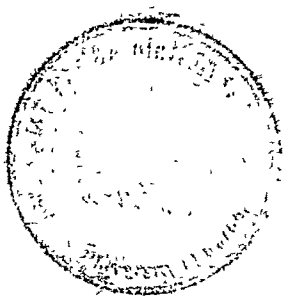


Chapter 3

Soft computing: Techniques and Development Tools

Chapter 3
Soft computing: Techniques and Development Tools

Chapter gives a brief overview of the computing techniques such as Fuzzy logic, Neural network and Neuro-fuzzy networks. The most popular tools used by the researchers for development and simulation study of the system under test such as MATLAB, SIMULINK and associated tool boxes for development of control applications are also described.

Use of tools has been illustrated by an application related to electrical drives. An FLC for the AC and DC drives has been developed. The response of the designed system is simulated using SIMULINK and compared with conventional PID controller.. Single Layer Fuzzy compensator for Non-linearity compensation and Two layer for pre-compensation are also developed. SIMULINK is used for testing the performance of the compensator.

3.1 Introduction

Correct model of process may not be available or mode may be, complex with to many unacceptable assumptions The classical modeling algorithm may not respond well to the measurement noise in sensors or performance through classical algorithms may not be adequate.

The FUZZY LOGIC based systems may be developed to overcome classical algorithm problems. The fuzzy logic frees us from the true/false reasoning of logical system of type that are used in symbolic languages.

Fuzzy linguistic models [1] hold the promise of providing a finite qualitative partition of a quantitative dynamic system while being applicable to any system that can be described in linguistic terms. Fuzzy models provide a succinct and robust representation of systems that lack a complete quantitative model or have uncertain system perturbations. Consistency in reasoning, however, has not yet been proven for a fuzzy linguistic representation of a quantitative system.

Fuzzy linguistic models use fuzzy sets to create a finite number of partitions MBF of the inputs, outputs and states of a quantitative system. Currently most fuzzy models are implemented as a set of **if-then** rules, where the system input is used to evaluate the rules' antecedents and the model's output is the combined output of all the rules evaluated in parallel . This simple logical system, a Fuzzy Inference System (**FIS**) , does not implement inference chaining and can only evaluate a simplified qualitative model of a plant. Recent work has expanded the usefulness of this structure by providing machine learning methodologies to adapt and **tune** fuzzy linguistic models and to automatically generate new models through **self-organization.**

Input output relations (mapping) in the form of traditional mathematical modeling

is replaced by ANN learning the synaptic weights by undergoing a training process. ANN has built in adaptability or can be trained to modify the weights with the change in environment. The ANN can deal naturally with contextual information. Since knowledge is represented by the regular structure and activation state of network. Every neuron is potentially affected by the global activity of all other neurons. ANN can be trained to make decisions and they are also fault tolerant in the sense that if a neuron or connecting link is damaged, recalling a pattern will be impaired in quality but due to distribution of information in the network damage has to be extensive for overall degradation. Since neurons are the common ingredients for all ANN, it is possible to Share the algorithm and structures in different applications. So it is possible to have a seamless integration of modules. The ANN is suitable in the following situation....

Learning or tuning allows the initial linguistic fuzzy model developed from heuristic domain knowledge to be optimized. Learning is achieved by using a neuro-fuzzy structure and exploiting the supervised learning strategies originally developed for neural networks.

These strategies include gradient descent back-propagation, least-mean-squares, and a hybrid methodology that combines least-squares to optimize linear parameters and back-propagation to optimize the nonlinear parameters. These same supervised learning methodologies can automatically learn any arbitrary nonlinear mapping between input and output without an initial linguistic fuzzy model. The resulting self-organized fuzzy models do not necessarily have a linguistic interpretation that would be recognized by a human expert. Often systems developed through self-organization are never interpreted linguistically, but are utilized effectively for pattern matching and curve fitting. Fuzzy networks are often preferred for curve fitting because the fuzzy rules used by the network have only a local effect, in effect providing an adaptive mechanism for implementing B-splines.

It is possible to integrate the fuzzy logic controller with ANN so that the expression for the knowledge used in the systems is understood by humans. This reduces difficulties in describing the ANN. Fuzzy controller learns to improve its performance using ANN structure & thus learns by Experience. Neuro-computing is fast compared to conventional computing because of massive parallel computation. Besides, it has the properties of fault tolerance and noise filtering. Here neural network is used as if estimator. Neural network-based control strictly does not need a mathematical model of a plant like a conventional control method does with the required precision.

3.2 Fuzzy Logic

Fuzzy logic has rapidly become one of the most successful of today's technologies for developing sophisticated control system. Fuzzy logic is nothing but the extension of binary logic. The main difference between the Fuzzy logic and binary logic is that in binary logic we take only two cases, either 0 or 1, that means low or high states. In Fuzzy Logic we take each & every state into consideration. Fuzzy logic is a method for represent ting information in a way that resembles natural human

communication. It then manipulates that information similar to human reasoning. Fuzzy logic has been applied to problems that are either difficult to tackle mathematically or where the use of fuzzy logic provides improved performance. The development of fuzzy logic traces back to 1965 when Dr. Lotfi Zadeh presented a paper on Fuzzy sets. Since, then this Fuzzy logic as a tool has come in long way [2]. Considering industrial application using fuzzy logic controls, is going to be the most important data bank. This is termed as 'Knowledge Base'. Recently, methods of automatically identifying parameters for a fuzzy system has enabled application of fuzzy controls even to those processes where human heuristics are not easily available.

3.2.1 KEYWORDS, TERMINOLOGY

- **DEFINITION**

Fuzzy Logic is the logic using fuzzy set defined by membership functions in the logical expression corresponding to the rule base.

- **FUZZY CONTROL**

It can be defined as a way of defining non- linear table based control where the definition of non-linear transition function can be made without the need to specify each entry of the table individually. It can also be viewed as a knowledge based interpolation technique.

- **LINGUISTIC VARIABLES**

The primary building block of any fuzzy system is linguistic variable while the linguistic term indicate / represent possible values of a linguistic variable. The linguistic variables translate crisp value into a linguistic description.

- **MEMBERSHIP FUNCTION**

A relation between a variable and linguistic variable in terms of a value in the range (0,1) is called membership function.

- **FUZZY SET**

A set obtained by assigning fuzzy values to the linguistic variable using membership function.

- **PREMISE**

It is the fuzzy specification of linguistic variable.

- **CONCLUSION:**

Fuzzy output in terms of linguistic variable is called conclusion.

- **FUZZY RULE**

A linguistic rule derived from the expert's behavior to control process under study. It contains premise as first part and conclusion as second part.

- **FUZZIFICATION**

It is a process of assigning the fuzzy values to linguistic variable using fuzzy set.

- **DE-FUZZIFICATION**

A process of obtaining crisp value for fuzzy output is called defuzzification.

- **UNIVERSE OF DISCOURSE**

Universe of discourse is defined as the total range of all available information in a given problem. Once this universe of discourse is known, we can define certain events on this information space.

- **CRISP and FUZZYSETS**

Crisp sets or classical sets contain information whether it belongs to the set or not. It is very much same as digital logic or Boolean logic. Fuzzy sets on the other hand contain information for which belongings varies from [0,1].This is therefore multi-valued logic.

- **MEMBERSHIP FUNCTIONS**

A membership function (MBF) is a curve that defines how each point in input space is mapped to a membership value (or degree of membership) between 0 and 1. The simplest MF is formed using straight lines. Simplest is the triangular MBF, a collection of three points forming a triangle. Other types include the Trapezoidal, Sigmoidal, pi, Bell shape. MBF.

- **FUZZIFICATION**

It is the process of assigning fuzzy values to linguistic variables using fuzzy set. In the real world, hardware such as a digital voltmeter generates crisp data, but these data are subjected to experimental error. We want to compare a crisp voltage reading to fuzzy sets say as 'low voltage' or 'high voltage'. The range of the output voltage becomes the universe of discourse and the process of identifying a crisp quantity as a fuzzy is called Fuzzification. Assigning of range of membership to the transferred fuzzy quantity in this way is termed as "Fuzzy Measures'. The assignment process can be intuitive, heuristic or it can be based upon the algorithms or logical operations. Fuzzification is the first step where a crisp input is fuzzified.

- **RULE BASE and DECISION MAKING**

Perhaps the most common way to represent knowledge is to form it into natural language expression of the type :

If premise (antecedent), Then conclusion (consequent)

This form of expression is referred to as IF-THEN rule base format. It typically expresses

An inference such that if we know a fact (premise, hypothesis, and antecedent) then we can either infer or derive another fact called conclusion (consequent). This form of knowledge is quite appropriate in the content of linguistics as it expresses human empirical and heuristic knowledge in our own language of communication. Collection of such rules forms a rule base. The dimensions of rule base are dependent upon the fuzzification level of inputs. It is important to note that all the inputs need not condition the output. One input may have an influence on the output independent of other input. The formation of rule is a copyright process of the designer. The process of combining the effect of these rules to produce an output is called as the process of inference.

- **FUZZY INFERENCE**

The fuzzy inference process consists of two phases:

1. The composition of rules applied.
2. Obtaining crisp output from fuzzy inference.

Task of meaningful combination of control actions prescribed by different rules activation or firing is called as composition.

- **DEFUZZIFICATION**

It is a process of rounding off from multi-valued output to a single value output. The output of a fuzzy process can be the logical union of two or more fuzzy

Membership function defined on the universe of discourse of output variable. The Choice of methods is again context dependent. The general criteria are that the method must have continuity, unambiguity and computational simplicity. The centroid method correctly fits into these requirements and almost invariably used.

3.2.2 ARCHITECTURE OF FUZZY CONTROL

The architecture of fuzzy control is shown in Fig 3.1.

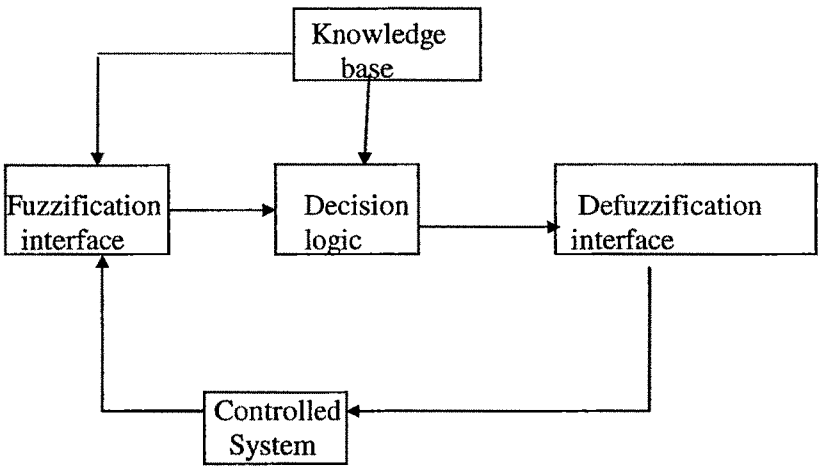


Fig 3.1: THE ARCHITECTURE OF FUZZY CONTROL.

The knowledge base contains information about the boundaries of the linguistic variables data base. Fuzzification interface receives the current I/P value,maps it to a suitable domain i.e it converts the value to a linguistic term /Fuzzy set. Decision logic determines the O/P value from the measured I/P according to the knowledge base. De-fuzzification interface has the task of determining the crisp value. The design steps [3] are as follows:

I. Create linguistic variables and vocabulary for the I/P-O/P variables in which the rules of operation are specified.
II. With reference to the I/P-O/P and control variables, define the structure of the system to represent the flow within a system.
III. Formulate the strategy as fuzzy rules, based on sound Egg. Judgment.
IV. Evaluate rule for current situation to obtain fuzzy output.
V. Use any of the defuzzification technique to obtain crisp value.

3.3 ARTIFICIAL NEURAL NETWORK

Almost all information-processing needs of today are met by digital computers. So we can consider their possibility of information processing techniques that are different from those used in conventional digital computers. Research is being vigorously pursued concerns the possibility of building information processing devices that mimic the structures and operation principles commonly found in humans and other living creatures, which has produced a new breed of computers called neuro-computers.

Neural computing is a new approach to information processing. It is the fast, vital alternative to normal sequential processing. The large computing power lies in parallel processing architecture. It is capable of imitating brain's ability to make decisions and draw conclusions when presented with complex, noisy, irrelevant or partial information. Thus it is the domain in which an attempt is made to make compute think, react and compute.

Neurons are nerve cells and neural networks are networks of these cells. Thus they are one of a group of intelligent technologies for data analysis that differ from other classical analysis techniques by learning about user's chosen subject from the data user provides them, rather than being programmed by the user in a traditional sense. Neural networks gather their knowledge by detecting the patterns and relationships in user's data, learning from relationships and adapting to change.

"A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

- Knowledge is acquired by the network through a learning process.
- Interneuron connections strengths known as synaptic weights are used to store knowledge.

The interconnection architecture can be very different for different networks. Architectures can vary from feed forward and recurrent structures to latticed structures.

3.3.1 The Basic Artificial Model

An *artificial* **neuron** is defined as follows:

It receives a number of inputs either from original data, or from the output of other neurons in the **neural network. Each** input comes via a connection that has strength *(weight)*; these weights correspond to synaptic efficacy in a biological neuron. Each neuron also has a single threshold value. The weighted sum of the inputs is formed, and the threshold subtracted, to compose the *activation* of the neuron (also known as the **post-synaptic potential**, or PSP, of the neuron).

The activation signal is passed through an activation function (also known as a transfer function) to produce the output of the neuron. Inputs and outputs correspond to sensory and motor nerves such as those coming from the eyes and leading to the hands.

There can be hidden neurons that play an internal role in the network. The input, hidden and output neurons need to be connected together.

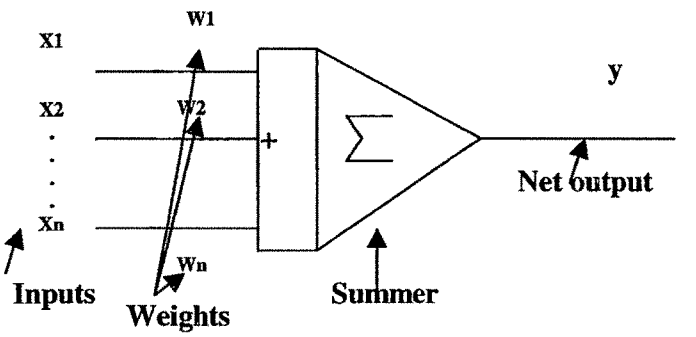


Fig 3.2: Simple model of an Artificial Neuron

A simple network has a *feed forward* structure; signals flow from inputs, forwards through any hidden units, eventually reaching the output units. Such a structure has stable behavior. However, if the network is *recurrent* (contains connections back from later to earlier neurons) it can be unstable, and has very complex dynamics.

When the network is executed (used), the input variable values are placed in the input units, and then the hidden and output layer units are progressively executed. Each of them calculates its activation value by taking the weighted sum of the outputs of the units in the preceding layer, and subtracting the threshold. The activation value is passed through the activation function to produce the output of the neuron. When the entire network has been executed, the outputs of the output layer act as the output of the entire network.

3.3.2 Implementation of Artificial Neural Network:

The specification and design of an ANN application should aim to produce the best system and performance overall. This means that conventional methods should be used if and where possible and ANNs are used to supplement them or only if they can add some benefit. A neural network design involves at least five main tasks:

- ✓ Data collection.
- ✓ Raw data preprocessing.
- ✓ Feature extraction from preprocessed data.
- ✓ Selection of an ANN type and topology (architecture).
- ✓ ANN training, testing and validation.

