizers for different signal-to-noise ratios, for the channel (6.28) with equalizer order $n = 2$ and lag $d = 1$. The optimal bit error rate was computed by applying the optimal equalizer (6.25) to a realization of 10^6 points of the sequences $s(k)$ and $e(k)$. For the RLS fuzzy adaptive filter, we chose the filter parameters to be the same as in Example 6.1. For the LMS fuzzy adaptive filter, the parameters were chosen as in Example 6.3. We ran the RLS and LMS fuzzy adaptive filters for the first 1000 points in the same 10^6 point realization of $s(k)$ and $e(k)$ as for the optimal equalizer, and then used the trained fuzzy equalizers to compute the bit error rate for the same 10^6 point realization. Figure 6.15 shows the bit error rates of the optimal equalizer

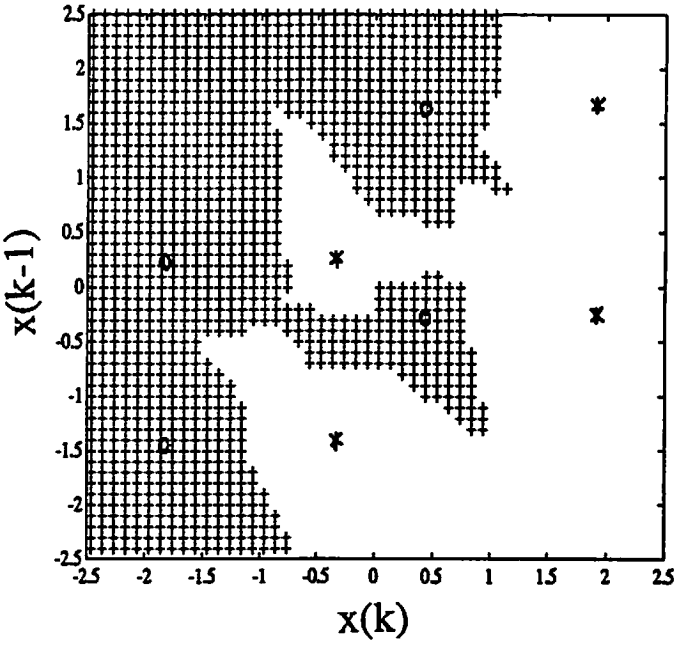


Figure 6.11: Decision region of the LMS fuzzy adaptive filter after incorporating some of the fuzzy rules illustrated in Fig. 6.6 and when the adaptation stopped at $k = 100$.

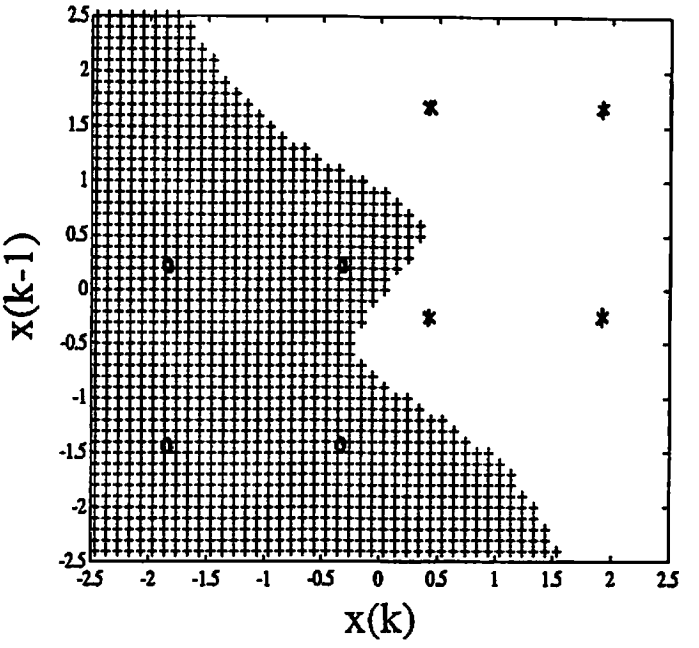


Figure 6.12: Optimal decision region for the channel (6.28), Gaussian white noise with variance $\sigma_e^2 = 0.2$, and equalizer order $n = 2$ and lag $d = 1$.

