ABSTRACT

REAL TIME CONTROL OF NONLINEAR DYNAMIC SYSTEMS USING NEURO-FUZZY CONTROLLERS

by
Amitava Jana

The problem of real time control of a nonlinear dynamic system using intelligent control techniques is considered. The current trend is to incorporate neural networks and fuzzy logic into adaptive control strategies. The focus of this work is to investigate the current neuro-fuzzy approaches from literature and adapt them for a specific application. In order to achieve this objective, an experimental nonlinear dynamic system is considered. The motivation for this comes from the desire to solve practical problems and to create a test-bed which can be used to test various control strategies. The nonlinear dynamic system considered here is an unstable balance beam system that contains two fluid tanks, one at each end, and the balance is achieved by pumping the fluid back and forth from the tanks.

A popular approach, called ANFIS (Adaptive Networks-based Fuzzy Inference Systems), which combines the structure of fuzzy logic controllers with the learning aspects from neural networks is considered as a basis for developing novel techniques, because it is considered to be one of the most general framework for developing adaptive controllers. However, in the proposed new method, called Generalized Network-based Fuzzy Inferencing Systems (GeNFIS), more conventional fuzzy schemes for the consequent part are used instead of using what is called the Sugeno type rules. Moreover, in contrast to ANFIS which uses a full set of rules, GeNFIS uses only a limited number of rules based on certain expert knowledge. GeNFIS is tested on the balance beam system, both in a real-time actual experiment and the simulation, and is found to perform better than a comparable ANFIS under supervised learning. Based on these results, several modifications of GeNFIS are considered, for example, synchronous defuzzification through triangular as well as bell shaped membership functions. Another modification involves simultaneous use of Sugeno type as well as conventional fuzzy schemes for the consequent part, in an effort to create a more flexible framework. Results of testing different versions of GeNFIS on the balance beam system are presented.

REAL-TIME CONTROL OF NONLINEAR DYNAMIC SYSTEMS USING NEURO-FUZZY CONTROLLERS

by
Amitava Jana

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 Engineering

January 1996

APPROVAL PAGE

REAL TIME CONTROL OF NONLINEAR DYNAMIC SYSTEMS USING NEURO-FUZZY CONTROLLERS

Amitava Jana

12/22/95

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

12/29/95

Dr. David M Auslander, Advisor — Date
Professor of Mechanical Engineering, University of California, Berkeley

12/22/95

Dr. Rong-Yaw Chen, Committee Member — Date
Professor of Mechanical Engineering
and Associate Chairperson for Graduate Studies, NJIT

12/22/95

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

12/22/95

Dr. Bernard Koplik, Committee Member — Date
Professor and Chairperson of Mechanical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Amitava Jana

Degree: Doctor of Philosophy in Mechanical Engineering

Date: January 1996

Date of Birth:

Place of Birth:

Undergraduate and Graduate Education:

Doctor of Philosophy in Mechanical Engineering,
New Jersey Institute of Technology, Newark, NJ, 1996

Master of Science in Mechanical Engineering,
New Jersey Institute of Technology, Newark, NJ, 1987

Bachelor of Engineering in Mechanical Engineering,
Calcutta University, Calcutta, India, 1976

Major: Mechanical Engineering

Presentations and Publications:

Jana, A., and Auslander, D.M., "Workcell Programming Environment for Intelligent Manufacturing Systems," *Design and Implementation of Intelligent Manufacturing Systems*, Parsaei, H.R., and Jamshidi, M., Eds. Prentice-Hall, 1995 Chapter 1, pp. 1-18, 1995.

Jana, A., Auslander, D.M., Dave, R.N., Chehl, S.S.," Task Planning for Cooperating Robot Systems," *Proc. of 1994 Int. Symposium on Robotics and Manufacturing*, Maui, Hawaii, August 14-18, 1994.

Jana, A., Dave, R.N., Chehl, S.S., Wang, C. S.,"Computer-Integrated Manufacturing and Robotics Laboratory Design for Undergraduate Education," *Proc. of 1994 Int. Symposium on Robotics and Manufacturing*, Maui, Hawaii, August 14-18, 1994.

Jerro, H. D., Jana, A., Dave R. N., "Multiprocessing Systems: A Design Experience," *Proc. of 1994 ASEE GSW Conference*, Baton Rouge, LA, March 1994.

Jana, A., Wang, C. S., Chehl, S.S.,"Integrated System Design and Simulation," *Proc. of 1993 Int'l Simulation Technology Multiconference*, San Francisco, LA, November 8-10,1993.

Auslander, D.M.,Hanidu, G., Jana, A., Jothimurugesan, K., Seif, S., Young, Y.,"Tools for Teaching Mechatronics," *Proc. of 1993 ASEE Annual Conference*, Univ. of Illinois, IL, June 18-21,1993.

Jana, A., Chehl, S.S.,"Developing Instrumentation Laboratory with Real Time Control Component," *Proc. of 1993 American Control Conference*, San Francisco, June 1993, pp. 2046-2049.

Auslander, D.M., Hanidu, G., Jana, A., Landesberger, S., Seif, S., Young, Y.,"Mechatronics Curriculum in the Synthesis Coalition,"*Proc. of 1992 IEEE/CHMT Int'l Electronic Manufacturing Tech. Symp.*, Baltimore,MD,Sept 1992,pp.165-168.

Jana, A., Wang, C. S., Chehl, S.S.,"Development of a Flexible Manufacturing Workcell," Presented in the 9th Int'l Conf. on CAD/CAM, Robotics and Factories of the Future, Newark, NJ, August 18-20,1993.

Dave, R.N., Jana,A.,"Development of a PC-Based Robotic Simulation Package," *Proc. of 1987 ASME Computers in Engineering Conference, Vol-1*, pp.295-300.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisors Dr. Rajesh N. Dave of New Jersey Institute of Technology, and Dr. David M. Auslander of University of California, Berkeley. This work would not have been possible without their guidance and moral support throughout this research.

I am also grateful to Dr. Rong-Yaw Chen, Dr. Zhiming Ji, and Dr. Bernard Koplik for serving as members on my doctoral committee.

Special thanks to Dean M. Q. Burrel, Dr. S.S Chehl, and Dr. George Whitfield of Southern University, Baton Rouge, for their continuous support and encouragement.

I would also like to thank my friends Yang, Mark, Chris of UC Berkeley, Reza of Southern University, Sumit and Anupam of NJIT for their very helpful ideas, constructive comments and technical supports during various phases of this research.

Finally, and most importantly, I would like to thank my parents for the love and encouragement they have given me all my life and specially to reach this milestone of my life by sacrificing their happiness.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure | Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The field of intelligent control has emerged due to the need for increased autonomy in manufacturing, demand for intelligent manufacturing processes and intelligent products, and also to cope with the increased complexity and stringent performance requirement of modern control systems. The recent advancements in connectionist and linguistic based learning research offer opportunities for designing approximate reasoning based intelligent control systems and management. Artificial neural networks and fuzzy logic are two most significant areas related to the field of intelligent control. Artificial neural networks or simply neural networks (NN) were developed to emulate human brain's neural-synaptic mechanism which can learn and retrieve information [13]. On the other hand, fuzzy logic was developed to emulate human reasoning, which is not just two-valued or multivalued logic but the logic of fuzzy truths and are represented by linguistic terms like *high* or *low*.

In 1965, Zadeh suggested a modified set theory to characterize nonprobabilistic uncertainties, which he called fuzzy sets and developed a consistent framework for dealing with them [62]. Over the past few decades, fuzzy sets and their associated fuzzy logic have been applied to a wide range of multi-disciplinary problems. These include automatic control, pattern recognition and classification, consumer electronics, signal processing, management and decision making, operations research, data base management, and others. In the recent years, new research initiatives to integrate the field of neural network with fuzzy logic have been made, and a new research field known as neuro-fuzzy modeling and control has emerged.

In this thesis, the problem of real time control of a nonlinear dynamic system is considered. The emphasis is on the use of practical approaches which exploit good features from recently developed schemes that successfully combine neural networks and fuzzy logic controllers. There is an abundance of literature in the areas of conventional as well as modern control of nonlinear systems. However, in general most reported methods are based on either an assumption regarding linearization of the model or some knowledge about the type and order of nonlinearity [19,48]. For many industrial and manufacturing problems, the knowledge of the plant may be very limited and the task of utilizing the latest state-of-the-art technique from literature becomes formidable for a practicing engineer. This fact has led to the emergence of the area called intelligent control, and the use of alternate approaches such as neural networks and fuzzy logic controllers. The justification for using such an approach is based in part due to the fact that humans can achieve complicated control tasks without having an exact knowledge of the plant. The reasons for using neural networks or fuzzy logic then are quite natural, since most of the decision making by humans is based on the intrinsic use of logic and learning within human brain that perhaps is closely imitated by these two technologies. Both fuzzy logic and neural networks have been proven to be universal approximators [32], thus they become good candidates for tasks such as control of a nonlinear dynamic system, where the exact model is unknown and the system behavior must be understood from its input-output relations.

Ever since Mamdani [41] applied fuzzy logic for control of steam engine boiler combination, there has been a tremendous growth in application of fuzzy logic for controls, see for example [25,34,35]. Similarly, after pioneering work such as Narendra's [38,43,44] where neural networks were used for adaptive control, there has been an abundance of articles in a variety of journals and magazines, see [20] and references therein. In the next section, a very brief review of the prior work which is relevant to this thesis is presented. This is followed by the section that outlines the main objectives and the scope of this work.

1.1 Background

The use of neural networks (NNs) and fuzzy logic control (FLC) to solve the problem of controlling nonlinear dynamic systems has received attention from many researchers due to their potential in dealing with complex and nonlinear mappings. NN can map complex relations without an explicit set of rules and has a very good learning ability, on the other hand fuzzy logic can estimate functions and control systems with partial knowledge of the systems. Encouraging results of applying these methods for the control of complex systems are available in literature. Narendra and Parthasarathi used NN [43] in the problem of system identification and control of nonlinear systems. Takagi and Sugeno used fuzzy logic for system identification and control [55]. Although both the techniques have a great potential to solve the problem of controlling nonlinear systems, there are some drawbacks in each method. The architecture of NN depends on designer's experience, and there is no guideline to determine the number of layers or the number of nodes in each layer. In the case of FLC, the development of the linguistic rules and corresponding membership functions relies on the availability of expert knowledge, and this domain knowledge is often not available. Inspite of these limitations, these methods have complimenting strengths. For example, FLC provides a compact structure for rule representation that NN lacks, whereas NN provides structured learning ability which is not available with FLC. Recent research trend indicates a use of combined approach to overcome these limitations. The cooperative use of neural network and fuzzy logic are also appearing in consumer goods [64].

In NN driven fuzzy reasoning (NDF), Takagi and Hayashi used NNs to define membership functions[31]. Hayashi *et al* [33] proposed an algorithm that can adjust fuzzy inference rules to compensate for a change of inference environment. This neural network driven fuzzy reasoning with learning function (NDFL) can determine the optimal membership functions and obtain the coefficients of linear equations in the consequent parts by the searching function of the pattern search method. The authors used a computer controlled inverted pendulum system to test the control algorithm. The input output data were collected by manually balancing the pendulum.

Jang has developed an adaptive network-based fuzzy inference system (ANFIS), by using linear functions in the consequent part of the fuzzy inferencing rules[17]. He used a hybrid learning approach that combines the gradient descent method and least-squares estimator for fast identification of parameters. He also proposed a self learning method using temporal back propagation for model based adaptive control problems[16]. Park *et al*, proposed a controller design method for an on-line self-organizing fuzzy logic controller without using any plant model [29]. The controller is developed from the concept of human learning process and called as fuzzy auto-regressive moving average (FARMA). Berenji and Khedkar proposed a generalized approximate-reasoning based intelligent control architecture which consists of action evaluation network (AEN), action selection network (ASN), and a stochastic action modifier (SAM) [15]. ASN is the fuzzy controller whose output and a reinforcement signal produced by neural network AEN are fed into SAM to generate final control action.

It appears that there are a number of good schemes that combine concepts from neural network and fuzzy logic. However, in most of them, few authors have used actual systems to test their results. Generally, they test their results through simulated examples. There appears to be a need for real-time testing of some of the more attractive schemes using an experimental test-bed, which is suitable for an academic environment. This is the main objective of this dissertation. In the next section, specific objectives are outlined.

1.2 Objective and Scope of the Work

As mentioned earlier, the main objective of this dissertation is the study of the problem of real time control of a nonlinear dynamic system. In order to achieve this objective, a real experimental nonlinear dynamic system is considered. The focus is to investigate the current neuro-fuzzy approaches from literature and adapt them for the specific application. The motivation for this comes from the desire to solve practical problems and to create an experimental test-bed which can be used to test various control strategies. The nonlinear dynamic system considered here is an unstable balance beam system that contains two fluid tanks, one at each end, and the balance is achieved by pumping the fluid back and forth from the tanks. This system is in many ways similar to the ubiquitous inverted pendulum system, since both are examples of unstable nonlinear fourth order nonlinear dynamic systems. However, it is perhaps a more realistic example of engineering control problems. This system is interfaced to a personal computer and various control schemes are applied for its balance. This system is also simulated through its known dynamic equations for making detailed observations regarding controller performance.

Neuro-fuzzy inference systems used for control of dynamic systems are considered in chapter 2. The chapter begins with the introduction to fuzzy inference systems, fuzzy control schemes, and neural networks. This is followed by a brief description of neuro-fuzzy controller schemes popular in the literature. This discussion naturally leads to a conclusion that ANFIS (Adaptive networks-based fuzzy inference systems) is one of the most general framework for representing such schemes. The specific version of ANFIS as described by Jang [17] utilizes linear form of what is called the Sugeno type rules in the consequent part of the inference system. Although the use of these rules along with a minimization of least-squared error based approach to learning the consequent parameters results in an extremely fast learning algorithm, it may be worthwhile to explore other more conventional fuzzy schemes for the consequent part, albeit at a certain loss of speed of learning. This is the motivation for development of a neuro-fuzzy inference scheme, called Generalized Network-based Fuzzy Inferencing Systems (GeNFIS), which is introduced in this chapter. One disadvantage of ANFIS is that the structure of the network increases very rapidly with an increase in the number of inputs and the number of rules. One may utilize certain schemes (for example [18]) to alleviate this problem. However, in GeNFIS this problem is avoided by using some expert knowledge to specify a limited number of rules. The chapter ends with the derivation of the equations for the network output and back-propagation training.

In chapter 3, the balance beam system and the system model are described. The simulation of this system using a conventional PID controller is also presented. Real-time control of this system is presented in chapter 4. In this chapter, rule construction for GeNFIS is considered and the performance of this control scheme is studied.

Chapter 5, experimenting with GeNFIS, presents the test results of different ideas which have been implemented to improve the performance of GeNFIS.

Chapter 6 concludes this dissertation with summary of the research results and the directions for the future research.

The derivation of the proposed schemes are given in appendix A and B. The details of balance beam system parameters are included in appendix C.

CHAPTER 2

NEURO FUZZY CONTROL

2.1 Introduction

Control systems have been the most successful application of the fuzzy set theory and the fuzzy inferencing system. Fuzzy inferencing systems are also popularly known as fuzzy controllers, fuzzy-rule based systems or fuzzy associative memories. Ever since Prof. Lofti Zadeh introduced the fuzzy set theory in his seminal paper "Fuzzy sets", there have been a tremendous growth in the research of fuzzy logic and fuzzy set theory [62].

In this chapter, the basics of fuzzy logic and neural networks are briefly discussed. These include, the fundamental definitions and methods popularly used in these areas as well as the basic techniques to blend neural network and fuzzy inferencing systems for control applications. This is followed by the discussion on some of the more cited work in the field of neuro-fuzzy control and modeling. Finally, the proposed neuro-fuzzy controller is presented.

2.2 Fuzzy Inferencing Systems

The fundamental idea of the theory of fuzzy sets is that the human reasoning is not just two-valued or multivalued logic but the logic of fuzzy truths. Fuzzy sets are the extension of crisp sets which allow partial memberships, whereas crisp sets allow only full membership or no membership [9].

Fuzzy Set: A fuzzy set, A for a set of objects of interest $X = \{x_1, x_2, x_3 \ldots x_n\}$ is defined as a set of ordered pairs

$$A = \{(x_i, \mu_A(x_i), \quad i = 1,2,3 \ldots n\} \tag{2.1}$$

The variable $\mu_A(x_i)$ is a real number in the interval of $[0,1]$ and called a membership function. The value of the membership function or MF in short, represents the membership grade or truth value of x_i in A. A subset of X for which the value $\mu_A(x_i)$

of each element is positive, is called the support of A. The value $\mu_A(x_i) = 1$ indicates that the support x_i is completely in A. Similarly $\mu_A(x_i) = 0$ indicates that x_i does not belong to A. The X is generally referred as the "Universe of Discourse" and can also have continuous values.

In general, for a fuzzy set A in X for continuous membership function μ_A with universe of discourse X is represented by

$$A = \{(x, \mu_{A/x}(x)) | x \in X\}$$

$$\mu_A : X \rightarrow [0, 1] \tag{2.2}$$

In case of crisp sets the membership values are:

$$\mu_A(x) = \begin{cases} 1 \textit{ if } x \in A \\ 0 \textit{ if } x \notin A \end{cases} \tag{2.3}$$

Like crisp sets, fuzzy sets are also subject to fundamental set operations performed on the membership functions. Ordinary set operations, like intersection, union, and complement are also extended to the fuzzy set operations. Let A, B, and C be the three fuzzy sets with corresponding membership functions μ_A, μ_B, and μ_C respectively. Then the following fuzzy set operations can be defined:

Union (OR or Triangular conorms): The Union C of A and B is represented by $C = A \cup B$ and corresponding membership functions are related by

$$\mu_C(x) = \max\{\mu_A(x), \mu_B(x)\}$$

$$= \mu_A(x) \vee \mu_B(x) \tag{2.4}$$

Intersection(AND or T-norm): The Intersection of A and B is C is represented by $C = A \cap B$ and corresponding membership functions are related by:

$$\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\}$$

$$= \mu_A(x) \wedge \mu_B(x) \tag{2.5}$$

Complement: The complement of A is defined by $\overline{A}$ or $\neg A$ is defined by

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

The basic techniques of developing a fuzzy logic controller are the selection of *if-then* type rules with linguistics variables, and to find the suitable control actions by combining the output of each rule. This process is also known as fuzzy reasoning. These rules are constructed from the domain of human expert's knowledge. The selection of right control parameters and proper levels of linguistic variables are needed to construct *if-then* type rules. For example, in a simple rule like *'If x is A and y is B Then z is C'*, x, y, and z are linguistic variables with corresponding values like { *High, Medium, Low*}. *High*, *Medium*, and *Low* are the set of membership functions for variable x. A convenient way to express this rule by human experts may be:

"**If** Outside temperature is {*Low*} **AND** the Room temperature is {*High*}

THEN Run the AC {*Medium Low*}.

In propositional logic, two very important rules are frequently used for inferencing. They are known as *Modus Ponems* and *Modus Tollems*. *Modus Ponems* are used for forward inferencing whereas *Modus Tollems* are used for backward inferencing. These two concepts are also extended to fuzzy logic and are known as *Generalized Modus Ponems* (GMP) and *Generalized Modus Tollems* (GMT). In fuzzy logic, fuzzy reasoning is mostly based on GMP fuzzy inference rules.

GMP: With fuzzy sets denoted by A, B, and C the GMP has the form

$$\begin{array}{ll} \textit{premise 1:} & \textit{if } x \textit{ is } A \textit{ and } y \textit{ is } B \textit{ then } z \textit{ is } C \\ \underline{\textit{premise 2:}} & \underline{x \textit{ is } A' \textit{ and } y \textit{ is } B'} \\ \textit{consequence:} & z \textit{ is } C' \end{array} \tag{2.7}$$

In this case, if A and A' are the fuzzy sets in the universe of discourse U, and B and B' are fuzzy sets in the universe of discourse V, and C and C' are the fuzzy sets in the universe of discourse W, then for a given input signal (x, y) the fuzzy consequence C' is evaluated by taking the 'max-min' composition (operator $\circ$) of the fuzzy relation *([A and B] →C)* in $U \times V \times W$ and the fuzzy set (A' and B') in $U \times V$. This can be written as:

$$C = ([A \text{ and } B] \rightarrow C) \circ (A' \text{ and } B') \tag{2.8}$$

or
$$\mu_{C'}(z) = \text{max–min}\left\{\left[(\mu_A(x) \wedge \mu_B(y)) \rightarrow \mu_C(z)\right], \left[\mu_{A'}(x) \wedge \mu_{B'}(y)\right]\right\} \tag{2.9}$$

Again the relation *([A and B] →C)* can be transformed into a ternary fuzzy relation R and can be specified by:

$$\mu_R(x, y, z) = \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z) \tag{2.10}$$

Thus the relation (2.9) can be written as

$$\begin{aligned}\mu_{C'}(z) &= \vee_{x\,y}\left[\mu_{A'}(x) \wedge \mu_{B'}(y)\right] \wedge \left[\ \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)\ \right] \\ &= \vee_{x,y}\left[\mu_{A'}(x) \wedge \mu_{B'}(y) \wedge \mu_A(x) \wedge \mu_B(y)\right] \wedge \mu_C(z) \\ &= \left\{\vee_x\left[\mu_{A'}(x) \wedge \mu_A(x)\right]\right\} \wedge \left\{\vee_y\left[\mu_{B'}(y) \wedge \mu_B(y)\right]\right\} \wedge \mu_C(z)\end{aligned} \tag{2.11}$$

If w_1 is the degree of match between A and A', evaluated from the operation

$$\left\{\vee_x\left[\mu_{A'}(x) \wedge \mu_A(x)\right]\right\}$$

and if w_2 is the degree of match between B and B', evaluated similarly from

$$\left\{\vee_y\left[\mu_{B'}(y) \wedge \mu_B(y)\right]\right\}$$

then the relation (2.11) can be written as:

$$\mu_{C'}(z) = w_1 \wedge w_2 \wedge \mu_C(z) \tag{2.12}$$

$w_1 \wedge w_2$ is called the firing strength of the rule or consequence C' [25].

2.2.1 Fuzzy Control

Conventional controllers, both linear and nonlinear, are derived from control theory based on mathematical models of the systems to be controlled. Linear controllers are the mapping of n input state vectors of a process and the control action to a hyperplane of $(n+1)$ dimensions. Nonlinear controllers are very difficult to synthesize, and this difficulty is the key factor in the research of alternative control synthesis techniques, such as Fuzzy Logic Controllers (FLC) [12].

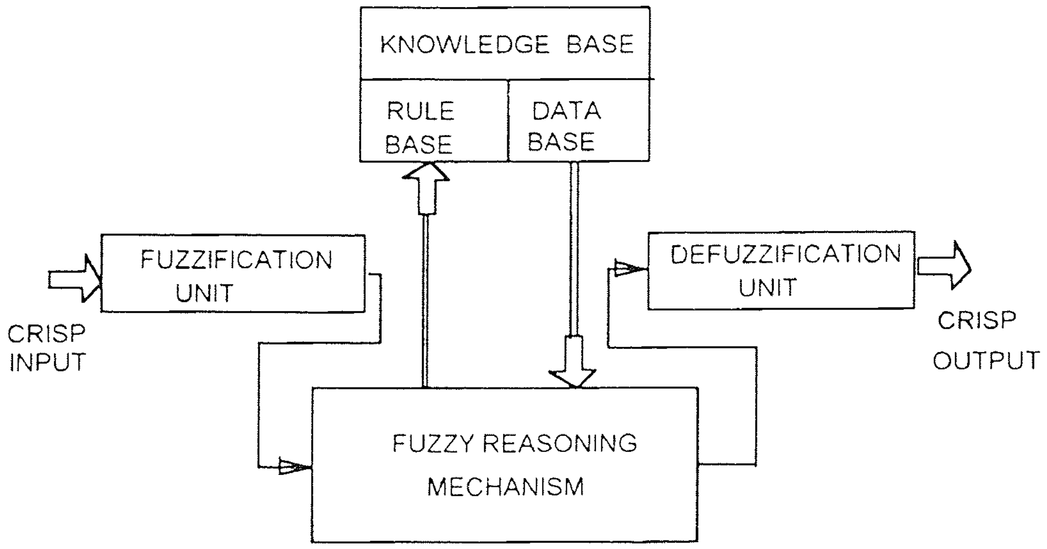


Figure 2.1 Fuzzy Controller

FLC's are knowledge-based controllers, usually developed from the process operator's or a product engineer's prior knowledge or automatically synthesized from self-organizing control architectures in the form of *if-then* rules. Essentially a fuzzy logic controller consists of four main elements as shown in Figure 2.1. Fuzzification unit converts the input crisp data to the corresponding fuzzified value in the respective universe of discourse. The knowledge base module of fuzzy controller consists of two submodules; rulebase and database. The rulebase part of the knowledge base consists of number of *if-then* rules to establish the control relationships. The rule base maps fuzzy values of the input to fuzzy values of the output, whereas database defines the membership functions of the fuzzy sets, used as values for each system variable. Fuzzy reasoning mechanism performs fuzzy inference to determine the fuzzy control actions by fuzzified inputs. The final crisp control action is inferred through defuzzification unit by combining the calculated outputs of each rule.

Ever since Mamdani [41] applied fuzzy set theory to control a steam engine and boiler combination by a set of rules, there have been several fuzzy inferencing systems proposed by various researchers reported in literature [34,35,27]. The popular methods, which are related to this work will be discussed here.

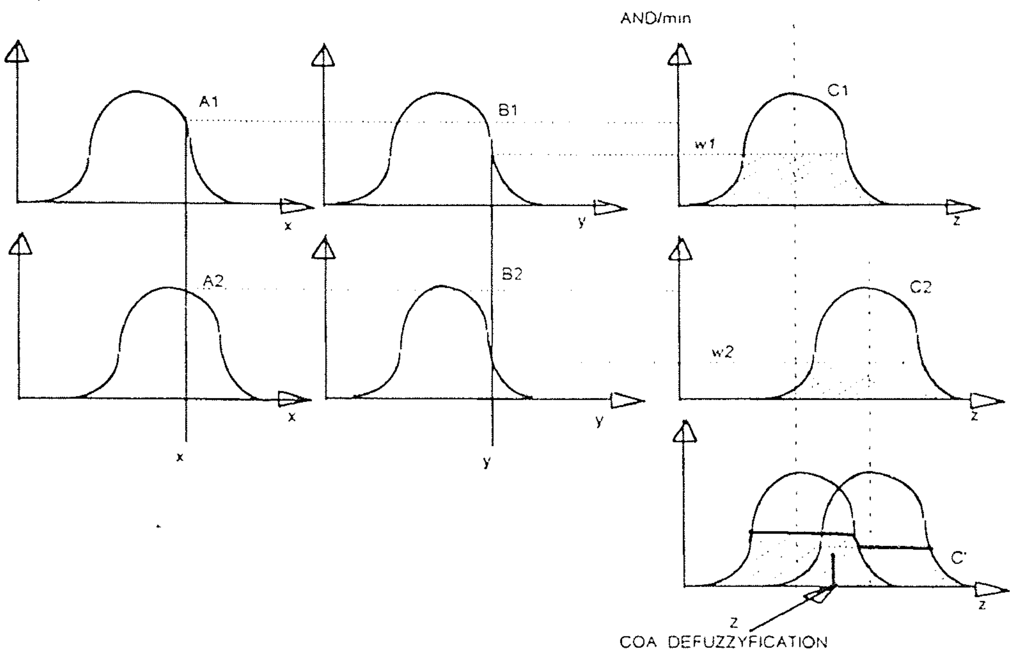


Figure 2.2 Fuzzy Inferencing; Mamdani Type

Mamdani type: In Mamdani type fuzzy inference system, the resultant control action of two rules is shown in Figure. 2.2. In this case, the resulting action is based on *min max* composition. The final crisp value is obtained by calculating the centroid of area. This process of defuzzification is known as center of area (COA) defuzzification method. Other frequently used defuzzification methods mentioned in the literature are: mean of maximum (MOM), largest of maximum, bisector of area etc. All these strategies are computation intensive and there is no systematic way to evaluate them except through experiments [25]. As Mendel mentioned in his tutorial paper on fuzzy logic systems, "Many defuzzifiers have been proposed in the literature; however, there are no scientific bases for any of them (i.e. no defuzzifier has been derived from a first principle, such as maximization of fuzzy information or entropy); consequently, defuzzification is an art rather than a science" [42].