3.3.3 Single layer and multi-layer networks:

For single layer neural network, the output signals of the neurons in the first layer are the output signals of the network. Here each neuron adjusts its weights according to what output was expected of it, and the output it gave. The Perceptron Delta Rule can mathematically express this:

$$\Delta w_i = x_i \delta$$

where, δ = (desired output) - (actual output).

This is of no use though when you extend the network to multiple layers to account for non-linearly separable problems. When adjusting a weight anywhere in the network, we have to be able to tell what effect this will have on the overall effect of the network. To do this, we have to look at the derivative of the error function with respect to that weight.

Multi layer perceptrons are feed forward nets with one or more layers of nodes between the input and output nodes. Multilayer feed forward networks normally consist of three or four layers; there is always one input layer and one output layer and usually one or more hidden layers. The term input layer neurons are a misnomer; no sigmoid unit is applied to the value of each of these neurons. Their raw values are fed into the layer downstream the input layer (the hidden layer). Once the neurons for the hidden layer are computed, their activations are then fed downstream to the next layer, until all the activations eventually reach the output layer, in which each output layer neuron is associated with a specific classification category. In a fully connected multilayer feed forward network, each neuron in one layer is connected by a weight to every neuron in the layer downstream it. Thus in computing the value of each neuron in the hidden and output layers one must first take the sum of the weighted sums and the bias(if any) and then apply f(sum)(the sigmoid function) to calculate the neuron's activation.

The capabilities of multi layer perceptrons stem from the nonlinearities used within nodes. The number of nodes must be large enough to form a decision region that is as complex as is required by a given problem. It must not, however, be so large that the many weights required cannot be reliably estimated from the available training data. For example, two nodes are sufficient to solve the exclusive OR problem.

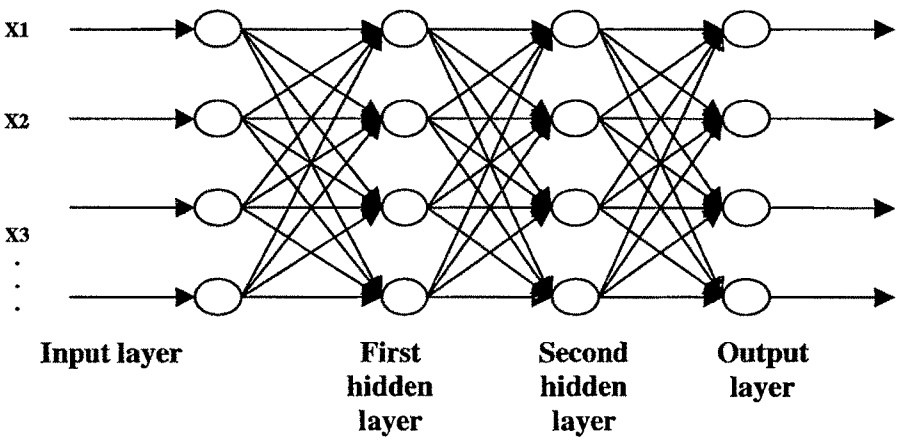


Fig 3.3: Multi-layer Network.

There should be no more than three layers in perceptron like feed forward nets because a three-layer perceptron can form arbitrarily complex decision regions and can separate the meshed classes. There should typically be more than three times as many nodes in the second as in the first layer. The behavior of these nets is more complex because decision regions are typically bounded by smooth curves instead of by straight-line segments and analysis is thus more difficult. These nets, however, can be trained with the new back-propagation training algorithm.

3.3.4 Types of Neural Network Learning

The Artificial Neural Network (ANN) [4] produces response, based on the information encoded in its structure. Usually weights on interconnections between the neurons, store the information. They are adjusted to produce desired response. "The algorithmic process of weight adjustments is called learning rule". The goal of any rule is to adjust weights so as to minimize the difference between the desired and expected response.

The method of setting the value for the weights enables the process of learning or training. 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 that takes place when a network is trained is called learning.

- Supervised learning: Supervised learning is a process of training a neural network by giving it examples of the task we want it to learn. i.e. it is a learning 33with a teacher. The way this is done is by providing a set of pairs of vectors (patterns), where the first pattern of each pair is an example of an input pattern that the network might have to process and the second pattern is the output pattern that the network should produce for that input which is known as a target output pattern for whatever input pattern.

  Supervised leaning means "a learning process in which changes in a network's weights and biases are due to the intervention of any external teacher. The teacher typically provides output targets." This technique is mostly applied to feed forward type of neural networks.

  During each learning or training iteration the magnitude of the error between the desired and actual network response is computed and used to make adjustments to the internal network parameters or weights according to some learning algorithm. As the learning proceeds, the error is gradually reduced until it achieves a minimum or at least and acceptably small value.

  Sometimes if it is not require computing exact error between the desired and the actual network response, and for each training example the network is given a pass/fail signal by the teacher, then it is called *Reinforcement learning* which is a special type of supervised learning. If a fail is assigned, the network continues to readjust its parameters until it achieves a pass or continues for a predetermined number of tries, whichever comes first.

- **Unsupervised learning:**

It is the learning process in which changes in a network's weights and biases are not due to the intervention of any external teacher. Commonly changes are a function of the current network input vectors, output vectors, and previous weights and biases.

The network is able to discover statistical regularities in its input space and automatically develops different modes of behavior to represent different classes of inputs (in practical applications some labeling is required after training, since it is not known at the outset which mode of behavior will be associated with a given input class). In this type of learning due to absence of desired output it is difficult to predict what type of features network will extract. Although learning in these nets can be slow, running the trained net is very fast - even on a computer simulation of a neural net. Table gives comprehensive summary of techniques.

Supervised learning	Unsupervised learning
1 ADALINE. 1 MADALINE. 1 Perceptron. 1 Multilayer perceptron (MLP). 1 Radial Basis Function Network (RBFN). 1 Probabilistic Neural Network (PNN). 1 General Regression Neural Network (GRNN).	1. Hamming Networks. 2. Kohonen's self-organizing maps. 3. Adaptive Resonance Theory (ART). 4. Counter propagations networks(CPN). 5. Neo-cognitions. 6. Adaptive Bidirectional Associative Memory.

Table 3.1: Networks following supervised and unsupervised learning.

3.3.5 BACK PROPAGATION NETWORK (BPN)

Back propagation is a systematic method for training multi –layer artificial networks. It has a mathematical foundation that is strong if not highly practical. It is a multi-layer forward network using extend gradient descent based delta learning rule, commonly known as back propagation (of errors) rule. Back propagation provides a computationally efficient method for changing the weights in a feed forward network,

with differentiable activation function units, to learn training a set of input-output examples. Being a gradient descent method it minimizes the total error of the output computed by the net. The network is trained by supervised learning method. The aim of this network is to be train the net to achieve a balance between the ability to respond correctly to the input patterns that are used for training and the ability to provide good response to the input that are similar.

The training algorithm of back propagation involves four stages , viz.

1. Initialize of weights
2. Feed forward
3. Back propagation
4. Updating of the weights and biases.

During first stage which is the initialization of weights, some small random values are assigned. During feed forward stage each input unit (xi) receives an input signal and transmits this signal to each hidden units z1...zp. Each hidden unit then calculates the activation function and sends its signal zj to each output unit. The output unit calculates the activation function to form the response of the net for the given input pattern.

During back propagation of errors, each output unit compares its computed activation y_k with its target value t_k to determine the associated error for that pattern with that unit. Based on the error, the factor δk (k= 1,...m) is computed and is used to distribute the error at output unit y_k back to all units in the previous layer. Similarly, the factor δj(j=1,..p) is computed for each hidden unit zj. During the final stage, the weight and biases are updated using the δ factor and the activation. During final stage, the weight and biases are updated using δ factor and the activation.

PARAMAETER

x: Input training vector.	t: output target vector
δk : error at output unit yk	δj : error at hidden unit zj
α = learning rule	Voj = bias on hidden unit j
zj= hidden unit j	wok = bias output unit k
y = output unit k.	

Initialization of weights

Step: 1 Initialize weight to small random values.

Step: 2 While stopping condition is false, do steps 3-10

Step: 3 For each training pair do steps 4-9

- **Feed forward**

Step 4: Each input unit receives the input signal xi and transmits this signals to all units in the layer above i.e. hidden units.

Step 5: Each hidden unit (zj, j = 1,..p) sums its weighted input signals

$$z_{-inj} = voj + \sum_{i=1}^{n} xivij \tag{3.1}$$

Applying activation function

$$Zj = f(z_{inj}) \tag{3.2}$$

And sends this signal to all units in the layer above i.e. output units.

Step 6 : Each output unit (yk, k = 1,..m) sums its weighted input signals

$$y_{-ink} = wok + \sum_{j=1}^{p} zjwjk \tag{3.3}$$

And applies its activation function to calculate the output signals

$$Yk = f(y_{-ink}) \tag{3.4}$$

- **Back propagation of errors**

Step 7: Each output (yk, k=1,..m) receives a target pattern corresponding to an input pattern, error information term is calculated as

$$\delta k = (tk - yk) f(y_{-ink}) \tag{3.5}$$

Step 8: Each hidden unit (zj, j = 1,...n) sums its delta inputs from units in the layer above

$$\delta_{-inj} = \sum_{k=1}^{m} \delta kwjk \tag{3.6}$$

The error information term is calculated as

$$\delta j = \delta_{-inj} f(z_{-inj})$$

Updation of the weights and biases.

Step: 9 Each output unit (y, k =1,...m) updates its bias and weights (j=0,...p)

The weight correction term is given by

$$\Delta Wjk = \alpha\delta kzj \tag{3.7}$$

And the bias correction term is given by

$$\Delta Wok = \alpha\delta k \tag{3.8}$$

Therefore, $Wjk(new) = Wjk(old) + \Delta Wjk$, $Wok(new) = Wok(old) + \Delta Wok$

(3.9)

The weight correction term

$$\Delta Vij = \alpha\delta jxi \tag{3.10}$$

The bias correction term

$$\Delta Voj = \alpha\delta j$$

Therefore, $Vij(new) = Vij(old) + \Delta vij$, $Voj(new) = Voj(old) + \Delta Voj$

(3.11)

Step 10: Test the stopping condition. The stopping condition may be to the minimization of the errors, number of epochs etc

3.4 Software Development Tools

The developments tools such as MATLAB, SIMULINK, and tools boxes are described in the section. Their use is illustrated by applications.

3.4.1 MATLAB 7

MATLAB [5] is a high-performance language for technical computing. The name **MATLAB** stands for ***matrix laboratory***. A numerical analyst called Cleve Moler wrote the first version of MATLAB in the 1970s. It has since evolved into a successful commercial software package. The MATLAB system consists of five main parts:

Development Environment: - This is the set of tools and facilities that help to use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing help, the workspace, files, and the search path. Main window of MATLAB is as shown in figure 3.4

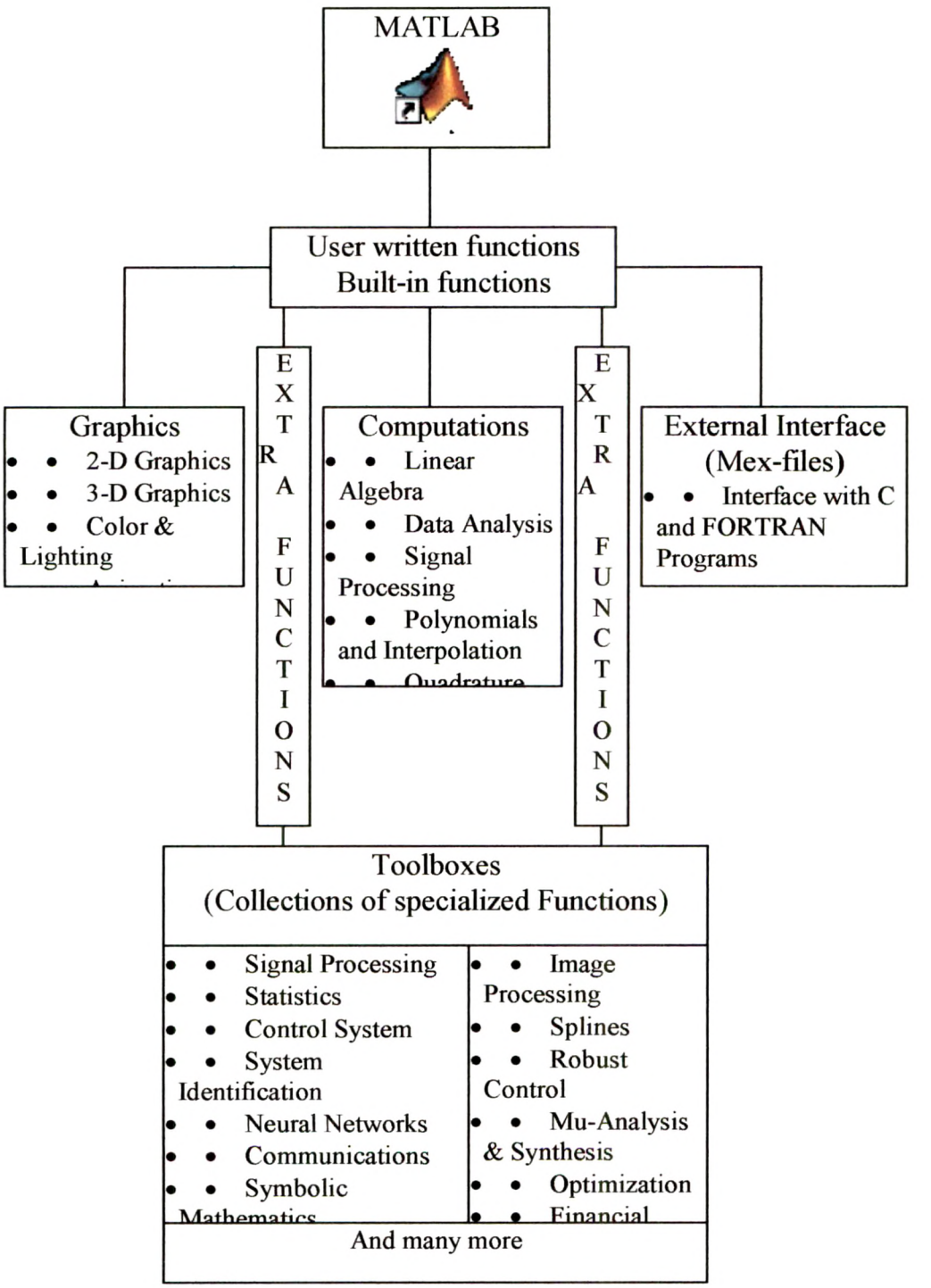


Fig 3.4(A): A schematic diagram of Main features: MATLAB

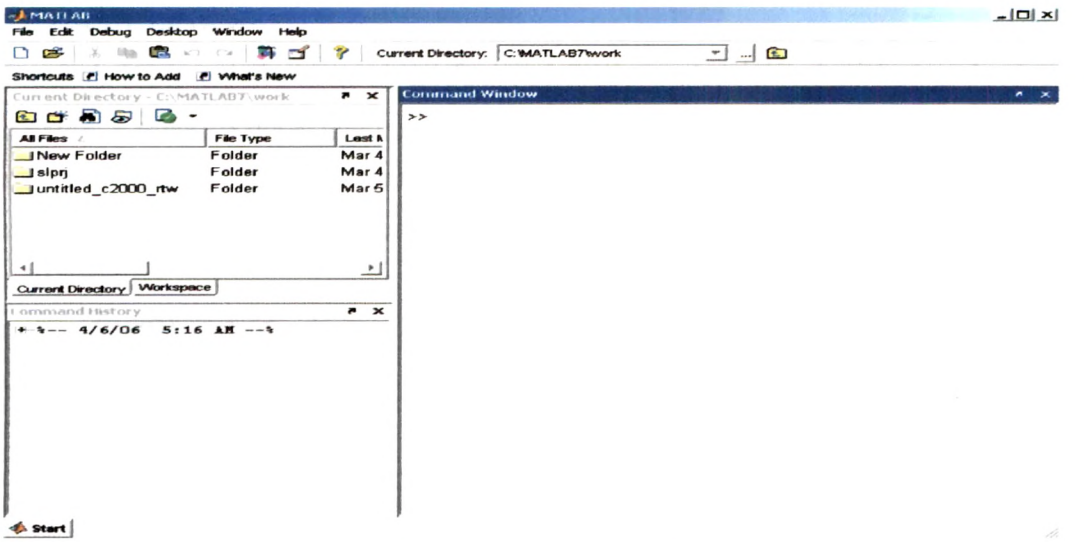


Fig 3.4: MATLAB command window

The MATLAB Mathematical Function Library: - This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic.

