CONTROLLER DESIGN FOR BALL BEAM SYSTEM USING VARIOUS CONTROL TECHNIQUES

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Control and Guidance)

By

SWATI SWARNKAR

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2011

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation entitled **"CONTROLLER DESIGN FOR BALL BEAM SYSTEM USING VARIOUS CONTROL TECHNIQUES"** submitted for the award of the degree of **Master of Technology** with specialization in **Control & Guidance** in the Department of Electronics & Computer Engineering, **Indian Institute of Technology Roorkee**, under the guidance of **Dr. R. Mitra,** Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee.

Date : 20/06/2011

Place: Roorkee

[illegible]

Swati Swarnkar

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

[illegible]

(Dr. R. Mitra)

Department of Electronics & Computer Engineering,

Indian Institute of Technology Roorkee

Roorkee-247667, India

ACKNOWLEDGEMENT

I express my foremost and deepest gratitude to **Dr. R. MITRA**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee for his valuable guidance, support and motivation for this work. The valuable hours of discussion and suggestions that I had with him have undoubtedly helped in supplementing my thoughts in the right direction for attaining the desired objective. I consider myself extremely fortunate for having got the opportunity to learn and work under his able supervision over the entire period of my association with him.

My sincere thanks to faculty members of Control and Guidance for their constant encouragement, caring words, constructive criticism and suggestions towards the successful completion of this work. My sincere thanks to the laboratory staff to access the computers and other resources at will for completion of this work.

Last but not the least, I am highly indebted to my parents, friends and family members, whose sincere prayers, best wishes, moral support and encouragement have a constant source of assurance, guidance, strength and inspiration to me.

ABSTRACT

The ball and beam system represents a standard nonlinear plant for both classical and modem control techniques. This system is widely used because it is simple to understand as a system and provides the opportunity to analyze the control techniques. The main aim of this research is to develop an intelligent controller to control the ball position and is able to perform satisfactory in presence of parameter variation and external disturbance. In order to achieve this objective, the thesis investigates the performance of a robust nonlinear control method called Sliding Mode Control (SMC) and various Artificial Intelligence (AI) techniques.

In the formulation of any control problem, there will typically be discrepancies between the actual plant and the mathematical model developed for controller design. This mismatch may be due to unmodelled dynamics, variation in system parameters or the approximation of complex plant behaviour by a straightforward model. The designer must ensure that the resulting controller has the ability to produce required performance levels in practice despite such plant/model mismatches. This has led to an intense interest in the development of robust control methods which seek to solve this problem. One particular approach to robust-control controller design is the so-called sliding mode control methodology. Since, the actual model contains the ball on beam dynamics as well as dynamics of DC motor, making the controller design task difficult. So, initially a simplified model has been used (without considering DC motor). An SMC has been designed for this model. Next, an SMC has been designed for more realistic model.

Although SMC is able to stabilize the system but it requires the model of system to be known. As modeling these complex nonlinear systems is often troublesome, hence, various soft computing techniques has been investigated, which allow controller design on model free basis. A Fuzzy Logic Controller has been designed for simplified ball beam system. Then, learning and generalizing capabilities of Neural Network has been explored. Finally, a hybrid approach called Adaptive Neuro Fuzzy Inference System (ANFIS) controller has been proposed. These techniques have been applied to ball beam system and are evaluated by simulation as well as real time control with the help of laboratory set up. In the end, the simulation and experimental results obtained from various controllers have been compared.

Table of Contents

CHAPTER 1

Introduction

As this dissertation work is extension of the project work, the work done in the project has been included in brief along with the work that was proposed to be done in dissertation. The present work concerns the application of some of the conventional and modern control techniques to control the Ball-Beam system. The ball beam system is inherently unstable system, in which ball tends to move to one of the ends of the beam. The system is highly unstable because for bounded control input or bounded change in beam angle, produces unbounded movement of ball. The task is thus to apply a sequence of upward and downward forces of fixed magnitude to control the system so that the ball can be dynamically balanced at any location of the beam within a short period of time and beam can be kept in horizontal position.

To control the Ball Beam system, various conventional and modern control schemes have been applied earlier. The classical (P, PI, PID) control technique has been the basis in simple control systems. Its simplicity has been the main reason for its wide applications in industry. Since classical controllers are fixed-gain feedback controllers, they can't compensate the parameter variations in the plant and can't adapt changes in the environment. Standard linear techniques are based on approximate linear models which is valid only around a small region of an operating point. In addition, in conventional techniques, mathematical modelling of the plants and parameter tuning of the controller has to be done before implementing the controller. Most real systems, relevant from a control perspective, exhibit nonlinear behaviour; furthermore, to model these systems are often troublesome. The need to overcome such problems and to have a controller well-tuned not only for one operating point but also for a whole range of operating points has motivated the other control techniques and the idea of a robust and adaptive controller.

Nonlinear control techniques, on the other hand, taking account of plant non-linearities in the design of control law are capable of achieving better performance in a greater operating region. Specifically Sliding Mode Control (SMC) has been widely accepted as an effective method in dealing with uncertainties such as parameter variations and external disturbances. Sliding Mode controllers are a class of robust controllers which uses discontinuous control. By proper design of the switching or sliding surface, VSC attains the conventional goals of control such as stabilization, tracking, regulation, etc. These controllers, however, require an infinitely (in the ideal case) fast Switching mechanism. The phenomenon of non ideal but fast switching was labelled as chattering. The high frequency components of the chattering are undesirable because they may excite unmodeled high-frequency plant dynamics.

In the last four decades, numerous alternative control techniques, such as neural and fuzzy control, have been proposed instead of conventional classical technique. Development of artificial neural networks (ANN'S) and fuzzy logic theory (FL) have inspired new resources for possible implementation of better and more efficient control. ANN'S have capability of learning the dynamical systems that estimate input-output functions. Fuzzy systems transform sets of structured information into the appropriate control actions. Especially, neither ANN nor fuzzy systems need mathematical modeling of the plants. Fuzzy control systems can be developed along with linguistic lines and need some expertise information about the plant. On the other hand, before used for control purposes, ANN have to be trained and they need some information (not based on mathematical model but sometimes taken measurement from plant) about the plant.

In this dissertation work, first of all, a Sliding Mode Controller has been designed for ball beam system. Essentially, SMC is a kind of Variable Structure Control (VSC) which utilizes a high-speed switching control law to drive the nonlinear plant's state trajectory onto a specified and user-chosen surface in the state space (called the sliding or switching surface), and to maintain the plant's state trajectory on this surface for all subsequent time. . The SMC controller developed assures desired behaviour of the closed loop system. Then, Fuzzy controller has been designed for Ball Beam system. The design of Fuzzy Logic Controller (FLC) is based on the knowledge of expert about the behaviour of system and it does not require rigorous mathematical calculation as in case of SMC. Next, the two controllers designed earlier (SMC and FLC) are utilized to achieve two level control action called supervisory controller. The two level control includes the simplicity and smooth control action of FLC as well as robustness of SMC. Then, several other soft computing methods have been employed to control the Ball Beam system. These include Neural Network Controller and ANFIS controller. The learning and generalizing capability of ANNs to imitate a particular input-output mapping presents a new approach to control complex nonlinear systems. ANFIS is essentially a Fuzzy Inference System (FIS) which provides the functioning and advantages of an FLC but is generated with the help of training ability of Neural Network. Hence, it represents a hybrid neuro-fuzzy technique to deal with complex nonlinear problems. Finally, the performance of each controller has been evaluated by MATLAB/Simulink Simulation. Moreover, some of these controllers will also be applied to real time control of Ball Beam system and performance has been compared.

1.1 Problem Statement

The prime objectives of this research work will focus on developing Sliding Mode and Fuzzy Logic controllers, applied to ball beam system individually and in combination resulting in two level control scheme. Then, designing and examining the control performance of Neural Network technique to control ball beam system. Finally, a hybrid neuro-fuzzy approach called ANFIS has been proposed. The control performance of various controllers designed will be compared on the basis of various specifications such as settling time, maximum overshoot, steady state error etc.

1.2 Literature Review

Various control schemes have been discussed in literature and have been applied to control the Ball Beam system. The control of Ball beam system using PD controller has been discussed in [1]. Several nonlinear control schemes for the ball beam system can be found in [2][3]. Conventional constant gain controllers used for ball beam system becomes poor due to strong non-linearities, modelling inaccuracy, parameter variation and uncertainties. Although PID control is a proficient technique for the handling of non-linear systems and can be used to control nonlinear systems [1] but modelling these complex nonlinear systems is often difficult and sometimes impossible using the laws of physics. Hence using a classical controller is not suitable for nonlinear control application [4]. This necessitates the design of robust controllers which makes the system performance insensitive to parameter variations and uncertainties.

The models used in feedback design should include some unstructured uncertainty to cover unmodeled dynamics, particularly at higher frequency. For a definite class of nonlinear systems there is an appropriate robust control method called sliding mode control [5]. The subject has been treated in detail in [6]. This control method can be applied very well in the presence of model uncertainties, parameter fluctuations and disturbances provided that the upper bounds of their absolute values are known. The disadvantage of this method is the drastic changes in the manipulated variable. However, this can be avoided by a small modification: a boundary layer is introduced near the switching line which smoothes out the control behaviour and ensures the states remaining within the layer [7]. Given that the upper bounds of the model uncertainties are known, stability and high performance of the controlled system are guaranteed.

To overcome the difficulty resulted due to modelling and to utilize the advantage of model free approach, interest in developing Artificial intelligence techniques has increased considerably. There have been an increasing amount of publications presenting research and implementations of artificial neural networks (ANNs) and fuzzy logic (FL). Being branches of artificial intelligence (AI), both utilizes the human way of using past experiences, adapting themselves accordingly and generalizing.. The work of L. A. Zadeh on Linguistic approach and system analysis based on the theory of fuzzy sets [8] was the basis of Fuzzy Logic. Zadeh's work motivated Mamdani and his colleagues to research on fuzzy control [9]-[11]. Fuzzy Logic Control is useful when the processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted qualitatively, inexactly, or uncertainly [12]. Fuzzy modelling or fuzzy identification, first explored systematically by Takagi and Sugeno [13], has found numerous practical applications in control, prediction and inference. FLC gives better transient and steady state performance than PID in case of nonlinear systems [14]. Membership functions make the controller less sensitive to slight variations in the physical parameters. It does not require any system modelling or complex mathematical equations governing the relationship between inputs and outputs. It typically takes only a few rules to describe systems that may require several lines of conventional software code, which reduces the design complexity [15]. In [16], single input FLC has been presented which results in less computational requirement. The application of Fuzzy controller for real time control of Ball and beam system has been discussed in [17].

On the other hand, ANN works with parallel connected units, called neurons, which process inputs in accordance with their adaptable weights usually in a recursive manner for approximation. The pioneering work of McCulloch and Pitts was the foundation stone for the growth of ANN architectures. In their paper, McCulloch and Pitts suggested the unification of neuro-physiology with mathematical logic, which showed way for some significant results in NN research. Perceptron rule and the LMS algorithm, the two early rules for training adaptive elements were first published in 1960. In the years following these discoveries, many new techniques have been developed in the field of neural networks [18] and the discipline is growing rapidly. The first major extension of the feedforward neural network beyond Madaline I took place in 1971 when Werbos developed a backpropagation training algorithm which, in 1974, he first published in his doctoral dissertation. In 1982, Parker rediscovered the technique and in 1985, published a report on it at M.I.T. Not long after Parker published his findings, Rumelhart, Hinton, and Williams [19] also rediscovered the technique and as a result of the clear framework, they presented their ideas and they finally succeeded in making it widely known. The elements used by Rumelhart et al. in the backpropagation network differ from those used in the earlier Madaline architectures. The adaptive elements in the original Madaline structure used hard-limiting quantizers (signums), while the elements in the backpropagation network use only differentiable nonlinearities, or "sigmoid" functions. In digital implementations, the hard-limiting quantizer is more easily computed than any of the differentiable nonlinearities used in backpropagation networks [18]. ANNs have the ability to approximate a non-linear function. The performance of ANNs depend greatly on data used for training, [20] gives the idea for generating data and type of proper signal to be used for training. In past, ANNs have been employed by several researchers to control the ball and beam system., The real time control of Ball Beam system by a two layer network, developed using error back propagation (BP), temporal difference, and the reinforcement learning algorithms has been discussed in [21]. ANNs have proven superior learning and generalizing capabilities even on completely unknown systems that can only be described by its input– output characteristics. A neural network developed with genetic algorithms [22] shows the benefit of achieving more efficient solutions than regular neural network learning methods such as backpropagation (BP).

The disadvantage in case of fuzzy controllers is problem involved with tuning, this implies the handling of a great quantity of variables like the ranges of the membership functions, shape of these functions, percentage of overlap among the functions, the design of the rule base, mainly, and when system is multivariable system, the number of parameters grows exponentially with the number of variables. On the other hand, ANNs can be advantageous since they provide an adjusting mechanism and can adapt themselves to changes in the environment or optimize the system after an initial guess, which leads to learning and generalizing. Hence, various attempts have been made to combine these two techniques. A neural integrated fuzzy controller for Ball Beam system has been discussed in [23] which extracts fewer rules from operator and generates remaining rules by utilizing learning and interpolating capabilities of ANNs, taking advantage of the best of FLC and ANN. A Neural-Fuzzy algorithm for ball beam control system has been discussed in [24]. The superior features of ANNs combined with FLC to create hybrid neuro-fuzzy controller called Adaptive Neuro Fuzzy Inference System (ANFIS) was proposed by Jang [25]. ANFIS is also proved to be useful in modelling complex non-linear system utilizing input-output data sets [26]. A good example of how ANFIS is designed is given in [27], where it has been applied to control autonomous underwater vehicle.

1.3 Organization of Dissertation

This thesis consists of nine chapters and the topics covered in various chapters are:
Chapter 1 gives the introduction part as well as literature review.
Chapter 2 deals with the modelling and operation of Ball Beam System.
Chapter 3 discusses the design of Sliding Mode controller for Ball Beam system.
Chapter 4 demonstrates the Fuzzy Logic and its application to Ball Beam system control.
Chapter 5 discusses the two level supervisory control design which comprises of FLC as well as Sliding mode controller.
Chapter 6 investigates Neural Network controller design for ball beam system
Chapter 7 discusses the hybrid Neuro Fuzzy approach called ANFIS which combines the capabilities of Fuzzy as well as Neural techniques to control complex nonlinear systems.
Chapter 8 discusses the performance of various controllers by demonstrating simulation and experimental result as well as comparison between various controllers.
Chapter 9 summarizes the whole dissertation.

CHAPTER 2

Ball and Beam System Modelling

2.1 Ball Beam System Description

The Ball and Beam plant consists of a ball, beam, DC motor, lever arm, gear and belt pulley as shown in Fig. 2.1. The steel ball can roll freely along the whole length of the beam. The beam, at one end, is connected to lever arm, which is controlled by DC motor through gear and pulley. Two sensors are used to provide feedback information to the control system. An angle sensor is equipped in motor to provide current rotary position of motor shaft and other sensor, which is linear position sensor, is equipped along length of the beam that senses current position of ball on the beam. As the servo gear turns by an angle θ, the lever changes the angle of the beam by α. When the angle is changed from the horizontal position, gravity causes the ball to roll along the beam.

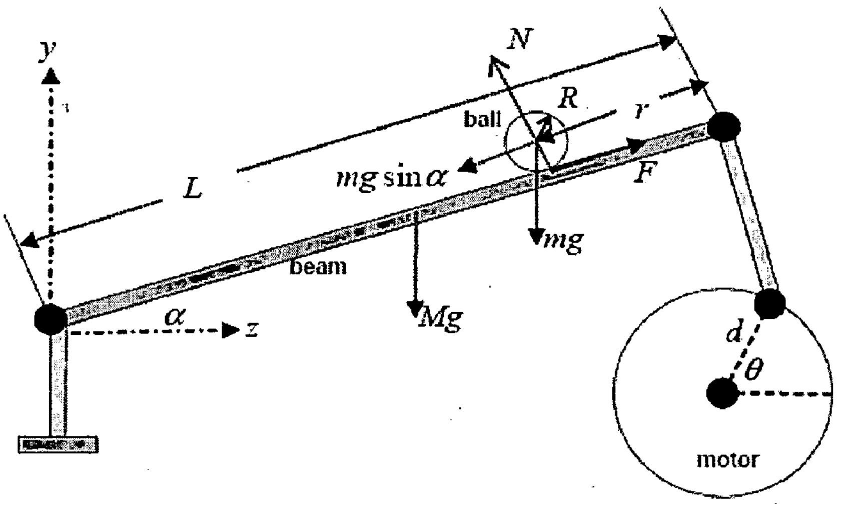

Fig. 2.1 Ball and Beam system

The objective is to design and implement controller which controls voltage to the motor in such a way so that desired position of ball is stabilized on the beam by changing the angle of the beam. This is difficult control task because the ball does not stay in one place on the beam but moves with an acceleration that is proportional to the tilt of the beam. In the control terminology, the system is open loop unstable because the system output (ball position) increases without limit for a fixed input (beam angle). Feedback control must be used to keep the ball in desired position on the beam.

2.2 Simplified Model (Neglecting The Motor Dynamics)

Let us consider, for simplicity, we neglect the motor dynamics. Then, the system model consists of ball on a beam only. The equation of motion of ball along the beam, using Lagrangian model, as given in [28], is

$$\left(\frac{J}{R^2}+m\right)\ddot{r}+mg\sin\alpha-mr\dot{\alpha}^2=0 \tag{2.1}$$

Where, g is acceleration due to gravity (m/s^2)

m is mass of the ball (Kg)

R is radius of the ball (m)

J is ball moment of inertia (Kg m^2)

r(t) is ball position coordinate and *α(t)* is beam angle

2.3 Complete Model

The actual mathematical description of the system consists of the dynamics of a DC servomotor and the dynamic model of the ball on the beam. Modelling DC servomotor can be divided into electrical and mechanical two subsystems.

The equations of motion describing the Ball on a Beam system can be written as [29]:

$$(mr^2+k_1)\ddot{\alpha}+(2mr\dot{r}+k_2)\dot{\alpha}+(mgr+\frac{L}{2}Mg)\cos\alpha=u \tag{2.2}$$

$$k_4\ddot{r}-r\dot{\alpha}^2+g\sin\alpha=0 \tag{2.3}$$

Where, r(t) : ball position

θ(t) : servo gear angle

m: mass of the ball

M : mass of the beam

L : length of the beam

g : gravitational constant

The parameters of the system are

R_m : armature resistance of the motor

K_m : motor torque constant

J_m : effective moment of inertia

K_g : gear ratio

K_b : back EMF constant

J_1 : moment of inertia of the beam

d : lever arm offset

V_i : input voltage to motor

i_m : armature current

L_m : armature inductance

V_b : back EMF

B_m : viscous friction coefficient

$V_b = K_b\dot{\theta}$

$\dot{\alpha}$: angular velocity of the beam.

J_2 : moment of inertia of the ball

k_1, k_2, k_3 and k_4 are functions of the system parameters as:

$$k_1 = \frac{R_m J_m L}{K_m K_g d} + J_1$$

$$k_2 = \frac{L}{d}\left(\frac{K_m K_b}{R_m} + K_b + \frac{R_m B_m}{K_m K_g}\right)$$

$$k_3 = 1 + \frac{K_m}{R_m}$$

$$k_4 = 7/5$$

$u(t) = k_3 V_i(t)$: control input to the ball on a beam system

2.3.1 Equilibrium of the system :

The equilibrium can be found from the dynamic equations (2.2) and (2.3) as,

$$\left(mgr_e + \frac{L}{2}Mg \right)\cos \alpha_e = u_e$$
$$\sin \alpha_e = 0$$

Since, the beam angle α is limited by $-\pi/2 < \alpha < \pi/2$, then $\sin \alpha_e = 0$ implies that $\alpha_e = 0$. Therefore, the above equation implies that $r_e = \frac{u_e}{mg} - \frac{L}{2}\frac{M}{m}$. Hence, any equilibrium point of the Ball on a Beam system must be such that $\alpha_e = 0$ and $r_e = r_d$, where r_d is desired ball position.