In Figure 2.2, bell shaped membership functions are used with COA defuzzification method. The first part of Figure 2.2 shows fuzzification and *min* (AND) operations to compute the firing strength of each rule, while the second part shows *max* (OR) operation. Mamdani also used product operation to substitute AND or *min* operation, keeping the *max* operation as before.

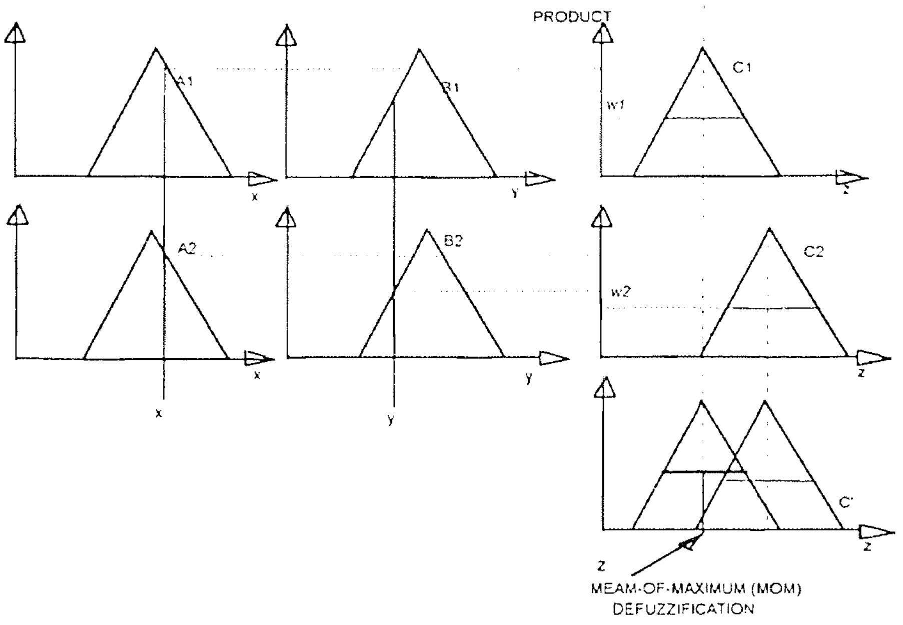


Figure 2.3 Symmetrical MF, Product Operator, MOM Defuzzification

Figure 2.3 shows the product operation with mean-of-maximum (MOM) defuzzification strategy on symmetrical membership function. In case of COA defuzzification, the final control action can have any value (continuous) between the centers of two output membership functions (C_1 and C_2), whereas in MOM defuzzification the final control action will oscillate (jump around or discrete) between the centers of consequent membership functions.

Tsukamoto type: Figure 2.4 shows Tsukamoto type fuzzy model for the same rules as discussed in the Mamdani fuzzy model. Here the operations on premise parts i.e. fuzzification and *min* operations, are same as before. However, Tsukamoto used monotonical membership functions in the consequent part [58]. The overall control action is the weighted average of each rule's crisp output. Although the consequent membership functions are not compatible with linguistic terms such as "*medium*" whose membership function should be bell shaped [27], this method is computationally efficient.

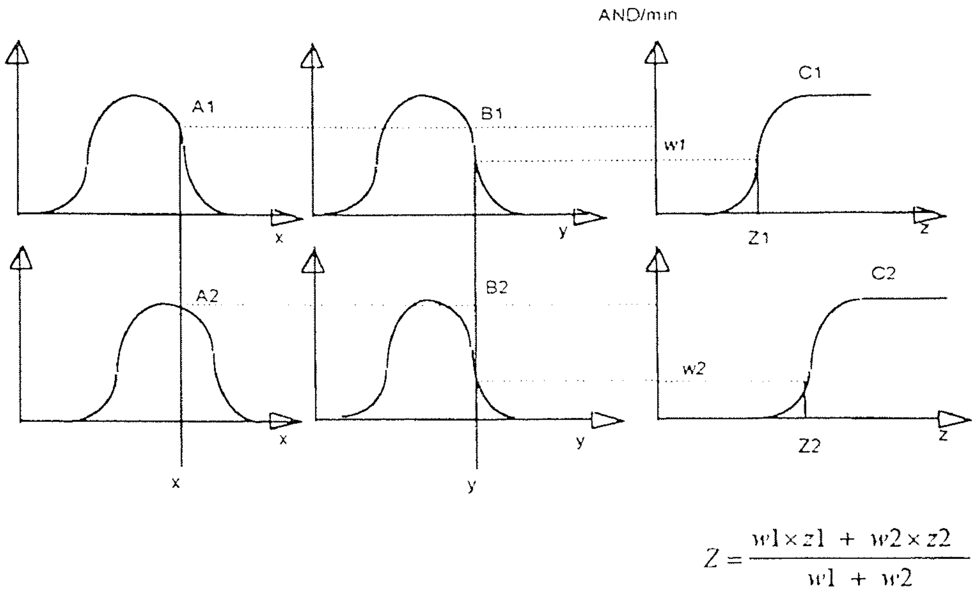


Figure 2.4 Tsukamoto Type Fuzzy Inferencing

Sugeno type: In Sugeno type fuzzy model, also known as the TSK fuzzy model, the output of each fuzzy rule is evaluated by a crisp function. The final control action is the weighted average of each rule's crisp output. This model was originally proposed by Takagi, Sugeno and Kang [52,55]. For a three input fuzzy inferencing system the typical output of a rule is given by:

$$\textbf{if } x_1 \text{ is } A \text{ and } x_2 \text{ is } B \text{ and } x_3 \text{ is } C \textbf{ then } y = f(x_1, x_2, x_3) \qquad (2.13)$$

where, the function $y = f(x_1, x_2, x_3)$ represents the consequent part of the rule, which is a crisp function of the crisp input variables x_1, x_2, x_3. The premise part of the rule is same as discussed in other types. The computation of rule firing strength and fuzzification methods are also similar to other models. If the function $y=f(.)$ is a first order polynomial, the model is called first-order Sugeno model [52, 55, 25]. If the function $y=f(.)$ is a fuzzy singleton or a constant, the model is known as zero-order sugeno model. The most of the work on adaptive network-based fuzzy inference system (ANFIS) architecture (will be discussed later) are based on first order Sugeno model [26, 27, 29, 53]. Figure 2.5 shows the fuzzy inferencing procedure using two inputs and two rules for a first order Sugeno fuzzy model. In this figure, the antecedent membership functions are of trapezoidal shape.

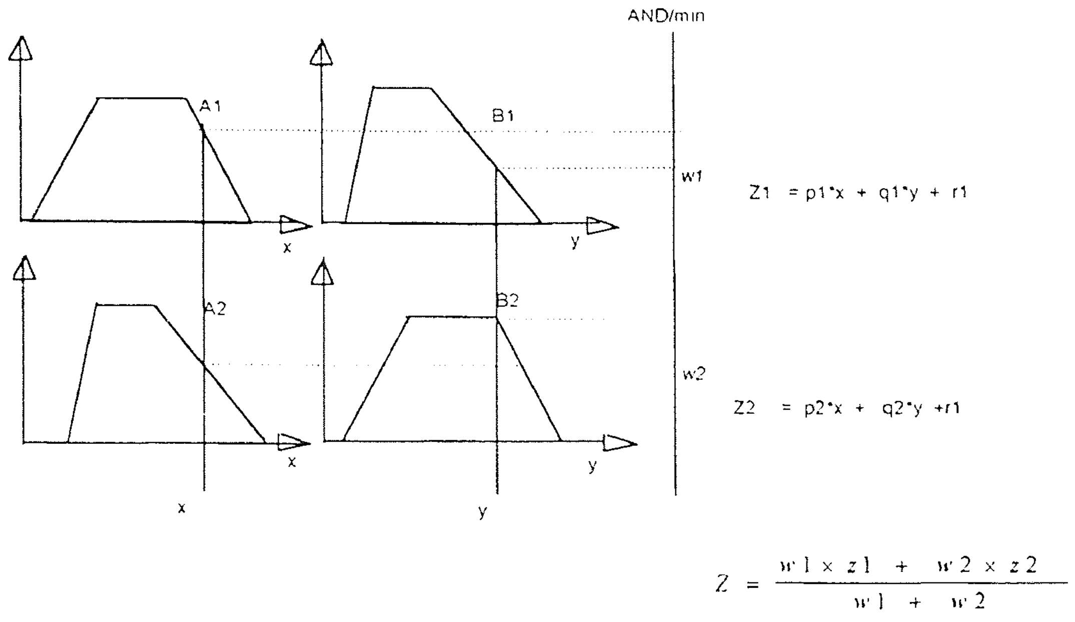


Figure 2.5 Sugeno Type Fuzzy Inferencing System

In the output side p_1, q_1, and r_1 are the constants for *rule* 1 and p_2, q_2 and r_2 are the constants for *rule* 2. The value of these constants have to be determined before the application. The main disadvantage in this type of fuzzy model is that it is very difficult to assign linguistic variables to the consequent part.

2.3 Neural Networks

A neural network (NN) is a structure that contains several neuron-like processing elements connected together. A multilayered feed forward neural network is shown in Figure 2.6. Each neuron receives several input signals which are then modified by the interconnection weights and summed up to a single result. This result is then modified by a transfer function, also known as activation function, and is then transmitted to the output path.

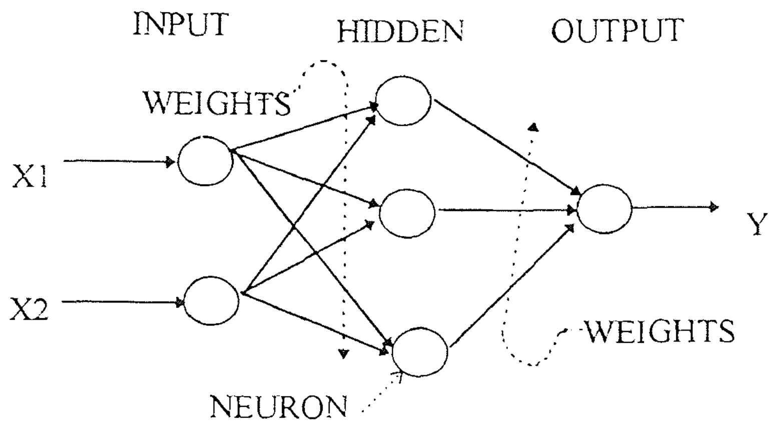


Figure 2.6 Multilayered Neural Net

In Figure 2.6, one hidden layer is shown between input and output layers. There may be many intermediate hidden layers, but each neuron in hidden units must send its output to a forward layer and must receive its input from a layer behind. Figure 2.7 shows the activities of a neuron or processing element. The *i*th input to a neuron is x_{ji}, and the corresponding weight is w_{ji}. The activation function g(.) may be a sigmoid ($1/(1+e^{-x})$), hyperbolic tangent (*tanh(x)*) or *sinh(x)*, etc. A bias is may also be added to the sum.

For a given input vector, output vector will be computed by processing the input vector layer by layer through each neuron until the output layer is reached. Each neuron will use the following relation for computation:

$$y_i = g\left(\sum w_{jii} x_{ji}\right) \tag{2.14}$$

In most of the applications, sigmoid is used as function $g()$. There is no standard method to determine the number of layers or the number of neurons in each layer of a neural network. However, it has been shown that one hidden layer is enough to represent any standard function [18].

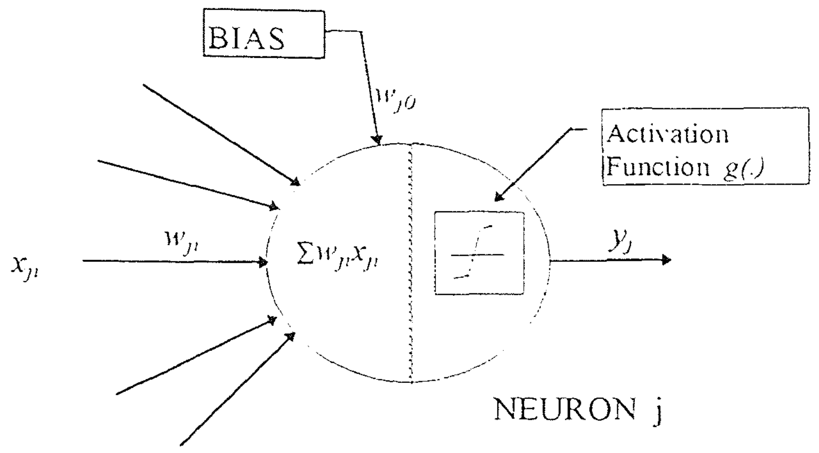


Figure 2.7 Processing element--Neuron

The learning of network is done by two phases: the forward pass and the backward pass. In forward pass, the input is presented to the input layer and is fed forward from layer to layer until the output is obtained. The output is compared with the desired output and an error term is computed. In backward pass, this error is fed back to the input layer and the weights are updated to minimize the error. The algorithm can be described as follows:

For n training samples, the objective is to minimize total error E

$$E = \sum_{i=1}^{n} \left(Y_d^i - NN_w(X^i)\right)^2 \tag{2.15}$$

where, X^i and Y_d^i are input and desired output. $NN_w(X^i)$ is network output which depends on the weights w of network NN. E is the mean squared error of the network output and is differentiable over w. By minimizing E using gradient descent method, we get the weight update equation as:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{2.16}$$

where η is the learning rate. The partial derivative of weights for each layer is computed by chain rule [31].

2.4 Neuro Fuzzy Controller

Neural networks and fuzzy systems are universal approximators. As stated before, NN can map complex relations without an explicit set of rules, while fuzzy systems can estimate functions and control systems with only a partial description of system behavior [31]. Recent research on applying NN and FLC techniques in the control of highly complicated systems has shown encouraging results [65]. Although both NN and FLC are independently useful for controlling nonlinear systems, each method has some limitations. NN are very slow in learning and also need sufficient amount of training data to map a relation. FLC needs a large number of rules which are often not available. Consequently, recent research trend is to combine both the techniques in order to overcome the limitations of individual schemes.

The basic concept of most of the hybrid controllers (Neuro-fuzzy controller) is to design a FLC whose rules can be modified using NN learning techniques. In addition, some of the reported hybrid controllers provide the facilities for the structure identification. In this section, the research growth of the neuro-fuzzy controllers as well as some popular schemes to blend NN concepts with fuzzy logic controllers are reviewed.

Kosko developed a fuzzy associative memory system, popularly known as FAM [31] to map fuzzy input sets to fuzzy output sets. The system consists of a set of rules and a set of weights associated with rules. By feeding the system with the training data, the firing frequency of each rule is calculated. The weights are then modified by comparing the firing frequency of each rule with a prescribed threshold value. Thus the learning process determines a set of weights which can produce an optimal association of a fuzzy output to a fuzzy input. The scheme doesn't allow any modification of the membership functions and requires a large number of training cycles for learning. However, the scheme provides a way to find the number of rules required for mapping. The FAM can be treated as a FLC and has no direct relation with NN, other than the concept of weights and training. Figure 2.8 shows the FAM architecture with one input and one output.

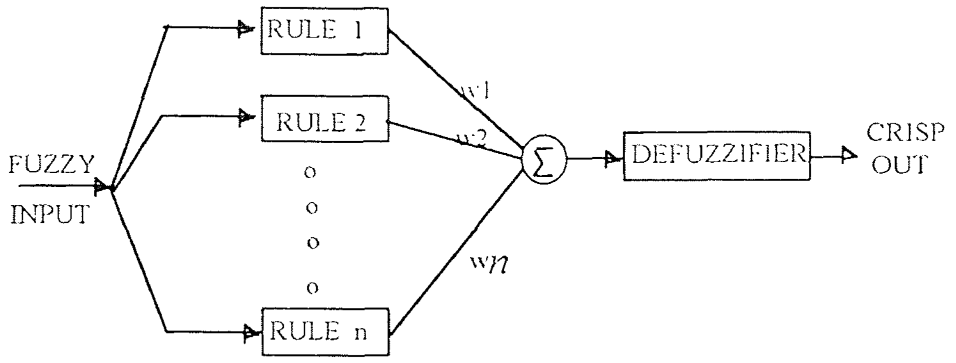


Figure 2.8 Fuzzy Associative Memories (FAM)

Jang has developed a neuro fuzzy controller known as Adaptive Network-based Fuzzy Inference System (ANFIS) [26,27] which can modify the parameters of the membership functions of fuzzy control rules. Although several other researchers, like Lin and Lee [40] and Wang and Mendel [59], independently proposed similar types of neuro fuzzy frame work, Jang's main contribution is in the development of a hybrid learning algorithm which combines the gradient descent method and least-squares estimators for fast identification of parameters. However, this hybrid learning scheme is mostly suitable for Sugeno type rules where each rule output is a linear function of input variables. Figure 2.9 shows the ANFIS architecture with two inputs and one output.

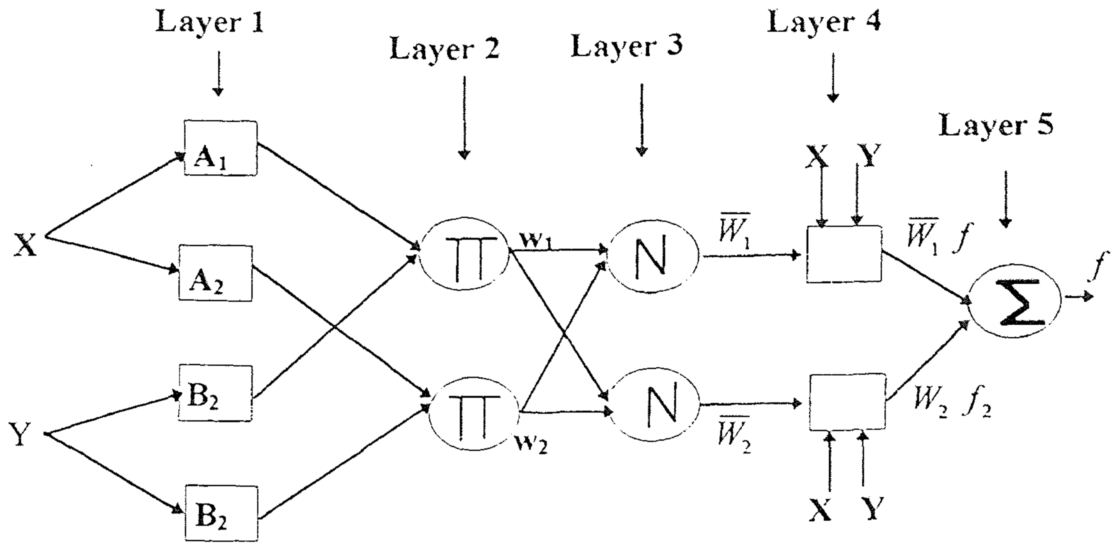


Figure 2.9 Adaptive network based fuzzy inferencing system (ANFIS)

ANFIS provides a very good approach for parameter identification for an FLC. The only problem of the ANFIS is the structure of the network increases exponentially with the increase in number of inputs and number of rules [53]. To overcome this problem, recently Jang has proposed a novel approach to determine the structure of ANFIS [28]. Jang also developed a self learning method for ANFIS controller on the basis of temporal back propagation [26].

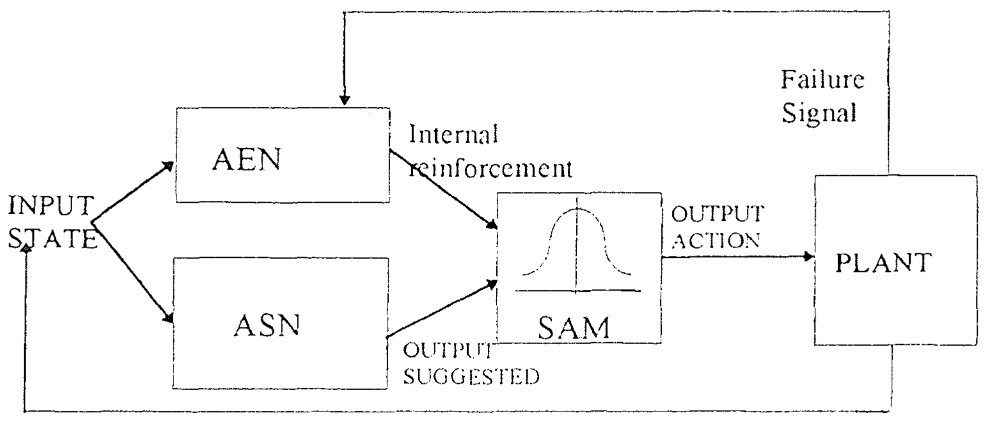


Figure 2.10 GARIC

Supervised learning algorithms for neural networks and neuro-fuzzy controllers require precise training data sets for identification of weights and parameters. This precise training/learning data are generally difficult and expensive to obtain for some real-world applications. For this reason, reinforcement learning algorithms are initially developed for NN [39,40]. In case of reinforcement learning, the training data are not precise like supervised learning, instead they are evaluative. Using reinforcement learning paradigm, Berenji and Khedkar proposed a generalized approximate reasoning-based intelligent control architecture (GARIC). The GARIC architecture consists of three main elements: the action selection network (AEN), the action evaluation network(ASN), and a stochastic action modifier (SAM). The ASN is a fuzzy controller which maps a state vector into a recommended action. The AEN is a two-layer NN used to produce an internal reinforcement based on a given state and failure signal. The SAM uses both recommended action and internal reinforcement to produce a final output which is applied to the plant. The learning takes place by fine-tuning the weights of AEN and the parameters describing membership functions of ASN using reinforcement learning algorithm. Figure 2.10 shows the architecture of GARIC.

Although GARIC has been reported as an effective tool to control nonlinear dynamic systems, the main problem in practical implementation is the need to determine the structure of ASE. Lin and Lee [39], independently proposed a similar reinforcement neural-network-based fuzzy logic control systems (RNN-FLCS) like GARIC to solve various reinforcement learning problems.

Recently, Chang has proposed a scheme known as Fuzzy Logic Adaptive Network (FLAN) by combining some features of ANFIS and FAM [14]. The FLAN is basically the ANFIS structure with the weights associated with each rule as mentioned in FAM. The performance of FLAN is comparable with ANFIS and in some situations training time of FLAN is less than ANFIS. Figure 2.11 shows the architecture of FLAN. Chang has used

FLAN to identify nonlinear dynamic systems with unknown parameters using the identification models from Narendra and Parthasarathy [43].

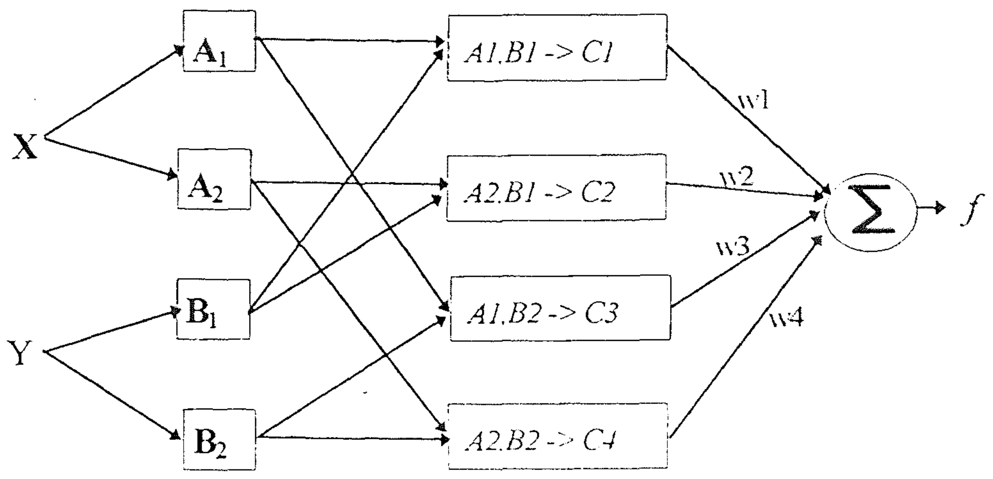


Figure 2.11 Fuzzy logic adaptive network (FLAN)

Although all of the above mentioned schemes are very important, it is evident that more efficient hybrid controller can be developed by cleverly combining certain good features from the above methods. The main objective of this research is to develop a generalized scheme to design an adaptive FLC and to apply the controller on a nonlinear engineering system to study the performance. ANFIS, as discussed earlier, can be treated as a generic framework, and in that sense appears to be an excellent basis for an improved neuro-fuzzy controller. However, Sugeno type rules with hybrid learning scheme [26,27] may result in unbounded, nonphysical defuzzification. To avoid this problem of defuzzification, more conventional fuzzy schemes for the consequent part are used instead of using Sugeno type rules. The proposed FLC is trained by using the learning concepts of NN. The back propagation algorithm, which is the most popular for the training of NN is used to train the proposed controller. The proposed controller is also used on a nonlinear dynamic system to study its performance.

2.5 Proposed Neuro-Fuzzy Controller

As discussed in the section 2.4, the main thrust in the research of neuro-fuzzy controller is to find a novel method to structure the fuzzy inferencing systems in the form of a node based network with differentiable parameters. This will allow the network to train by using a suitable back propagation algorithm available in the neural network literature. There should be enough flexibility to accommodate multiple input with various combination of rules. In the first part of this dissertation, a network architecture suitable to represent all types of fuzzy model is developed. Since the philosophy behind this architecture is to blend fuzzy inferencing system with neural network, the proposed structure resembles action selection network (ASN) of GARIC [9], and ANFIS [26]. However, provisions are provided to incorporate new concepts resulting from experimental part of this research. The findings of the experimental research and the subsequent modifications will be discussed in the next chapters. Since this proposed network will be used to incorporate strengths of various independently developed neuro-fuzzy networks, hereafter this network will be referred as Generalized Network based Fuzzy Inferencing System or in short GeNFIS.

2.5.1 GeNFIS Architecture

GeNFIS is a five layer network as shown in Figure 2.12. Each layer consists of several nodes which perform specified action to represent fuzzy inferencing mechanism. For simplicity a two input three rule network is considered for illustration. Although the rules selected here are of Mamdani type with bell shaped membership functions, the final control action is very similar to Tsukamoto type with suitable modifications. In Figure 2.12, k_i is the output of rule i, and is given by:

$$k_i = \frac{w_i\left(\mu^{-1}(w_i)\right)}{\sum w_i} \tag{2.17}$$

where, w_i is the rule firing strength of rule i.

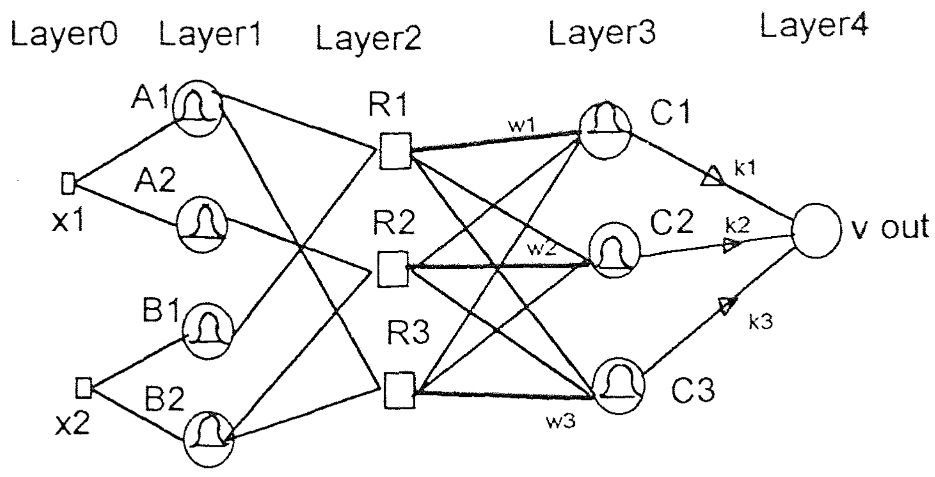


Figure 2.12 GeNFIS Architecture

Layer 0: This is the input layer where each node represents the real-valued state variable or a computed value from the state variable like position error or velocity error (x_i). The input output relation is simply

$$O_i^{L0} = x_i \tag{2.18}$$

where O_i^{L0} is crisp value of crisp input x_i.

