Spring 1994

Adaptive nonlinear control using fuzzy logic and neural networks

Shu-Chieh Chang
New Jersey Institute of Technology

Adaptive nonlinear control using fuzzy logic and neural networks

Chang, Shu-Chieh, Ph.D.

New Jersey Institute of Technology, 1994

ABSTRACT

ADAPTIVE NONLINEAR CONTROL USING FUZZY LOGIC AND NEURAL NETWORKS

By
Shu-Chieh Chang

The problem of adaptive nonlinear control, i.e. the control of nonlinear dynamic systems with unknown parameters, is considered. Current techniques usually assume that either the control system is linearizable or the type of nonlinearity is known. This results in poor control quality for many practical problems. Moreover, the control system design becomes too complex for a practicing engineer. The objective of this thesis is to provide a practical, systematic approach for solving the problem of identification and control of nonlinear systems with unknown parameters, when the explicit linear parametrization is either unknown or impossible.

Fuzzy logic (FL) and neural networks (NNs) have proven to be the tools for universal approximation, and hence are considered. However, FL requires expert knowledge and there is a lack of systematic procedures to design NNs for control. A hybrid technique, called fuzzy logic adaptive network (FLAN), which combines the structure of an FL controller with the learning aspects of the NNs is developed. FLAN is designed such that it is capable of both structure learning and parameter learning. Gradient descent based technique is utilized for the parameter learning in FLAN, and it is tested through a variety of simulated experiments in identification and control of nonlinear systems. The results indicate the success of FLAN in terms of accuracy of estimation, speed of convergence, insensitivity against a range of initial learning rates, robustness against sudden changes in the input as well as noise in the training data. The performance of FLAN is also compared with the techniques based on FL and NNs, as well as several hybrid techniques.

ADAPTIVE NONLINEAR CONTROL USING FUZZY LOGIC AND NEURAL NETWORKS

by
Shu-Chieh Chang

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Mechanical and Industrial Engineering

May 1994

APPROVAL PAGE

ADAPTIVE NONLINEAR CONTROL USING FUZZY LOGIC AND NEURAL NETWORKS

Shu-Chieh Chang

Dr. Rajesh N. Dave, Dissertation Advisor — Date
Associate Professor of Mechanical and Industrial Engineering, NJIT

Dr. Nirwan Ansari, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Rong-Yaw Chen, Committee Member — Date
Professor of Mechanical and Industrial Engineering, NJIT

Dr. Zhiming Ji, Committee Member — Date
Assistant Professor of Mechanical and Industrial Engineering, NJIT

Dr. Nouri Levy, Committee Member — Date
Associate Professor of Mechanical and Industrial Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Shu-Chieh Chang

Degree: Doctor of Philosophy in Mechanical Engineering

Date: May 1994

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mechanical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1994

- Master of Science in Mechanical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1990

- Diploma in Mechanical Engineering,
  Ming-Chi Institute of Technology, Taipei, Taiwan, R.O.C., 1984

Major: Mechanical Engineering

Presentations and Publications:

Chang, Shu-Chieh, and Rajesh N. Dave. 1993. "Identification of Nonlinear Dynamical Systems Using Fuzzy Neural Networks." Presented in the 9-th International Conference on CAD/CAM, Robotics, & Factories of the Future (a best student paper award).

Chang, Shu-Chieh, and Rajesh N. Dave. 1993. "Singularity Avoidance for Redundant Robots Using Fuzzy Neural Networks." Presented in the 9-th International Conference on CAD/CAM, Robotics, & Factories of the Future.

Chang, Shu-Chieh, and Rajesh N. Dave. 1993. "Singularity Avoidance for Redundant Robots Using Self-Learning Fuzzy Logic Controllers." *The Proceedings of the 12-th Annual Meeting of the North-American Fuzzy Information Processing Society (NAFIS-'93):*134 - 138.

This dissertation is dedicated
to my father, God rest his soul
and to my mother.

ACKNOWLEDGMENT

First I would like to express my sincere gratitude to my dissertation advisor Dr. Rajesh N. Dave. This work would not have been possible without his guidance and continuous support.

I am also very grateful to Dr. Nirwan Ansari, Dr. Rong-Yaw Chen, Dr. Zhiming Ji, and Dr. Nouri Levy for serving on my doctoral committee.

I thankfully acknowledge the continuous financial support from New Jersey Institute of Technology.

Finally and most importantly, my deepest gratitude goes to my mother, my wife and my sisters, for their patience and constant moral support throughout these years of studies at New Jersey Institute of Technology.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 General Background

Control, in general, means to regulate a process by comparing the measurements with some prescribed performance specification. It happens so common in our everyday life that no one can exactly describe its history from the very beginning. However, we still can roughly categorize the development of the control theories as *conventional control theory* and *modern control theory*.

Generally speaking, the control theory developed through the late 1950s may be classified as conventional control theory, which is considerably effective and sufficient for simple systems, such as *single-input single-output* (SISO) systems and *linear time-invariant* (LTI) systems. After World War II, due to the requirements of faster and more accurate control systems for military and space technologies, and the merge from other disciplines, researchers faced more complicated systems, such as *multiple-input multiple-output* (MIMO) systems and linear time-varying (LTV) systems. Great efforts were made to meet these requirements and promising results led the development of control theory into a new era. The age of modern control theory began in 1957, the time when the first sputnik was launched [Friedland 1986]. A great diversity of practical approaches and successful industrial applications have been found since then.

However, most of the methods based on modern control theory are useful only when the system is operated in a small range because they are developed on the basis of linear control theory which assumes the system can be controlled by a linear controller to fulfill the performance specification. Due to the technological advances, there is a greater need to control systems where there may be *hard nonlinearities* (unknown or difficult to model nonlinearities [Slotine and Li 1991]) and model uncertainties which make the task of controller design more and more complicated. An industrial robot, for example, involves nonlinearities such as the friction force at each joint, nonlinear torques caused by Coriolis and centrifugal effects, and uncertainties originated in modeling and operating, such as the deflections in robot arms and the variation in working load, etc. Developing an appropriate controller to control a robot, therefore, becomes a challenging problem [Craig 1988, Bobrow *et al.* 1983, Newman and Souccar 1991].

Most engineering control problems, like the case of robot control, are nonlinear in nature. While considerable progress has been made in linear control theory, the developments in nonlinear control theory has been limited [Isidori 1989, Slotine and Li 1991, Kokotovic 1991, Friedland 1986]. This motivates the study of nonlinear control theory. Moreover, the recent improvements in microprocessors and computer simulation techniques enable researchers to explore more details of nonlinear systems. Since late 1980s, growing research on adaptive control [Goodwin and Sin 1984, Narendra and Annaswamy 1989, Åström and Wittenmark 1989] and intelligent control [Barto 1989, and Lee 1990] has shown a great potential for these methods to deal with nonlinear systems.

Nonlinear control problems can be divided in three types. In the first type the nonlinearities in the system are known. For this case, the most common approach is to apply feedback linearization or compensation of nonlinearities [Isidori 1989]. In robotics for example, nonlinearities are canceled by means of a nonlinear feedback controller as done in the classic work of [Paul 1981] and [Bejczy 1974]. In these techniques, however, knowledge of full dynamic model and accurate parameters of the payload and manipulator are required. These requirements are difficult to meet in practice as mentioned before. In fact, for practical systems the cancellations of nonlinearities may not be achieved and then the dynamic performance of the robot is poor, and may lead to instability [Hewit 1979, Egeland 1986]. Moreover, these situations require complicated stability analysis. In the second type, there are uncertainties in the input and external disturbances. In this case, the common approach is to employ robust control strategies [Hached 1990]. Usually, the uncertainties are assumed to be bounded, and the knowledge of nonlinearities is required. In the third type, the nonlinear model is unknown. In this case, adaptive nonlinear control is used to model or approximate the unknown nonlinearities [Hunt and Turi 1993, Newman and Souccar 1991, Seraji, 1987]. While much attention has been given recently to these types of control problems, in particular to the first two cases, there are limitations to most proposed techniques. Therefore more research is required, as will be discussed in the next section, in order to develop flexible and practical approaches that can be easily adopted in industry.

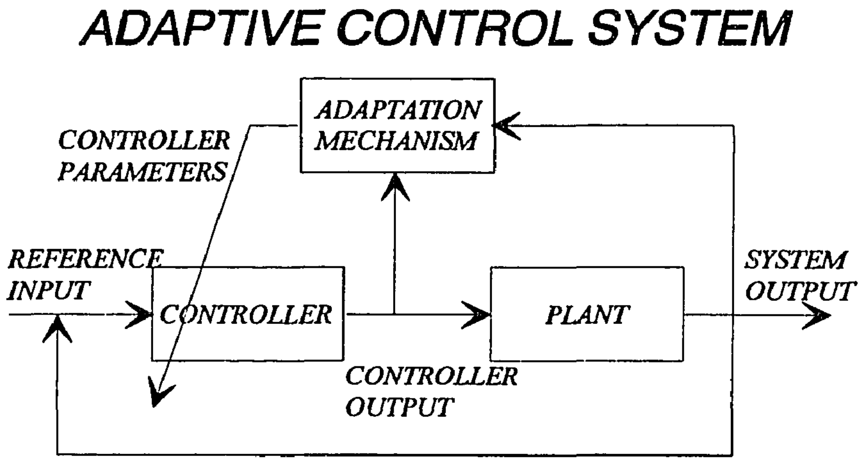

Figure 1.1 Basic structure of an adaptive control system.

1.2 Adaptive Nonlinear Control

Adaptive control, an important branch of modern control theory, refers to the control of a system, either linear or nonlinear, with uncertain or unknown parameters. These uncertain or unknown parameters may result from imperfect modeling or measurements of the system, or unpredictable changes of the inputs and disturbances to the system. Figure 1.1 shows the basic idea of adaptive control. It differs from a conventional controller in that its parameters can be modified by an adaptation mechanism so as to meet the prescribed performance specification. There are three basic components in common adaptive control: namely, 1) an adaptation mechanism that contains a reference model that gives the desired system output or an identification model that estimates the unknown parameters, and produces a performance index by comparing the estimated output with the actual one; 2) an effective adaptation law, also called a controller, that contains modifiable parameters and generates proper signals by tuning the controller parameters based on the performance index and then send these signals to the system; and 3) the unknown system.

When the unknown system is linear, there are two principal approaches to adaptive control, which are *model reference adaptive control* (MRAC) and *self-tuning regulators* (STR). In the MRAC, a reference model is used to produce the desired trajectory for a given command input or reference input. The objective of the adaptive controller is to generate the control input that forces the system to follow the desired trajectory. For example, in many control problems related to mechanical engineering applications, such as robotics control, a linear model of the robot manipulator under control is developed and then used in MRAC based adaptive control scheme [Landau 1979, Dubowsky and DesForges 1979, Balestrino, DeMaria and Sciavicco 1983, Craig 1988]. Instead of having a reference model, the STR uses an identification model to estimate the system parameters. The parameters of the controller are then modified based on the estimated system parameters. Excellent information on the history and development of adaptive control of linear time-invariant systems with unknown parameters can be found in [Goodwin and Sin 1984, Chalam 1987, Narendra and Annaswamy 1989, Åström and Wittenmark 1989].

Recent advances in nonlinear control theory and adaptive control theory [Isidori 1989, Slotine and Li 1991, Kokotovic 1991] have motivated studies aimed at developing adaptive control schemes for nonlinear dynamic systems. The result of these activities have led to the emergence of the new subject, termed *nonlinear adaptive control*. A quick review of recent articles in leading control magazines and journals indicates the abundance of literature in this new area. However, careful scrutiny indicates that most methods are based on either an assumption regarding linearization of the model or some knowledge about the type or the order of nonlinearity. For example, in [Ortega *et al.* 1993] it is required that the time varying unknown loads are linearly parametrizable. For control of manipulators, [Seraji 1987] assumes that the nonlinear plant is approximated by quasi-linear time invariant model of plant variables. In [Newman and Souccar 1991] the nonlinearity is assumed to be of second order and feedback linearization based on the classification of the system state into certain regions is required. Most systems usually rely on some kind of feedback linearization, and that becomes an essential tool for nonlinear systems. However, quoting from [Hunt and Turi 1993] "... even if a system is feedback linearizable it can be extremely difficult, if not impossible, to construct an exact feedback linearizing transformation." Apart from some of these limitations, the major problem with these conventional nonlinear control schemes is that they are quite complex in terms of implementation and almost always require a certain amount of knowledge of the plant. Moreover, most of these schemes pose a great challenge for an average practicing engineer. These reasons have led to the emergence of the new field of intelligent control.

1.3 Intelligent Control Using Fuzzy Logic and Neural Networks

Some of the ideas for intelligent control are based on the observation that an experienced human being can achieve complicated control tasks without having exact knowledge of the plant. Consequently, the use of fuzzy control (FC) and neural networks (NNs) to solve the problem of controlling nonlinear dynamic systems with unknown parameters has received the attention of many researchers [Takagi and Sugeno 1985, Miller, Sutton, and Werbos 1990, White and Sofge 1992] recently because of their great potential in dealing with complex, nonlinear mappings. In fact, fuzzy logic and NNs have been proven to be universal approximators, and therefore are ideal for approximating unknown nonlinearities

[Kosko 1992b, Wang 1992, Funahashi 1989, Hornik *et al.* 1989]. Recent research has shown the promising results of applying these techniques to the control of nonlinear systems. For example, FC has been used successfully for automobile transmission control, anti-lock brake system operation, subway system control, air conditioner control, etc. (see [Lee 1990] and references therein). NNs have been used for system identification and control of nonlinear systems [Narendra and Parthsarathy 1990], and inverse robot kinematics [Kim and Yoon 1992].

Despite the great potential of these two techniques to solve complex and ill-structured problems in nonlinear control, there are several drawbacks to each approach. In the FC approach, the development of the membership functions and linguistic control rules relies on the availability of the experienced operators or experts. In practice, such knowledge is either unavailable or is difficult to extract or represent particularly for complex systems. Similarly, in the NN approach, the design of the network architecture is based on designer's experience. There is no explicit guideline for determining the configuration, such as the number of layers and the number of nodes in each layer of an NN. Due to these problems, the applications of the FL and NNs are very limited.

1.4 Objective and Scope of the Dissertation

The main objective of this dissertation is to develop a systematic and flexible approach for solving control problems with unknown nonlinearities. The idea is to propose a method that is easy to implement and is highly practical and therefore suitable for application in industrial environment.

Based on the discussion presented here, it is apparent that the conventional adaptive nonlinear control techniques are not simple enough for this purpose. On the other hand, FC and NN approaches are very good candidates for this objective. Unfortunately, although FC and NNs have been successfully applied to the control of nonlinear systems with uncertain or unknown parameters, there are some difficulties in the practical implementation as mentioned previously. The interesting point about these two methods, however, is that they have complimenting strengths, i.e., the FC provides a compact structure for control, while NNs provide learning ability that FC lacks. Therefore, combination of these techniques may provide answer to the question of developing practical approaches for control of complex nonlinear systems.

The major contribution of this dissertation is develpoment of a hybrid scheme that merges the concepts of FC and NNs together to identify and control nonlinear systems having unknown parameters. As mentioned earlier, the FC provides a nice structure for control, and provides a meaningful physical description of the control scheme. On the other hand, the controller based on NN appears like a black box and does not provide a good physical interpretation of the control scheme. In addition, its structure is rather arbitrary as compared to the structure of an FLC. Therefore, it is decided to base the proposed hybrid scheme essentially on an FLC, but use the learning procedure in the NNs to tune the parameters of the FLC. The resulting FLC may be used as an approximator. The approximator is then used as an identification model and a controller in the adaptive control system. The proposed scheme treats the FLC as an adaptive network, and thus is called Fuzzy Logic Adaptive Network (FLAN). Since the architecture of the FLAN is determined only by the number of membership functions, it simplifies the design procedure for developing an FLC. A brief review of FL and NNs along with the details of the new scheme (FLAN) is presented in chapter 2. A discussion of other hybrid approaches is also included.

Chapter 3 presents the problem of identifying nonlinear dynamic systems with unknown parameters using the FLAN. The identification models used in this chapter are adopted from [Narendra and Parthasarathy 1990], where NNs are used for identification. Two examples are given to demonstrate the successful use of the FLAN to identify nonlinear systems with unknown parameters. The identification results using the FLAN are compared with the results from [Narendra and Parthasarathy 1990] and others. The comparison includes the accuracy and learning aspects.

In chapter 4, the issue of designing the adaptive controller for the nonlinear control system is discussed. The indirect adaptive control scheme is used to demonstrate the result of combining the identification model developed in chapter 3 and the adaptive controller developed in chapter 4. An example is included to show that the controller, designed using the FLAN, is able to closely follow the desired trajectory given by a reference model. Here also, these results are compared with the work of [Narendra and Parthasarathy 1990].

Chapter 5 concludes this dissertation with summary of the research results. The merits and limitations of the new scheme are discussed, followed by the directions of future research.

The detailed derivation of the FLAN is given in appendix A. Although the stability analysis of FLAN is beyond the scope of this work, appendix B presents a brief discussion on stability of the nonlinear system used in chapter 4.

CHAPTER 2

DEVELOPMENT OF FUZZY LOGIC ADAPTIVE NETWORK (FLAN)

2.1 Introduction

In this chapter, first some basic concepts of fuzzy logic (FL) and neural networks (NNs), and their applications to the control problems are briefly reviewed. Discussion on the relative merits of the recently proposed hybrid schemes that combine FL and NNs is also presented. First, the basic concepts of FL, and the definitions of a fuzzy set and some important operators are reviewed in section 2.2. This is followed by a brief explanation of the basic structure of a fuzzy logic controller (FLC). This introduction is necessary for the development of FLAN. Next, the fundamental ideas of an NN and the leaning procedure used to train the network are reviewed in section 2.3. A backpropagation neural network (BPNN) is used as an example to describe how to employ an NN to solve a control problem. Then, with the basic concepts of FL and NNs in mind, several hybrid schemes that combine FL and NNs together are discussed in Section 2.4. This discussion naturally leads to the development of the proposed new scheme, FLAN. The detailed derivation of the learning procedure for the FLAN is given in appendix A.

2.2 Fuzzy Logic

2.2.1 General Introduction

In real life, we are commonly faced with many vague or inexact objects. In fact, if we observe carefully, we can find that most of the subjects are matters of degree. For example, the words *commonly* and *most of the subjects* in the previous sentences are subjective terms that describe the fuzziness of some facts. This fuzziness is different from what traditional science tells us. We need a fuzzy description for the fuzzy world.

Kosko said [Kosko 1993], *"Fuzzy logic is reasoning with fuzzy sets."* FL is a mathematical scheme that allows us to analyze the vague or inexact objects. No one would speak about the subject of fuzzy logic without mentioning the great founder, Dr. Lotfi A. Zadeh, who invented fuzzy set theory [Zadeh 1965]. He opened the door to a new subject that allows scientists and engineers to explore the decision space between 0 and 1. His remarkable contribution has allowed us, therefore, to use FL as a tool to describe this fuzzy world.

A fuzzy set is defined as a collection of the degrees of compatibility to a subject. For example, to discuss the tallness, we may use words *short* and *tall*, which are linguistic labels of the fuzzy sets *short* and *tall*, to describe the observation. The degree of compatibility to each fuzzy set then can be subjectively determined, usually between 0 and 1. For example, given a person 5'5" tall, we can label the person as short to the degree 0.7 and tall to the degree 0.3. Therefore, the ordered pair (5'5", 0.7) belongs to fuzzy set *short*. Similarly, the ordered pair (5'5", 0.3) belongs to fuzzy set *tall*.

Definition 2.1 If x is the object of interest in the universe of discourse $\mathbf{X}$ then a fuzzy set $\tilde{\mathbf{A}}$ in $\mathbf{X}$ is defined as a set of ordered pairs:

$$\tilde{\mathbf{A}} = \left\{ \left(x, \mu_{\tilde{\mathbf{A}}/x} \right) \middle| x \in \mathbf{X} \right\} \tag{2.1}$$

where $\mu_{\tilde{\mathbf{A}}/x}$ is the degree of compatibility of object x to fuzzy set $\tilde{\mathbf{A}}$.

The above definition presents the fuzzy set in a discrete format, where the elements of the fuzzy set are enumerated pair by pair. Alternatively, we can use a function, called membership function, to describe the mapping from the observation space to membership space. Then equation (2.1) can be reproduced as

$$\tilde{\mathbf{A}} = \left\{ \left(x, \mu_{\tilde{\mathbf{A}}}(x) \right) \middle| x \in \mathbf{X} \right\} \tag{2.2}$$

where $\mu_{\tilde{A}}(x)$ is the degree of membership function defined as

$$\mu_{\tilde{A}} : \mathbf{X} \rightarrow [0,1] \tag{2.3}$$

Next, several important operations and the terminologies that are frequently used in the fuzzy set theory are presented.

- fuzzy union (Triangular Co-Norms, or S-norm):

$$\begin{aligned}\mu_{(A \cup B)}(x) &= \max(\mu_A(x), \mu_B(x)) \\ &= \mu_A(x) \vee \mu_B(x) \\ &= \mu_A(x) + \mu_B(x)\end{aligned} \tag{2.4}$$

- fuzzy intersection (Triangular Norms, or T-norm):

$$\begin{aligned}\mu_{(A \cap B)}(x) &= \min(\mu_A(x), \mu_B(x)) \\ &= \mu_A(x) \wedge \mu_B(x) \\ &= \mu_A(x) \cdot \mu_B(x)\end{aligned} \tag{2.5}$$

- fuzzy negation (complement):

$$\mu_{\bar{A}} = 1 - \mu_A(x) \tag{2.6}$$

- fuzzy implication (the generalization of modus ponens (GMP) inference):

$$\begin{aligned}[(A \text{ and } B) \rightarrow C] &= \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z) \\ &= \mu_A(x) \cdot \mu_B(y) \cdot \mu_C(z)\end{aligned} \tag{2.7}$$

There are many more fuzzy operations and implication methods available in the fuzzy set theory. However, the intention here is to simply review some important and frequently used operators for the sake of development of the later sections. See [Kaufmann 1975, Dubois and Prade 1980, Kandel 1986, Lee 1990, Pedrycz 1993] for more details.

2.2.2 Fuzzy Control

Fuzzy control (FC) has been an active research topic since Mamdani first applied fuzzy set theory to control problems [Mamdani 1974]. The literature and applications in FC have been growing exponentially in recent years. An excellent survey of FC along with a general scheme for constructing a fuzzy logic controller, and the direction for further research is presented in [Lee 1990]. In [Mizumoto 1988, 1989, 1992], various fuzzy implications and defuzzification processes are introduced.

An FLC is a rule-based expert system designed for the purpose of control. It contains a set of linguistic control rules which enable the controller to capture the vague or inexact nature of the control system. Basically, a domain expert is needed to transform expert knowledge into a set of linguistic control rules, which is the essential part of the FLC. However, an FLC is not an expert control system because the design of an FLC may not entirely depend on the domain expert.

Generally, there are four methods used to develop the control strategy of an FLC:

- extracting the domain expert's opinion or knowledge
- modeling the actions of the experienced operator
- observing the response of the system
- introducing the self-learning process

The first three methods are not highly suitable from the practical point of view, because the domain expert in the first case may not be available many times; the action of the experienced operator in the second case is not easy to observe or transform to numerical data; and trial and error has to be used in the third case. The last method appears to be promising, but it is not simple in practice because there are no clear rules about how to introduce the learning algorithm into an FLC. This has been a new and active research area, and will be explored in more detail later in this chapter. This self learning aspect is the basic idea on which FLAN will be built upon.

An FLC usually consists of three parts: namely, the fuzzification process, which transforms a crisp input from the measurement into a fuzzy membership value, the fuzzy inference process, which contains a set of linguistic control rules and the decision-making mechanism, and the defuzzification process, which transforms the inferred result, a fuzzy number, into a crisp output as a control signal. Figure 2.1 shows the basic structure of an FLC.

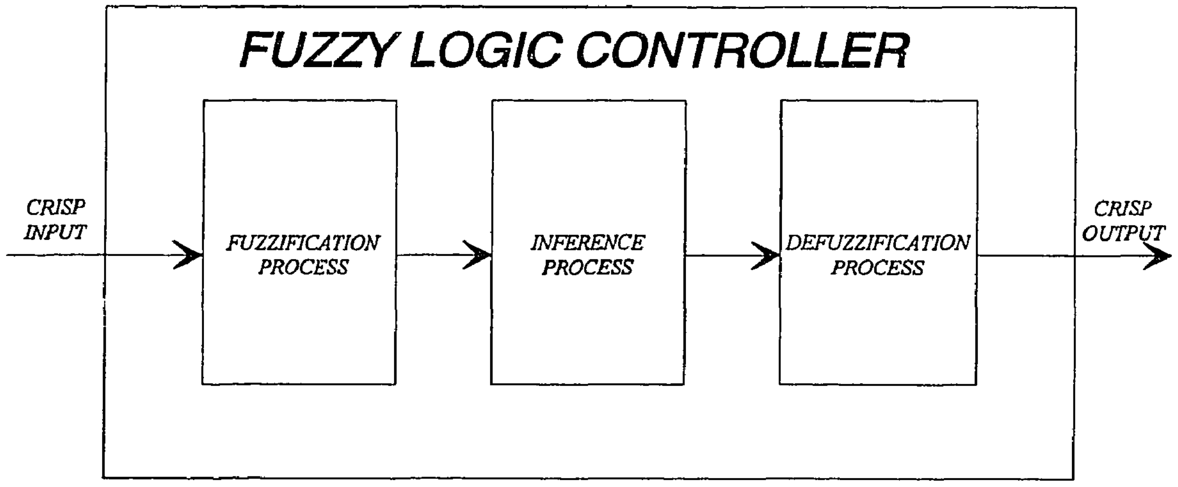

Figure 2.1 Basic structure of a fuzzy logic controller.

In the fuzzification process, the measured input signals are mapped to membership space. For each input signal, a grade of membership is assigned to each pre-defined fuzzy set. For example, given a crisp input signal x_0 and n linguistic values, $A_1, A_2, A_3, \ldots, A_n$, which may be treated as the labels of fuzzy sets, n fuzzy membership values $\mu_{A_i}(x_0)$, $i = 1,2,3,\ldots,n$ may then be obtained. To summarize the process, the notation in [Lee 1990] can be used:

$$x = \mathit{fuzzifier}(x_0) \tag{2.8}$$

where x_0 is usually a crisp input signal taken from the sensor, x represents a set of fuzzified numbers $\mu_{A_i}(x_0)$, $i = 1,2,3,\ldots,n$, and $\mathit{fuzzifier}(\cdot)$ represents the fuzzification process that maps the crisp input signal into the fuzzy membership value of each linguistic label. In the case that the input signal is taken from the observation of a human operator or in a noisy environment, x_0 can be a fuzzy number. The membership value is then determined by the max-min operation on the fuzzy input, which means taking the membership value to a certain fuzzy membership function as the maximum values of the intersections of the fuzzy input and the fuzzy membership function. Figure 2.2(a) and 2.2(b) show the cases of a crisp and a fuzzy input respectively. To simplify the demonstration, triangle membership functions are used. From the figure, it can be observed that a crisp input and a fuzzy input are assigned to the different degrees of membership. For simplicity, We shall only use the crisp inputs in the latter discussion of this dissertation.

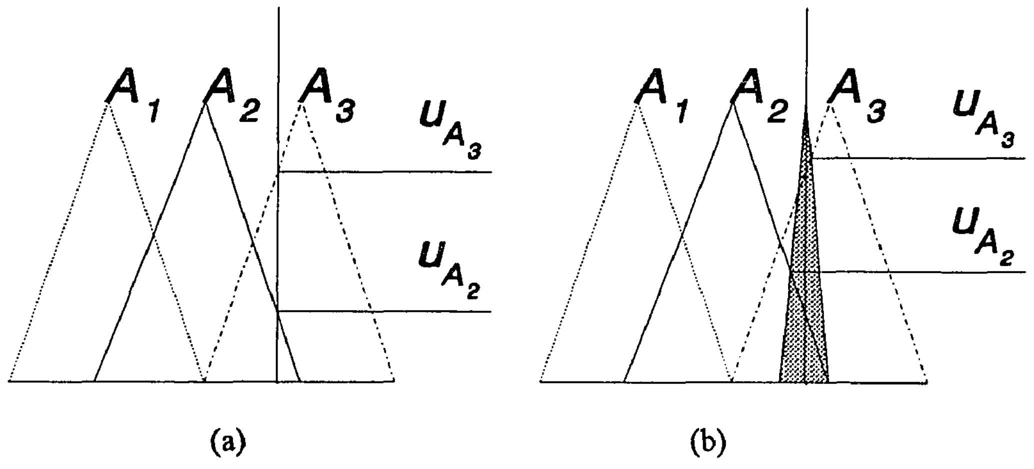

Figure 2.2 Fuzzification Process.