CHAPTER 3

Sliding Mode Controller

3.1 Overview of Sliding Mode Control

Sliding mode has been successfully applied to the problem of maintaining stability and consistent performances for nonlinear systems with modelling errors. In control theory sliding mode control, is a form of variable structure control (VSC). It is a nonlinear control method that alters the dynamics of a nonlinear system by application of a high-frequency switching control. The multiple control structures are designed so that trajectories always move toward a switching surface, and hence the ultimate trajectory will not exist entirely within one control structure. Instead, the ultimate trajectory will slide along the boundaries of the control structures. The motion of the system as it slides along these boundaries is called a sliding mode.

Intuitively, for a dynamic system sliding mode control uses practically infinite gain to force the trajectories to slide along the restricted sliding mode subspace. The main strength of sliding mode control is its robustness. Because the control can be as simple as a switching between two states (e.g., "on"/"off" or "forward"/"reverse"), it need not be precise and will not be sensitive to parameter variations that enter into the control channel. Additionally, because the control law is not a continuous function, the sliding mode can be reached in finite time (i.e., better than asymptotic behaviour). Sliding mode control is an appropriate robust control method for the systems, where modeling inaccuracies, parameter variations and disturbances are present.

Sometimes sliding mode control has a demerit of chattering of the control variable and some of the system states. The strengths of SMC include: Low sensitivity to plant parameter uncertainty, greatly reduced-order modeling of plant dynamics and finite-time convergence (due to discontinuous control law).

3.2 Sliding Mode Condition

With sliding mode controller, the system is controlled in such a way that the error in the system states always moves towards a sliding surface. The sliding surface is defined with the tracking error (e) of the state and its rate of change ($\dot{e}$) as variables. The distance of the error trajectory from the sliding surface and its rate of convergence are used to decide the control input (u) to the system. The sign of the control input must change at the intersection of the tracking error trajectory with the sliding surface. In this way the error trajectory is always forced to move towards the sliding surface. The system in sliding mode can be described by a linear system of lower order than the original system. The state asymptotically approaches the state origin on the intersection of the hyperplanes.

Consider the class of nonlinear systems that can be modelled by following equation:

$$x'' = A(X) + B(X)u$$

Then the problem of designing the sliding mode controller is to find:

(1) m switching functions, represented in vector form as S(X), the surface which represents desired system dynamics on which control u has discontinuity.

(2) A variable structure control

$$u = u^{+}(X) \quad \text{when} \quad S(X) > 0$$

$$u = u^{-}(X) \quad \text{when} \quad S(X) < 0$$

such that any state X outside the switching surface is driven to reach the surface in finite time. On the switching surface, the sliding mode takes place, following the desired dynamics. In this way the overall system is globally asymptotically stable.

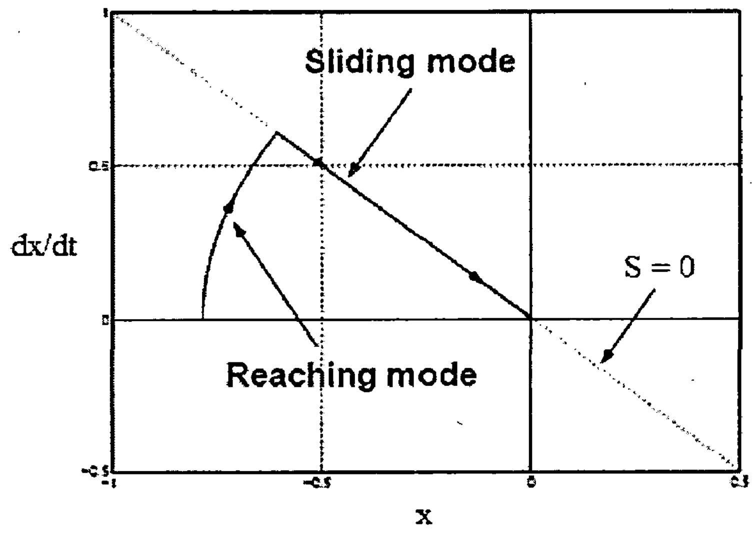

Fig. 3.1 Phase plot showing Sliding mode

Suppose at time t = 0, S(x,t) > 0.Then, state x reaches the switching line S = 0 in a finite time. Then, it crosses the switching line and enters the region S(x,t) < 0 , resulting in the value of u being altered from u+ to u-. Depending on the value of system parameters and sliding function co-efficient, the state trajectory may continue in the region S(x,t) < 0, yielding bang-bang control or the state trajectory may immediately re-cross the switching line and enter the region S(x,t) > 0. This yields sliding motion. Assuming that the switching logic works infinitely fast, the state x is constrained to remain on the switching line S = 0 by the control which oscillates between the values u+ and u-.

For sliding motion to occur, we need on opposite sides of the switching line,

$$\lim_{S\to 0^+} \dot{S} < 0 \qquad \text{and} \qquad \lim_{S\to 0^-} \dot{S} > 0$$

This ensures the motion of state x on either side of S = 0 is towards the switching line. Hence the two conditions can be combined to give,

$$S\dot{S} < 0$$

in the neighbourhood of switching surface.

3.3 Sliding Mode Controller design for Ball Beam system

3.3.1 SMC design for simplified Ball Beam model:

The Lagrange model of the system developed in eqn. (2.1) is

$$\left(\frac{J_b}{R^2}+m\right)\ddot{r}+mg\sin\alpha-mr\dot{\alpha}^2=0 \tag{3.1}$$

Let, for small tilt in the beam, sin$\alpha \approx \alpha$, then,

$$\ddot{r}=\frac{1}{\left(\frac{J}{R^2}+m\right)}\left[-mg\alpha+mr\dot{\alpha}^2\right]$$

Since,

$$\alpha=\frac{d}{L}\theta$$

Although the control input is applied to motor from where the motion is transferred through gear to beam, but for simplicity in design, motor is neglected, so that the control input u directly changes the gear angle θ. Hence,

$$\theta = u$$

So,

$$\alpha = \frac{d}{L}u$$

Now, let $r = x_1$

Then, the state equations can be written as,

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{m x_1 x_4^2}{k_1} - \frac{mgd}{k_1 L}u$$
$$= f_s + h_s u$$

Let, the sliding surface be,

$$s_1 = \dot{y} + \lambda(y - r_d) \tag{3.2}$$

Where,

$$y = x_1$$

$$\dot{y} = x_2$$

Hence,

$$s_1 = x_2 + \lambda(x_1 - r_d)$$

Where, s_1 = $x_2+\lambda(x_1-r_d)$ = 0 is also called the switching surface. The term switching illustrates the fact that the control law u commutes while crossing the line s_1 = 0. The behaviour of this second- order system on the switching line s_1 = $x_2+\lambda(x_1-r_d)$ = 0 is described by first order differential equation $x_2+\lambda(x_1-r_d)$ = 0. It is important to note, that the behaviour of our system on s_1 = 0 is dependent only on the slope λ of the switching line. This means the system is insensitive to any variation or perturbation of the plant parameters contained in state space model.

If r_d is a constant value, then

$$\dot{s}_1 = \dot{x}_2 + \lambda x_2$$

$$or, \dot{s}_1 = f_s + h_s u + \lambda x_2 \tag{3.3}$$

Now, if we choose u as,

$$u = \frac{1}{h_s}[-f_s - \lambda x_2 - p_1 \operatorname{sgn}(s_1)] \tag{3.4}$$

Where, p_1 is a positive constant, then it can be shown from (3.3) and (3.4) that,

$$\dot{s}_1 = -p_1 \operatorname{sgn}(s_1)$$

Which guarantees that

$$s_1 \dot{s}_1 < 0$$

i.e., the system states reach the desired trajectory [30] after a finite time.

3.3.2 SMC design for Complete Ball Beam model :

In this design, the dynamic model developed in eqn. (2.2) and (2.3) will be used. However, for this model, the relative degree of the system is not well defined around the equilibrium $\alpha = 0$. Hence, the centrifugal acceleration term $r\dot{\alpha}^2$ is neglected, so that it results in a well defined relative degree of the system and Sliding mode control law is designed as [31]:

From eqn. (2.2) and (2.3),

$$\ddot{\alpha} = \frac{1}{(mr^2 + k_1)}\left[u - (2mr\dot{r} + k_2)\dot{\alpha} - \left(mgr + \frac{L}{2}Mg\right)\cos\alpha\right]$$

$$\ddot{r} = -\frac{g}{k_4}\sin\alpha$$

Let, the state variables representing the system states be,

$$x_1 = r$$
$$x_2 = \dot{r}$$
$$x_3 = \alpha$$
$$x_4 = \dot{\alpha}$$

Then, the equations of the system can be written as,

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\frac{g}{k_4}\sin x_3$$
$$\dot{x}_3 = x_4$$
$$\dot{x}_4 = \frac{1}{(mx_1^2 + k_1)}\left[u - (2mx_1x_2 + k_2)x_4 - \left(mgx_1 + \frac{L}{2}Mg\right)\cos x_3\right]$$

Let, the output of the system be,

$$y = x_1$$

Then,

$$\dot{y} = x_2$$
$$\ddot{y} = -\frac{g}{k_4}\sin x_3$$

$$y^{(3)} = -\frac{g}{k_4}x_4\cos x_3$$
$$y^{(4)} = \frac{g}{k_4(mx_1^2 + k_1)}\left[-u\cos x_3 + (2mx_1x_2 + k_2)x_4\cos x_3 + \left(mgx_1 + \frac{L}{2}Mg\right)\cos^2 x_3 + x_4^2(mx_1^2 + k_1)\sin x_3\right]$$
$$= f_s + h_s u$$

Where,

$$f_s = \frac{g}{k_4}\left[\left\{\frac{(2mx_1x_2 + k_2)x_4 + (mgx_1 + \frac{L}{2}Mg)\cos x_3}{(mx_1^2 + k_1)}\right\}\cos x_3 + x_4^2\sin x_3\right]$$

$$h_s = \frac{-g}{k_4(mx_1^2 + k_1)}\cos x_3$$

Let, the sliding surface is defined as

$$s_2 = \dddot{y} + b_1\ddot{y} + b_2\dot{y} + b_3(y - r_d)$$

$$= -\frac{g}{k_4}x_4 \cos x_3 - b_1\frac{g}{k_4}\sin x_3 + b_2 x_2 + b_3(x_1 - r_d)$$

Define the sgn function to be,

$$\operatorname{sgn}(\varphi) = \begin{cases} +1 & \text{if } \varphi > 0 \\ -1 & \text{if } \varphi < 0 \end{cases}$$

To guarantee switching, we need,

$$s_2\dot{s}_2 < 0$$

If p_2 is a positive constant, then the sliding mode control given by,

$$u = \frac{1}{h_s}\left(-f_s + b_1\frac{g}{k_4}x_4 \cos x_3 + b_2\frac{g}{k_4}\sin x_3 - b_3 x_2 - p_2 \operatorname{sgn}(s_2)\right) \tag{3.5}$$

always results in

$$s_2\dot{s}_2 < 0$$

i.e., it will asymptotically stabilize the output of the system to its desired value r_d and forces the other states of the system to their equilibrium values.

Since, $\dot{s}_2 = \ddddot{y} + b_1\dddot{y} + b_2\ddot{y} + b_3\dot{y}$

By substituting u in the above equation, we get

$$\dot{s}_2 = -p_2 \operatorname{sgn}(s_2)$$

3.4 Problem of Chattering

An ideal sliding mode does not exist in practice since it would imply that the control commutes at an infinite frequency. In the presence of switching imperfections, such as switching time delays and small time constants in the actuators, the discontinuity in the feedback control produces a particular dynamic behaviour in the vicinity of the surface, which is commonly referred to as chattering and is undesirable for proper operation of system. This phenomenon is a drawback as, even if it is filtered at the output of the process, it may excite unmodeled high frequency modes, which degrades the performance of the system. Chattering also leads to high wear of moving mechanical parts and high heat losses in electrical power circuits. That is why many procedures have been designed to reduce or eliminate this chattering.

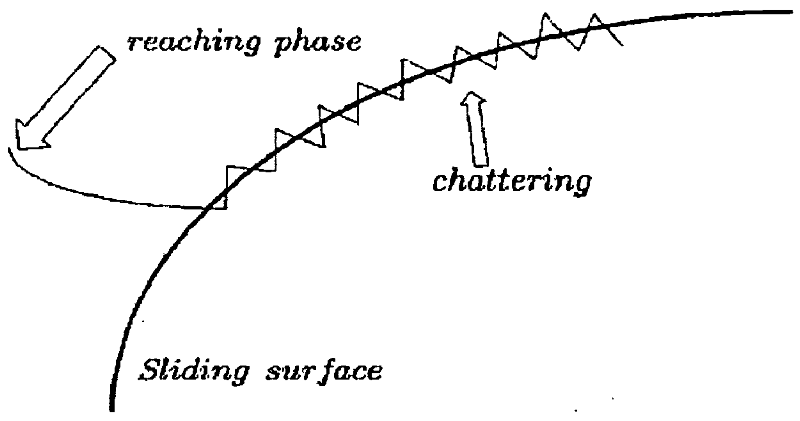

Fig. 3.2 Chattering Phenomenon

One way to reduce chattering consists in a regulation scheme in some neighbourhood of the switching surface which, in the simplest case, merely consists of replacing the signum function by a continuous approximation with a high gain in the boundary layer: for instance, sigmoid functions or saturation functions as shown in Fig. 3.3 and Fig.3.4.

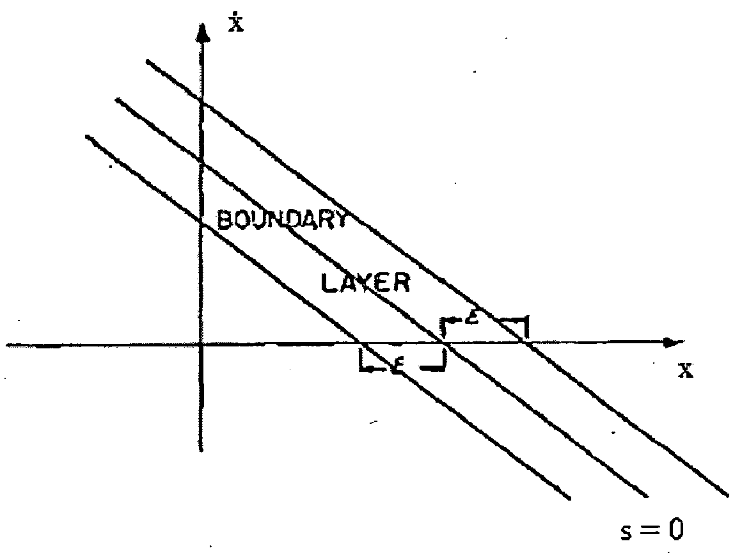

Fig. 3.3 Boundary layer around sliding line

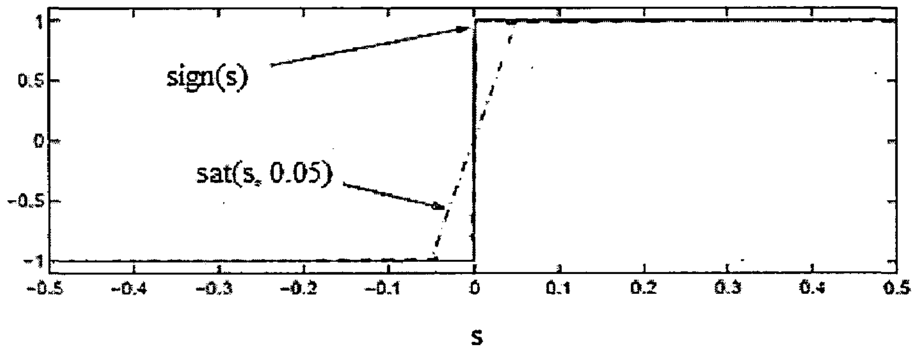

Fig. 3.4 Continuous approximation of the sign-function

Slotine and Sastry suggested a solution to avoid drastic changes in control by smoothing out the control discontinuity in a thin boundary layer neighbouring the switching surface. This is achieved by choosing control law u outside boundary layer as before (i.e. satisfying the sliding condition, which guarantees boundary layer attractiveness) and then interpolating u inside boundary layer. We now quantify the trade-off thus achieved between tracking precision and robustness to unmodelled high frequency dynamics. The method of softening the discontinuous control part by a continuous approximation is very attractive for its simplicity. One possible candidate is the saturation function [7], [30].
If we substitute function sgn(s) by sat (s/ϵ) [32] in eqn. (3.4) and (3.5), where,

$$sat\left(\frac{s}{\varepsilon}\right) = \begin{cases} s/\epsilon & \text{if } |s/\epsilon| < 1 \\ \operatorname{sgn}(s/\epsilon) & \text{if } |s/\epsilon| \geq 1 \end{cases}$$

Where, ϵ is width of boundary layer.

However, although the chattering can be removed, the robustness of sliding mode is also compromised. Another solution to cope with chattering is based on the recent theory of higher-order sliding modes.

CHAPTER 4

Fuzzy Control

4.1 Overview of Fuzzy Control

Fuzzy systems have been successfully applied to a wide variety of practical problems. Recent advances of fuzzy memory devices and fuzzy chips [33] [34] make fuzzy systems especially suitable for industrial applications. In [35], it has been shown that fuzzy systems are universal approximators, i.e., they are capable of approximating any real continuous function on a compact set to arbitrary accuracy. FLCs are suitable to be used in process control systems [36]. FLC has the advantage over conventional controllers as it does not need the exact mathematical model of the system and thus it does not rely upon dynamically changing parameters.

Fuzzy logic starts with the concept of a fuzzy set. A fuzzy set is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership. In fuzzy logic, the truth of any statement becomes a matter of degree. Any statement can be fuzzy. The major advantage that fuzzy reasoning offers is the ability to reply to a yes-no question with a not-quite-yes-or-no answer. In the practical world, we generally use modifiers like very, little, good, high, exactly etc. to express the extent related to a quantity. Humans do this kind of thing all the time (think how rarely you get a straight answer to a seemingly simple question), but it is a rather new trick for computers.

4.1.1 Fuzzy sets

The main concept of fuzzy theory is a notion of fuzzy set. Fuzzy set is an extension of crisp set. Zadeh gave the following definition:

A fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between zero and one.

A fuzzy set is denoted by an ordered set of pairs, the first element of which denotes the element (x) and the second $\mu_A(x)$, the degree of membership:

$A = \{(x, \mu_A(x)) \mid x \in X\}$.

Where, μ_A takes value in the interval [0, 1].

The fuzzy subset A on the universe X is defined by membership function, μ_A from X to the real interval [0,1], which associates a number $\mu_A(x) \in [0,1]$ to each element x of universe X. For example, the equation $\mu_A(x) = 0.7$ means element x belongs to the set A with degree 0.7.

4.1.2 Linguistic variable

We can use fuzzy sets to represent linguistic variables. Linguistic variables represent the process states and control variables in a fuzzy controller. Their values are defined in linguistic terms and they can be words or sentences in a natural or artificial language. For example, for the linguistic variable: temperature, we can define a set of terms:

T(temperature) = { negative big, negative medium, negative small, close to zero, positive small, positive medium, positive big }

4.1.3 Membership function

Every fuzzy set can be represented graphically by its membership function. Membership values are discrete values defined in [0, 1]. If the referential set is infinite set, we can represent these values as a continuous membership function. In general, the shape of membership function depends on the application and can be monotonic, triangular, trapezoidal or bell-shaped as shown in Fig. 4.1.