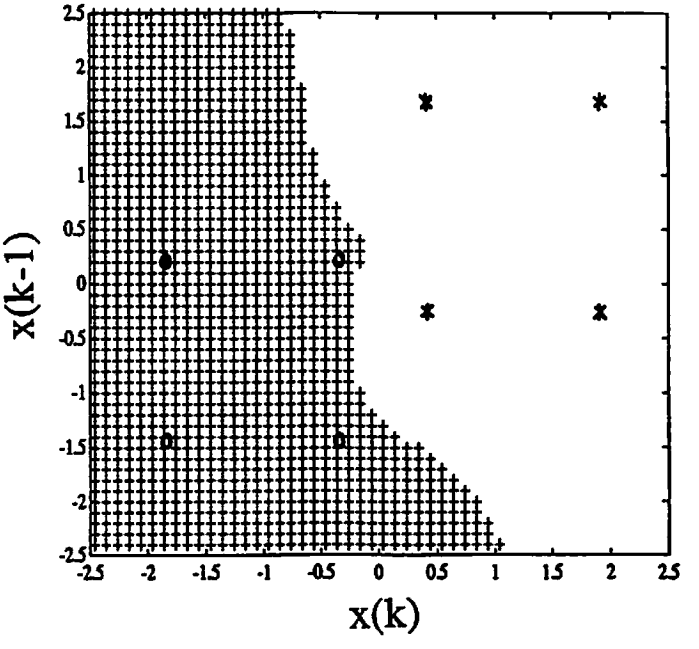


Figure 6.13: Decision region of the RLS fuzzy adaptive filter for the case of Example 6.5 when the adaptation stopped at $k = 20$.

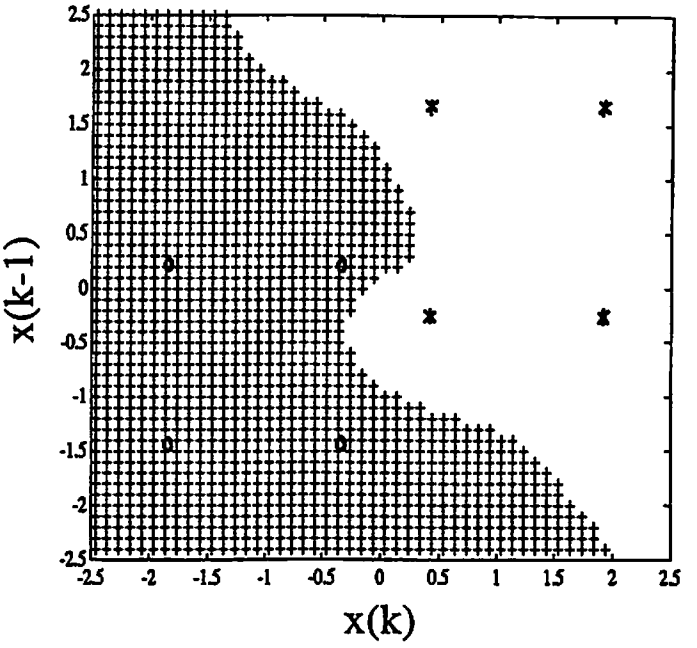


Figure 6.14: Decision region of the RLS fuzzy adaptive filter for the case of Example 6.5 when the adaptation stopped at $k = 50$.

and the two fuzzy equalizers for different signal-to-noise ratios, where the bit error rate curves for the RLS and LMS fuzzy equalizers are indistinguishable. We see from Fig. 6.15 that the bit error rates of the fuzzy equalizers are very close to the optimal one.