The MATLAB Language: - This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features.

The MATLAB Application Program Interface (API): - This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files. Toolboxes available in MATLAB 7.0 and used in the thesis are listed in table 3.2

Control System Toolbox	Model Predictive Control Toolbox
Optimization Toolbox	Robust Control
Neural Network	Fuzzy Logic

TABLE 3.2: TOOL BOXES used from MATLAB 7

Functions	description
Addmf	Add a membership function to an FIS
Addrule	Add a rule to an FIS
Addvar	Add a variable to an FIS
Evalfis	Perform fuzzy inference calculations
Newfis	Create new FIS
Trimf	Triangular membership function

Table 3.4 functions used to create fuzzy system

3.4.4 Neural Network Toolbox

The MATLAB neural network toolbox provides a complete set of functions and a graphical user interface for the design, implementation, visualization, and simulation of neural networks. It supports the most commonly used supervised and unsupervised network architectures and a comprehensive set of training and learning functions. The neural network toolbox extends the MATLAB computing environment to provide tools for the design, implementation, visualization, and simulation of neural network. Table 3.5 lists MATLAB functions used for training and learning of the ANN controller [8].

Functions	description
newff :	Create a Feed forward back propagation network.
purelin	Linear transfer function
tansig	hyperbolic tangent sigmoid transfer function.
Traingd	Gradient descent back propagation.
sim	Simulation of simulink model
gensim	Gnerate simulink block simulate a neural network.
Train	trains a network NET according to NET.trainFcn and NET.trainParam.

Table 3.5: Functions used from ANN Toolbox

3.5 Application: Electrical Drives

An electric drive [9] is a well established industrial drive as it has several advantages and special features. AC/DC drives are widely used in the applications requiring adjustable speed, good speed regulation and frequent starting, braking and reversing. Some important applications of the DC motor is in the rolling mills, machine tools, printing press, cranes, It is being predicted that the AC drives will replace DC drives. But variable speed applications are dominated by DC drives, because of lower cost, reliability and simple control.

The electric drive system employs closed loop controls and principles of feedback control theory. The conventional feedback control theory can be applied to determine the time domain and frequency domain behavior of the system. The stability of the drive, which is necessary but not sufficient conditions, may be analyzed using the conventional Routh-Hurwitz and Nyquist stability criteria. Based on these methods the design of controllers for stabilization of the system is possible both in time domain using root locus techniques and frequency domain using Bode plots.

The AC drive system utilizing induction and synchronous motors may be considered to be multivariable systems. These can be analyzed using state space techniques to determine the drive behavior. Conventionally AC and DC variable speed drive incorporate a number of separate controllers. They are used for input variable controls of the machine. The closed loop control for AC motors is discussed in the following section.

Conventional controllers are based on the mathematical model of the linear process. Conventional controllers will be effective if the speed and accuracy requirements of control systems are not critical under varying environment of the systems. PID controllers can be used but it can not cope with the varying control environments resulting due to the load disturbances, non-linearity of the systems and also change of plant parameters.

All the physical systems have some kind of non-linearity. Sometimes it may even be desirable to introduce a non-linearity in order to improve the performance of the system and make its operation safer. In most of the control system we cannot avoid the presence of certain types of non-linearities. Some common non-linearities are saturation, dead-zone, friction etc. The nonlinear system may be highly sensitive to input amplitude. The stability of the nonlinear systems is dependent on the input and also the initial state. The stability study of the non-linear systems infact requires the information about the type and amplitude of anticipated inputs, initial conditions, in addition to the usual requirements of the physical and mathematical models of the systems. There are two methods from which the information about transient behavior and stability is easily obtained. One is the Phase plane method and another is the Describing Function method, based on the harmonic linearization. The input to the non-linear component is sinusoidal and depending on the filtering property of the linear part of the overall system. The output is represented by the fundamental frequency term in the Fourier series.

Fuzzy Logic Controller (FLC) [10-11] is used for the application of the non-linear industrial application process. FLC yields superior results to those obtained by the conventional controllers. FLC's have common feature of not requiring a detailed mathematical model and lead to much faster and accurate results. The robustness of FLC is a commendable feature in motor drive applications, where the system parameters are widely varying during plant operation. FLC design is made easier by tools of Fuzzy Logic.

Due to the nonlinear structure of the FLC, the main design problem lies in the determination of the consistent and complete rule ser and the shape of the membership functions. A lot of modifications and trial and error has to be done in order to obtain the desired response which is time consuming. Another powerful technique which is the most accurate and faster is Neuro-Fuzzy Controller (NFC) [12] design. It helps to generate and optimise membership functions as well as the rule base from the simple data provided. Combining the learning power of the neural network with the Fuzzy Logic gives the Neuro-Fuzzy system. Neuro-Fuzzy controller design is done by using ANFIS (Adaptive Neuro Fuzzy Interface System).

1 The simulation is carried out using the fuzzy logic toolbox and SIMULINK in MATLAB. The plant response without any controller for step input shows oscillations. In order to get the accurate speed control of the DC motor, the response is improved incorporating FLC design. Fuzzy logic compensator compensates the effects of the non-linearity. The dead-zone ANFIS is incorporated to eliminate tedious procedure of the modifications in the membership functions and rule base in FLC and obtain desired response, quickly and easily. The results in the SIMULINK shows the comparative responses of the controllers with the original response of the plant, when step input is applied.

2 The dead-zone (non-linearity) is added to the system. The Fuzzy Logic Compensator effects are also seen after adding dead-zone. The SIMULINK shows the comparative responses of the controllers, when sinusoidal input is applied to the system.

The electric drive system employs closed loop controls and principles of feedback control theory. The conventional feedback control theory can be applied to determine the time domain and frequency domain behavior of the system. The stability of the drive, which is necessary but not sufficient conditions, may be analyzed using the conventional Routh-Hurwitz and Nyquist stability criteria. Based on these methods the design of controllers for stabilization of the system is possible both in time domain using root locus techniques and frequency domain using Bode plots.

The AC drive system utilizing induction and synchronous motors may be considered to be multivariable systems. These can be analyzed using state space techniques to determine the drive behavior. Conventionally AC and DC variable speed drive incorporate a number of separate controllers. They are used for input variable controls of the machine. The closed loop control for AC motors is discussed in the following section.

3.5.1 MOTOR CONTROL: (Conventional Controller)

The AC drive [9] chosen for the system is three phase induction motor. The specifications of induction motor (IM) are given in Appendix - 1. In IM, speed control can be obtained by various conventional techniques. Due to their inherent limitations on overall performance, a new inverter fed IM controls are used for variable speed drives. For a constant air gap flux in the motor, the ratio of voltage to frequency (V/f) is to be kept constant in all the operating region. A simple and popular closed loop V/f speed control method is shown in Fig 3.6

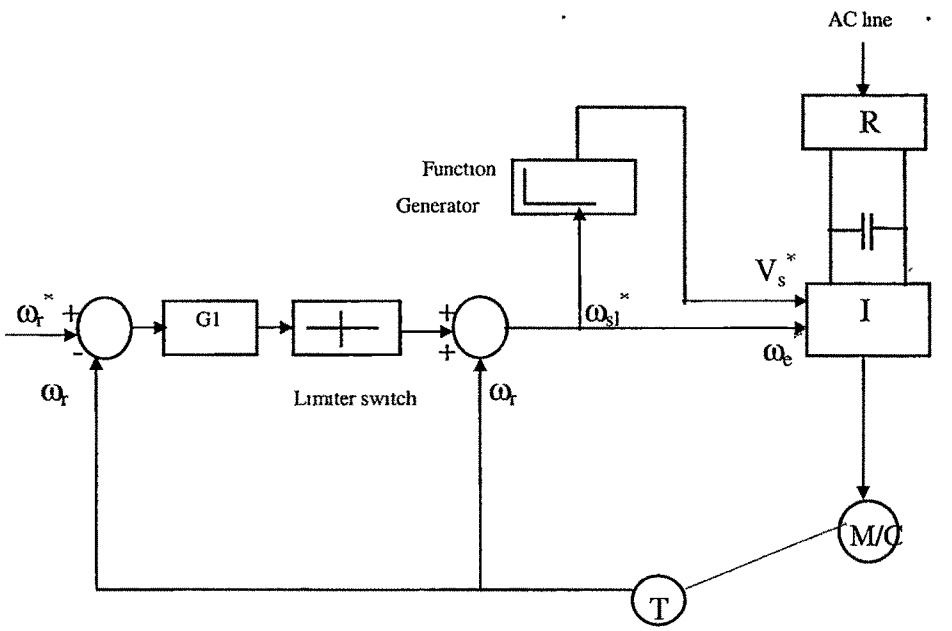


Fig 3.6: Constant volts/hertz. Speed Control with slip regulation

The scheme is defined as V/f control because voltage and frequency command are obtained by speed feedback through a controller. In steady state operation, the machine (M/C) air gap flux (Ψm) is approximately related to ratio of V/f. Therefore maintaining the rated air gap flux will provide the maximum torque sensitivity with stator current which is similar to that of DC machine. This slip regulating scheme is shown in Figure-1, where error of speed control loop generates the slip command (ω_{sl}^*) through a proportional integral controller & limiter.

The slip is added with speed signal (ω_r) to generate frequency command (ω_e^*). The frequency command also generates the voltage command (V_s^*) through a V/f function generator which incorporates the low frequency stator drop compensation. Since the slip is proportional to the developed torque, the scheme can be considered as torque control within a speed control.

The block diagram Fig 3.7 consists of the controller block, inverter block & the machine dynamics block with the feedback loop. The representation of these blocks is in the form of simple transfer function.

The machine dynamics can be taken of higher order but since the response is dominated by the dominant roots, a simple representation is taken into account. A tachogenerator feedback is taken for closed loop performance.

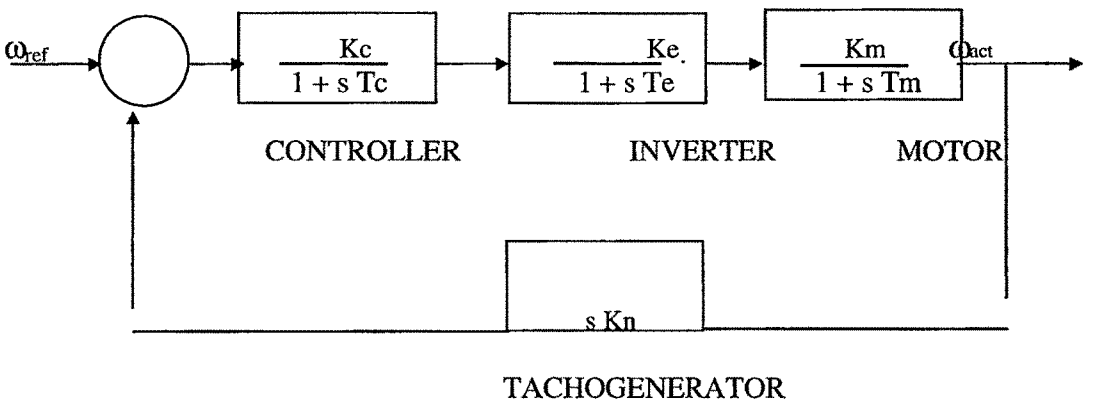


Fig 3.7: BLOCK DIAGRAM REPRESENTATION OF THE SYSTEM

- **LIST OF SYMBOLS:** The symbols used in Fig 3.6 and Fig 3.7 are as under:

  * ω_r^* = command speed
  * ω_r = actual speed
  * G_1= controller
  * ω_{sl}^* = command slip
  * V_s^* = rectifier command voltage
  * ω_e^* = command frequency
  * R = Rectifier
  * I = Inverter
  * M/C = Induction motor
  * T = Tachometer
  * K_c = controller gain constant
  * K_e = inverter gain constant
  * K_m = motor gain constant
  * K_n = tachometer gain constant
  * T_c = controller time constant
  * T_e = inverter time constant
  * T_m = motor time constant
  * ω_{ref} = reference angular speed
  * ω_{act} = actual angular speed

- **MATHEMATICAL MODELLING:** The values of the time constants, amplifier gain and motor gain constants and tachometer feedback gain are given below:

I. M.: 3 ϕ, 2W, 415 V, 4 pole

Kc	=	1.4
Ke	=	3
Km	=	10
Tc	=	0.02
Te	=	0.05
Tm	=3	0.15
Kt	=	1

The value of Kc is determined with the help of Nyquist criteria to obtain the desired performance i.e stable operation.

The open loop transfer function of the motor is given by:

$$G(s) = \omega_{act} / \omega_{ref}$$

$$G(s) = K_c*K_e*K_m / (1+sT_c)(1+sT_e)(1+sT_m)$$

$$G(jw) = K_c*K_e*K_m / [1-w^2(T_c*T_m+T_c*T_e+T_e*T_m)] - jw[w^2*T_c*T_e*T_m-T_c-T_e-T_m]$$

$$G(jw) = K_c^*K_e^*K_m^*\{1-w^2[T_c*T_m+T_c*T_e+T_e*T_m] + jw[w^2*T_c*T_e*T_m-T_c-T_e-T_m]\}/\{1-w^2(T_c*T_m+T_c*T_e+T_e*T_m)^2 + w^2(w^2*T_c*T_e*T_m-T_c-T_e-T_m^2)\}$$

The locus of G (jw) as w changes from zero to infinity is drawn which is known as Nyquist plot. The Nyquist plot as shown in Fig 3.8 represents the stability frequency response of a closed loop system. If the frequency response of open loop transfer function encloses critical point (-1,0), the closed loop system becomes unstable. The value of K_c is found out when it crosses (-1,0). Therefore,

$$K_c* K_e* K_m*w (w^2*T_c*T_e*T_m-T_c - T_e-Tm)=0 \text{ --------} \quad (3.12)$$

Since: real part should be -1:

$$K_c*K_e*K_m / \{1-w^2[T_c*T_m+T_c*T_e + T_e*T_m]\} = -1\text{--------} \quad (3.13)$$

Using 1 & 2:

$$|0.429 * K_c| = 1 \rightarrow Kc = 2.33$$

Hence for stable operation K_c should never exceed beyond 2.33. The value of K_c should be 0.6 to 0.8 times this value. For our system we have taken it as 0.6 times the critical value of K_c i.e $K_c = 0.6 * 2.33 \cong 1.4$. With Kn = 1V / rad / sec reduces to the one as shown in Fig 3

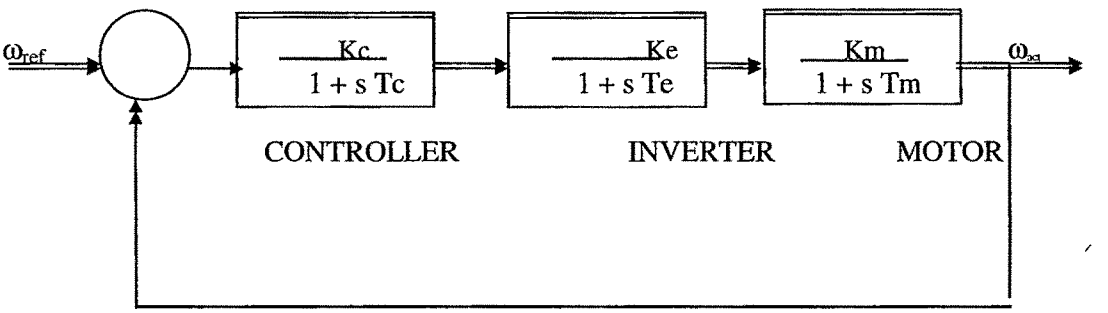


Fig 3.8: BLOCK DIAGRAM WITH K_n = 1 V/rad/sec.

$$\omega_{act}/\omega_{ref} = \frac{Kc*Ke*Km/(1+sTc)*(1+sTe)*(1+sTm)}{1 + \frac{Kc*Ke*Km}{(1+sTc)*(1+sTe)*(1+sTm)}}$$

$$= \frac{Kc*Ke*Km}{(1+sTc)*(1+sTe)*(1+sTm)+Kc*Ke*Km}$$

$$= \frac{Kc*Ke*Km/Tc*Te*Tm}{\frac{s^3 + (Te*Tm+Tc*Tm+Tc*Te)*s^2}{Tc*Te*Tm} + \frac{(Tm+Tc+Te)*s}{Tc*Te*Tm} + \frac{(1+Kc*Ke*Km)}{Tc*Te*Tm}}$$

Substituting the values of K_c, K_e, K_m, T_c, T_e and T_m the transfer function is:

$$\omega_{act} / \omega_{ref} = 2800000 / (s^3+526.67*s^2+13466.667*s+2866666.7) \quad (3.14)$$

The conventional controller is based on plant dynamics and rigorous mathematical models with linear or straightforward relationships between just few variables. The control action is to tune the controller parameters, but this cannot cope with the varying control environment or system non-linearities.