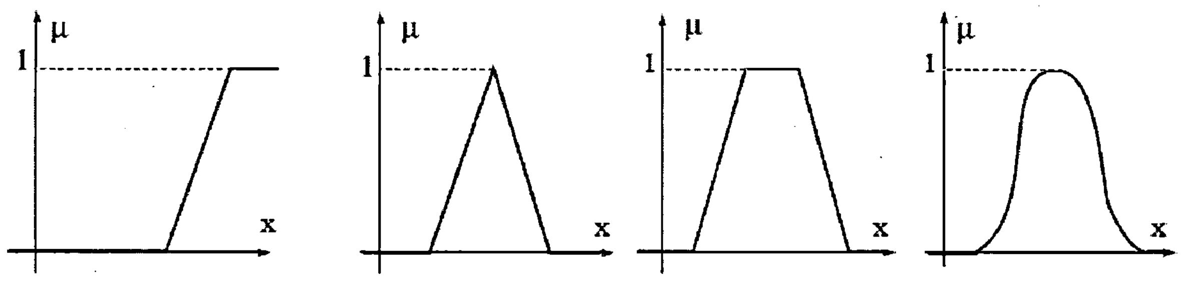

Fig. 4.1 Different shapes of membership functions: monotonic, triangular, trapezoidal and bell-shaped

4.2 Fuzzy Logic Mechanism

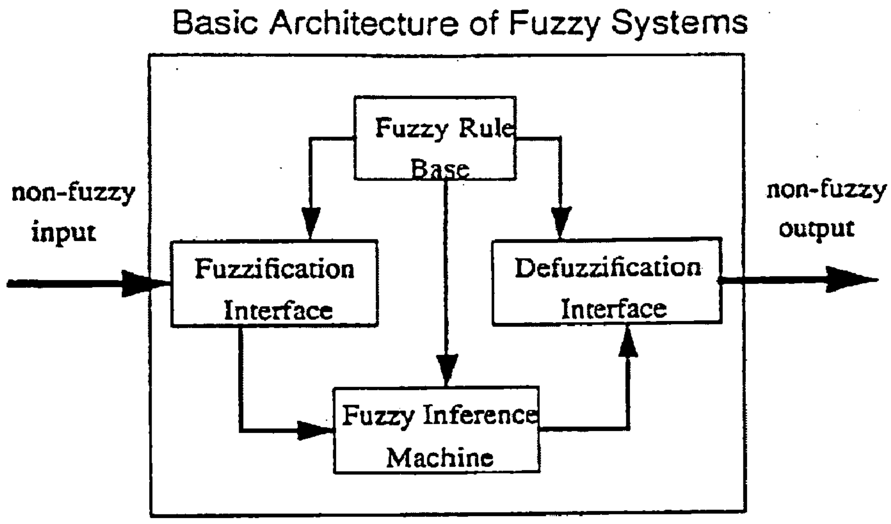

Fig. 4.2 Architecture of Fuzzy Systems

The basic configuration of a fuzzy system is shown in Fig. 4.2. It comprises of four principal components: Fuzzification interface, fuzzy rule base, fuzzy inference machine, and defuzzification interface.

4.2.1 Fuzzification Interface

The fuzzification interface is a mapping from the observed input universe of discourse $U \in R$ to the fuzzy sets defined in U. There are two factors which determine a fuzzification interface: (1) the number of fuzzy sets defined in the input universe of discourse; and, (2) the specific membership functions for these fuzzy sets. If these two factors are specified, we obtain a fuzzification interface; hence, we can view these two factors as design parameters of a fuzzification interface.

4.2.2 Fuzzy Rule Base

The fuzzy rule base is a set of linguistic statements in the form of "IF a set of conditions are satisfied, THEN a set of consequences are inferred", where the conditions and the consequences are associated with fuzzy concepts (i.e., linguistic terms). For example, in the case of an n-input-single-output fuzzy system, the fuzzy rule base may consist of the following rules:

R_j : IF x_1 is A_1^j and x_2 is A_2^j and .... and x_n is A_n^j, THEN z is B^j

Where, x_i (i = 1, 2, ..., n) are the inputs to the fuzzy system, z is the output of fuzzy system, A_i^j and B^j (j = 1,2, ..., K) are linguistic terms and K is the number of fuzzy rules in the fuzzy rule base. These fuzzy IF-THEN rules provide a natural form in which human experts represent their knowledge. By relating each linguistic term in the fuzzy rules with a membership function, we specify the meaning of the fuzzy rules in a determined fuzzy sense. This type of fuzzy rule is known as Mamdani FIS. Another form of fuzzy if-then rule, proposed by Takagi and Sugeno [13], has fuzzy sets involved only in the premise part. The consequent part is described by a non fuzzy equation of the input variable. There are many different kinds of fuzzy rules; see [37] for a complete discussion. Here, we consider only fuzzy rules in the form of Mamdani FIS. The design parameters of a fuzzy rule base are: (1) K, the number of fuzzy rules in the fuzzy rule base; and, (2) the specific statement of each fuzzy rule.

4.2.3 Fuzzy inference process

The fuzzy inference mechanism is decision making logic which employs fuzzy rules from the rule base to determine fuzzy outputs of a fuzzy system corresponding to the fuzzified inputs. It is the fuzzy inference machine that simulates a human decision making procedure based on fuzzy concepts and linguistic statements. There are many different kinds of fuzzy logic which may be used in a fuzzy inference machine; see [37] for a comprehensive review. The design parameter of a fuzzy inference machine is: which specific fuzzy logic is used.

4.2.4 Defuzzification

The defuzzification interface defuzzifies the fuzzy output of a fuzzy system to generate a non-fuzzy output. There are several existing defuzzification methods, namely: Centroid (i.e, center of area), bisector of area, and mean of maximum (see [37] for details). The design parameters of a defuzzification interface are: (1) number of fuzzy sets defined in the output universe of discourse ; (2) specific membership functions of these fuzzy sets; and, (3) which defuzzification method is used.

4.3 Effect of different design parameters

The number of fuzzy sets defined in the input and output universes of discourse and the number of fuzzy rules in the fuzzy rule base heavily influence the complexity of a fuzzy system, where complexity includes time complexity, i.e., the computational time requirements of the fuzzy system, and space complexity, i.e., the storage requirements of the fuzzy system. These parameters can be viewed as structure parameters of a fuzzy system. In general, the larger these parameters are, the more complex is the fuzzy system, and the higher is the expected performance of the fuzzy system. Hence, there is always a trade-off between complexity and accuracy in the choice of these parameters; and, their choice is usually quite subjective.

The membership functions of the fuzzy sets heavily influence the "smoothness" of the input-output surface determined by the fuzzy system. In general, the "sharper" the membership functions are, the less smooth is the input-output surface. The choice of membership functions is also quite subjective.

The linguistic statements of the fuzzy rules are the heart of a fuzzy system in the sense that it is these linguistic statements that contain most of the information concerning the fuzzy system design; all other design parameters assist in the effective representation and use of the information. The fuzzy rules usually come from two sources: human experts, and training data. A general method to generate fuzzy rules from numerical data was proposed in [38][39].

The decision making logic used by the fuzzy inference machine is very important, and may be the most flexible component in the fuzzy system. If we compare a fuzzy system with a human controller, then the Fuzzification interface corresponds to our sensory organs (e.g., eye, ear, etc.), the defuzzification interface corresponds to our action organs (e.g., arms, feet, etc.), the fuzzy rule base corresponds to our memory, and the fuzzy inference machine corresponds to our thought process.

4.4 Fuzzy controller for Ball Beam system

In general, the structure of an FLC can be of multi input and multi-output (MIMO). However, double-input and single-output (DISO) fuzzy logic controllers are more preferable for practical considerations. Specifically, a fuzzy PD controller using error and change of error as inputs, if properly designed, can infer a suitable control output resulting in good transient responses for a general nonlinear system. When comparing with conventional PD controllers, fuzzy PD controllers are usually found to be capable of improving tracking performance remarkably. By defining error and change in error as fuzzy inputs, the proposed fuzzy PD controller will be designed for second-order nonlinear system.

The control object is to design a controller u such that the output y will approximately track a desired signal y_d. The FLC being applied to Ball Beam system is shown in Fig. 4.3. If $y_d(t)$ is the desired trajectory and $y(t)$ is actual trajectory, then, the error signal is given by,

$$e(t) = y(t) - y_d(t)$$

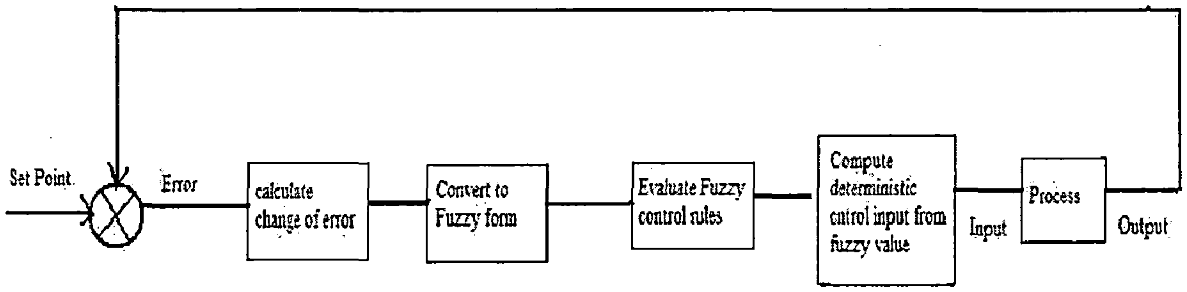
Set Point
Error
calculate change of error
Convert to Fuzzy form
Evaluate Fuzzy control rules
Compute deterministic cntrol input from fuzzy value
Input
Process
Output

Fig. 4.3 Closed loop control system using Fuzzy rules

Membership functions chosen for two inputs are shown in Fig 4.4. Both inputs have three linguistic terms namely: Positive (P), Zero (Z) and Negative (N), whereas control output shown in Fig. 4.5 is described by five linguistic terms namely: Negative big (NB), Negative small (NS), Zero (Z), Positive small (PS) and Positive big (PB). Triangular membership function has been used for both inputs and output.

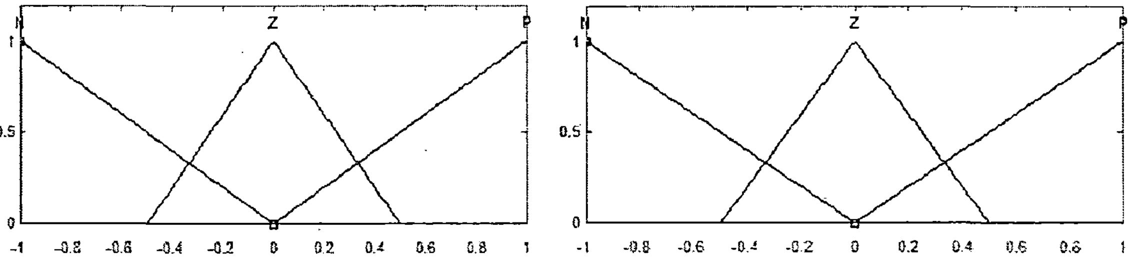

Fig. 4.4 Membership function for error and change in error

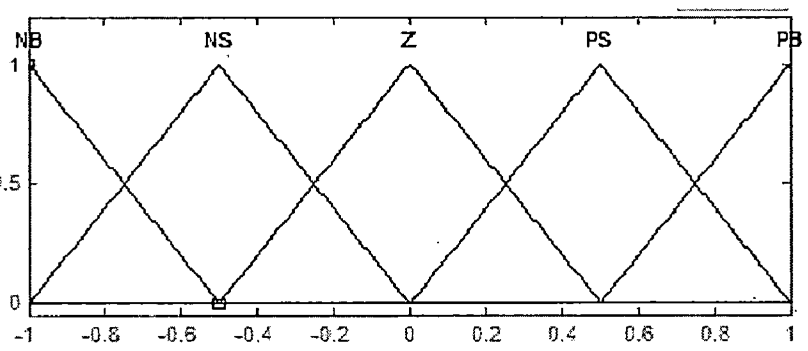

Fig. 4.5 Membership function for control output

The fuzzy rule for 2 input FIS with each inputs described by three membership functions or linguistic terms have 3^2 i.e., 9 rules with error and change in error as premise variables and control output as consequent variable. The fuzzy rule base looks like shown below:
Rule Base

1. If error is N and Change in error is N, then control is PB
2. If error is N and Change in error is Z, then control is PS
3. If error is N and Change in error is P, then control is Z
4. If error is Z and Change in error is N, then control is PS
5. If error is Z and Change in error is Z, then control is Z
6. If error is Z and Change in error is P, then control is NS
7. If error is P and Change in error is N, then control is Z
8. If error is P and Change in error is Z, then control is NS
9. If error is P and Change in error is P, then control is NB

CHAPTER 5

Supervisory Control

5.1 Introduction

For complex practical systems, the single loop control systems may not effectively achieve the control objectives. In order to improve the tracking performance of the control system, we consider two-level control structures where first level is constructed from fuzzy PD controller and the second level is Supervisory controller [40]. FLC was earlier designed for ball beam system which has the advantages of simplicity and smooth control with inexact information but an often remarked disadvantage of the methods based on the fuzzy logic is the lack of appropriate tools for analysing the controller performance, such as stability, optimality, robustness, etc. The main advantage is the possibility to implement a human experience, intuition and heuristics into the controller. Among several approaches to solve stability problem with FLCs, one approach is to design FLC first without any stability consideration. Then, append another controller with FLC to take care of the stability requirement. Here, the supervisor has been derived based on variable structure control theory. This approach provides much flexibility in designing the fuzzy controller and hence the resulting fuzzy control system is expected to show high performance.

5.2 Two Level Control

5.2.1 Overview of supervisory Control

A supervisory controller is a controller which operates only when some undesirable phenomena occur, e.g., when the state hits the boundary of constraint set. In this note, we develop a supervisory controller for nonlinear fuzzy control systems. The supervisory controller works in the following way: if the fuzzy control system (without the supervisory controller) is stable in the sense that the state is inside the constraint set, the supervisory control is idle; if the state hits the boundary of the constraint set, the supervisory controller begins operation to force the state back to the constraint set. The fuzzy control system equipped with this supervisory controller is globally stable in the sense that the state is guaranteed to be within the constraint set specified by the system designer. The two-level control structure is shown in Fig. 5.1. The first level FLC controller and second level supervisor will be switched by $M = \max \{ |s_1|, |s_2| \}$, where s_1 and s_2 are two states of the system.

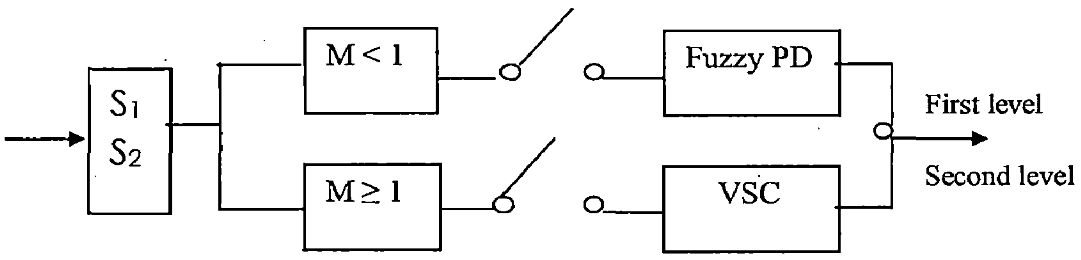

Fig. 5.1 Two level control scheme

5.2.2 Need for Supervisory control

Despite successful application of Fuzzy logic controllers to many practical problems, there are many issues remaining to be further addressed such as stability, controllability and observability.

Conceptually, there are at least two different approaches to guarantee the stability of a fuzzy control system. The first approach is to specify the structure and parameters of the fuzzy controller such that the closed-loop system with this fuzzy controller is stable. An example of this approach is [41]. This approach often requires the fuzzy controller to satisfy some strong sufficient conditions which greatly limit the design flexibility and, therefore, the performance of the fuzzy controller. In the second approach, the fuzzy controller is designed first without any stability consideration, then another controller is appended to the fuzzy controller to take care of the stability requirement. This approach has been utilized here to control Ball beam system. Because there is much flexibility in designing the fuzzy controller in this second approach, the resulting fuzzy control system is expected to show high performance.

The key is how to design the appended controller to guarantee stability. Because we want the fuzzy controller to perform the main control action, the appended controller would be better a safeguard rather than a main controller. Therefore, we choose the appended controller to work in the following supervisory fashion: if the fuzzy controller works well, the appended controller is idle; if the pure fuzzy control system tends to be unstable, the appended controller begins operation to guarantee stability. Thus, we call the appended controller a supervisory controller. In this note, we say a system is stable if its state variables are uniformly bounded.

5.3 Design principle of Supervisory controller

Designing a supervisory controller for a nonlinear fuzzy controller system where the fuzzy controller already exists has been discussed in and proposes modifications of the supervisory control which switch to the supervisory mode gradually.
Consider the nonlinear system governed by the differential equation,

$$x^{(n)} = f(x, \dot{x}, \ldots\ldots, x^{(n-1)}) + g(x, \dot{x}, \ldots\ldots, x^{(n-1)})u \tag{5.1}$$

where x ϵ R is the output of the system, u ϵ R is the control, $\underline{x} = (x, \dot{x}, \ldots\ldots, x^{(n-1)})^T$ is the state vector which is assumed to be measurable or computable, and f and g are unknown nonlinear functions. We assume that g > 0. From nonlinear control theory we know that this system is in normal form and many general nonlinear systems can be transformed into this form. The main restriction is that the control u is required to appear linearly in the equation.

Now suppose that we have already designed a fuzzy controller,

$$u = u_f(\underline{x}) \tag{5.2}$$

This can be done by synthesizing fuzzy control rules from human experts and/or by trial and error using designing tools. Our task is to guarantee the stability of the closed-loop system, and, at the same time, without changing the existing design of the fuzzy controller. More specifically, we are required to design a controller whose main control action is the fuzzy control u_f and that the closed-loop system with this controller is globally stable in the sense that the state $\underline{x}$ is uniformly bounded, i.e., $|\underline{x}(t)| \leq$ M, for all t > 0, where M is a constant given by the designer.
For this task, we append the fuzzy controller u_f with a supervisory controller u_s which is nonzero only when the state x hits the boundary of the constraint set { $\underline{x}$: $|\underline{x}| \leq$ M }, i.e., the control now is,

$$u = u_f(\underline{x}) \qquad \text{if} \qquad |\underline{x}| \leq M \qquad (5.3)$$

$$= u_s(x) \qquad \text{if} \qquad |\underline{x}| > M$$

Therefore, the main control action is still the fuzzy control u_f. Our task now is to design u_s such that we always have $|\underline{x}| \leq M$, for all $t > 0$.

CHAPTER 6

Neural Network Control

6.1 Introduction

Artificial neural networks (ANNs) are the highly simplified models of biological nervous system and therefore have drawn their motivation from the way of computing performed by human brain. ANN is a highly interconnected network of a large number of processing elements called neurons in an architecture inspired by the human brain. An ANN can be massively parallel and therefore is said to exhibit parallel distributed processing. The power of neural networks is in their ability to learn and to store knowledge. Neural networks purport to simulate in a simplified manner the activities of processes that occur in the human brain. The ability to learn is one of the main advantages that make neural networks so attractive. In control engineering neural networks can be used to control multi input, multi output non-linear systems having delays. The ability of neural networks to control engineering processes without prior knowledge of the system dynamics is very appealing to researchers and engineers in the field. [42] discusses the neural network approach to control systems. Many papers has been published dealing with the application of artificial neural network technique to control Ball beam system.