6.5 Conclusions

In this chapter, we developed two new nonlinear adaptive filters, namely: RLS and LMS fuzzy adaptive filters. The key elements of the fuzzy adaptive filters are a fuzzy system, which is constructed from a set of fuzzy IF-THEN rules, and an adaptive algorithm for updating the parameters in the fuzzy system, which, for the RLS fuzzy adaptive filter, is an RLS type of algorithm, and for the LMS fuzzy adaptive filter, is an LMS type of algorithm. The most important advantage of the fuzzy adaptive filters is that linguistic information from human experts (in the form of fuzzy IF-THEN rules) can be directly incorporated into the filters. If no linguistic information is available, the fuzzy adaptive filters become well-defined

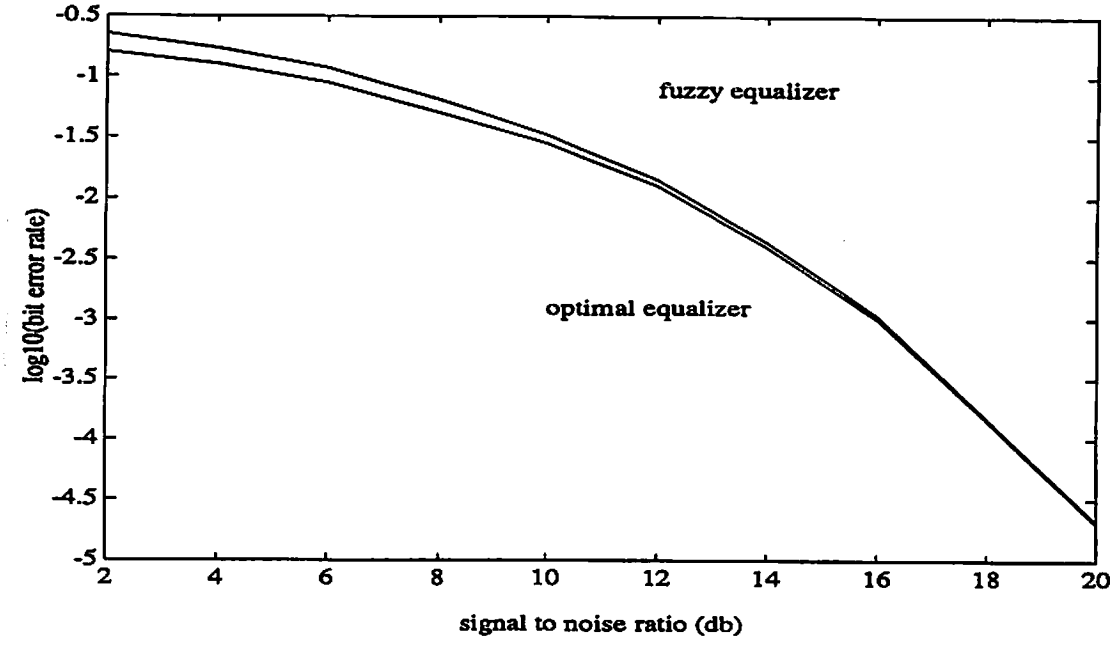


Figure 6.15: Comparison of bit error rates achieved by the optimal and fuzzy equalizers (Example 6.6).

nonlinear adaptive filters, similar to the polynomial, neural nets, or radial basis function adaptive filters. We applied the two fuzzy adaptive filters to nonlinear channel equalization problems. Simulation results showed that: 1) the fuzzy adaptive filters worked quite well without using any linguistic information; 2) by incorporating some linguistic rules into the fuzzy adaptive filters, the adaptation speed was greatly improved; and, 3) the bit error rates of the fuzzy equalizers were close to that of the optimal equalizer.

Chapter 7

CONCLUSIONS AND FUTURE WORK

In this thesis, we:

1) Present a systematic description of fuzzy systems and show that probabilistic general regression is a special case of fuzzy systems.

2) Prove that the fuzzy systems are capable of approximating any nonlinear function over a compact set to arbitrary accuracy.

3) Develop a back-propagation algorithm to train the fuzzy systems to match desired input-output pairs, and use the back-propagation fuzzy systems as identifiers of nonlinear dynamic systems.