Facing these problems, the investigators realize that incorporating human intelligence into automatic control system would be more efficient solution and this leads to the development of Fuzzy Logic Controller.

3.5.2 Modeling and Simulation of FLC

The modeling and simulation of fuzzy control is carried out in **MATLAB**. The **MATLAB** is an interactive programmed for scientific and engineering calculations. The MATLAB family of programmed includes the base programmed plus variety of Toolboxes such as fuzzy, control, semolina etc.

The simulation of induction motor is done using transfer function analysis in MATLAB. The fuzzy logic controller is designed with the help of fuzzy logic toolbox. What makes the fuzzy logic toolbox so powerful is the fact that most of human reasoning and concept formulation is linked to the use of fuzzy rules.

There are five primary **Graphic User Interface** (GUI) tools for building, editing and observing fuzzy inference system in fuzzy logic toolbox. These are the fuzzy inference system or FIS Editor, the Membership Function Editor, the Rule Editor, the Rule Viewer and the Surface Viewer. These different GUI's are all effective siblings in that we can have any or all of them open for any given system. Standard blocks available in the SIMULINK Library the combined block diagram of Fig 3.9 is created for the plant, plant with PID & plant with FLC.

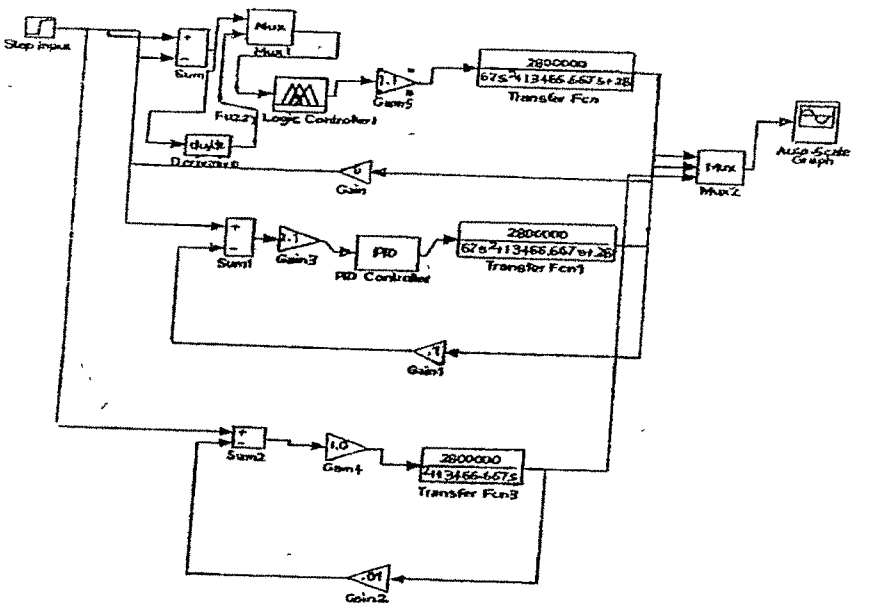


Fig 3.9: Combined Block Diagram; Plant, with PID and with FLC

3.5.3 FLC ALGORITHM & Pseudo code

```
# include < stdio.h>
# include < math.h>
# include < dos.h>
# include < conio.h>
# include < graphics.h>
```

```
# define port  0x300
# define base 0x2c0
main( )
{
float tab [11] [11] =  {{ -1,-1,-0.7,-0.7,-0.7,-0.6,-0.4,-0.3,-0.2,0,0},
                        { -1,-1,-0.7,-0.7,-0.7,-0.6,-0.5,-0.4,0.2,0.2},
                        { -1,-1,-0.7,-0.7,-0.7,-0.3,-0.2,-0.2,0.1,0.3,0.3},
                        { -1,-1,-0.7,-0.7,-0.7,-0.15,0.09,0.1,0.2,0.4,0.4},
                        {-1,-1,-0.6,-0.5,-0.5,-0.3,-0.1,0.1,0.2,0.4,0.4},
                        {-0.7,-0.7,-0.56,-0.5,-0.2,0,0.15,0.15,0.4,0.5,0.5},
                        {-0.5,-0.5,-0.3,0,0.05,0.09,0.2,0.3,0.5,0.8,0.8},
                        {-0.5,-0.5,-0.3,0,0.08,0.1,0.3,0.4,0.7,0.9,0.9},
                        {-0.2,-0.2,0.01,0.2,0.3,0.5,0.5,0.55,0.85,1,1},
                        {0.1,0.1,0.3,0.5,0.55,0.7,0.7,0.75,0.9,1,1},
                        {0.1,0.1,0.3,0.5,0.55,0.7,0.7,0.75,0.9,1,1},

                };
unsigned hb1, lbl, lb2, lb3,lb4,sts, al, a2;
unsigned hibyte = 0, lobyte = 0;
unsigned cntr, ccc, rrr;
float r,c, r1,c1,lbt,lbt1,lbt2,lbt3,AD[2], l, lbt1, lbt2;
int start = 0, stop=1, ch,rr,cc,l1,r2;
clrscr( );
outport  b (port + 9, 0x70);
while(1)
{
        for (ch = start; ch < = stop; ch++)
{
        outport b (port + 8 ,0); /*clear the interrupt */
        outport b (port + 2 ,0);/*mux channel selection on pc 1770 */
        outport b (port + 3 ,ch); /*select the ch 0-7 */
        outport b (port + 0 ,0); /* start  A/D conversion */
        start : sts = inportb (port + 8); /* Read A/D status */
        if (sts > 127) goto start;
        lobyte = inport b (port + 0); /* input A/D lo-wbyte */
        lb1 = (lobyte | 0xf0) / 16; / * separate channel no & low byte data */
        hibyte = inport b (port + 1) ; /* HIBYTE READ */
        lb 2 = (hibyte | 0xf) / 16; /*Shift the hibyte data*/
        lbt=lb1 + lb2;
        if (ch= =0)
        {
        lbt1 = lbt;
        AD[0]=lbt1;
        r1=((AD[0] - 2048) / 4096) /10
        }
        else
        {
        lbt 2 = lbt;
```

```
        AD[1] = lb2;
        c 1 = ((AD[1] - 2048) / 4096) /10;
        }
        /* printf ("%f /n", c1); */
        r = r1-c1;
        /* printf ("%f /n", r1); */
        r2 = r;
        a1 = r2/256;
        a2 = r2 % 256;
        outport b (base + 0, a1); /*ch #1*/
        outport b (base + 1 ,a2);
        outport b (port + 8,0); /*clear the interrupt */
        outport b (port + 2 ,0);/*mux channel selection on pc 1770 */
        outport b (port + 3 ,2); /*select the ch 0-7 */
        outport b (port + 0 ,0); /* start  A/D conversion */

start 1: sts = inportb (port + 8); /* Read A/D status */
        if (sts > 127) goto start1;
        lobyte = inport b (port + 0); /* input A/D lowbyte */
        lb3 = (lobyte | 0xf0) / 16; / * separate channel no & low byte data */
        hibyte = inport b (port + 1) ; /* HIBYTE READ */
        lb 4 = (hibyte | 0xf) / 16; /*Shift the hibyte data*/
        lbt3=lb3 + lb4;
        AD[2]=lbt3;
        c=((AD2[0] - 2048) / 4096) /10
         if (r = = 0)
        rr = 0;
        else
        {if (r > = 0 && r < = 0.5)
        rr = 1;
        else
        { if (r > = 0.5 && r < = 1)
        rr = 2;
        else
        { if (r > = 1 && r < = 1.5)
        rr = 3;
        else
        { if (r > = 1.5 && r < = 2)
        rr = 4;
        else
        { if (r > = 2 && r < = 2.5)
        rr = 5;
        else
        { if (r > = 2.5 && r < = 3)
        rr = 6;
        else
        { if (r > = 3 && r < = 3.5)
        rr = 7;
```

```
else
{ if (r >= 3.5 && r <= 4)
rr = 8;
else
{ if (r >= 4 && r <= 4.5)
rr = 9;
else
{ if (r >= 4.5 && r <= 5)
rr = 10;
}
}
}
}
}
}
}
}
}
if (c == 0)
cc = 0;
else
{if (c >= 0 && c <= 0.5)
cc = 1;
else
{ if (c >= 0.5 && c <= 1)
cc = 2;
else
{ if (c >= 1 && c<= 1.5)
cc = 3;
else
{ if (c >= 1.5 && c <= 2)
cc = 4;
else
{ if (c >= 2 && c <= 2.5)
cc = 5;
else
{ if (c >= 2.5 && c <= 3)
cc = 6;
else
{ if (c >= 3 && c <= 3.5)
cc = 7;
else
{ if (c >= 3.5 && c <= 4)
cc = 8;
else
{ if (c >= 4 && c <= 4.5)
cc = 9;
```

```
else
{ if (c > = 4.5 && c < = 5)
cc = 10;
}
}
}
}
}
}
}
}
}
}
rrr = rr;
ccc=cc;
l = ((tab [rrr][ccc] / 5) * 2048) + 2048;
l1 = ( l * 5);
printf ("%d %d \n",rrr,ccc);
a1=l1 / 256;

a2 = l1 % 256;
outport b (base + 0,a1); /*ch#1) */
 outport b (base + 1,a2);
}
if ( kbhit( )) break;
}
}
```

3.5.4 Simulation Setup

The parameters for these fuzzy sets are as under:

NVL	=	[-1.2 -1 -0.784]
NL	=	[-1 -0.784 -0.568]
NM	=	[-0.784 -0.568 -0.316]
NS	=	[-0.568 -0.316 -0.1]
NVS	=	[-0.316 -0.1 0]
ZE	=	[-0.1 0 0.1]
PVS	=	[0 0.1035 0.345]
PS	=	[0.1035 0.345 0.552]
PM	=	[0.345 0.552 0.7935]
PL	=	[0.552 0.7935 1]
PVL	=	[0.7935 1 1.2]

Table 3.6 depicts the Rule Base

e ce	NVL	NL	NM	NS	NVS	ZE	PVS	PS	PM	PL	PV L
NVL	NVL	NVL	NM	NVL	NVL	NVL	NM	NM	NS	ZE	ZE
NL	NVL	NVL	NM	NVL	NVL	NL	NM	NM	NS	ZE	ZE
NM	NVL	NVL	NVL	NL	NL	NM	NS	NS	ZE	PS	PS
NS	NVL	NVL	NVL	NM	NM	NS	ZE	ZE	PS	PM	PM
NVS	NL	NL	NL	NM	NM	NVS	ZE	ZE	PS	PM	PM
ZE	NVL	NL	NM	NS	NVS	ZE	PVS	PS	PM	PL	PV L
PVS	NM	NM	NS	ZE	ZE	PVS	PS	PS	PL	PVL	PV L
PS	NM	NM	NS	ZE	ZE	PS	PM	PM	PL	PVL	PV L
PM	NS	NS	ZE	PS	PS	PM	PL	PL	PVL	PVL	PV L
PL	ZE	ZE	PS	PM	PM	PL	PVL	PVL	PVL	PVL	PV L
PVL	ZE	ZE	PS	PM	PM	PL	PVL	PVL	PVL	PVL	PV L

Table 3.6: RULE BASE FOR FUZZY LOGIC CONTROLLER

- **Simulation Steps**

1. From the source library, step input (with step time 0, initial value, final value1) is selected which is input to the plant.

2. Linear block library the blocks sum, derivative, gain, transfer function and state space are selected. In gain block, we can adjust the desired gain in forward as well as feedback path. The transfer function block allows us to modify the transfer function according to our system. The state space block reads the ABCD parameters for the IM model developed.

3. FLC block is selected from semolina. While running simulation, make sure that the FIS matrix corresponding to the fuzzy system used, is saved in both MATLAB workspace and referred to by name in the dialog box associated with The fuzzy logic controller block. Also a PID controller block is selected for the Design of PID controller.

4. Connection library allows choosing the block such as OUT, MUX. The MUX block is selected in which the inputs can be varied according to the requirement.

5. Sink Library: From this library Auto scale graph is selected for observing simulation results. The Auto scale graph has initial time range of 5, initial y-min of -10, initial Y-max of 10 and storage points of 200.The Simulation results forfeedback gain of 0.1 are as shown in Fig 3.10 The effect of variation of gain is shown studied. Results are shown in Fig 3.11, 3.12 & 3.13 respectively. It may be concluded that FLC improves the dynamic performance of the plant.