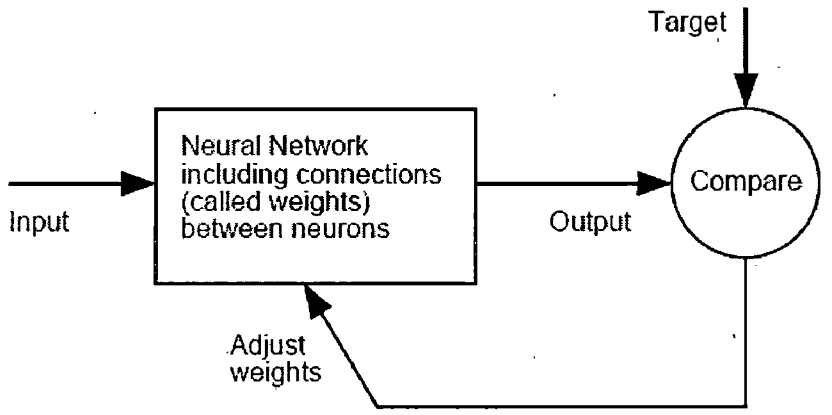
Target
Input
Neural Network including connections (called weights) between neurons
Output
Compare
Adjust weights

Fig. 6.1 Neural Network functioning

A neural network performs a particular function by adjusting the values of the connections (weights) between elements so that a particular input leads to a specific target output. Such a situation is shown in Fig. 6.1 where the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used to train a network in this type of learning called supervised learning.

6.2 Neuron model

An elementary neuron with *R* inputs is shown in Fig. 6.2. Each input is weighted with an appropriate weight. The sum of the weighted inputs and the bias forms the input to the transfer function *f*. Neurons may use any differentiable transfer function *f* to generate their output.

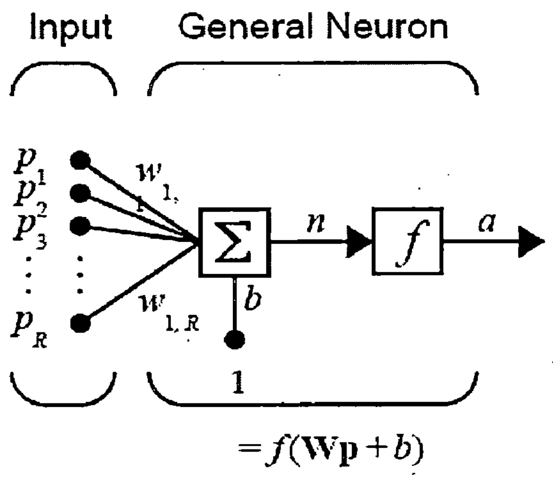

Fig. 6.2 Single Neuron Model

Their sum is simply **Wp**, the dot product of the (single row) matrix **W** and the vector **p**. The neuron has a bias *b*, which is summed with the weighted inputs to form the net input *n*. This sum, *n*, is the argument of the transfer function *f*.

6.3 Neural Network Architecture

There are several classes of ANNs, classified according to their learning mechanism. The three basic classes of networks include Single layer feedforward, Multilayer feedforward and Recurrent networks.

1. Single Layer feedforward network

This type of network comprises only two layers, namely input and output layer. The input neurons receive the input signals and the output neurons give the output signal. The synaptic links carrying the weights connect every input neuron to the output neuron but not vice-versa. The graph is acyclic in nature, hence called feedforward network. Despite the two layers, the network is termed single layer as it is only output layer that performs computation. Input layer only transmits the signals to the output layer.

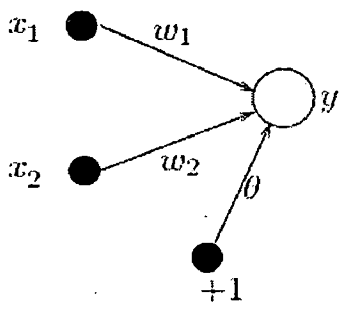

Fig. 6.3 Single layer network with two inputs and one output

2. Multi Layer Feedforward Network

This type of network consists of multiple layers. The architecture besides possessing input and output layer also have one or more intermediately layers called hidden layers. The hidden layer aids in performing useful intermediately computation before directing input to output layer. A multilayer feedforward network with *l* input neurons, m_1 neurons in first hidden layer, m_2 neurons in the second hidden layer and n output neurons is written as *l*-m_1- m_2-*n*.

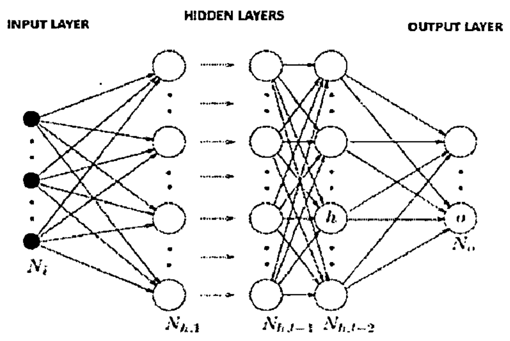

Fig. 6.4 Multilayer feedforward network

3. Recurrent Networks

These networks differ from feedforward network architectures in the sense that there is at least one feedback loop. There could also be neurons with self-feedback links, i.e. the output of a neuron is fed back into itself as input.

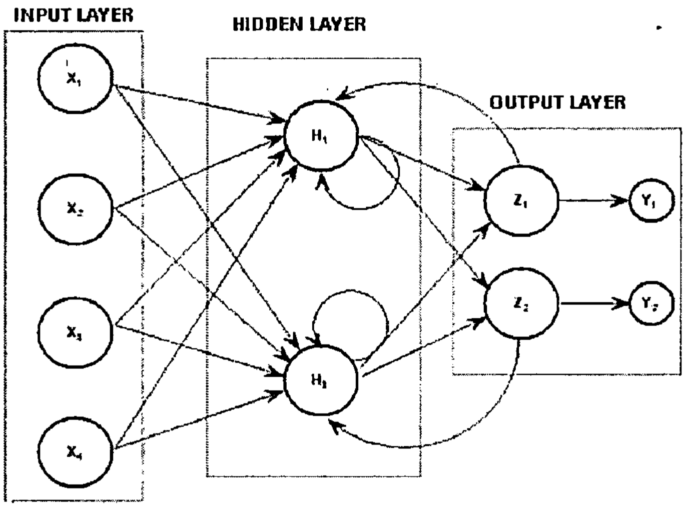

Fig. 6.5 Recurrent Network

6.4 Types of activation functions:

A function F_k is needed which takes the total input $s_k(t)$ and produces a new value of the activation of the unit k. Often, the activation function is a non decreasing function of the total input of the unit:

$$y_k(t+1) = F_k(s_k(t)) = F_k\left(\sum_j w_{jk}(t)y_j(t) + \theta_k(t)\right)$$

Although activation functions are not restricted to nondecreasing functions, generally, some sort of threshold function is used: a hard limiting threshold function (a sgn function), or a linear or semi-linear function, or a smoothly limiting threshold.

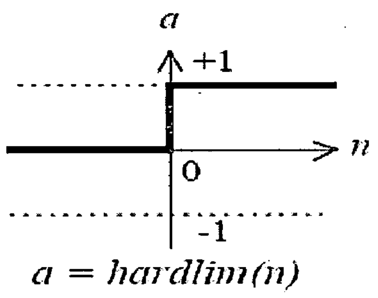

Fig. 6.6 Hard-Limit Transfer Function

The perceptron neuron produces a 1 if the net input into the transfer function is equal to or greater than 0; otherwise it produces a 0. The hard-limit transfer function gives a perceptron the ability to classify input vectors by dividing the input space into two regions. Specifically, outputs will be 0 if the net input *n* is less than 0, or 1 if the net input *n* is 0 or greater.

Differentiable Activation functions

The back propagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem. Since this method requires computation of the gradient of the error function at each iteration step, we must guarantee the continuity and differentiability of the error function. Obviously we have to use a kind of activation function which is differentiable otherwise the composite function produced by interconnected perceptrons will be discontinuous, and therefore the error function too. One of the more popular activation functions for backpropagation networks is the sigmoid, a real function s : $\mathbb{R} \rightarrow (0, 1)$ defined by the express

$$s(x) = \frac{1}{1+e^{-x}}$$

The derivative of the sigmoid with respect to x, is

$$\frac{d}{dx}s(x) = \frac{e^{-x}}{(1+e^{-x})^2} = s(x)(1-s(x))$$

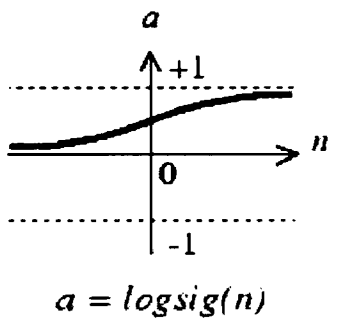

Fig. 6.7 Sigmoidal transfer function

The sigmoid function generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity. Alternatively, multilayer networks may use the tan-sigmoid transfer function as given by,

$$s_t(x) = \frac{2}{(1+e^{-2n})^{-1}}$$

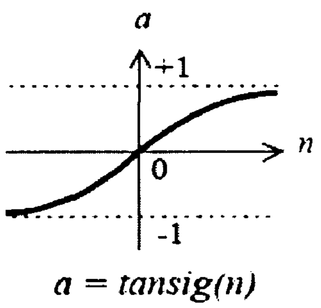

Fig. 6.8 Tan-sigmoid Transfer function

Ocassionally, the linear transfer function purelin is used in backpropagation networks given by,

$$s_p(x) = x$$

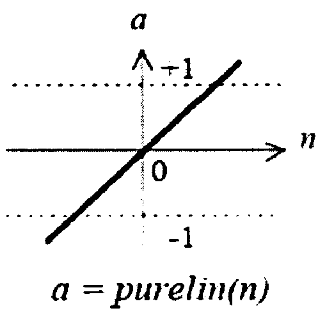

Fig. 6.9 Linear Transfer Function

If the last layer of a multilayer network has sigmoid neurons, then the outputs of the network are limited to a small range. If linear output neurons are used the network outputs can take on any value. Each of the transfer functions above, tan-sigmoid, log-sigmoid and linear function, have a corresponding derivative function.

6.5 General feed-forward network

6.5.1 The learning problem

In general definition, a feed-forward neural network is a computational graph whose nodes are computing units and whose directed edges transmit numerical information from node to node. Each computing unit is capable of evaluating a single primitive function of its input. In fact the network represents a chain of function compositions which transform an input to an output vector (called a pattern). The network is a particular implementation of a composite function from input to output space, which we call the network function. The learning problem consists of finding the optimal combination of weights so that the network function approximates a given function f as closely as possible. However, we are not given the function f explicitly but only implicitly through some examples.

Consider a feed-forward network with n input and m output units. It can consist of any number of hidden units and can exhibit any desired feed-forward connection pattern. We are also given a training set $\{(x_1,t_1),....,(x_p,t_p)\}$ consisting of p ordered pairs of n- and m-dimensional vectors, which are called the input and output patterns. Let the primitive functions at each node of the network be continuous and differentiable. The weights of the edges are real numbers selected at random. When the input pattern xi from the training set is presented to this network, it produces an output o_i different in general from the target t_i. What we want is to make o_i and t_i identical for $i = 1,.....,p$, by using a learning algorithm. More precisely, we want to minimize the error function of the network, defined as,

$$E = \frac{1}{2}\sum_{i=1}^{p}(o_i - t_i)^2 \tag{6.1}$$

After minimizing this function for the training set, new unknown input patterns are presented to the network and we expect it to interpolate. The network must recognize whether a new input vector is similar to learned patterns and produce a similar output.

6.5.2 Paradigms of Learning

Neural Networks learn by examples. They can therefore be trained with known examples of a problem to acquire knowledge about it. Once appropriately trained, the network can be put to effectively utilize in solving unknown or untrained instances of the problem. The process of modifying the weights in the connections between network layers with the objective of achieving the expected output is called training a network. The internal process which takes place when a network is trained is called learning. Generally, there are 3 types of learning as follows :

1. Supervised Learning or Associative Learning:
   It is the process of providing the network with a series of sample inputs and comparing the output with the expected responses. The training continues until the network is able to provide the expected response. The weights may then be adjusted according to a learning algorithm. This process is called supervised learning. Some of the supervised learning algorithm includes Backpropagation, Adaline, madaline, Hebb net, counter propagation net etc.

2. Unsupervised Learning or Self organized Learning:
   If for the training input vectors, the target output is not known, the training method adopted is called supervised training. The net may modify the weight so that most similar input vector is assigned to the same output unit. Unsupervised networks are far more complex and difficult to implement. It involves looping connections back into feedback layers and iterating through the process until some sort of stable recall is achieved. These are also called self learning or self organizing networks because of their ability to carry out self learning. Self organizing feature maps, Adaptive resonance theory are examples of this category.

3. Reinforcement Training
   In this method, a teacher is also assumed to be present, but the right answer is not presented to the network. Instead, an indication of whether the output answer is right or wrong is presented. The network then uses this information to improve its performance. This type of learning is applied when knowledge required for supervised learning is not available.

6.5.3 Modifying patterns of connectivity

All learning paradigms discussed above result in an adjustment of the weights of the connections between units, according to some modification rule. Virtually all learning rules for models of this type can be considered as a variant of the Hebbian learning rule suggested by Hebb in his classic book Organization of Behaviour (1949) (Hebb, 1949). The basic idea is that if two units j and k are active simultaneously, their interconnection must be strengthened. If j receives input from k, the simplest version of Hebbian learning prescribes to modify the weight w_{jk} with

$$\Delta w_{jk} = \gamma y_j y_k \tag{6.2}$$

where γ is a positive constant of proportionality representing the learning rate. Another common rule uses not the actual activation of unit k but the difference between the actual and desired activation for adjusting the weights:

$$\Delta w_{jk} = \gamma y_j (d_k - y_k) \tag{6.3}$$

in which d_k is the desired activation provided by a teacher. This is often called the Widrow-Höff rule or delta rule.

6.6 Networks with linear activation functions: The delta rule

For a single layer network with an output unit with a linear activation function the output is simply given by,

$$y = \sum_j w_j x_j + \theta \tag{6.4}$$

Such a simple network is able to represent a linear relationship between the value of the output unit and the value of the input unit. Suppose we want to train the network such that a hyperplane is fitted as well as possible to a set of training samples consisting of input values x^p and desired (or target) output values d^p. For every given input sample, the output of the network differs from the target value d^p by $(d^p - y^p)$, where y^p is the actual output for this pattern. The delta-rule now uses a cost or error-function based on these differences to adjust the weights.

The error function, as indicated by the name least mean square, is the summed squared error. That is, the total error E is defined to be

$$E = \sum_p E^p = \frac{1}{2}\sum_p (d^p - y^p)^2 \tag{6.5}$$

where the index p ranges over the set of input patterns and E^p represents the error on pattern p. The LMS procedure finds the values of all the weights that minimise the error function by a method called gradient descent. The idea is to make a change in the weight proportional to the negative of the derivative of the error as measured on the current pattern with respect to each weight:

$$\Delta_p w_j = -\gamma \frac{\partial E^p}{\partial w_j} \tag{6.6}$$

where γ is a constant of proportionality. The derivative is

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial y^p}\frac{\partial y^p}{\partial w_j}$$

Because of the linear units,

$$\frac{\partial y^p}{\partial w_j} = x_j$$

And

$$\frac{\partial E^p}{\partial y^p} = -(d^p - y^p)$$

Such that

$$\Delta_p w_j = \gamma \delta^p x_j \tag{6.7}$$

where $\delta^p = d^p - y^p$ is the difference between the target output and the actual output for pattern p.

The delta rule modifies weight appropriately for target and actual outputs of either polarity and for both continuous and binary input and output units. These characteristics have opened up a wealth of new applications.

Minsky and Papert (Minsky & Papert, 1969) showed in 1969 that a two layer feed-forward network can overcome many restrictions and are capable of computing a wider range of functions than networks with a single layer of computing units, but did not present a solution to the problem of how to adjust the weights from input to hidden units. The computational effort needed for finding the correct combination of weights increases substantially when more parameters and more complicated topologies are considered. An answer to this question was presented by Rumelhart, Hinton and Williams in 1986 [19], and similar solutions appeared to have been published earlier (Werbos, 1974; Parker, 1985; Cun, 1985). The central idea behind this solution is that the errors for the units of the hidden layer are determined by back-propagating the errors of the units of the output layer. For this reason the method is often called the back-propagation learning rule. Back-propagation can also be considered as a generalisation of the delta rule for non-linear activation functions and multilayer networks.

Although back-propagation can be applied to networks with any number of layers, just as for networks with binary units it has been shown (Hornik, Stinchcombe, & White, 1989; Funahashi, 1989; Cybenko, 1989; Hartman, Keeler, & Kowalski, 1990) that only one layer of hidden units success to approximate any function with finitely many discontinuities to arbitrary precision, provided the activation functions of the hidden units are non-linear. In most applications a feed-forward network with a single layer of hidden units is used with a sigmoid activation function for the units.

6.7 The Generalised delta or Back-Propagation rule

Back propagation is a supervised learning method, and is a generalization of the delta rule. It requires a teacher that knows, or can calculate, the desired output for any input in the training set. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backward propagation of errors". Backpropagation networks are necessarily multilayer perceptrons. In order for the hidden layer to serve any useful function, multilayer networks must have non-linear activation functions for the multiple layers: a multilayer network using only linear activation functions is equivalent to some single layer, linear network. The detailed explanation of Backpropagation networks can be found in [43]. Since we are now using units with nonlinear activation functions, we have to generalise the delta rule which was presented for linear functions to the set of non-linear activation functions.

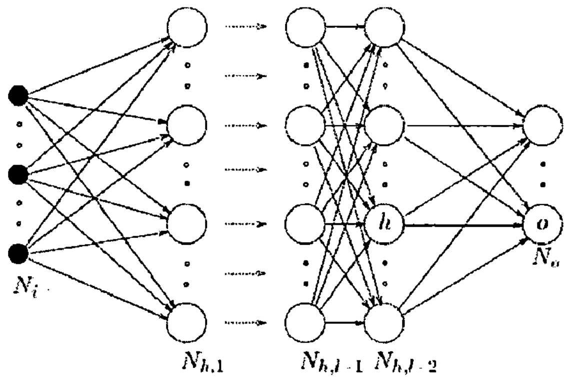

Fig. 6.10 Multilayer feedforward network

The activation is a differentiable function of the total input, given

$$y_k^p = F(s_k^p) \tag{6.8}$$

in which

$$s_k^p = \sum_j w_{jk} y_j^p + \theta_k \tag{6.9}$$

To get the correct generalisation of the delta rule, we must set

$$\Delta_p w_{jk} = -\gamma \frac{\partial E^p}{\partial w_{jk}} \tag{6.10}$$

The error measure E^p is defined as the total quadratic error for pattern p at the output units:

$$E^p = \frac{1}{2} \sum_{o=1}^{N_o} (d_o^p - y_o^p)^2 \tag{6.11}$$

where d_o^p is the desired output for unit o when pattern p is clamped. We further set E $=\sum_p E^p$ as the summed squared error. We can write

$$\frac{\partial E^p}{\partial w_{jk}} = \frac{\partial E^p}{\partial s_k^p} \frac{\partial s_k^p}{\partial w_{jk}}$$

By equation (6.9) we see that the second factor in above eqn. is

$$\frac{\partial s_k^p}{\partial w_{jk}} = y_j^p \tag{6.12}$$

When we define

$$\delta_k^p = -\frac{\partial E^p}{\partial s_k^p} \tag{6.13}$$

we will get an update rule which is equivalent to the delta rule, resulting in a gradient descent on the error surface if we make the weight changes according to:

$$\Delta_p w_{jk} = \gamma \delta_k^p y_j^p \tag{6.14}$$

The trick is to figure out what δ_k^p should be for each unit k in the network. The interesting result, which we now derive, is that there is a simple recursive computation of these δ's which can be implemented by propagating error signals backward through the network.