The fuzzy inference process receives the fuzzified input values, compares them with the antecedent parts of the linguistic control rules, and then produces a set of fuzzified numbers from the conclusion parts of the rules. In the case of two inputs and one output, the linguistic control rules can be typically expressed as:

$$\begin{aligned} &R_1\text{: if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } z \text{ is } C_1 \\ &R_2\text{: if } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } z \text{ is } C_2 \\ &R_3\text{: if } x \text{ is } A_3 \text{ and } y \text{ is } B_3 \text{ then } z \text{ is } C_3 \\ &\quad\cdots \\ &R_n\text{: if } x \text{ is } A_n \text{ and } y \text{ is } B_n \text{ then } z \text{ is } C_n \end{aligned} \tag{2.9}$$

where R_i, $i = 1,2,3,\ldots,n$ are the labels of the rules, x and y are input variables and z is the output variable. A_i, B_i, and C_i, $i = 1,2,3,\ldots,n$ are linguistic values, such as *tall*, *short*, and so on, for x, y, and z, respectively. The types of a control rules vary with the fuzzy reasoning mechanisms. In [Jang 1993], the author reviewed three types of commonly used rules.

Fuzzy control rules in equation (2.9) are actually implemented by a fuzzy implication:

$$[\mu_{A_i}(x) \text{ and } \mu_{B_i}(y)] \rightarrow [\mu_{C_i}(z)] \tag{2.10}$$

where $\mu_{A_i}(x)$, $\mu_{B_i}(y)$, and $\mu_{C_i}(z)$ are the grades of membership to A_i, B_i, and C_i, respectively, for $i = 1,2,3,\ldots,n$, and $\rightarrow$ denotes the process of a fuzzy implication. A variety of fuzzy implications can be chosen, see [Lee 1990, Mizumoto 1988, 1989] for more details.

Using the forward inference, named the generalized modus ponens (GMP), for example in [Lee 1990], we may demonstrate how to obtain the fuzzy conclusions. The GMP has the form:

$$\begin{array}{l} \textit{premise } 1\text{: } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z \text{ is } C \\ \underline{\textit{premise } 2\text{: } x \text{ is } A' \text{ and } y \text{ is } B' \qquad\qquad} \\ \textit{consequence}\text{: } z \text{ is } C' \end{array}$$

where A and A' are fuzzy sets in the universe of discourse U, B and B' are fuzzy sets in the universe of discourse V, and C and C' are fuzzy sets in the universe of discourse W.

Given x_0 and y_0 as the input signals, we may deduce the fuzzy consequence C' or $\mu_{C'}(z)$ from premises 1 and 2 by taking the max-min composition $\circ$ of the fuzzy relation $([(A \text{ and } B) \rightarrow C])$ in $U \times V \times W$ and the fuzzy set $(A' \text{ and } B')$ in $U \times V$. That is,

$$C' = [(A \text{ and } B) \to C] \circ (A' \text{ and } B') \tag{2.11}$$

or

$$\mu_{C'}(z) = \underset{x_0, y_0}{\vee} \left\{ [\mu_{A'}(x_0) \wedge \mu_{B'}(y_0)] \wedge [(\mu_A(x) \wedge \mu_B(y)) \to \mu_C(z)] \right\} \tag{2.12}$$

Now, if we translate the fuzzy implication $\mu_A(x) \wedge \mu_B(y) \to \mu_C(z)$ into the form $\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)$, we may obtain

$$\mu_{C'}(z) = \underset{x_0}{\vee} \left\{ \mu_{A'}(x_0) \wedge \mu_A(x) \wedge \mu_C(z) \wedge \underset{y_0}{\vee} [\mu_{B'}(y_0) \wedge \mu_B(y) \wedge \mu_C(z)] \right\} \tag{2.13}$$

Furthermore, from $\mu_{A'}(x_0) = \mu_{B'}(y_0) = 1$ and $\mu_{A'}(x) = \mu_{B'}(y) = 0$ for all $x \neq x_0$ and $y \neq y_0$ we can simplify equation (2.13) to

$$\mu_{C'}(z) = \mu_A(x_0) \wedge \mu_B(y_0) \wedge \mu_C(z) \tag{2.14}$$

We may further generalize the result in equation (2.14) to the case of inferring from n linguistic rules:

$$C' = C_1' \text{ or } C_2' \text{ or } C_3' \text{ or} \cdots \text{ or } C_n' \tag{2.15}$$

or

$$\begin{aligned} \mu_{C'}(z) = & [\mu_{A_1}(x_0) \wedge \mu_{B_1}(y_0) \wedge \mu_{C_1}(z)] \\ & \vee [\mu_{A_2}(x_0) \wedge \mu_{B_2}(y_0) \wedge \mu_{C_2}(z)] \\ & \vee [\mu_{A_3}(x_0) \wedge \mu_{B_3}(y_0) \wedge \mu_{C_3}(z)] \\ & \vee \cdots \vee [\mu_{A_n}(x_0) \wedge \mu_{B_n}(y_0) \wedge \mu_{C_n}(z)] \end{aligned} \tag{2.16}$$

Thus the process of fuzzy inference can be summarized by equation (2.14) and (2.16). Finally, the last part of an FLC is the defuzzification process that transforms the fuzzy consequences into a crisp output signal. The reason for having this process is that a fuzzy number is not acceptable as an output in practice. In the defuzzification process, the consequences of the inference, equation (2.15) or (2.16), are first summarized, and then mapped to the range of the output variable based on the defuzzification algorithm. The nonfuzzy output is used as the control action.

As in the case of the previous two steps, there are various defuzzification methods one can choose from, see [Lee 1990, Mizumoto 1989, 1992] for more details. Among them, the center of the gravity method is the most frequently used. Here, we simply use the notation in [Lee 1990] to represent the defuzzification process.

$$z_0 = defuzzifier(z) \tag{2.17}$$

2.2.3 Practical Issues

The inherent simplicity of an FLC makes it a very attractive tool for many practical applications, ranging from use in simple appliances such as cameras and washing machines [Lee 1990] to control of helicopters [Sugeno 1993]. For relatively simple problems, for example, anti-lock brakes in the car Saturn [Legg 1993], expert knowledge can be easily adapted into a controller without requiring much trial and error. However, when the problem is complex and the expert knowledge cannot be easily obtained or interpreted, the problem of control becomes very challenging [Sugeno 1993].

To summarize the discussion from the previous section, the following steps are required in order to construct an FLC. First, for each input, the number of membership functions and their shape parameters must be determined from expert knowledge. Next, a set of inference rules must be developed along with the parameters for the inference mechanism. Here, once again, expert knowledge and/or trial and error are required. Finally, the defuzzification mechanism has to be selected. All these steps may be easy for a very simple problem that does not require a high degree of accuracy. For example, in parking a car, one does not require a very high degree of accuracy in position relative to the curb. In that case, an FLC may be constructed without much trial and error. However, in control problems requiring high accuracy, for example, in robotics, the task of constructing an FLC is rather involved. For example, [Wu *et al.* 1992], present experiments using an FLC for control of both linear and nonlinear time varying dynamic DC motor driven systems. They show that an FLC for nonlinear control can be constructed with minimal knowledge about the system dynamics. They also point out, however, that through a self organizing scheme, the fuzzy membership parameters may be tuned to significantly improve the performance. Similarly, [Shao 1988] reports that tuning of the rules significantly improves the performance of a DC motor speed. Without the tuning, the FLC results in excessive oscillations and large errors. Although these examples are recent, the need for the use of learning procedures in design of an FLC has been noticed since [Procyk and Mamdani 1979]. These and other examples show a varying amount of success achieved through a variety of tuning procedures. However, there is a need for development of a more systematic approach. This is one of the major objectives of this work. The techniques that borrow ideas from the learning abilities of NNs show a promise for development of a systematic approach. Discussion on these techniques is presented later, in section 2.4, after NN based techniques are reviewed in the next section.

2.3 Neural Networks

2.3.1 General Introduction

For centuries, scientists have been searching for the explanation to how a human being recognizes the face of a friend, memorizes a special event, or simply speaks a word. The answer is not yet complete. Generally speaking, these behaviors that a human being performs daily are some kinds of nonlinear transformations. However, it is hard to describe mathematically how these nonlinear transformations are done. In the case of recognizing a face, we may briefly conclude that the brain associates a linguistic description with a face pattern. In addition to this amazing ability of association, a brain can also, for example, adapt to the change of a face caused by aging. It refers to the ability of generalization. These extraordinary abilities have inspired researchers to develop so-called artificial neural networks (or simply termed NNs) to emulate the functions the brain performs everyday.

An NN is a computational paradigm that contains a large amount of neuron-like processing units, called nodes or threshold units. A node takes the input signals from other connecting nodes and summarizes an output signal based on its node function, also named activation function. Figure 2.3 illustrates the essence of a neuron-like processing unit.

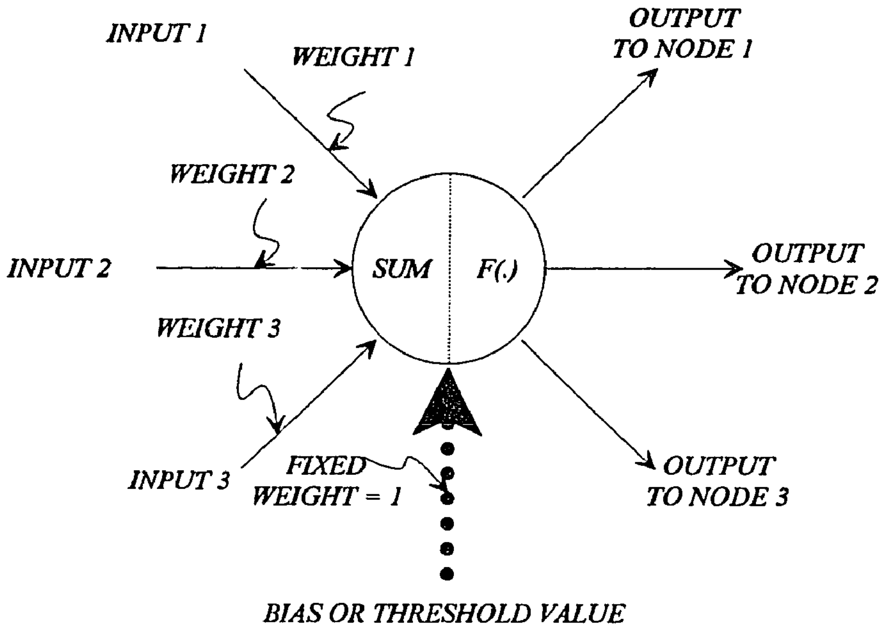

Figure 2.3 A neuron-like processing unit.

Depending on the arrangement of the nodes and the learning algorithm used, researchers have developed different kinds of NNs to perform different tasks, such as system control, pattern recognition, medical diagnosis, and so on. Some of the frequently used NNs are: BPNNs, Hopfield networks, Kohonen's feature map, counterpropagation, etc. [Lippmann 1987, Dayhoff 1990].

Table 2.1 List of some important work in the area of using neural network to control nonlinear systems.

[Barto 1989]	A reinforcement learning control system
[Li and Slotine 1989]	Using BPNNs to model and control nonlinear systems
[Chen 1990]	Using BPNNs for adaptive nonlinear control
[Miller, Sutton, and Werbos 1990]	A collection of NNs for control
[Narendra and Parthasarathy 1990]	Static and dynamic backpropagation methods
[Nguyen and Widrow 1990]	Using Adalines and BPNNs to control problems
[Levin, Gewirtzman, and Inbar 1991]	Using BPNNs for adaptive nonlinear control
[Sanner and Slotine 1991a, 1991b]	Gaussian networks for direct adaptive control
[Tzirkel-Hancock and Fallside 1991a, 1991b]	A direct control scheme for nonlinear systems using radial basis function (RBF) networks
[Polycarpou and Ioannou 1991]	Using a BPNN and a RBF network to model and control nonlinear systems
[Hunt *et al.* 1992]	An excellent survey
[Levin and Narendra 1992a, 1992b]	Controllability and stabilization
[Sanner, Slotine 1992]	Gaussian networks for direct adaptive control
[White and Sofge 1992]	A collection of intelligent control schemes
[Gomi and Kawato 1993]	Using the feedback-error-learning scheme for an NN to adaptive nonlinear feedback control
[Schiffmann and Geffers 1993]	Using a BPNN to adaptive control problems

2.3.2 Neural networks for Control

Many researchers have successfully used NNs to solve control problems that involve nonlinear dynamic systems with unknown parameters. The main reason of this success is that an NN can approximate arbitrary nonlinear functions [Funahashi 1989, Hornik, Stinchcombe, and White 1989, Cybenko 1989, Funahashi and Nakamura 1993, Leshno *et al.* 1993, Bulsari 1993]. Table 2.1 summarizes important work in this area.

Generally, the choice of a specific type of NNs to model and/or control a nonlinear system depends on several factors:

- the characteristics of the nonlinear system
- the availability of the input and output data

- whether the training process will be on-line or off-line
- the accuracy required

The literature survey reveals that the multilayer feedforward neural networks (MFNNs) with the backpropagation learning algorithm (BPNN) appears to be one of the most commonly used neural network schemes for control system design (see, for example, Table 5 in [Hunt *et al.* 1992]). Therefore, we focus on its use in the control problems.

The first step in constructing a BPNN is to determine the configuration of the network, such as the number of layers, the number of nodes in each layer, the connections of nodes between layers, and the type of the activation function for each node in the network. Since there is usually no activation function in the input layer, we commonly classify a network by the number of the hidden layers plus one. For example, a three-layer network consists of one input layer, two hidden layers, and one output layer.

It has been proven that a two-layer feedforward neural network (having one hidden layer) can theoretically approximate any continuous mapping, but only under the following assumptions: there are a *sufficiently* large number of nodes in the hidden layer, and a nonconstant, bounded, continuous, and monotone increasing activation function is used [Funahshi 1989]. However, in practice we may consider adding one more hidden layer to avoid the need for an unlimited number of neurons [Chester 1990]. Therefore, a three-layer BPNN is used as an example to demonstrate the idea of using NNs for control.

The next step is the determination of the number of nodes in each layer. For the purpose of demonstration, one may assume that these design parameters are given. Once these parameters have been determined, we may form the connections between nodes in different layers as shown in figure 2.4. The structure of the BPNN is thereby determined. Figure 2.4 shows a typical three-layer feedforward neural network. The bias signal to each node is neglected in the figure for simplicity.

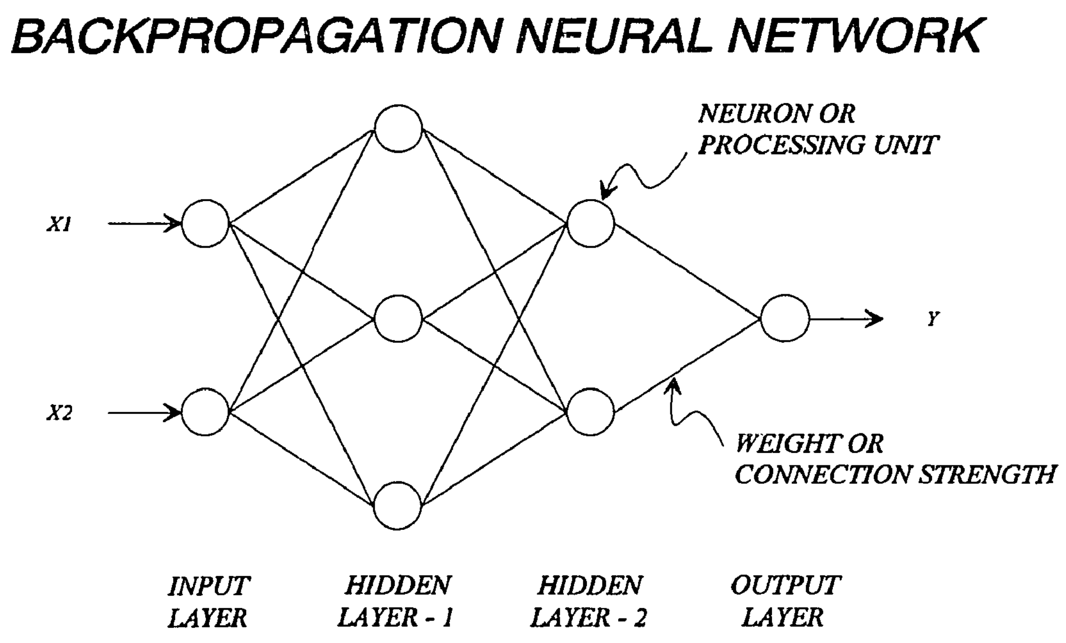
BACKPROPAGATION NEURAL NETWORK
NEURON OR PROCESSING UNIT
X1
X2
Y
WEIGHT OR CONNECTION STRENGTH
INPUT LAYER
HIDDEN LAYER - 1
HIDDEN LAYER - 2
OUTPUT LAYER

Figure 2.4 A three-layer backpropagation neural network.

After the structure of the network is determined, we need to prepare a set of training data that can appropriately represent the characteristic of the control system. This is also a crucial point to the successful use of a BPNN. The training data are usually collected from the test run of the system so that the data can best present the control system.

Then, we have to train the network with the set of prepared training data. The training process is called a supervised learning if the desired value of the system output is known. Usually, the training process can be carried out either on-line or off-line. For the purpose of demonstration, we shall use the off-line training process.

The training procedure contains two phases: the forward pass and the backward pass. In the forward pass, the input signals are forward propagated through layers to produce a set of network outputs. This process can be described by the following equation:

$$\mathbf{Y} = \mathbf{F}^3\left\{\mathbf{W}^3\mathbf{F}^2\left[\mathbf{W}^2\mathbf{F}^1\left(\mathbf{W}^1\mathbf{X}+\mathbf{B}^1\right)+\mathbf{B}^2\right]+\mathbf{B}^3\right\} \tag{2.18}$$

where the vector $\mathbf{Y}$ represents a set of network outputs, the vector $\mathbf{X}$ represents a set of input signals, the vector function $\mathbf{F}^i(\cdot)$ is a set of the node functions or activation functions for nodes in layer i, the matrix $\mathbf{W}^i$ is a set of weights or connection strengths for the nodes between layer i and $i-1$, and the vector $\mathbf{B}$ represents a set of the bias values to the nodes in layer i, for $i = 1,2,3$.

Based on the difference between the desired outputs stored in the training set and the network outputs, an objective function or cost function is defined. The network is trained when the objective function is minimized. Usually the objective function is defined as the square of the sum of the differences between the desired outputs and the network outputs from all the patterns in the training set.

$$E = \sum_{i=1}^{P}\left(\mathbf{Y}_d^i - \mathbf{Y}^i\right)^2 \tag{2.19}$$

where the total error E is a scalar, and the vector $\mathbf{Y}_d^i$ is the desired output of pattern i, for $i = 1$ to P.

To minimize the objective function, the gradient decent method is commonly used in a BPNN. When the generalized delta rule, represented in equation (2.20) or (2.21) below, is used, one can show that the derivative of the error with respect to each weight is proportional to the weight change that should be made to minimize to error.

$$\Delta\mathbf{W} \propto -\frac{\partial \mathrm{E}}{\partial \mathbf{W}} \tag{2.20}$$

or

$$\Delta\mathbf{W} = -\eta\frac{\partial \mathrm{E}}{\partial \mathbf{W}} \tag{2.21}$$

where η is the learning rate.

The reader is referred to [Rumelhart, McClelland, and the PDP Research Group 1986] for further details.

There are two types of applications in the adaptive control: the direct adaptive control approach and the indirect adaptive control approach. In the direct adaptive control, the error signals of the system is usually employed to train the NN to produce the proper control actions [Venugopal, Sudhakar, and Pandya 1994, Sanner and Soltine 1992, Hunt, *et al.* 1992]. Figure 2.5 shows the basic structure of a direct adaptive control, where the error signals modify parameters of the NN controller. This approach is more feasible for real-time control problems.

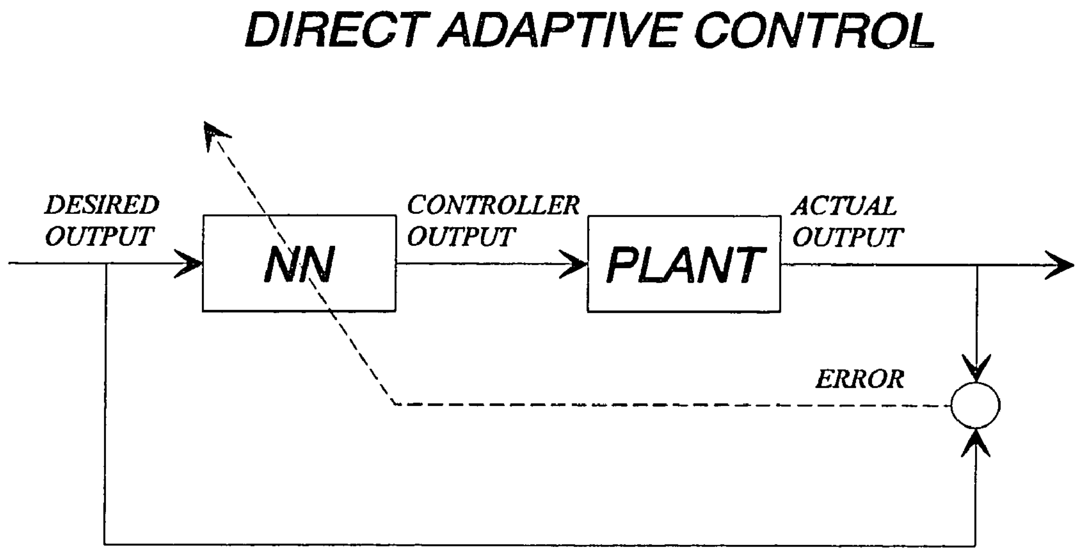

Figure 2.5 Direct adaptive control scheme.

In the indirect adaptive control, usually two NNs are used: one for the identification model and the other for controller [Narendra and Parthasarathy 1990, Hunt, *et al.* 1992]. When the NN serves as the identification model, it learns the mapping of the input-output relation. The estimation error is used to tune the weights in the NN model. In the case of using an NN as the controller, the system error, similar to the direct adaptive control approach, is used to modify the weights of the NN controller. Figure 2.6 shows the block diagrams of the indirect adaptive control scheme.

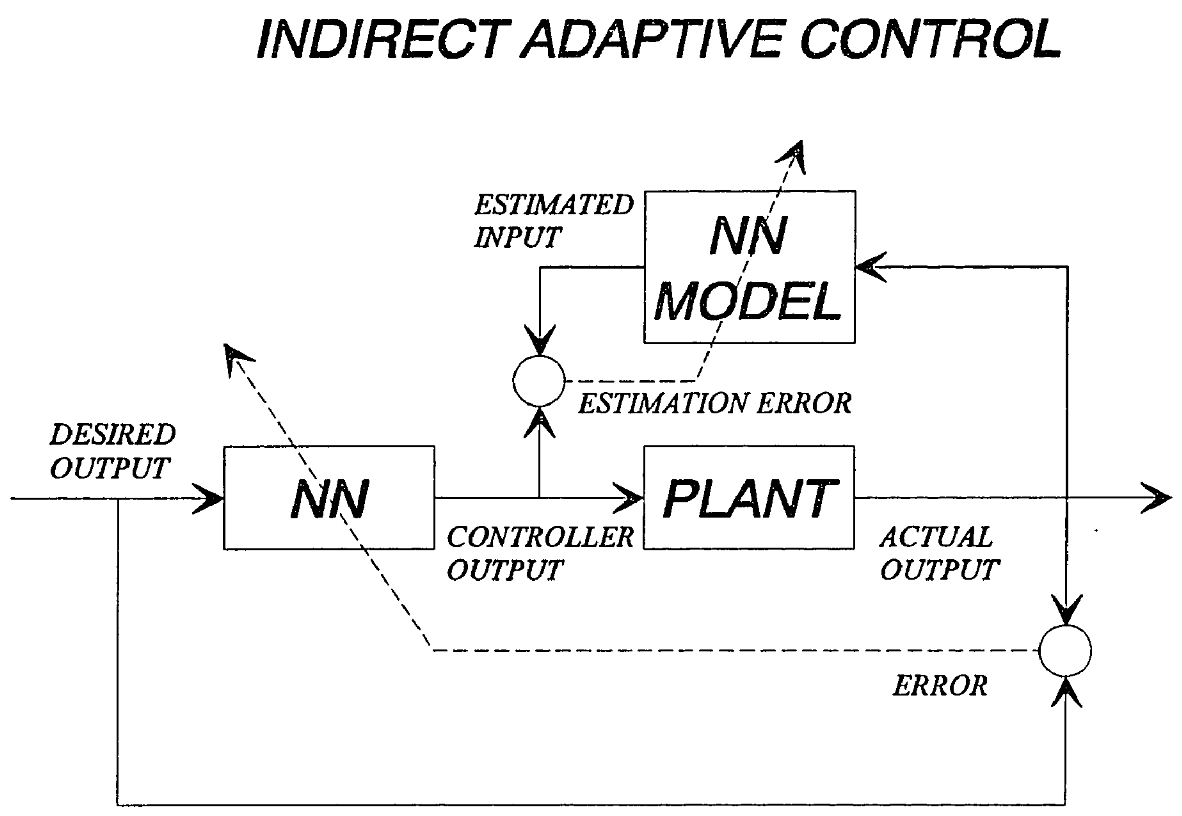

Figure 2.6 Indirect adaptive control scheme.

2.3.3 Practical Issues

Main reasons for a tremendous growth of papers in the area of applying NNs for control problems is due to; 1) their ability to approximate nonlinear mappings, 2) their ability for learning and adaptation, 3) a highly parallel structure that is suitable for parallel hardware implementation, and 4) an ability to process many inputs and outputs [Hunt *et al.* 1992]. Despite this outgrowth and successful applications, there are several problems with applying NNs to control problems. First, the procedures for the development of the network architecture are not systematic. Second, the relationship between the system that is being modeled and the specified structure of the NN used is not clear [Hunt *et al.*

1992]. While an FLC is ideal for a structured representation of knowledge, an NN appears more or less like a black box to a practicing engineer. Third, the common problem with BPNN type networks is the need for a large amount of learning cycles, for example it usually takes over 50,000 cycles for learning in an identification problem considered later in chapter 3 [Narendra and Parthsarathy 1990]. Fourth, although there is an exhaustive amount of literature reporting proofs of stability of adaptive systems using Lyapunov methods, in almost all the cases the underlying plant is assumed to be linear and time-invariant [Hunt *et al.* 1992]. There is a need for extending this work for nonlinear systems [Hunt *et al.* 1992, Narendra and Parthsarathy 1990].

Despite the problems listed above, the success of the NNs in control area is indisputable. It is clear that issues related to stability of these methods will require further investigation, but lack of reliable techniques for stability analysis has not been detrimental to development of a large number of practical applications. In terms of the first three problems listed above, the most practical approach appears to be merging of NNs with techniques like FLC. In the next section, methods that borrow learning concepts from NNs into FLC are reviewed.

2.4 Combined Approach of Fuzzy Logic and Neural Networks

2.4.1 General Introduction

As discussed in the previous two sections, there are problems in using FLC or NNs alone in control problems. Therefore, recent research has shown a new trend of combining FL and NNs to provide an alternative to using either of these modern techniques alone. The discussion here will focus on the schemes that incorporate the learning concept in an NN to an FLC.

Essentially, there are three possibilities in combining these two approaches. First, one may either employ an NN or NNs to play the main role and apply an FLC or FLCs as an auxiliary, or vice versa [Berenji and Khedkar 1992, Lin and Lee 1994]. Second, one may either use an NN to help model the mapping of an FLC, or vice versa [Wang and Kim 1994]. Finally, third, one may simply *fuse* them together and obtain a hybrid architecture. Here, by *fuse* we mean merging the concept of learning from NNs and ability of approximate reasoning from FL [Jang 1993].

The first two approaches may improve the overall performance of the system in a certain sense; however, they still may have the same problems that one faces when using an FLC or an NN alone. In the third case, the problems caused by using FL and NNs alone may be avoided if the topology of the network is properly arranged. It is the main objective of this research to develop a systematic procedure for nonlinear control problems, and a hybrid scheme that merges these two techniques together may be a better alternative. Later in this chapter, this will be discussed in more detail.