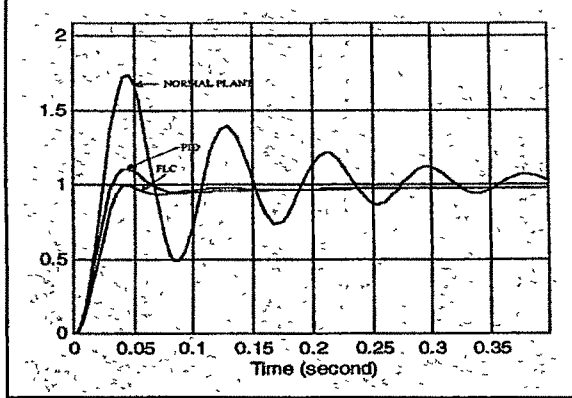


Fig 3.10: Feedback Gain (FBG) = 0.01

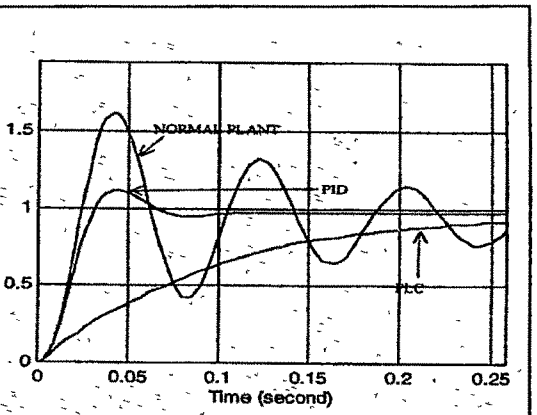


Fig 3.11: FLC = 1.4, PID = 1.1 FBG = 0.1

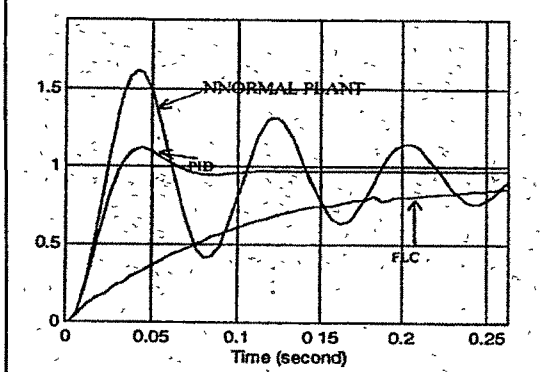


Fig 3.12: FLC = 1.3, PID = 1.1 FBG = 0.1

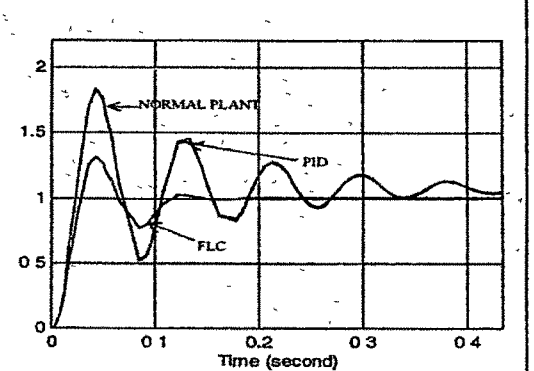


Fig 3.13: FFG = 1.1, FBG = 0.001

(6). The simulation is also carried out using the look-up table (Table 3.7(a),)b) prepared on the basis of Rule Base, for the block diagram shown in Fig 3.14.

S	SS
-0.2500	-0.2500
-0.2000	-0.2000
-0.1500	-0.1500
-0.1000	-0.1000
-0.0500	-0.0500
0	0
+0.0500	+0.0500
+0.1000	+0.1000
+0.1500	+0.1500
+0.2000	+0.2000
+0.2500	+0.2500

Table 3.7(a) : Lookup Table for FLC

A										
-1 0000	-1.0000	-0 7000	-0.7000	-0 7000	-0.6000	-0.5000	-0 5000	-0.4000	0.2000	0 2000
-1 0000	-1.0000	-0.7000	-0.7000	-0.7000	-0.6000	-0 5000	-0 5000	-0.4000	0.2000	0.2000
-1.0000	-1.0000	-0.7000	-0.7000	-0.7000	-0 3000	-0 2000	-0.2000	0.1000	0.3000	0.3000
-1.0000	-1.0000	-0.7000	-0 7000	-0 7000	-0 1500	-0.0900	0 1000	0.2000	0 4000	0.4000
-1 0000	-1.0000	-0 6000	-0 5000	-0.5000	-0 3000	-0.1000	0.1000	0.2000	0.4000	0.4000
-0 7000	-0.7000	-0.5600	-0.5000	0.2000	0	0 1500	0.1500	0.4000	0 5000	0.5000
-0.5000	-0 5000	-0 3000	0	0.0500	0 0900	0.2000	0.3000	0 5000	0.8000	0.8000
-0 5000	-0.5000	-0 3000	0	0 0800	0.1000	0.3000	0.4000	0.7000	0 9000	0.9000
-0 2000	-0.2000	0.0100	0 2000	0 3000	0.5000	0 5000	0.5500	0.8500	1.0000	1.0000
0.1000	0.1000	0 3000	0.5000	0.5500	0.7000	0 7000	0.7500	0 9000	1.0000	1.0000
0.1000	0.1000	0 3000	0.5000	0.5500	0 7000	0.7000	0.7500	0.9000	1 0000	1.0000

Table 3.7(b) : Lookup Table for FLC

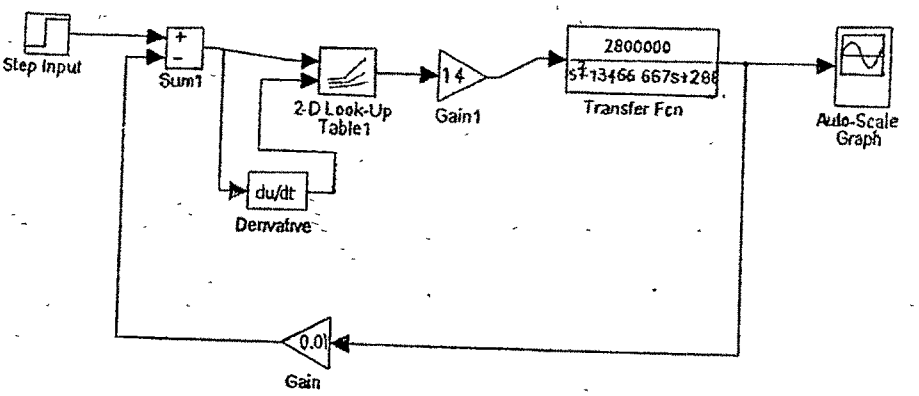


Fig 3.14: SIMULINK setup for Lookup Table based FCL

The output response for step input is shown in Fig 3.15. The controller surface is presented in Fig 3.16.

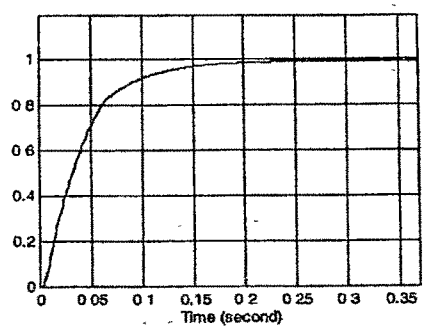


Fig 3.15: Step response for Lookup Table FLC

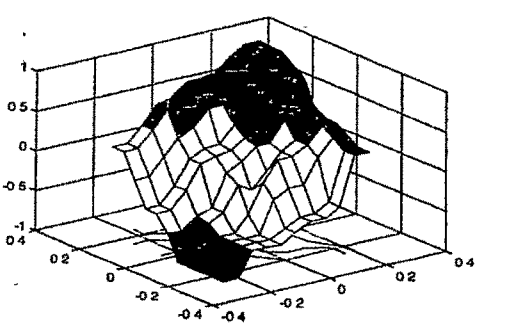


Fig 3.16: Controller Surface view for Lookup Table FLC

3.6 NEURO-FUZZY CONTROLLER

Combining the learning power of neural network with knowledge representation of fuzzy-logic gives Neuro-Fuzzy (NF) systems. Neuro-Fuzzy software tools work as an intelligent assistant to design. It helps to generate and optimize membership function as well as rule base from the simple data.Neuro-Fuzzy controller (NFC) design is done using ANFIS (Adaptive Neuro-Fuzzy Inference System). ANFIS is about taking a fuzzy inference system (FIS) and tuning it with a back propagation algorithm based on some collection of I/P-O/P data. This allows the fuzzy systems to learn. ANFIS supports only Sugeno systems subject to the following constraints:

* First order Sugeno type Systems
* Single O/P derived by weighted average defuzzification
* Unity weight for each rule

An error occurs if FIS matrix for ANFIS learning does not comply with these constraints. Moreover, ANFIS is highly specialized for speed and cannot accept all the customization options that basic fuzzy inference allows, i.e. one cannot make his own MF and defuzzification functions.

- **ANFIS LEARNING** NFC design is imparted using ANFIS. To start ANFIS learning, training data set that contains desired I/P-O/P data pairs of the target system to be modeled. The target is decided based on the ideal response of the system under unit step input. When the IM is in running condition, parameter variation causes drift in the system response. This may be due to the changing rotor/stator resistance due to load change. A damped response in O/P for any step change in me /P is required. In ideal condition, the O/P response for step I/P is (1-e^-a*t).Thus damped response is obtained by optimizing the value of 'a'. The O/P response for the system is taken as:

  y = [1-e^ -5.67*x]

- Program in MATLAB for ANFIS training is given below:

```
>> mumps =51;
   x = linspace (0,1,numPts)';
   z = linspace (0,1,numPts)';
   y = [1-(exp(-5.67*x))];

   plot(z,y);
   pause
   plot(z,y);
   pause
   data=[x z y];
   trnData1=data(1:3:numPts,:);
   chkData1=data(3:3:numPts,:);
   plot(trnData1(:,1),trnData1(:,3), '0', chkData1(:,1),chkData1(:,3), 'x');
   pause;
```

```
trnData2=data(2:3:numPts,:);
chkData2=data(8:3:numPts,:);
plot(trnData2(:,2),chkData2(:,3),'o',....
    (chkData2(:,2),chkData2(:,3),'x')
numMFs=[7,7];
mfType=str2mat('trimf');
fismat=genfis1(data,numMFs,mfType);
[x,mf1]=plotmf(fismat,'input',1);
plot(x,mf1);
pause
[x,mf2]=plotmf(fismat,'input',2);
plot(x,mf2);
pause
numEpochs=20;
[fismat1,trnErr1,ss,fismat2,chkErr1] = ...
anfis(trnData1,fismat,numEpochs,NaN,chkData1);
[fismat2,trnErr2,ss,fismat2,chkErr2] = ...
anfis(trnData2,fismat,numEpochs,NaN,chkData2);
trnOut1=evalfis(trnData1(:,1),fismat1);
trnRMSE1=norm(trnOut1 - trnData1(:,3)) / sqrt(length(trnOut1));
trnOut2=evalfis(trnData2(:,2),fismat1);
trnRMSE2=norm(trnOut2 - trnData2(:,3)) / sqrt(length(trnOut2));
epoch = 1:numEpochs;
plot(epoch,trnErr1, 'o',epoch,chkErr1, 'x')
pause
plot(epoch,trnErr2, 'o',epoch,chkErr2, 'x')
pause
hold on;
plot(epoch,[trnErr1 chkErr1] );
pause
plot(epoch,[trnErr2 chkErr2] );
pause
hold off
pause
plot(epoch,ss,'*',epoch,ss,'x')
pause
[x mf]=plotmf(fismat1,'input',1);
plot(x,mf)
pause
[x,mf]=plotmf(fismat1,'input',2);
plot (x,mf)
pause
anfis_y = evalfis(x,fismat1);
plot(trnData1(:,1),trnData1(:,3), 'o',....
     chkData1(:,1),chkData1(:,3), 'x',....
     x,anfis_y, '-');
pause
```

```
anfis_y = evalfis(x,fismat1);
plot(trnData2(:,2),trnData2(:,3), 'o',....
     chkData2(:,2),chkData2(:,3), 'x',....
      x,anfis_y, '-');
```

The design parameters of ANFIS are as follows:

1. Number of total data pairs are decided. (51)
2. Training data set and checking data set are defined
3. Fuzzy inference system for training is specified. (For our system MF= 7 and MF type = triangular)
4. Number of epochs are chosen to be 20 to start the training.
5. Learning results are verified and Root Mean Square Error (RMSE) is obtained.
6. Step size is mentioned.
7. Final MF are plotted
8. FIS O/P is plotted.

- **SIMULATION RESULTS:** Fig 3.17 depicts complete ANFIS learning and modified MF, while Fig 3.18 shows plot of output response.

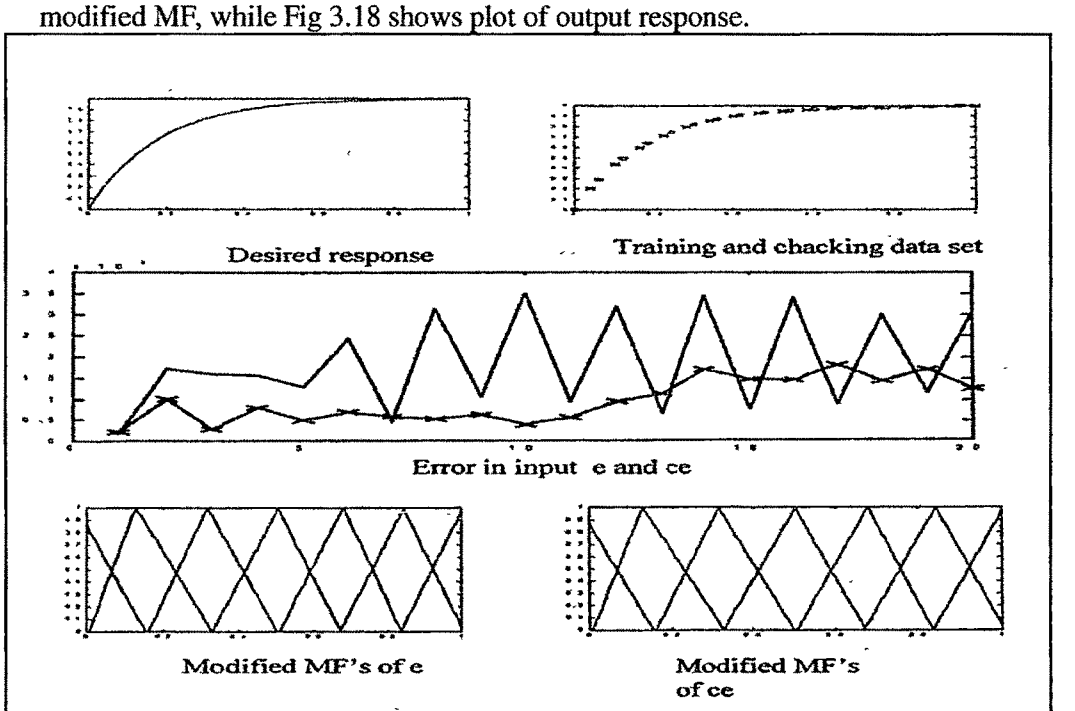


Fig 3.17: ANFIS Training

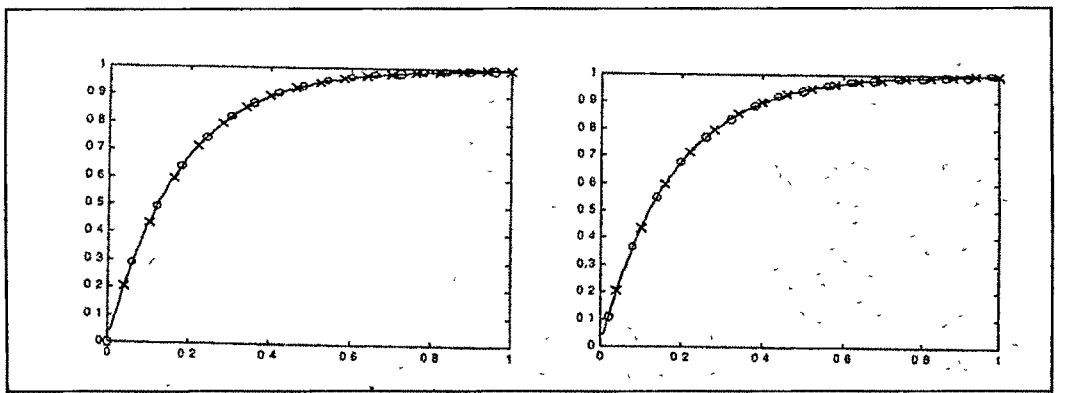


Fig 3.18:Output Response of NFC for input 1 & 2

The simulation results and ANFIS information are given below. ANFIS training completed after 20 epochs for error gives:

* Minimal training RMSE = 0.000002
* Minimal checking RMSE = 0.000164696

For change of error it is:

* Minimal training RMSE = 0.000001
* Minimal checking RMSE = 0.000222164

• EXPERIMENTAL SET-UP

The proposed PC based experimental set-up is shown in Fig 3.19 The details of the experimental set-up are as follows:

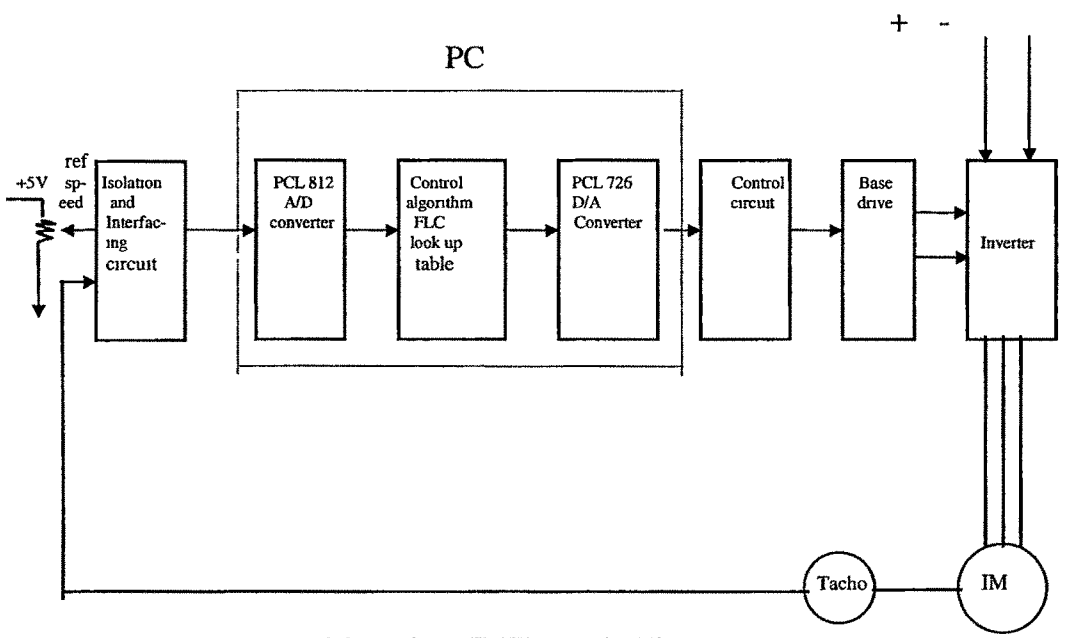


Fig 3.19: EXPERIMENTAL SET-UP

1. **Reference signal for speed:** 0-5V DC reference signal corresponding to zero to rated speed is generated from regulated power supply. This voltage is varied by using preset (10KΩ).
2. **Feedback Signal:** Feedback signal from tacho-generator (AC Tacho) is converted to DC signal by rectifier. This is scaled to 5V DC corresponding to rated speed (1500 rpm). This is achieved through the proper resistance divider.
3. **Interfacing Circuit:** These analogue signals are fed to PC. Before connecting them to PC they are isolated and normalized by using interfacing card. This is shown in Figure-18.
4. **A/D and D/add-on cards:** A/D conversion is obtained by using PCL 812 card. D/A conversion are obtained by using PCL 726 card. These are add-on cards placed inside the PC.
5. **Control Algorithm:** An algorithm for FLC is build up as given in Appendix-3. The inputs error (e) and change of error (ce) is given through this algorithm. A look up table is prepared which is scanning error & change of error and according to the rules; output is obtained in the form of crisp value.
6. **Control Circuit:** The output of the controller through PCL 726 is interfaced to control circuit. This consists of PWM control circuit. This generates the PWM signals for the inverter.
7. **Base Drive:** Base drive for IGBT inverter is used to interface the control signals to inverter. This is based on opto- isolation technique. This circuit is hardwired to main IGBT's used for inverter.
8. **Inverter :** Bridge type IGBT inverter is built for AC motor drive. Output of the inverter is varied so as to achieve the speed control. A V/f technique as mentioned in the Figure-1 is used .

This complete scheme may be tested with the fuzzy logic based speed controller.

3.7 Compensation of Dead-zone Non-linearity

The conventional controller [13] is based on plant dynamics and rigorous mathematical models with linear as well as non-linear variables. The control action is to tune the controller parameters, but this cannot cope with the varying control environment or system non-linearities. Facing these problems, the investigators realize that incorporating human intelligence into automatic control system would be more efficient solution and this leads to the development of Fuzzy Logic Controller. Due to the non-linear component added in the system Fuzzy Logic Pre compensator is also made which leads to the better response of the system.

The pre compensator [14] is made in order to reduce the effects of the non-linearity like dead-zone, backlash, hysterisis. The Fuzzy logic pre compensator is designed for the dead-zone. The model for this is shown in the following section. The power pf Fuzzy logic systems is that they allow one to use intuition based on experience to assign control systems, then provide the mathematical machinery for rigorous analysis and modification of the intuitive knowledge, through learning or adaptation to give guaranteed performance. The FL pre compensator [15] effectively provides a pre inverse of the dead-zone. FL techniques are in terms of membership functions, which are needed for compensation of the non-linear mechanical systems.

The fuzzy logic [16] pre compensator is designed in order to tune the error such that the error will be as small as possible. This tunes the dead-zone pre compensator into an adaptive fuzzy logic compensator. To create a new fuzzy inference system start the FIS Editor by entering >> **fuzzy** at MATLAB prompt Two inputs are defined as error (e) and change of error (ce).The output is the control input(u) to the motor. The output which is the control input can be represented as: u = f(e, ce) for a non-linear function f.

To define the membership functions associated open MBF editor (Fig 3.20) by pulling down the View Menu item and selecting Edit Membership Functions. The MBF for error(e), change of error(ce) and output(u) are all defined within the range [-1,1]. The ranges are nothing but normalized domains [universe of Discourse] which is required for scale transformation which maps the physical values of process state variables into normalized domain. The choice of membership function is done taking into account the output response of the system.

The characteristics of a good response is low setting time, less overshoot, low steady state error etc. Thus different types of membership functions are tried & the best suitable response if found to be of triangular membership function. The shape of the membership functions for both inputs and single output is chosen as triangular.

The number of MF's for both input and output variables are chosen to be eleven. We choose eleven fuzzy sets that are specified on the domains of 'e' and 'ce'. These are NVL, NL, NM, NS, NVS, ZE, PVS, PS, PM, PL and PVL corresponding respectively to negative very large, negative large, negative medium, negative small, negative very small, zero error, positive very small, positive small, positive medium, positive large and positive very large. In the same manner, the eleven fuzzy sets are defined on the domain of definition of the output u.

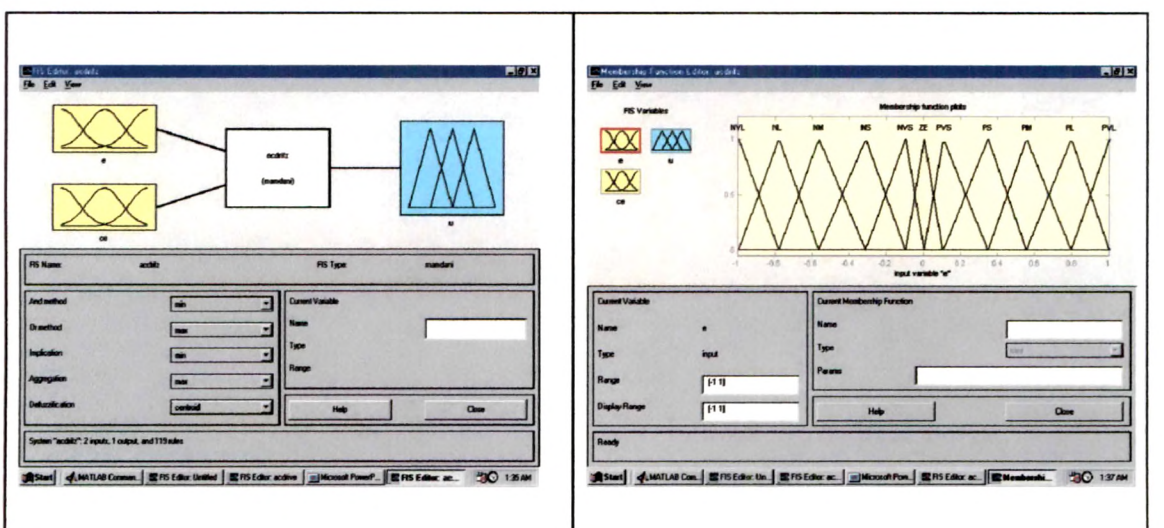


Fig 3.20(a) Membership FunctionEditor

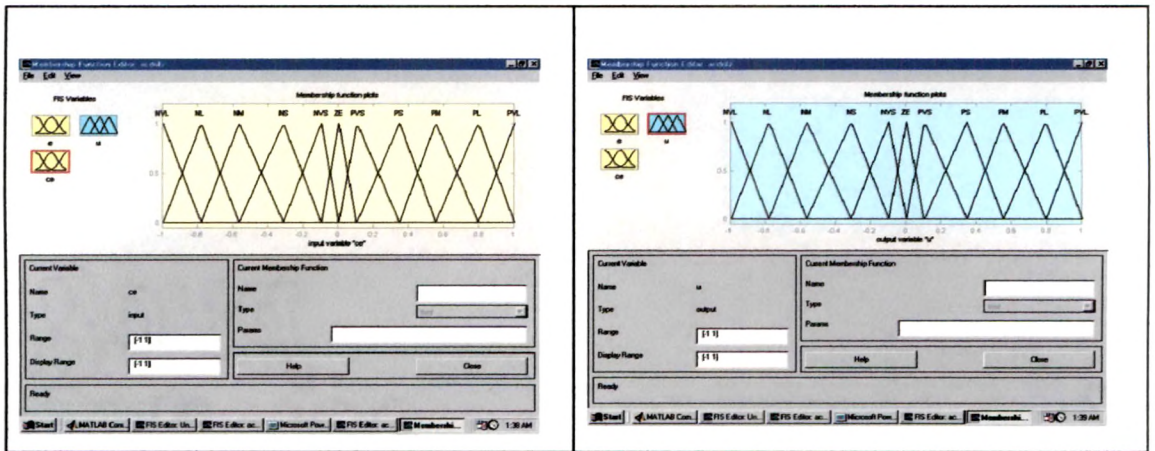


Fig 3.20(b) Membership Function Editor

The parameters for these fuzzy sets are as under:

NVL	=	[-1.2 -1 -0.784]
NL	=	[-1 -0.784 -0.568]
NM	=	[-0.784 -0.568 -0.316]
NS	=	[-0.568 -0.316 -0.1]
NVS	=	[-0.316 -0.1 0]
ZE	=	[-0.1 0 0.1]
PVS	=	[0 0.1035 0.345]
PS	=	[0.1035 0.345 0.552]
PM	=	[0.345 0.552 0.7935]
PL	=	[0.552 0.7935 1]
PVL	=	[0.7935 1 1.2]

3.7.1 FLC DESIGN

- **DC MOTOR SPECIFICATIONS**

Kc	=	1.57
Ke	=	2
Km	=	5
Tc	=	0.01
Te	=	0.06
Tm	=	0.15
Kt	=	1

The values of K_c, K_e, K_m, T_c, T_e and T_m leads to the transfer function of the motor as:

$\omega_{act} / \omega_{ref} = 174444.44 / (s^3+122.22*s^2+2444.44*s+11111.11)$

To activate the Rule Editor, go to the view menu and select Edit rules. Rule editor contains a large editable text for displaying and editing rules. Since, there are eleven membership functions for error and change of error, the maximum possible rules can be written are 121.A total number of 119 rules are formed to give robust performance of the controller. A portion of rule base is shown in fig 3.21

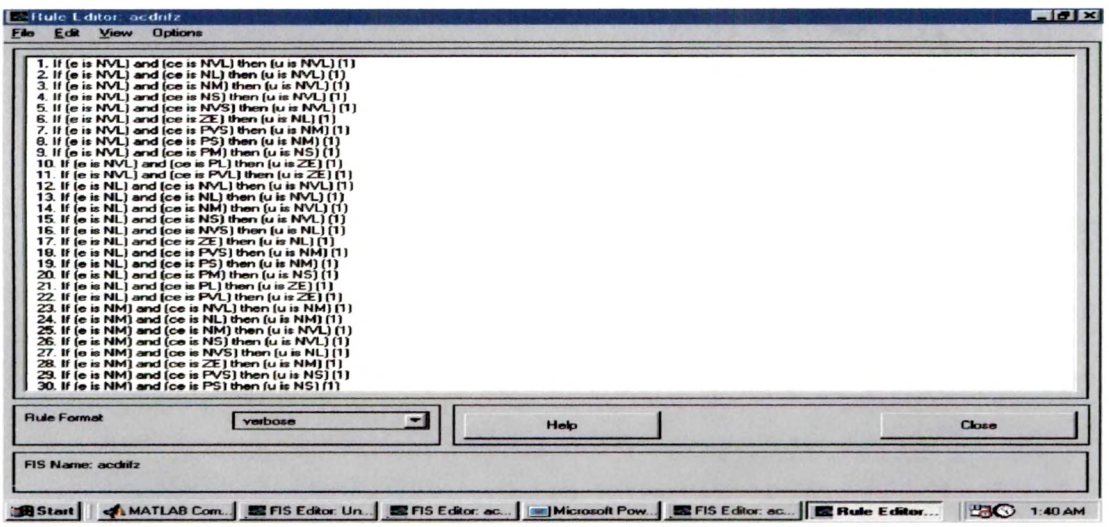


Fig: 3.21 Rule Base Editor for Creation of a Rule Base

The linguistic rules of error with negative sign mean that the current process output has a value below the steady state value. On the other hand linguistic values of error with a positive sign means that current output is above the steady state value. The linguistic values of change of error with negative sign means that the output has increased when compared to its previous value. The magnitude of such a negative value is given by the magnitude of this increase. Linguistic values of Ce with a positive sign mean the output has decreased its value when compared with its previous value. The magnitude of such a value is the magnitude of decrease.

A linguistic value of zero for error means that the current process output is at steady state. A zero for change of error means that the current process output has not changed from its previous value. The rule can be entered in verbose or symbolic or index fashion. The rules written are shown in Rule Editor, which can be parsed by pressing ctrl-

enter. To view the rules in Rule Viewer, select View from the View Menu. Select the blocks from the semolina library, the block diagram for study is created. Fig-3.22 indicates the combined block diagram of plant, plant with PID & plant with FLC.