4) Represent fuzzy systems as expansions of fuzzy basis functions, use the orthogonal least squares algorithm to select the siginificant fuzzy basis functions, and apply this method to the nonlinear ball and beam control problem.

5) Develop a simple method for fuzzy system design which performs only a one-pass operation over the data pairs, and apply this method to the truck backer-upper control and time series prediction problems.

6) Develop two nonlinear adaptive filters based on fuzzy system models, namely RLS and LMS fuzzy adaptive filters, and use them as nonlinear channel equalizers.

Continuing the research in this thesis, we may do the following:

1) A disadvantage of the back-propagation algorithm in Chapter 3 is that it requires the fuzzy systems to be differentiable with respect to the parameters;

however, many fuzzy systems use max, min operations which are nondifferentiable. To solve this problem, we can use the classical random direction stochastic approximation algorithm.

2) In the RLS and LMS fuzzy adaptive filters, we need to determine the order of the filters before processing. A good way to determine the order as well as the parameters is to use the lattice filter idea, i.e., we can develop a fuzzy lattice filter.

3) To expand the work in Chapter 6 to blind equalization, we may use unsupervised learning algorithms to determine the cluster centers of the data, then view the centers, in some way, as learning samples, and use the fuzzy adaptive filters to process them.

4) A weakpoint of the methods in Chapters 3-6 is that there is no theoretical analysis of the performance of the proposed algorithms, i.e., theoretical analysis of these methods is clearly needed.

5) In Section 1.3 we showed that probabilistic general regression is a special case of fuzzy systems. To continue this work, we may compare fuzzy systems with multilayer perceptron, radial basis function expansions, potential function expansions, etc.. This kind of analyses may provide a deep understanding of the relationships among these nonlinear approaches.

Chapter 8

REFERENCES

[1] Bellman, R. E. and L. A. Zadeh, "Local and fuzzy logics," in *Modern Uses of Multiple-Valued Logic*, J. M. Dunn and G. Epstein, eds., Reidel Publ., Dordrecht, Netherlands, pp. 103-165, 1977.

[2] Bernard, J. A. "Use of rule-based system for process control," *IEEE Contr. Syst. Mag.*, Vol.8, No.5, pp.3-13, 1988.

[3] Biglieri, E., A. Gersho, R. D. Gitlin and T. L. Lim, "Adaptive cancellation of nonlinear intersymbol interference for voiceband data transmission," *IEEE J. on Selected Areas in Communications*, Vol. SAC-2, No. 5, pp. 765-777, 1984.

[4] Box, G. E. P. and G. M. Jenkins, "Time Series Analysis: Forecasting and Control," Holden-Day Inc., 1976.

[5] Chen, S., C. F. N. Cowan and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. on Neural Networks*, Vol. 2, No. 2, pp. 302-309, 1991.

[6] Chen, S., 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.

[7] Chen, S., G. J. Gibson, C. F. N. Cowan and P. M. Grant, "Adaptive equalization of finite non-linear channels using multilayer perceptrons," *Signal Processing*, Vol. 20, pp. 107-119, 1990.

[8] Chen, S., G. J. Gibson, C. F. N. Cowan and P. M. Grant, "Reconstruction of binary signals using an adaptive radial-basis-function equalizer," *Signal*

Processing, Vol. 22, pp. 77-93, 1991.

[9] Chiu, S., 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.

[10] Ciliz, K., J. Fei, K. Usluel and C. Isik, "Practical aspects of the knowledge-based control of a mobile robot motion," *Proc. 30th Midwest Symp. on Circuits and Systems*, Syracuse, NY, 1987.

[11] Cowan, C. F. N. and P. M. Grant (ed.), "Adaptive Filters," Prentice-Hall, Inc., New Jersey, 1985.

[12] Cybenko, G. "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals, and systems*, 1989.

[13] Dubois, D. and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, Inc., Orlando, Florida, 1980.

[14] Falconer, D. D., "Adaptive equalization of channel nonlinearities in QAM data Transmission systems," *The Bell System Technical Journal*, Vol. 57, No. 7, pp. 2589-2611, 1978.