To compute δ_k^p we apply the chain rule to write this partial derivative as the product of two factors, one factor reflecting the change in error as a function of the output of the unit and one reflecting the change in the output as a function of changes in the input. Thus, we have

$$\delta_k^p = -\frac{\partial E^p}{\partial s_k^p} = -\frac{\partial E^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial s_k^p} \tag{6.15}$$

Let us compute the second factor. By equation (6.8) we see that

$$\frac{\partial y_k^p}{\partial s_k^p} = F'(s_k^p) \tag{6.16}$$

which is simply the derivative of the squashing function F for the kth unit, evaluated at the net input s_k^p to that unit. To compute the first factor of equation (6.15), we consider two cases. First, assume that unit k is an output unit k = o of the network. In this case, it follows from the definition of E^p that

$$\frac{\partial E^p}{\partial y_o^p} = -(d_o^p - y_o^p) \tag{6.17}$$

which is the same result as we obtained with the standard delta rule. Substituting this and equation (6.16) in equation (6.15), we get,

$$\delta_o^p = (d_o^p - y_o^p) F_o'(s_o^p) \tag{6.18}$$

for any output unit o. Secondly, if k is not an output unit but a hidden unit k = h, we do not readily know the contribution of the unit to the output error of the network. However, the error measure can be written as a function of the net inputs from hidden to output layer and we use the chain rule to write

$$\frac{\partial E^p}{\partial y_h^p} = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p} \frac{\partial s_o^p}{\partial y_h^p} = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p} \frac{\partial}{\partial y_h^p} \sum_{j=1}^{N_h} w_{ko} y_j^p = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p} w_{ho} = -\sum_{o=1}^{N_o} \delta_o^p w_{ho} \qquad (6.19)$$

Substituting this in equation (6.15) yields

$$\delta_h^p = F'(s_h^p) \sum_{o=1}^{N_o} \delta_o^p w_{ho} \qquad (6.20)$$

Equations of (6.18) and (6.20) give a recursive procedure for computing the δ's for all units in the network, which are then used to compute the weight changes. This procedure constitutes the generalised delta rule for a feed-forward network of non-linear units.
The application of the generalised delta rule thus involves two phases: During the first phase, the input x is presented and propagated forward through the network to compute the output values y_o for each output unit. This output is compared with its desired value d_o, resulting in an error signal E_o for each output unit. We strive to change the connections in such a way that, next time around, the error e_o will be zero for this particular pattern. The second phase involves a backward pass through the network during which the error signal is passed to each unit in the network and appropriate weight changes are calculated.

6.8 Back Propagation network parameter selection:

For the efficient operation of BP network, it is necessary to select appropriately the parameters used for training.

6.5.1 Initial weights

It will influence whether the net reaches a global minima of the error and if so how rapidly it converges. If the initial weight is too large, the initial signals to each hidden or output unit will fall in saturation region where the derivative of sigmoid has very small value. If weights are too small, the net input to hidden or output unit will approach zero, which then causes extremely slow learning. So, initial weights and biases are set to random numbers between -0.5 to 0.5 or -1 to 1.

6.5.2 Learning rate

A high learning rate leads to rapid learning but the weights may oscillate, while a lower learning rate leads to slower learning. One method is to start with a high learning rate and steadily decrease it. Changes in weight vector must be small in order to reduce oscillations or any divergence.

6.9 Deficiencies of back-propagation

Despite the apparent success of the back-propagation learning algorithm, there are some aspects which make the algorithm not guaranteed to be universally useful.

Slow Training Process

Most troublesome is the long training process. This can be a result of a non-optimum learning rate and momentum. A lot of advanced algorithms based on back propagation learning have some optimised method to adapt this learning rate.

Network paralysis

As the network trains, the weights can be adjusted to very large values. The total input of a hidden unit or output unit can therefore reach very high (either positive or negative) values, and because of the sigmoid activation function the unit will have an activation very close to zero or very close to one. The weight adjustments which are proportional to $y_k^p(1-y_k^p)$ will be close to zero and the training process can come to a virtual standstill.

Local minima

The error surface of a complex network is full of hills and valleys. Because of the gradient descent, the network can get trapped in a local minimum when there is a much deeper minimum nearby. Probabilistic methods can help to avoid this trap, but they tend to be slow. Another suggested possibility is to increase the number of hidden units. Although this will work because of the higher dimensionality of the error space, and the chance to get trapped is smaller, it appears that there is some upper limit of the number of hidden units which, when exceeded, again results in the system being trapped in local minima. Figure 6.11 shows an example of a local minimum with a higher error level than in other regions. There is a valley in the error function and if gradient descent is started there the algorithm will not converge to the global minimum.

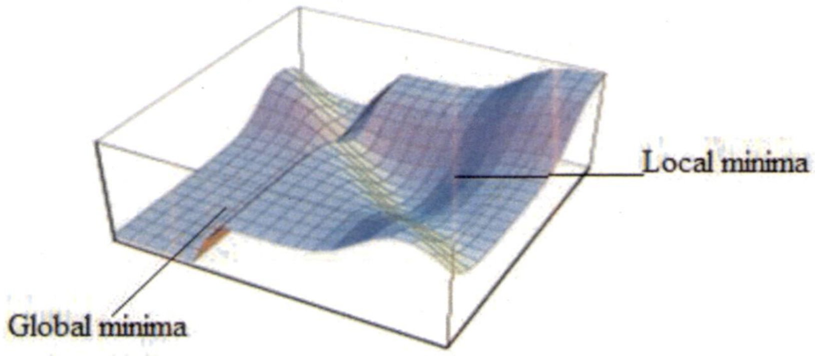

Fig. 6.11 Local and global minima in the error surface

6.10 Neural Network controller for Ball Beam system

Ball Beam system is a nonlinear system with delayed feedback and hence needs some special techniques and complicated mathematical derivation in conventional control methods. Neural network is a non-classical means to solve the ball beam control problem. The special features of ANN over conventional approach include model free approach, learning and generalizing capability and high noise tolerance.

A Neural Network controller has been designed for ball beam system using MATLAB Neural Network toolbox [44]. There are various functions available such as functions to create a network with adjustable number of inputs and outputs nodes as well as number of hidden layers and number of neurons in each layer and there are functions also to train and simulate the networks. The network designed has two inputs error and change in error and one control output and utilizes Backpropagation algorithm to train the network.

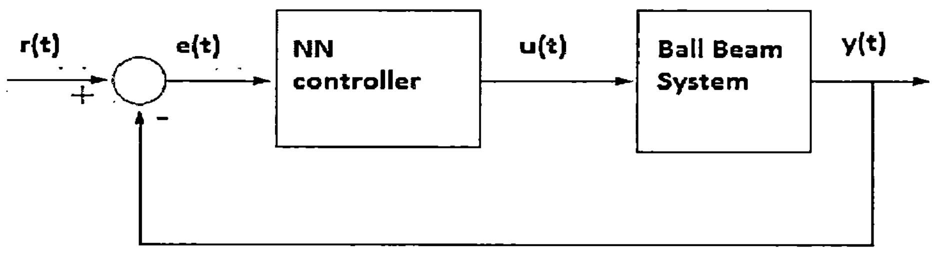

Fig. 6.12 Closed loop system with Neural Network controller

CHAPTER 7

Neuro-Fuzzy Hybrid Controller

7.1 Introduction

The fuzzy model generated after training the adaptive neural network is called Adaptive Neuro-Fuzzy Inference system (ANFIS).This is one of the most researched forms of hybrid systems and has resulted in a stupendous quantity of publication and research results. Neural Network and Fuzzy logic represent two distinct methodologies to deal with uncertainty. Each of them has its own merits and demerits. Neural Networks are highly simplified model of human nervous system which mimics our ability to adapt to circumstances and learn from past experience. NNs can model complex nonlinear relationships and are appropriately suited for classification phenomenon into predetermined classes. But, the precision of outputs is quite often limited and does not ensure zero error but only minimization of least error. Besides, the training data has to be chosen carefully so as to cover the entire range over which the different variables are expected to vary. In addition, the training time required for NN can be substantially large.

On the other hand, Fuzzy logic system addresses the imprecision or vagueness in input-output description of system by defining them using fuzzy sets and allows for a greater flexibility in formulating system description at the appropriate level of detail. But, the disadvantages in designing FLC include: 1) No standard methods exist for transforming human knowledge or experience into the rule base and database of a fuzzy inference system. 2) There is a need for effective methods for tuning the membership functions (MF's) so as to minimize the output error measure or maximize performance index.

Various attempts have been successfully made to synergize these different techniques in whole or in part to solve problems for which these technologies could not find solutions individually. The objective of hybridization has been to overcome the weaknesses in one technology during its application with the strength of other by appropriately integrating them.

Adaptive Neuro Fuzzy Inference Systems are a form of Neuro-Fuzzy hybrid. These are a class of adaptive networks that are functionally equivalent to fuzzy inference systems. ANFIS represent Takagi Sugeno fuzzy model and uses a hybrid learning algorithm which is the combination of Gradient descent and least square estimates method.

7.2 ANFIS Architecture

An adaptive network is a multilayer feedforward network in which each node performs a particular function (node function) on incoming signals as well as a set of parameters pertaining to this node. The formulas for the node functions may vary from node to node, and the choice of each node function depends on the overall input-output function which the adaptive network is required to carry out. Note that the links in an adaptive network only indicate the flow direction of signals between nodes; no weights are associated with the links. To reflect different adaptive capabilities, we use both circle and square nodes in an adaptive network. A square node (adaptive node) has parameters while a circle node (fixed node) has none. The parameter set of an adaptive network is the union of the parameter sets of each adaptive node. In order to achieve a desired input-output mapping, these parameters are updated according to given training data and a hybrid learning algorithm. Functionally, there are almost no constraints on the node functions of an adaptive network except piecewise differentiability. Structurally, the only limitation of network configuration is that it should be of feed forward type.

If the FIS under consideration has two inputs and one output, then the fuzzy rule base algorithm in Takagi Sugeno form [54] can be represented as :

Let, x and y are two inputs with A_1, A_2,.........A_n be the linguistic terms used to describe membership function for x. Similarly, B_1, B_2,............B_n be the linguistic term for y, then,

Rule 1: If x is A_1 and y is B_1 , then $f_1 = p_1 x + q_1 y + r_1$

Rule 2: If x is A_2 and y is B_2 , then $f_2 = p_2 x + q_2 y + r_2$

...

...

...

...

Rule n: If x is A_n and y is B_n , then $f_n = p_n x + q_n y + r_n$

The ANFIS architecture with above rules is shown in Fig. 7.1. The architecture shown consists of 5 layers, which are defined as:

Layer 1: Every ith node in this layer is an adaptive node with its output defined as,

$$O_{1,i} = \mu_{A_i}(x) \quad \text{, for nodes with input x}$$

$$O_{1,i} = \mu_{B_i}(y) \quad \text{, for nodes with input y}$$

The membership function for A and B can be any appropriate parameterized membership function, where the adjustable parameters, in this layer are called premise parameter.

Layer 2: Every node in this layer is fixed node shown as Π, which multiplies incoming signals and outputs the product or T norm.

e.g. $$O_{2,i} = \mu_{A_i}(x) \times \mu_{B_i}(y)$$

Each node output represents the firing strength of a rule.

Layer 3: Every node in this layer is fixed node shown as N, called normalizing function. The ith node calculates the ratio of ith rule's firing strength to the sum of all rule's firing strength,

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2 + \dots\dots + w_n}$$

Layer 4: Every ith node in this layer is an adaptive node with function:

$$O_{4,i} = \overline{w_i} . f_i = \overline{w_i}(p_i x + q_i y + r_i)$$

Where {p, q, r} is parameter set called consequent parameter.

Layer 5: The node in this layer is function Σ which computes sum of all incoming signals

$$O_{5,i} = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

This results in an adaptive network which modifies premise and consequent parameter for minimum error and having same function as Sugeno fuzzy model.

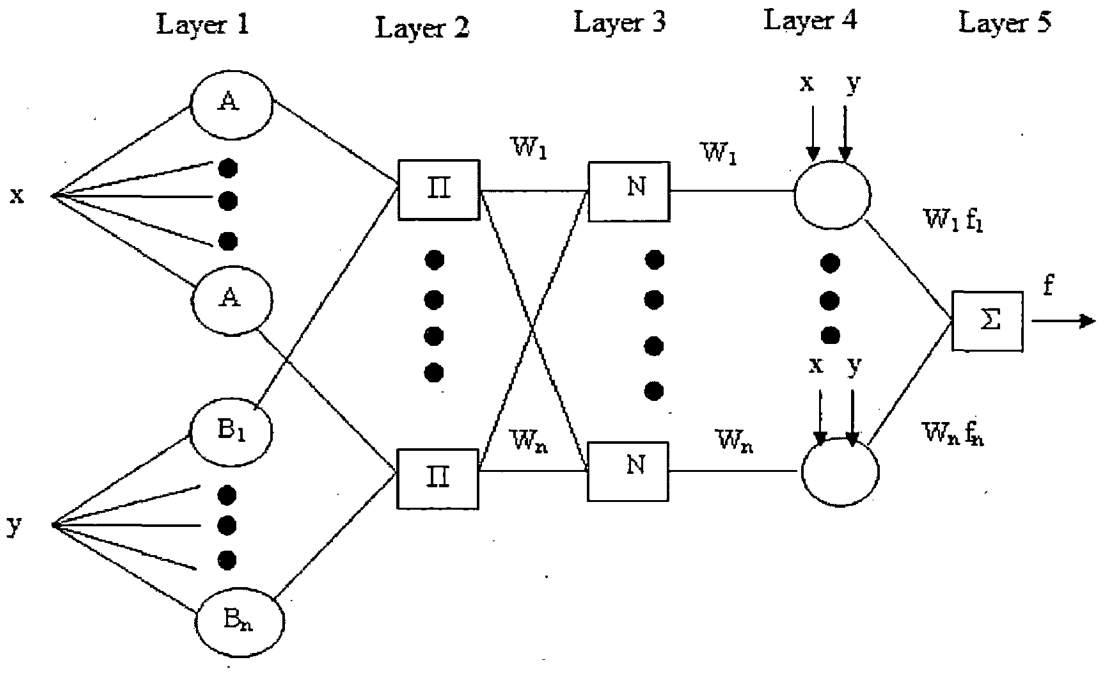

Fig. 7.1 Architecture of 2- input 1-output ANFIS

7.3 Hybrid Learning Algorithm

The ANFIS can be trained by a hybrid learning algorithm presented by Jang [108]. From the proposed ANFIS architecture, it is observed that given the values of premise parameters, the overall output can be expressed as a linear combinations of the consequent parameters. More precisely, the output *f* in Fig. 7.1 can be rewritten as,

$$\begin{aligned} f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\ &= \bar{w}_1 f_1 + \bar{w}_2 f_2 \\ &= (\bar{w}_1 x) p_1 + (\bar{w}_1 y) q_1 + (\bar{w}_1) r_1 + (\bar{w}_2 x) p_2 + (\bar{w}_2 y) q_2 + (\bar{w}_2) r_2 \end{aligned}$$

which is linear in the consequent parameters (p_1, q_1, r_1, p_2, q_2 and r_2) . As a result, we have

S = set of total parameters

S_1 = set of premise parameters

S_2 = set of consequent parameter

More specifically, in the forward pass of the hybrid learning algorithm, functional signals go forward till layer 4 and the consequent parameters are identified by the least squares estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent.

The consequent parameters thus identified are optimal (in the consequent parameter space) under the condition that the premise parameters are fixed. Accordingly the hybrid approach is much faster than the strict gradient descent. However, it should be noted that the computation complexity of the least squares estimate is higher than that of the gradient descent.

7.4 ANFIS controller for Ball Beam system

ANFIS solves the tuning problem involved with fuzzy controllers by tuning the fuzzy premise and consequent parameters to minimize error with the help of neural network learning capability. It serves as an advanced technique to control complex non-linear systems.

While manually designing Fuzzy controller for Ball beam system, it was difficult to decide the universe of discourse for input and output fuzzy variables. This problem is solved by ANFIS by utilizing training data obtained from PD controller. Other tuning parameters such as overlap among membership functions and membership functions parameters are also tuned by ANFIS.

Fig. 7.2 shows a 2-input ANFIS with nine rules. Three membership functions are associated with each input, so the input space is partitioned into nine fuzzy subspaces, each of which is governed by fuzzy if-then rules. The premise part of a rule delineates a fuzzy subspace, while the consequent part specifies the output within this fuzzy subspace.

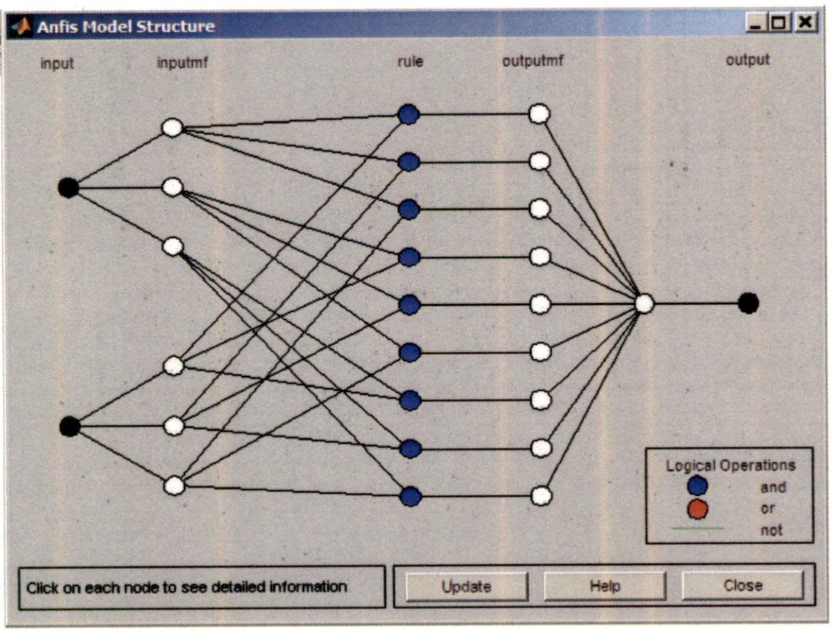

Fig. 7.2 ANFIS Model

CHAPTER 8

Results and Discussion

8.1 Simulation Results

8.1.1 Sliding Mode Controller

The Ball position response of SMC with simplified model given in eqn. (2.1), obtained with the help of MATLAB/Simulink simulation is shown in Fig. 8.1. The system parameters used in simulation are given in Appendix. The figure shows that it takes around 2 sec to settle the ball for given initial condition of 0.1 m. The system does not show any overshoot and undershoot as shown in figure for well tuned parameters. However, the position response shows little oscillation in the steady state. The value of both constants λ and p_1 used in sec.3.2.1 were chosen to be 1.5.