Layer 1: This is the antecedent layer, where each node is a value of corresponding linguistic input variable. In this layer, the input crisp variables are fuzzified using respective modifiable parameters of membership functions defined in that particular node. The output of a layer 1 node is given by

$$O_i^{L1} = \mu_{A1}(O_i^{L0}) \tag{2.19}$$

where O_i^{L1} is output of node i of layer 1. A_j represents the linguistic value. A bell shaped membership function is used here to compute the fuzzified value of input. The modifiable parameters of this function are $\{a_i, d_i, g_i\}$, where a_i is spread, d_i controls curvature, and g_i is center of the curve.

$$\mu_{A1}\left(O_i^{L1}\right)=\frac{1}{1+\left[\dfrac{O_i^{L0}-g_i}{a_i}\right]^{2d_i}} \tag{2.20}$$

Layer 2: This layer computes AND or *min* operations to evaluate the value of *if* part of each rule. A differentiable *softmin* operator is used here [9] to perform the T-norm operation. The output of this layer is given by

$$O_i^{L2} = w_i = \frac{\sum_i \mu_i(O_i^{L0})e^{-k\mu_i(O_i^{L0})}}{\sum e^{-k\mu_i(O_i^{L0})}} \tag{2.21}$$

where k is a constant, controls the hardness of the *softmin* operation, and for $k=\infty$, the original *min* operator is recovered [9]. In ANFIS a *product* operator is used [27]. These operators (*product* or *softmin*) are suitable for computing derivatives for backpropagation learning algorithm.

Layer 3: Each node in this layer represents the consequent part of the rule. A bell shaped membership function is used here, which has three modifiable parameters $\{a, c, b\}$. The evaluation of membership function at the label i is given by

$$\mu_i(\mu_i^{-1}(w_i) = \frac{1}{1+\left[\dfrac{x-c}{a}\right]^{2b}} \tag{2.22}$$

where $\mu_i^{-1}(w_i)$ is the defuzzified value.

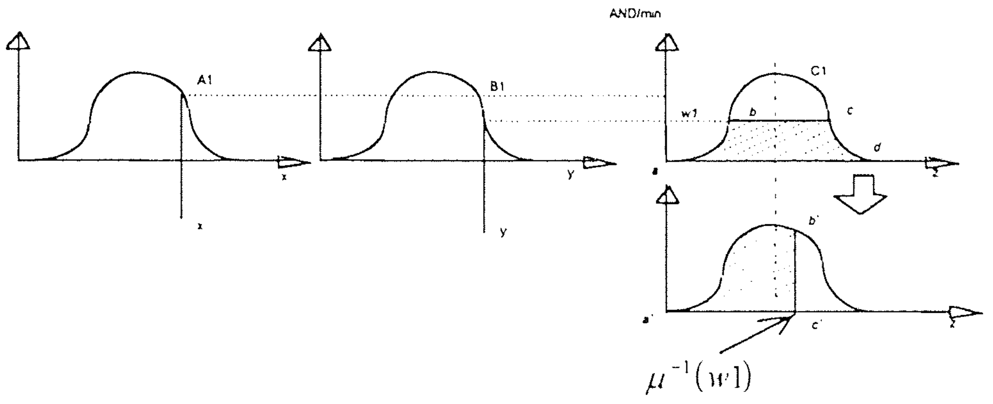


Figure 2.13 Single Rule Defuzzification

A local defuzzification method has been used at each label. This local defuzzifiction method (LDM) is suitable for a symmetrical membership function. For a symmetrical membership function, a local defuzzification method like LMOM (local mean-of-maximum) [9], will always yield a constant value. Hence LDM is developed from the concept of area of a membership function clipped by a single rule firing strength [25] and mapping this area from the left hand side as shown in Figure 2.13. In Figure 2.13, the clipped area *abcd* has been mapped as a'b'c' for defuzzification, and the final defuzzified value is c'. For a given rule firing strength, multiplying the clipped area by a suitable scale factor λ ($1.0 > \lambda \geq 0.5$) and by mapping the scaled area as before, we can set the upper limit of defuzzification.

Since the computation of area is complicated, a simple approach has been taken to approximate the area mapping concept of defuzzification. The defuzzified value or $\mu_i^{-1}(w_i)$ is the centroid of the right angled triangle formed by the left intersection between w_i and membership function as vertex, and a prespecified point on the Z axis, located on the other side of membership function, as shown in Figure 2.14. This prespecified point "Q" is a linear function of spread a.

$$CQ = l = f(a),$$

$$\text{or } l = k_{df} * a \tag{2.23}$$

where l is the distance of point Q from center C and k_{df} is a constant. By denoting length $UV = P$, we have

$$\mu_i^{-1}(w_i) = c + \frac{k_{df} * a}{3} - (2/3) * p$$

$$\text{or} \quad \mu_i^{-1}(w_i) = c + \frac{k_{df} * a}{3} - (2/3) * a\left[\frac{1 - w_i}{w_i}\right]^{\frac{1}{2b}} \tag{2.24}$$

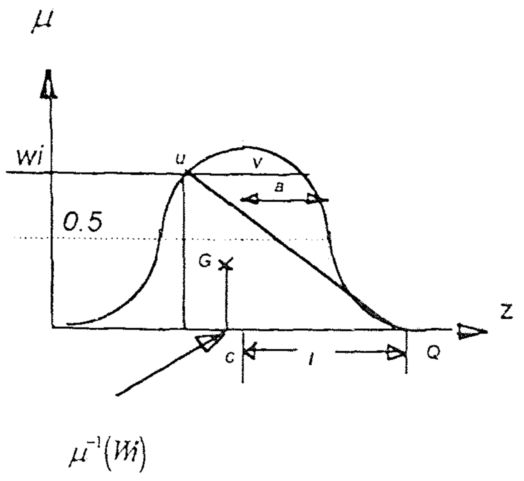


Figure 2.14 Local Defuzzification Method (LDM)

Once the defuzzified value is computed, the final node output value is obtained by multiplying $\mu_i^{-1}(w_i)$ with normalized rule firing strength t_{ni}. Where t_{ni} is given by

$$t_{ni} = \frac{w_i}{\sum_{i=0}^{\#of\ rules} w_i} \tag{2.25}$$

Layer 4: The output of the layer 4 is the final control action. The number of nodes in this layer is equal to number of outputs. Each input link of these nodes are associated with a weight j_i. The total output O_i of a node in this layer is given by

$$O_i^{L4} = \sum_{i=0}^{\#\ of\ rules} j_i\ O_i^{L3} \tag{2.26}$$

For generalized structure GeNFIS, the initial value of each weight is unity. In modified GeNFIS structure, which will be discussed later, these weights are used to blend output of the same rule but with different defuzzification scheme.

Learning. The output of GeNFIS is final control action and the inputs are the state variables at that time step. During learning, training data are presented in pairs of input and output. At the end of forward pass, output of GeNFIS is compared with the desired output and an error term E is computed by squaring the difference as:

$$E = \sum_{i=1}^{\#\,of\ data\ set} (O_i^{d} - O_i^{L4})^2 \tag{2.27}$$

$$\frac{\partial E}{\partial O_i^{L4}} = -2 * (O_i^{d} - O_i^{L4}) \tag{2.28}$$

An error rate for each layer is computed from E by using chain rule of derivatives. Equation (2.28) shows the error rate for output layer L4. In order to update modifiable parameters to implement gradient descent method, the partial derivative of E in parameter space is computed. In case of GeNFIS, the derivative of E with respect to the parameters of layer 1 is given by:

$$\frac{\partial E}{\partial P_j^{L1}} = \sum \frac{\partial E}{\partial O^{L4}} \cdot \frac{\partial O^{L4}}{\partial O^{L3}} \cdot \frac{\partial O^{L3}}{\partial O^{L2}} \cdot \frac{\partial O^{L2}}{\partial \delta O^{L1}} \cdot \frac{\partial O^{L1}}{\partial P_j^{L1}} \tag{2.29}$$

where, P_j^{L1} is the jth modifiable parameter of a particular node in layer 1 (L1). The derivative of the output of layers (O^{li})with respect to its preceding layers ($O^{l(i-1)}$) for GeNFIS structure are given as:

$$\frac{\delta O^{L4}}{\delta O^{L3}} = J_i \tag{2.30}$$

$$\frac{\partial O_i^{L3}}{\partial O_i^{L2}} = \frac{\mu^{-1}(O_i^{L2}) + O_i^{L2}\left[\mu^{-1}(O_i^{L2})\right]' - O_i^{L3}}{\sum O_i^{L2}} \tag{2.31}$$

$$\frac{\partial O_i^{L2}}{\partial O_i^{L1}} = \frac{e^{-kO_i^{L1}}(1 + k(O_i^{L2} - O_i^{L1}))}{\sum e^{-kO_i^{L1}}} \tag{2.32}$$

In GeNFIS structure, each node of layer 3 receives input from every nodes of layer 2 (see Figure 2.12). So the derivative of output of ith node in layer 3 $\left(O_i^{L3}\right)$ with respect to the output of jth node of layer 2 $\left(O_j^{L2}\right)$ is given as:

$$\frac{\partial O_i^{L3}}{\partial O_j^{L2}} = -\frac{O_i^{L3}}{\sum O_i^{L2}} \tag{2.16}$$

Using the equations (2.12) to (2.15) the derivative of overall error measure E with respect to each modifiable parameter of every nodes $\left(\partial E / \partial P_i^{Lj}\right)$ can be computed. The details of these derivations are given in Appendix A.1. Using these derivatives, the ΔP for updating of each parameter is computed as:

$$\Delta P_i^{Lj} = -\eta \frac{\partial E}{\partial P_i^{Lj}} \qquad (2.17)$$

where η is the learning rate.

CHAPTER 3

A NON LINEAR DYNAMIC SYSTEM

3.1 Introduction

Application of fuzzy logic in control of nonlinear dynamic systems is advantageous, in particular when the mathematical model of the plant under control is either not available or very complicated. Moreover, when the operating conditions of the plant vary significantly, designing a controller using conventional control theory becomes difficult. After Mamdani's first effort to control a steam engine and boiler combination by a set of linguistic control rules using the knowledge of experienced operators [25,41], a significant effort has been made by various investigators to apply fuzzy logic to industrial problems where the model of the plants are not available or ill defined. However, in the literature, most of the results reported on the research of neuro-fuzzy controllers are tested on simulation. Most newly proposed neuro-fuzzy controllers are evaluated through simulation of the bench mark problem of balancing an inverted pendulum to represent nonlinear dynamic system charectaristics [9,26,39,46,25]. However, in [56] an experimental setup of an inverted pendulum is used to test the proposed neural network driven fuzzy reasoning (NNDF) model. In this experiment, training data sets were collected by balancing the pendulum manually. Other than this work [56], few experimental studies on neuro-fuzzy controllers, suitable for academic environment are reported in the literature.

One of the major objective of this dissertation is to develop a neuro-fuzzy controller and to apply it in an experimental setup suitable for academic environment. In order to meet this objective a fluid beam balancing system is used as a test bed of non-linear dynamic system for this work. In this chapter the details of experimental setup, the model of the system, and its simulation using a conventional controller are presented.

3.2. Balance Beam System

The basic problem of the balance beam system is to balance a beam containing two fluid tanks, one at each end, by pumping the fluid back and forth from the tanks [37]. Figure3.1 shows the schematic diagram of the fluid beam balancing system. The beam is comprised of a wooden plank clamped on top of a shaft about which it can rotate. The shaft is supported by two low friction bearings, and at the one end of the shaft a Hall effect sensor is connected to measure angular position of the beam. The center of the mass of the complete system is above the center of rotation. This feature makes the system unstable. Figure 3.2 shows that the net torque due to disturbance is in the same direction of rotation.

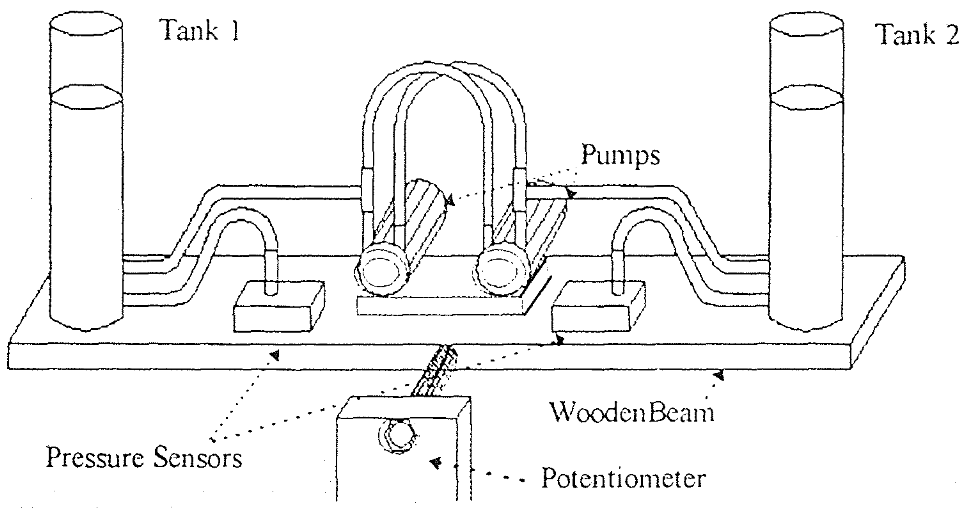


Figure 3.1 Balance beam system

Control effort is created by pumping water between two plastic tanks, thereby creating a moment due to weight imbalance. Two d.c. pumps powered by linear amplifiers are biased and connected in parallel to provide the pumping between the two tanks. The input/output characteristics of the pump shows that there exists a dead zone in the region of small input where input cannot incur effective output. To avoid this dead zone two pumps are used in parallel [37]. In addition to position measurement sensor, there are two pressure sensors to measure the mass of the liquid provided for each tank. Signal conditioning and calibration of the pressure readings provide necessary mass information.

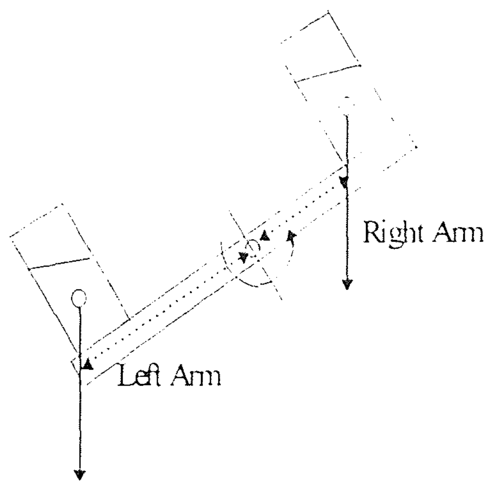


Figure 3.2 Net torque in the same direction of rotation

3.2.1 System Model

The balance beam system has been modeled as a fourth order nonlinear system by the following relations

$$\frac{dx_1}{dt} = x_2 \tag{3.1}$$

$$\frac{dx_2}{dt} = \frac{(-B * x_2 + T(x_1, h))}{J(h)} \tag{3.2}$$

$$\frac{dh}{dt} = \frac{Q}{A} \tag{3.3}$$

$$\frac{dQ}{dt} = \frac{(-Q + K_{pump}U)}{T_{pump}} \tag{3.4}$$

where

x_1 = angular position of beam

x_2 = angular velocity of beam

h= height of water in left tank

Q= flow rate of water

B= friction coefficient of bearing

$T(x_1,h)$ = torque due to water

$J(h)$= rotational moment of inertia of the system

A= area of tank

K_{pump}= motor constant of pump

T_{pump}= time constant of motor

U = output of controller (voltage)

The equations (3.1) and (3.2) are from the dynamics of beam, which is given by

$$J(h)\ddot{x}_1 + B\dot{x}_1 = T(x_1,h) \tag{3.5}$$

and the equation (3.3) is from the dynamics of tank. The fourth equation (3.4) is the equation of pump flow rate which has been modeled as a first order system with the following transfer function

$$\frac{Q(s)}{U(s)} = \frac{K_{pump}}{1+T_m s} \tag{3.6}$$

Using this equations, a state feedback control law is given in equation (3.7). Details of these equations and the values of the constants are provided in the Appendix C.

$$\begin{aligned} U(k) = {} & kp*(x_{1\text{-}ref}(k) - x_1(k)) + ki*\Sigma(x_{1\text{-}ref}(k) - x_1(k)) + \\ & kd*(x_{2\text{-}ref}(k) - x_2.estimate(k)) \\ & + km*(h(k) - h\_ref(k)) \end{aligned} \tag{3.7}$$

where

$x_{1\text{-}ref}$= position set point

$x_{2\text{-}ref}$= velocity set point

h_ref= equilibrium height of the left tank(i.e. water height needed to make $T(x_1,h) = 0$), which is given by

$$h\_ref = (-7.429 - 0.2238*H)*x_1 + H/2 \tag{3.7A}$$

$x_2 .estimate$ = estimated velocity=$(x_1(k+1) - x_1(k))/sample\_time$; H is the total water height. The cascaded control loops equivalent to the control law of equation (3.7) is given in Appendix.C.

3.2.2 Plant Simulation

For the purpose of simulation and also for collection of training data, the balance beam system has been modelled by fourth order Runge-Kutta method based on Simpson's 3/8th rule. The control input vector at time t is $\bar{u}(t)$, and the corresponding state variable vector $\bar{y}(t)$ of balance beam system consists of four state varables; position(x_1), angular velocity(x_2), left tank water height(h) and flow rate(Q).

$$\bar{y}(t) = [\bar{x}_1(t), \bar{x}_2(t), \bar{h}(t), \bar{Q}(t)]^T \tag{3.8}$$

$$\dot{\bar{y}}(t) = f(\bar{y}(t), \bar{u}(t), t) \tag{3.9}$$

If g is the sample time and k is the step number starting from initial condition at $t = 0$, then the state vector $\bar{y}(t)$ at $t = (k*g + g)$ or at the next time step is given as

$$\bar{y}(k*g+g) = \bar{y}(k*g) + (1/8)(\bar{r}_1 + 3*\bar{r}_2 + 3*\bar{r}_3 + \bar{r}_4) \tag{3.10}$$

where

$$\begin{aligned}
\bar{r}_1 &= g*\bar{f}(\bar{y}(k*g),\quad \bar{u}(k*g),\quad k*g)) \\
\bar{r}_2 &= g*\bar{f}((\bar{y}(k*g) + \bar{r}_1/3),\quad \bar{u}(k*g),\quad k*g + g/3)) \\
\bar{r}_3 &= g*\bar{f}((\bar{y}(k*g) + \bar{r}_1/3 + \bar{r}_2/3),\quad \bar{u}(k*g),\quad k*g + (2*g)/3)) \\
\bar{r}_4 &= g*\bar{f}((\bar{y}(k*g) + \bar{r}_1 - \bar{r}_2 + \bar{r}_3),\quad \bar{u}(k*g),\quad k*g + g)
\end{aligned} \tag{3.11}$$

The relations (3.1) to (3.2) are used for simulation. By tuning the control gains, it is observed that the control law given in equation (3.7) can balance the beam form the folowing initial conditions

at $t=0,\ \ x_1(t) = -0.03\ rad,\ x_2(t) = 0.0\ rads/sec$

$H = 10.4\ cm$ *and* h_ref *from equation (3.7A)*

where position set point ($x_{1\text{-}ref}$) is *0.0 rad* and velocity set point ($x_{2\text{-}ref}$) is *0.0 rads/sec*. In simulation the initial value of water height error *(h(t) - h_ref(t))* has been assumed to be zero, but in real application this is not true and very difficult to compute.

Figure 3.5 shows the system response in simulation using the above control law and initial conditions. The parameters used here are directly measured from the beam system and are given in Appendix C. Figure 3.4 (a) is the response of position error over a 6 second time span. A sample time of 0.01 sec is used in the simulation. Figure 3.4 (b) is the water level error of left hand tank which starts from zero. Figure 3.4 (c) and (d) are velocity error and final control action or motor input voltage respectively. Figure3.5 (a) is the plot of position error against velocity error and the Figure 3.5 (b) is the pump flow rate. Data from this simulation along with the simulation of system response for an initial beam angle of 0.03 rads are used to collect training data set for the proposed neuro fuzzy controller.

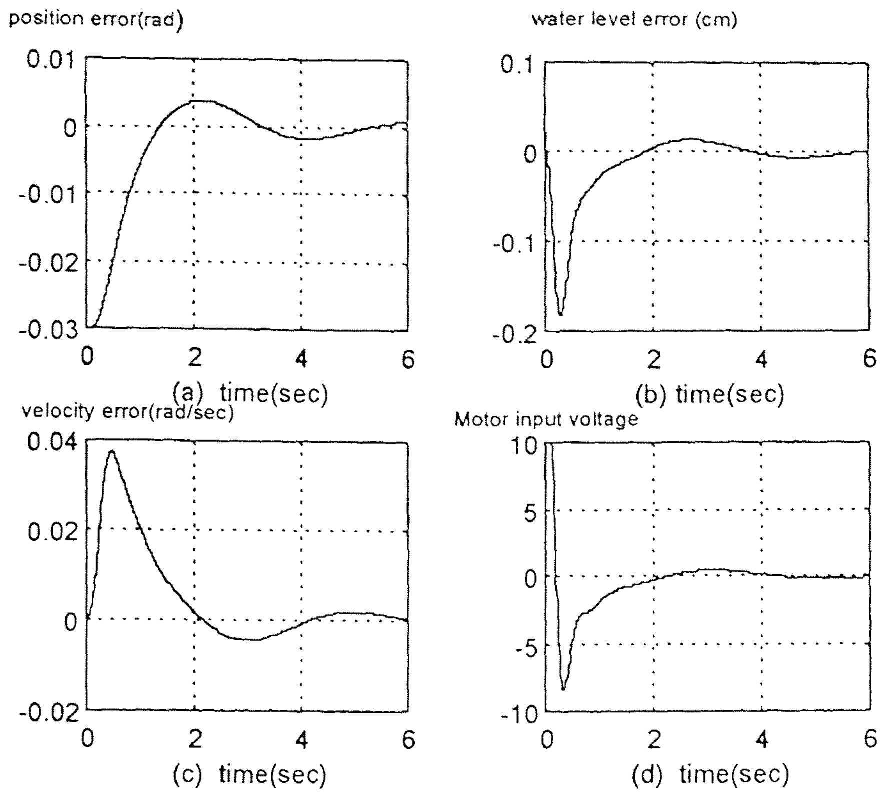


Figure 3.4 (a) Position error, (b) water level error of the left tank (c) velocity error (d) motor input voltage

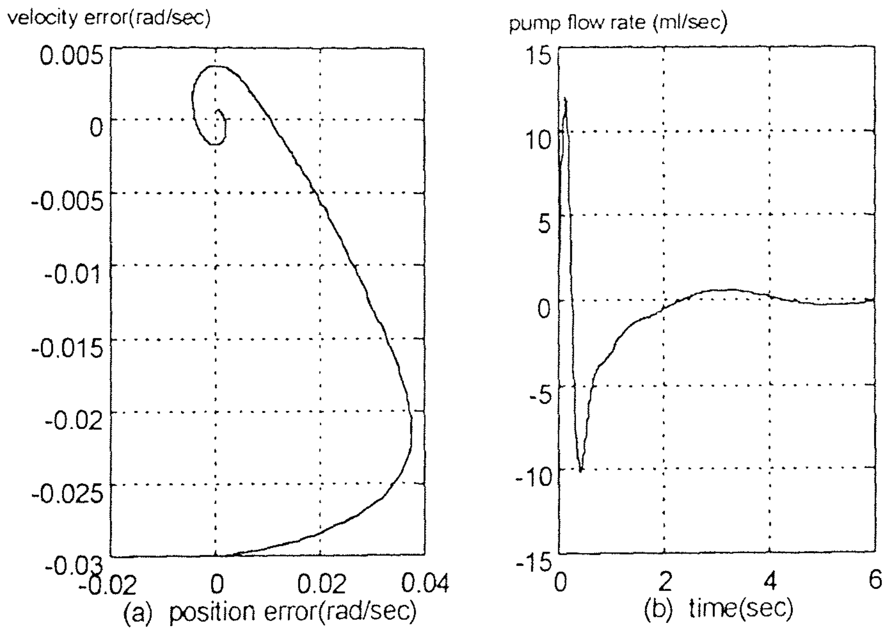


Figure 3.5 (a) Position error vs velocity error (b) Pump flow rate

CHAPTER 4

REALTIME CONTROL

4.1 Implementation.

The balance beam control system is implemented on a personal computer through a pc-based data acquisition system. Figure 4.1 shows the schematic diagram of the complete set up. All the sensors are connected to the data acquisition card, installed in the pc, via an electronic interface module for signal conditioning. Two pressure measuring sensors are used to measure the water height of each tank. The details of the pressure sensor calibration is discussed in the Appendix C. Beam angle is measured by a potentiometer installed in the axis of beam rotation. For the purpose of real time control, angle is measured in voltage. Two d.c. pumps powered by linear amplifiers are biased and connected in parallel to provide the pumping between two tanks. Two analog output channels are used to control the voltage of pumps, and three analog input channels are used to read voltages of three sensors. There is no sensor to measure the angular velocity, so it is estimated from the position data. To reduce the noise of position sensor reading, averaging method is used. Within every sampling period, the position is measured about 50 times and the mean is used as position reading.

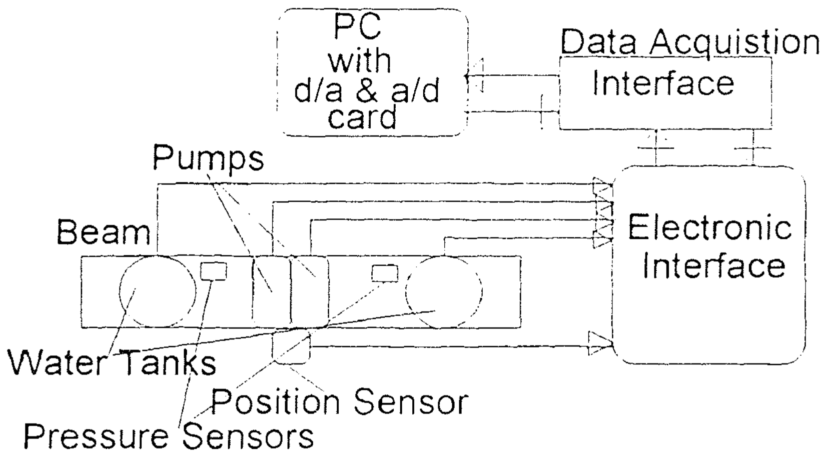


Figure 4.1 Schematic diagram of the balance beam set up

4.2 Real Time Programming

Real time computer systems should deliver the results in correct coordination with other systems operating asynchronously to the computer and to each other. The most of the simple real time problems can be solved by *synchronous* programming. A typical program of this class has a section to initialize data, place physical devices in appropriate initial states and run the program in an unending loop. A sample real time synchronous program for data acquisition is shown in Figure 4.2. In this program an analog voltage is read from a sensor through "analog in" channel of the data acquisition card, and the instantaneous digital value of this voltage is sent to the "analog out" channel using a continuos loop.

```
#include <stdio.h>
#include "io-fun.h"   /* I/O function definition */
main()
{
double volts;
int channel_1 = 1, channel_2 =2;

while(!kbhit();)
   {
    volts = a2d(channel_2);/* read a voltage from analog in
channel 2 */
    d2a(channel_1,volts);  /* output the same voltage to
analog out channel 2 */
   }
}
```

Figure 4.2 Synchronous program

In order to achieve true multitasking environment in a time critical or event driven situation, *asynchronous*, or *multi-thread* programming is needed [3]. Asynchronous programs are implemented by *interrupts*, which are hardware mechanisms in the computer that allow for the interruption of one thread of execution by another higher priority thread. The terms foreground and background are often used in connection with the high and low priority sections of such programs [3].