[15] Fujitec, F. "FLEX-8000 series elevator group control system," Fujitec Co., Ltd., Osaka, Japan, 1988.

[16] Goodwin, G. C. and R. L. Payne, *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, Inc., 1977.

[17] Hale, J. K., *Ordinary Differential Equations*, Wiley-Inter Science, New York, 1969.

[18] Hauser, J., S. Sastry and P. Kokotovic, "Nonlinear control via approximate input-output linearization: the ball and beam example," *IEEE Trans. on Automatic Control*, Vol. AC-37, No. 3, pp. 392-398, 1992.

[19] Holden, A. V., *Chaos*, Princeton University Press, 1986.

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

[21] Isik, C., "Identification and fuzzy rule-based control of a mobile robot

motion," *Proc. IEEE Int. Symp. Intelligent Control*, Philadelphia, PA, 1987.

[22] Itoh, O., K. Gotoh, T. Nakayama and S. Takamizawa, "Application of fuzzy control to activated sludge process," *Proc. 2nd IFSA Congress*, Tokyo, Janan, pp.282-285, 1987.

[23] Kasai, Y. and Y. Morimoto, "Electronically controlled continuously variable transmission," *Proc. Int. Congress on Transportation Electronics*, Dearborn, MI, 1988.

[24] Kickert, W. J. M. and H. R. Van Nauta Lemke, "Application of a fuzzy controller in a warm water plant," *Automatica*, Vol.12, No.4, pp.301-308, 1976.

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

[26] Kiszka, J., M. Gupta and P. Nikiforuk, "Energetistic stability of fuzzy dynamic systems," *IEEE Trans. Systems, Man, and Cybern.*, Vol. SMC-15, No. 5, pp. 783-792, 1985.

[27] Klir, G. J. and T. A. Folger, "Fuzzy Sets, Uncertainty, and Information," Prentice Hall, New Jersey, 1988.

[28] Kong, S. G. and B.Kosko, "Comparison of Fuzzy and Neural Truck Backer-Upper Control Systems," *Proc. IJCNN-90*, Vol.3, pp.349-358, June, 1990.

[29] Langari, G. and M. Tomizuka, "Stability of fuzzy linguistic control systems," *Proc. IEEE Conf. on Decision and Control*, Hawaii, pp. 2185-2190, 1990.

[30] Lapedes, A. and R. Farber, "Nonlinear signal processing using neural networks: prediction and system modeling," *LA-UR-87-2662*, 1987.

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

[32] Larsen, P. M., "Industrial application of fuzzy logic control," *Int. J. Man Mach. Studies*, Vol.12, No.1, pp.3-10, 1980.

[33] Lee, C. C., "Fuzzy logic in control systems: fuzzy logic controller, part I," *IEEE Trans. on Syst., Man, and Cybern.*, Vol.SMC-20, No.2, pp.404-418, 1990.

[34] Lee, C. C., "Fuzzy logic in control systems: fuzzy logic controller, part II," *IEEE Trans. on Syst., Man, and Cybern.*, Vol.SMC-20, No.2, pp.419-435, 1990.

[35] Li, Y. F. and C.C.Lan, "Development of Fuzzy Algorithms for Servo Systems," *IEEE Control Systems Magazine*, Vol.9, No.3, pp.65-72, 1989.

[36] Lindley, D. V., "The probability approach to the treatment of uncertainty in artificial intelligence and expert systems," *Statistical Science*, Vol. 2, No. 1, pp. 17-24, 1987.

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

[38] Ljung, L., *System Identification — Theory for the User*, Prentice-Hall: Englewood Cliffs, NJ, 1987.

[39] Ljung, L., "Issues in system identification," *IEEE Control Systems Magazine*, pp. 25-29, Jan., 1991.

[40] Maiers, J. and Y. S. Sherif, "Applications of fuzzy sets theory," *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-15, No. 6, pp. 175-189, 1985.

[41] Mamdani, E. H., "Applications of fuzzy algorithms for simple dynamic plant," *Proc. IEE*, Vol.121, No.12, pp.1585-1588, 1974.