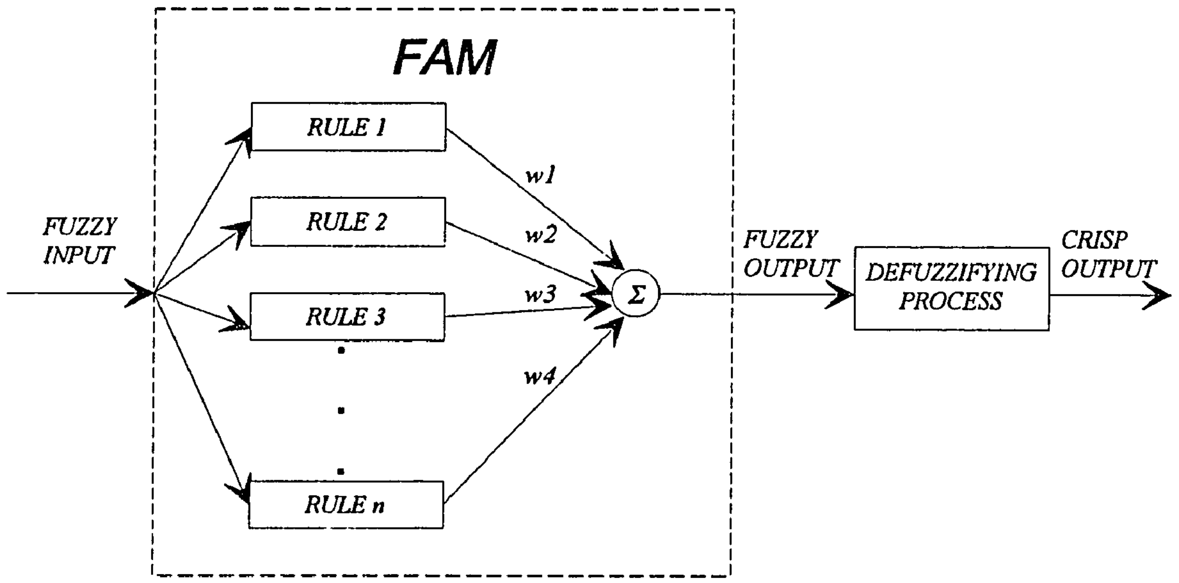

Figure 2.7 A fuzzy associative memory system.

Before the hybrid scheme is developed, we will briefly review some pioneering work in this area. Kosko developed a scheme, called FAM, which maps fuzzy sets to fuzzy sets [Kosko 1992a]. Figure 2.7 shows the architecture of the FAM with one fuzzy input and one fuzzy output. The FAM system consists of a set of FAM rules and a set of weights associated with the FAM rules, which is quite similar to a FMNN. The weights are modified by feeding the system with training data, calculating the firing frequency of each rule, and then determining the weight modification of each rule by comparing the firing frequency of each rule with a prescribed threshold value. The result of this learning process allows one to determine a set of weights associated with the rules in FAM so as to produce an optimal association of a fuzzy output to a fuzzy input. However, there are no changes in membership functions in both premise and consequent parts of the fuzzy rules. That is, this scheme only learns the fuzzy rules, and requires the fixed membership functions to be determined before the training process starts. In other words, expert knowledge to set up the membership functions is still required. The main advantage of the FAM is that it provides a way in which the number of rules required may be automatically determined.

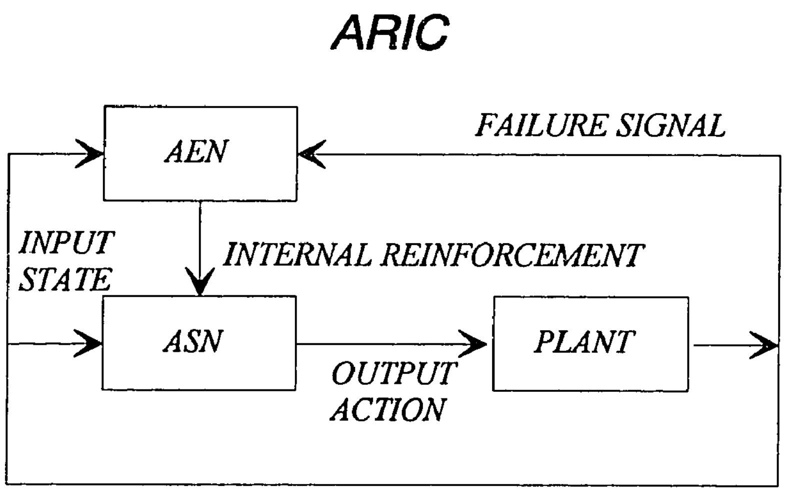

Figure 2.8 Basic structure of the ARIC architecture.

Berenji introduced the ARIC and its generalization GARIC [Berenji 1992, Berenji and Khedkar 1992]. These schemes consist of two main elements: the action-state evaluation network (AEN) and the action selection network (ASN), as shown in Figure 2.8. The AEN is an NN used to produce a prediction of the future reinforcement based on a given state; while ASN is also an NN that models an FLC and generates the control actions. In the ASN, an FLC is modeled by a two-layer neural network with the hidden layer containing as many nodes as the number of the rules. Thus the structure of ASN can be easily determined. However, in the AEN, the determination of the network configuration poses problems similar to those in constructing NNs alone for control. The weights in the AEN and ASN can be modified by the reinforcement leaning algorithm. Since there is no input and output training data used in ARIC, it employs unsupervised learning.

In addition to ASN and AEN in ARIC, GARIC has one more component, called stochastic action modifier (SAM), that stochastically generates the control action and sends it to the system. GARIC also provides an algorithm to tune the fuzzy labels globally in all rules, instead of tuning them locally in ARIC.

Although ARIC and GARIC have been reported as effective tools to control nonlinear dynamic systems, the main problem in practical implementation is the need to determine the architecture of ASE, similar to the problem in applying a BPNN.

Jang proposed the Adaptive Network-based Fuzzy Inference System (ANFIS) which can determine the near-optimal membership functions and the control rules [Jang 1993]. The ANFIS is essentially a special case of a MFNN named adaptive network. However, unlike the regular MFNNs, each node in the adaptive network can have different node function. In this approach, Jang used different node functions in different layers to replace the processes of an FLC in different phases.

Figure 2.9 shows the basic structure of an ANFIS. The circle represents the node containing modifiable parameters; while the square represents the node that has no modifiable parameters. Since ANFIS employs supervised learning algorithm, input and output data are required. Jang also developed a hybrid training procedure, which is claimed to be an improvement of the basic gradient descent method. The training process is composed of two phases. In the forward training pass, the input signal passes through layers, like in an FLC, and produces the network output. The network output is then compared with the desired output stored in the training data. In the backward training pass, the difference of the comparison, which is the training error, is used as a reference to modify the parameters in the system. The entire training process repeats until it reaches the prescribed terminative condition.

Jang's work provides a good approach for solving the problem of parameter identification for an FLC. However, as the number of inputs and the number of membership functions for each input increase, the size of the network also grows exponentially. ANFIS should include a structure identification process in order to reduce the size of the network [Sun 1994].

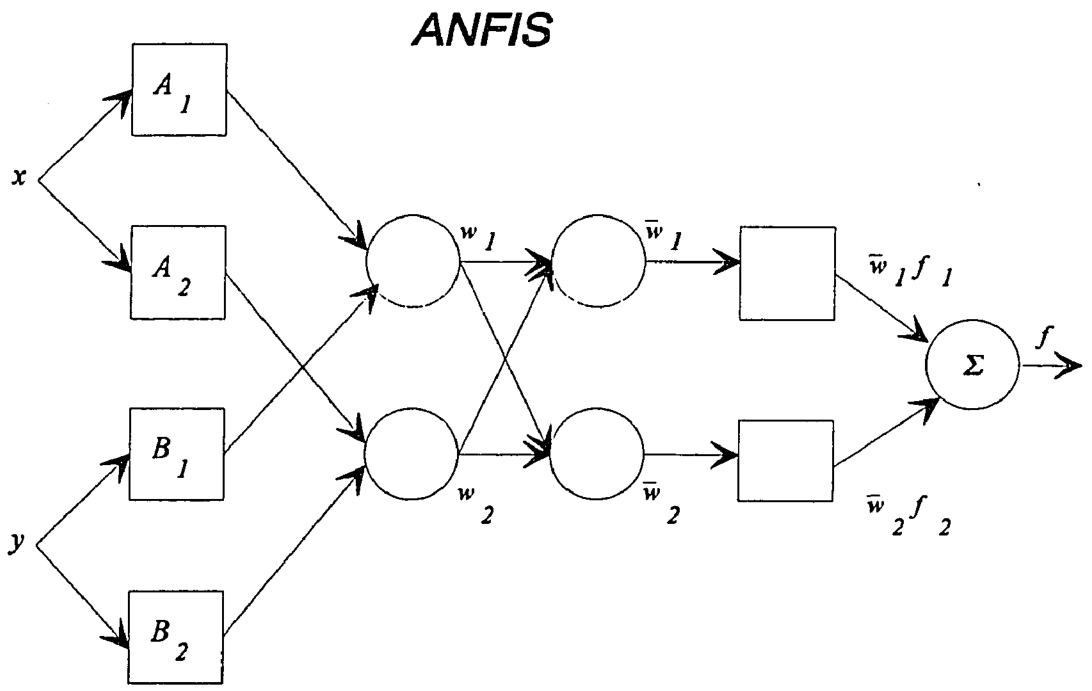

Figure 2.9 An ANFIS architecture.

In summary, it is apparent that all the methods discussed above have certain good features. However, each also has certain weak points. It is believed that combination of various ideas from some of these techniques may be desirable. Essentially, the hybrid scheme should have the following properties:

- provide a general guideline to determine the hybrid architecture
- determine the parameters of the hybrid system automatically

In other words, the ultimate goal of developing the hybrid scheme is to automatically identify the parameters and the structure of the hybrid scheme in a sort of optimal sense so that the hybrid scheme can accurately approximate complex nonlinear mappings. In the next section, such a scheme is developed.

2.4.2 Proposed Hybrid Scheme: FLAN

As mentioned before, our main interest is in incorporating the learning process in the design of an FLC so as to determine the parameters and the structure of the FLC automatically, and thus provide a convenient tool for solving practical problems. As for the parameter identification, one effective approach is to use the MFNN architecture to present an FLC. The architecture of MFNN can be determined simply by the initial number of membership functions and initial number of linguistic control rules required. Once the structure of the network is set, the parameters of an FLC can be tuned by the learning algorithms used in NNs. While the structure identification is done by setting a set of additional parameters associated with the linguistic control rules and calculating the firing frequency of each linguistic control rule so that the rules with low firing frequencies can be removed from the hybrid system.

The following discussion on developing the proposed hybrid scheme consists of two parts: the structure identification and the parameter identification. Figure 2.10 shows the flow chart illustrating this process. The first part, which is the structure identification, includes the determination of the number of initial membership functions and the adjustment of the number of final membership functions and the number of final linguist control rules. The number of initial membership functions can be determined by analyzing the distribution of the given input patterns of each input variable within its definition interval. A membership function can be removed if there are only a few data falling in that interval. As a consequence of removing a membership function, a set of rules related to this membership function is also removed. In the case of a two-input system, for example, the linguistic control rules can be represented as a look-up table. Removing a membership function from an input variable is equivalent to deleting an entire column or row from the look-up table. At the end of this process, one is left with the initial set of linguistic control rules. Thus the number of initial linguistic control rules is determined by the number of membership functions of each input.

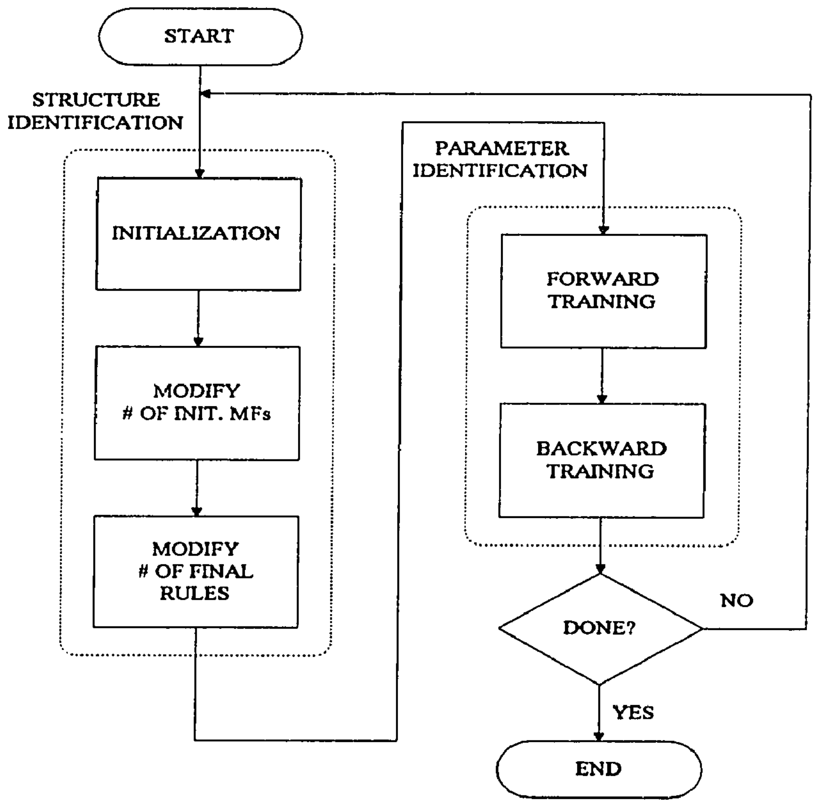

Figure 2.10 Flowchart of the proposed hybrid scheme.

The number of final linguistic control rules is determined by counting the firing frequency weighted by the firing strength of each rule. A rule should be removed if the cumulative weighted frequency is low.

The second part, which is the parameter identification, performs the adjustments of the premise parameters and consequent parameters in an FLC. The parameter identification part is based on learning concepts from the MFNN. Therefore the steps in the parameter identification include steps similar to the design of an MFNN. All the steps in that part are discussed below.

The first step is to determine the architecture, namely, the number of layers and the number of nodes in each layers required. The hybrid architecture consists of four layers: the input layer that receives the input signals from the sensor, the fuzzifying layer, which performs the fuzzification process, the inferring layer, which contains the fuzzy reasoning mechanism, and the defuzzifying layer, which produces the crisp output signals. The number of nodes required in the input layer depends on the number of signals received by the hybrid system. In the same manner, the number of nodes needed for the defuzzifing layer is decided by the physical plant. The number of the nodes in the fuzzifying layer is equal to the sum of the number of membership functions used over the number of input signals; and the number of nodes required in the inferring layer is determined by the number of rules required, which, in turn, is decided by the number of membership functions used. Therefore, the only design factor required in this scheme is the number of membership functions to be used for each input variable.

The second step is to connect the nodes in different layers. The configuration and the connection of the hybrid architecture with two input signals and one output signal is shown in Figure 2.11.

The third step is to develop the training process that learns the parameters in the proposed hybrid scheme, which are the premise parameters in the fuzzifying layer, the consequent parameters in the inferring layer, and the weights in the defuzzifying layer. In the case of controlling a dynamic system, the input and output data are usually known or measurable. Therefore, a supervised learning algorithm may be used in this case. To train the hybrid scheme with the supervised learning algorithm, we first present the input data to the hybrid system and produce a set of corresponding outputs, and next calculate the difference between this output values with the desired ones. Like in a BPNN, the generalized delta rule [Rumelhart, McClelland, and the PDP Research Group 1986] may be used to modify the parameters of the system so as to achieve a desired input-output mapping.

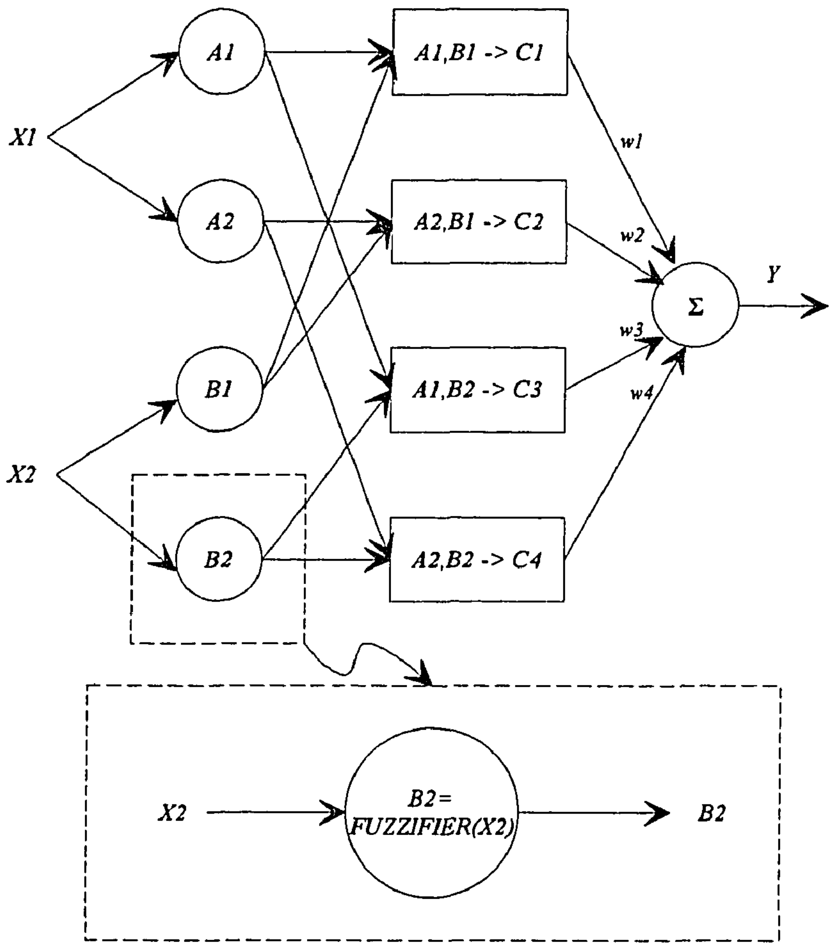

Figure 2.11 Basic structure of the proposed hybrid scheme - FLAN.

The network described above can be thought of as a fuzzy logic adaptive network. Hence the term FLAN is used to describe this scheme. The training process in FLAN has two phases: the forward training phase and the backward training phase. To train the hybrid system, both the batch learning and the pattern learning can be used. To simplify the notations, we may just use the pattern learning approach.

In the forward training phase, a set of input data is first presented to the hybrid system to obtain a corresponding output value. In the fuzzifying layer, the input data are fuzzified to fuzzy values based on the corresponding membership functions.

$$\mathbf{O}^1 = \mathbf{FUZZIFY}(\mathbf{X}) \tag{2.22}$$

where the vector function $\mathbf{FUZZIFY}(\cdot)$ denotes the fuzzification process that maps a crisp input vector $\mathbf{X}$ to the corresponding fuzzy membership denoted by the vector $\mathbf{O}^1$. The vector function $\mathbf{FUZZIFY}(\cdot)$ is a collection of fuzzy membership functions which are used as the node functions for the nodes in the fuzzifying layer. These node functions are usually bounded, piecewise continuous and differentiable. Typical examples are triangular, trapezoidal and bell-shaped membership functions. For example, a bell-shaped membership function is determined by a set of three parameters. Thus if we change the values of these three parameters, we may change the shape and/or the location of the membership function.

In the inferring layer, each node represents a linguistic rule which associates the output to the inputs, denoting as $(A \text{ and } B \rightarrow C)$. Since the fuzzified input values activate each linguistic control rule to a different degree, we may calculate the firing strength of a rule using the approach introduced in subsection 2.2.2. Applying the similar procedure developed in that subsection,

$$\mathbf{O}^2 = \mathbf{INFER}(\mathbf{O}^1) \tag{2.23}$$

where the vector function $\mathbf{INFER}(\cdot)$ denotes the inferring process that produces proper values at the consequent parts based on the degrees of satisfaction at the premise parts of the control rules, and the vector $\mathbf{O}^2$ is a set of the firing strengths of the linguistic control rules.

In the defuzzifying layer, the fuzzy output equals a weighted sum of the individual consequences. We then may calculate the nonfuzzy output by one of the defuzzification method mentioned in subsection 2.2.2, for example, center of the gravity method.

$$\mathbf{Y} = \mathbf{O}^3 = \mathbf{DEFUZZIFY}(\mathbf{W}^T \cdot \mathbf{O}^2) \tag{2.24}$$

where the vector $\mathbf{Y}$ represents a set of nonfuzzy outputs, which is the output of the defuzzifying layer $\mathbf{O}^3$, the vector function $\mathbf{DEFUZZIFY}(\cdot)$ denotes the defuzzification process that maps the set of fuzzy outputs to a set of nonfuzzy values in the universes of discourse, and the vector $\mathbf{W}$ is a set of adaptation weights associated with the linguistic control rules. From equation (2.24) we may learn that the fuzzy outputs are additively combined with the sum operators; therefore, this fuzzy system is essentially an additive fuzzy system. In addition, the fuzzy system becomes a conventional fuzzy system when all the elements of the vector $\mathbf{W}$ are ones.

In the backward training phase, a procedure similar to the one developed for a BPNN may be used. First of all, an objective function has to be defined. In fact, it can be defined in a similar way as used in the learning process for a BPNN.

$$\begin{aligned} E &= \sum_{i=1}^{P} \frac{1}{2} (\mathbf{e}^i)^T \mathbf{e}^i \\ &= \sum_{i=1}^{P} \frac{1}{2} (\mathbf{Y}_d^i - \mathbf{Y}^i)^T (\mathbf{Y}_d^i - \mathbf{Y}^i) \end{aligned} \tag{2.25}$$

where P is the number of training patterns, the vector $\mathbf{e}^i = (\mathbf{Y}_d^i - \mathbf{Y}^i)$ denotes a set of errors between the desired outputs and the actual outputs.

Second, the modifiable parameters need to be identified. There are three sets of parameters needed to be trained so as to produce the proper mapping that represents the input-output relation of the training data: one is the set of parameters associated with the membership functions at the consequent side of the linguistic control rules, second is the set of parameters associated with the membership functions at the premise side of the linguistic control rules, and the third is the set of weights associated with the linguistic control rules.

To modify these parameters, the generalized delta rule used in a BPNN may be utilized to determine the amount of change that should be made for a given modifiable parameter.

$$\Delta p_{ji}^{l} = \eta \delta_{j}^{l} x_{i}^{l-1} \tag{2.26}$$

where Δp_{ji}^{l} is the change in the parameter that should be made, η, as in equation (2.21), is the learning rate, δ_{j}^{l} is the error between the desired output and the actual output produced by the given input pattern at the current node j in layer l that contains the parameter under consideration, and $x_{i}^{l-1}$is the input from node i in the preceding layer l-1 to current node j in layer l. Since a delta rule implements a gradient descent in E, we can produce the following from equation (2.21) and equation (2.26).

$$\frac{\partial E}{\partial p_{ji}^{l}} = -\delta_{j}^{l} x_{i}^{l-1} \tag{2.27}$$

Because the change of a given parameter p_{ji}^{l} only affects the nodes that contain this parameter, and the output of the current node j is a function of the given parameter, the chain rule can be used to represent the partial derivative as the product of two parts:

$$\frac{\partial E}{\partial p_{ji}^{l}} = \frac{\partial E}{\partial O_{j}^{l}} \frac{\partial O_{j}^{l}}{\partial p_{ji}^{l}} \tag{2.28}$$

where O_{j}^{l} is the output of node j in layer l. The first part represents the change in error as a function of the change of the node output. If the node is in the defuzzifying layer, we may use equation (2.28) directly. When the node is in the hidden layer, we, again, use the chain rule to derive this part. The second part of equation (2.28) is the change in node function with respect to the change in the given parameter.

The updated parameters are then used for the next training iteration. The training process is repeated until the objective function reaches a prescribed value or a certain number of iterations. At this moment, the system with the trained parameters can represent the nonlinear mapping to a certain accuracy.

In this algorithm, we are free to choose the number of membership functions for each input, which is the only design factor that determines the whole architecture of the scheme. The variation of the scheme can be made by using different fuzzy reasoning mechanisms or different training rules. In the present analysis, bell shaped membership functions are used for the premise part and Sugeno type fuzzy rules [Takagi and Sugeno 1985] are used for the consequent part. Appendix A presents the derivations in more detail.

In the next two chapters, use of FLAN in identification and control of unknown nonlinear systems is considered.

CHAPTER 3

IDENTIFICATION OF NONLINEAR DYNAMIC SYSTEMS WITH UNKNOWN PARAMETERS

3.1 Introduction

The major objective of an identification model is to represent the input-output relationship of the unknown dynamic system based on the input and output signal measurements. The input signals may be either some known functions commonly used for the purpose of identification, or in some form that is tractable; while the output signals are usually the readings from the sensors or some function of the sensor readings. A mathematical equation, either differential or difference equation, is usually used to express the relationship based on observed input and output signal measurements.

Conventional approaches assume that some prior information concerning the system is available, and the mathematical equation of the model is completely known except for a finite set of parameters. However, this is not always true in the cases of identifying nonlinear dynamic systems because some systems are just too complex to be expressed by a mathematical equation. Without the mathematical model, the conventional approaches fail because most of them are based on the certainty equivalence principle [Åström and Wittenmark 1973, 1989] which starts with solving the system with known parameters, and then uses the recursively estimated parameters to generate control gains for the system with unknown parameters. In other words, if there is no analytical solution in closed form available to the problem of identifying a nonlinear system, then other approaches are needed. One may use a linear estimation scheme for the nonlinear problems, which in turn results in a poor estimation because the linear estimator just cannot catch the nature of the nonlinearity of the function to be identified. Alternatively, one may use stochastic processes, for example, from a Bayesian point of view, to estimate the unknown nonlinear functions.

In many mechanical engineering applications, a great effort has been devoted to improve the understanding of dynamics of nonlinear systems so as to obtain a better control quality. However, in the literature, most mathematical models used are either based on linear system theory or obtained from experience and/or experiments. For example, in robotics, because the arm dynamics is highly nonlinear, a mathematical model of the robot arm is usually obtained by making simplifyied assumptions. On the other hand, as reported in [Kim and Singh 1993], instead of using a linear model, the authors developed an experimental setup to measure the parameters of the mathematical model for a hydraulic engine mount, which can be considered as a complex nonlinear dynamic system. This is the type of problems where the FLAN can fit in. Although the example in [Kim and Singh 1993] is not used here, a more general framework is utilized to demonstrate the potential of the FLAN.

This chapter concentrates on the identification problem of nonlinear dynamic systems with unknown parameters. Section 3.2 provides the necessary mathematical and conceptual background concerning the subject. Section 3.3 devotes to the use of the proposed hybrid scheme to the identification of nonlinear dynamic systems with unknown parameters. Simulations and results are included. Section 3.4 presents brief concluding remarks on the applicability of the FLAN to the identification of nonlinear dynamic systems with unknown parameters.

3.2 Identification Models

Identification is the process of assigning a set of values, called the estimates, to a set of parameters of the mathematical model regarding the system under testing, based on the measurements or observations of the input and output signals [Goodwin and Sin 1984]. In order to measure the accuracy of the identification, it is meaningful to define a cost function, or sometimes an objective function, which should be a function of the estimation error. An identification or estimation is called *optimal* if the assignment of an estimate minimizes some objective function.

Now, let us consider a discrete-time nonlinear dynamic system described by

$$\begin{aligned} \mathbf{x}(k+1) &= \Phi[\mathbf{x}(k), \mathbf{u}(k), k], \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y}(k) &= \Psi[\mathbf{x}(k)] \end{aligned} \tag{3.1}$$

where $\Phi[.]$ and $\Psi[.]$ are the unknown nonlinear functions, $\mathbf{x} \in \Re^m, \mathbf{u} \in \Re^n, \mathbf{y} \in \Re^p$ are the state, input and output, respectively. The objective of the identification of (3.1) is to replace the unknowns with equivalent functions which we are familiar with.

$$\begin{aligned} \mathbf{x}_m(k+1) &= \Phi_m[\mathbf{x}_m(k), \mathbf{u}(k), k], \quad \mathbf{x}_m(0) = \mathbf{x}_{m0} \\ \mathbf{y}_m(k) &= \Psi_m[\mathbf{x}_m(k)] \end{aligned} \tag{3.2}$$

where $\Phi_m[.]$ and $\Psi_m[.]$ are the estimation models, $\mathbf{x}_m \in \Re^m, \mathbf{y}_m \in \Re^p$ are the estimated state, and estimated output respectively.

Now, the problem of identification of the unknown functions in (3.1) can be defined as the determination, according to the input and output data, of the model represented by $\Phi_m[.]$ and $\Psi_m[.]$ within a set of particular models so that it is equivalent or close, in most nonlinear case, to the actual system represented by $\Phi[.]$ and $\Psi[.]$. In other words, the difference between the estimated output from the identification model and the actual output from the nonlinear dynamic system should be as small as possible.

The problem may be attacked from two different approaches: the *parallel identification model*, in which the plant and the identification model are independent of each other as shown in figure 3.1(a), and the *series-parallel model*, in which the output of the plant is fed back to the identification model as shown in figure 3.1(b) [Narendra and Annaswamy 1989]. Since the series-parallel identification is more preferable to generate stable adaptive laws [Narendra and Parthasarathy 1990], we will use this approach for the rest of the development of this research.