Figure 4.3 shows a program template which has been used to build the control program. This requires an action taking place on a strict time schedule plus another activity that is not time critical. In a control program, the time-critical section is used to implement a controller loop while the non-time critical section is used to get a new setpoint commands from the user. Because it is connected to the interrupt mechanism, the interrupt service routine will preempt the CPU resource whenever the clock interrupt single is present. Execution of the interrupt service routine will then continue until it is done, at which time execution of the non-time critical section will resume [21,3].

```
#include <stdio.h>
#include <8259.h>
#include <xignal.h>
#include <alarm.h>
#include "io-fun.h"  /* I/O function definition */
#define TIME  10.0  /* 10.0 millisec */
void isr();
double volts = 0.0;
int channel_1 = 1, channel_2 =2;
main()
{
xignal(XIGTMR, isr);  /* setup interrupt service routine */
setalarm(TIME);        /* at an interval of 10.0 milliseconds */

while(!kbhit()) /* Wait for user keyboard input to stop
   {              /* non-time critical process */
                  /* Put code here that can be interrupted */
    disable();  /* Turn off the interrupt */
                  /* Put code here that cannot be interrupted */
    enable();   /* Turn the interrupt back on again */
   }
 /*User has given "done" signal-- put computer's interrupt and timing
      system back to normal */
disable();
xignal(XIGALL,XIG_DFL); /* Set the interrupt vectors to default */
setalarm(-1.0); /* Set clock back to default */
enable();
}
void isr(void)  /* Time critical process */
{
   /* Put code here for the time critical ( interrupt-driven) task*/
}
```

Figure 4.3 Asynchronous Program

The program in Figure 4.2 is a single thread or synchronous operation. Both the tasks, analog to digital conversion as well as digital to analog conversion are executed sequentially from an unending loop. But in Figure 4.3, the time critical portion, the *interrupt service routine (isr)* is used to get the data from relevant instruments. The *main()* function is the non-time critical section.

4.3 Rule Construction

As discussed in the section on GeNFIS architecture, the structure of proposed neuro-fuzzy controller depends on the selection of rules. The fuzzy control rules for the beam balancing system have been constructed from the training data set. Although no formal rule generation algorithm has been developed in this work, the rules have been selected by finding the relation between input space and output space as discussed in FAM [31]. In addition, the association between different inputs are used to construct the premise part of the rules. The output and input data sets are first grouped in the different fuzzy sets like *positive high*, *negative low*, *zero*, etc. Next for each of the output fuzzy level, all the corresponding fuzzy sets of each input variables are tabulated. Each row of such table is a possible rule. The initial set of rules are selected by resolving the conflict among the rules. Then the conflict from the premise parts of the rules are removed. A further reduction in number of rules, if required, is done by removing similar type of rules. Finally, the rule base is enhanced by observing the performance of the system under control. The success of this rule generation method depends on the availability of a good set of training data. In this work, the training data sets are generated from simulation.

The GeNFIS structure used in this experiment consists of three inputs, eleven rules and one output. After an exhaustive on line investigation, starting from seven rules, it has been observed that eleven rules and three inputs are required to balance the beam in the horizontal position or at zero set point. It has also been observed that a GeNFIS controller can even balance the beam with only seven to nine rules. But in case of fewer number of rules, the controller can not stabilize the beam around the given set point. The beam will move away from the set point in a balanced condition.

The three inputs used here are position error, velocity error, and the water height error of left hand beam. The output of the controller is the motor control voltage. As discussed in chapter two, the fuzzy control rules are constructed by using linguistic variables. Table 4.1 shows the different labels used to represent state variables. The position of the beam is measured directly by using a potentiometer, whereas the velocity is calculated from position data and time. The water height error is also computed from the two pressure measuring sensors located at the bottom of each water tank Five labels are used to define the linguistic values of position error and water height error. These labels are: Negative Large (NL), Negative Small (NS), Zero (ZE), Positive Small (PS), Positive Large (PL). Since the measurement of velocity is indirect, only three labels, Negative (N), Zero (Z) and Positive (P) are used to define velocity error. Table 4.2 explains the nine labels of output voltage recommended by the fuzzy control rules.

Table 4.1 Different labels of input variables

STATE VARIABLES	LABELS
θ (Position error)	NL NS ZE PS PL
$\dot{\theta}$ (Velocity error)	N Z P
h....(Water height error)	NL NS ZE PS PL

Table 4.2 Different labels of output

OUTPUT VARIABLE	LABELS
u-- (Control voltage)	NM (Negative Maximum) NL (Negative Large) MN (Medium Negative) NS (Negative Small) ZE (Zero) PS (Positive Small) MP (Medium Positive) PL (Positive Large) PM (Positive Maximum)

As mentioned earlier, a total number of eleven fuzzy control rules are stored in the rule base of GeNFIS for this experiment. Table 4.3 shows the details of each rule. These rules can be read as:

Rule 1: **If** *position error* is NL **and** *velocity error* is N **and** *water height error* is NL

then the *control output* is NM

Rule 2: **If** *position error* is NS **and** *water height error* is NL

then the *control output* is NL

...

Rule 11: **If** *position error* is PL **and** *velocity error* is P **and** *water height error* is PL

then the *control output* is PM

Table 4.3 The 11 fuzzy control rules of GeNFIS

RULE #	Position	Velocity	Left water height	Control voltage
1	NL	N	NL	NM
2	NS	--	NL	NL
3	---	Z	NS	MN
4	NL	Z	---	MN
5	NL	--	ZE	NS
6	ZE	Z	ZE	ZE
7	PL	--	ZE	PS
8	PL	Z	---	MP
9	---	Z	PS	MP
10	PS	--	PL	PL
11	PL	P	PL	PM

After finalizing the structure of GeNFIS, training is done by using the data set obtained from simulation, as discussed in the section of plant simulation (section 3.2.2). The input-output data pairs are collected by running the simulation program twice, with two different sets of initial conditions. In both the cases, initial values of velocity and water height errors are taken as zero. The initial angle of beam is taken as 0.03 radians for the first run, and -0.03 radians for the second run. In each simulation, a sample time of 0.01 second is used with the run time of 6 seconds. The simulation results (Figure 3.4) show that the PID controller takes about 6 seconds to reach steady state. Although the simulation of 6 seconds with 0.01 second sample time will generate 600 data pairs, only a few number of data pairs (150) have been collected from each simulation. Figure 4.4 shows the comparison between the training data and the controller output. The training was terminated after about 800 cycles with a minimum RMS error of about 0.5872. It has been observed that the higher number of training cycles do not reduce the error measure, instead the training gets trapped around the error surface of local minimum. The high RMS error is due to the sudden peaks in the training data set. Two different initial conditions in the simulation (two different runs) are the cause of these peaks. Other than these peaks, the training is satisfactory. However, in comparing the RMS error of this training with the other published results, it should be noted that the raw data is used here, whereas most of reported results are based on the computations using normalized data [12]. For instance, if this data are normalized the minimum RMS error would be about 0.0293.

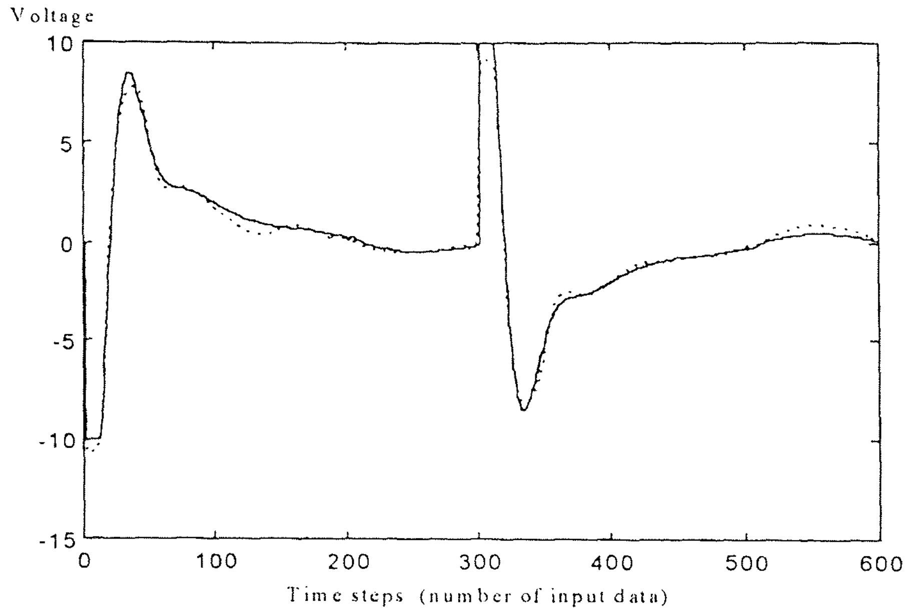


Figure 4.4 Output control voltage. Training data (solid), and after learning (dashed)

Figure 4.5(a) shows the five initial membership functions (MF) of the position error input. All the parameters of each initial MFs are same. The centers of the bell shape functions are equally spaced in the input data space. The position is measured in voltage, so for training and simulation the conversion from radians to voltage is needed. Figure 4.5(b) shows the final MFs.

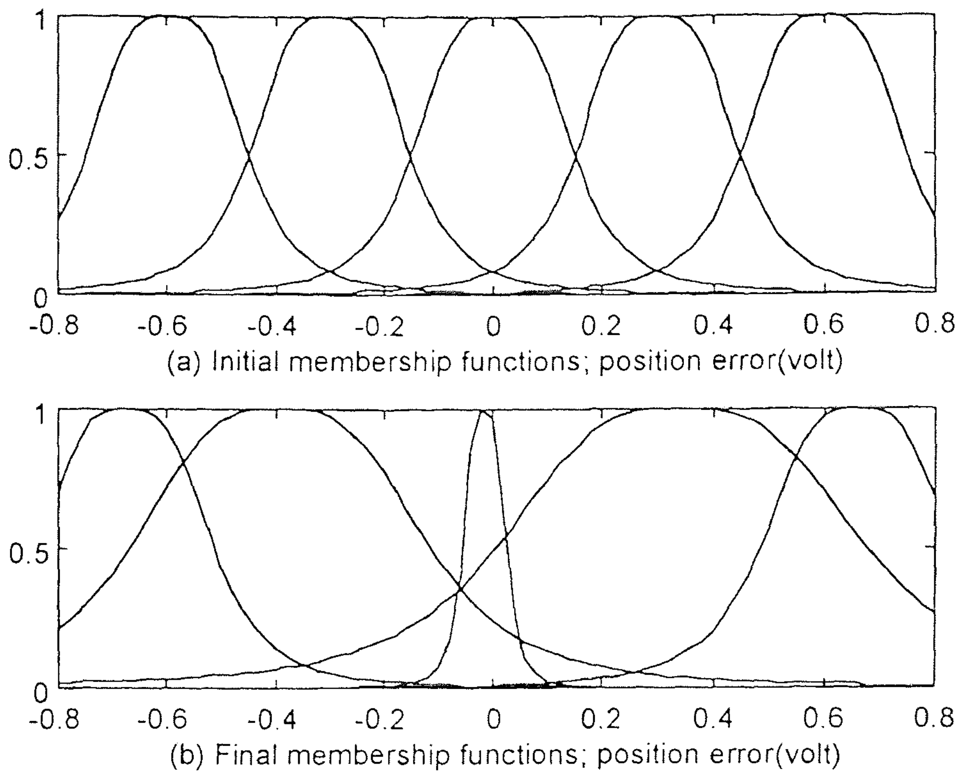


Figure 4.5 (a)Initial MF, and (b) Final MF of position error (volt)

The three initial and final membership functions for the velocity error are shown in Figure 4.6 (a) and (b) respectively. The initial and final membership functions for left hand water height error are shown in Fig. 4.6(c) and (d) respectively. It may be noted that the width of the three membership functions became very small. The water height error is computed in cm from the raw pressure data in voltage. The eleven output initial and final membership functions are shown in Figure 4.7 (a) and (b) respectively. Although the range of output control voltage is from -10.0 to +10.0, the centers of some of the final membership functions have moved outside this range. This will recommend a out of range control voltage, but the voltage applied to the motor is set with in the given range.

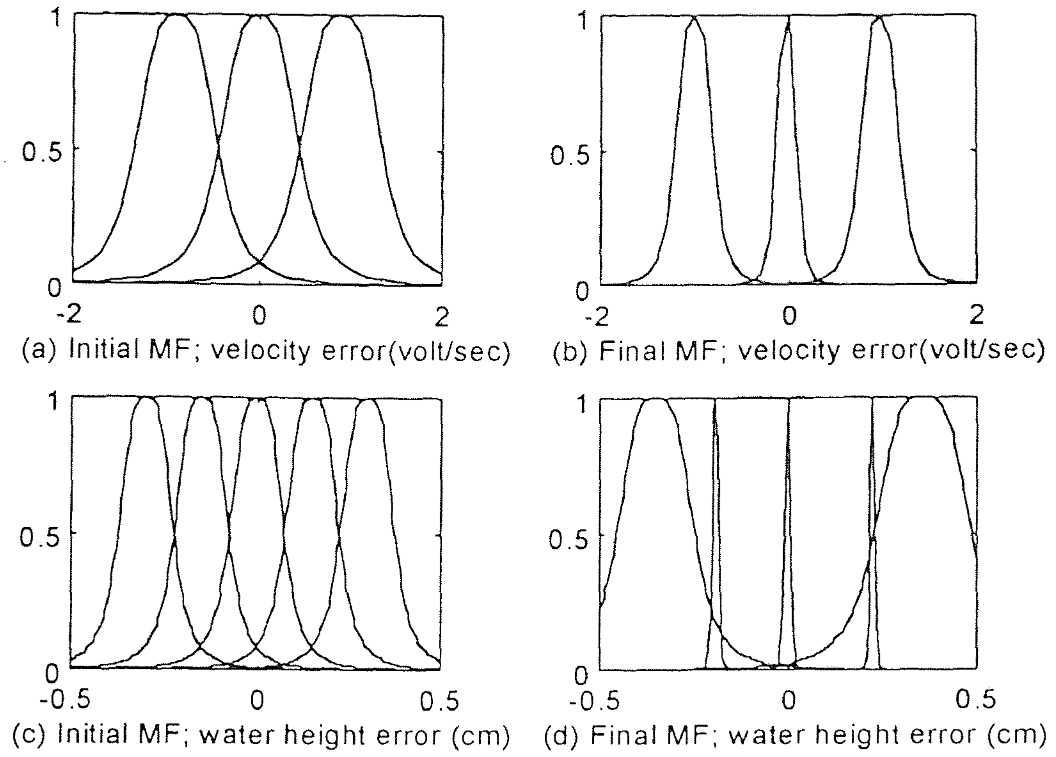


Figure 4.6 (a) Initial MF, and (b) final MF of velocity error (volt); (c) Initial MF, and final MF of left hand water height error (cm).

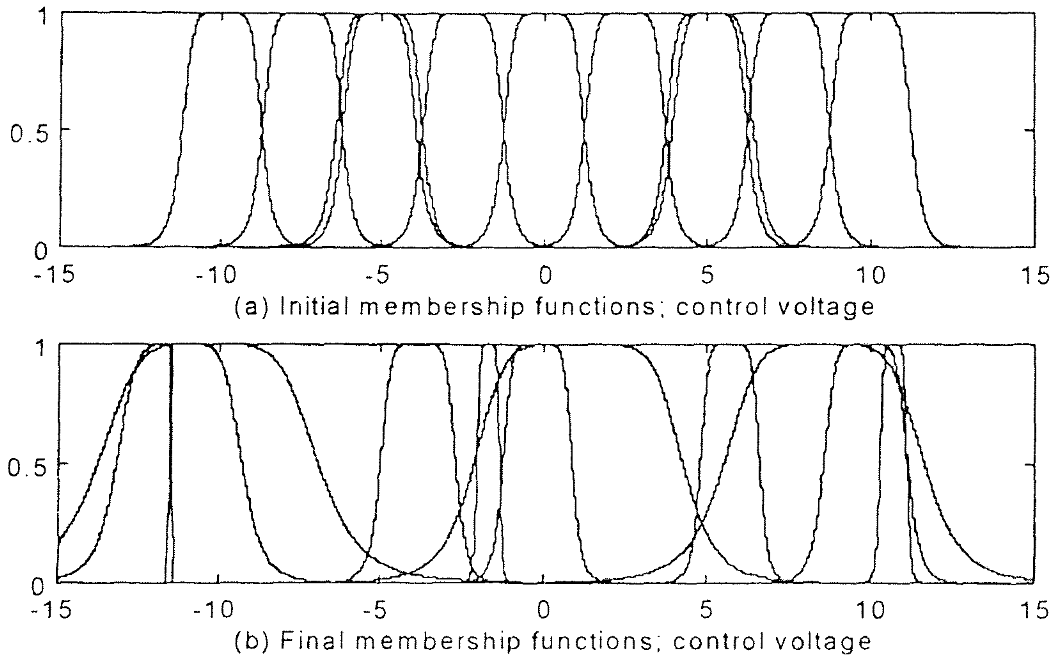


Figure 4.7 (a) Initial MF, and (b) final MF of motor control voltage

4.4 Experimental Results

Simulation of the balance beam system as well as real-time control of the equipment, both have been conducted to evaluate the performance of GeNFIS type controller. This section presents the detailed experimental results.

4.4.1 Simulation

Figure 4.8 shows the simulation results of GeNFIS controller after training with eleven rules as given in the Table 4.3. In this simulation, initial position error (beam angle) is taken as 0.03 radians. Initial conditions of velocity and water height error are set to zero. Figure 4.8(a), (b), and (c) are the plots of position error, velocity error and left hand water height error respectively. Figure 4.8(d) is the plot of output motor control voltage from GeNFIS controller. Although the simulation data for only 4 second is shown, the partial state space curve of position error against velocity error in Figure 4.9(a) indicates that the plant is approaching to the steady state. This simulation may be compared with the simulation of PID controller presented in the previous chapter (see Figure 3.4).

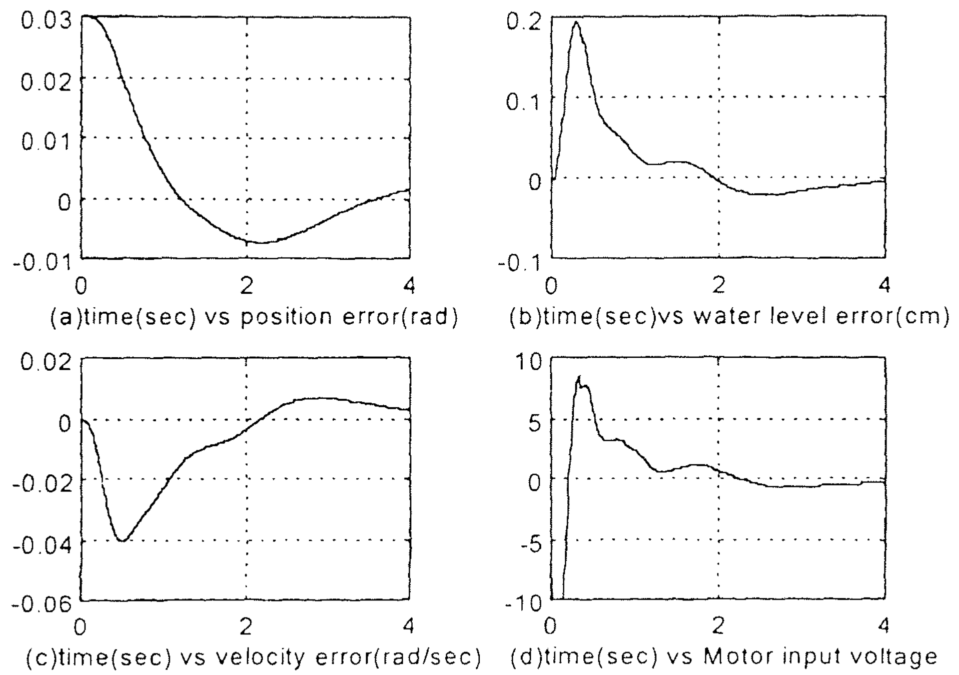


Figure 4.8 Simulation with GeNFIS controller: Initial position 0.03 radians. (a) position error, (b) left-hand water height error, (c) velocity error, and (d) motor control voltage

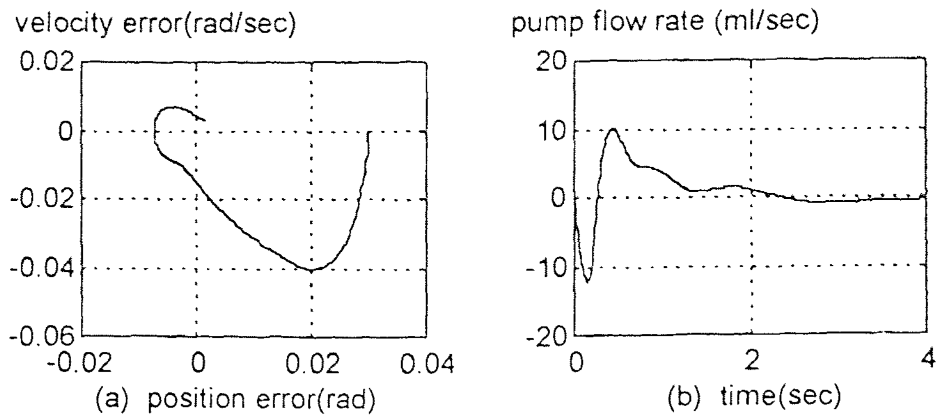


Figure 4.9 Simulation with GeNFIS controller: Initial position 0.03 radians, (a) state space (error), and (b) pump flow rate.

Figure 4.10 shows the balance beam simulation with the different initial conditions. In this test, initial position error is taken as -0.03 radians and the remaining variables are set to zero as the previous simulation. Figure 4.10 and Figure 4.11 explain the simulation results for 4 seconds.

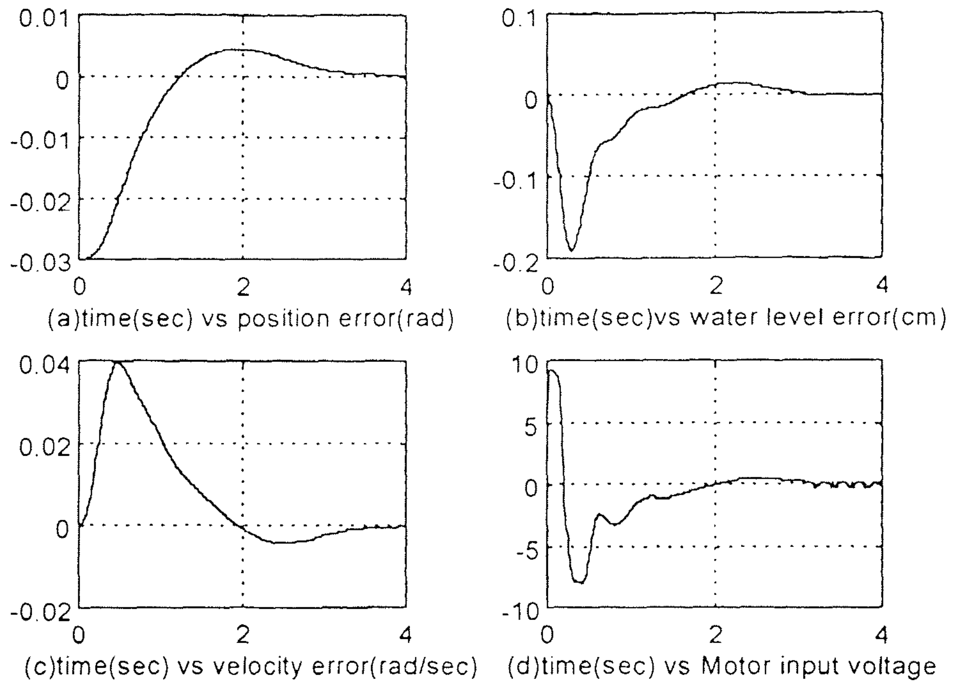


Figure 4.10 Simulation with GeNFIS controller: Initial position -0.03 radians. (a) position error, (b) left-hand water height error, (c) velocity error, and (d) motor control voltage.

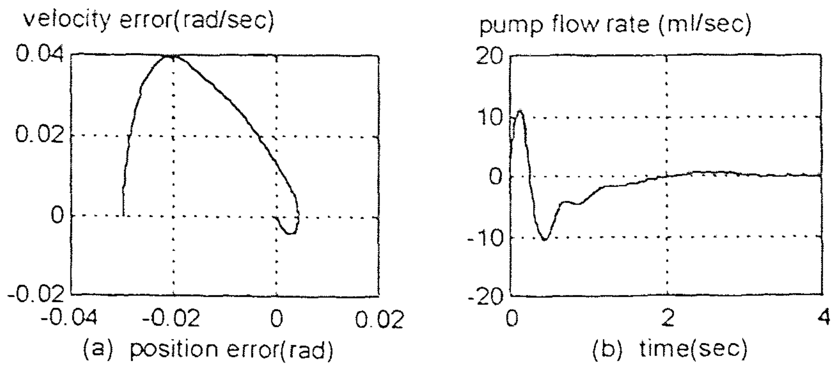


Figure 4.11 Simulation with GeNFIS controller: Initial position -0.03 radians, (a) state space (error), and (b) pump flow rate.

To test the robustness of the GeNFIS controller, two more experiments were done. In each of these, the controller is asked to balance the beam from different sets of initial conditions. In the first test, initial position error is taken as -0.025 radians and the velocity error is set to zero as earlier. However, an initial left hand water height error of -0.2 cm is introduced. Figure. 4.12 shows that the controller can balance the beam within a reasonable time. Figure 4.13(a) and (b) are the plot of position error against velocity error and water height error respectively.

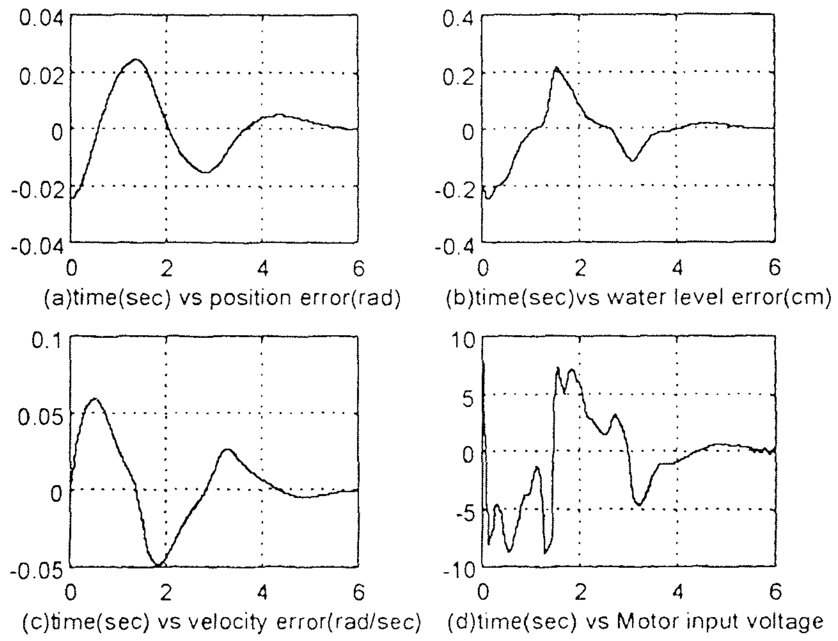


Figure 4.12 GeNFIS simulation with initial angle -0.025 radians and initial water height error -0.2 cm

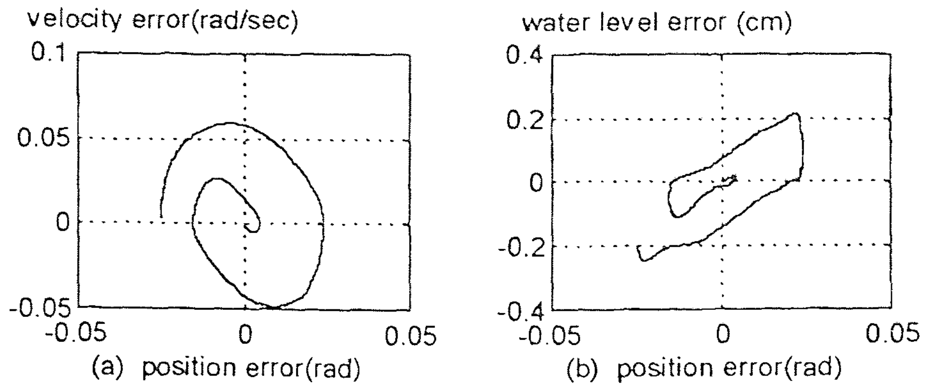


Figure 4.13 Simulation with GeNFIS controller: Initial position 0.025 radians and initial water height error -0.2 cm; position error Vs (a) velocity and (b) water level errors.

Figure 4.14 and Figure 4.15 are the simulation results with an initial beam angle of -0.05 radians for a 5 second duration. Although the results within this time span are not encouraging, the partial state space curves, as shown in Figure. 4.15(a) and (b), reveal that the system is slowly approaching to the steady state.