[41] Mamdani, E. H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man Mach. Studies*, Vol.7, No.1, pp.1-13, 1975.

[43] Mathews, V. J., "Adaptive ploynomial filters," *IEEE Signal Processing Magazine*, pp. 10-26, July, 1991.

[44] Meisel, W. S., "Potential functions in mathematical pattern recognition," *IEEE Trans. on Computers*, Vol. C-18, No. 10, pp. 911-918, 1969.

[45] Mulgrew, B. and C. F. N. Cowan, "Adaptive filters and Equalisers," Kluwer Academic Publishers, 1988.

[46] Murakami, S., "Application of fuzzy controller to automobile speed control system," *Proc. IFAC Symp. on Fuzzy Information, Knowledge Representation*

and Decision Analysis, Marseille, France, pp.43-48, 1983.

[47] Murakami, S. and M. Maeda, "Application of fuzzy controller to automobile speed control system," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.105-124, 1985.

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

[49] Narendra, K. S. and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks*, Vol.1, No.1, pp.4-27, 1990.

[50] Naylor, A. W. and G. R. Sell, *Linear Operator Theory in Engineering and Science*, Springer-Verlag, New York, 1982.

[51] Negoita, C. V. and P. A. Ralescu, "Application of fuzzy sets to system analysis," ISR. II, Birkhaeuser, Basel, 1975.

[52] Nguyen, D. and B. Widrow, "The truck backer-upper: an example of self-learning in neural networks," *IEEE Cont. Syst. Mag.*, Vol.10, No.3, pp.18-23, 1990.

[53] Ostergaad, J. J., "Fuzzy logic control of a heat exchange process," in *Fuzzy Automata and Decision Processes*, M.M.Gupta, G.N.Saridis and B.R.Gaines, Eds., Amsterdam: North-Holland, pp.285-320, 1977.

[54] Pappis, C. P. and E. H. Mamdani, "A fuzzy logic controller for a traffic junction," *IEEE Trans. Syst. Man Cybern.*, Vol.SMC-7, No.10, pp.707-717, 1977.

[55] Parzen, E., "On estimation of a probability density function and mode," *Ann. Math. Statist.*, Vol. 33, pp. 1065-1076, 1962.

[56] Pitas, I. and A. N. Venetsanopoulos, "Nonlinear Digital Filters," Kluwer Academic Publishers, 1990.

[57] Powell, M. J. D., *Approximation Theory and Methods*, Cambridge University Press, Cambridge, 1981.

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

[59] Rudin, W., *Principles of Mathematical Analysis*, McGraw-Hill, Inc., 1976.

[60] Rumelhart, D. E. and J. L. McCleland (eds.), *Parallel Distributed Processing I, II*, Cambridge: MIT Press, 1986.

[61] Sakai, Y., "A fuzzy controller in turning process automation," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.139-152, 1985.

[62] Scharf, E. M. and N. J. Mandic, "The application of a fuzzy controller to the control of a multi-degree-freedom robot arm," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.41-62, 1985.

[63] Schetzen, M., "The Volterra and Wiener Theories of Nonlinear Filters," J. Wiley, 1980.

[64] Self, K., "Designing with fuzzy logic," *IEEE Spectrum*, pp. 42-44, Nov., 1990.

[65] Specht, D. F., "A general regression neural network," *IEEE Trans. on Neural Networks*, Vol. 2, No. 6, pp. 568-576, 1991.

[66] Specht, D. F., "Generation of polynomial discriminant functions for pattern recognition," *IEEE Trans. Electron. Comput.*, Vol. EC-16, pp. 308-319, 1967.

[67] Stallings, W., "Fuzzy set theory versus Bayesian statistics," *IEEE Trans. Systems, Man, and Cybern.*, Vol. 7, No. 3, pp. 216-219, 1977.

[68] Sugeno, M. and K. Murakami, "Fuzzy parking control of model car," *Proc. 23rd IEEE Conf. on Decision and Control*, Las Vegas, 1984.