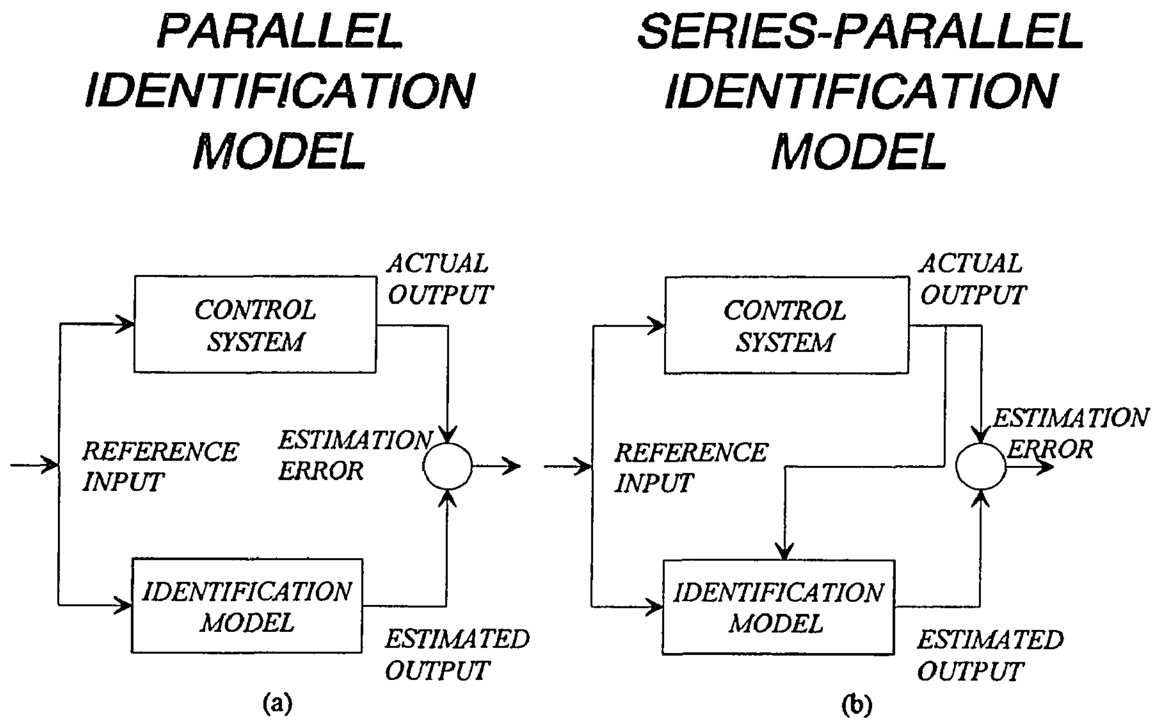

Figure 3.1 Parallel identification model and series-parallel identification model.

Depending on the interdependency between the state and the input, we can further characterize the discrete-time nonlinear dynamic system described by equation (3.1) into the following types [Narendra and Parthasarathy 1990]:

Type 1: The dependence (of the present output) on the past output value $\mathbf{y}(k-i)$ (where $i = 0,1,\cdots,n-1$) is linear while the dependence on the past input values $u(k-j)$ (where $j = 0,1,\cdots,m-1$) is nonlinear.

$$\mathbf{y}(k+1) = \sum_{i=0}^{n-1} \mathbf{a}_i \mathbf{y}(k-i) + g[\mathbf{u}(k), \mathbf{u}(k-1), \cdots, \mathbf{u}(k-m+1)] \quad (3.3)$$

where $\mathbf{a}_i$ $(i = 0,1,\cdots,m-1)$ are the unknown coefficients.

Type 2: The dependence on the past output value $\mathbf{y}(k-i)$ (where $i=0,1,\cdots,n-1$) is nonlinear while the dependence on the past input values $u(k-j)$ (where $j=0,1,\cdots,m-1$) is linear.

$$\mathbf{y}(k+1)=f[\mathbf{y}(k),\mathbf{y}(k-1),\cdots,\mathbf{y}(k-n+1)]+\sum_{j=0}^{m-1}\mathbf{b}_j\mathbf{u}(k-j) \tag{3.4}$$

where $\mathbf{b}_j$ $(j=0,1,\cdots,m-1)$ are the unknown coefficients.

Type 3: The nonlinear dependence on the past output value $\mathbf{y}(k-i)$ (where $i=0,1,\cdots,n-1$) and the past input values $u(k-j)$ (where $j=0,1,\cdots,m-1$) is separable.

$$\mathbf{y}(k+1)=f[\mathbf{y}(k),\mathbf{y}(k-1),\cdots,\mathbf{y}(k-n+1)]+g[\mathbf{u}(k),\mathbf{u}(k-1),\cdots,\mathbf{u}(k-m+1)] \tag{3.5}$$

Type 4: The nonlinear dependence on the past output value $\mathbf{y}(k-i)$ (where $i=0,1,\cdots,n-1$) and the past input values $u(k-j)$ (where $j=0,1,\cdots,m-1$) is not separable.

$$\mathbf{y}(k+1)=f[\mathbf{y}(k),\mathbf{y}(k-1),\cdots,\mathbf{y}(k-n+1),\mathbf{u}(k),\mathbf{u}(k-1),\cdots,\mathbf{u}(k-m+1)] \tag{3.6}$$

Type 4 system is the most general form of nonlinear systems because in practice one usually does not have prior information about the control system. In using type 1, 2, and 3 systems, minor assumptions need to be made about the system or some prior information about the system must be available. Type 1 and 2 systems are more preferable in most applications because they are more analyzable. In practical applications, the type of identification model to be used is determined based on the characteristics of the system under consideration. Since Type 1 and 2 are similar from the identification point of view, only one case is considered for a simulation study in the next section. Type 3 is also considered to illustrate how FLAN can be applied for such problems.

3.3 Simulation Studies

In the following simulation studies, for the sake of simulations the equation of the control system is completely specified so that the identification model according to the classification of the systems described in the previous section can be determined.

3.3.1 Case 1: Identification of a Type 1 Nonlinear System

First, we consider the example used in [Narendra and Parthasarathy 1990, Jang 1993], where a 1-20-10-1 BPNN [Narendra and Parthasarathy 1990] and an ANFIS with seven rules [Jang 1993] are used to identify the unknown nonlinear function of input in a control system. The plant is described by the following difference equation, which is a type 1 nonlinear system in equation (3.3):

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f[u(k)] \tag{3.7}$$

where $y(\cdot)$ and $u(\cdot)$ are the output and input of the system respectively, $f(\cdot)$ represents the unknown nonlinear function of the input with the following form:

$$f(u) = 0.3\sin(\pi u) + 0.6\sin(3\pi u) + 0.5\sin(5\pi u) \tag{3.8}$$

We may discuss the identification of the nonlinear system described by (3.7) in two phases, which are the identification of the mapping from *u(k)* to *f[u(k)]* and then the identification of the mapping from *y(k)*, *y(k-1)*, and *u(k)* to the estimated output $\hat{y}(k+1)$. The reason will become clear later.

First, since the plant under control is type 1 nonlinear system, we may consider the following series-parallel model for identification:

$$\hat{y}(k+1) = 0.3y(k) + 0.6y(k-1) + FLAN[u(k)] \tag{3.9}$$

where $\hat{y}(\cdot)$ represents the estimated output of the plant under consideration, and $FLAN(\cdot)$ denotes a function implemented by the FLAN to approximate the unknown function $f(\cdot)$.

To identify the unknown function $f(\cdot)$ by $FLAN(\cdot)$, we need the measurements of the input *u(k)* and *f[u(k)]*. A number of data are collected as the training data, which can best represent the characteristic of the unknown function. For the purpose of simulation, the input to the plant is given by the following equation:

$$u(k) = \begin{cases} \sin(2\pi k / 250) & 1 \le k \le 250 \text{ and } 501 \le k \le 700 \\ 0.5\sin(2\pi k / 250) + 0.5\sin(2\pi k / 25) & 251 \le k \le 500 \end{cases} \tag{3.10}$$

A change of the input to the system is introduced at the time steps between 251 and 500. The reason is to demonstrate if the identification model can tolerate the change of the input which is not represented in the training data. Figure 3.2 shows the given input at the time steps from 0 to 700.

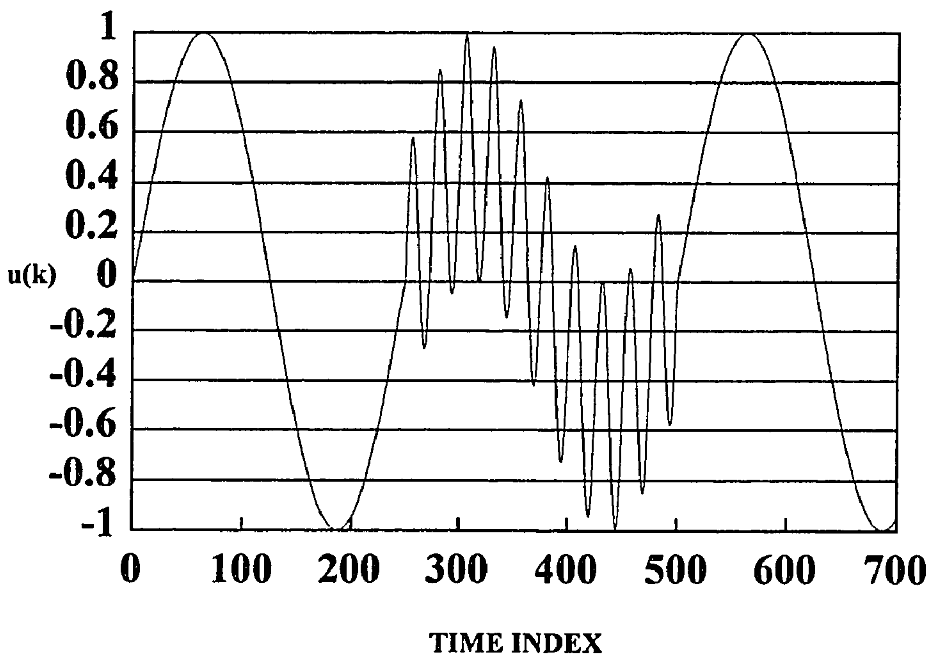

Figure 3.2 Input to the nonlinear system (for case 1).

In this case, we employ the FLAN to estimate the unknown function with one input and one output. The hybrid architecture is now determined by the number of the membership functions used. Three, five and seven are commonly used numbers. Suppose we start with three membership functions for the input. Then the hybrid architecture consists of one node in the input layer, three nodes in the fuzzifying layer, three nodes in the inferring layer, and one node in the defuzzifying layer. If Sugeno type fuzzy reasoning method is used, the linguistic control rule has the following form:

$$R_i\text{: } If\ u(k)\ is\ A_i,\ then\ f_i = p_i * u(k) + q_i; \tag{3.11}$$

where R_i denotes the i-th linguistic control rule, A_i is the linguistic label, f_i is the estimated output from i-th linguistic control rule, and p_i and q_i are the modifiable parameters of the consequent part. The configuration of the hybrid architecture with three membership functions is shown in table 3.1.

Table 3.1 The configuration of the FLAN with three membership functions.

	Number of Nodes	Modifiable Parameters
Input layer	1	0
fuzzifying layer	3	9
inferring layer	3	6
defuzzifying layer	1	3

There are two important factors that affect the training process: namely, the preparation of the training data and when to terminate the training process. Depending on the characteristic of the nonlinear function, we may select the number of training data that can best fit the function under estimating. In this study, 125 training data was sampled from the first 250 entities of the input and the output measurements. The time to end the training process, such as in training a BPNN, is usually determined by the training time available or whether the value of the objective function reaches the prescribed value or not. We used fifty training epochs in all the following training processes.

Since the parameters in the hybrid system are not known in advance, we initialize the parameters as follows. The parameters in the membership functions can be estimated be equally partitioning the input space depending on the number of membership functions used. The parameters in the consequent part can be set to zeros. The weights associated with each rules are set to ones. It is emphasized here that unlike a BPNN where the initialization is usually random, a consistent scheme described above is used for initializing FLAN. When the initialization is random as in BPNN, one needs to perform Monte Carlo type study to validate the identified model. For FLAN on the other hand, since the same kind of initialization is always used, the question of dependency on a particular random initialization is avoided. Needless to say, as the training process moves on, these parameters are properly adjusted so as to represent the nonlinear mapping.

There is no need for structure identification in this case because there is only one input variable and the training data are equally distributed in the input space. As mentioned in the previous chapter, the training process consists of two phases. The forward pass of the training process includes the estimation of the parameters at the consequent parts and the calculation of the estimated output, as explained in the Appendix A. While the backward pass of the training process involves the updating operations of the parameters in the fuzzifying and defuzzifying layer. The selection of proper learning rate is problem-dependent, and is also an important factor to the success of training process. A larger learning rate helps in speeding up the convergence but causes oscillations in the estimation if it becomes too large. Nevertheless, a smaller learning rate helps in reducing the oscillations but takes longer time to rearch the point of convergence. Here, Jang's adaptive learning rate scheme is employed because it has the positive effect of improving the speed of convergence (see page 676 of [Jang 1993]).

The learning scheme utilized by FLAN also has another feature. A threshold based on the root mean square error (RMSE) is prescribed to separate the updating operations into two stages. If the RMSE is smaller than the prescribed value, only the parameters in the fuzzifying layer are updated. Otherwise, both the parameters in the fuzzifying and defuzzifying layersare updated. In this example, the training procedure is repeated fifty times. Table 3.2 shows the results, in terms of the root mean square errors (RMSE), of the training process with different numbers of membership functions.

Table 3.2 The RMSEs resulting from different numbers of membership functions.

Number of Membership Functions	RMSE	Max. Error
3	0.29	0.742
5	0.09	0.250
7	0.0086	0.021

From the above table, it is observed that more the number of membership functions used, smaller the RMSE and the maximum error. Therefore, we may conclude that the determination of the number of membership functions used depends on the accuracy of the estimation required. Although there is no restriction in selecting the number of membership functions, one does not want to use too many membership functions. Further work is required in developing schemes to automatically determine the optimum number of membership functions. Figure 3.3 shows the initial and final membership functions of the simulation result for seven membership functions.

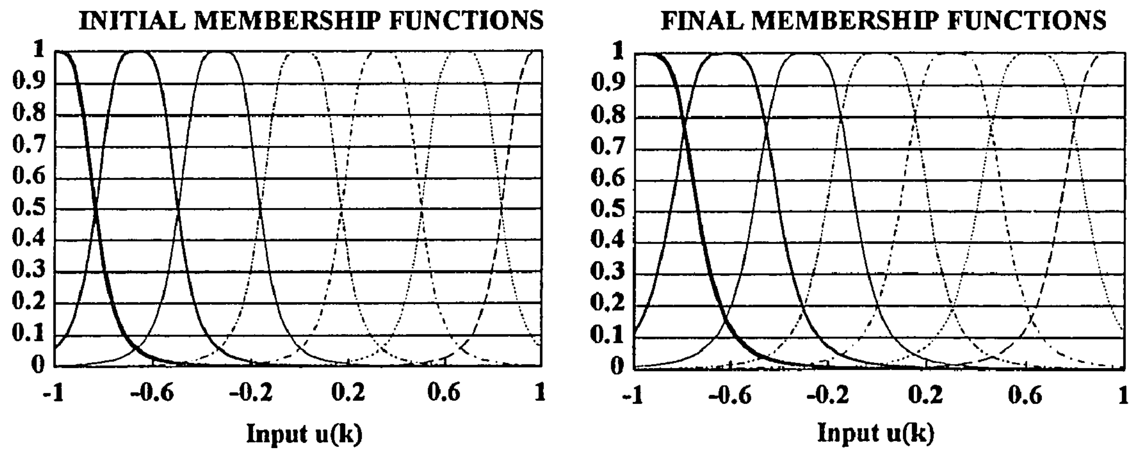

Figure 3.3 Initial membership functions and final membership functions.

The initial and final parameters of the consequent part are given in table 3.3. The weights associated with the linguistic rules are: 1.00063208, 1.00080695, 1.00058118, 0.99999654, 1.00056460, 1.00089715, and 1.00067018. Since the weight adaptation starts after the RMSE is already small, the weights are close to unity. The effect of weight adaptation is demonstrated in figure 3.4. In this figure, the comparison between having weight adaptation and not having weight adaptation is made. The weight adaptation started after about 26 iterations. It is observed from this figure that the accuracy of the estimation is further improved even through there are only small changes in the weights. It is also observed through this and other examples not reported here that using these weights results in a smoother convergence. A more intensive study on this issue, however, is required before any conclusions can be made.

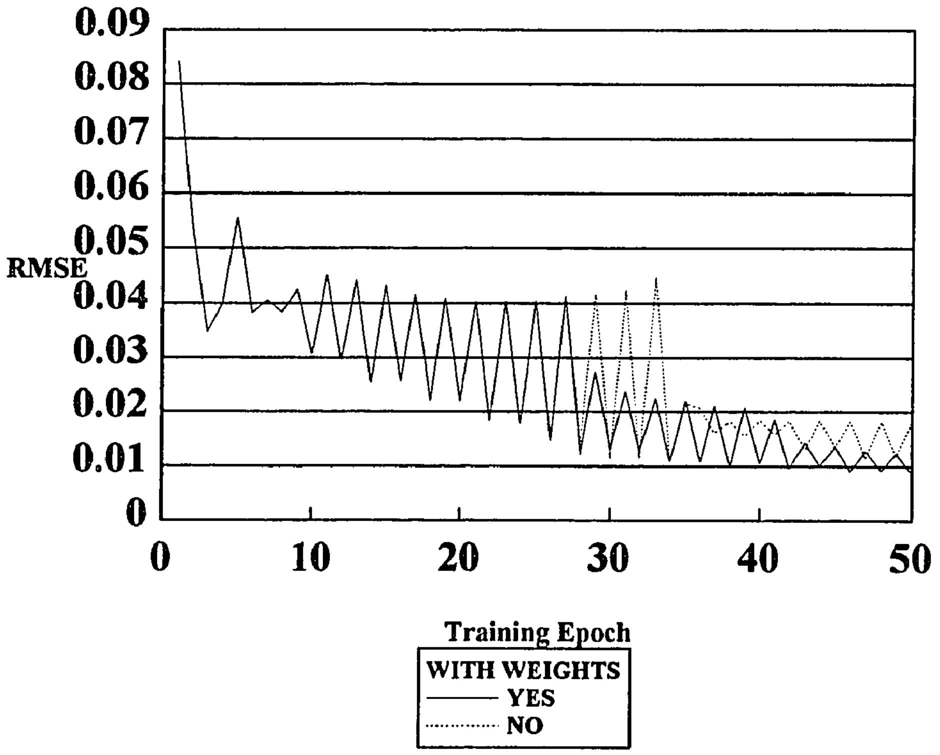

Figure 3.4 The RMSE curves for the cases of having and not having weights.

Table 3.3 Initial and final parameters at the consequent part of the linguist control rules.

Rule Number	Initial *p*	Final *p*	Initial *q*	Final *q*
#1	0	-23.994517	0	-24.70773
#2	0	-17.948412	0	-9.471423
#3	0	4.433724	0	3.919599
#4	0	31.725769	0	0.020006
#5	0	4.477868	0	-3.936794
#6	0	-17.736848	0	9.349239
#7	0	-24.044038	0	24.760621

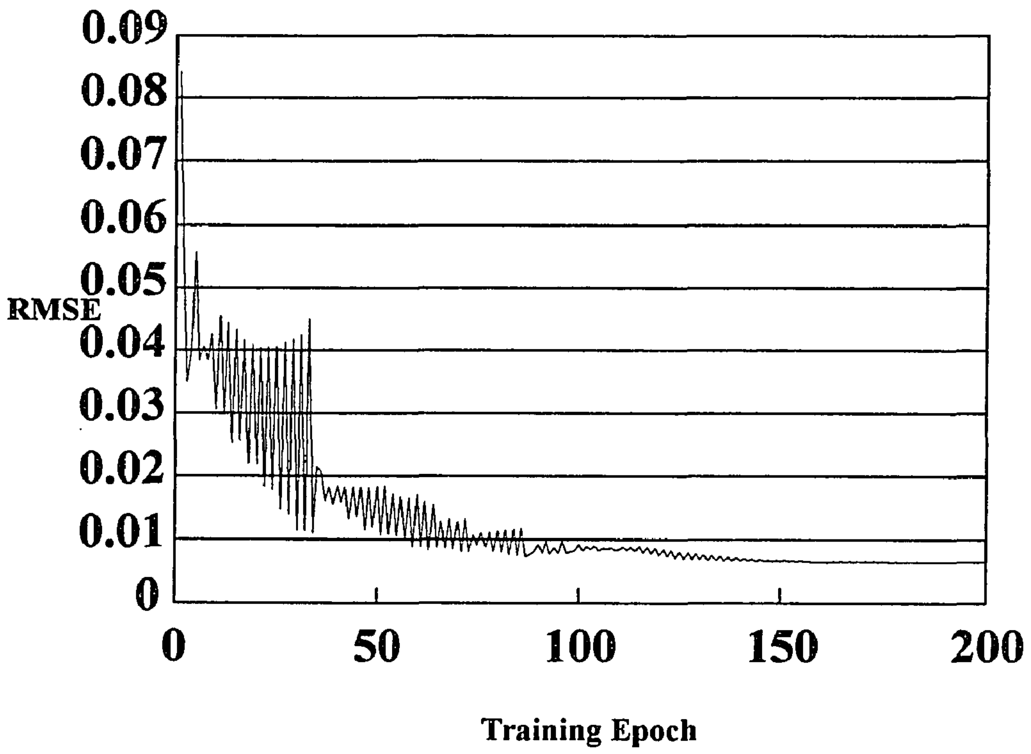

Figure 3.5 The RMSE curve for 200 epochs.

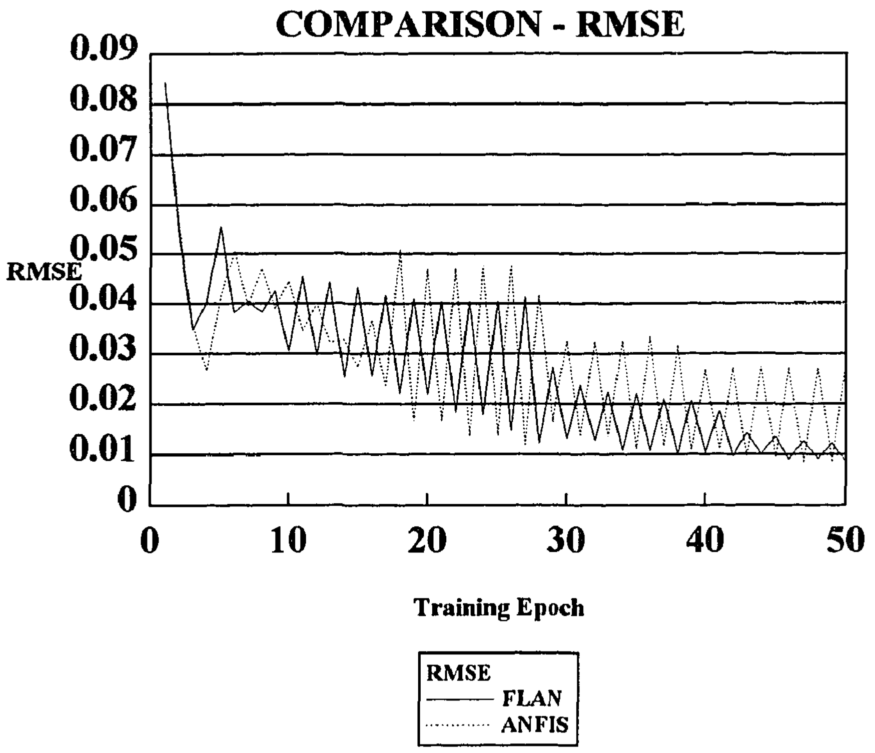

Figure 3.6 Comparison of the RMSE values.

Figure 3.6 illustrates the comparison of the results of the FLAN with ANFIS. The results of this figure are obtained through running both the networks under identical conditions. It is emphasized here that using a particular learning rate has resulted in a better converegence for ANFIS than reported in [Jang 1993]. The figure shows that the performance of FLAN is comparable to ANFIS, but exhibits smaller oscillations in RMSE. The neural network (1-20-10-1 BPNN) used by [Narendra and Parthasarathy 1990] is reported to require about 50,000 epochs before the RMSE values are comparable to what is shown in this figure. In that network, there are at least 230 modifiable parameters versus 42 in FLAN. Our experience with a similar NN shows that the number of epochs are even higher for a typical random initialization. The comparison with NN is summarized in Table 3.4. The results in this table are based on what was reported in each work.

Table 3.4 Comparison of FLAN with NN and ANFIS.

Method	Parameter Number	Training Epochs
NN	261	50,000
ANFIS	35	250
FLAN	42	50

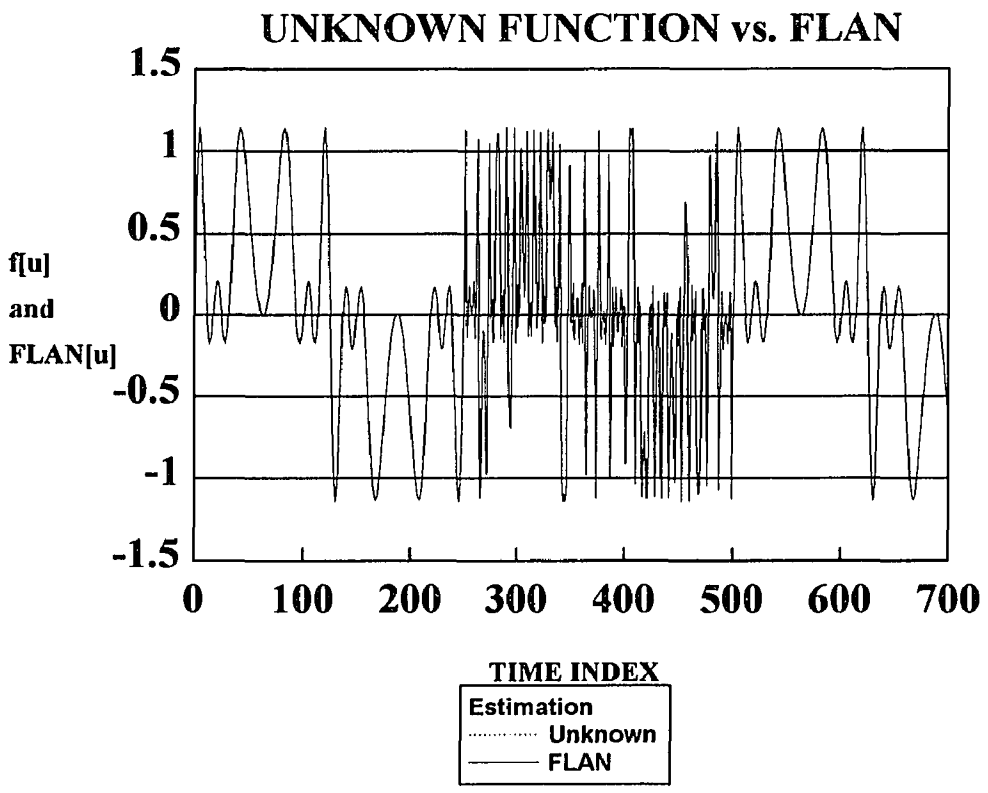

Figure 3.7 The actual and the estimated outputs of the unknown function.

Figure 3.7 shows the results of the actual output as *f[u(k)]* and the one from the FLAN (with seven membership functions for the input variable *u*) as *FLAN[u(k)]* with the input *u* given by equation (3.10). The data of the actual output is plotted as the dashed line; while the estimate by *FLAN[u(k)]* is presented as the solid line. The figure also reveals that the estimated model was able to catch up with the change of the input quite well, even when the changing signals were not represented in the training data. In the scale of figure 3.7, two curves are virtually indistinguishable. The difference between the two can be seen in figure 3.8, where the scale is enlarged and the curve illustrates the estimation error, which is the difference between *f[u(k)]* and *FLAN[u(k)]*, at each time step. Both [Jang 1993] and [Narendra and Parthasarathy 1990] obtained similar results, but over 50,000 training epochs were required in [Narendra and Parthasarathy 1990].

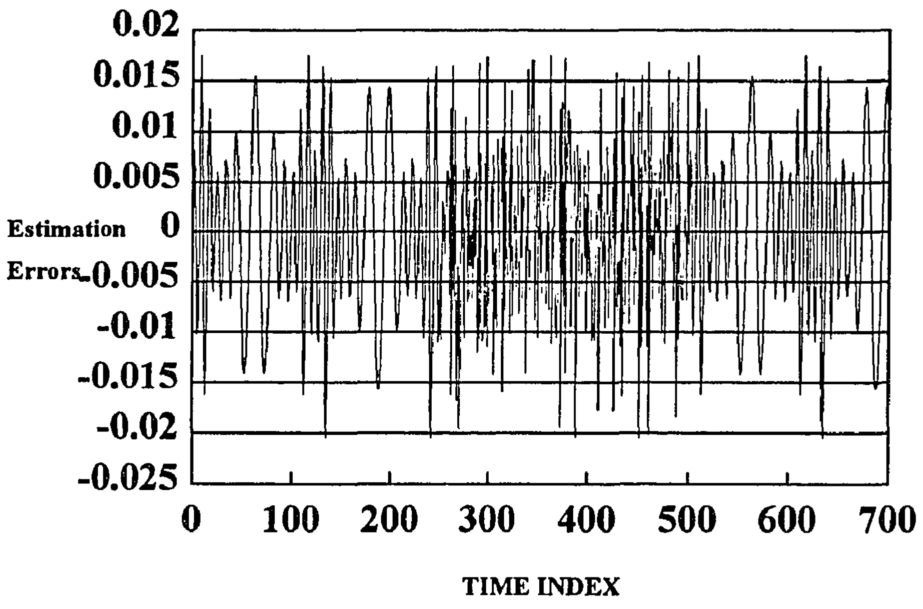

Figure 3.8 Estimation error at each time step.