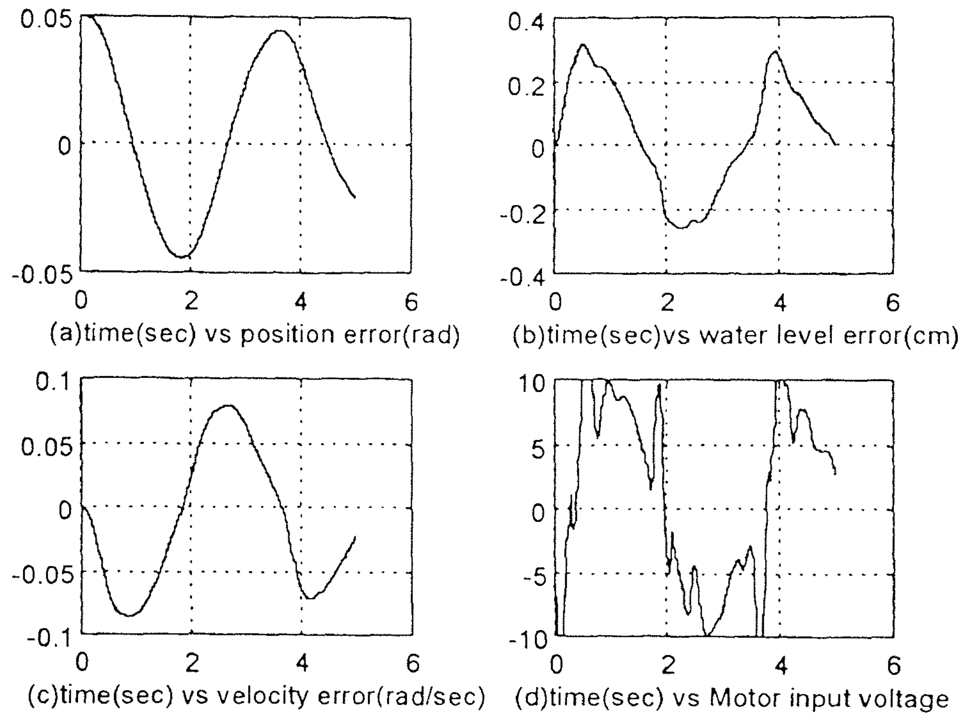


Figure 4.14 GeNFIS simulation with initial angle -0.05 radians

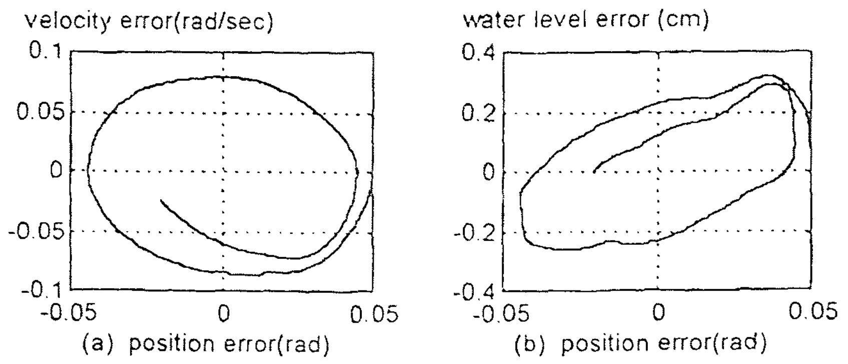


Figure 4.15 Simulation with GeNFIS controller; initial position -0.05 radians; position error against (a) velocity and (b) water level errors.

4.4.2 Real-Time Control

In real time operation, the performance of GeNFIS controller is investigated against a PID controller. The same GeNFIS controller, which has been used in the simulation, is included in the real time control program. Figure 4.16 shows the steady state control of balance beam at the horizontal position using GeNFIS controller. Whereas Figure 4.17 shows the same control using a PID controller. Figure 4.16(a) indicates a very small steady state position error, oscillating on one side of the set point, with the GeNFIS controller. The corresponding figure under PID controller, Figure 4.17(a), shows that the position error is oscillating around the set point. A sample time of 30 milliseconds is used in all the experiments.

To test robustness of the controllers, different levels of disturbances are applied on the system. These disturbances are of step input type, and are created in the software. The software will be able to produce the same level of disturbances repeatedly. To measure the robustness, step inputs of different voltage levels are applied to the system at the steady state with the zero set point. It has been observed that the GeNFIS controller can sustain such disturbances up to the magnitude of 6 volts. Figure 4.18 shows the test results of 6 volt step input for GeNFIS controller. The system survives and recovers slowly after the impact. In case of PID controller, it has been observed that the system can easily sustain different step inputs up to the level of 8 volts. Figure. 4.19 shows the corresponding test results of PID controller. The GeNFIS controller fails, when a disturbance of 7 volt is applied. Figure 4.20 illustrates the result of this test. Figure 4.21 shows the failure of PID controller at a disturbance label of 9 volt.

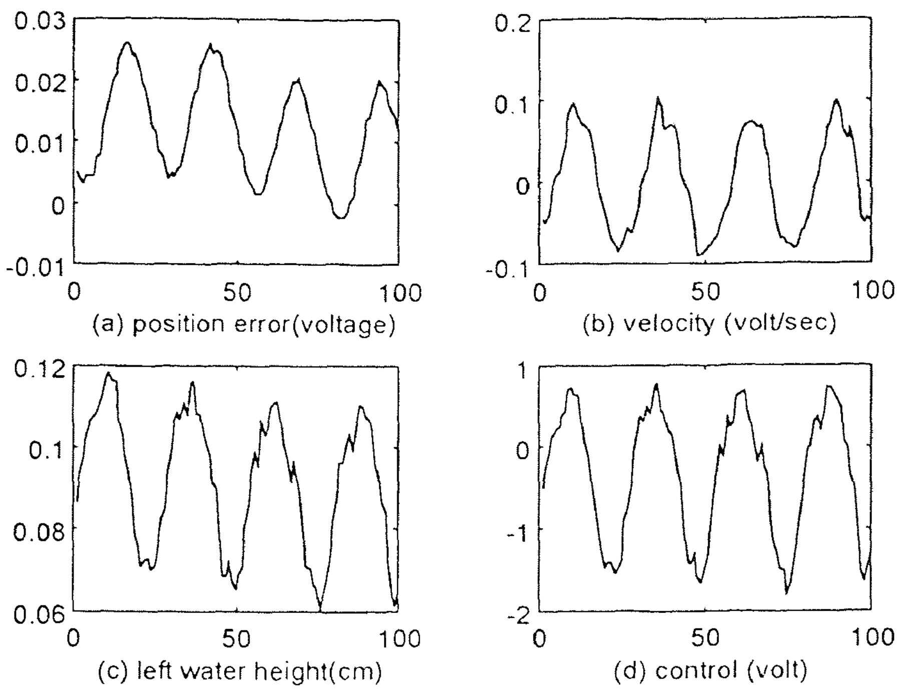


Figure 4.16 Control of balance beam using Neuro-Fuzzy controller (GENFIS) at the set point of 0 radian (horizontal). Time steps (0 to 100) vs. (a) position error in volt (b) left water height in cm(c) velocity error in volt/sec (d) control action in volt

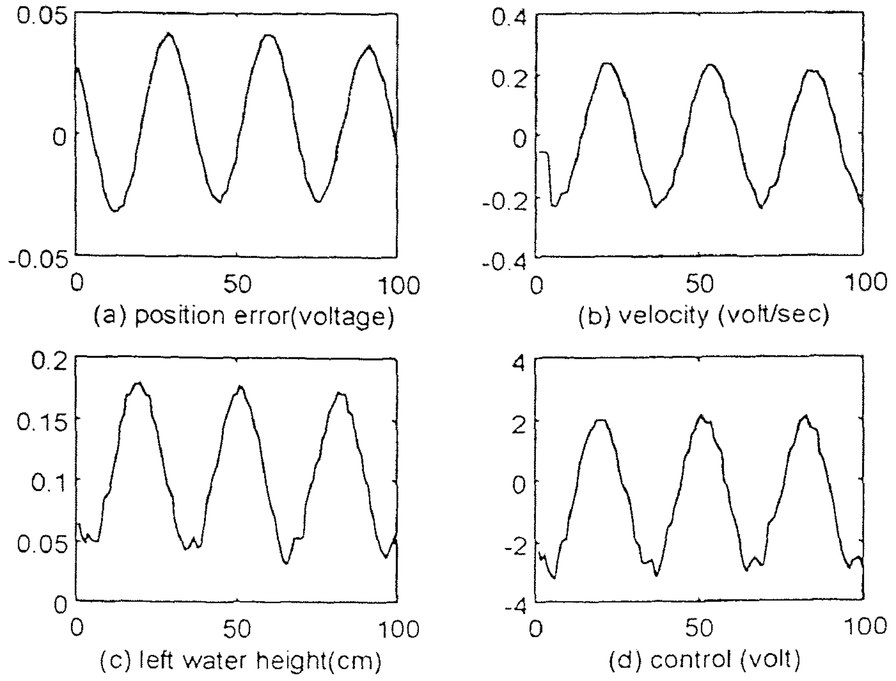


Figure 4.17 Control of balance beam using PID controller at the set point of 0 radian (horizontal). Time steps (0 to 100) vs. (a) position error in volt (b) left water height in cm(c) velocity error in volt/sec (d) control action in volt

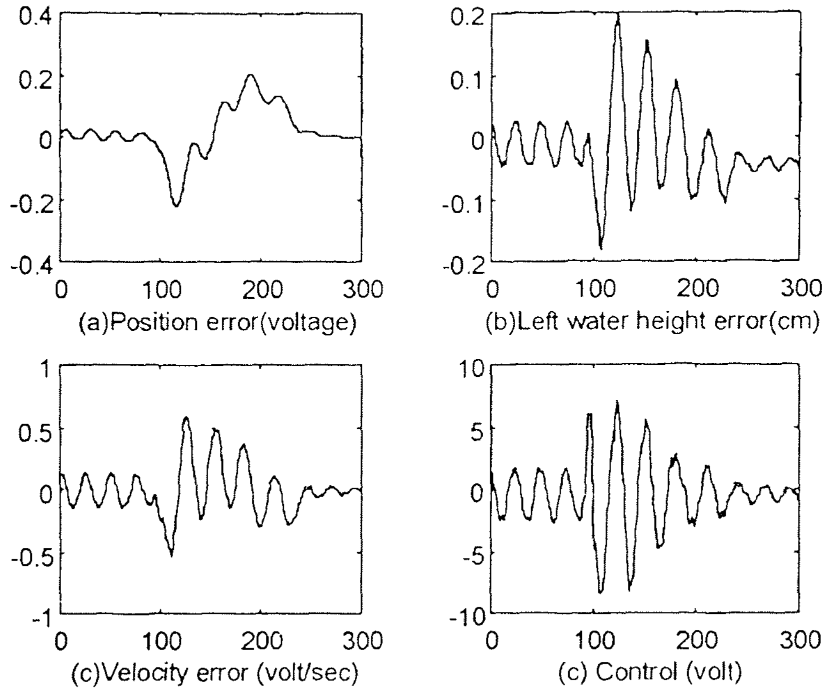


Figure 4.18 Control of balance beam using Neuro-Fuzzy controller with a disturbance of 6 Volt. Time steps (0 to 300) vs. (a) position error in volt (b) left water height in cm(c) velocity error in volt/sec (d) control action in volt

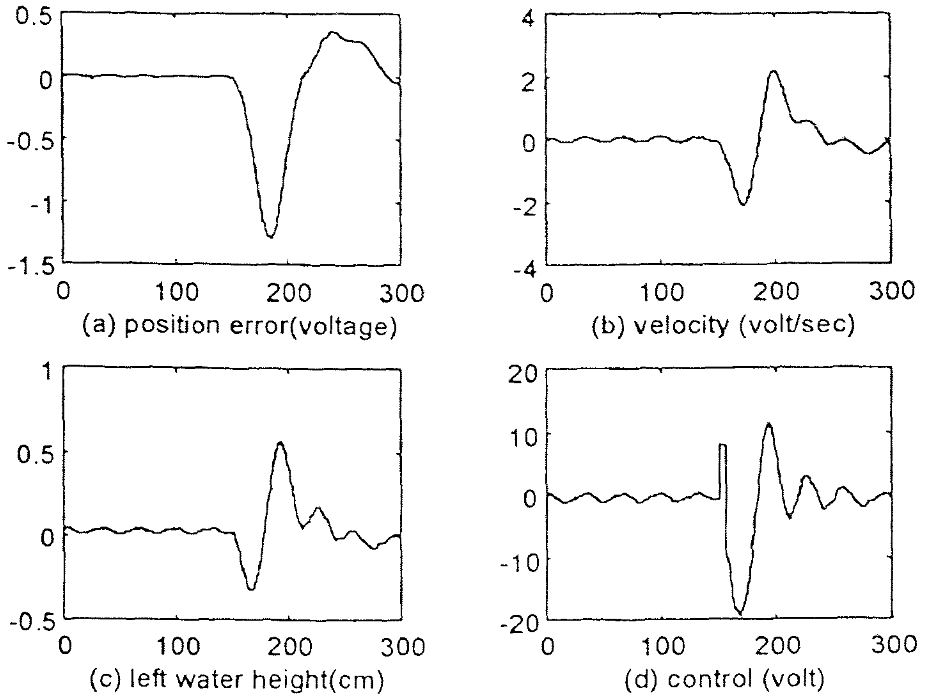


Figure 4.19 Control of balance beam using PID controller with a disturbance of 8 Volt. Time steps (0 to 300) vs. (a) position error in volt (b) left water height in cm(c) velocity error in volt/sec (d) control action in volt

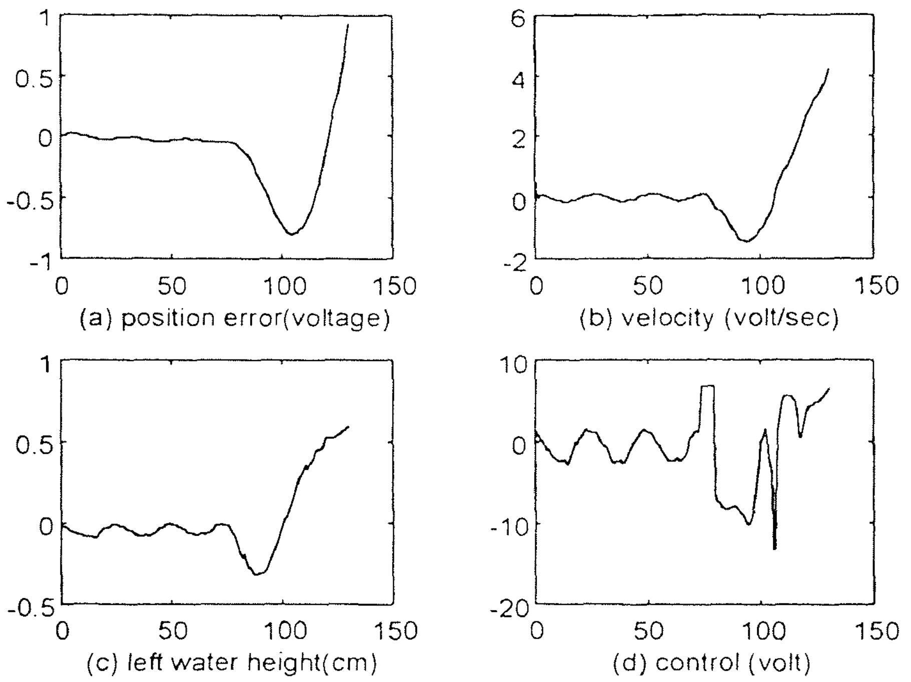


Figure 4.20 Failure of balance beam control using Neuro-Fuzzy controller with a disturbance of 7 Volt. Time steps (0 to 150) vs. (a) position error in volt (b) left water height in cm(c) velocity error in volt/sec (d) control action in volt

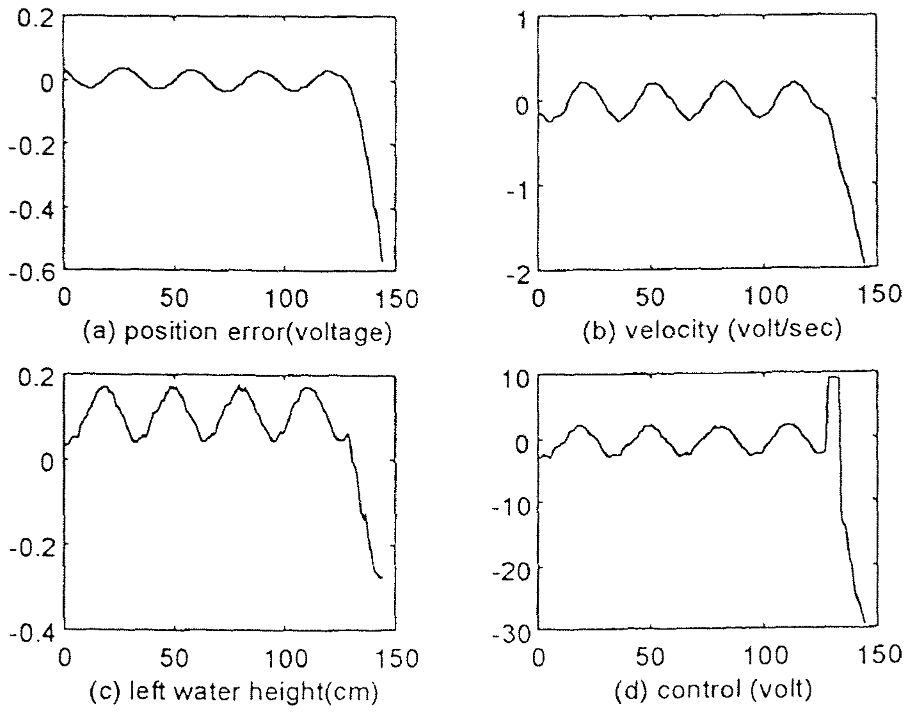


Figure 4.21 Failure of balance beam control using PID controller with a disturbance of 9 Volt. Time steps (0 to 150) vs (a) position error in volt (b) left water height in cm

CHAPTER 5

EXPERIMENTING WITH GeNFIS

5.1 Synchronous Defuzzification Scheme

Defuzzification is an important aspect of fuzzy logic control which determines a crisp value from the set of consequent fuzzy sets. Several methods are available for defuzzification in fuzzy inferencing system in order to select a crisp value from the possibility distribution over the output space [46]. However, to implement these defuzzification schemes in a neuro-fuzzy network some modifications are needed. In recent years, several successful methods to do this have been reported in the literature of neuro-fuzzy control. Although all these methods are very useful for practical applications, comparative performance evaluation of these schemes are not available. In this chapter a simple method of defuzzification using different types of membership functions in consequent level of the same rule of a neuro-fuzzy controller is used with the objective of performance evaluation as well as enhancement of defuzzification schemes.

A parallel path is included in the consequent layers of GeNFIS with a different type of membership function. Figure 5.1 shows the modified structure of GeNFIS. Note that C11 and C12 are the same consequent level of rule 1, with same linguistic value but with different types of membership functions. The weights (ji_i) are used to compute weighted sum of the output from each parallel path of same rules (R_i). In the present scheme, weights are used as constants and the total sum is one for each rule. *K11* and *K12* are the output path 1 and 2 respectively of *R1*. The necessary relations to compute each rule output is given by:

$$ji_1 = \theta$$
$$ji_2 = 1 - \theta$$
$$R1_{out} = K11 + K12 = \theta * \frac{w'_1\left(\mu^{-1}_{C11}(w'_1)\right)}{\sum w'_i} + (1-\theta) * \frac{w'_1\left(\mu^{-1}_{C12}(w'_1)\right)}{\sum w'_i}$$

where, limit of θ is $1.0 \leq \theta \geq 0.0$ (5.1)

The computations in the remaining layers are same as in the case of GeNFIS structure. The weights can be tuned manually to mix the results of different membership functions. For training, the same back propagation algorithm is used.

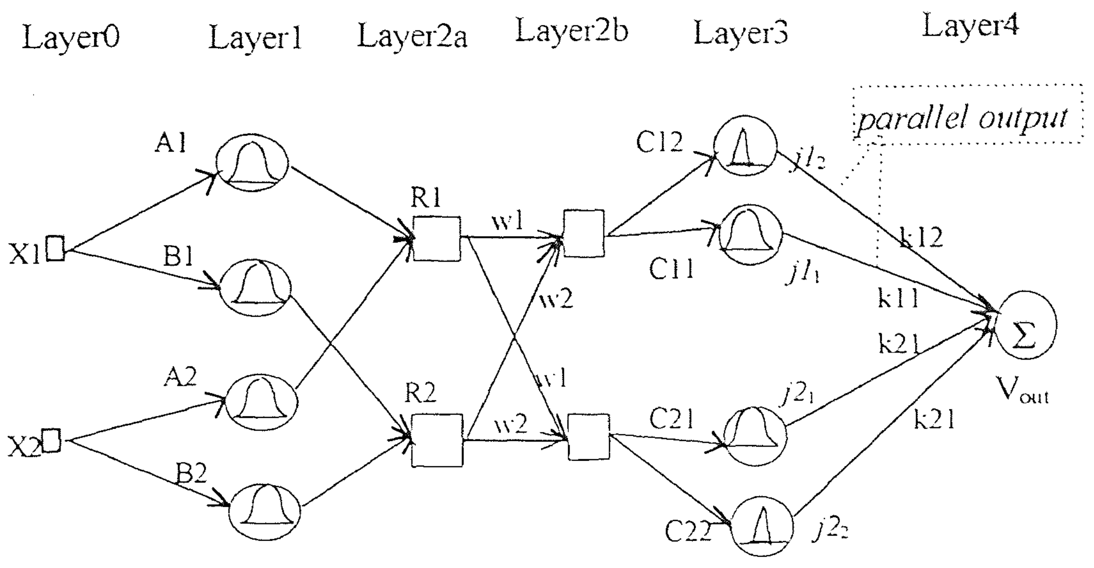


Figure 5.1 Modified GeNFIS structure

In this experiment, triangular membership functions, with some modifications, have been used in the parallel path. As suggested in [9], a local mean of maximum (LMOM) method is used for defuzzification. However, a linear membership function defined by two parameters is used instead of three parameters of a triangular membership functions. In [9], three parameters used are center (c), left spread (s_L) and right spread (s_R) (Figure 5.2a). Two parameters used in this test are: p to account for center or position, and c to account for spread. The LMOM is defined as:

LMOM or $\mu_c^{-1}(w_R)$ is the X-coordinate of the centroid of the set $\{x: \mu_c(x) \geq w_R\}$

However, for a triangular membership function, the LMOM is the projection of the median. In case of two parameters, the straight line defined by c and p may be treated as the same median. In such situation, LMOM is simply the " *X-coordinate of the intersecting point between the line c p_1 and w_R*". Figure 5.2(a) and (b) illustrates the assumption. From Figure 5.2(b) defuzzified value can be written as:

$$\mu^{-1}(w'_R) = C*(\mu(x)) + P*(1-(\mu(x))) \qquad (5.2)$$

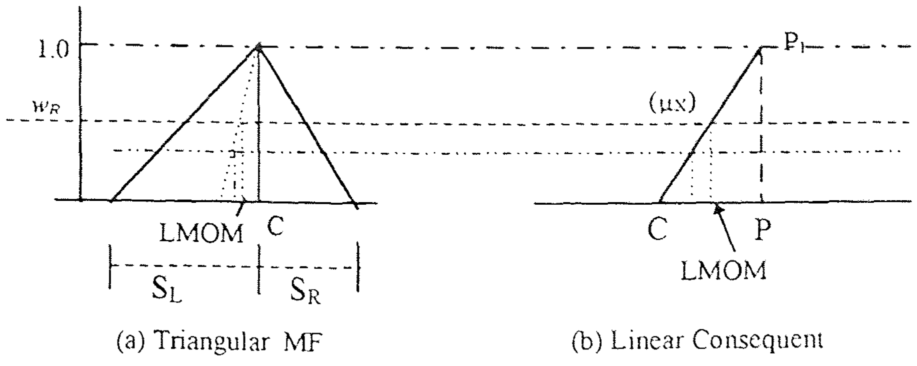


Figure 5.2(a) LMOM of triangular MF; (b) Linear MF

5.2 Experimental Results

The same experiment is also done with modified GeNFIS controller. In addition to the eleven bell shaped membership functions, eleven linear functions were chosen for the parallel path. The same eleven rules along with the same training data were used for training. After 500 cycles of training, RMS error reached the minimum value of 0.888 (0.044 normalized value). Figure 5.3 shows the final input MFs, and Figure 5.4 shows MFs for control voltage. Table 5.1 shows the parameters of input and output linear functions. The weights are taken as: 0.6 for bell shaped functions, and 0.4 for linear functions. Figure 5.5 shows the output data after training at the RMS error of 0.888.

5.2.1 Simulation

Simulation results show that the modified GeNFIS controller can balance the beam. Figure 5.6 indicates that the control is smooth for -0.03 rad initial condition. However a steady state position error of about 0.017 radians exists in the simulation. Figures 5.7(a) and (b) show the plot of velocity error against position error, and the pump flow rate against time respectively.

5.2.2 Plant Control

Figure 5.8 shows the details of plant control. The controller can withstand small disturbances. It can tolerate the disturbances of up to 3 volts. The experimental results indicate that synchronous method of defuzzification using triangular MFs can not enhance the performance of GeNFIS. Recall that the original GeNFIS can sustain the disturbance of up to 6 volts.

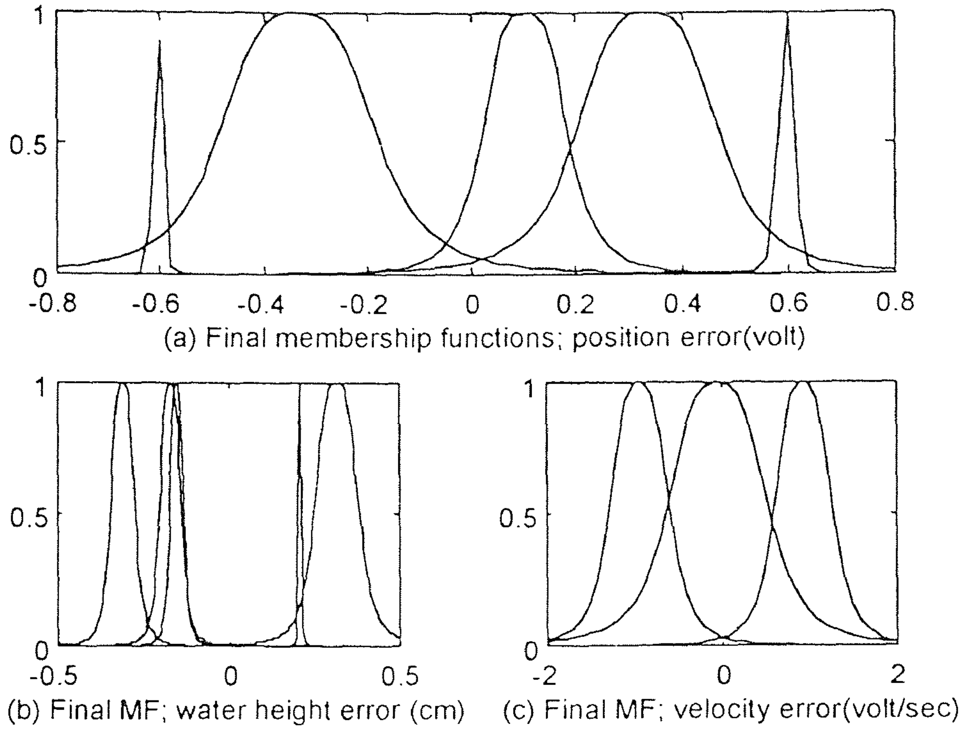


Figure 5.3 Input membership functions

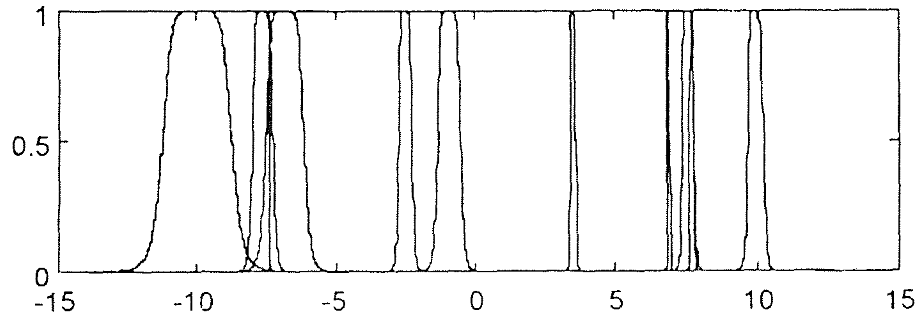


Figure 5.4 Final membership functions: control voltage.