[69] Sugeno, M. and M. Nishida, "Fuzzy control of model car," *Fuzzy Sets Syst.*, Vol.16, pp.103-113, 1985.

[70] Sugeno, M. and K. Murakami, "An experimental study on fuzzy parking control using a model car," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.125-138, 1985.

[71] Tanscheit, R. and E. M. Scharf, "Experiments with the use of a rule-based self-organising controller for robotics applications," *Fuzzy Sets Syst.*, Vol.26, pp.195-214, 1988.

[72] Tong, R. M., "A control engineering review of fuzzy systems," *Automatica*, Vol. 13, pp. 559-569, 1977.

[73] Tong, R. M., "Some properties of fuzzy feedback systems," *IEEE Trans. Systems, Man, and Cybern.*, Vol. SMC-10, No. 6, pp. 327-330, 1980.

[74] Tong, R. M., M. B. Beck and A. Latten, "Fuzzy control of the activated sludge wastewater treatment process," *Automatica*, Vol.16, No.6, pp.695-701, 1980.

[75] Togai, M. and H. Watanabe, "Expert system on a chip: an engine for real-time approximate reasoning," *IEEE Expert Syst. Mag.*, Vol.1, pp.55-62, 1986.

[76] Togai, M. and S. Chiu, "A fuzzy accelerator for a programming environment for real-time fuzzy control," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.147-151, 1987.

[77] Umbers, I. G. and P. J. King, "An analysis of human-decision making in cement kiln control and the implications for automation," *Int. J. Man Mach. Studies*, Vol.12, No.1, pp.11-23, 1980.

[78] Uragami, M., M. Mizumoto and K. Tananka, "Fuzzy robot controls," *Cybern.* Vol.6, pp.39-64, 1976.

[79] Wang, L. X., "Fuzzy systems are universal approximators," *Proc. IEEE International Conf. on Fuzzy Systems*, pp. 1163-1170, San Diego, 1992.

[80] Wang, L. X., "A neural detector for seismic reflectivity sequences," *IEEE Trans. on Neural Networks*, Vol. 3, No. 2, pp. 338-340, 1992.

[81] Wang, L. X., G. Z. Dai and J. M. Mendel, "One-pass minimum variance deconvolution algorithms," *IEEE Trans. on Automatic Control*, Vol. AC-35, No. 3, pp. 326-329, 1990.

[82] Wang, L. X. and J. M. Mendel, "Generating fuzzy rules by learning from examples," *Proc. 6th IEEE International Symposium on Intelligent Control*, pp. 263-268, 1991.

[83] Wang, L. X. and J. M. Mendel, "Generating fuzzy rules from numerical data, with applications," USC SIPI Report, No. 169, 1991; also, accepted for publication in *IEEE Trans. on Systems, Man, and Cybern.*, 1992.

[84] Wang, L. X. and J. M. Mendel, "Analysis and design of fuzzy logic controller," USC SIPI Report, No. 184, 1991; also, accepted for publication in *IEEE Trans. on Automatic Control*, 1992.

[85] Wang, L. X. and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning," to appear in *IEEE Trans. on Neural Networks*, Sept., 1992.

[86] Wang, L. X. and J. M. Mendel, "Back-propagation fuzzy systems as nonlinear dynamic system identifiers," *Proc. IEEE International Conf. on Fuzzy Systems*, pp. 1409-1418, San Diego, 1992.

[87] Wang. L. X. and J. M. Mendel, "Structured trainable networks for matrix algebra," *Proc. 1990 International Joint Conf. on Neural Networks*, Vol. 2, pp. II125-II132, 1990.

[88] Wang, L. X. and J. M. Mendel, "Matrix computations and equation solving using structured networks and training," *Proc. IEEE 1990 Conf. on Decision and Control*, pp. 1747-1750, 1990.

[89] Wang, L. X. and J. M. Mendel, "Parallel structured network for solving a wide variety of matrix algebra problems," to appear in *Journal of Parallel and Distributed Computing*, March, 1992.