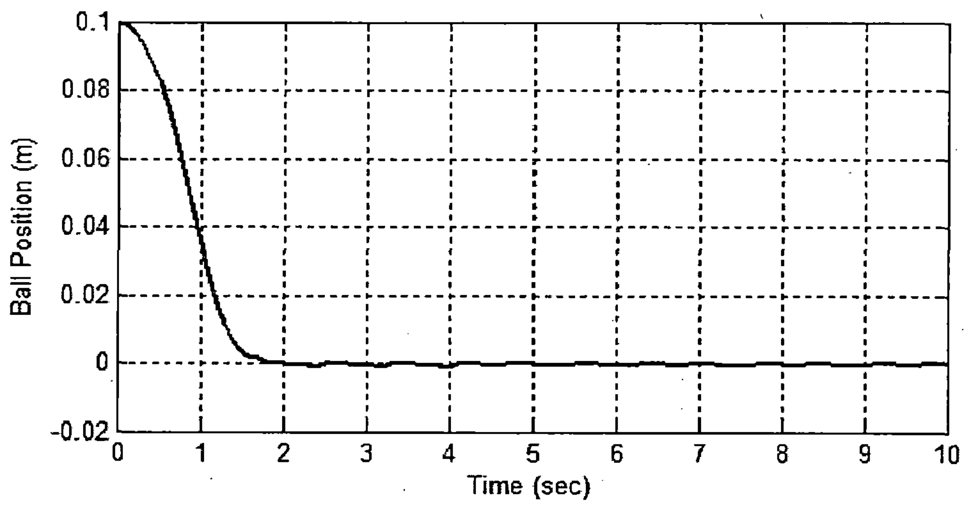

Fig. 8.1 Ball position for simplified model using SMC

The control output of SMC is shown in Fig. 8.2 for simplified ball beam model. It can be seen from figure that control signal is discontinuous unlike conventional controllers. The phase plane plot for the ball position and velocity states is shown in Fig. 8.3, given initial condition (0.1, 0), the SMC is able to reach the sliding line, however, chattering exists along the sliding line.

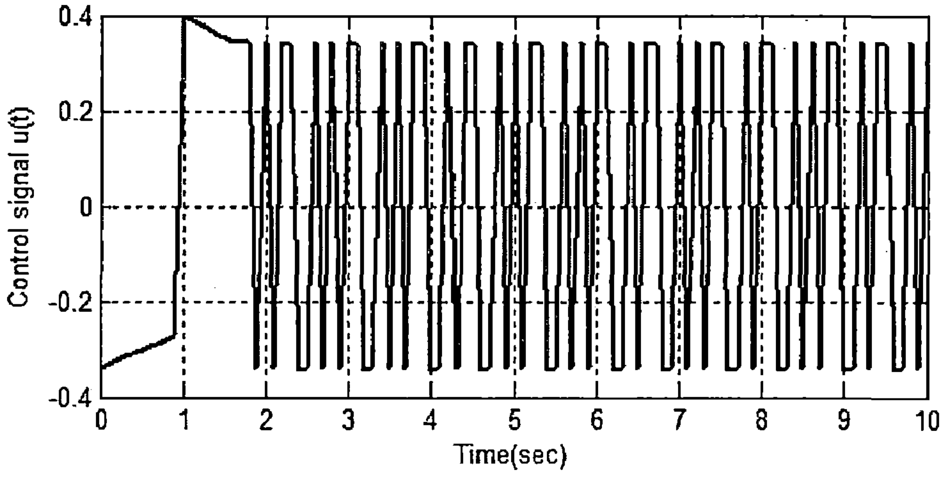

Fig. 8.2 SMC Control output for simplified model

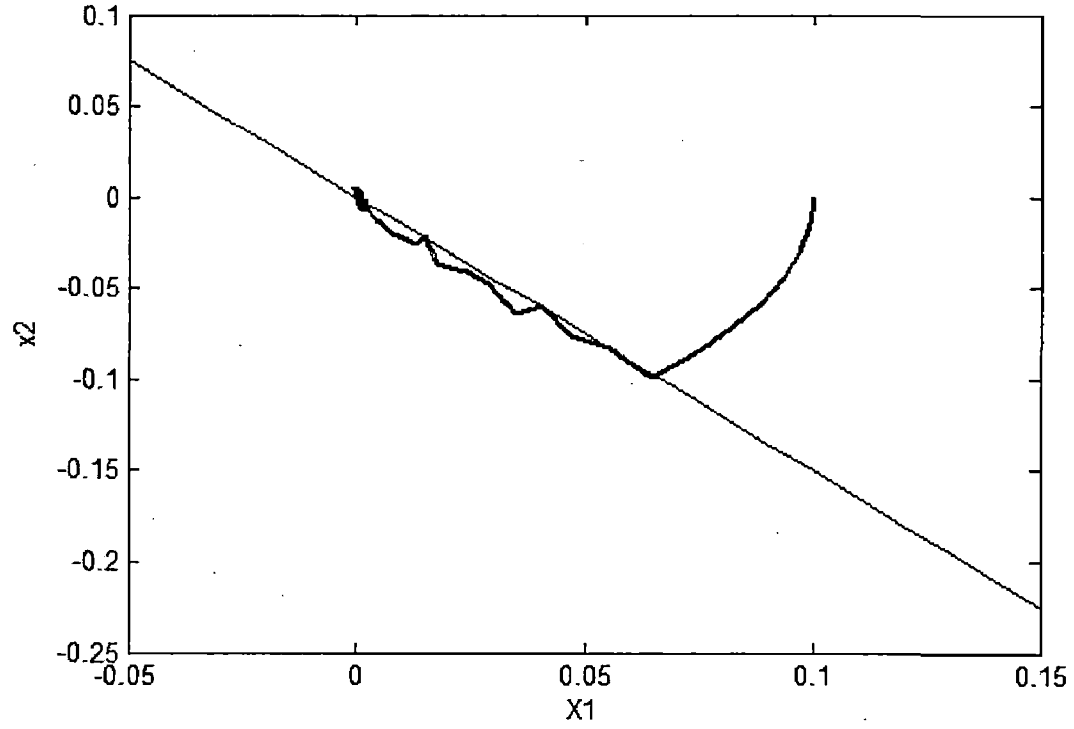

Fig. 8.3 Phase plane plot ball position and velocity state

Simulation result for complete model:

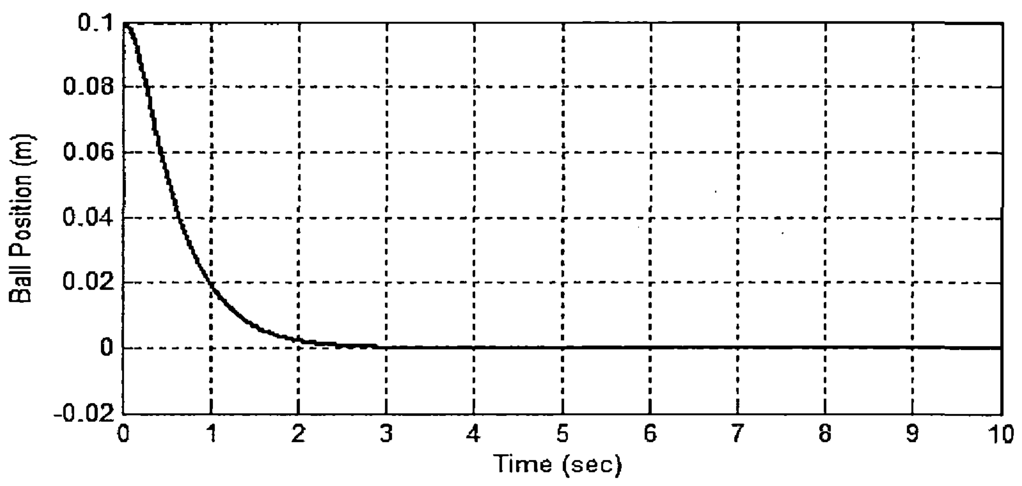

Fig. 8.4 Ball position for complete model using SMC

The SMC control law obtained in eqn. (3.5) for the complete Ball beam model given by eqn. (2.2) and (2.3) has been designed and simulated in MATLAB/Simulink. The ball position response is given in Fig. 8.4 which shows smooth control action with 3 sec to reach steady state. Ball velocity and Beam angle has been shown in Fig. 8.5 and Fig. 8.6 respectively.

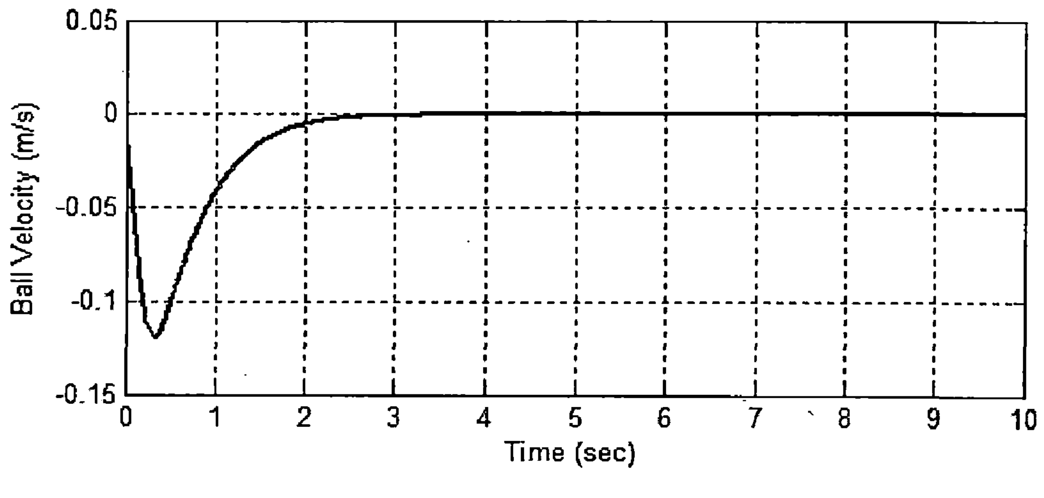

Fig. 8.5 Ball velocity for actual model using SMC

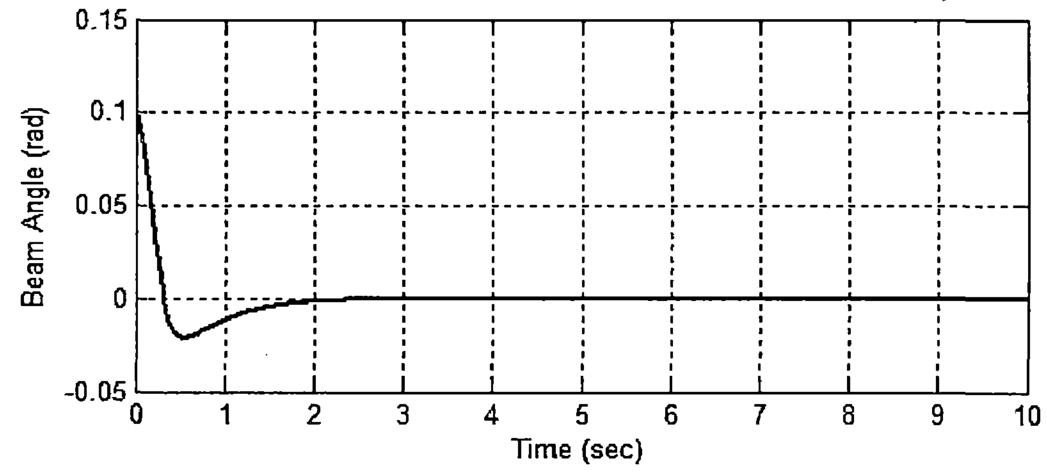

Fig. 8.6 Beam angle vs. Time

The control output considering the complete model is shown in Fig. 8.7. The above figures show that all the states of system reach equilibrium; hence, the SMC designed is able to stabilize the system. The parameters used during simulation have been given in appendix.

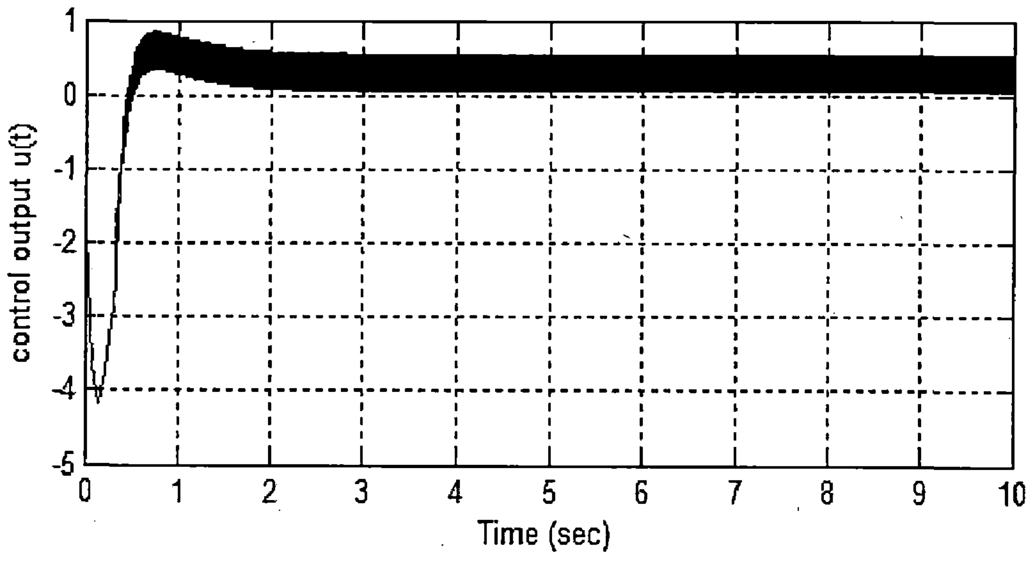

Fig. 8.7 SMC control input for actual model

8.1.2 Fuzzy Logic Controller

The Simulink model of Ball beam system given in eqn. (2.1) is used for simulation purpose. The fuzzy controller designed is a Mamdani FIS with 2 inputs and 1 output. The Ball position error e(t) and change in error $\dot{e}(t)$ are the two inputs whose membership function s was shown in Fig. 4.4. The output membership function was shown in Fig. 4.5. The fuzzy scaling parameters used for control of Ball Beam system are ge = 15, gce = 6 and gu = 3.The ball position response is shown in Fig. 8.8. The surface view plotted with error, change in error and control is shown in Fig. 8.9.

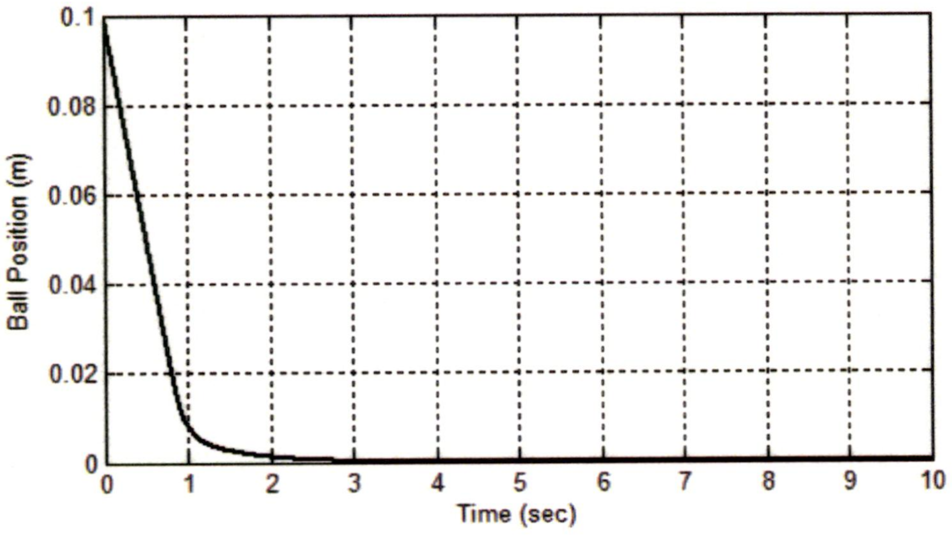

Fig. 8.8 Ball Position response of Fuzzy controller

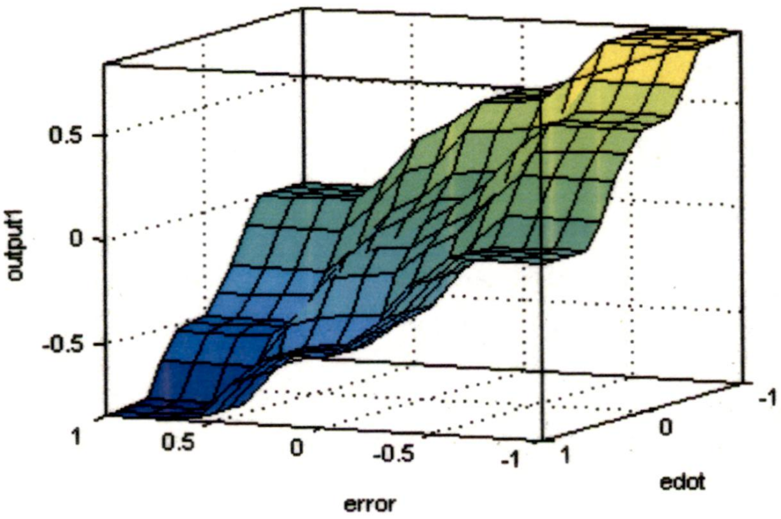

Fig. 8.9 Fuzzy Controller Surface Viewer

8.1.3 Two Level Controller (Fuzzy with SMC Supervisor)

The fuzzy controller designed in the previous section could balance the system with small initial ball positions but as the initial ball position is increased, i.e. the states of the system goes beyond the universe of Fuzzy controller, it is not able to provide required control and hence system does not settle to the desired position. Fig. 8.10 shows the response of Fuzzy controller with initial ball positions of 0.05, 0.1, 0.15, 0.3 and 0.5. It is clear from figure that FLC is able to stabilize the system only for initial ball positions of 0.05 and 0.1 and if this value is increased, FLC is unable to stabilize the ball to reference position which is taken as 0.

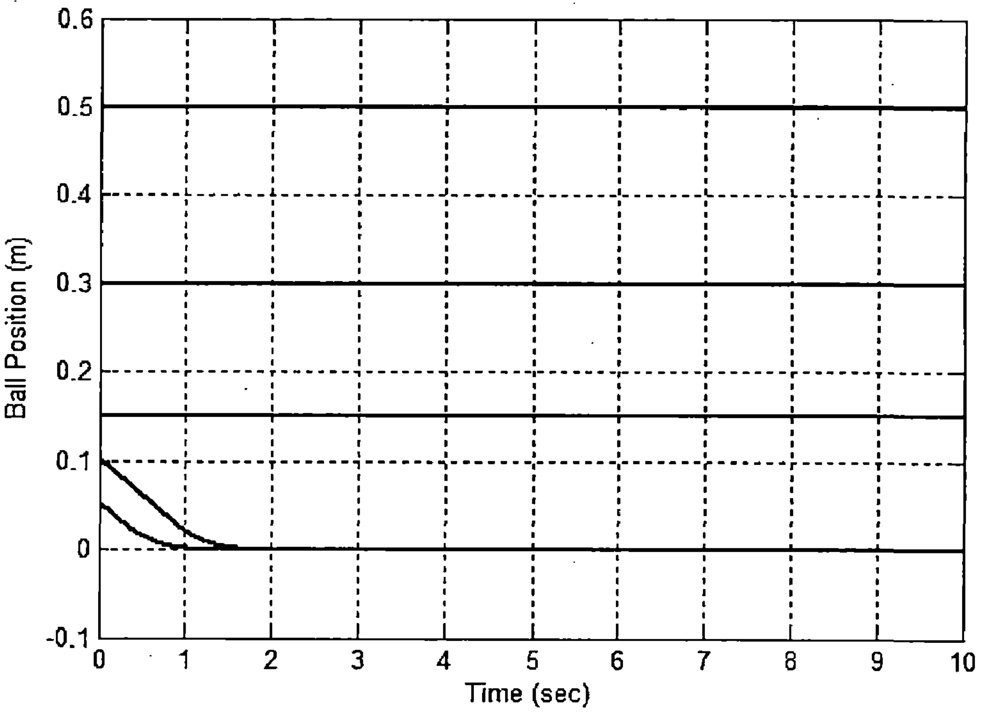

Fig. 8.10 FLC response with different initial conditions

Now, SMC designed earlier is appended to FLC as a supervisory controller using a switching logic such that when FLC is unable to stabilize or the state goes beyond its range, SMC supervisor immediately becomes active and is then responsible for providing sufficient control. FLC with SMC supervisor response with different initial ball position has been shown in Fig 8.11. It is clear from figure that if initial condition is increased beyond 0.1m, this two level scheme is able to stabilize the system. In other words, FLC with Supervisor is able to stabilize the system for any initial ball position.