Table 5.1 Initial and Final Linear functions

Rule #	Input "C"	Input "P"	Final"C"	Final"P"
1	-12.0	-10.0	0.0042992208	-10.0008273887
2	-11.0	-7.5	-11.2250090688	-7.5010937835
3	-8.5	-0.5	-9.9039023358	-6.0528130969
4	-8.5	-5.0	-8.8826881619	-7.9492404629
5	-7.0	-2.5	-6.9731060446	-2.5425974559
6	-3.5	0.0	-4.1697309867	-0.7009948482
7	-1.5	2.5	-1.0130359479	3.4442059148
8	1.5	5.0	1.2141521395	9.2531529035
9	1.5	5.0	3.2422564502	5.7108002736
10	3.5	7.5	3.5776639781	7.5011757425
11	6.5	10.0	6.4402147456	9.9997155710

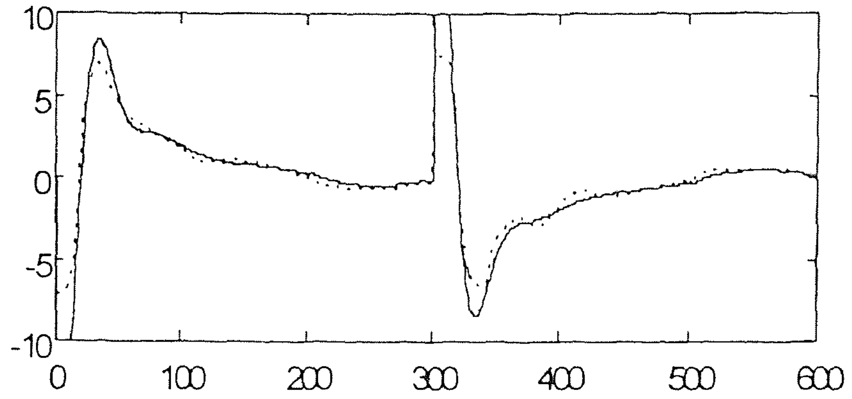


Figure 5.5 Output with modified rules: Training data(solid) and output data (dashed)

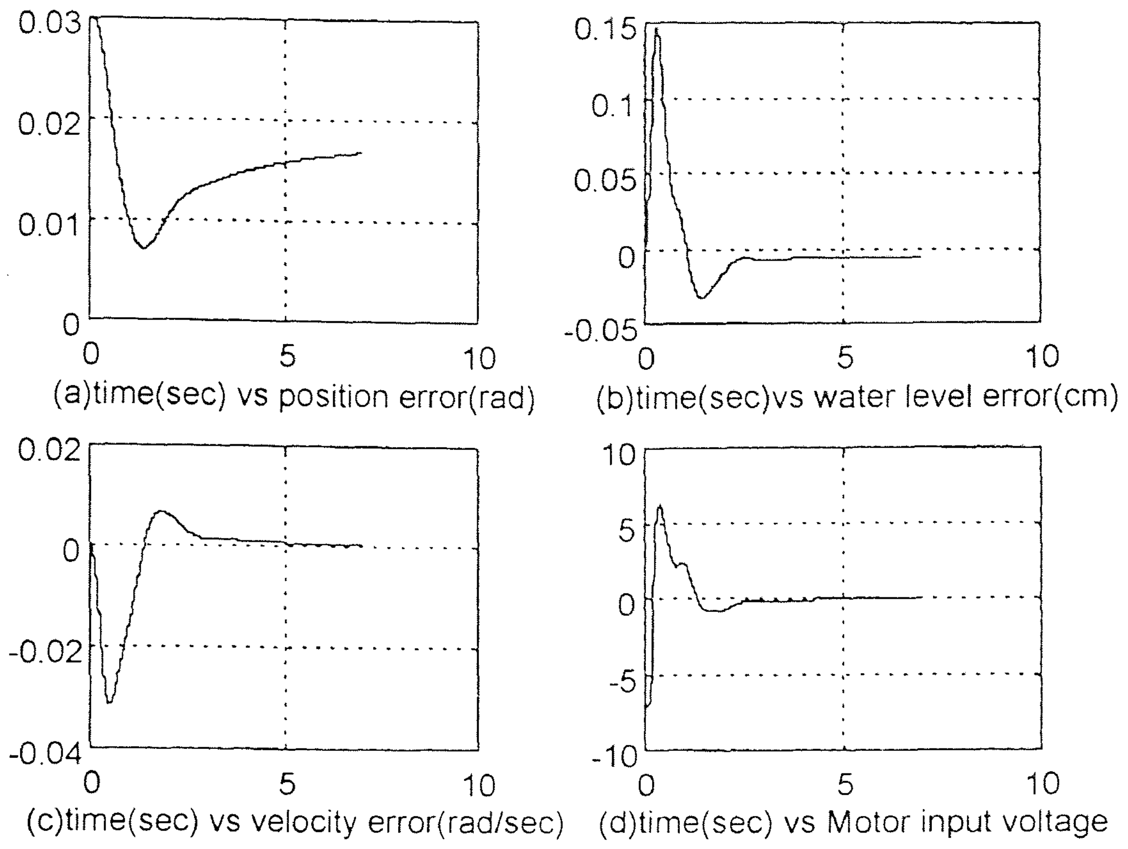


Figure 5.6 Simulation with modified rule antecedent

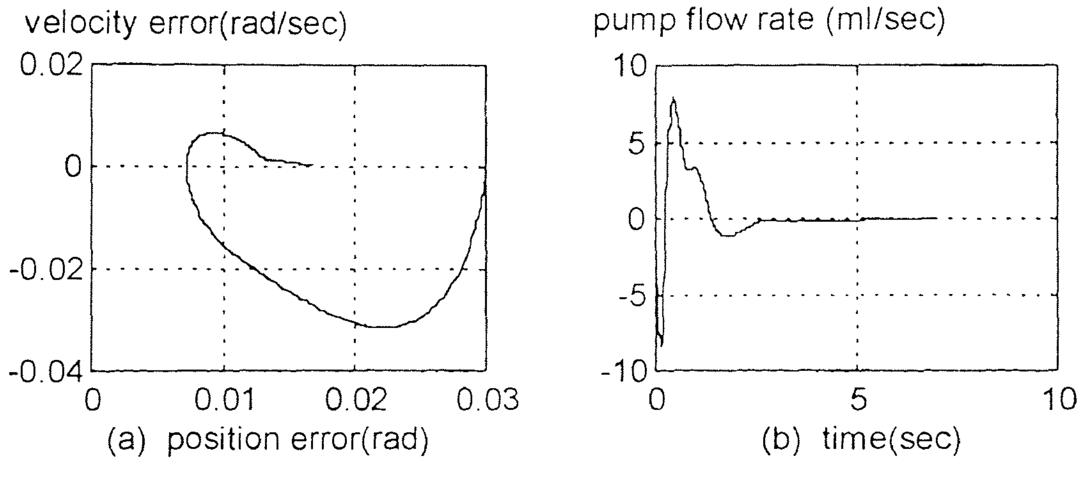


Figure 5.7 Modified rules;(a) state (position error vs. velocity error) (b) pump flow rate

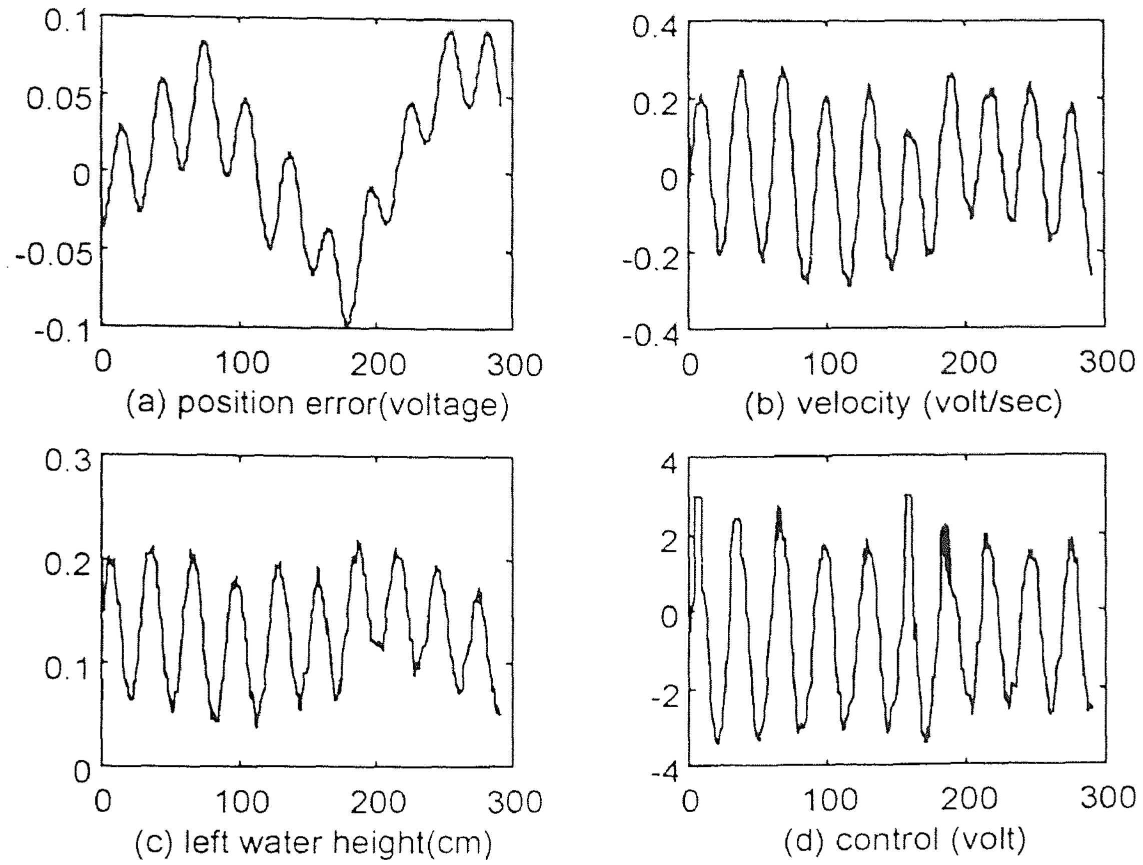


Figure 5.8 Modified Rules; Normal operation (disturbance 3.0 volts)

5.3 Hybrid Learning

Since the synchronous defuzzification scheme with triangular membership function does not improve the performance of GeNFIS, it is replaced by Sugeno type rule consequent. The modified GeNFIS structure is shown in Figure 5.9. The weight equations are same as before (see Equation 5.1). However, the rule output computation is changed due to the introduction of first-order Sugeno fuzzy model [25]. The modified relation is given by

$$R1_{out} = K11 + K12 = \theta * \frac{w_1\left(\mu_{C11}^{-1}(w_1)\right)}{\sum w_i} + (1-\theta) * \frac{w_1(p_1 x_1 + q_1 x_2 + r_1)}{\sum w_i} \qquad (5.3)$$

where, p_l, q_l, and r_l are the constants of Sugeno type rule consequent.

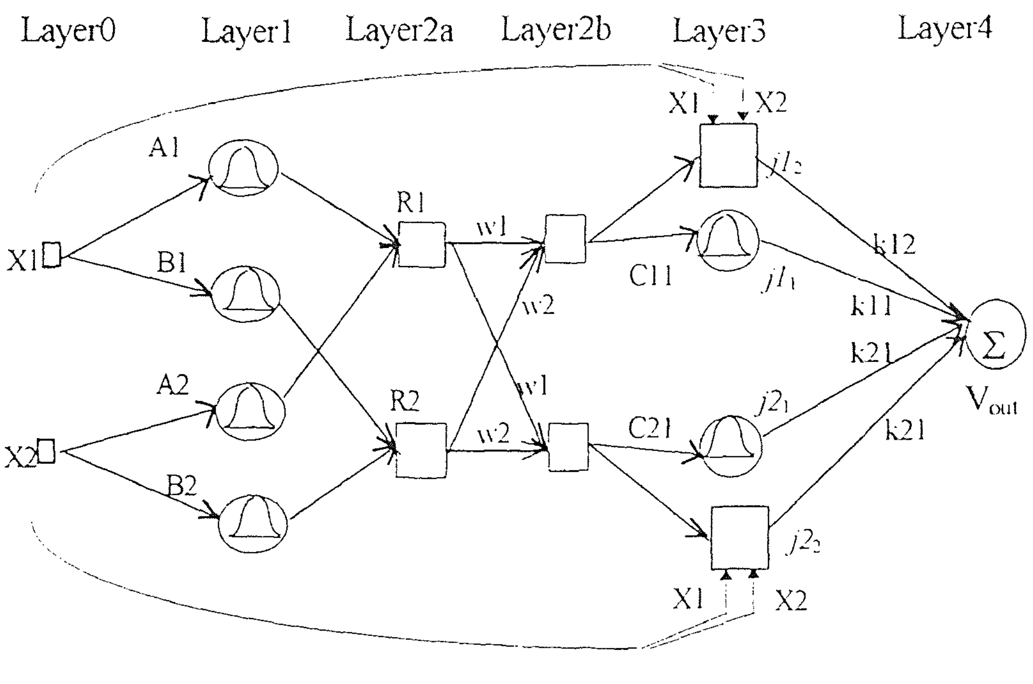


Figure 5.9 Modified GeNFIS with Sugeno type rules

The training is done in two phases. First, the GeNFIS parameters are trained using only back propagation algorithm with unit weights. The premise membership functions are tuned in this phase. The weights of the synchronous Sugeno layers are treated as zero. In the second phase, Sugeno parameters are identified by using least-square estimators.

The parameters of the input layers are kept as constants with the trained values from the phase one. In this phase, the weights in the Sugeno layers were set to one, whereas the weights in GeNFIS layers were set to zero. During experiment, the weights were adjusted or tuned to get the best results.

5.3.1 Experimental Results

First, we test the controller using equal weights of 0.5 in both the paths. Figure 5.10 shows the simulation results with initial position of -0.3 radians. The results indicate that the controller can balance the beam smoothly. Comparing with the previous simulation results, Figure 5.10 shows a very smooth motor input voltage curve. However, in the actual experiment, the controller could not balance the beam using 0.5:0.5 weight allocation. Next, we tuned the weights to find a feasible controller. We observed the controller can reasonably balance the beam when the weight ratio is 0.75:0.25. The weights of 0.25 were used in the Sugeno layers. Figure 5.11 shows that the controller can even sustain a disturbance of 4 volts. These results lead us to investigate further about the ANFIS scheme.

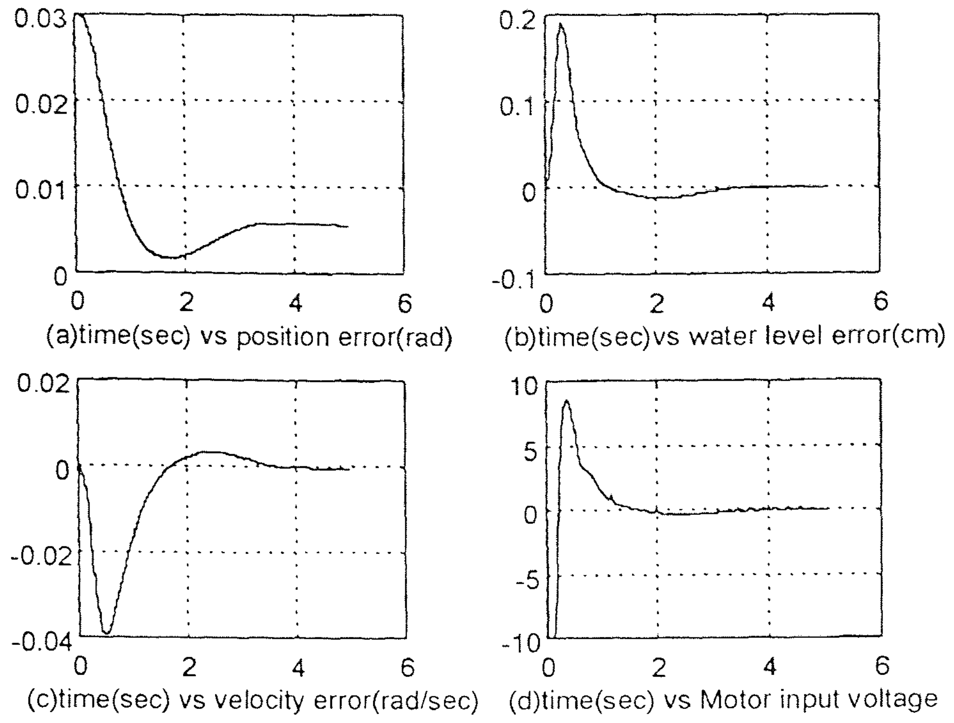


Figure 5.10 Simulation with modified GeNFIS (weights: 0.5 in both)

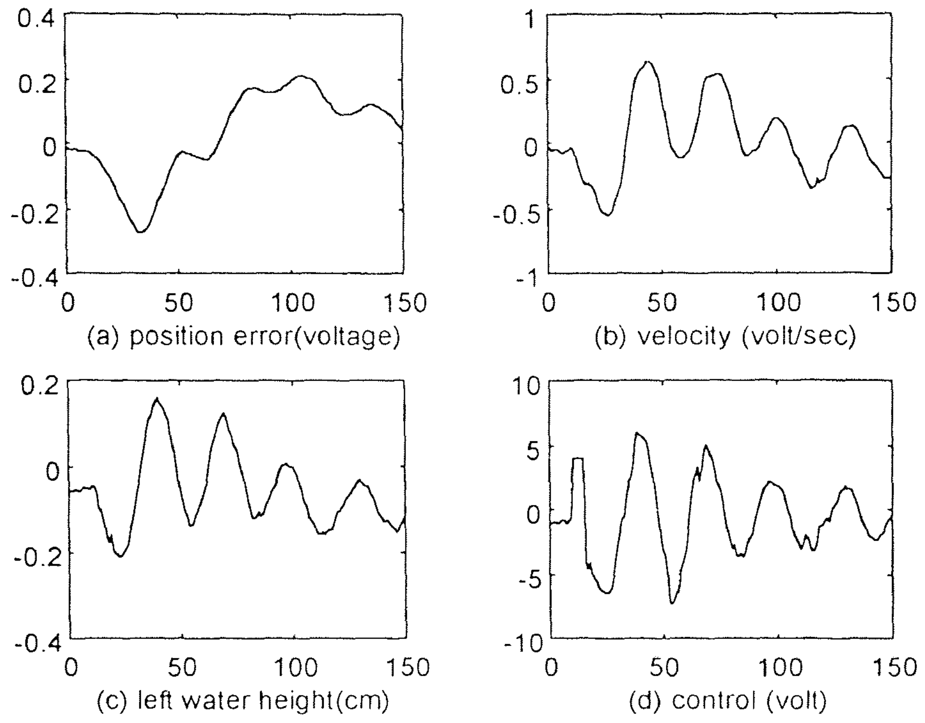


Figure 5.11 Modified GeNFIS with 4 volt disturbance level (weights; 0.75:0.25)

5.4 ANFIS Implementation

In this section, ANFIS means the use of first-order Sugeno fuzzy model in the consequent layers of GeNFIS with hybrid learning scheme using least-square estimators and gradient descent [27]. All the structures under investigation have been trained by the same set of training data under supervised learning.

First we develop an ANFIS structure using 12 rules. Since complete partitioning of the input data space is required for ANFIS, we have to reduce the number of membership functions of each input. If we use all 13 membership functions for 3 inputs, we need to use 75 rules (5×3×5), which is rather difficult to implement on a personal computer. In this experiment we have used a total of 7 membership functions, 3 for position and 2 each for velocity and water height errors. This will yield 12 rules (3×2×2). Linear combination of three Sugeno type parameters (linear combination), have been used in the consequent part of each rule. A blending of least square estimators for the identification of consequent parameters and the gradient descent method of backpropagation for updating the premise as well as consequent parameters have been used for training. The training converges within few cycles and the RMS error was about 0.32 (0.016 normalized value). Figure 5.12 shows the matching curve.

5.4.1 Experimental Results

Figure 5.13 and 5.14 show the simulation results of ANFIS with initial position error of 0.03 radians. Although simulation results indicate that ANFIS can balance the beam from the given initial condition, the control voltage curve consists of number of sharp changes. Figure 5.15 shows the simulation results with 0.025 radians initial position. Although the control task is easier in the case of 0.025 radians, the motor control curve includes higher number of sharp changes than the corresponding curve of 0.03 radians. Figure 5.16 shows the test results of actual real time control of balance beam with ANFIS controller. The controller cannot stabilize at the zero set point

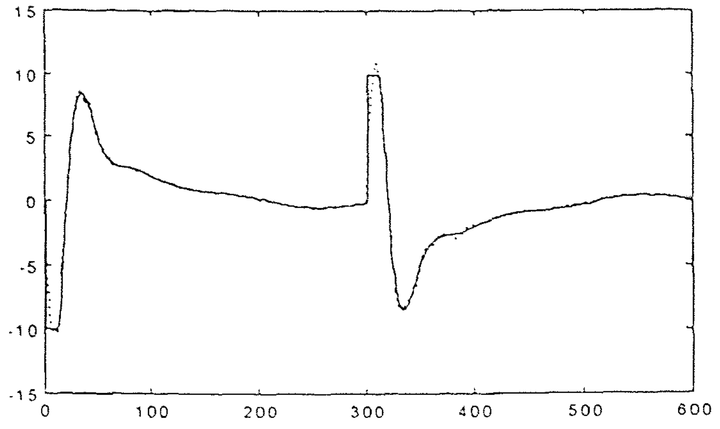


Figure 5.12 ANFIS: Training and output data

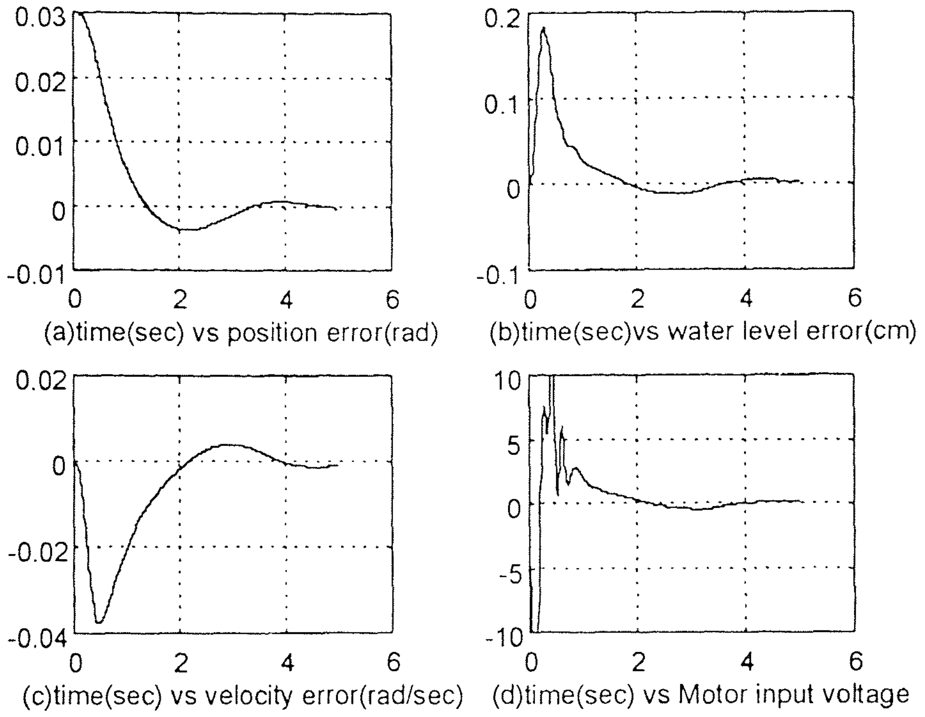


Figure 5.13 ANFIS controller Initial position -0.3 radians

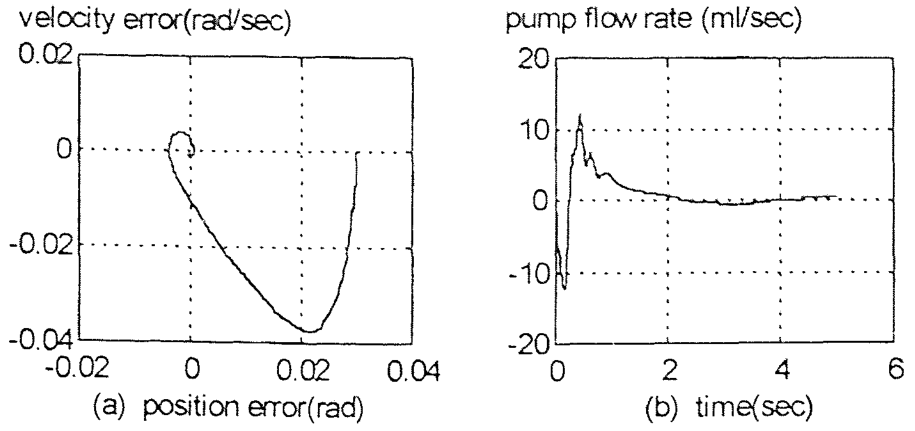


Figure 5.14 Simulation ANFIS (initial position -0.3 rad); (a)state (b) pump flow rate

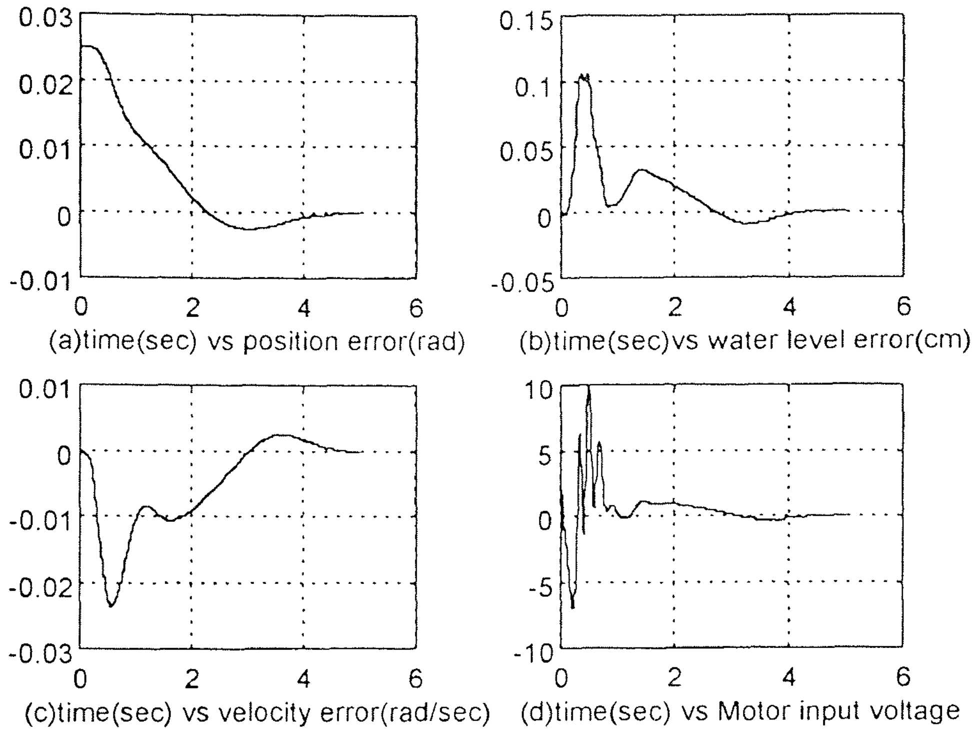


Figure 5.15 Simulation ANFIS controller Initial pos -0.25

5.4.2 Limitations of ANFIS Implementation

In the above experiment, ANFIS was trained with the static data collected from model simulation with PID controller. So the least square technique have been used for identification of consequent parameters which converges very quickly with low RMS error. However, in his later work, Jang [26] has used self learning method with temporal back propagation for training of ANFIS controller. It is expected that such an approach may improve dynamic performance. Also in the above experiments, a very small number of rules have been used due to limitations in personal computer. Use of higher number of rules with complete partitioning of the input data space, may increase the performance of ANFIS [12].