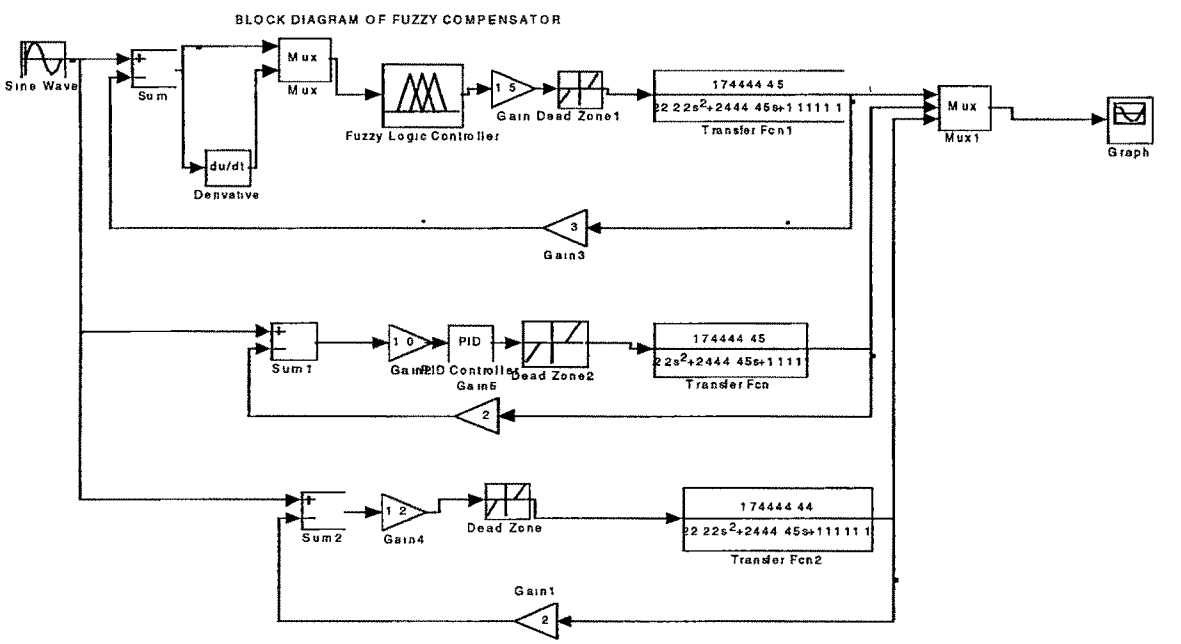


Fig: 3.22 Combined SIMULINK set up including dead-zone for Plant, Plant with PID and Plant with FLC

3.8.2 RUNNING the SIMULATION

1. From the source library, step input (with step time 0, initial value 0, final value 1) is selected which is input to the plant. From the source library, sinusoidal input is taken whose amplitude is 0.5 and frequency 2Hertz/sec and sample time 0 phase 0 is selected and is given to input of the plant with pre compensator and dead-zone nonlinearity.

2. Linear block library has various blocks such as sum, product, derivative, integration, gain, and transfer function, zero-pole and state space. For our study the blocks selected are sum, derivative, gain, transfer function and state space. In gain block, we can adjust the desired gain in forward as well as feedback path. The transfer function block allows us to modify the transfer function according to our wish.

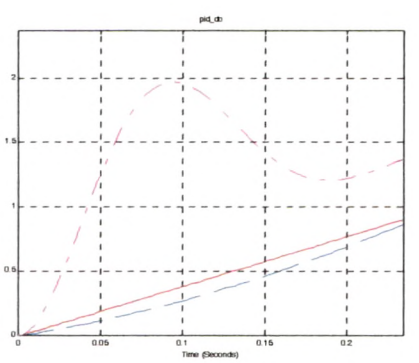
3. Dead-zone is selected from the nonlinearity from the main library of the simulink.

4. FLC block is selected from simulink.While running simulation, make sure that the FIS matrix corresponding to the fuzzy system used, is saved in

both MATLAB workspace and referred to by name in the dialog box associated with the fuzzy logic controller block. Also a PID controller block is selected for the design of PID controller.

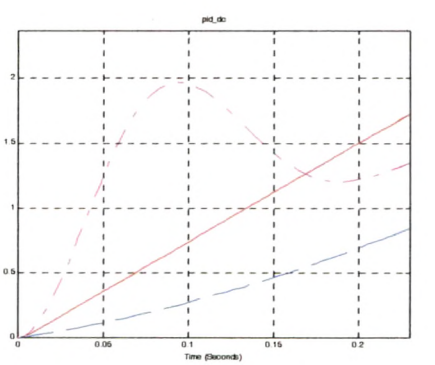
5. Connection library allows choosing the block such as OUT, MUX. The MUX block is selected in which the inputs can be varied according to the requirement. Also the signs are changed as per the need.

6. Sink Library: From this library Auto scale graph is selected for observing the simulation results. The Auto scale graph has initial time range of 5,initial y-min of -10,initial y-max of 10 and storage points of 200.The Simulation results are as shown in Fig 3.23(i) for a feedback gain equal to 0.01. Also results are added for different gains as shown in Fig 3.23(ii), (iii) and (iv) respectively.



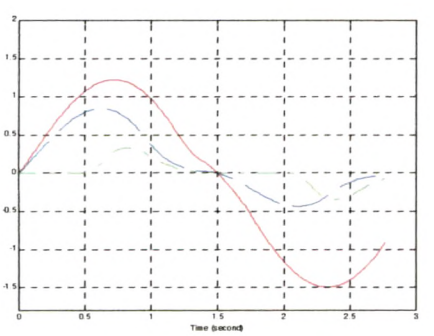
Comparative Response
Plant, Plant with PID, Plant with FLC

Fig 3.23 (i)



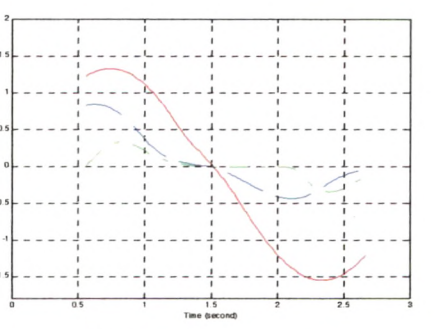
FFG of FLC=1.57 , FBG=.1
FFG of PID=1.2, FBG =.1

Fig 3.23 (ii)



Comparative Response
Plant with dead zone, PID with dead zone
Adaptive fuzzy compensator

Fig 3.23 (iii)



Comparative response
FBG of fuzzy compensator =0.1
Dead zone=-0.4 ,0.4

Fig 3.23 (iv)

It may be concluded from figure that FLC improves the dynamic performance of the plant.

The conventional controller is based on plant dynamics and rigorous mathematical models with linear as well as non-linear variables. The control action is to tune the controller parameters, but this cannot cope with the varying control environment or system non-linearities. Facing these problems, the investigators realise that incorporating human intelligence into automatic control system would be more efficient solution and this leads to the development of Fuzzy Logic Controller. Due to the non-linear component added in the system Fuzzy Logic Pre compensator is also made which leads to the better response of the system.

3.8 Adaptive Fuzzy Logic Compensator for Backlash

Fuzzy Logic (FL) systems have several major properties that make them useful in feedback control, including the function approximation property, the classification property, ability to select initial parameter values based on sound control engineering experience and ability to tune the parameters adaptively to yield guaranteed closed-loop performance.

- Function approximation property of fuzzy system: For the fuzzy system it has been shown in various research papers that the fuzzy system functions provides a basis for continuous function if the membership function and rule are properly chosen. This justifies their usual name of fuzzy basis function (FBF), and implies a universal approximation result for FL system. The uniformly spaced triangular MBF suffice for any smooth f (w) if the number of MF N_j for each component w_j is selected large enough.

- Classification property of fuzzy system: Implicit in the definition and the philosophy of FL systems is a classification property. Each component w_j of the input is classified as belonging to some MBF X_1^{ij} , depending on the region within which w_j fallswith in the region of support of $X_j^{ij}(w_j)$.

  In feedback control application, this allows a very convenient technique for defining different control methods actions depending on different regions of w. No analytic actuator nonlinearities such as the backlash have different effects depending on the region within which the argument lies, so that FL systems seem very natural in compensating for them. The FL approach thus subsumes other approaches based on switching logic and indicator functions.

In this section, a fuzzy pre compensator [16] is designed for symmetric backlash non linearities in actuation of system in the class.

3.8.1 Backlash non linearities

If u and τ are scalars, the symmetric backlash non linearity may be described as:

$$\tau = B_b(u) = \begin{cases} u - b/2 & u < b_- \\ U - b/2 & b_- < u < b_-/2 \\ u + b/2 & b_-/2 < u < -b_+ \\ U + b/2 & -b_+/2 < u < -b_+ \\ u - b/2 & u = -b_+ \end{cases} \qquad (3.15)$$

The parameter vector $b = [b_-/2\ b_+/2]$ characterizes the width of the motion backlash. In practical motion control system, since the width of backlash is unknown, the compensation is difficult. Most compensation techniques covers the case of symmetric backlash where $b_-/2 = b_+/2$.

- **FL compensation of multi-input systems with output backlash**

To offset the deleterious effect of backlash, one may place a pre compensator there the desired function of the pre compensator is to composite throughout from ω to τ to be unity. The power of fuzzy logic is that to allow one to use intuition, based on experience to design control system and then provide the mathematical machinery for rigorous analysis and modification of the intuitive knowledge, for the example through learning or adoption, to give the guaranteed performance due to classification property of FL, they are very powerful particularly when the non linearity depends on the region in which the argument u of the non linearity is located, as in the non symmetric backlash.

In most practical motion control systems, there are several control inputs so that ω, u , τ are generally u vectors. There may be different backlash characteristics in each channel so that for i = 1,2,....,n for each components ω_i , u_i , τ_i one has a symmetric backlash.

$$\tau_i = B_{bi}(u_i) = u_i - \text{Sat } b_i(u_i) \qquad (3.16)$$

with $b_i = [\ b_-/2\ b_+/2\ ]^T$, one can write this in a vector form as

$$\tau_i = B_b(u) = u - \text{Sat } b(u) \qquad (3.17)$$

where the block diagonal matrix diag($b_1/2\ b_2/2 \ldots$) $\in R^{2n \times n}$, the vector saturation function is defined as

$$\text{Sat } B(u) = [\ \text{Sat } B_i(u_i)\ ] \qquad (3.18)$$

where [z_i] denotes the vector with component z_i. Then one must use a fuzzy logic (FL) compensator for each channel.

The backlash pre compensator, designed using engineering experience would be discontinuous and depend on the region with in which ω occurs. It would naturally be described only the rules

If ($w_i \in X_+(w_i)$) then ($w_{Fi} = B_{i+}$)

If ($w_i \in X_-(w_i)$) then ($w_{Fi} = -B_{i-}$) (3.19)

to accomplish this, define

$$u_i = w_i + w_{Fi} \quad (3.20)$$

with MF's $X_+(.)$, $X_-(.)$ defined for each component according to the following :

$$X_+(w_i) = \begin{cases} 0 , w_i < 0 \\ 1 , 0 = w_i \end{cases}$$

$$X_-(w_i) = \begin{cases} 1 , w_i < 0 \\ 0 , 0 = w_i \end{cases} \quad (3.21)$$

which are shown in fig-3.

Define the estimate vector $B_i = [\, B_{i+}/2 \; B_i/2 \,]^T$.The fuzzy logic pre compensator may be conveniently expressed in vector form as follows.

Define the vector $W_F = [\, W_{F1}, W_{F2}, \ldots W_{FN} \,]^T$ so that

$$u = w + w_f$$

$$u = w + B^T x\,(w) \quad (3.22)$$

where the block diagonal matrix of estimated backlash width is

$B = \text{diag} [\, b_1 \, b_2 \, b_3 \ldots b_n \,]$ and

the vector fuzzy logic based function is given by

$$X\,(w) = [X_+(w_1) \; -X_-(w_1) \;\; X_+(w_2) \;\; -X_-(w_2) \;\; \ldots\ldots \; X_+(w_n) \; -X_-(w_n)]^T \quad (3.23]$$

The throughput of the compensator plus backlash for vector ω, u , $\tau \in R^n$ is

$$\tau = w - B^T x\,(w) + B^T \delta \quad (3.24)$$

where the matrix backlash width estimation is $\tilde{B} = B - \hat{B}$

$= \text{diag} [\, b_1 \, b_2 \, b_3 \ldots b_n \,]$.

The vector mismatch δ has a component δ_i and satisfies

$$\| \delta \| = vn \quad (3.25)$$

Tuning or learning of backlash width estimation B(k) is done on line. So that the tracking error is guaranteed mnimum and all internal states are bounded. This will make discrete time adaptive FL backlash compensator. In this estimation it is assumed that the actuator output $\tau(k)$ is not measurable. It is important to note here that the discrete time FL backlash compensation signal w(k) is injected exactly at the same instant at which standard dithering, signals are injected. Thus, this backlash compensation scheme could be considered as adaptive dithering.

3.8.2 RUNNING the SIMULATION

The adaptive FL compensator for output backlash compensation is simulated on a digital computer using MATLAB Simulink toolbox. It is proved to be very efficient in nullifying the effect of output backlash.

We simulate the response for the known transfer function model of DC motor with output backlash, with and without FL compensator. The result taken are comparative for the same plant with uniform output backlash, using PID controller, with no controller and with FL controller. The DC motor transfer function considered in this particular case study is

$$0.2481\ [z/(z-e^{13.01T}) - z/(z-e^{6.98T})] \qquad (3.26)$$

Taking sampling time T=0.1 sec, the system transfer function reduces to,

$$0.41\ z\ /\ z^2 - 5.68\ z + 7.38 \qquad (3.27)$$

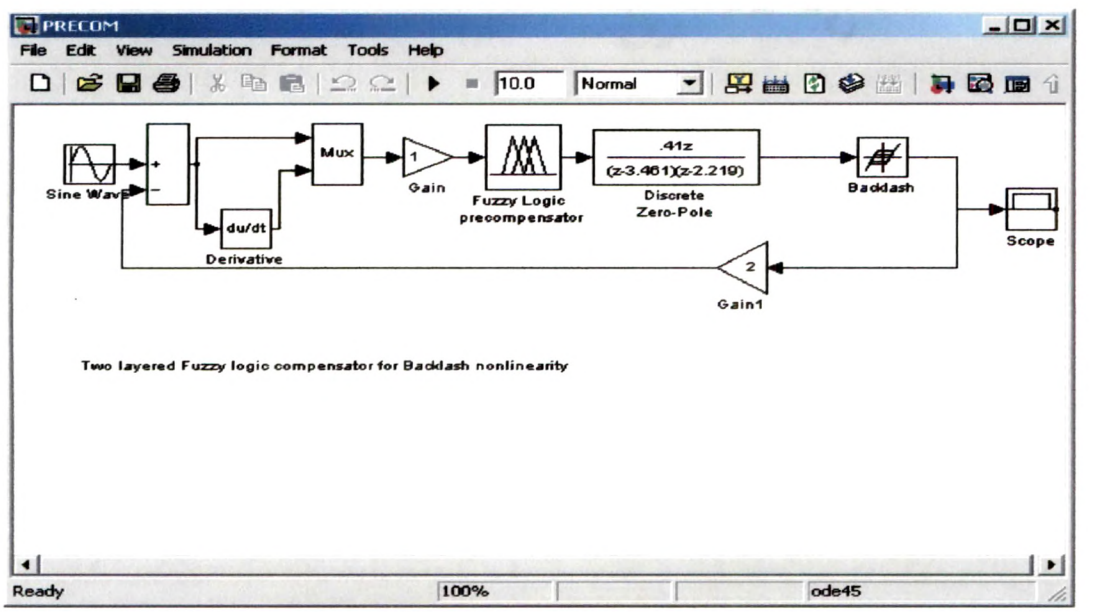


Fig 3.24 FLC Based Compensator

3.8.3 Two-Layered Adaptive Fuzzy Logic Controller

The direct use of FLC to a system having Backlash nonlinearities results in poor transient and steady state performance. Paper presents a fuzzy logic based scheme which eliminates above listed problem. Our control scheme consists of two layers: a fuzzy pre compensator and a usual fuzzy logic controller. Proposed system shows better performance as compared to normal one layer approach. Two layer fuzzy logic controllers consist of two fuzzy logic compensator:

1. Conventional Fuzzy Logic Controller
2. Fuzzy pre compensator

- **Conventional Fuzzy Logic Controller:** The fuzzy logic controller is usually designed on fuzzy lgic control law. In this paper the FL law taken is F[e(k),Δe(k)]. In that e(k) and Δe(k) are two input given to the controller and F[e(k),Δe(k)] is one output of the FLC. The two inputs considered are output error $e(k) = y_m(k)-y_p(k)$ and second input is change in output error $\Delta e(k)= e(k)-e(k-1)$. In the design of the FLC set of linguistic variable values and

membership functions are considered. In this FLC set of linguistic values L is considered and are

L = { NB,NM,NS,ZO,PS,PM,PB} and set of membership function (MBF) is considered as : M= { M_{NB},M_{NM},M_{NS},M_{ZO},M_{PS},M_{PM},M_{PB} } and is a real line to the interval [0,1]. The meaning of the linguistic variable values are clear from their mnemonic and is taken as

First character	Second Character
N - Negative	B- Big
ZO- Zero	M- Medium
P - Positive	S- Small

In standard fuzzy logic controller it consists of three stages i.e
fuzzification,
decision making fuzzy logic and defuzzification.