The next step in the identification of the nonlinear system is to impose the estimated model into the control system. If the coefficients of *y(k)* and *y(k-1)* in (3.7) are assumed to be known, then the estimated output of the control system can be determined by equation (3.9). Now, we can compare the actual outputs of the control system with the one from the estimated outputs produced from equation (3.9). The result of the comparison are shown in figure 3.9. The dashed line represents the output signals from the control system; while the solid line represents the output signals from the identification model. Once again, in this scale, these two curves are virtually indistinguishable. In figure 3.10, the curve shows the difference between the outputs from the control system and the identification model. Since the nonlinear element in equation (3.7) is replaced by a well estimated model *FLAN[u]*, the estimated output closely follows the actual output from the control system. This reveals that the series-parallel identification model with an FLAN as an estimator can successfully estimate the control system described by equation (3.7). The results presented here are comparable to [Jang 1993] and [Narendra and Parthasarathy 1990].

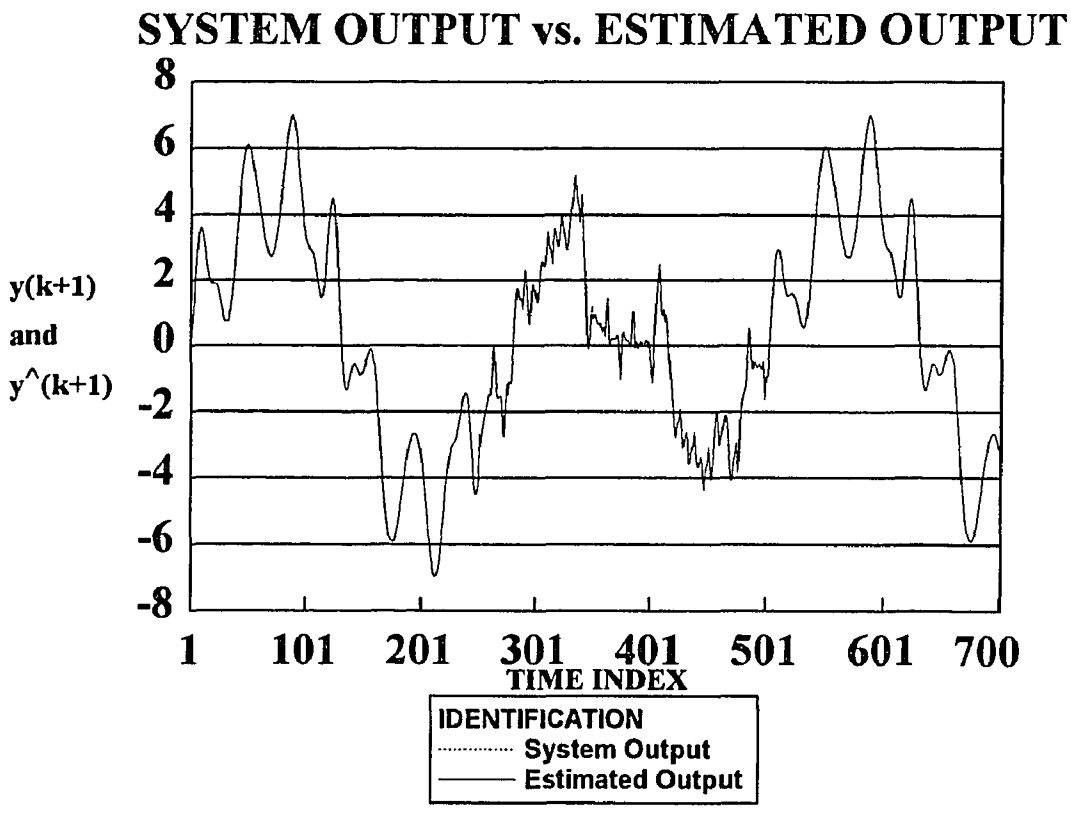

Figure 3.9 The outputs from the control system and the identification model.

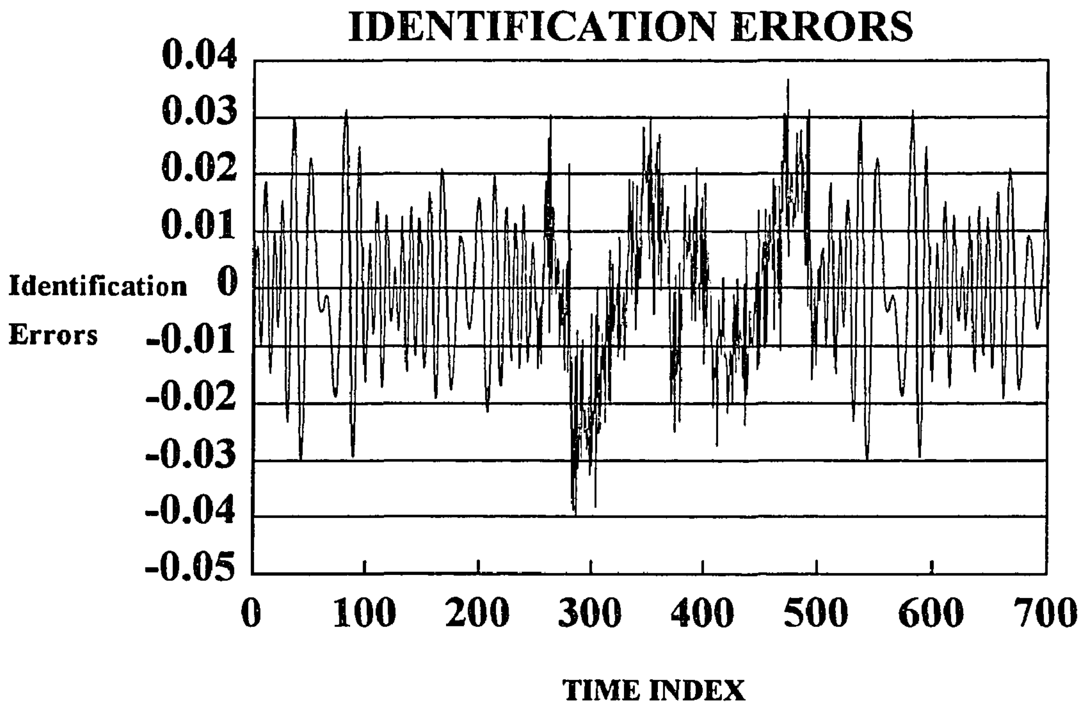

Figure 3.10 Identification errors.

The results presented so far indicate that the FLAN is successful in identification of unknown nonlinear system. The results also indicate that the performance is at least as good as ANFIS if not better, and it is significantly better than NNs at least in terms of the speed of learning. Next, several different variations of the scheme are considered to illustrate, i) the robustness of the scheme in terms of changes in the input, ii) the robustness in terms of sensitivity against noise in the training data, iii) the insensitivity regarding the learning rates over a certain range, and iv) why a scheme such as FLAN is necessary for the nonlinear identification problems as compared with using FLCs alone.

For illustrating robustness (case i, called Case 1a hereafter), a change in input, which includes a higher frequency nonlinear term as the following is considered,

$$u(k)=\begin{cases} \sin(2\pi k/250) & 1\le k\le 250 \text{ and } 501\le k\le 700 \\ 0.5\sin(2\pi k/250)+0.5\sin(2\pi k/2.5) & 251\le k\le 500 \end{cases} \tag{3.12}$$

where the change of the input to the system introduced at the time steps between 251 and 500 is adjusted from $\sin(2\pi k/25)$ to $\sin(2\pi k/2.5)$. The input *u(k)* described by (3.12) is shown in figure 3.11. The same network is used to replace the nonlinear function *f[u(k)]* in equation (3.8) but the nonlinear system is now subjected to a different input expressed by equation (3.12). The simulation result of identification is shown in figure 3.12. As illustrated in figure 3.12, the identification error is small even when a higher frequency nonlinear term is introduced. This reveals that the proposed scheme is able to tolerate a wide range of change of input to the system. Figure 3.13 shows the identification errors.

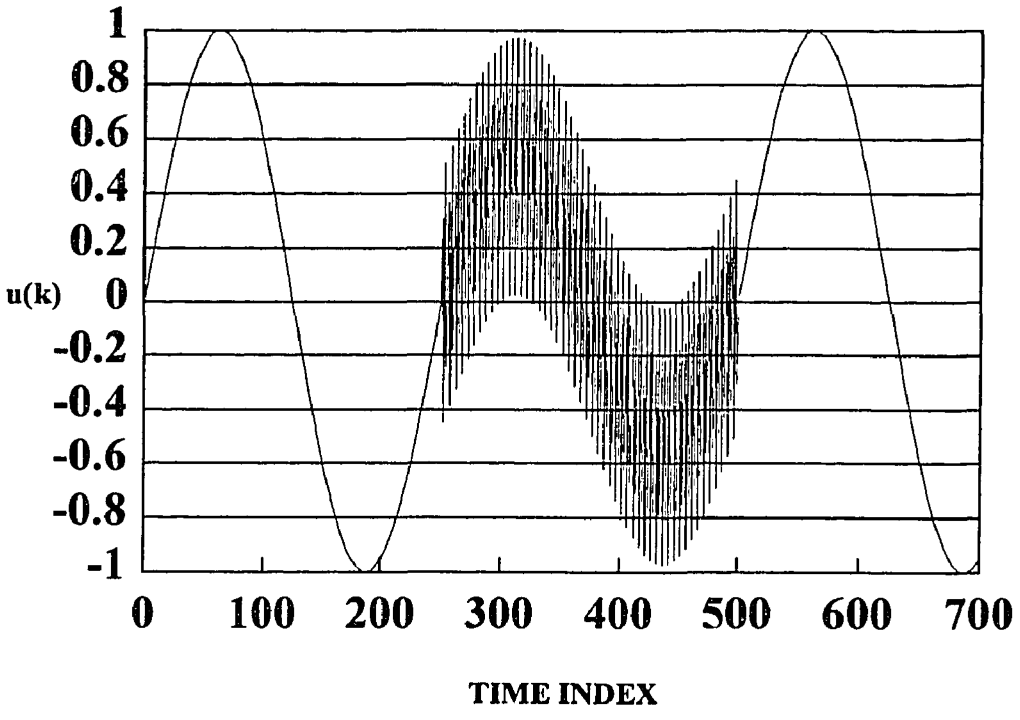

Figure 3.11 A different input to the system.

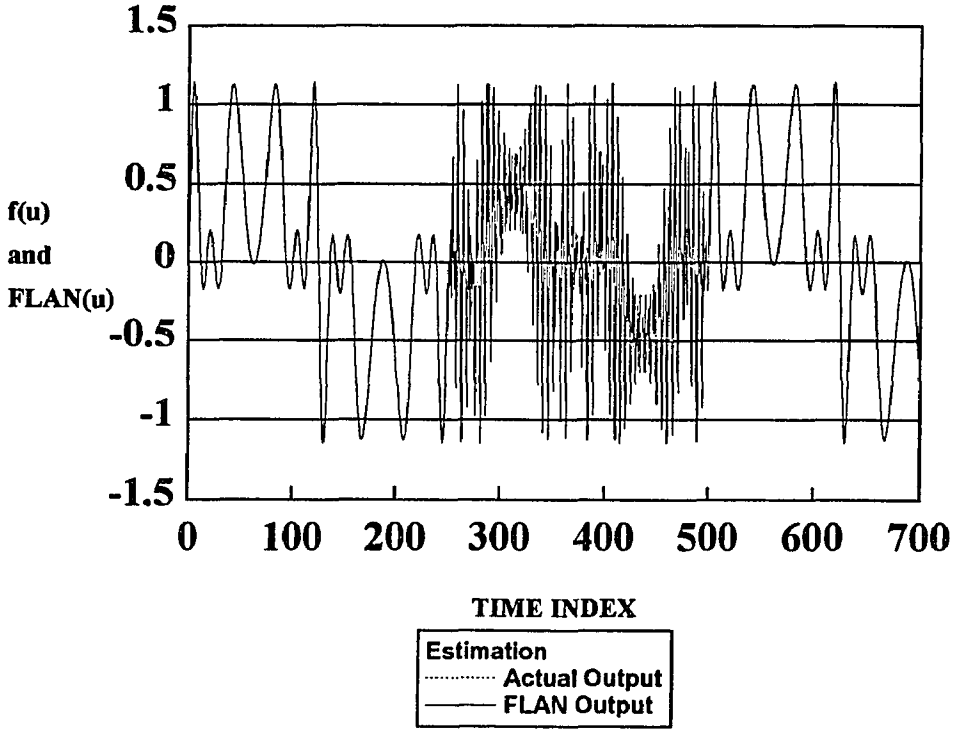

Figure 3.12 Identification result.

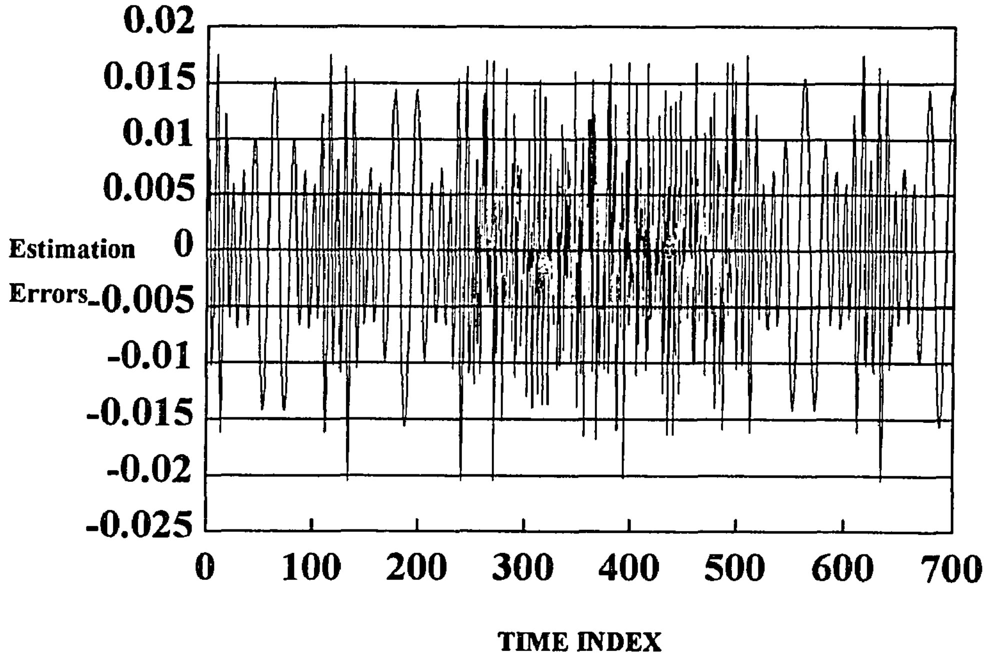

Figure 3.13 Identification errors.

Effects of noise in the training data (case ii, called Case 1b hereafter) is considered next. In practical situations, for a given input, the output of the unknown nonlinear system is measured, for example, using sensors, and thus the input-output pairs are generated for use in training. In real life, then there will be a certain amount of noise in the measurement of the output. In this experiment, a uniform random noise of ± 5 % of the maximum amplitude is added to the output, thus simulating 5 % error in the measurements. This noisy data is used for training the FLAN. After 50 epochs the minimum RMSE error (the difference between the predicted output and the actual measured noisy output) of 0.031619 is obtained. The results of nonlinear model estimation are shown in figure 3.14, and the difference in the estimate and actual model is shown in figure 3.15. The estimation does have a small amount of errors, but is certainly smaller than ± 5 %. Through this example, it is shown that the FLAN is able to come up with a reasonable estimate even with noisy training data. More investigation, however, is required in this area.

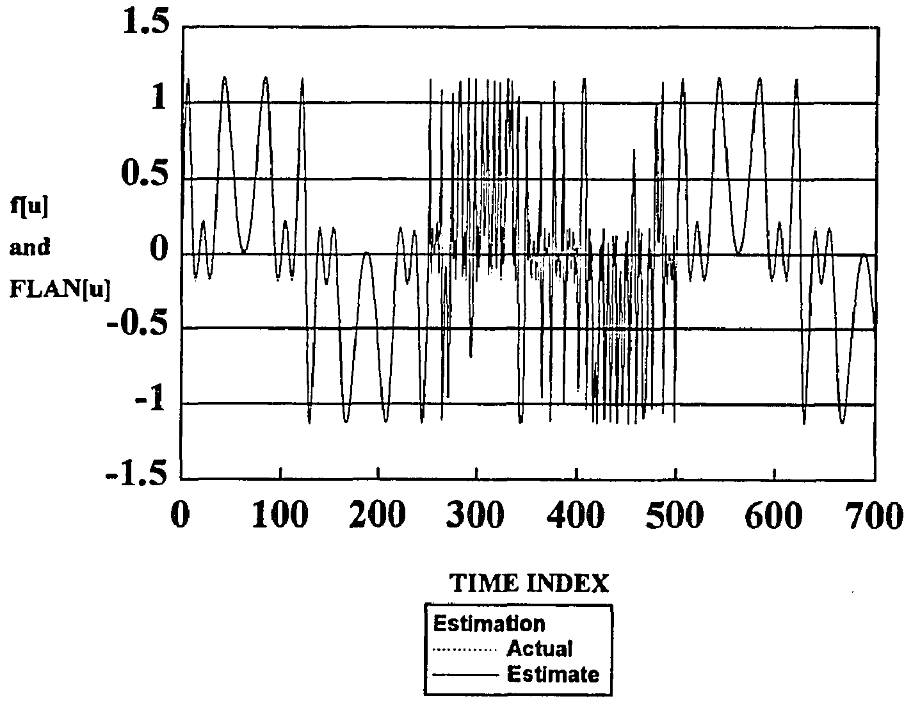

Figure 3.14 Identification results with noisy outputs in the training data.

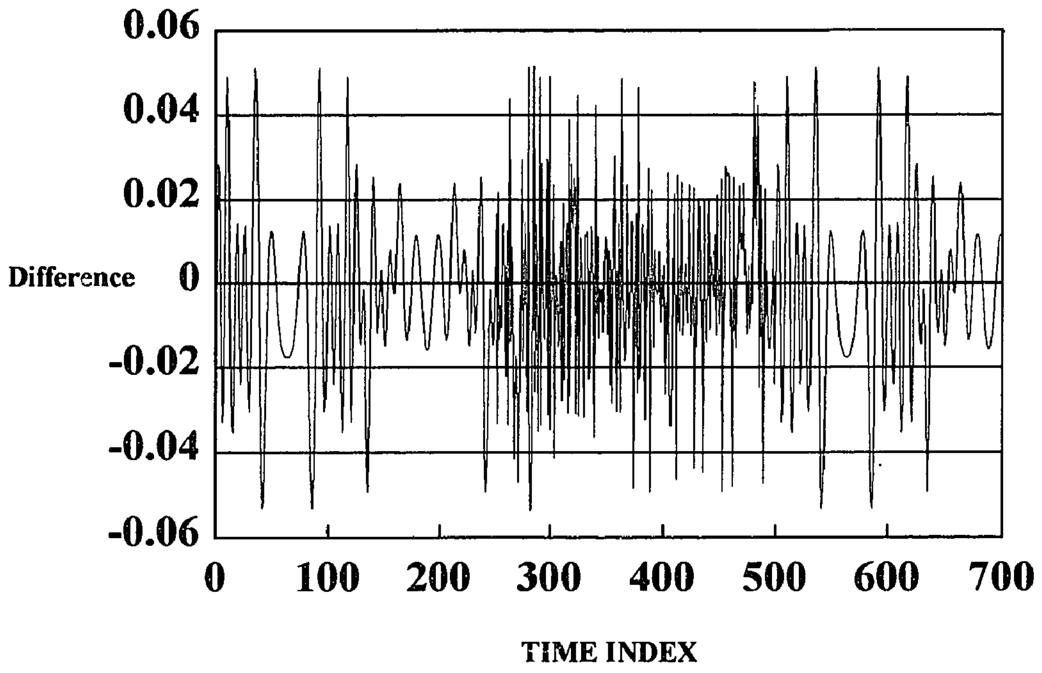

Figure 3.15 Identification errors.

Effects of learning rates (case iii, called Case 1c hereafter) are illustrated as follows. The example of case 1 is considered with a wide range of initial learning rates. As mentioned before, a simple heuristic procedure (similar to [Jang 1993]) within the algorithm adjusts the learning rate depending upon the rate of change of RMSE. Several different initial learning rates ranging from 0.01 to 0.5 were selected and for each case, FLAN was trained for 500 epochs. The results of the minimum RMSE for each case are listed in Table 3.5. The convergence trend is depicted in figure 3.16, where only three different cases representing the whole range are presented in order to avoid the clutter in the figure. As can be seen from the figure, the FLAN basically converges to a low RMSE value within about 250 epochs. When the learning adaptation scheme is not used, then much smaller fixed values of learning rates (0.005 or smaller) are required for convergence within 500 epochs. Using a fixed value is inefficient and is not recommended in general as there will be a higher tendency for getting stuck at local minima. In summary, this and other similar examples that were tried indicate that the technique is not very sensitive to the initial learning rate. Similar simulation experiments were performed using ANFIS. The ANFIS also utilizes a similar learning adaptation scheme, and thus it is expected that the results would be similar. It is interesting to note, however, that for this example, one of the initial learning rates exhibits a rather non-conforming behavior. As shown in figure 3.17, for initial learning rate of 0.4, the ANFIS fails to provide a small RMSE even after 500 epochs (the minimum RMSE is 0.034897). This result, of course, may be coincidental and does not lead to any definite conclusions. However, it is generally clear from our experiments that both FLAN and ANFIS have a similar performance.

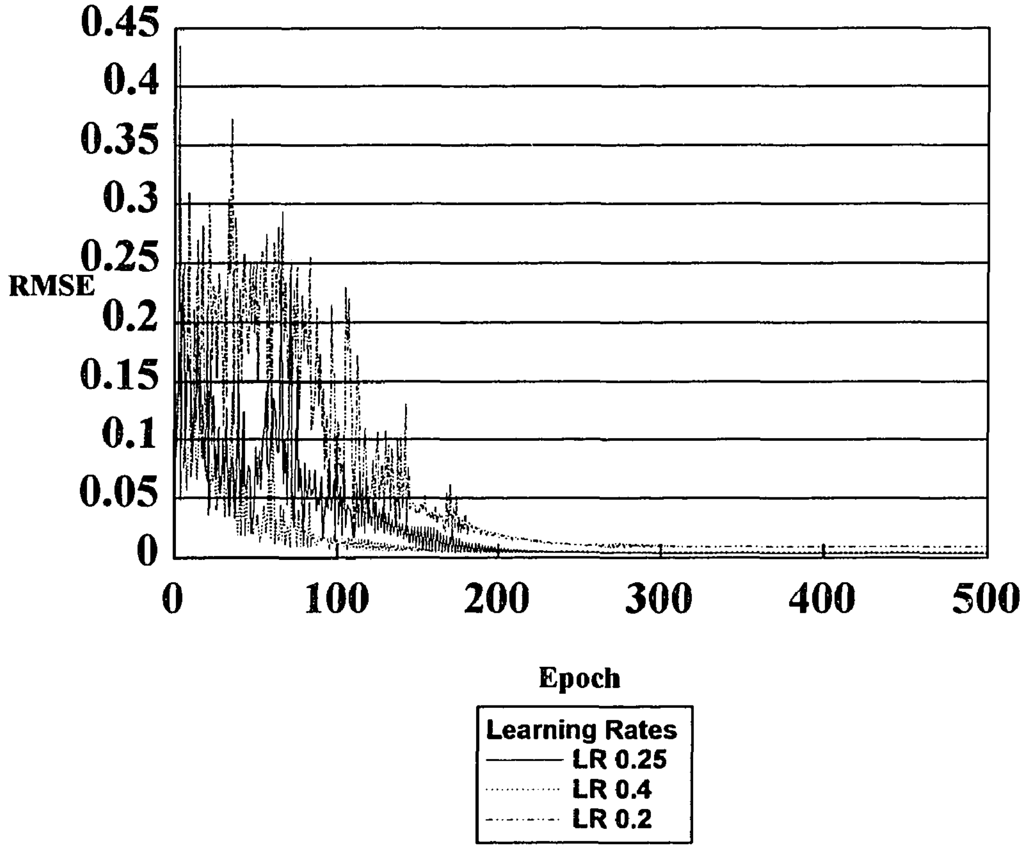

Figure 3.16 The RMSE curves of FLAN under different learning rates.

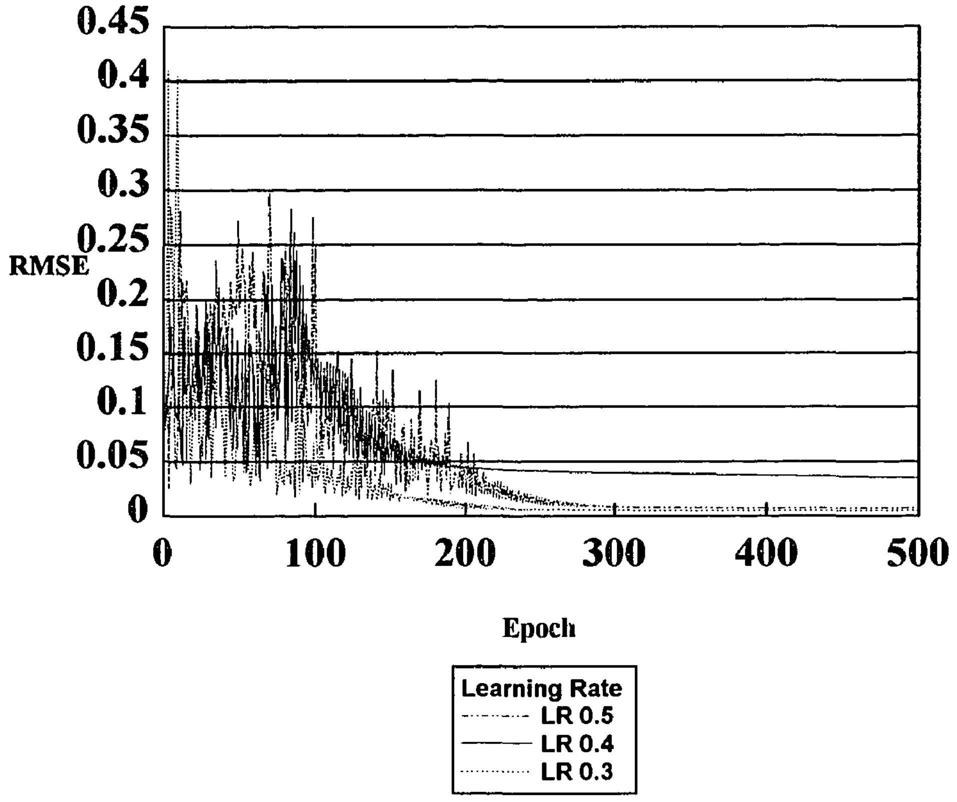

Figure 3.17 The RMSE curves of ANFIS under different learning rates.

Table 3.5 Comparison of RMSEs for various learning rates.

Learning Rate	Min. RMSE (FLAN)	Min. RMSE (ANFIS)
0.5	0.006126	0.006855
0.4	0.008414	0.034879
0.3	0.004760	0.004311
0.25	0.003272	0.004752
0.2	0.004570	0.004488
0.1	0.008560	0.005779
0.05	0.006513	0.005886
0.01	0.006179	0.005712

Next, we illustrate through the same example why a technique such as FLAN is necessary (case iv, called Case 1d hereafter) as compared to using an FLC alone. The reader may recognize that once the learning is over, FLAN is a basically an FLC if the weights for the linguistic rules are set to 1. We would like to point out that the idea of having these weights helps in structure identification and learning. However, FLAN can also be used with all these weights set to unity, without significantly affecting the final accuracy of identification. In the following experiment, the final status of FLAN (as shown by dashed curve in figure 3.4, but at end of 100 epochs) is taken as an FLC except that the final membership functions are replaced by the initial membership functions. As can be seen in figure 3.3, the difference between the initial and the final membership functions is not large, and in fact, is within the errors that a domain expert may make. Thus the resulting FLC may be considered a typical one that a domain expert may have constructed. However, even with only this small a difference between FLAN and FLC, the actual estimation using a FLC is very poor, with minimum RMSE as large as 1.112662. The results of estimation are plotted in figure 3.18 to further illustrate this effect along with the estimation errors shown in figure 3.19.