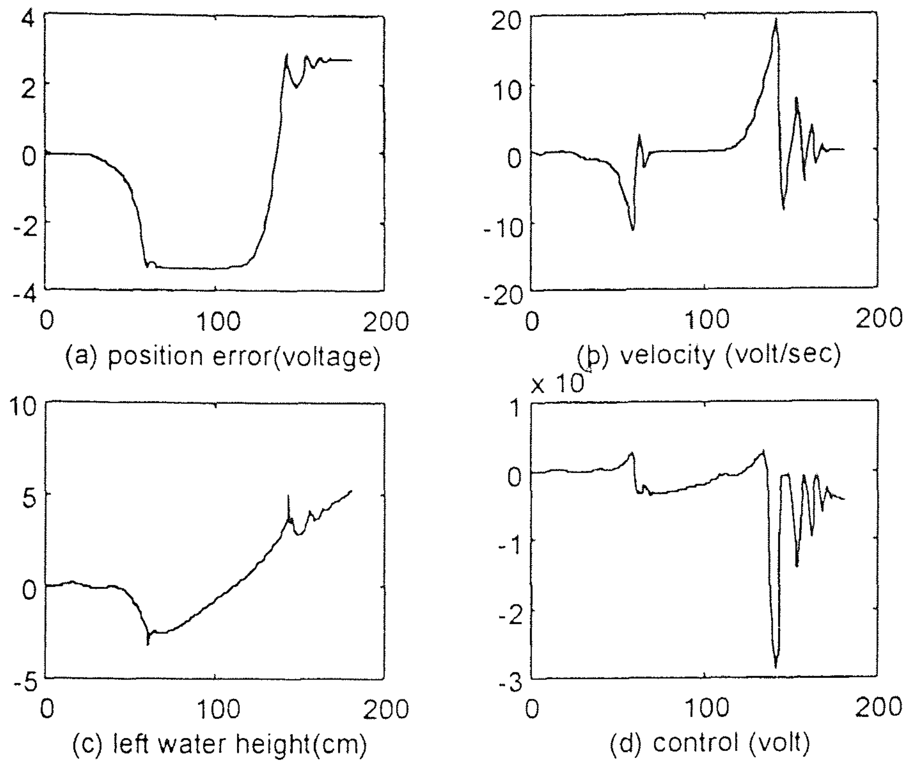


Figure 5.16 Controller failure (real time)

5.5 Modified Rules for ANFIS and GeNFIS

The first three layers of ANFIS and GeNFIS have the same type of structure. However, in case of GeNFIS the structure is predetermined. Whereas in ANFIS, structure depends on the number of membership functions for each input. ANFIS uses all possible combinations of membership functions for each input in rule construction. So the structure of ANFIS increases rapidly as the number of input and the number of membership functions of each input increases [53]. Also, ANFIS uses Sugeno type rules in the consequent part, which is very difficult to define by a linguistic label. To overcome these problems, various tests were conducted with both ANFIS and GeNFIS, with minor modifications in the structures as well as in the rules. Results of two such tests are presented here.

5.5.1 Experimental Results

Test 1: ANFIS with preselected rule antecedent and Sugeno type consequent:

The same 11 rules which have been used earlier with GeNFIS, are used here with modified consequent. In this case, consequent part of each rule is replaced by a Sugeno type consequent. So the consequent part of each rule is now having four modifiable parameters. The initial values of all these parameters are set to zero. The rules are now like:

$$\text{if } x \text{ is } A \text{ and } y \text{ is } B \text{ and } z \text{ is } C \text{ then } u = px + qy + rz + s \qquad (5.4)$$

The same data were used for training, and after about 20 cycles minimum RMS error was observed as 0.32 (0.016 normalized value). Figure 5.17 shows the simulation results. It is clear from the simulation results that the state variables (error) can not smoothly reach to the desired (zero) state. Experiment results as shown in Figure 5.18 also indicate failure of the controller.

Test 2: ANFIS and GeNFIS with modified Sugeno type consequent:

In this test, the same 11 rules were used with some modification in the Sugeno type consequent. The constant 's' is now replaced by a predefuzzified consequent. Predefuzzified consequent is also known as a fuzzy singleton. In this case, 's' of each rule is initialized by the value of center (bell shaped MF) of corresponding linguistic level. Rest of the parameters are initialized with zero as before. In training algorithm, instead of using the combination of least-squares and gradient descent methods, only gradient descent method is used.

The same data were used for training. However, in using only the gradient descent method, the training took about 600 cycles to reach a level of 0.65 RMS error (0.033 normalized value). Figure 5.19 shows the simulation results with initial position errors of

0.03. The results indicate that the controller with modified rules can balance the beam. Figure 5.20 shows the experimental results.

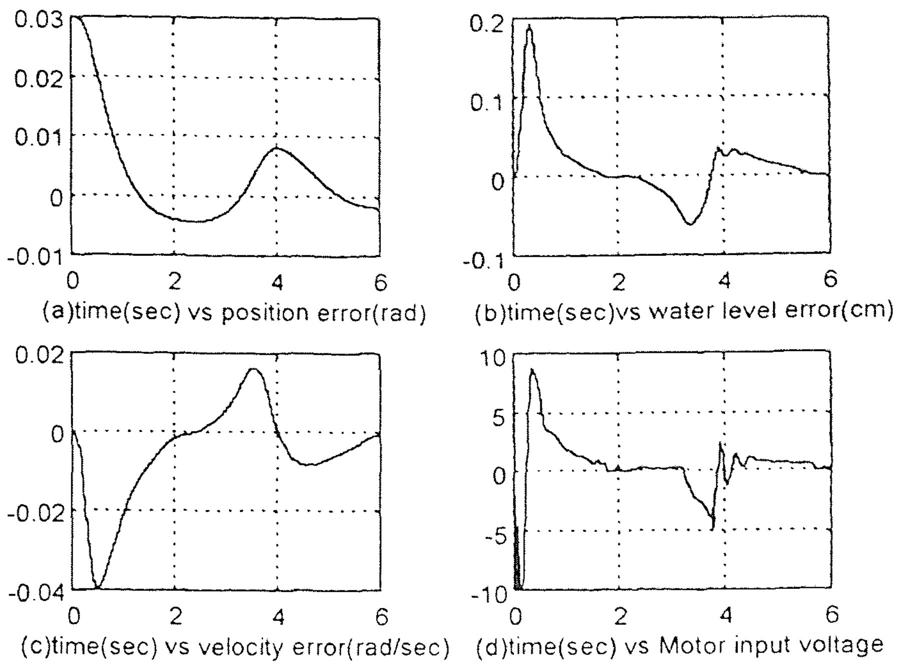


Figure 5.17 Simulation: Failure of ANFIS with preselected rule antecedent

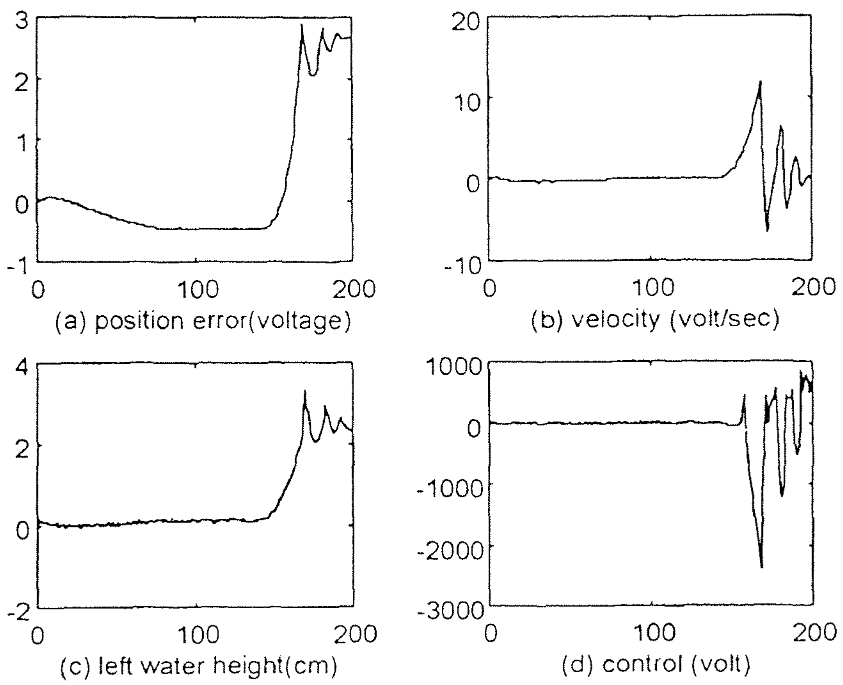


Figure 5.18 Experiment: Failure of ANFIS with preselected rule antecedent

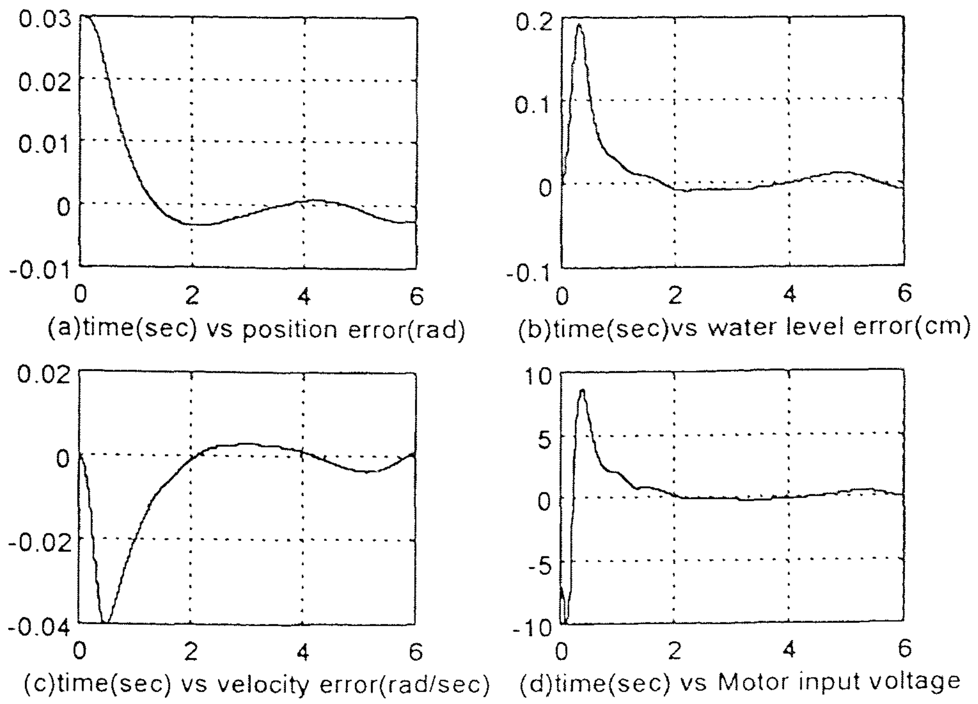


Figure 5.19 Simulation of ANFIS with modified rules

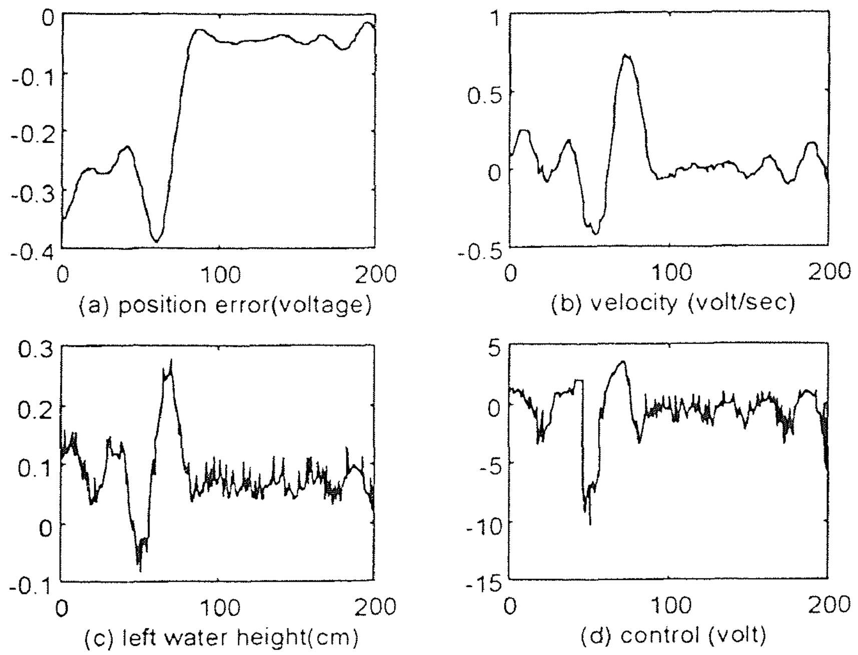


Figure 5.20 Experiment Results: ANFIS with modified rules

The results obtained in this section point out some interesting features of the neuro-fuzzy controllers. The most noteworthy of these is the fact that even when the matching of the input data is very well, i.e. small RMS error, the controller may not be successful in balancing of the beam. In the next chapter, these and other results are summarized.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this dissertation, a flexible network called Generalized Network-based Fuzzy Inferencing System (GeNFIS) is developed by combining the good features from neural networks and fuzzy inferencing systems, and is applied to the problem of controlling a higher order nonlinear dynamic system. The nonlinear dynamic system used here is an unstable balance beam system that contains two fluid tanks, one at each end, and the balance is achieved by pumping the fluid back and forth from the tanks. Both the results, simulation as well as experimental, indicate that GeNFIS controller can successfully balance the beam in real time. The GeNFIS is designed to be flexible and can accomodate different variations of rule structure and defuzzification schemes. The results obtained are summarized in this chapter.

First, the performance of GeNFIS controller was evaluated against a conventional PID controller. The test results indicate that the PID controller is more stable and sensitive to the set point than GeNFIS controller. However, it may be mentioned that the GeNFIS is a rule-based controller and only eleven rules have been used to control the system, and did not explicitly require the exact model of the plant. Moreover, the performance of GeNFIS can be improved by increasing the number of rules. Besides, the objective here was not to find a best neuro-fuzzy controller for comparison with a conventional PID controller, but to find a feasible neuro-fuzzy controller with a flexible structure which can be used to investigate the current neuro-fuzzy approaches from literature for performance evaluation.

Next, as a part of the flexible structure for GeNFIS, a concept of synchronous defuzzification is proposed in this work. This method allows simultaneous use of different types of membership functions in the consequent part of GeNFIS rules and provides the flexibility to incorporate strengths from other neuro-fuzzy approaches. Using this concept, different versions of GeNFIS are tested with the balance beam control problem, and the results of these tests are presented next.

First, synchronous defuzzification is implemented on GeNFIS with simultaneous use of triangular as well as bell shaped membership functions. However, no significant improvement is noticed with the use of this scheme for this particular nonlinear dynamic system. Nevertheless, through this experiment it is shown that it is possible to combine two different defuzzification schemes in any proportion, train the controller and achive success in balance of the beam. The structure developed can allow for future experiments on different nonlinear systems. In the next experiment, the triangular membership functions are replaced by the first order Sugeno type rules, thus there is a simultaneous use of Sugeno type rules as well as bell shaped membership functions. A hybrid learning scheme [25,27] is used, such that the consequent parameters of Sugeno type rules are trained through least squared error minimization, while rest of the parameters are trained through back propoagation. The weights of the final layer are all set to 0.5 for equal mixing of results from each defuzzification scheme. The results in this case are not satisfactory, i.e., the beam balancing is not possible. When the weight of 0.25 is used in the consequent layer of Sugeno type rules, the controller is able to balance the beam. However, the performance of the controller is not very smooth. This brings out an interesting feature of the proposed neuro-fuzzy control scheme. It appears that the proposed scheme is better at beam balancing under the supervised learning than the scheme which uses Sugeno type rules. Since ANFIS also uses Sugeno type rules, this is a motivation for some additional tests on balance beam system using only ANFIS [27] type controller under supervised learning. In this work, ANFIS means the use of Sugeno type rules in the consequent part of GeNFIS along with hybrid learning scheme. The system is tested with only 12 rules ($3\times2\times2$) using complete partitioning of the input data space as discussed in [27]. In this case, the controller fails to balance the beam. It appears that although the training is good, i.e. resulting in a very low RMS error, the tests under dynamic conditions indicate failure of the controller. This failure can be attributed in part to the fact that only a small number of fuzzy labels, hence rules are utilized.

Next, in order to do a fair comparison of GeNFIS and ANFIS, the premise parts of the original 11 rules (used in GeNFIS) are used with Sugeno type consequent and the training is done by using hybrid learning method. This scheme is equivalent to ANFIS with pre-selected, limited set of rules. However, this scheme, although succesful in simulation, fails to balance the beam for the actual physical experiment. In both these experiments, the training data is matched very well by ANFIS-like schemes, yet they fail in dynamic situations. It is noted that these results are a special case where a limited number of rules, and a specific training data are utilized, thus the failure of ANFIS-like scheme is not a general conclusion. However, these results raise a flag regarding the use of Sugeno type rules when the rule base is limited. For practical problems where one cannot utilize a full set of rules, or an optimal smaller set of rules is not known, one may excercise caution in using ANFIS-like schemes, and it may be worthwhile to consider the proposed GeNFIS scheme.

Although a detailed mathematical analysis of the use of Sugeno type rules in ANFIS is not done in this work, it appears that the hybrid learning scheme may produce arbitrary values for the Sugeno constants, and it would be difficult to assign any good physical significance to the values obtained. Consequently, a modification in the consequent part of Sugeno type rules is proposed. This modification replaces the constant term by a suitable fuzzy singleton in order to bound the rule output around the desired linguistic label (say for example, *high*). For training, back propagation algorithm may still be used. As shown in the previous chapter, the test results for ANFIS/GeNFIS controller using modified version of Sugeno type rules were satisfactory. A summary of results is presented in Table 6.1.

Table 6.1 SUMMARY OF RESULTS

NO	CONTROLLER	LEARNING TYPE	TRAINING CYCLES/ RMSE	WEIGHTS (OUTPUT LAYERS)	PERFOR-MANCE (Disturbance Level inVolts)
1	GeNFIS with 11 Rules	Gradient Descent(GD)	800/0.0293	1.00	6
2	GeNFIS Synchronous defuzzification with triangular MFs (11 Rules)	GD	600/0.044	0.5	3
3	GeNFIS Synchronous defuzzification with Sugeno Rules (11 Rules)	Hybrid LSE for Sugeno Rules GD for Rest	For Bell Shaped MFs 800/0.03 For Sugeno 20/0.016	0.5	Failure
4	GeNFIS Synchronous defuzzification with Sugeno Rules (11 Rules)	Hybrid LSE for Sugeno Rules GD for Rest	For Bell Shaped MFs 800/0.03 For Sugeno 20/0.016	0.75 for Bell Shaped MFs 0.25 for Sugeno Layers	4
5	ANFIS with 12 (3x2x2) Rules (7 input MFs)	LSE forward pass. GD Back propagation	10/0.016	1.00	Failure
6	ANFIS/GeNFIS with 11 Rules. Same premise as # 1	LSE forward pass. GD Back propagation	10/0.017	1.00	Failure
7	ANFIS/GeNFIS with modified Sugeno type rules (fuzzy singleton)	GD Back propagation	600/0.033	1.00	4

In summary, a flexible framework for neuro-fuzzy modeling is developed. The proposed method "GeNFIS" has been used successfully for control of a nonlinear dynamic system using real time control methodologies. However, considerable amount of research is needed to explore the full potential of GeNFIS and to develop a suitable self learning scheme for GeNFIS. The GeNFIS architecture along with the fluid beam balancing system developed here provide an excellent test-bed for further research in neuro-fuzzy modeling and real time control.

5.2 Future work

The GeNFIS has been developed with the objective of providing a flexible structure which can accommodate different types of fuzzy control rules. However, a suitable learning scheme is requred so that the controller can learn from the system behavior. The possible approachs are to implement reinforcement learning or real time recurrent learning schemes on GeNFIS architecture. In the present work, the rules were generated from the expert knowledge along with the training data set. No formal algorithm for rule generation is developed. Considerable amount of research is needed to develop a rule generation algoritm for GeNFIS. A discussion on the rule generation procedure, used in this research, has been presented in chapter 4. The methods used here may be used as a guideline for the development of an automatic rule generation algorithm.

Another area of further research is development of techniques to find a suitable fuzzy controller which can balance the beam from the raw sensor data. In the present balance beam system, two linear pressure sensors are used to measure the water pressures of two tanks. Although the relation between water pressure and sensor output voltage is linear, the constant amount of water may have different output voltage at different angular positions due to geometric effect. The actual water height is a function of position and pressure, which is calibrated by using Least Square Method. See Appendix C for details.

APPENDIX A

A.1 TRAINING OF GeNFIS

The training of the network includes two phases: *forward* pass and *backward* pass. Before training, the network architecture has to be built by determining the number of inputs, number of linguistic labels for each input, and the total number of rules. The construction of rules is not automatic in GeNFIS structure, so the antecedent parts of each rule have to be identified and stored in a data file with the respective consequent labels in a sequential manner, such as very small to very high.

Training data sets are organized in pairs of input vector and output vector. In the forward pass, each node produces a output signal based on its input and its function. In the backward pass, each node generates a updating vector of its parameters depending on the gradient descent learning algorithm.

Layer 0 (input layer L0)

$$O_i^{L0} = x_i \tag{A.1}$$

Layer 1: (fuzzifying layer L1)

$$O_i^{L1} = \mu_i(O_i^{L0})$$

$$O_i^{L1} = \frac{1}{1 + \left[\dfrac{O_i^{L0} - g_i}{a_i}\right]^{2d_i}} \tag{A.2}$$

Layer 2 (T-norm operator)

As discussed in GNFIS, a *softmin* operator is used instead of *min or product* operator. The output of *layer 2* is the rule firing strength w_i.

$$O_i^{L2} = w_i = \frac{\sum_i \mu_i(O_i^{L0}) e^{-k\mu_i(O_i^{L0})}}{\sum e^{-k\mu_i(O_i^{L0})}} \tag{A.3}$$

Layer 3 (Normalized Rule strength and defuzzifying layer)

The t_{ni} is the normalized rule firing strength, and is given by equation (A.4)

$$t_{ni} = \frac{O_i^{L2}}{\sum O_i^{L2}} \tag{A.4}$$

$$O_i^{L3} = t_{ni} * \mu^{-1}(O_i^{L2})$$

$$= \frac{O_i^{L2}}{\sum O_i^{L2}} \mu^{-1}(O_i^{L2}) \tag{A.5}$$

In *layer 3*, the defuzzifying membership function is also a bell shaped curve given by:

$$\mu_i(\mu^{-1}(O_i^{L2})) = \frac{1}{1 + \left[\frac{\mu^{-1}(O_i^{L2}) - c}{a}\right]^{2b}} \tag{A.6}$$

where $\mu^{-1}\left(O_i^{L2}\right)$ is the local defuzzified value (LDM), and is given as:

$$\mu^{-1}(O_i^{L2}) = c + \frac{K_{df} * a}{3} - \tfrac{2a}{3}\left[\frac{1 - O_i^{L2}}{O_i^{L2}}\right]^{1/2b} \tag{A.7}$$

Layer 4 (Output Layer)

$$O_i^{L4} = O^{L4} = \sum J_i O_i^{L3} \tag{A.8}$$

where J_i s are the weights associated with each input link of output node.

In backward pass the error, difference between the network output (O^{L4}) and desired output (O^d) is computed. The objective function E is the square of this error which is back propagated using the respective derivatives in each layers.

For the output *layer 4*,

$$E = (O^d - O^{L4})^2$$

$$\frac{\partial E}{\partial O^{L4}} = -2 * (O^d - O^{L4}) \tag{A.9}$$

the error rate $\partial E / \partial O^{L4}$ is computed first using equation (A.9).

The parameters in this layer (L_4) is J_i, ,so from (A.8),

$$\frac{\partial O^{L4}}{\partial J_i} = O_i^{L3} \tag{A.10}$$

$$\frac{\partial E}{\partial J_i} = \frac{\partial E}{\partial O^{L4}} \cdot \frac{\partial O^{L4}}{\partial J_i} \tag{A.11}$$

There are three parameters in defuzzifying layer (a, c, b), these will be treated as p_i^{L3}, the parameter set. Using chain rule of derivatives:

$$\frac{\partial E}{\partial P_i^{L3}} = \sum \frac{\partial E}{\partial O^{L4}} \cdot \frac{\partial O^{L4}}{\partial O_i^{L3}} \cdot \frac{\partial O_i^{L3}}{\partial P_i^{L3}} \tag{A.12}$$

Form relation(A.8)

$$\frac{\partial O^{L4}}{\partial O^{L3}} = J_i \tag{A.13}$$

where J_i is the weight associated with each input link of layer 4 nodes.

To compute derivatives of the *layer 3* node's parameter set $\{c,a,b\}$, equations (A.5) and (A.7) are used.

$$O_i^{L3} = t_{n_i}\left[c + \frac{k_{df} * a}{3} - \tfrac{2}{3}\left[\frac{1 - O_i^{L2}}{O_i^{L2}}\right]^{\frac{1}{2b}}\right]$$

$$\frac{\partial O_i^{L3}}{\partial c} = t_{n_i} \tag{A.14}$$

$$\frac{\partial O_i^{L3}}{\partial a} = \tfrac{1}{3}\left[k_{df} - 2\left[\frac{1 - O_i^{L2}}{O_i^{L2}}\right]^{\frac{1}{2b}}\right] \tag{A.15}$$

$$\frac{\partial O_i^{L3}}{\partial b} = \tfrac{2a}{3}\left[\frac{1 - O_i^{L2}}{O_i^{L2}}\right]^{1/2b}\left[\ln\left(\frac{1 - O_i^{L2}}{O_i^{l2}}\right)\right]\frac{1}{2b^2} \tag{A.16}$$

In *layer 2* ,there is no modifiable parameter. Using equation (A.5),

$$\frac{\partial O_i^{L3}}{\partial O_i^{L2}} = \frac{\left[O_i^{L2}\mu^{-1}(O_i^{L2})\right]'\left[\sum O_i^{L2}\right] - \left[\sum O_i^{L2}\right]'\left[O_i^{L2}\mu^{-1}(O_i^{L2})\right]}{\left[\sum O_i^{L2}\right]^2}$$

$$= \frac{\left[\mu^{-1}(O_i^{L2}) + O_i^{L2}\left[\mu^{-1}(O_i^{L2})\right]'\right]\sum O_i^{L2} - \left[O_i^{L2}\mu^{-1}(O_i^{L2})\right]}{\sum O_i^{L2}\sum O_i^{L2}}$$

$$\frac{\partial O_i^{L3}}{\partial O_i^{L2}} = \frac{\mu^{-1}(O_i^{L2}) + O_i^{L2}\left[\mu^{-1}(O_i^{L2})\right]' - O_i^{L3}}{\sum O_i^{L2}} \qquad \text{(A.17)}$$

where O_i^{L2} and O_i^{L3} are on the same rule i.