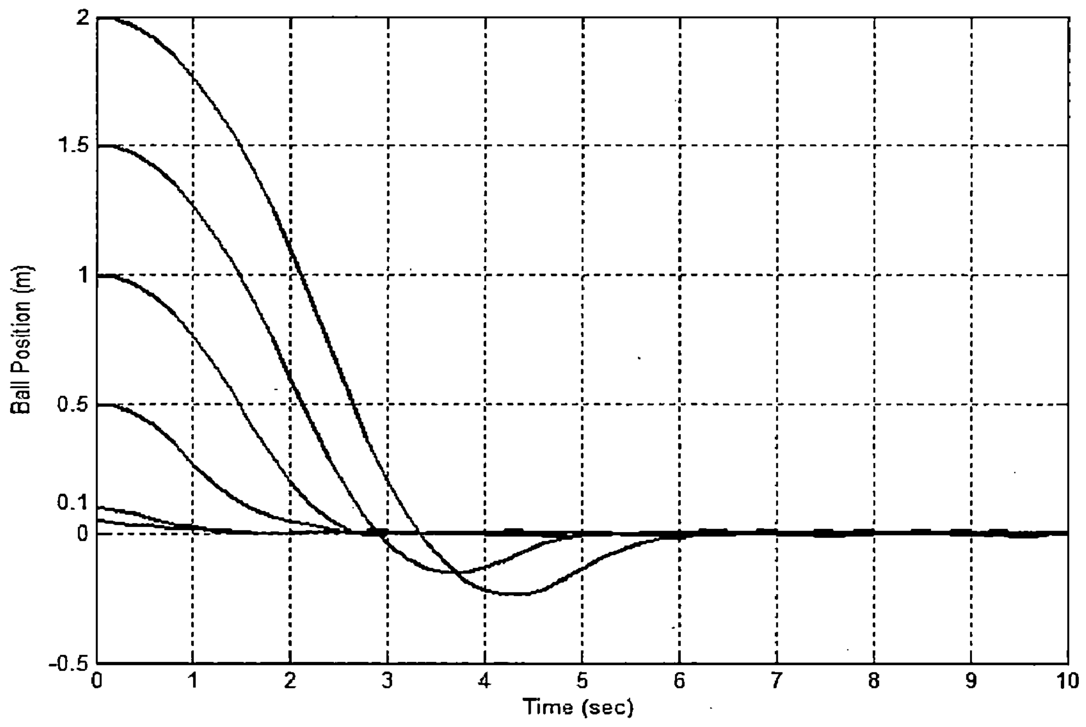

Fig. 8.11 FLC with Supervisory controller for different initial conditions

8.1.4 Neural Network Controller

To design the Neural Controller, first step was to collect the relevant data that may be used for training. To train the NN Controller for its proper behaviour, the idea was to force it to perform similar to well tuned PD controller [20]. The ball position error e(t) and change in error de(t)/dt are the two inputs whose values obtained from well tuned PD controller is fed to train NN. The control signal u(t) obtained from PD is fed as target output. The training operation is done by writing a MATLAB m-file code which defines a backpropagation network and trains it using data provided for training. NN is trained for 1000 epochs using a Levenberg-Marquardt backpropagation network training function. The weight and bias values are adjusted according to gradient descent with momentum weight and bias learning function.

Fig. 8.12 shows the performance plot of NN generated after training, which is plot of Mean square error vs. Number of epochs. Fig. 8.13 shows the fitting of NN actual output with the target output.

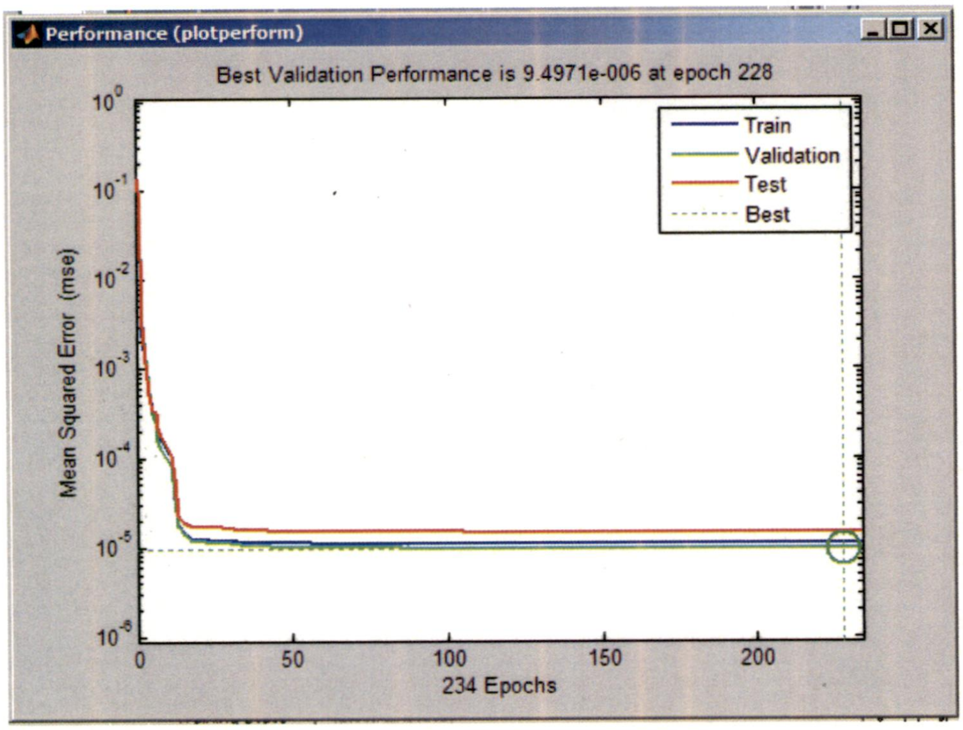

Fig. 8.12 Performance plot for NN

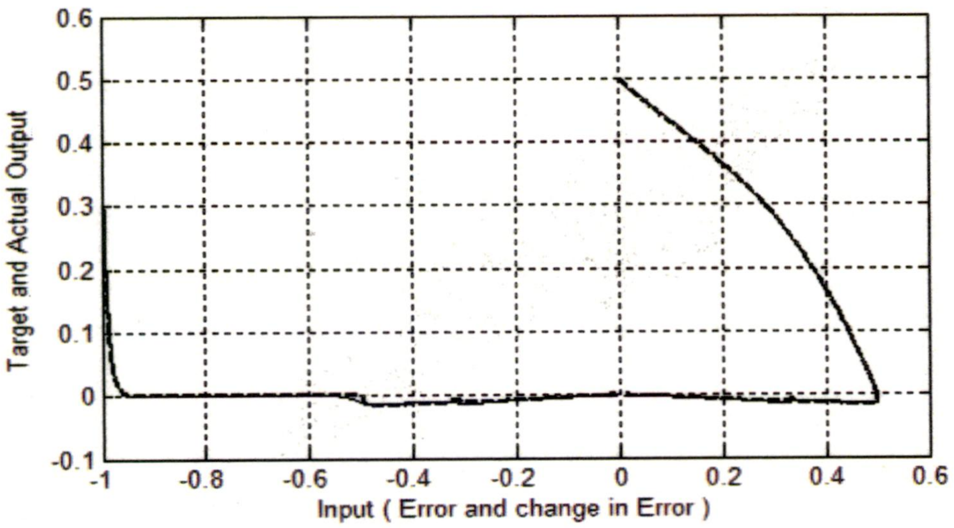

Fig. 8.13 Target and Actual output of NN plotted against input

The dashed line in Fig. 8.12 shows the target output whereas solid line represents actual output of NN controller. It is clear from above figure that NN is able to fit the data exactly.

After observing simulation results, it is clear from Fig., 8.14 that for given, initial condition 0.1 m for ball position and 0 m as the reference position, the NN controller controls the ball and takes around 3.2 seconds to settle it to reference position.

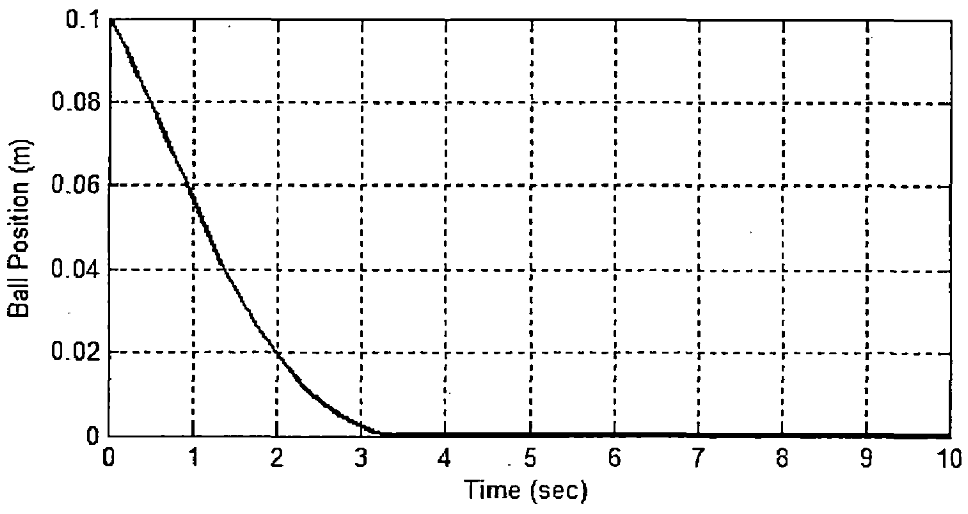

Fig. 8.14 Ball position response of Neural network Controller

8.1.5 ANFIS Controller

To design the ANFIS controller, first step was to collect the relevant data that may be used for training of ANFIS. To train the ANFIS for its proper behaviour, the idea was to force it to perform similar to well tuned PD controller [20] as was done in case of neural network. The second step was to choose the type of signal to be fed to ANFIS as inputs so that all the possible combinations of the different inputs is fed. If the ANFIS is trained with the data obtained only from PD, then in case of parameter variation and external disturbance, ANFIS will not perform satisfactory. Hence, training ANFIS with all input combination ensures proper behaviour in presence of noise and parameter variation.

The ball position error e(t) and change in error de(t)/dt are the two inputs whose all possible combination with in a given range is fed to train ANFIS. The control signal u(t) obtained from PD is fed as target output. Then, ANFIS is trained for 25 epochs using a hybrid training algorithm which is a combination of least square and back propagation gradient descent method. Consequently, membership function parameters of single-output, Sugeno type fuzzy inference systems (FIS) are obtained. Fig. 8.15(a) and 8.15(b) shows the properly tuned membership function generated after training ANFIS. Fig. 8.16 shows the control surface of ANFIS generated Fuzzy controller which is completely linear. The resulted FIS is then used in closed loop with ball and beam system.

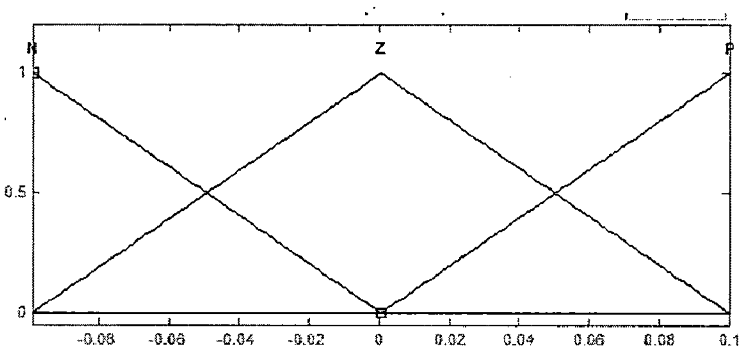

Fig. 8.15(a). Membership function for error

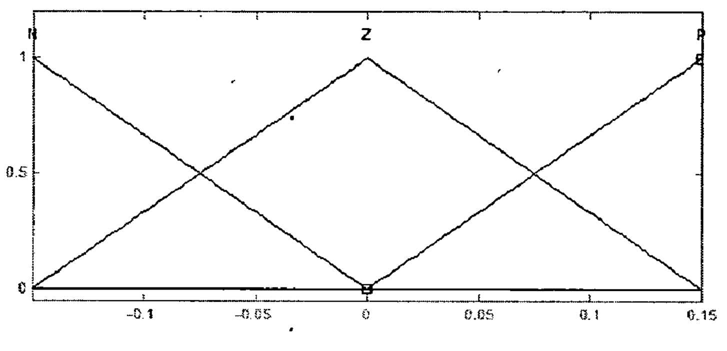

Fig. 8.15(b). Membership function for change in error

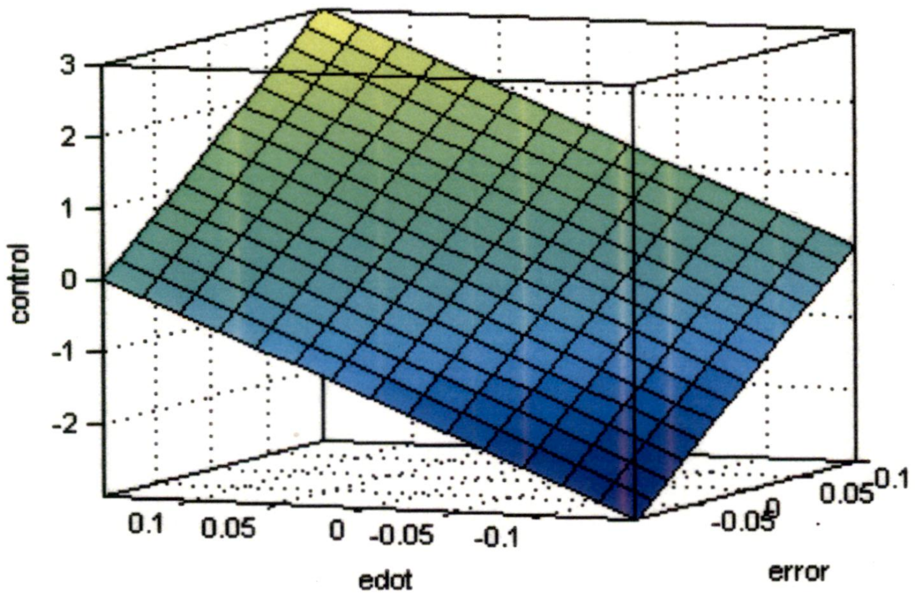

Fig. 8.16 Control surface of ANFIS controller

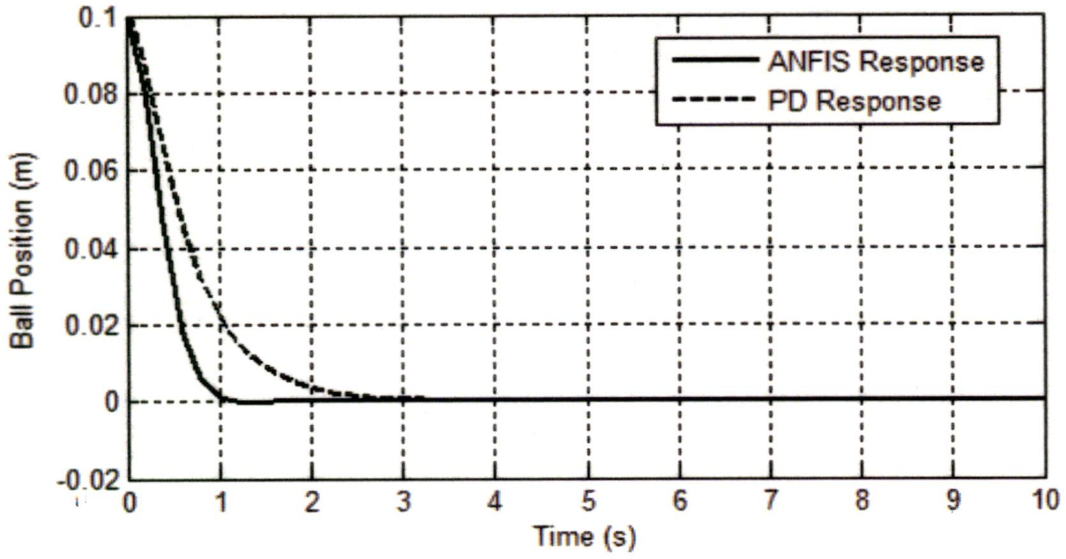

Fig. 8.17 Ball position response of PD and ANFIS controller

After observing simulation result, it is clear from Fig.8.17, given, initial condition 0.1 m for ball position and 0 m as the reference position, the ANFIS controller controls the ball and takes 0.9 seconds to settle it to reference position while PD takes 2.2 seconds.

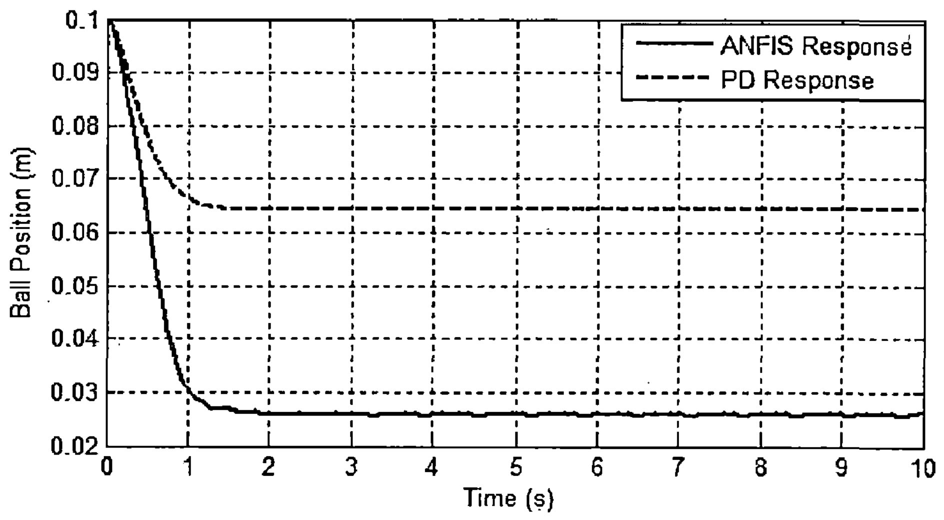

Fig. 8.18 Ball position response in presence of disturbance

The difference in performance of these 2 controllers becomes more observable when external random disturbance is applied to the plant. Fig. 8.18 shows the output of two controllers with disturbance acting on plant. In this case also ANFIS performs better as steady state error is very small as compared to that of PD.

8.1.6 Comparison of various controllers

The output response of different controllers designed and simulated in MATLAB/Simulink is compared with the help of Fig. 8.19. Table 8.1 gives the comparison of simulation results of SMC, FLC, NN and ANFIS controllers designed for ball beam system. It can be observed from the table that the settling time and rise time is least in case of ANFIS. Overshoot and Steady state error is zero in case of all controllers.