[90] Wang, L. X. and J. M. Mendel, "Three-dimensional structured networks for matrix equation solving," *IEEE Trans. on Computers*, Vol. 40, No. 12, pp. 1337-1346, 1991.

[91] Wang, L. X. and J. M. Mendel, "Cumulant-based parameter estimation using structured networks," *IEEE Trans. on Neural Networks*, Vol. 2, No. 1, pp. 73-83, 1991.

[92] Wang, L. X. and J. M. Mendel, "Adaptive prediction-error deconvolution and source wavelet estimation using Hopfield neural networks," to appear in Geophysics, May, 1992.

[93] Wang, L. X. and J. M. Mendel, "A fuzzy approach to hand-written rotation-invariant character recognition," *Proc. ICASSP-92*, pp. III145-III148, San Francisco, 1992

[94] Watanabe, H. and W. Dettloff, "Reconfigurable fuzzy logic processor: a full custom digital VLSI," *Proc. Int. Workshop on Fuzzy System Applications*,

Iizuka, Japan, pp.49-50, 1988.

[95] Werbos, P., "New Tools for Predictions and Analysis in the Behavioral Science," Ph.D. Thesis, Harvard U. Committee on Applied Mathematics, 1974.

[96] Widrow, B. and S. D. Stearns, "*Adaptive Signal Processing*," Prentice-Hall, Englewood Cliffs, NJ, 1984.

[97] Yamakawa, T. and T. Miki, "The current mode fuzzy logic integrated circuits fabricated by standard CMOS process," *IEEE Trans. Computer*, Vol.C-35, No.2, pp.161-167, 1986.

[98] Yamakawa, T. and K. Sasaki, "Fuzzy memory device," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.551-555, 1987.

[99] Yamakawa, T., "Fuzzy microprocessors – rule chip and defuzzification chip," *Proc. Int. Workshop on Fuzzy System Applications*, pp.51-52, 1988.

[100] Yagishita, O., O. Itoh and M. Sugeno, "Application of fuzzy reasoning to the water purification process," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.19-40, 1985.

[101] Yasunobu, S., S. Miyamoto and H. Ihara, "Fuzzy control for automatic train operation system," *Proc. 4th IFAC/IFIP/IFORS Int. Congress on Control in Transportation Systems*, Baden-Baden, 1983.

[102] Yasunobu, S. and S. Miyamoto, "Automatic train operation by predictive fuzzy control," in *Industrial Application of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.1-18, 1985.

[103] Yasunobu, S., S. Sekino and T. Hasegawa, "Automatic train operation and automatic crane operation systems based on predictive fuzzy control," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.835-838, 1987.

[104] Yasunobu, S. and T. Hasegawa, "Evaluation of an automatic container crane operation system based on predictive fuzzy control," *Control Theory Adv. Technol.*, Vol.2, No.3, pp.419-432, 1986.

[105] Yasunobu, S. and T. Hasegawa, "Predictive fuzzy control and its application for automatic container crane operation system," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.349-352, 1987.

[106] Zadeh, L. A., "Fuzzy sets and systems," *Proc. Symp. Syst. Theory*, Polytech. Inst. Brooklyn, pp. 29-37, 1965.

[107] Zadeh, L. A., "Fuzzy sets," *Informat. Control*, Vol. 8, pp. 338-353, 1965.

[108] Zadeh, L. A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. on Systems, Man, and Cybern.*, Vol. SMC-3, No. 1, pp. 28-44, 1973.

[109] Zadeh, L. A., "A theory of approximating reasoning," in *Machine Intelligence*, J. E. Hayes, D. Michie and L. I. Mikulich, eds., Vol. 9, Elsevier, New York, pp. 149-194, 1979.

[110] Zadeh, L. A., "Syllogistic reasoning in fuzzy logic and its application to usuality and reasoning with dispositions," *IEEE Trans. on Systems, Man, and Cybern.*, Vol. SMC-15, No. 6, pp. 754-763, 1985.