The derivative of output of i th node in layer 3 $\left(O_i^{L3}\right)$ with respect to the output of j th node of *layer 2* $\left(O_j^{L2}\right)$ is given as

$$\frac{\partial O_i^{L3}}{\partial O_j^{L2}} = \frac{O_i^{L2}\mu^{-1}(O_i^{L2})}{\left[\sum O_i^{L2}\right]^2}$$

$$= \frac{O_i^{L3}}{\sum O_i^{L2}} \qquad \text{(A.18)}$$

Using equation (A.7) $\left[\mu^{-1}(O_i^{L2})\right]'$ may be computed as:

$$\frac{\partial \mu^{-1}(O_i^{L2})}{\partial O_i^{L2}} = -\tfrac{2}{3}\tfrac{a}{2b}\left[\frac{1}{O_i^{L2}} - 1\right]^{\frac{1}{2b}-1}(-1)(O_i^{L2})^{-2}$$

$$\frac{\partial \mu^{-1}(O_i^{L2})}{\partial O_i^{L2}} = \tfrac{a}{3b}\frac{\left[1 - O_i^{L2}\right]^{\frac{1}{2b}-1}}{\left[O_i^{L2}\right]^{\frac{1}{2b}+1}} \qquad \text{(A.19)}$$

There are also three parameters (p_i^{L1}) in each node of *layer 1*. These modifiable parameters $\{a, d, g\}$ are given by the equation of bell shaped membership function.

$$\mu\left(O_i^{L0}\right) = \frac{1}{1 + \left[\dfrac{O_i^{L0} - g_i}{a_i}\right]^{2d_i}} \qquad \text{(A.20)}$$

The chain rule to compute the derivatives of error E with respect to each parameters of *layer1* node is given by

$$\frac{\partial E}{\partial P_j^{L1}} = \sum \frac{\partial E}{\partial O^{L4}} \cdot \frac{\partial O^{L4}}{\partial O^{L3}} \cdot \frac{\partial O^{L3}}{\partial O^{L2}} \cdot \frac{\partial O^{L2}}{\partial \delta O^{L1}} \cdot \frac{\partial O^{L1}}{\partial P_j^{L1}} \qquad \text{(A.21)}$$

where $\frac{\partial O_i^{L2}}{\partial O_i^{L1}}$ may be derived from equation (A.3).

The derivative of the output of layers (O^{li})with respect to its preceding layers ($O^{l(i-1)}$) may be calculated from the following relations:

$$O_i^{l2} = \frac{\sum_i O_i^{L1} e^{-kO_i^{L1}}}{\sum e^{-kO_i^{L1}}} \qquad \text{(A.22)}$$

where $O_i^{L1} = \mu_i(O_i^{L0})$.

$$\frac{\partial O_i^{L2}}{\partial O_i^{L1}} = \frac{\left[\sum_i O_i^{L1} e^{-kO_i^{L1}}\right]' \left[\sum e^{-kO_i^{L1}}\right] - \left[\sum_i O_i^{L1} e^{-kO_i^{L1}}\right]\left[\sum e^{-kO_i^{L1}}\right]'}{\left[\sum e^{-kO_i^{L1}}\right]^2}$$

$$= \frac{\left[e^{-kO_i^{L1}} + O_i^{L1}(-k)e^{-kO_i^{L1}}\right]\left[\sum e^{-k}O_i^{L1}\right] - \left[\sum O_i^{L1} e^{-kO_i^{L1}}\right]\left[-ke^{-kO_i^{L1}}\right]}{\left[\sum e^{-kO_i^{L1}}\right]^2}$$

$$= \frac{\left[e^{-kO_i^{L1}} - kO_i^{L1} e^{-kO_i^{L1}} + ke^{-kO_i^{L1}} . O_i^{L2}\right]}{\sum e^{-kO_i^{L1}}}$$

$$\frac{\partial O_i^{L2}}{\partial O_i^{L1}} = \frac{e^{-kO_i^{L1}}(1 + k(O_i^{L2} - O_i^{L1}))}{\sum e^{-kO_i^{L1}}} \qquad \text{(A.23)}$$

Using equations (A.2), the derivatives of the parameter set $\{a_i, g_i, d_i\}$ of *layer 1* node can be found as:

$$\frac{\partial O_i^{L1}}{\partial a_i} = -\left[1 + \left(\frac{x_i - g_i}{a_i}\right)^{2d_i}\right]^{-2} (2d_i)\left(\frac{x_i - g_i}{a_i}\right)^{2d_i - 1} (x_i - g_i)(-1)a_i^{-2}$$

$$= \frac{2d_i(x_i - g_i)\left(\frac{x_i - g_i}{a_i}\right)^{2d_i}}{a\left[1 + \left(\frac{x_i - g_i}{a_i}\right)^{2d_i}\right]^2} \tag{A.24}$$

$$\frac{\partial O_i^{L1}}{\partial g_i} = -\left[1 + \left(\frac{x_i - g_i}{a_i}\right)^{2d_i}\right]^{-2}(2d_i)\left(\frac{x_i - g_i}{a_i}\right)^{2d_i - 1}\frac{1}{a_i}(-1)$$

$$= \frac{2d_i\left(\frac{x_i - g_i}{a_i}\right)^{2d_i}}{(x_i - g_i)\left[1 + \left(\frac{x_i - g_i}{a_i}\right)^{2d_i}\right]^2} \tag{A.25}$$

$$\frac{\partial O_i^{L1}}{\partial d_i} = -\left[1 + \left(\frac{x_i - g_i}{a_i}\right)^{2d_i}\right]^{-2}\left[\ln\left(\frac{x_i - g_i}{a_i}\right)\right]\left(\frac{x_i - g_i}{a_i}\right)^{2d_i}(2)$$

$$\frac{\partial O_i^{L1}}{\partial d_i} = \frac{-\left[\ln(\left(\frac{x_i - g_i}{a_i}\right)^2\right]\left[\left(\frac{x_i - g_i}{a_i}\right)^{2d_i}\right]}{\left[1 + \left(\frac{x_i - g_i}{a_i}\right)^{2d_i}\right]^2} \tag{A.26}$$

where $x_i = O_i^{L0}$.

APPENDIX B

SYNCHRONOUS DEFUZZIFICATION

B.1 Triangular MF in defuzzification level

Derivations for the consequent layers are only presented:

Layer 4 (Output layer)

$$O^{L4} = \sum w_i O^{L3} \tag{B.1}$$

$$\frac{\partial E}{\partial P_i^{L4}} = \frac{\partial E}{\partial O^{L4}} \cdot \frac{\partial O^{L4}}{\partial P_i^{L4}}$$

where, P_i is a modifiable parameter

$$E = \sum \left(O^d - O^{L4}\right)^2$$

$$\frac{\partial E}{\partial O^{L4}} = 2\Sigma\left(O^{l4} - O^d\right)$$

From equation (B.1) $$\frac{\partial O^{L4}}{\partial w_i} = O^{L3}$$

So $$\frac{\partial E}{\partial P_i^{L4}} = \frac{\partial E}{\partial w_i^{L4}} = 2\Sigma\left(O^{L4} - O^d\right) \cdot O^{L3} \tag{B.2}$$

Layer L3(Defuzzification layer)

For defuzzification Triangular MFs are replaced by linear functions

From Equation 5.2 (chapter 5): (*c*, *p* are modifiable parameters)

$$x = \mu(x)c + p(1 - \mu(x))$$

$$\frac{\partial \mu^{-1}(O_i^{L2})}{\partial c} = \mu(x) \tag{B.3}$$

$$\frac{\partial \mu^{-1}(O_i^{L2})}{\partial p} = 1 - \mu(x) \tag{B.4}$$

Each rule output (For details, see also appendix A)

$$O_i^{L3} = t_m\left[\mu(x)c + p(1 - \mu(x))\right] \tag{B.6}$$

$$\mu^{-1}(O_i^{L2}) = cO_i^{L2} + p(1 - O_i^{L2}) \tag{B.7}$$

$$\frac{\partial \mu^{-1}(O_i^{L2})}{\partial O_i^{L2}} = c - p$$

$$[\mu^{-1}(O_i^{L2})]' = c - p \tag{B.8}$$

B.2 Defuzzification with Sugeno type rule

Layer 4 (Output layer--L4)

$$O^{L4} = \sum w_i O^{L3}$$

Using the same steps as in **B.1**

$$\frac{\partial O^{L4}}{\partial P_i} = \left[-2\Sigma\left(O^d - O^{L4}\right)\right]\left[O^{L3}\right] \tag{B.9}$$

Layer L3(Defuzzification layer--L3)

$$\frac{\partial E}{\partial O^{L4}} = -2\Sigma\left(O^d - O^{L4}\right)$$

$$O^{L4} = \Sigma w_i O^{L3}$$

To find $\frac{\delta O^{L3}}{\delta P_i^{L3}}$ use defuzzify equation

$$f_i = p_i x_1 + q_i x_2 + r_i x_3 + t_i$$

where, f is the rule output, and p, q, r, t are constants.

$$O^{L3} = t_n . f_i$$

where, t_n is the normalized rule firing strength.

$$t_{n_i} = \frac{O_i^{L2}}{\sum O_i^{L2}}$$

So
$$\frac{\partial O^{l3}}{\partial P^{i}} = t_{n_i} x_1 \qquad \frac{\partial O^{L3}}{\partial r_i} = t_{n_i} x_3$$

$$\frac{\partial O^{L3}}{\partial q_i} = t_{n_i} x_2 \qquad \frac{\partial O^{L3}}{\partial t} = t_{n_i} \tag{B.10}$$

$$O_i^{L3} = \frac{O_i^{l2}}{\sum\limits_{i=0}^{\#L2} O_i^{L2}} \left(\sum_{i=1}^{\#ofinput} p_i x_i + t_i \right) \tag{B.11}$$

$$R_{outi} = \sum_{i=1}^{\#ofinput} p_i x_i + t_i \tag{B.12}$$

Where, R_{outi} is output of rule i.

$$\frac{\partial O_i^{L3}}{\partial O_i^{L2}} = \frac{\left[O_i^{L2} R_{outi}\right]' \left[\Sigma O_i^{L2}\right] - \left[\Sigma O_i^{L2}\right]' \left[O_i^{L2} R_{outi}\right]}{\left[\Sigma O_i^{L2}\right]^2}$$

$$\frac{\partial O_i^{L3}}{\partial O_i^{L2}} = \frac{R_{outi} - O_i^{L3}}{\Sigma O_i^{L2}} \tag{B.13}$$

and,
$$\frac{\partial O_i^{L3}}{\partial O_i^{L2}} = - \frac{O_i^{L3}}{\Sigma O_i^{L2}} \tag{B.14}$$

APPENDIX C

BALANCE BEAM PARAMETERS

C.1 Balance Beam System Parameters

See chapter 3 for details of balance beam system model (section 3.2.1):

x_1 = angular position of beam

x_2 = angular velocity of beam

h= height of water in left tank

Q= flow rate of water

B= friction coefficient of bearing

=0.015 newton m.s/rad = 150000 dyne cm.s/rad

$T(x_1, h)$ = torque due to water

$$= 261186.67*(2h-H)*\cos(x_1) + 1114.3265 *(0.5*(h^2 + (H-h)^2)+4.2*H*\sin(x_1)) + 4190627* \sin(x_1)) \tag{C.1}$$

H = total water height (left tank water height h plus the water height of right tank)

$J(h)$= rotational moment of inertia of the system

$$= 389836.99+2.8353*((h^3 + (H-h)^3)+47.6328*(h^2 + (H-h)^2) +6463.21*H \tag{C.2}$$

A= area of tank

= 3.14*(1.9*1.9) sq cm

K_{pump}= motor constant of pump

=1.389

T_{pump}= time constant of motor

0.061 sec

U = output of controller (voltage)

C.2 Equilibrium Water Height

From system model, as presented in chapter 3 (Section 3.2.1), the net torque $T(x_1,h)$ due to water heights should be zero to keep the beam in equilibrium at a given set point (or angular position). For every position x_1 there is a relative equilibrium left water height *h_ref* to balance the beam, i.e., to make $T(x_1,h)=0$. Instead of solving the nonlinear equation of $T(x_1,h)$ to find *h_ref* (as given in the equation C.1), a linear equation is approximated by using least square method within the working range of angular position for a given total water height H. The relationship among the slopes of the lines with respect to different total heights H is also linear and may be treated as a constant within the operating range of H (from 8 cm to 14 cm).

h_ref= equilibrium height of the left tank (i.e. the water height needed to make $T(x_1,h) = 0$), is given by:

$$h\_ref = (-7.429 - 0.2238*H)*x_1 + H/2 \tag{C.3}$$

C.2.1 Cascaded Control Loop

Using above equations, a state feedback control law is given by:

$$\begin{aligned} U(k) = kp*(x_{1\text{-}ref}(k) - x_1(k)) + ki*\Sigma(x_{1\text{-}ref}(k) - x_1(k)) + \\ kd*(x_{2\text{-}ref}(k) - x_2.estimate(k)) \\ + km*(h(k) - h\_ref(k)) \end{aligned} \tag{C.4}$$

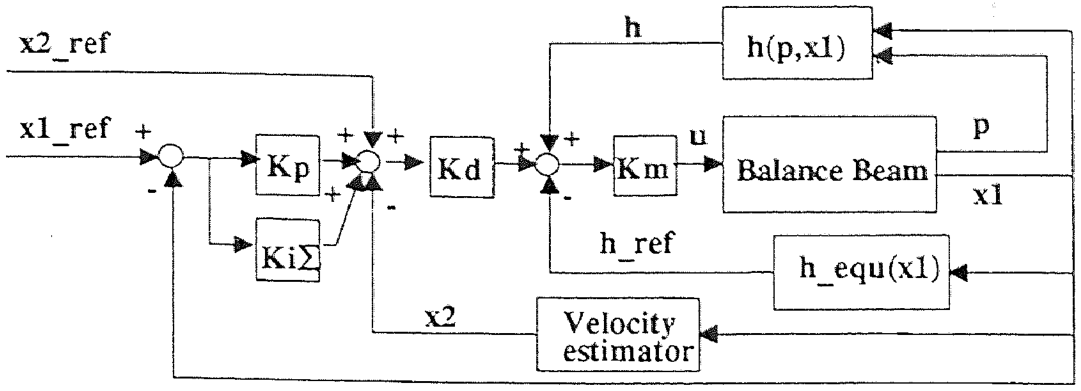
where

$x_{1\text{-}ref}$= position set point

$x_{2\text{-}ref}$= velocity set point

$x_2.estimate$ = estimated velocity=$(x_1(k+1) - x_1(k))/sample\_time;$.

The cascaded control loops equivalent to the control law C.4 is shown in Figure C.1. Where the control gains are *km*=Km, *kd*= Km*Kd, *kp*=Km*Kd*Kp, and *ki*= Km*Kd*Ki.



x1 = position; **x2** = velocity; **p** = pressure; **h** = water height; **u** = control voltage

Figure C.1 Cascaded Control Loops

The following control gains have been used in simulation:

Kp = 2.0 * 0.1*55.0*2.0/0.046

Ki = 4.0 * 0.1*55.0*2.0/0.046

Kd = 0.1 * 0.1*55.0*2.0/0.046

Km = 55.0*2.0 where 0.046 is the conversion factor from volts to radians. The total water height H is used as 10.4 cm in all simulation.

Key to Keep Balance. The Figure C.2 indicates that to balance the beam in the position

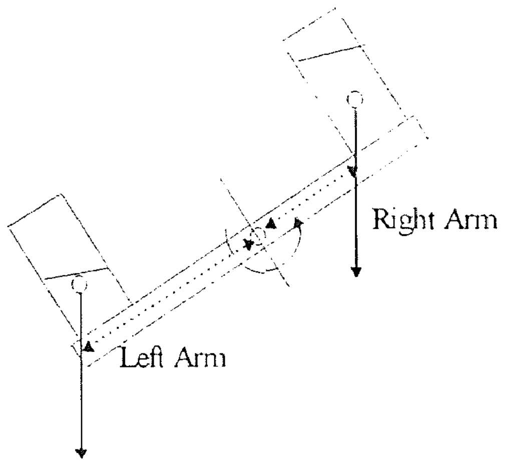


Figure C.2 Net torque in the same direction of rotation

as shown in figure, the right tank water height should be higher than the left tank water height since the right moment arm is much less than the left moment arm. To move the beam from above equilibrium position to the horizontal position (i.e. to rotate clock wise), first a little water is to be pumped from left tank to right tank in order to start the rotation in clockwise direction and then immediately the water should be pumped in the opposite direction to keep the balance until it arrives in the horizontal position.

C.2.1 Push-Pull Control

As mentioned in chapter 3, two equivalent gear pumps in parallel have been used to pump the water back and forth from the tanks. The "Push-Pull" strategy is used to avoid the dead zone of the pumps. Generally the dead zone exists in the region of small input where input cannot incur effective output. In order to avoid dead zone and also to use the linear region of pumping operation, two actuators are used in the same control channel but in the opposite directions such that their total effect is equivalent to the original desired control effect. If C is the central operating voltage of the pumps, and the desired control voltage is U, then the following relations are used to compute voltage input of each pump:

$u1 = C + U/2;$

$u2 = -(C - U/2);$

Where, u1 and u2 are the input voltages of pump1 and pump2 respectively. This strategy will also prevent the pumps from changing directions.

C.2.2 Pressure Sensor Calibration

Two linear pressure sensors are used to measure the water pressures of two tanks. Although the relation between water pressure and sensor output voltage is linear, the constant amount of water may have different output voltage at different angular positions due to geometric effect. The actual water height is a function of position and pressure.

The following relation is used to find the water height from the position and pressure sensors data.

$$\text{water\_height} = C0 + C1*\text{position} + C2*\text{position}*\text{pressure} + C3*\text{pressure} \qquad (C.4)$$

Where water heights are read from the rulers on tanks at horizontal beam position while values of position and pressure come from the potentiometer and pressure sensors respectively. The coefficients C0,C1,C2, and C3 are obtained by using Least Square Method during calibration procedure. In the present system a dynamic calibration method is used in order to reduce the hysteresis effect [37].

REFERENCES

[1] H.Adeli, and S.L.Hung,"Fuzzy neural network learning model for image recognition," *Integrated Computer-Aided Engineering*, Willey-Interscience,vol 1,no 1,1993.

[2] K.J. Astron and B. Wittenmark, *Computer Controller Systems: Theory and Design*, Prentice Hall, Engle wood Cliffs, NJ, 1984.

[3] D.M Auslander and C.H. Tham, *Real-Time Software for Control*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[4] D.M.Auslander, G.Hanidu, A.Jana, S.Landsberger, S.Seif, and Y.Young, "Mechatronics Curriculum in the Synthesis Coalition," in *Proc. of 1992 IEEE/ CHMT Int'l Electronic Manufacturing. Tech. Symp.*, Baltimore, MD, Sept. 1992, pp. 165-168.

[5] D.M.Auslander, "Unified real time task notation for mechanical system control," *Symposium on Mechatronics*, DSC-Vol.50, ASME 1993, pp.143-149.

[6] D.M.Auslander, G.Hanidu, A.Jana, K.Jothimurugesan, S.Seif, andY.Young, "Tools for Teaching Mechatronics," *Proc. of 1993 ASEE Annual Conference*, Univ. of Illinois, IL, June 18-21,1993.

[7] D.M Auslander and Paul Sagues, *Microprocessor for Measurement and Control*, Berkeley, CA, Osborne/McGraw-Hill, 1981.

[8] A.G.Barto, R.S.Sutton, and C.W.Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems, " *IEEE Trans. Systems., Man, and Cybernetics.*, vol. SMC-13, no. 5,pp. 834-846,1983.

[9] H.R.Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Nural Networks*, vol. 3,pp.724-740, May 1992.

[10] H. R.Berenji, "Fuzzy Q-learning: A new approach for fuzzy dynamicprogramming," in *Proc. IEEE Int. Conf. on Fuzzy Syst.*, Orlando, FL, June 1994. vol.1,pp.486-487.

[11] H.R.Berenji, "A reinforcement learning-based architecture for fuzzy logic control," *Int.J.of Approximate Reasoning*,vol.6,pp267-292, 1992.

[12] P.P.Bonissone, V. Badami, K.H.Chaing, P.S.Khedkar, K.W.Marcelle, and M.J. Schutten, "Industrial applications of fuzzy logic at General Electric," in *Proc. IEEE*, vol.83, no.3.,pp.450-465, March1995.

[13] M.Brown, and C. Haris, *Neurofuzzy Adaptive Modelling and Control*, Prentice Hall Hall International (UK) Limited,1994.

[14] S.-C.Chang, "*Adaptive Nonlinear Control Using Fuzzy Logic and Neural Network*," Ph.D Dissertation, Department of Mechanical Engineering, NJIT, May 1994.

[15] I.Hayashi,H.Nomura,H.Yamasaki,and N.Wakami, "Construction of fuzzy inference rules by NDF and NDFL," *Int.J.of Approximate Reasoning*,vol.6,pp241-261, 1992.

[16] C.M.Higgins, and Rodney M. Goodman, "Fuzzy rule-based networks for control," *IEEE Trans. Fuzzy Syst.*,vol.2 ,no.1,pp. 82-88, Feb.1994.

[17] G.E.Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, no.1, pp.143-150, 1989.

[18] K.Hornik, M. Stinchcombe, and H White, "Multilayer feed forward networks are universal approximators," *Neural Networks*, no. 2, pp 359-366, 1989.

[19] L.R.Hunt, and J. Turi, "A new algorithm for constructing approximate transformation for nonlinear systems," *IEEE Transaction on Automatic Control*, 38:10, pp. 1553-1556.

[20] K.J.Hunt, D. Sbarbaro, R.Zbikowski, and P.J.Gawthrop, "Neural Networks for control systems-A survey," *Automatica*, vol.28., no.6, pp1083-1112, 1992.

[21] A.Jana, and D.M. Auslander, "Workcell Programming Environment for Intelligent Manufacturing Systems," *Design and Implementation of Intelligent Manufacturing Systems,* Parsaei, H.R., and Jamshidi, M., Eds., Chapter 1, pp. 1-18, 1995. Prentice-Hall, 1995

[22] A.Jana, D.M.Auslander,R.N Dave, and S.S. Chehl, "Task Planning for Cooperating Robot Systems," *Proc. of 1994 International Symposium on Robotics And Manufacturing,* Maui, Hawaii, August 14-18, 1994.

[23] A.Jana, R.N. Dave, S.S. Chehl, C. S. Wang, "Computer-Integrated Manufacturing and Robotics Laboratory Design for Undergraduate Education," *Proc. of 1994 International Symposium on Robotics And Manufacturing,* Maui, Hawaii, August 14-18, 1994.

[24] A.Jana, S.S.Chehl, "Developing Instrumentation Laboratory with Real Time Control Component," *Proc. of 1993 American Control Conference*, San Francisco, June 1993, pp. 2046- 2049.

[25] J.-S.R.Jang and C.-T.Sun, "Neuro-fuzzy modeling and control," in *Proc. IEEE,* vol. 83, no.3., pp.378-406, March 1995.

[26] J.-S.R.Jang, "Self-learning fuzzy controller based on temporal back-propagation," *IEEE Trans. Nural Networks*, vol.3, pp.714-723, Sept.1992.

[27] J.-S.R.Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Systems., Man, and Cybernetics.*, vol. 23,pp.665-685,May 1993.

[28] J.-S.R.Jang, "Structure Determination in fuzzy modeling: A fuzzy CART approach," in *Proc. IEEE Int. Conf. on Fuzzy Syst.*, Orlando, FL, vol. 1, pp480-485,June 1994.

[29] J.-S.R.Jang and C.-T.Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. Nural Networks*, vol. 4, no.1 pp.156-159, Jan 1993.

[30] H. D. Jerro, A Jana, R.N. Dave, "Multiprocessing Systems: A Design Experience," *1994 ASEE/GSW Conference*, Baton Rouge, LA, March '94.

[31] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach.*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[32] B. Kosko, "Fuzzy systems as universal approximators," in *Proc. of IEEE Int. Conference on Fuzzy Systems* 1992, SanDiego, March 8-12, pp. 1153-1162.

[33] W.A.Kwong, K.M.Passino, E.G.Laukonen, and S.Yurkovich, "Expert supervision of fuzzy learning systems for tolerant aircraft control," in *Proc. IEEE*, vol.83, no.3., pp.466-483, March1995.

[34] C.-C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller-Part 1," *IEEE Trans. Systems., Man, and Cybernetics.*, vol. 20,pp.404-418,Feb.1990.

[35] C.-C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller-Part 2," *IEEE Trans. Systems., Man, and Cybernetics.*, vol. 20,pp.419-435, Feb.1990.

[36] C.-C. Lee, "A self learning rule based controller employing approximate reasoning and neural net concept," *Int. J. Intell.Syst.*, vol. 5, no. 3, pp. 71-93, 1991.

[37] M.Lemkin, P.H. Yang, and D.M.Auslander, "Using hydraulically balanced beam as a case study in control implementation for control education," *IFAC* 1994.

[38] A.U.Levin,and K.S. Narendra, "Control of nonlinear dynamical systems using neural networks: Controllability and stabilization," *IEEE Trans. Nural Networks*, vol.4, no.2, pp.192-206,March 1993.

[39] C.-T.Lin and C.S.G Lee, "Reinforcement structure/parameter learning for neural-network based fuzzy logic control systems," *IEEE Trans. Fuzzy Syst.*,vol. 2, Jan. 1994.

[40] C.-T.Lin and C.S.G Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Computers*, vol. 40. pp.1320-1336, Dec.1991.

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

[42] J.M.Mendel, "Fuzzy logic systems for engineering: A tutorial,"in *Proc. IEEE*, vol.83 no.3.,pp345-377, March 1995.

[43] K.S. Narendra,and K.Parthasarathy, "Identification and control of dynamical systems using Neural Networks," *IEEE Trans. Neural Networks*, vol. 1, no.1 pp.4- 27, March 1990.

[44] K.S. Narendra,and K.Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol. 2, no.2, pp. 241-262, March 1991.

[45] D.H.Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Syst. Magazine*, pp.18-23, Apr. 1990.

[46] Y.-M. Park, U. Moon, and K.Y.Lee, "A self-organizing fuzzy logic controller for dynamic systems using a fuzzy auto-regressive moving average (FARMA) model," *IEEE Trans. Fuzzy Syst.*,vol.3, no.1, pp 75-82,Feb.1995.

[47] T.J.Procyk and E.H.Mamdani, "A linguistic self-organizing process controller," *Automatica*, vol.15,pp.15-30,1978.

[48] H. Serraji, "A new approach to adaptive control of manipulators," *Transaction of the ASME, Journal of Dynamic Systems, Measurement, and Control* 109, pp. 193-202.

[49] G.Strang, *Introduction to Applied Mathematics,* Wellesley-Cambridge Press, Wellesley, MA,1986.

[50] M.Sugeno,Ed.,*Industrial Applications of Fuzzy Control.* New York: Elsevier, 1985.

[51] M.Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol.1,pp.7-31,Feb. 1993.

[52] M.Sugeno and G.T.Kang., "Structure Identification of fuzzy model," *Fuzzy Sets and Systems*,vol.28,pp.15-33,1988.

[53] C.-T.Sun, "Rulebase structure identification in an adaptive network based fuzzy inference system," *IEEE Trans. Fuzzy Syst.*,vol.2 pp. 64-73, Feb.1994.

[54] C.-T.Sun ,T.Y.Shuai, and G.-L. Dai, "Using fuzzy filters as feature detectors," in *Proc. IEEE Int. Conf. on Fuzzy Syst.*, Orlando, FL, June 1994. vol.1,pp.406-410.

[55] T.Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control", *IEEE Trans. Systems., Man, and Cybernetics.*, vol. 15, pp. 116-132, 1985.

[56] H.Takagi, and I. Hayashi, "NN-driven fuzzy reasoning," *Int.J.of Approximate Reasoning*,vol.5,pp191-212, 1991.

[57] K.Tanaka and M. Sugeno, "Stability analysis and design of fuzzy control systems,"*Fuzzy Sets and Systems* vol.45,pp.135-156,1992.

[58] Y.Tsukamoto, "An approach to fuzzy reasoning method," in *Advancec in Fuzzy Set Theory and Applications*, M.M. Gupta, R.K.Ragade, and R.R.Yager, Eds. Amsterdam: North-Holland, 1979, PP.137-149

[59] L.-X. Wang, and J.M. Mendel, "Back-propagation fuzzy systems as nonlinear dynamic system identifiers," in *Proc. IEEE Int. Conf. on Fuzzy Systems*, vol. 1, pp. 161-170, March 1992.

[60] B.Widrow and M.A.Lehr, "30 years of adaptive neural networks: Perceptron, madline, and backpropagation,"in *Proc. IEEE*, vol.78 pp.1415-1442, Sept.1990.

[61] R.R.Yager and D.P.Filev, "SLIDE: A simple adaptive defuzzification method," IEEE Trans.Fuzzy Syst., vol. 1,pp. 69-78, Feb. 1993.

[62] L.A Zadeh, "Fuzzy sets," *Information and control*, Vol. 8,pp 338-353,1965.

[63] L.A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans.Syst., Man,and Cyber.*, vol.3,pp.28-44, Jan.1973.

[64] L.A.Zadeh, *Industrial Application of Fuzzy Logic and Intelligent Syatems*, J.Yen, R. Lanagri, and L. A. Zadeh, Eds. New York., IEEE Press,1995.