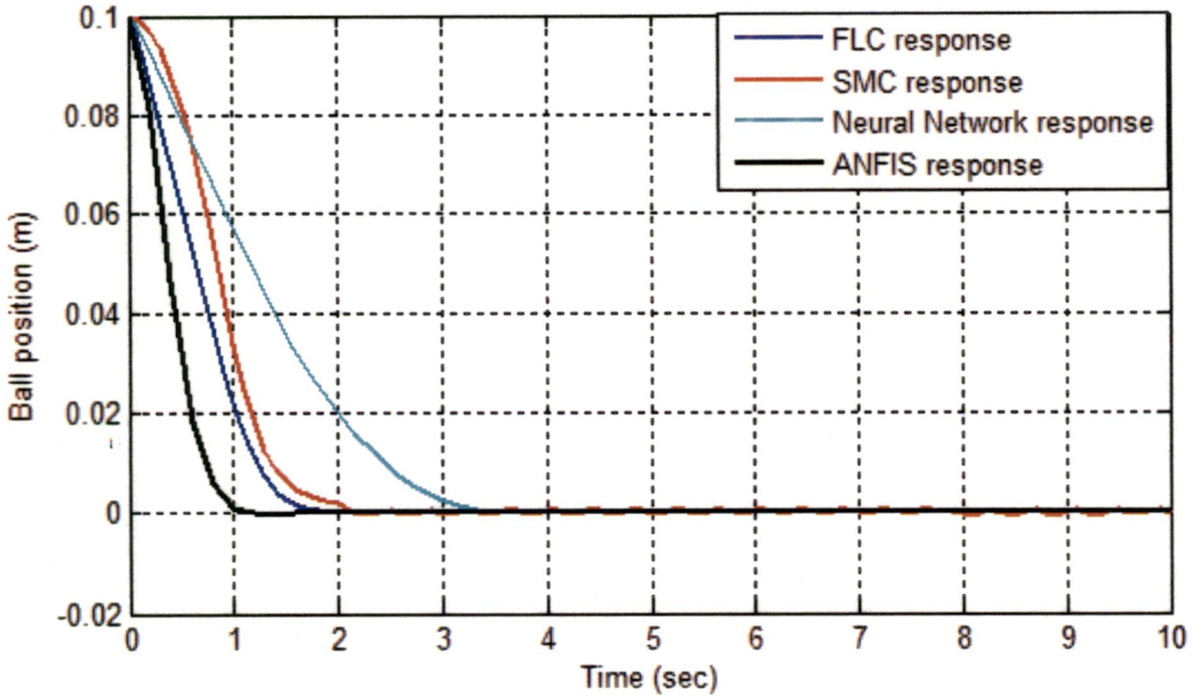

Fig.8.19 Ball position response of different controllers

Specification	SMC	FLC	NN	ANFIS
Settling time	1.9 s	1.5 s	3.05 s	0.9 s
Rise time	1.15 s	1.05 s	2.1 s	0.65 s
Steady state error	0	0	0	0
Overshoot	0	0	0	0

Table 8.1 Comparison of various controllers

8.2 Experimental Results

The real time control application was carried out in the ball and beam hardware. The beam is 40 cm long, the radius and mass of ball are 0.01 m and 0.028 Kg respectively. Input to the system is motor control voltage and output is motor position (θ) and ball position (r). The motion of the motor's shaft is governed by IPM100 intelligent drive which is included within hardware. This is a high precision, fully digital servo drive with embedded intelligence and 100W power amplifier suitable for brushless/brush motors.

Based on feedback information from sensors, it computes error between reference and current ball position and then applies appropriate PWM modulated voltage to the motor windings in such a way that a sufficient torque moves the motor shaft according the user programmed control algorithm.

IPM100 communicates with PC through RS-232 interface. The DC voltage to the drive is provided by the DC power supply. The control program is operated in windows Xp under MATLAB/SIMULINK. Since, resulting ANFIS controller requires ball position as well as velocity to be fed to it, derivative block of Simulink has been used to calculate velocity, as direct velocity measurement is not available. This requires position signals should be smooth enough; hence, first order LPF has been used.

8.2.1 PD controller for Ball beam system

The proportional and derivative gains used for PD controller were 20 and 10 respectively. Fig. 8.20 shows the ball position error as function of time when PD controller controls the system. The response has somewhat larger overshoot and small steady state error.

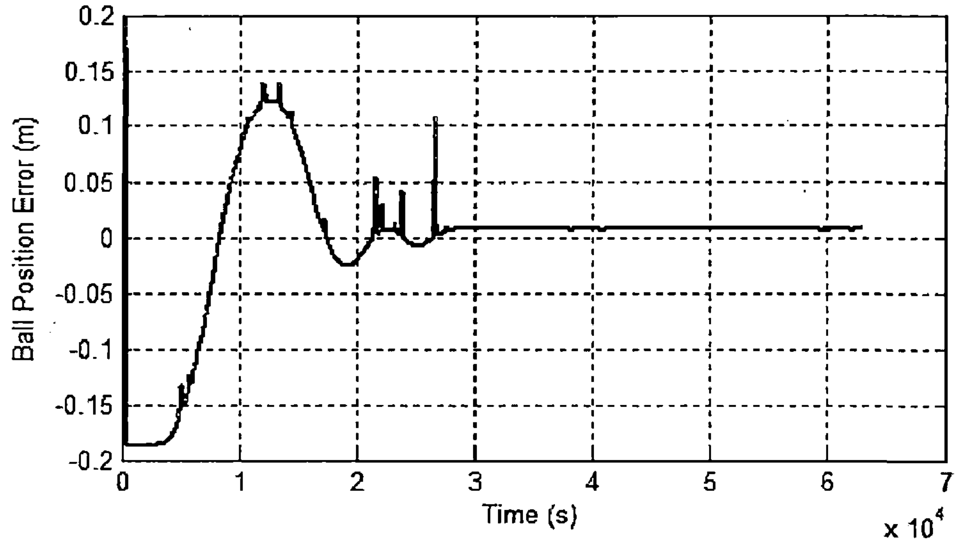

Fig.8.20 Ball position error response of PD controller

8.2.2 NN controller for Ball beam system

The position error response of neural network controller implemented for the real time control is shown in Fig. 8.21. The network was designed with 2 inputs, 1 hidden layer with 5 nodes and 1 output and was trained using Levenberg-Marquardt backpropagation network training function.

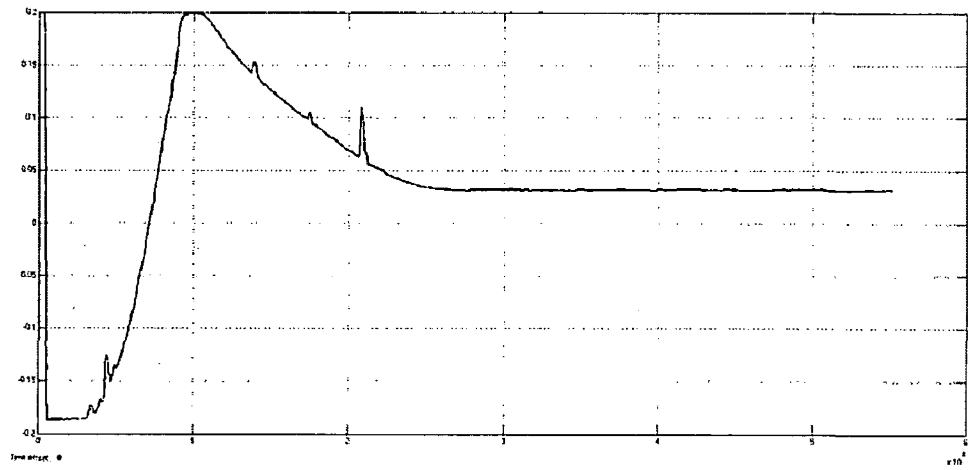

Fig. 8.21 Ball position error response of NN controller

8.2.3 ANFIS controller for Ball beam system

The scaling factor used for error, change in error and control output in case of ANFIS controller were 0.5, 110 and 4 respectively. Fig. 8.22 shows the ANFIS control which stabilizes the system to reference position of 20 cm and results in small steady state error and small overshoot than PD.

8.2.4 Experimental comparison of various controllers

The real time performance of PD, NN and ANFIS controllers has been compared in Table. 2. The table shows that overshoot and steady state error is least in case of ANFIS whereas settling time is least in case of NN.

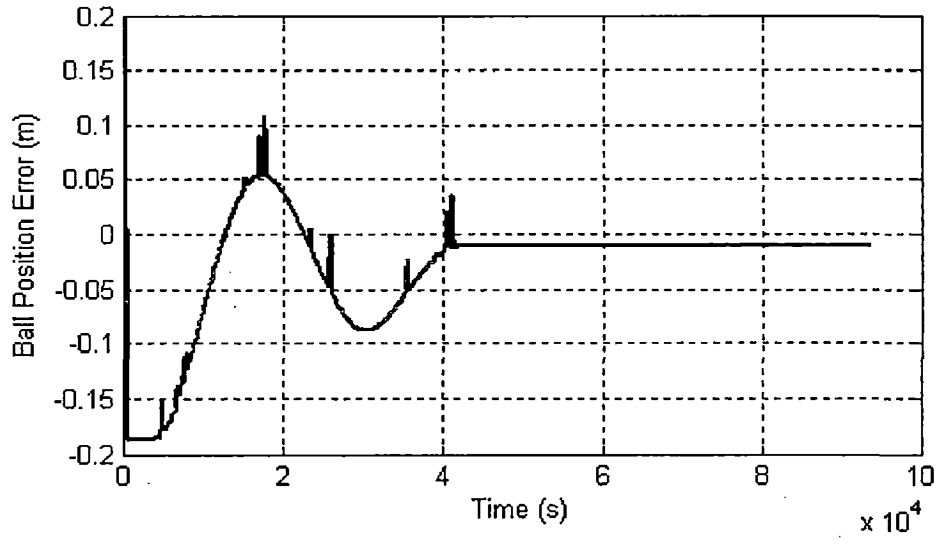

Fig.8.22 Ball position error response of ANFIS controller

Specification	PD	NN	ANFIS
Settling time	2.7 s	2.3 s	3.9 s
Overshoot	65.7 %	105 %	26.4%
Steady state error	0.015 m	0.032 m	0.01 m

Table 8.2 Comparison of PD, NN and ANFIS in real time control

CHAPTER 9

Conclusion and Future scope

Various techniques have been investigated in this research work to control the Ball beam system. The ball beam system is a good selection as it is a standard unstable nonlinear system with simplicity in understanding its operation and characteristics. However, the performance depends on the type of controller used. A Sliding mode controller has been designed and its performance has been analyzed for simple as well as complete model. The simulation results show that the presented SMC is able to control the system satisfactorily.

As the ball beam system's operation is simple to understand and can easily be described in terms of linguistic labels using if-then statements, it is simpler to implement a model free technique than designing an SMC after deriving a complex nonlinear model. The Fuzzy controller designed simply depends on several if then statements and presents a nonlinear control strategy. The results obtained from simulation show the better performance of FLC over SMC in terms of smoothness and speed of response. However, it is seen that SMC has good robustness property than FLC.

A two-level control system with a fuzzy PD controller and a variable structure based supervisor has been presented. The main advantage of two-level control is that different controllers can be designed to target different objectives to meet practical design specifications. The supervisory controller for fuzzy control systems can guarantee that the state of the closed-loop system is uniformly bounded. The advantage of this approach is that we do not need to change the design of the fuzzy controller to guarantee stability; this permits us to design high performance fuzzy controller. The robustness property of variable structure control has also been utilized to make it insensitive to parametric uncertainty and disturbances. The approach has been applied to balance the ball beam system and it has been shown that how the supervisory controller forces the state to be bounded and fuzzy controller balances the system.

To utilize the advantage of model free approach, another controller using neural network technique has been proposed. Neural network represents a good black box, because it has an arbitrary internal configuration that is capable of modelling poorly defined processes. The advantages of Neural Networks include fast computation and fault tolerant feature thus able to perform well under incomplete information. The proposed NN controller is able to smoothly control the ball beam system. However, the response is slower than FLC and SMC.

To further improve the performance and get more intelligent control, the two intelligent control techniques FLC and NN, with each having distinct advantage over other, combined together resulting in ANFIS approach. This eliminates the tuning requirement in FLC thus makes its design simple. The main objective behind designing ANFIS controller was to implement real time control for Ball beam system. When the FLC was designed manually i.e., by manually tuning the membership functions and designing the rule base, the resulting FLC was not able to provide appropriate control action and hence was unable to stabilize the Ball beam hardware. ANFIS, on the other hand generates a properly tuned FLC with the help of training data provided to it. The ANFIS generated FLC, as seen from simulation and experiment results, results in better control action with fastest response among all controllers that have been designed.

The work presented in this dissertation can be extended further to design more robust and intelligent controller by eliminating the problems involved with the present work such as:

i. The chattering problem shown by SMC could be eliminated by designing the control law which softens the discontinuity inside boundary layer as suggested in chapter 3.
ii. The Sliding mode controller as well as Supervisory control scheme works well in simulation. These controllers can be applied for real time control of ball beam system.

References

[1] W. Yu and F. Ortiz, "Stability analysis of PD regulation for ball and beam system," Proc. IEEE Conference on Control Applications, pp. 517-522, 2005.

[2] J. Huang and C. F. Lin, "Robust nonlinear control of the ball and beam system," Proc. American Control Conference, pp. 306–310, 1995.

[3] R. O. Saber and A. Megretski, "Controller design for the beam-and-ball system," Proc. 37th IEEE Conference on Decision and Control, pp. 4555–4560, 1998.

[4] C. Karakuzu, "An Experimental Comparison of Fuzzy, Neuro and Classical Control Techniques," 21st IEEE convention of the Electrical and Electronic Engineers, pp 160-166, 2000.

[5] V. I. Utkin, "Variable structure systems with sliding modes: A Survey," IEEE Transaction on Automatic control, Vol. 22, No. 2, pp. 212-222, 1977.

[6] R. A. DeCarlo, S.H. Zak and G.P. Matthews, "Variable structure control of nonlinear multivariable systems: a tutorial," Proc. IEEE, Vol. 76, No. 3, pp. 212-232, 1988.

[7] J. J. E. Slotine, "Sliding controller design for nonlinear systems," Int. Journal of Control, Vol. 40, No. 2, pp. 421-434, 1984.

[8] L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," IEEE Trans. on System, Man and Cybernetics., Vol. 3, pp. 28-44, 1973.

[9] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," Proc. IEE. Control Theory and Applications, Vol. 121, No. 12, pp. 1585-1588, 1976.

[10] E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," IEEE Trans. on Computer, vol. C-26, No. 12, pp. 1182-1191, 1977.

[11] P. J. King and E. H. Mamdani, "Application of Fuzzy control systems to industrial processes," Automatica, Vol. 13, pp. 235-242, Pergamon Press, 1977.

[12] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller -Part I," IEEE Trans. on System, Man and Cybernetics, Vol. 20, No. 2, pp. 404-435, 1990.

[13] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," IEEE Trans. on System, Man, Cybernetics, Vol. 15, pp. 116-132, 1985.

[14] M. Amjad, M. I. Kashif, S.S Abdullah, and Z. Shareef, "Fuzzy Logic Control of Ball and Beam System," 2nd International Conference on Education Technology and Computer (ICETC), pp.v3-489 – v3-493, 2010.

[15] http://www.aptronix.com/fide/whyfuzzy.html.

[16] M. Amjad, M. I. Kashif, S.S. Abdullah and Z. Shareef, "A Simplified Intelligent Controller for Ball and Beam System," 2nd international Conference on Education Technology and Computer (ICETC), pp. v3-494 – v3-498, 2010.

[17] K. C. Ng and M. M. Trivedi, "Fuzzy logic controller and real time implementation of a ball balancing beam," Applications of Fuzzy logic Technology II, pp. 261-272, 1995.

[18] B. Widrow and M. A. Lehr, "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," Proc. IEEE, Vol. 78, No. 9, pp. 1415-1442, 1990.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," ICS Report 8506, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA, 1985.

[20] I. A. S. Ehtiwesh and M. A. Elhaj, "Design Neural Network Controller for Mechatronic System," World Academy of Science, Engineering and Technology 63 2010.

[21] Y. H. Jiang, C. McCorkell and R. B. Zmood, "Application of Neural Networks for Real Time Control of a Ball-Beam System," Proc. IEEE International Conference on Neural Networks, Vol. 5, pp. 2397-2402, 1995.

[22] Q. Wang, M. Mi, G. Ma and P. Spronck, "Evolving a Neural Controller for a Ball-and-Beam system," Proc. 3rd International Conference on Machine Learning and Cybernetics, Vol. 2, pp. 757-761, 2004.

[23] K. C. Ng and M. M. Trivedi, "Neural Integrated Fuzzy Controller (NiF-T) and Real-Time Implementation of A Ball Balancing Beam (BBB)," Proc. IEEE, International Conference on Robotics and Automation, Vol.2, pp.1590-1595, 1996.

[24] H. W. Tzeng and S. K. Hung, "Design of Ball-Beam Balance Control System Using Neural-Fuzzy Algorithm," IEEE International Conference on Fuzzy Systems, pp.1221-1225, 2009.

[25] J. S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," IEEE Trans. on System, Man and Cybernetics, Vol. 23, No. 3, 1993.

[26] S. N. Engin, J. Kuvulmaz and V. E. Omurlu, "Fuzzy control of an ANFIS model representing a nonlinear liquid-level system," Neural Computing & Application 13, pp.202–210, 2004.

[27] R. Sutton and P. J. Craven, "The ANFIS Approach Applied to AUV Autopilot Design," Neural Computing and Applications, Vol. 7, No. 2, pp.131-140, 1998.

[28] J. Hauser, S. Sastry, and P. Kokotovic, "Nonlinear Control Via approximate Input-Output Linearization: The Ball and Beam Example," IEEE Trans. on Automatic Control, Vol. 37, No. 3, pp. 392-398, 1992.

[29] F. O. Rodriguez, W. Yu, R. L. Feregrino and J. D. J. M. Serrano, "Stable PD control for Ball and Beam system," Proc. International Symposium on Robotics and Automation, pp.333-338, 2004.

[30] J. J. E. Slotine and W. Li, "Applied Nonlinear Control," Prentice Hall, Englewood Cliffs, 1991.

[31] N. B. Almutairi and M. Zribi, "On the sliding mode control of a Ball on a Beam system," Springer Science and Business Media, pp.221-238, 2009.

[32] R. Palm, "Sliding Mode Fuzzy Control," IEEE International Conference on Fuzzy systems, pp. 519-526, 1992.

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

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

[35] L. X. Wang, "Fuzzy systems are universal approximators," Proc. IEEE International Conference on Fuzzy Systems, pp. 1163-1170, 1992.

[36] J. A. Bernard, "Use of Rule-Based System for Process Control," IEEE Control Systems Magazine, Vo1.8, No.5, pp. 3-13, 1988.

[37] C. C. Lee, "Fuzzy logic in control systems: Fuzzy Logic Controller- part II," IEEE Trans. on System, Man and Cybernetics, Vol. SMC-20, No.2, pp.419-435. 1990.

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

[39] L. X. Wang, and J. Mendel "Generating fuzzy rules by learning from examples," IEEE Trans. on System, Man and Cybernetics, Vol. 22, No. 6, 1992.

[40] T. Lee, J. P. Su, and K. W. Yu, "A Practical Design of Fuzzy PD Controller and Its Application to Magnetic Levitation System," IEEE International Conf. on System, Man and Cybernetics, pp. 2018 – 2023, 2008.

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

[42] P. J. Werbos, "Overview of Neural Networks for Control," IEEE Control Systems Magazine, Vol. 11, No. 1, pp. 40-41, 1991.

[43] B. Krose and P. V. D. Smagt, "An Introduction to Neural Network," Eighth edition, 1996.

[44] H. Demuth and M. Beale, "NeuralNetwork Toolbox," The Math works.

APPENDIX

The system parameters for simple as well as actual ball and beam model are:

m	0.028 Kg
g	$9.81 m/s^2$
L	0.40 m
M	0.15 Kg
R_m	9 Ω
J_m	$7.35 X 10^{-4} Nm/rad/s^2$
K_m	0.0075 Nm/A
K_g	75
d	0.04 m
J_l	$0.001 Kgm^2$
K_b	0.5625 V/rad/s

The m-file code to create and train neural network controller:

```
load simerror;
load simedot;
load simcontrol;
P = [simerror'; simedot'];
TR = [simcontrol'];
net1 = newff(P,TR,3,{},'trainlm');
net1 = train(net1,P,TR);
a = sim(net1,P);
figure;
plot(P,a,P,TR)
gensim(net1,-1)
```