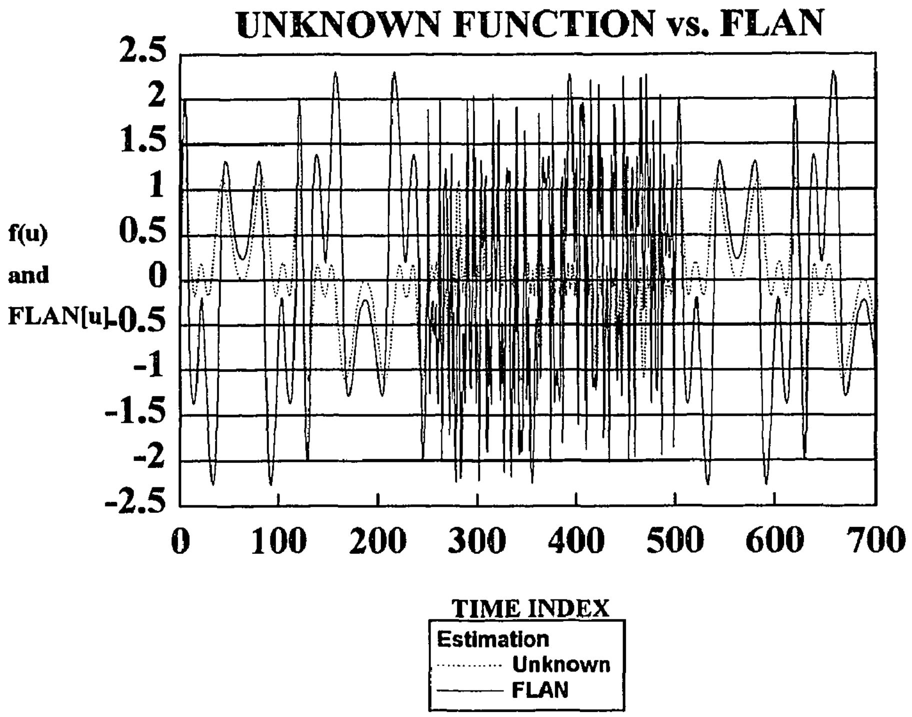

Figure 3.18 Estimation results.

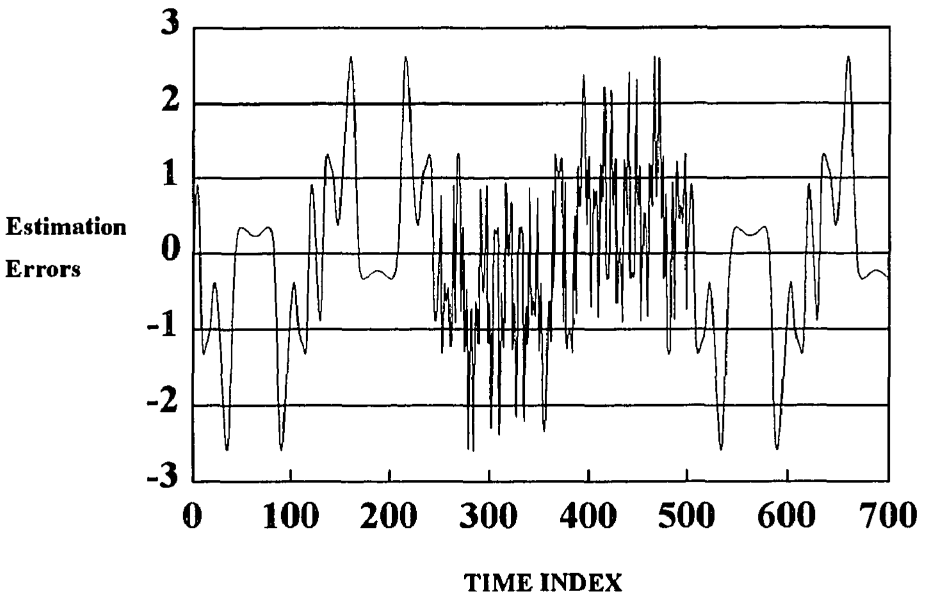

Figure 3.19 Estimation Errors.

3.3.2 Case 2: Identification of a Type 3 Nonlinear System

In this example, we consider the plant, from [Narendra and Parthasarathy 1990], described by the following equation, which is a type 3 model as in equation (3.5).

$$y(k+1) = f[y(k)] + g[u(k)] \tag{3.13}$$

where $f[\cdot]$ represents the unknown function of $y(k)$

$$f(y) = \frac{y}{1+y^2} \tag{3.14}$$

and $g[\cdot]$ denotes the unknown function of $u(k)$

$$g(u) = u^3 \tag{3.15}$$

where the input u(k), as shown in figure 3.20, is given as

$$u(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{\pi k}{5}\right) \tag{3.16}$$

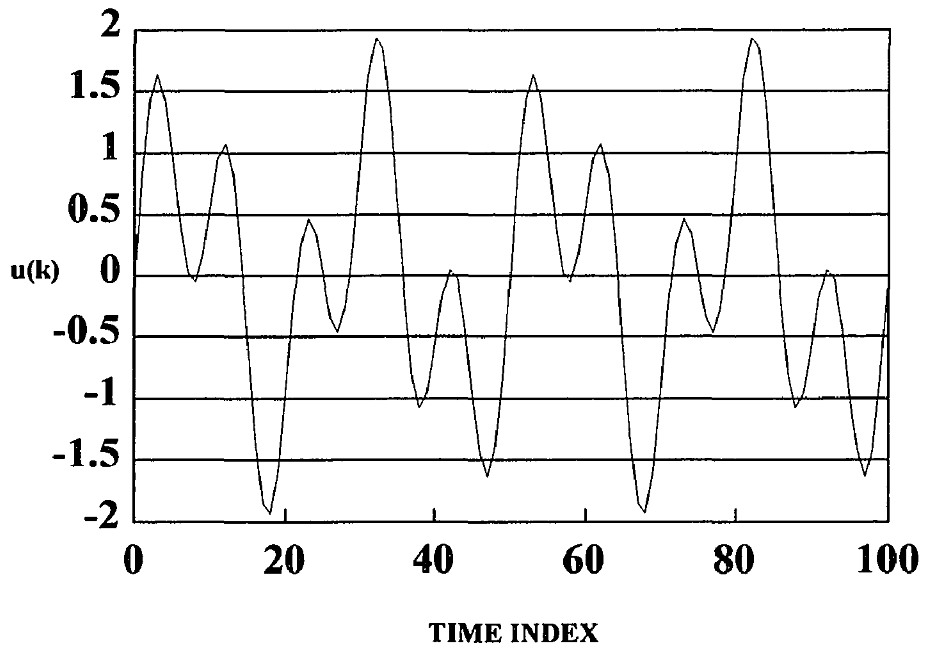

Figure 3.20 Input to the nonlinear system.

To model the plant, we may employ a series-parallel identification model involving two FLANs described as

$$\hat{y}(k+1) = FLAN_f[y(k)] + FLAN_g[u(k)] \tag{3.17}$$

where $FLAN_f[\cdot]$ and $FLAN_g[\cdot]$ are the estimates of the unknown function f and g in (3.13) respectively. Therefore, we need to train these two FLANs separately following the same procedure that was employed in the previous example (case 1).

In this example, seven membership functions are used for both $FLAN_f[\cdot]$ and $FLAN_g[\cdot]$. Ten input data are sampled in interval [-2, 2] for $FLAN_g[\cdot]$ from a set of measurements of the input-output relation described by equation (3.15) and another ten input data are sampled in interval [-10, 10] for $FLAN_f[\cdot]$ from a set of measurements of the input-output relation described by equation (3.14). Once we have the training data ready, we may start the training process. The number of training epochs is also set as 100 and the number of membership functions is selected as seven. With the same training procedure employed in the previous example, we have the following simulation results. Figure 3.21 shows the initial and final membership functions for $FLAN_f[\cdot]$.

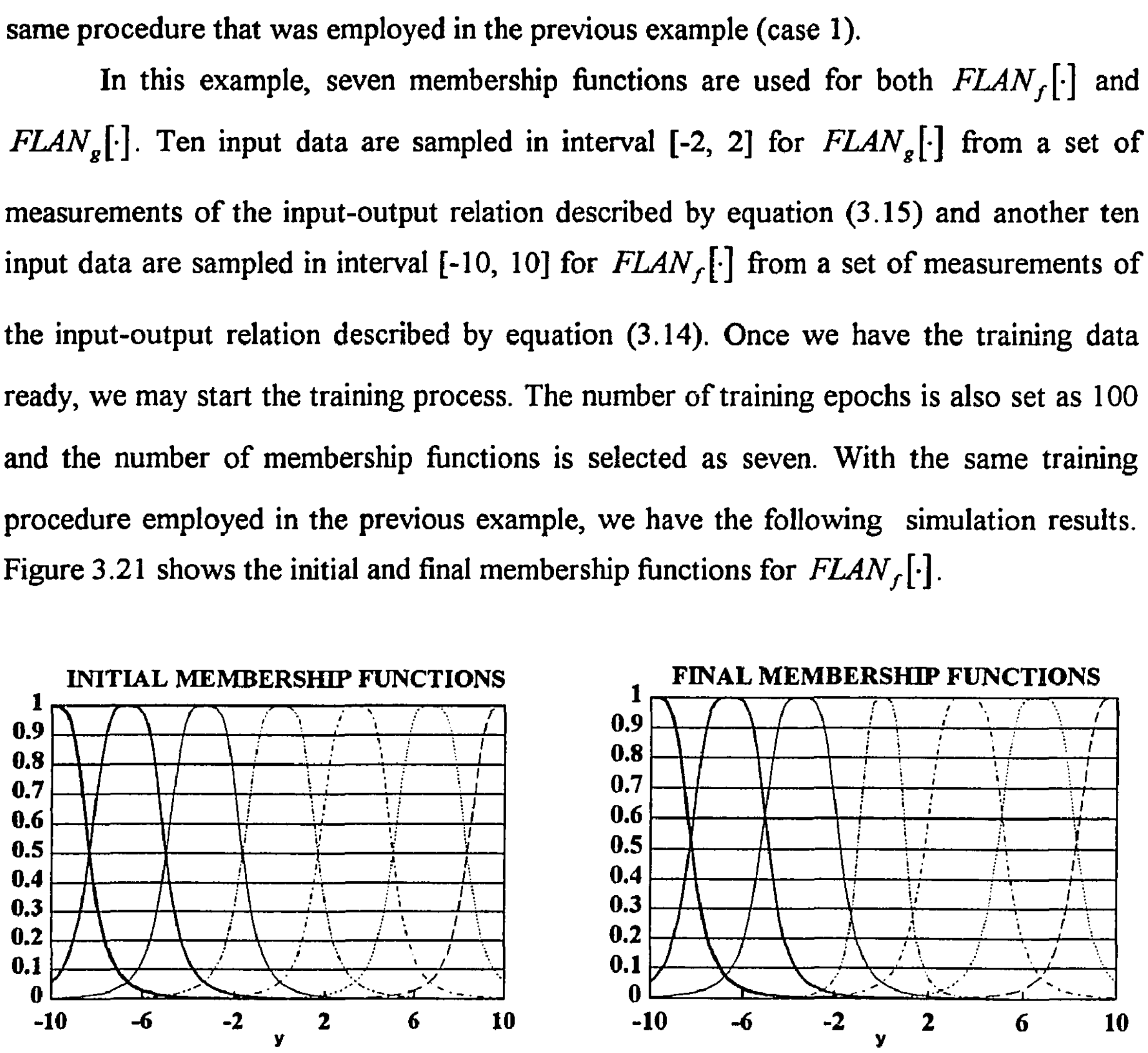

Figure 3.21 Initial and final membership functions for $FLAN_f[\cdot]$.

Table 3.6 shows the initial and final values of the parameters at the consequent part of the linguistic control rules. The weights associated with the linguistic rules are: 1.00000165, 0.99984032, 0.99982832, 1.00290391, 0.99983454, 0.99984118, and 1.00000171. The minimum RMSE for this case is 0.001464. Again, the weights have the effect of improving the accuracy of the estimation by a small amount, although the changes in the weights are small.

Table 3.6 Initial and final parameters at the consequent part of the linguistic control rules.

Rule Number	Initial *p*	Final *p*	Initial *q*	Final *q*
#1	0	-0.014456	0	-0.245291
#2	0	-0.024513	0	-0.314858
#3	0	-0.051978	0	-0.442687
#4	0	0.581997	0	-0.000006
#5	0	-0.051972	0	-0.442665
#6	0	-0.024509	0	-0.314828
#7	0	-0.014449	0	-0.245217

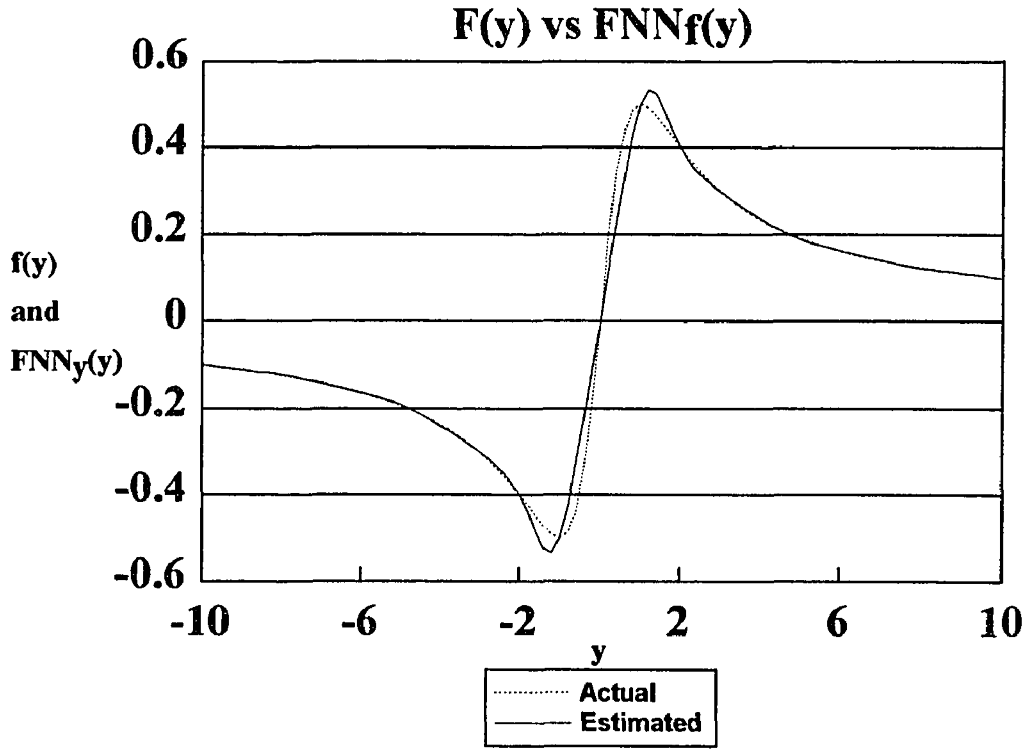

Figure 3.22 The actual outputs from *f[y]* and the estimated outputs from $FLAN_f[\cdot]$.

Figure 3.22 shows the results of the comparison of the actual and the estimated output of the unknown function *f* described by (3.14). The dashed line represents the output data from the actual output of *f* in (3.14); while the solid line denotes the estimated values from $FLAN_f[\cdot]$. They are almost identical except in interval [-2, 2]. This is caused by not having enough training data to cover the range of this sudden change. However, the result is still acceptable.

The next step is to train $FLAN_g[\cdot]$ in order to estimate g in (3.15) using the training data described before. The number of training epochs is also set as 100 and the number of membership functions is selected as seven. The following simulation results are obtained through the similar procedure used before. Figure 3.23 illustrates the initial and final membership functions for $FLAN_g[\cdot]$.

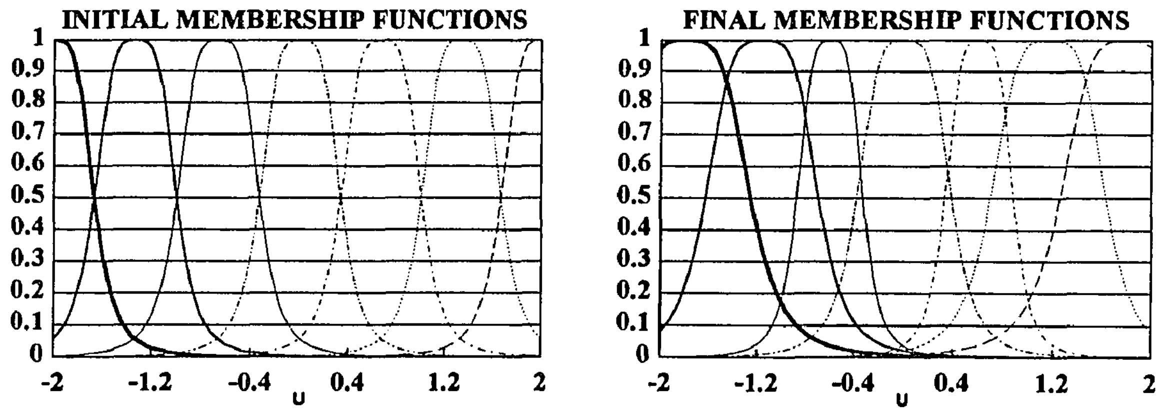

Figure 3.23 Initial and final membership functions for $FLAN_g[\cdot]$.

Table 3.7 shows the initial and final values of the parameters at the consequent part of the linguistic control rules. The weights associated with the linguistic rules are: 1.00011703, 1.00026690, 1.00009406, 0.99998587, 1.00008860, and 1.00010291. The minimum

RMSE for this case is 0.000925. Once more, the weights have the effect of further improving the accuracy of the estimation, although the changes are small.

Table 3.7 Initial and final parameters at the consequent part of the linguist control rules.

Rule Number	Initial p	Final p	Initial q	Final q
#1	0	11.939388	0	15.903503
#2	0	6.574062	0	4.879100
#3	0	2.074484	0	0.466522
#4	0	0.959759	0	0.000067
#5	0	2.074466	0	-0.466313
#6	0	6.574438	0	-4.879379
#7	0	11.93944	0	-15.903660

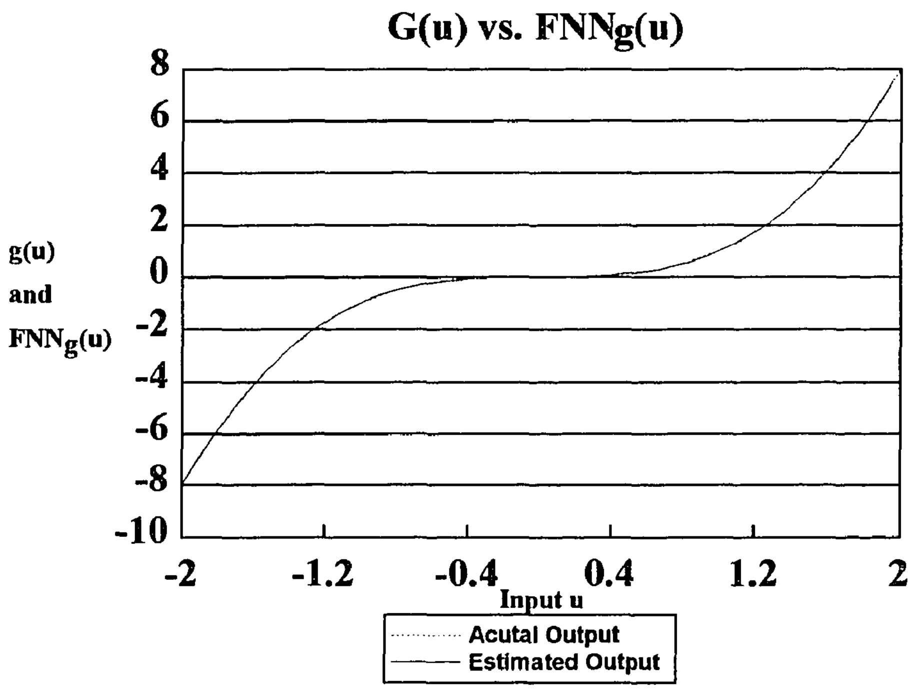

Figure 3.24 The actual outputs from $g[u]$ and the estimated outputs from $FLAN_g[\cdot]$.

Figure 3.24 shows the results of the comparison of the actual and the estimated output of the unknown function g described by (3.15). The dashed line represents the output data from the actual output of g in (3.15); while the solid line denotes the estimated values from $FLAN_g[\cdot]$. The two curves are almost identical in this case.

These two separately trained FLANs are combined to estimate the actual system output. At this point the input to the system is given by equation (3.16). Figure 3.25 depicts the result of the comparison of the actual system output and estimated output. The dashed line is the actual output from the control system and the solid line is the estimated output from the identification model. The result, likewise, shows that the identification model also successfully predicts the output of the control system.

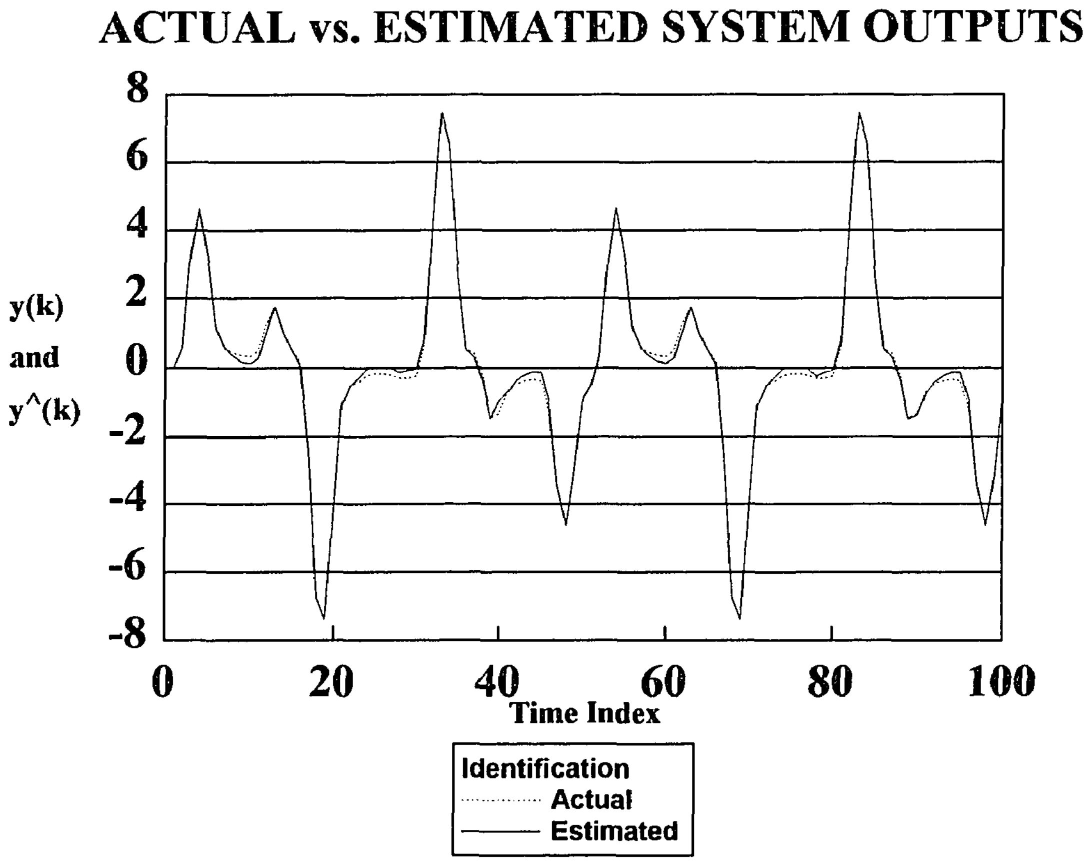

Figure 3.25 The outputs from the control system and the identification model.

3.4 Conclusions

In this chapter, an alternative approach is developed that uses the learning aspects of an NN in a structure of an FLC for the identification of nonlinear dynamic systems with unknown parameters. The simulation results reveal the strong potential of using this approach to solve complex, nonlinear problems. Through a variety of simulation experiments, advantages of the proposed technique, FLAN, over other approaches are demonstrated.

CHAPTER 4

ADAPTIVE CONTROL OF NONLINEAR DYNAMIC SYSTEMS WITH UNKNOWN PARAMETERS

4.1 Introduction

Adaptive control deals with the problem of controlling dynamic systems with time-varying parameters or unknown parameters. Most methods currently available are for control of linear time-varying systems. [Goodwin and Sin 1984, Gupta 1986, Chalam 1987, Åström and Wittenmark 1989, Narendra and Annaswamy 1989, Sastry and Bodson 1989]. These days, since control systems tend to be more and more complicated, assuming complex systems to be linear may result in very poor control quality.

Robotics, for example, is an active area of research in many disciplines, especially in mechanical engineering. In robotics, motion control requires moving the end-effector of a robot to a certain location, with a prescribed speed, along a predefined trajectory, and is a challenging problem because it involves modeling and controlling a highly complex nonlinear dynamic system. In addition to joint motion control and resolved motion control, adaptive control is one of the major approaches to solve this problem [Fu, Gonzalez, and Lee 1987, Lozano and Brogliato 1992]. In joint motion control and resolved motion control, one generally uses nonlinear compensation techniques which require an accurate model of the arm dynamics, and, usually neglects the changes in the control system, such as the variation of load in a task cycle. While in the approach of adaptive control, a linear and decoupled model of the arm dynamics is an essential component. These approaches usually result in nonuniform damping and other undesired effects [Craig 1989]. In other words, they all have a common shortcoming in modeling the nonlinearities and uncertainties in the robot arm dynamics. The result of this shortcoming reveals the need of advanced approaches to model complex nonlinear systems. Since a highly reliable tool to model the nonlinearities and uncertainties in the control system is not yet available, more research in this area is desired and expected. While the ultimate goal is to develop schemes to solve nonlinear control problems such as robotics control, reliable schemes for even simpler nonlinear systems are not available. In this chapter, we focus on applying the method developed in chapter 2 to solve a class of nonlinear control problems.

Recently, Kokotovic adopted the feedback linearization techniques and developed the adaptive nonlinear control theory [Kokotovic 1991]. Nevertheless, the assumption is that the parameters of the nonlinear dynamic system under control either appear, or can be made to appear, linearly. Other approaches to control nonlinear dynamic systems with unknown parameters, such as fuzzy logic (FL) and neural networks (NN), have also been reported to be able to approximate complex nonlinear mappings successfully [Lee 1990, Narendra and Parthasarathy 1990, Miller, Werbos and Williams 1992]. However, as mentioned previously, the applications of these approaches are still very limited because the major problem in applying these approaches is that they highly rely on expert knowledge and experience.

In this chapter, the proposed hybrid scheme, FLAN, described in chapter 2 is employed to design the controller in an adaptive control system. Following a brief introduction to adaptive nonlinear control, section 4.2 discusses the schemes that are commonly used in adaptive control. Section 4.3 presents the simulation study which uses the FLAN to design an adaptive controller for an unknown nonlinear dynamic plant. The results show that the proposed scheme is able to successfully control the nonlinear system with unknown parameters. The chapter ends with the conclusions of the simulation study in section 4.4.

4.2 Adaptive Control Schemes

There are essentially two major approaches in adaptive linear control: namely the model reference adaptive control (MRAC) and the self-tuning regulator (STR). Figure 4.1 shows the basic structure of each approach. Although there are several variations of these approaches, the common factor is that the parameters of the controller follow the change in the control system. Therefore, the main objective of using adaptive control is to develop a controller that can follow the changes in the system. In other words, we need to design a control law with adjustable parameters that can minimize a prescribed objective function. In the case of MRAC with the gradient approach, the cost function is defined as a function of the difference between the output of the control system and that of the reference model. Similarly, in the case of STR, the objective function may be defined as, for example, the variance.

The adaptive control scheme used in the next section is indirect MRAC, which is a combination of MRAC and STR. The FLAN is used as an estimation model and adaptive controller in this scheme.

4.3 Simulation Study

In this section the FLAN will be used to the design an adaptive controller. Let us consider the example described by the following equation in [Narendra and Parthasarathy 1990].

$$y(k+1)=\frac{(k)}{1+\left[y(k)\right]^2}+\left[u(k)\right]^3 \tag{4.1}$$

which is the same system that was used in chapter 3. The nonlinear dependence of the output and its past values is described by equation (3.13) as *f(.)*. While the nonlinear dependence of the input and its past values is expressed as equation (3.14) as *g(.)*. They are assumed to be separable. In chapter 3, we have successfully produced two identification models to estimate these two nonlinear functions, which are $FLAN_f[\cdot]$ and $FLAN_g[\cdot]$.