Fuzzification: It transforms the inputs e(k) and Δe(k) into the setting of the linguistic values. For each linguistic value $l \in L$, a pair of Mf values $n_e(l)$ and $\Delta n_e(l)$ to the inputs e(k) and Δe(k) using associated MF. The association is given as

$$n_e(l) = M_l(C_e e(k))$$
$$\Delta n_e(l) = M_l(C_{\Delta e}\Delta e(k)) \quad (3.28)$$

where: C_e and $C_{\Delta e}$ are scale factor and $n_e(l)$ and $\Delta n_e(l)$ are used in fuzzy logic decision process.

Decision making Fuzzy Logic: Using linguistic variable values and MF numbers of rules can be associated with the fuzzy logic controller. The expertise and available knowledge is utilized to build these rules. Trial and error method is also implemented. In our case fuzzy rules R= {R_1,R_2,R_3 ... R_r}.

Each rule in rule base set R is a triplet { l_e, $l_{\Delta e}$, l_w } where l_e, $l_{\Delta e}$, $l_w \in$ L.The rules are often written as "if error (input1) is l_e and change in error (inpu2) $l_{\Delta e}$ then output is l_w". On the bases of set of 7 linguistic values 49 rules in the rule set can be taken. Here experience, practical knowledge and trial and error method will help us to prepare appropriate rule base. In our design of controller 21 rules are taken as a rule base and are listed in table:3.8.

	e(K)							
Δe (k)		NB	NM	NS	ZO	PS	PM	PB
	NB				NB	NS		
	NM				NM	NS		
	NS				NS	ZO		PM
	ZO	NB	NM	NS	ZO	PS	PM	PB
	PS	NM	NS	ZO	PS			
	PM				PM			
	PB			PM	PB			

TABLE 3.8: FUZZY LOGIC RULES FOR FLC

Rule Ri= (l_e, $l_{\Delta e}$, l_w) takes e(k) and Δe(k) as inputs and assigns it to a function Pi(e(k), Δe(k),w), w ∈ [-1,1] and is given as

$$Nmin = min(n_e(l_e), n_{\Delta e}(l_{\Delta e}))$$

$$Pi(e(k),\Delta e(k),w)=min(Nmin,M_{lw}(w)) \qquad w \in [-1,1] \qquad (3.29)$$

Corresponding to each rule one pi function is associated and if we combine all 21 function in this case we get an overall function q as,

$$q(e(k),\Delta e(k),w) = max(\ p_1(e(k),\Delta e(k),w), \ldots p_r(e(k),\Delta e(k),w)), \ w \in [-1,1] \qquad (3.30)$$

Defuzzification: This stage in FLC leads us to the real number output. In FLC design the FL rules are mapped to get real number output F[e(k), Δe(k)] and is given as

$$F[e(k), \Delta e(k)] = C_F \frac{?^1 wq(e(k),\Delta e(k),w)dw}{?^1 q(e(k),\Delta e(k),w)dw} \qquad (3.31)$$

C_F is a scale factor. As the ratio of RHS of the above equation is center of area or centroid of the function q(e(k),Δe(k),w), the defuzzification is called center of Area or centroid method.

- **Fuzzy Pre compensator**

Fuzzy pre compensator as in the usual FLC also contains three stages. It contains set of linguistic values L' and set of Membership Functions (MF) M'. The linguistic values L' are used for the input to the pre compensator and linguistic values L are used for the output. The pre compensator uses linguistic values set:

L'= {NE, ZE, PO} and

Associated MF M' = $\{M_{NE}, M_{ZE}, M_{PO}\}$. : The mnemonic in L stands for negative, zero and positive respectively. It is also consisting three stages as usual FLC compensator: Fuzzification, decision making fuzzy logic and defuzzification.

Fuzzification: The three inputs of the fuzzy compensator e(k), Δe(k) and μ(k-1) are assigned with numbers me(l'), $\Delta m_e(l')$, and $m_\mu(l')$ respectively through,

$$m_e(l') = M_{l'}(C'_e e(k))$$
$$\Delta m_e(l') = M_{l'}(C'_{\Delta e}\Delta e(k))$$
$$m_\mu(l') = M_{l'}(C'_\mu \mu(k-1)) \qquad (3.32)$$

where C'_e, $C'_{\Delta e}$ and C'_μ are scale factor

Decision making fuzzy logic of pre compensator consists of 27 rule $\{R'_1, \ldots, R'_{27}\}$ as per table 3.9

IF			THEN
e(k)	Δe(k)	μ(k-1)	μ(k)
NE	NE	NE	NS
		ZE	ZO
		PO	ZO
	ZE	NE	PS
		ZE	ZO
		PO	NS
	PO	NE	PM
		ZE	PS
		PO	ZO
ZE	NE	NE	ZO
		ZE	NS
		PO	NS
	ZE	NE	ZO
		ZE	ZO
		PO	ZE
	PO	NE	PS
		ZE	PS
		PO	ZO
PO	NE	NE	PM
		ZE	PS
		PO	ZO
	ZE	NE	PM
		ZE	PS
		PO	ZO
	PO	NE	PB
		ZE	PS
		PO	ZO

TABLE 3.9: RULES FOR FUZZY PRECOMPENSATOR

Here each rule R'_i is a quadruplet $(l'_e, l'_{\Delta e}, l'_\mu, l_\mu)$ where $l'_e, l'_{\Delta e}, l'_\mu \in L'$ and $l_\mu \in L$(The linguistic values set of FLC). As in usual FLC, we had calculated function P'_i's using experience and practical knowledge and 27 P'_i related with each rule in rule set are combined to get similar function q' as in FLC. By defuzzification of the above function q we get single value output of the pre compensator and is given as

$$G[e(k), \Delta e(k), \mu(k-1)] = C_G \frac{\Sigma^1 \mu q'(e(k),\Delta e(k), \mu(k-1), \mu)d\mu}{\Sigma^1 q'(e(k),\Delta e(k), \mu(k-1), \mu)d\mu} + \mu(k-1) \quad (3.32)$$

C_G is a scale factor.

The adaptive two layer FL controller for output backlash compensation is simulated on a digital computer using MATLAB Simulink and fuzzy logic toolbox. It is proved to be very efficient in nullifying the effect of output backlash. We have carried out simulation for the known transfer function model of DC motor with output backlash, with single layer (usual FLC) and two layer compensator approach. The results are

compared for the same plant with uniform output backlash with single layer(Fig 3.25) and (ii) with two layer approach.

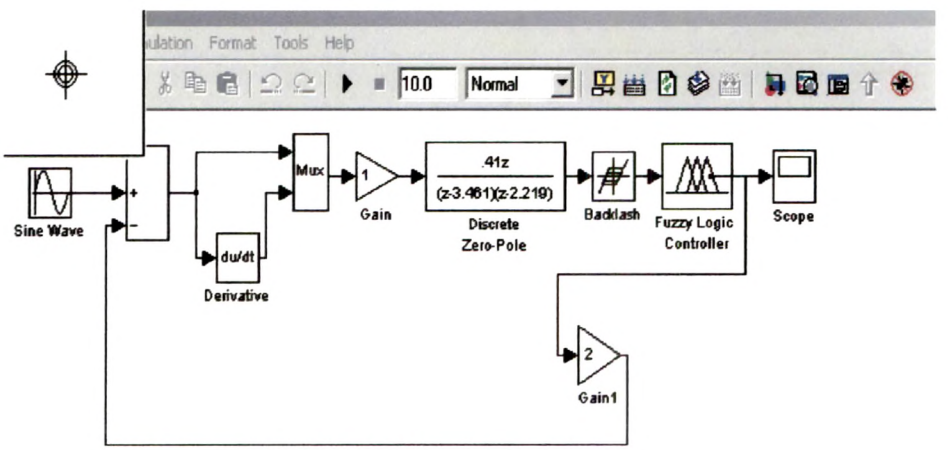


Fig 3.25: Single Layer FLC

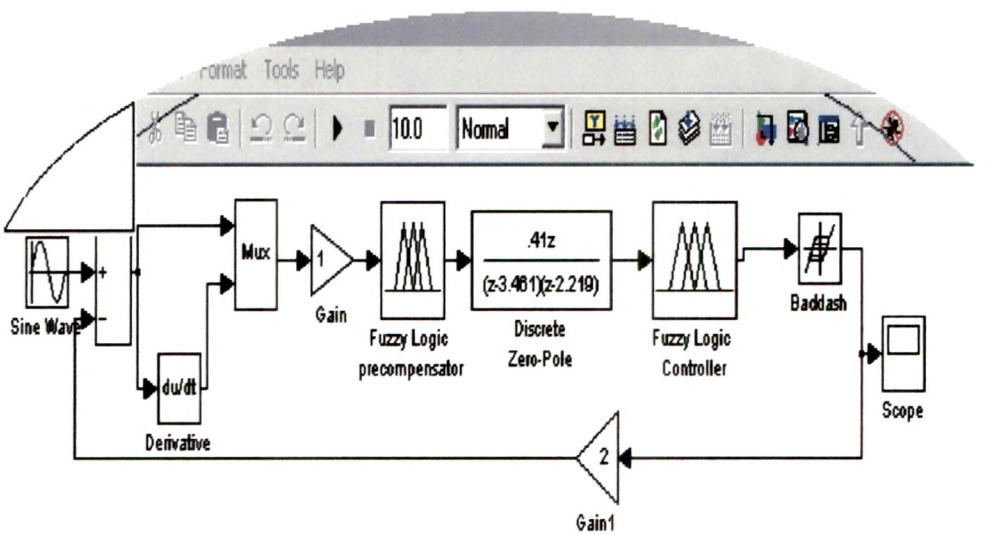


Fig 3.26: Two Layer FLC

The DC motor transfer function considered in this case study is

$$0.2481\ [z/(z-e^{13.01T}) - z/(z-e^{6.98T})] \qquad (3.33)$$

Taking sampling time T=0.1 sec, the system transfer function reduces to,

$$0.41\ z\ /\ z^2 - 5.68\ z + 7.38 \qquad (3.34)$$

Fig 3.27 shows the response and Fig 3.28 shows the effect of variation of width of backlash. The response indicates that the performance is independent of the width

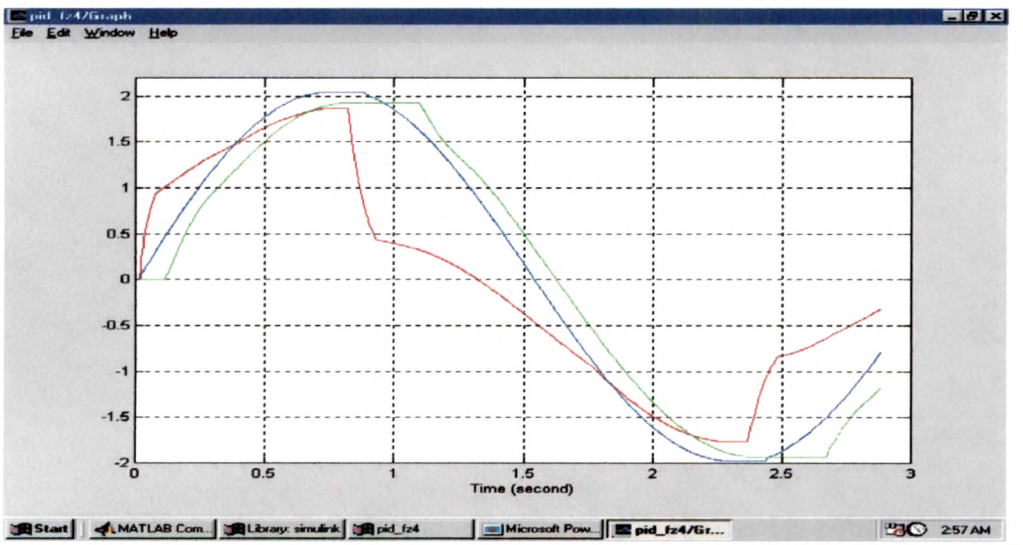


Fig: 3.27: Output Red - with FL Pre compensation; Blue - without FL compensation; Green – with PID

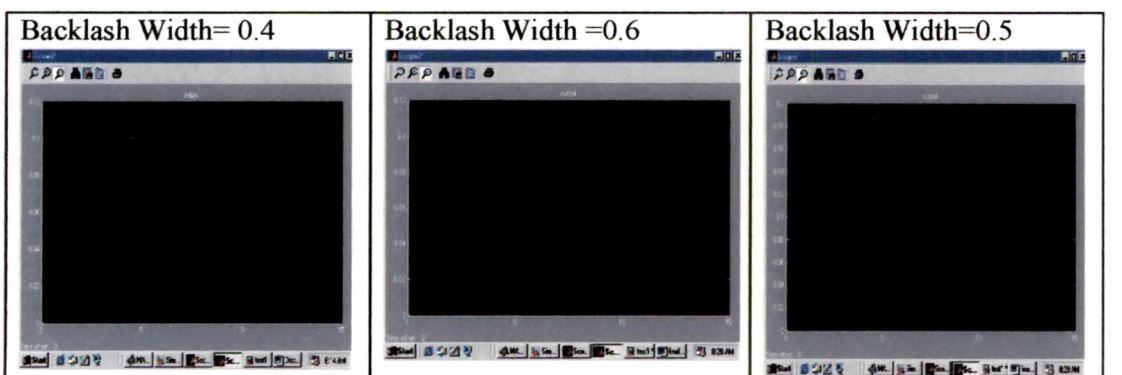


Fig 3.28 Response of two layer FLC with Backlash

The effect of variation of width of backlash is shown in the figure. The response indicates that the performance is independent of the width.

The basic fuzzy controller application which replaces the PID controller for variable speed controlled IM is presented. The comparison of this controller with conventional PID controller shows improvement in the output response (Low settling time and less overshoot).

This controller is modified further to tune the desired performance by tuning the fuzzy inference system. The training data generation is based on the required response

from the plant. Training of FLC modifies the membership function so as to adopt the change in input for a fixed tracking of O/P. This improves the insensitivity of the controller to plant parameter variation. Neuro-Fuzzy Controller design using ANFIS reduces the number of rules/membership functions and improves the speed of response in comparison to that of FLC.This improved controller makes the system robust. The system is easily implemented using crisp value (Sugeno type) on PC.A dedicated hardware can be used.

The Fuzzy Logic Controller and Fuzzy Logic Pre compensator is designed here. The fuzzy controller application replaces the PID controller for the variable speed controlled motor. From the responses obtained we can see that the Fuzzy Logic controller and FL compensator shows the improvement in the output response. The use of Fuzzy Logic compensator compensates the effects of the nonlinearity. This improved controller makes the system robust.

A discrete time two layered fuzzy logic controller (FLC) has been proposed for compensation of backlash non linearity in a control system. It consists of two layers i.e fuzzy pre compensator and a conventional fuzzy logic controller. The proposed scheme shows superior steady state and transient state performance, compared to usual single layer FLC.

The fuzzy pre compensator in this scheme can easily be added without affecting the existing system and retuning of system variables is also not required. It is a robust controller to the variation of backlash non linearity. The performance is verified through simulation result