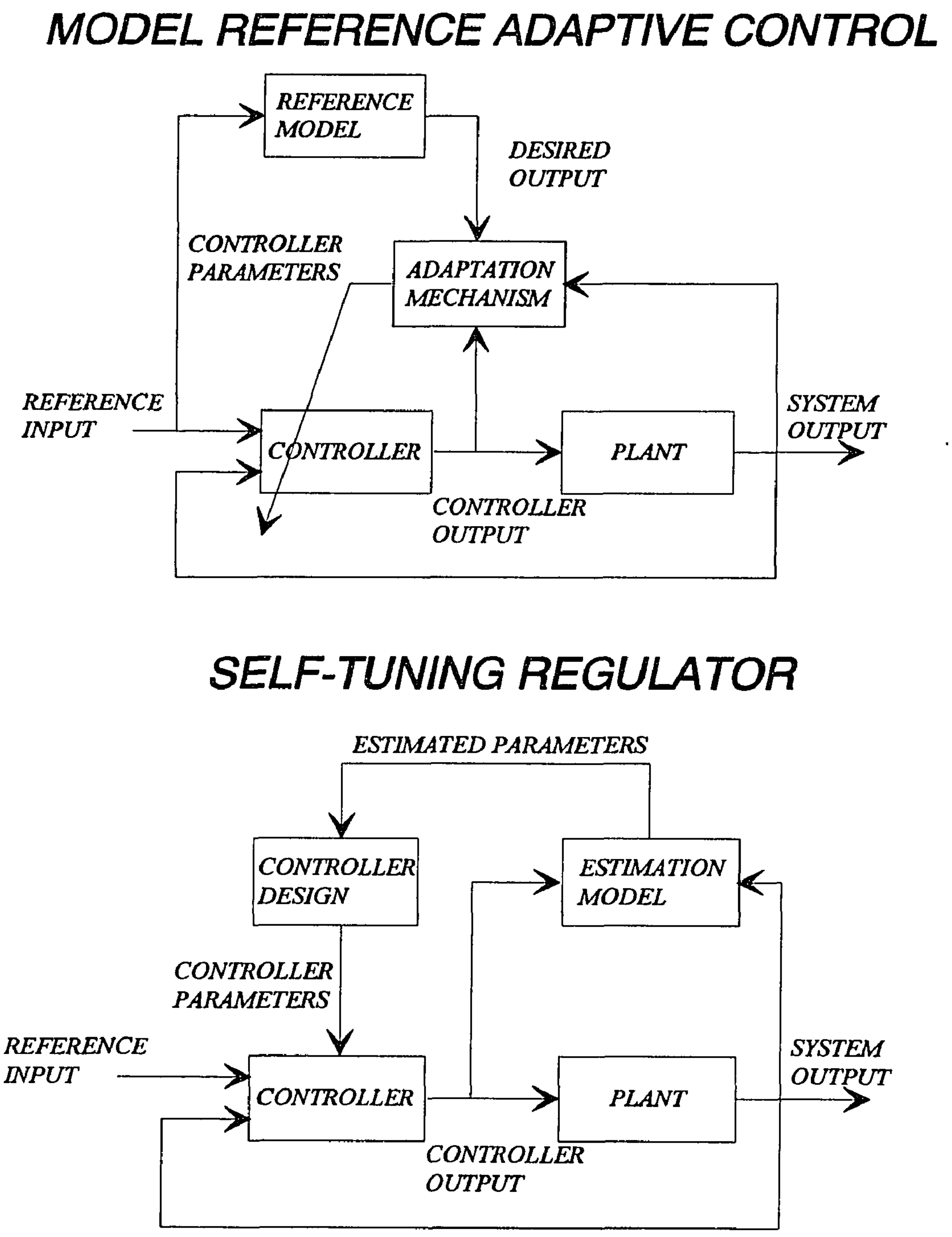

Figure 4.1 Block diagrams for model reference adaptive control and self-tuning regulator.

Suppose the desired trajectory of the system is given by a reference model as

$$y_m(k+1) = 0.6y_m(k) + r(k) \tag{4.2}$$

where y_m denotes the output of the reference model, and r represents the reference input. Figure 4.2 shows the block diagram of the control system.

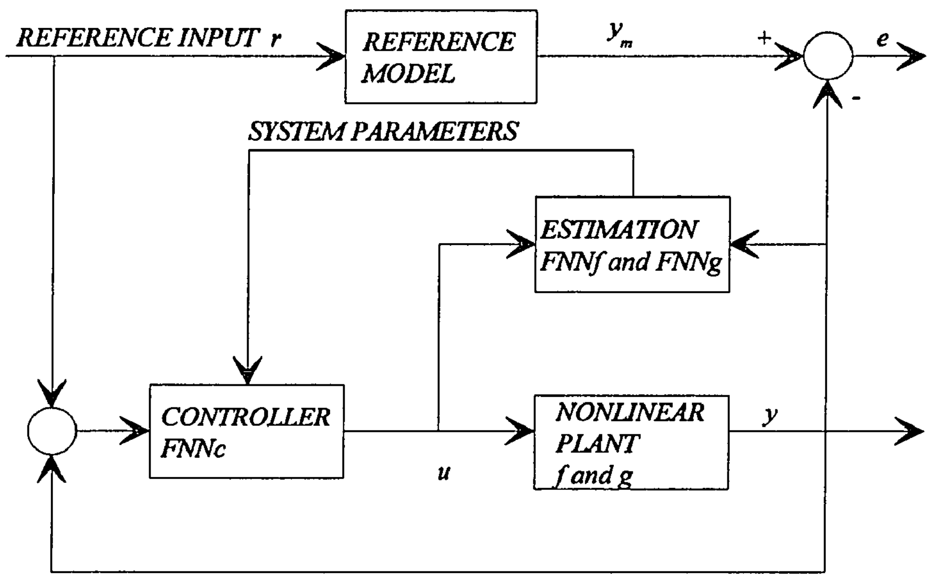

Figure 4.2 Block diagram of the control system in equation (4.1) and (4.2).

The objective of the controller is to provide the control signals so that the system may follow the output of the reference model as close as possible. In other words, we need to minimize the difference between the actual output from the control system and the desired output from the reference model. In this case, the control law is chosen as

$$u(k) = \hat{g}^{-1}\left[-\hat{f}[y(k)] + 0.6y(k) + r(k)\right] \tag{4.3}$$

where *u(k)* represents the control law, $\hat{g}^{-1}[\cdot]$ denotes the controller that generates the control signal *u(k)*, and $\hat{f}[\cdot]$ is the estimation of $f[\cdot]$ in equation (3.13). Since we have successfully developed an identification model $FLAN_f[\cdot]$, we shall use it to replace the estimation $\hat{f}[\cdot]$ in (4.3). In the mean time, we shall replace $\hat{g}^{-1}[\cdot]$ with $FLAN_c[\cdot]$, which is, similar to $FLAN_f[\cdot]$, a hybrid network.

The simulation study starts with training $FLAN_c[\cdot]$ so that $FLAN_g[FLAN_c(r)] \approx r$ as the reference input r varies in interval [-4, 4]. We may also use the identification model $FLAN_g[\cdot]$ developed in chapter 3 to test the results of training. Twenty training patterns are collected as r varies in [-4, 4] and the training epochs is set to be 100. Figure 4.3 shows the initial and final membership functions of the reference input r. Table 4.1 presents the parameters and the weights associated with the linguistic rules. The final weights are: 1.00003359, 0.99987993, 0.99998254, 1.00008291, 0.99998200, 0.99971454, and 1.00017453. The minimum root mean square error (RMSE) in this case is 0.001668.

Table 4.1 The final values of consequent parameters of the linguistic control rule.

Linguistic Control Rule	p	q
#1	0.1521673764	-0.9815163568
#2	0.1848535892	-0.8929286859
#3	0.3114866271	-0.6890313951
#4	2.758424389	-0.0187295369
#5	0.3021608388	0.7003829436
#6	0.1854860738	0.8911815786
#7	0.1527689597	0.9792456189

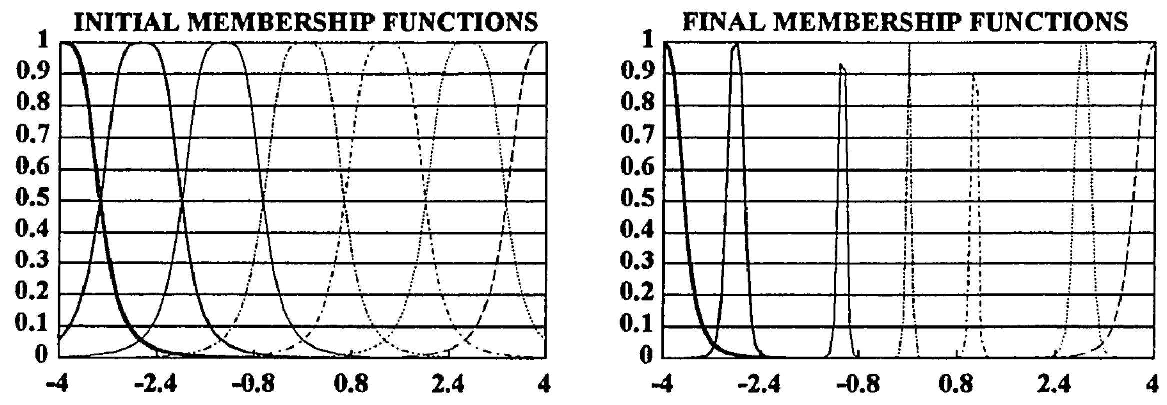

Figure 4.3 Initial and final membership functions of the reference input.

Once we have $FLAN_c[\cdot]$, we may test whether $FLAN_g[FLAN_c(r)] \approx r$ or not. Figure 4.4 plots the result of the test and shows that $FLAN_g[FLAN_c(r)]$ is close to r because the slope is about one in [-4, 4]. Now we may use $FLAN_c[\cdot]$ to replace $\hat{g}^{-1}[\cdot]$ in equation (4.3).

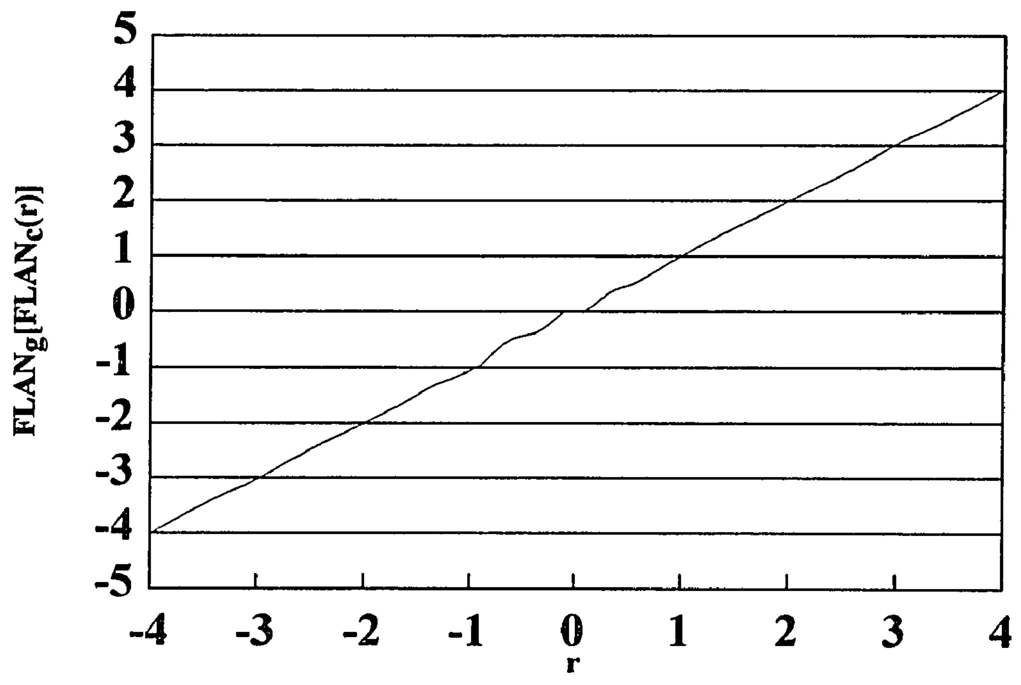

Figure 4.4 Plot of $FLAN_g[FLAN_c(r)]$ vs. r.

Suppose the reference input is described by the following equation.

$$r(k) = \sin\left(\frac{2\pi k}{25}\right) + \sin\left(\frac{\pi k}{5}\right) \tag{4.4}$$

We may study the behavior of the control in two cases. One is the case without control; and another is the case with control. In former case, the output of the control system can not follow the desired trajectory given by the reference model. Figure 4.5 show the results. The dashed line represents the desired output from the reference model; while the solid line depicts the actual output from the nonlinear plant.

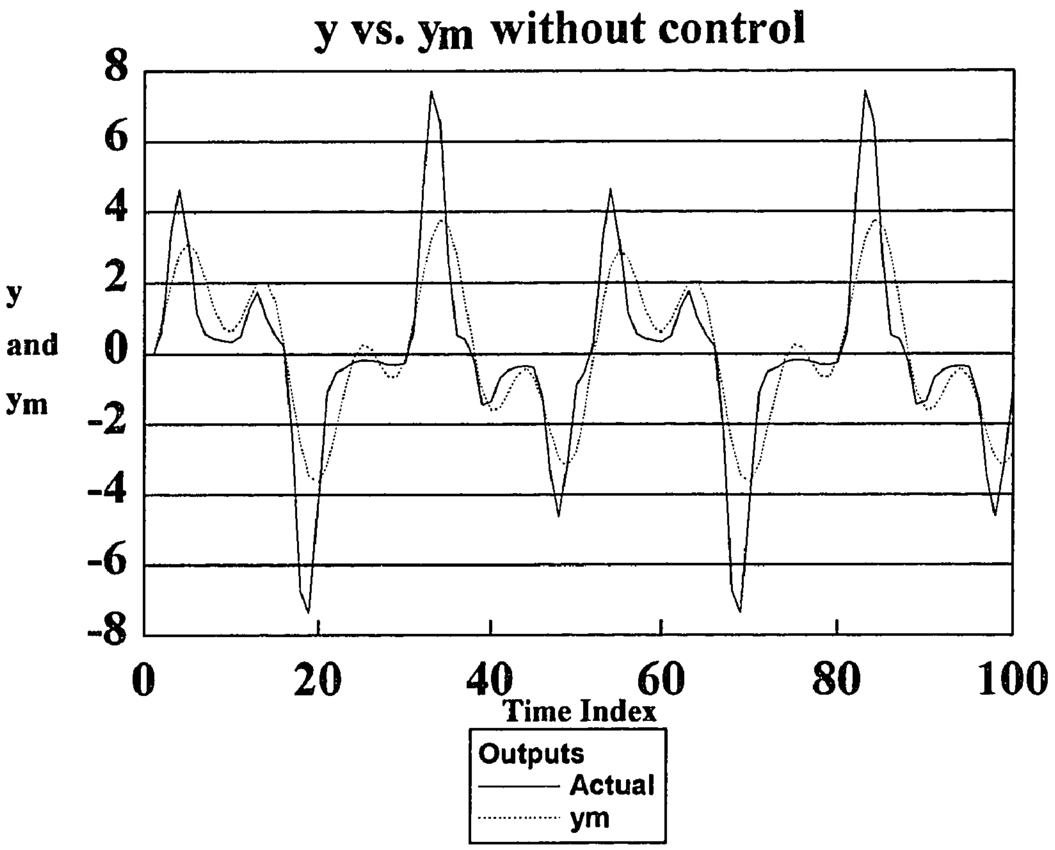

Figure 4.5 The outputs of the nonlinear plant and the reference model in the case without control.

In the latter case, the control signal is generated by

$$u(k) = FLAN_c\left[r(k) + 0.6y(k) - FLAN_f\left[y(k)\right]\right]. \tag{4.5}$$

The results are shown in figure 4.6 where the solid line and dashed line are virtually indistinguishable. It is observed that the output of the nonlinear plant closely follows the desired trajectory.

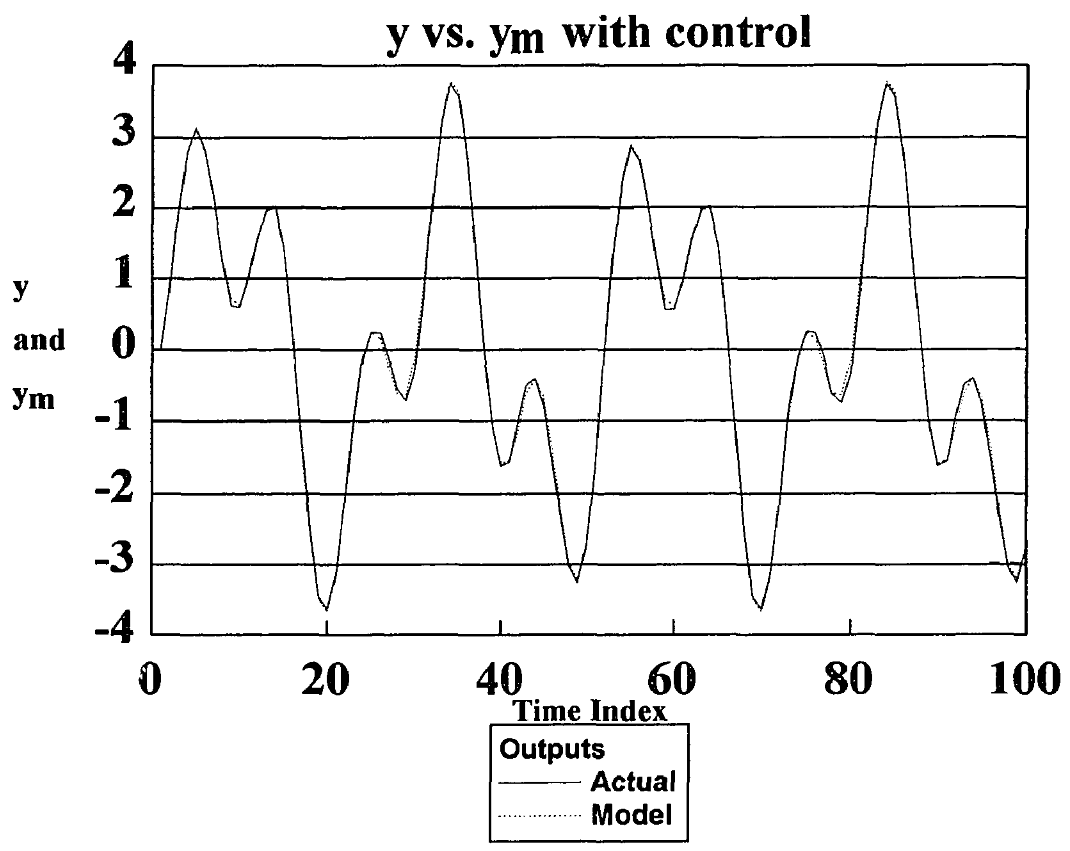

Figure 4.6 The outputs of the nonlinear plant and the reference model in the case with control.

4.4 Conclusions

We have shown that the FLAN can be utilized for the purpose of the adaptive control of nonlinear dynamic systems with unknown parameters. The simulation results show the promising potential of using this approach to design the controller in an adaptive control system.

For the example used in this chapter, we tried two different ways of tuning the weights in the defuzzifying layer to see the effect of the weights. In one case we fixed the weights to be one all the time, which is similar to the ANFIS [Jang 1993]; while in the other case we tuned the weights after the RMSE is a small value. After 100 training epochs, the former case resulted in the minimum RMSE of 0.01707 and the latter case in the minimum RMSE of 0.001688. This experiment further shows that the weights associated with the linguistic rules may help in fine tuning the results of the mapping.

There are several ways in which the results of the control may be improved. The first is to increase the accuracy of the *FLAN* 's. The second is to improve the control law by considering the effect of the control error. These are issues requiring further work.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this dissertation, we have developed a hybrid scheme, called Fuzzy Logic Adaptive Network (FLAN), that combines the good features from fuzzy logic (FL) and neural networks (NNs), and applied it to the problems of identifying and controlling nonlinear dynamic systems with unknown parameters. The results of simulation studies have shown that FLAN is an effective tool for adaptive nonlinear control problems. It is found to be a very practical tool which provides a very systematic procedure, and thus it fulfils the objectives of this work.

In this section, comparsions with conventional adaptive control schemes, NN approaches, FLC approaches, and other hybrid schemes are summarized, and the advantages of using FLAN are also discussed from different aspects. Section 5.2 discusses the issues needing futher studies.

Comparison with conventional adaptive nonlinear schemes: As discussed in chapter 1, most adaptive nonlinear control schemes are generalized from the adaptive control theory for linear systems. Although the adaptive control theory is now relatively mature, it is generally based on the assumption of linear time-invariant plant [Hunt *et al.* 1992]. Even with the adaptation process resulting in the overall system being nonlinear, the fundamentals of these methods lie in linear systems theory [Narendra and Annaswamy 1989] and most applications require a significant level of prior knowledge about the plant. Many schemes either use a linearized controller to control the nonlinear plant or use a nonlinear controller to control a linearized model. For example, in [Ortega, Canudas, and Seleme 1993], the authors proposed a estimation scheme to handel time-varying (linearly parameterized) unknown loads of induction motors. The estimation, also known as system identification, is usually done by assuming that a linear model is available and by using recursive parameter identification schemes to search a set of parameters to fit the linearized model [Ruck *et. al.* 1992]. Other approaches, such as adaptive feedback linearization control scheme by Kokotovic, also requires the system to be feedback linearizable [Kokotovic 1991]. Although this approach has been shown to be useful in nonlinear control problems, there are practical difficulties such as in constructing an exact feedback linearizing transformation. Therefore techniques based on NNs and FLCs are more attarctive for the control of nonlinear systems with unknown parameters. In comparison with conventional adaptive control methods, the FLAN has the advantage of emulating the unknown nonlinearities in the control system without making any assumptions. Thus this method has a great potential for solving more complex control problems that the conventional methods can not solve.

Comparison with NN approaches: The results of estimation, identification, and control in the previous chapters indicate that the performance of FLAN in terms of accuracy is as good as or better than achieved in [Narendra and Parthasarathy 1990] using an NN. It is shown based on the simulation results in chapter 3, Case 1, that compared to using NNs alone, FLAN has several advantages. First, there are at least 230 modifiable parameters in the NN used to solve the example of Case 1 [Narendra and Parthasarathy 1990] versus only 42 in FLAN. The fact that using an overparametrized model could lead to convergence problems is rather well known in adaptive control [Hunt *et al.* 1992]. Thus use of more parameters is not always a good idea in terms of trying to get a better approximation. Second, in terms of the training time, a longer training time, over 50, 000 epochs, is required for Narendra's NN; while the FLAN requires only 50 epochs to produce comparable results. Thus computationally, our scheme is more effective both in terms of training and in on-line control because it involves less number of processing units and less number of modifiable parameters. Third, in NN approaches, the design procedures are largely *ad hoc* because there are no general guidelines to determine the structure of the network. For example it is not entirely clear why a 1-20-10-1 BPNN is used in [Narendra and Parthsarathy 1990]. The FLAN, on the other hand, is more systematic and thus practical because the only factor involved in designing the initial structure is the number of membership functions for each input. The specification of the number of membership functions is much easier as compared to specifying the NN architecture, and the number of membership functions used also has a direct physical meaning for each practical problem. Fourth, the resulting estimation or identification network in case of FLAN has a physical structure that is very meaningful since it is essentially a fuzzy rule based inference system. The NN on the other hand is no more than a black box to a practicing engineer. Fifth, the FLAN always utilizes a consistent scheme for initializing network parameters, thus the problem of the results depending on a particular random initialization is avoided. On the other hand, the random initializations used in an NN may require technique such as Monte Carlo method for generating a set of initializations. Such procedures further add to the computations.

Both the FLAN and the techniques based on NN alone have a problem of selecting a learning rate. In this work, a simple learning rate adaptation scheme is found be satisfactory for the examples considered. It is noted here that as long as gradient descent type learning is used in either FLAN or an NN, the issue of finding a proper learning rate remains an open question.

Comparison with FLC techniques: Compared to the conventional FLC, FLAN has the benefit of automatically determining the parameters of the membership functions at both premise and consequent parts without requiring the knowledge of a domain expert. In other words, the FLAN provides a more practical approach by adopting the learning concepts from NNs, and thus eliminates the need of manually tuning the membership functions in a conventional FLC. Moreover, in the conventional FLC, even when the knowledge of a domain expert is available, further parameter tuning may be necessary as illustrated through the example of Case 1d in chapter 3. This tuning is automatically taken care of by FLAN.

Comparison with other hybrid techniques: In comparison with Kosko's fuzzy associative memory (FAM), where the fixed membership functions and the linguistic control rules are determined by so-called domain experts or simply by trial-and-error, the FLAN has the advantage in that the membership functions and the linguistic control rules are determined through the learning process. In a time-varying version of FAM, called adaptive FAM (AFAM) [Kosko 1992a, 1992b], the weights associated with linguistic rules are tuned to produce a better mapping through a learning process. However, this scheme still relies on the proper setting of the membership functions. In addition, FLAN also employs the weights associated with linguistic control rules, thus providing flexibility in terms of structure identification.

The FLAN is quite similar to the adaptive-network-based inference system (ANFIS) proposed by [Jang 1992, 1993]. However, use of the weights associated with linguistic control rules in FLAN make it more flexible. These weights appear to help in fine tuning the mappings and in removing the redundant control rules. It is obserevd through the simulation experiments that the performance of FLAN is at least as good as ANFIS, if not better, in most cases. Both of them exhibit similar learning trends (example of Case 1c in chapter 3) despite the fact that ANFIS employs a different type of hybrid learning scheme which is supposed to be less sensitive to the initial learning rate.

Other issues: In terms of robustness against changes in the input, the example of Case 1a in chapter 3 illustrates that the FLAN can tolerate a change in the input frequency. Case 1b illustrate that the FLAN is relatively insensitive to noise in the training data. It is interesting to observe how well the FLAN can approximate a noisy model. The issues such as these, however, require theoretical investigation.

In summary, a very practical and easy to use systematic approach is developed for identification and control of nonlinear systems with unknown parameters. However, considerable work remains to be done to in order to explore the full potential of the FLAN and improve it, and utilize it for the purpose of identification and control of nonlinear systems. Several important areas of further work are outlined next.

5.2 Future Work

The FLAN is designed for solving the problem of parameter identification as well as structure identification. It was shown that it is able to tune the membership functions and the linguistic rules in a fuzzy logic controller (FLC). However, more work is required to demonstrate its effectiveness in structure learning. Further study is also required in order to study the effects of the weights in the defuzzifying layer. Recent research suggests several approaches to hadling the problem of structure learning that may be of use in FLAN [Lin and Lee 1994, Sun 1994, Higgins and Goodman 1994].

More work is also required in the area of practical applications, such as in the field of robotics. Issues related to applying FLAN to complex, coupled multiple input-multiple output dynamic systems such as robot manipulators more detalied studies. For such applications, further investigation is required realted to the use of different types of rules in the consequent part. In this study, the rules used are those of Sugeno type, which assume that the consequents are the linear combination of the inputs. However, this is not always true, especially in the domain of system control. Therefore, higher order equations are needed. If the parameters are linear for these higher order equations, least sqaures estimates may be used for learning. The group method of data handling algorithms by Ivakhnenko [Farlow 1984] or other regression methods may used to determine the parameters of the higher order rules.

Most importantly, much theoretical and analytical work is required to address issues of stability, convergence, and robustness. Future developments in the fields of neural networks and adaptive systems may be helpful in developing theoretical frame work for the proposed approach. More work is needed to connect these developments with the proposed scheme.

APPENDIX A

DERIVATION OF THE PROPOSED HYBRID SCHEME

A.1 Step 1: Determining the Parameters of the Network

The following is the derivation of the proposed hybrid scheme. In order to simplify the derivation, a two-input-one-output system is used here.

The first step is to determine the architecture of the hybrid system, which includes the determination of the configuration of the hybrid architecture and the identification of the parameters associated with each node in the hybrid system.

Suppose that each input is quantified to two linguistic labels, such as *high*, *low*, etc. In this case, the input layer contains two node with node functions equals to ones; the fuzzifying layer consists of four nodes represent four membership functions; the inferring layer includes four nodes express the inferring process; and the last, the defuzzifying layer, which is the output layer has only one node that transform the fuzzy inputs into a crisp number. Table A.1 shows the parameters that determine the hybrid architecture.

Table A.1 The parameters that determine the hybrid architecture.

Description	Symbol	Number
Number of inputs	IN	2
Number of outputs	OUT	1
Number of linguistic labels for each input	MF	2
Number of nodes in the input layer	#(L0)	2
Number of nodes in the fuzzifying layer	#(L1) = IN * MF	4
Number of nodes in the inferring layer	#(L2) = MF ^ IN	4
Number of nodes in the defuzzifying layer	#(L3) = OUT	1
Total number of nodes in the hybrid system	TN	11

From the above table we may also observe that the configuration of the hybrid system is determined by the numbers of input signals, output signals and linguistic labels used for each input.

The second part of the first step is to identify the modifiable parameters associated with each node. For the nodes in the input layer, the number of the modifiable parameters is zero because their node functions are ones, that is, they don't perform any transformation.

For the node in the fuzzifying layer, the modifiable parameters are the parameters of the membership function. The number of parameters varies according on the type of the membership function used. Usually, they represent the center, the height, and the width of the membership function. Bell-shaped, triangular, and trapezoid membership functions which are commonly used for conventional fuzzy control can also use here. We shall use bell-shaped membership functions to demonstrate the node functions in the fuzzifying layer.

$$\mu_i(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}} \tag{A.1}$$

where i is the node index, a_i, b_i, c_i are the parameters associated with node i. By modifying these parameters, we may change the location and the shape of the membership function. The learning procedure can help us to determine the proper parameters that construct a membership function represent the transformation of a crisp number into a corresponding linguistic label.

For the nodes in the inferring layer, the modifiable parameters represent the parameters for the membership function at the consequent part of a linguist rule. There are three types of membership function usually used for the consequent part of a linguistic rule [Jang 1993]. Type 3 fuzzy reasoning mechanism, which is a Sugeno type fuzzy rule, will be used here. In a Sugeno type fuzzy rule, the result of the consequent part is the linear combination of the input signals plus a constant term. Therefore, the number of the modifiable parameters equals to the number of input signals plus one.

$$f_i = p_i x_1 + q_i x_2 + r_i \tag{A.2}$$

where p_i, q_i, r_i are the modifiable parameters for the node i in the inferring layer, f_i represents the consequent part of the i-th linguist control rule.

For the node in the defuzzifying layer, the modifiable parameters are the weights associated with the linguistic control rules. Therefore, the number of the modifiable parameters is the number of the fuzzy linguistic rules. Table A.2 summarizes the identification of the parameters in the hybrid system.

Table A.2 The number of parameters in the hybrid system.

Parameters	Symbol	Number
Input layer	#(P0)	0
Fuzzifying Layer	#(P1)	3
Inferring Layer	#(P2) = IN + 1	3
Defuzzifying Layer	#(P3)	1

A.2 Step 2: Connectting the Nodes

The second step is to connect the nodes in the different layers. The node in the input layer have only fan-out signals to nodes in the fuzzifying layer representing the linguistic labels of the given node.

The node in the fuzzifying layer, representing a linguistic label of the given input variable, receives a signal from the node in the input layer and produces a fuzzy number represents the degree of membership to the given linguistic label. The fuzzy number is then fan out to the nodes in the inferring layer that contain the fuzzy number at their premise parts.

The nodes in the inferring layer represent fuzzy linguistic rules. Each node receives a fuzzy number from each input variable and produces a fuzzy output to the node in the output layer.

The node in the output layer summarizes the fuzzy outputs from the inferring layer and produces a crisp output. Figure 2.7 represents the connection of a two-input-one-output hybrid system.

A.3 Step 3: Training the Network

The third step is to develop the learning procedure to tune the parameters in the hybrid system. This step includes two phases: the forward pass and the backward pass. In the forward pass, each node produces a output signal based on the input signals and its node function. In the backward pass, each node generate a set of new parameters based on the learning algorithm. The following is the derivation of the forward pass of the learning procedure:

Layer 0 (the input layer): In layer 0, a input signal from the training data set is supplied as the output of a node representing the input variable.

$$O_i^{L0} = x_i^p \tag{A.3}$$

where i = *0 to #(L0)*, denotes the index of the node in the input layer, O_i^{LO} represents the output of node i in layer *L0*, the label of the input layer, and x_i^p is the j-th item of the p-th set of the training data.

Layer 1 (the fuzzifying layer): Layer 1 performs the fuzzification of a crisp input to a fuzzy number with some linguistic label.

$$O_i^{L1} = \mu_i\left(O_j^{L0}\right) = \frac{1}{1+\left[\left(\frac{O_j^{L0}-c_i}{a_i}\right)^2\right]^{b_i}} \tag{A.4}$$

where i = *0 to #(L1)*, denotes the index of the node in the fuzzifying layer, O_j^{L1} represents the output of node j in layer $L1$, the label of the fuzzifying layer, $\mu_i(\cdot)$ is the membership function for node i, which represents the fuzzy label i. There are several ways to determine the initial parameters a_i, b_i, c_i, such as by randomly initializing, randomly partitioning the input space, etc. One of the easiest approaches to obtain a set of reasonable initial parameters a_i, b_i, c_i is by equally partitioning the input space into *MF* fuzzy subspaces. These parameters may be adjusted through the updating procedure in the backward pass phase and modified to proper values.

Layer 2 (the inferring layer): Layer 2 performs the fuzzy reasoning process that produces the fuzzy consequences based on the fuzzy premises.

$$O_i^{L2} = \bar{t}_i f_i = \bar{t}_i f_i \tag{A.5}$$

where $\bar{t}_i$ is the normalized firing strength of rule i:ule i:

$$\bar{t}_i = \frac{t_i}{\sum_{i=0}^{\#(L2)} t_i} \tag{A.6}$$

with $t_i = \prod_{j=0}^{IN} O_{j\to i}^{L1}$ represents the firing strength of rule i. $O_{j\to i}^{L1}$ is the fan-in signal from node j in layer $L1$ to node i in layer $L2$.

There also exist several approaches to determine the initial parameters p_i, q_i, r_i at the consequent part, such as the matrix inversion algorithm, the Kalman filtering algorithm, the group method of data handling algorithm [Farlow 1984] and regression methods, etc. What these algorithms do is essentially to extract a set of parameters of an estimated model, based on the training data and the given firing strengths.

Layer 3 (the defuzzifying layer): Layer 3 performs the defuzzification process that transforms the fuzzy consequences into a crisp number.

$$O_i^{L3} = \sum_{j=0}^{\#(L2)} w_j O_j^{L2} \tag{A.7}$$

where w_j is the weight associated with rule j. The index j of the output node j may be omitted if there is only one output.

The backward pass of the training process starts with the calculation of the estimation error for the given training data, which is the difference between the desired output stored in the training data set and the network output, usually refers to the actual output. The objective function E is then calculated by squaring the estimation error.

Next is to calculate equation (2.27). Since equation (2.27) can be calculate from two separate part, we shall start with the first part, which is the partial derivative of E with respect to the output of the given node. For the node in layer L3,

$$\frac{\partial E}{\partial O^{L3}} = -\left(O^d - O^{L3}\right) \tag{A.8}$$

where O^d is the desired output.

For the nodes in layers L2 and L1,

$$\begin{aligned}\frac{\partial E}{\partial O_i^{L2}} &= \frac{\partial E}{\partial O^{L3}} \frac{\partial O^{L3}}{\partial O_i^{L2}} \\ &= -w_i\left(O^d - O^{L3}\right)\end{aligned} \tag{A.9}$$

and

$$\frac{\partial E}{\partial O_j^{L1}} = \sum_{i=0}^{\#[MF\hat{\ }(IN-1)]} \frac{\partial E}{\partial O_{i\leftarrow j}^{L2}} \frac{\partial O_{i\leftarrow j}^{L2}}{\partial O_j^{L1}} \tag{A.10}$$

where the sum is carried out only for those nodes in layer L2 that receive the input from node j in layer L1; and $\frac{\partial O_{i\leftarrow j}^{L2}}{\partial O_j^{L1}}$ represents the derivative of the output node i-th node in layer L2 with respect to node j in layer L1, that fan out the signal to node i inlayer2. The part can be expressed more detail as

$$\frac{\partial O_{i\leftarrow j}^{L2}}{\partial O_j^{L1}} = \frac{O_{i\leftarrow j}^{L2}\left\{\left[\sum_{i=0}^{\#(L2)}\left(\prod_{j=0}^{\#(IN)} O_j^{L1}\right)\right]-\left[\sum_{i=0}^{\#(L2)}\left(\prod_{k=0}^{\#(IN)} O_{k\to i}^{L1}\right)\right]\right\}}{O_j^{L1}\left[\sum_{i=0}^{\#(L2)}\left(\prod_{j=0}^{\#(IN)} O_j^{L1}\right)\right]} \tag{A.11}$$

The second part is to calculate the partial derivative of E with respect to the parameter in the given node. The general form is given by

$$\frac{\partial E}{\partial P_i^L} = \frac{\partial E}{\partial O_i^L}\frac{\partial O_i^L}{\partial P_i^L} \tag{A.12}$$

where P_i^L denotes the parameter P of node i in layer L, O_i^L represents the node that contains the parameter P. Equation (A.13) to (A.19) represents the partial derivatives of E with respect to the parameters w,p,q,r,a,b,c.

Layer $1(a,b,c)$:

$$\begin{aligned}\frac{\partial O_i^{L1}}{\partial a_i} &= \frac{2b\left(O_{j\to i}^{L0}-c\right)\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b-1}}{a^2\left[1+\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}\right]^2}\\ &= \frac{2b\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}}{a\left[1+\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}\right]^2}\end{aligned} \tag{A.13}$$

$$\frac{\partial O_i^{L1}}{\partial b_i}=\frac{-2\left[\ln\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)\right]\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}}{\left[1+\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}\right]^2}$$

$$=\frac{-\left\{\ln\left[\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^2\right]\right\}\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}}{\left[1+\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}\right]^2} \qquad \text{(A.14)}$$

$$\frac{\partial O_i^{L1}}{\partial c_i}=\frac{\frac{2b}{a}\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b-1}}{\left[1+\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}\right]^2}$$

$$=\frac{2b\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}}{\left(O_{j\to i}^{L0}-c_i\right)\left[1+\left(\frac{O_{j\to i}^{L0}-c_i}{a_i}\right)^{2b}\right]^2} \qquad \text{(A.15)}$$

Layer 2 (*p,q,r*):

$$\frac{\partial O_i^{L2}}{\partial p_i}=\bar{t}_i x_0=\bar{t}_i O_0^{L0} \qquad \text{(A.16)}$$

$$\frac{\partial O_i^{L2}}{\partial q_i}=\bar{t}_i x_1=\bar{t}_i O_1^{L0} \qquad \text{(A.17)}$$

$$\frac{\partial O_i^{L2}}{\partial r_i}=\bar{t}_i \qquad \text{(A.18)}$$

Layer 3 (w):

$$\frac{\partial O^{L3}}{\partial w_i} = O_i^{L2} \tag{A.19}$$

In the case of off-line training, we update the weights after all the training patterns are presented to the network. Therefore, the new parameters can be expressed as

$$P_{new} = P_{old} - \eta \sum_{p=0}^{P} \frac{\partial E_p}{\partial P} \tag{A.20}$$

where η denotes the learning rate, similar to the one in an BPNN. The learning rate is usually a constant. However, a non-constant learning is also used in some case.

The whole training process is repeated until the objective function is optimized or a prescribed training time is reached.

APPENDIX B

STABILITY ANALYSIS

B.1 Introduction

In the appendix, we shall discuss the stability of the control scheme employed in the chapter 4. A brief introduction is give here. In section B.2, the detail of the stability analysis is described.

The concept of stability for nonlinear systems is more complicated than that for linear systems. There are a large number of theories available in the literature. We shall briefly introduce Lyapunov's second method for stability analysis theory here.

The main idea of Laypunov's second method for stability is that a system has an asymptotically stable state if the value of the energy function of the system decays as time increases until it finally reaches the minimum value at the equilibrium state. The energy function is so-called Lyapunov function.

Theorem B.1 [Narandra and Annaswamy 1989]:

The equilibrium state of a system described by (B.1) is uniformly asymptotically stable if the Lyapunov function candidate V with continuous first partial derivatives with respect to time exists such that $V(0,t)=0$ and if the following conditions are satisfied:

(1) V is positive-defined;

(2) V is decrescent;

(3) V is radially unbounded;

(4) $\dot{V}$ is negative-defined;

$$y = f(y,t), \quad f(0,t)=0, \quad t \geq t_0 \tag{B.1}$$

B.2 Stability Analysis

We now consider the design of stable control law of the systems of the form described by the following equation, which is a continuous version of equation (3.5).

$$\dot{y} = f(y) + g(u) \tag{B.2}$$

where $y \in \Re$ is the state variable for measurement, $u \in \Re$ denotes the control input, f and g represent the unknown nonlinear functions. Without loss of generality, we shall assume the initial condition as $f(0) = 0$, i.e. $y = 0$ is an equilibrium point.

We further assume that the control objective is to regulate the plant output y to follow the desired trajectory y_m given by a reference model:

$$\dot{y}_m = -a_m y_m + b_m r \tag{B.3}$$

where $r \in \Re$ is the bounded reference input of the reference morel, $y_m \in \Re$ denotes the output of the reference model, $a_m > 0$, and b_m are known scalar constants. Since the control objective is to regulate the plant output y to follow the desired trajectory y_m given by a reference model, we may define the control error, or called tracking error e_c as

$$e_c = y_m - y \tag{B.4}$$

and design a control law to make e_c tend to zero with time. Since an indirect adaptive control scheme is used in the example in chapter 4, we don't use e_c directly to design the control law. Instead, we use the estimated plant parameters to design the controller based on the certainty equivalence principle. The control input is then has the form

$$u = g^{-1}\left[-\hat{f}(y, \theta_f^*) - a_m y + b_m r\right] \tag{B.5}$$

where g^{-1} denotes the inverse of the unknown function g, and $\hat{f}$ represents the estimate of the unknown function f, and θ_f^* is a set of the optimal parameters of the identification model that estimates the unknown function f.

$$\theta_f^* \equiv \arg_{\theta_f} \min\left[\sup\left|\hat{f}(y, \theta_f) - f(y)\right|\right] \tag{B.6}$$

Rewrite equation (3.4) by substituting (B.2), (B.3), and (B.5)

$$\dot{e}_c = -a_m e_c + \left[\hat{f}(y,\theta_f^*) - f(x)\right] + \left[\hat{g}(u,\theta_u^*) - g(u)\right] + \left[\hat{g}^{-1}\left[g(u),\theta_{g(u)}^*\right] - g^{-1}\left[g(u)\right]\right] \quad \text{(B.7)}$$

where $\hat{g}$ represents the estimate of the unknown function g, θ_u^* is a set of the optimal parameters of the identification model that estimates the unknown function g, $\hat{g}^{-1}$ represents the estimate of g^{-1}, $\theta_{g(u)}^*$ is a set of the optimal parameters of the identification model that estimates g^{-1}.

$$\theta_u^* \equiv \arg_{\theta_u} \min\left[\sup\left|\hat{g}(u,\theta_u) - g(u)\right|\right] \quad \text{(B.8)}$$

$$\theta_{g(u)}^* \equiv \arg_{\theta_{g(u)}} \min\left\{\sup\left|\hat{g}^{-1}\left[g(u),\theta_{g(u)}\right] - g^{-1}\left[g(u)\right]\right|\right\} \quad \text{(B.9)}$$

The initial condition can also be expressed as

$$e_c(0) = y_m(0) - y(0) \quad \text{(B.10)}$$

Now we shall proceed with the synthesis of the control law using Lyapunov's stability theory [Åström and Wittenmark 1989]. Choosing the Lyapunov function candidate as

$$(e_c) = \frac{1}{2}e_c^2 \quad \text{(B.11)}$$

where γ is a positive constant that refers to the learning rate in the adaptive control law.

The time derivative of the Lyapunov's function candidate V is given by

$$\begin{aligned}\dot{V}(e_c,\Phi) &= -a_m e_c^2 + e_c\left[\hat{f}(y,\theta_f^*) - f(y)\right] + e_c\left[\hat{g}(y,\theta_u^*) - g(u)\right] \\ &\quad + e_c\left[\hat{g}^{-1}\left[g(u),\theta_{g(u)}^*\right] - g^{-1}\left[g(u)\right]\right] \\ &\leq -a_m e_c^2 + \beta\left|e_c\right|\end{aligned} \quad \text{(B.12)}$$

where β is defined as

$$\begin{aligned}\beta &= \left|\hat{f}(y,\theta_f^*)\right| + \left|f(y)\right| + \left|\hat{g}(y,\theta_u^*)\right| + \left|g(u)\right| \\ &\quad + \left|\hat{g}^{-1}\left[g(u),\theta_{g(u)}^*\right] - g^{-1}\left[g(u)\right]\right|\end{aligned} \quad \text{(B.13)}$$

Therefore, for all $e_c > \beta / a_m$, we have $\dot{V} < 0$, which implies that the tracking error is decreasing.

REFERENCES

Åström, K. J., and B. Wittenmark. 1973. "On Self-Turning Regulators." *Automatica* 9: 185-199.

Åström, K. J., and B. Wittenmark. 1989. *Adaptive Control.* Reading. MA: Addison-Wesley.

Balestrino, A., G. DeMaria and L. Sciavicco. 1983. "An Adaptive Model Following Control System for Robotic Manipulators." *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control* 105: 143 - 151.

Barto, A. G. 1989. Connectionist Learning for Control: An Overview. *COINS Technical Reports 89-89.* Department of Computer and Information Science, University of Massachusetts.

Bejczy, A. K. 1974. "Robot Arm Dynamics and Control." *Technical Memorandum* 33-669, Jet Propulsion Laboratory.

Berenji, H. R. 1992. "A Reinforcement Learning-Based Architecture for Fuzzy Logic Control." *International Journal of Approximate Reasoning* 6.2: 267 - 292.

Berenji, H. R., and P. Khedkar. 1992. "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements." *IEEE Transactions on Neural Networks* 3.5: 724 - 740.

Bobrow, J. E., S. Dubowsky, and J. S. Gibson. 1983. "On the Optimal Control of Robotic Manipulators with Actuator Constraints." *IEEE Transactions on Automatic Control* 6:782 - 787.

Bulsari, A. 1993. "Some Analytical Solutions to the General Approximation Problem for Feedforward Neural Networks." *Neural Networks* 6: 991 - 996.

Chalam, V. V. 1987. *Adaptive Control Systems.* New York: Marcel Dekker.

Chen, F.-C. 1990. "Back-propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control." *IEEE Control System Magazine* 10.3: 44 - 48.

Chester, D. L. 1990. "Why Two Hidden Layers are Better then One." *Proceedings of the International Joint Conference on Neural Networks* (Washington, D.C., January 15 - 19, 1990) I: 265 - 268.

Craig, J. J. 1988. *Adaptive Control of Mechanical Manipulators.* Reading, MA: Addison-Wesley.

Craig, J. J. 1989. *Introduction to Robotics: Mechanics and Control.* Reading, MA: Addison-Wesley.

Cybenko, G. 1989. "Approximation by Superpositions of a Sigmoidal Function." *Mathematics of Control, Signal, and Systems* 2: 303 - 314.

Dayhoff, J. E. 1990. *Neural Network Architectures: An Introduction.* New York: Van Nostrand Reinhold.

Dubowsky, S., and D. T. DesForges. 1979. "The Application of Model Referenced Adaptive Control to Robotic Manipulators." *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control* 101: 193 - 100.

Egeland, O. 1986. "On the Robustness of the Computed Torque in Manipulator Control." *Proceedings of IEEE International Conference on Robotics and Automation* :1203 - 1208.

Friendland, B. 1986. *Control System Design.* New York: McGraw-Hill.

Fu, K. S., R. C. Gonzalez, and C. S. G. Lee. 1987. *Robotics: Control, Sensing, Vision, and Intelligence*. New York: McGraw-Hill.

Funahashi, K.-I. 1989. "On the Approximate Realization of Continuous Mappings of Neural Networks." *Neural Networks* 2: 183 - 192.

Funahashi, K.-I., and T. Nakamura. 1993. "Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks." *Neural Networks* 6: 801 - 806.

Gomi, H., and M. Kawato. 1993. "Neural Network Control for a Closed-Loop System Using Feedback-Error-Learning." *Neural Networks* 6:933 - 946.

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

Gupta, M. M. ed. 1986. *Adaptive Methods for Control System Design.* New York: IEEE.

Hached, M. 1990. *A Variable Structure Control Approach to the Stabilization of Uncertain Dynamical Systems.* Doctoral Dissertation, Purdue University.

Higgens, C. M., and R. M. Goodman. 1994. "Fuzzy Rule-Based Networks for Control." IEEE Transactions on Fuzzy Systems 2.1: 82 - 88.

Hewit, J. R. 1979. "Decoupled Control of Robot Movement." *Electronics Letters* (IEE) 15.21: 670 - 671.

Hornik, K., M. Stinchcombe, and H. White. 1989. "Multilayer Feedforward Networks are Universal Approximators." *Neural Network* 2: 359 - 366.

Hunt, K. J., D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop. 1992. "Neural Networks for Control Systems - A Survey." *Automatica* 28.6: 1083 - 1112.

Hunt, L. R., and J. Turi. 1993. " A New Algorithm for Constructing Approximate Transformation for Nonlinear Systems." *IEEE Transactions on Automatic Control* 38.10: 1553 - 1556.

Isidori, A. 1989. *Nonlinear Control Systems: An Introduction*. Second Edition. New York: Springer-Verlag.

Jang, J.-S. R. 1992. "Self-Learning Fuzzy Controllers Based on Temporal Back Propagation." *IEEE Transactions on Neural Networks* 2.5: 714 - 723.

Jang, J.-S. R. 1993. "ANFIS - Adaptive-Network-Based Fuzzy Inference Systems." *IEEE Transactions on Systems, man, Cybernetics* 23.3: 665 - 685.

Kandel, A. 1986. *Fuzzy Mathematical Techniques with Applications*. Reading, MA: Addison-Wesley.

Kaufmann, A. 1975. *Introduction to the Theory of Fuzzy Subsets: Volume 1 Fundamental Theoretical Elements*. New York: Academic Press.

Kim, G., and R. Singh. 1993. "Nonlinear Analysis of Automotive Hydraulic Engine Mount." *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control* 115: 483 - 487.

Kim, K.-K., and Y.-S. Yoon. 1992. "Practical Incerse Kinematics of a Kinematically Redundant Robot Using a Neural Network." *Advanced Robotics* 6.4: 431 - 440.

Kokotovic, P. V. ed. 1991. *Foundations of Adaptive Control*. New York: Springer-Verlag.

Kosko, B. 1992a. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach To Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall.

Kosko, B. 1992b. "Fuzzy Systems as Universal Approximators." *Proceeding of IEEE International Conference on Fuzzy Systems 1992*, San Diego, March 8 - 12, 1992: 1153 - 1162.

Kosko, B. 1993. *Fuzzy Thinking: the New Science of Fuzzy Logic*. New York: Hyperion.

Landau, Y. D. 1979. *Adaptive Control: the Model Reference Approach.* New York: Marcel Dekker.

Lee, C. C.1990. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I & II.". *IEEE Transactions on Systems, Man, and Cybernitics* 20.2: 404-435.

Leeg, G. 1993. "Transmission's Fuzzy Logic Keep You on Track." *EDN* December 23:60 - 63.

Leshno, M., V. Ya. Lin, A. Pinkus, and A. Schocken. 1993. "Multilayer Feedforward Networks With a Nonpolynomial Activation Function Can Approximate Any Function." *Neural Networks* 6: 861 - 867.

Levin, A. U., and K. S. Narendra. 1992a. "Control of Nonlinear Dynamical Systems Using Neural Networks: Controllability and Stabilization." *Technical Report 9115*, Center For Systems Science, Yale University.

Levin, A. U., and K. S. Narendra. 1992b. "Stabilization of Non-linear Dynamical Systems Using Neural Networks." *Proceedings of the International Joint Conference on Neural Networks* (Baltimore, ML, June 7 - 1992) I: 275 - 280.

Levin, E., R. Gewirtzman, and G. F. Inbar. 1991. "Neural Network Architecture for Adaptive System Modeling and Control." *Neural Networks* 4: 185 - 191.

Li, W., and J.-J. Slotine. 1989. "Neural Network Control of Unknown Nonlinear Systems." *Proceedings of the 1989 American Control Conference* (Pittsburgh, PA, June 21 - 23, 1989) II: 1136 - 1141.

Lin, C.-T., and C. S. G. Lee. 1994. "Reinforcement Structure/Parameter Learning for Neural-Network-Based Fuzzy Logic Control Systems." *IEEE Transactions on Fuzzy Systems* 2.1: 46 - 63.

Lippmann, R. P. 1987. "An Introduction to Computing with Neural Nets." *IEEE ASSP magazine* 4.2: 4-22.

Lozano, R., and B. Brogliato. 1992. "Adaptive Control of Robot Manipulators with Flexible Joints." *IEEE Transactions on Automatic Control* 37.2: 174 - 181.

Mamdani, E. H. 1974. "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant." *Proceedings of IEE* 121.12: 1585 - 1588.

Miller, W. T., III, R. S. Sutton, and P. J. Werbos. ed. 1990. *Neural Networks for Control.* Cambridge, MA: MIT Press.

Mizumoto, M. 1988. "Fuzzy Controls under Various Fuzzy Reasoning Methods." *Information Sciences* 45: 129 - 151.

Mizumoto, M. 1989. "Improvement Methods of Fuzzy Control." *Proceedings of the 1989 IFSA Congress* (Seattle, Washington, August 6 - 11, 1989): 60 - 62.

Mizumoto, M. 1992. "Fuzzy Controls with Fuzzy Rules of Emphatic and Suppressive Types." *Proceedings of the 1992 International Fuzzy Systems and Intelligent Control Conference*: 134 - 149.

Narendra, K. S., and A. M. Annaswamy. 1989. *Stable Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall.

Narendra, K. S., and K. Parthasarathy. 1990. "Identification and Control of Dynamical Systems Using Neural Networks." *IEEE Transactions on Neural Networks* 1.1: 4 - 27.

Newman, W. S., and K. Souccar. 1991. "Robust, Near Time-Optimal Control of Nonlinear Second-Order Systems: Theory and Experiments." *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control* 113: 363 - 370.

Nguyen, D. H., and B. Widrow. 1990. "Neural Networks for Self-Learning Control Systems." *IEEE Control System Magazine* 10.2: 18 - 23.

Ortega, R., C. Canudas, and S. I. Seleme. 1993. "Nonlinear Control of Induction Motors: Torque Tracking with Unknown Load Disturbance." *IEEE Transactions Automatic Control* 38.11:1675 - 1680.

Paul, R. P. 1981. *Robot Manipulators: Mathematics, Programming, and Control*. Cambridge, MA: MIT Press.

Pedrycz, W. 1993. *Fuzzy Control and Fuzzy Systems*. Second, Extended Edition. Taunyon, Somerset, England: Research Studies Press.

Polycarpou, M. M., and P. A. Ioannou. 1991. "Identification and Control of Nonlinear Systems Using Neural Network Models: Design and Stability Analysis," *Technical Report 91-09-01*, Department of Electrical Engineering - Systems, University of Southern California.

Procyk, T. J., and E. H. Mamdani. 1979. "A Linguistic Self-Organizing Process Controller." *Automatica* 15.1: 15-30.

Ruck, D. W., et. al. 1992. "Comparative Analysis of Backpropagation and the Extended Kalman Filter for Training Multilayer Perceptrons." *IEEE Transactions on Patern Analysis and Machine Intelligent* 14.6: 686 - 691.

Rumelhart, D. E., J. L. McClelland, and the PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. Cambridges, MA: MIT Press.

Sanner, R. M., and J.-J. E. Slotine. 1991a. "Direct Adaptive Control with Gaussian Networks." *Proceedings of the 1991 Automatic Control Conference* (Boston, Massachusetts, June, 1991) III: 2153 - 2159.

Sanner, R. M., and J.-J. E. Slotine. 1991b. "Stable Adaptive Control and Recursive Identification Using Radial Gaussian Networks." *Technical Report NSL-910901.* Nonlinear System Laboratory, Massachusetts Institute of Technology.

Sanner, R. M., and J.-J. E. Slotine. 1992. "Gaussian Networks for Direct Adaptive Control." *IEEE Transactions on Neural Networks* 3.6:837 - 863.

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

Seraji, H. 1987. "A New Approach to Adaptive Control of Manipulators." *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control* 109: 193 - 202.

Shao, S. 1988. "Fuzzy Self-Organizing Controller and Its Application for Dynamic Processes." *Fuzzy Sets and Systems* 26: 151 - 164.

Slotine, J.-J. E., and W. Li.. 1991. *Applied Nonlinear Control.* Englewood Cliffs, NJ: Prentice-Hall.

Sun, C.-T. 1994. "Rule-Base Structure Identification in an Adaptive-Network-Based Fuzzy Inference System." *IEEE Transactions on Fuzzy Systems* 2.1: 64 - 73.

Takagi, T., and M. Sugeno. 1985."Fuzzy Identification of Systems and Its Applications to Modeling and Control." *IEEE Transaction on Systems, Man, and Cybernetics* 15.1: 116 - 132.

Tzirkel-Hancock E., and E. Fallside. 1991a. "Stable Control of Nonlinear Systems Using Neural Networks." *Technical Report CUED/F-INFENG/TR.81*. Engineering Department, Cambridge University.

Tzirkel-Hancock E., and E. Fallside. 1991b. "A Direct Control Method for a Class of Nonlinear Systems Using Neural Networks" *Technical Report CUED/F-INFENG/TR.65*. Engineering Department, Cambridge University.

Venugopal, K. P., R. Sudhakar, A. S. Pandya. 1994. "An Improved Scheme for Direct Adaptive Control of Dynamical Systems Using Backpropagation Neural Networks." to appear in *Journal of Circuits, Systems, and Signal Processing.*

Wang, F.-Y., and H. M. Kim. 1994. "Implementing Adaptive Fuzzy Logic Controllers with Neural Networks: Design Paradigms." in Special Issue on Fuzzy Logic and Neural Networks of *Journal of Intelligent and Fuzzy Systems.*

Wang, L. 1992. "Fuzzy Systems are Universal Approximators." *Proceeding of IEEE International Conference on Fuzzy Systems 1992*, San Diego, March 8 - 12, 1992: 1163 - 1169.

Wu, K., S. Outangoun, S. S. Nair. 1992. "Modeling and Experiments in Fuzzy Control." *Proceeding of IEEE International Conference on Fuzzy Systems 1992*, San Diego, March 8 - 12, 1992: 725 - 731.

Zadeh. L. A. 1965. "Fuzzy Sets." *Information and Control* 8: 338 - 353.