NEURO-FUZZY SYSTEM WITH INCREASED ACCURACY SUITABLE FOR HARDWARE IMPLEMENTATION

Kannan Govindasamy

Certificate of Approval:

Vishwani Agrawal James J. Danaher Professor Electrical and Computer Engineering	Bogdan Wilamowski, Chair Professor Electrical and Computer Engineering
Michael Baginski Associate Professor Electrical and Computer Engineering	George T. Flowers Dean Graduate School

NEURO-FUZZY SYSTEM WITH INCREASED ACCURACY SUITABLE FOR HARDWARE IMPLEMENTATION

Kannan Govindasamy

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
May 09, 2009

NEURO-FUZZY SYSTEM WITH INCREASED ACCURACY SUITABLE FOR HARDWARE IMPLEMENTATION

Kannan Govindasamy

Signature of Author

Date of Graduation

Vita

Kannan Govindasamy, son of Govindasamy Krishnasamy and Karthiga Govindasamy, was born on May 29, 1984 in Madurai, Tamil Nadu, India. He graduated in 2005 with a Bachelor of Engineering degree in Electronics and Communication Engineering from Thiagaraja College Engineering, Madurai, India. In the pursuit of enhancing his academic qualification he joined the M.S. Program at Auburn University in the Department of Electrical and Computer Engineering in August 2006.

Thesis Abstract

NEURO-FUZZY SYSTEM WITH INCREASED ACCURACY SUITABLE FOR HARDWARE IMPLEMENTATION

Kannan Govindasamy

Master of Science, May 09, 2009
(B.E., Anna University, 2005)

62 Typed Pages

Directed by Bogdan Wilamowski

Fuzzy controllers are easy to design for complex control surfaces but produce rough control surfaces which might lead to unstable operation. On the other hand neural controllers are hard and complex to train but they produce very accurate output control surfaces compared to that of fuzzy controllers.The Neuro-Fuzzy controller proposed in this thesis exploits the fuzzy systems nature of utilizing expert knowledge and also produces smooth surfaces by implementing it in neural networks.

Monotonic sigmoid membership function is used to make the neural implementation an easy task. Levenberg-Marquardt algorithm(LM) which is used for feed forward networks is implemented to train the neurons. A defuzzification with trigonometric approximation algorithm using LUT-Lookup Table is developed to implement in low cost microcontrollers to make the control system highly cost effective. It is shown through extensive simulations that the proposed model produces accurate and smooth control surfaces.

Acknowledgments

I take this opportunity to thank my academic advisor Dr. Bogdan Wilamowski, for his guidance in my research throughout my Masters degree program here in Auburn. I also have had the privilege of Dr. Vishwani Agrawal and Dr. Michael Baginski taking their precious time to be in my graduate committee and receiving from them valuable feedback on my work.

I also would like to thank all my friends who have been helpful throughout my degree. I wish to dedicate this degree and work to my family who made this possible and for all their support throughout my life.

Style manual or journal used IEEE

Computer software used Latex together with the departmental style-file `aums.sty`.

Table of Contents

List of Figures

Chapter 1

Introduction

Control systems for nonlinear system [1] still remains a challenge in modern control theory, when compared to control system for linear objects. The nonlinear system becomes more difficult to understand and analyze when the system becomes a function of time. A nonlinear object is usually linearized before a controller system can be designed for the system. This is more often done by adding a reverse nonlinear function to compensate the nonlinear behavior and make the input-output relationship more linear. An adaptive nonlinear system that has nonlinear characteristics which change with time is better managed by methods of computational intelligence such as neural network and fuzzy systems [1][3][4][5]. The adaptive nature of these systems enables them to model any dynamic non-linear behavior of a control system.

Fuzzy systems utilize expert knowledge and perception based information in the form of set of rules. The dynamics of a system is generally complicated, but sometimes its behavior can be defined more conveniently using linguistic terms in which case fuzzy logic is the best option to model the system. Moreover, fuzzy systems are easy to design and implement in hardware. But, the major drawback of fuzzy controllers is that the control surfaces obtained from these systems are rough, which can cause unstable operation.

Artificial neural systems, on the other hand are known by their property of performing complex nonlinear mappings. They also produce smooth control surfaces unlike the fuzzy systems. Even though neural networks are widely implemented in software many applications require the hardware implementation. But, these systems require the computation of tangent hyperbolic activation functions. This involves division algorithms which are often too complex for simple microprocessors. So, a suitable combination of the neural architecture and the fuzzy architecture is proposed in this thesis. It exploits the property of producing fine control surfaces from neural architecture and utilizing expert knowledge from fuzzy logic.

The Neuro-Fuzzy controller proposed in this work uses a fuzzy logic with monotonic sigmoid membership functions that makes the neural implementation a very easy task. Levenberg-Marquardt algorithm(LM) [6] which is used for feed forward networks is implemented to train the neurons. Since, the implementation of neural networks in low cost microcontrollers is a challenging task, a defuzzification with trigonometric approximation algorithm using LUT [7][8] is developed to implement neurons in low cost microcontrollers to make the control system highly cost effective.

The rest of the thesis is organized as follows.

Chapter 2 provides the relavent background information about the artificial neural network. The advantages and limitations of using artificial neural networks, their types,the structural information and history of neural network are discussed.

Chapter 3 provides the relavent background information about the fuzzy control system. The two major fuzzy controllers, the Mamdani fuzzy controller and TSK fuzzy controller are discussed. The advantages and limitations of using fuzzy control system are also discussed.

Chapter 4 provides information about the proposed neuro-fuzzy architechture. The usage of monotonic membership functions is discussed.

Chapter 5 provides information about the proposed LUT based defuzzification algorithm. The extension of the algorithm to higher dimension is also explained.

Chapter 6 shows the simulations results and chapter 6 concludes the work.

Chapter 2

Neural Network

2.1 Introduction

An Artificial Neural Network (ANN) is an interconnected group of nodes similar to the network of neurons in brain. The information processing system of neural network is inspired by the functioning model behavior of biological nervous systems, such as the brain. A key aspect of neural network is that, it has to be trained. The neural network by virtue does not have the expert knowledge to solve a problem. An artificial neural network much akin to the human nervous system learns by example. In biological system learning involves by adjustments to the synaptic connections that exist betweens the neurons. This is true of artificial neural network as well. An artificial neural network is configured for specific function [15],[16].

2.2 Historical Background

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

The history of neural networks consist of the following three periods [15].

2.2.1 A Promising and Emerging Technology

McCulloch and Pitts (1943) gave the first successful simulation model of the logic based on the working model of biological neurons. These models were based on how simple neurons worked and how they were akin to simple logic elements with thresholds.

This was followed by Rosenblatt‘s (1958) perceptron. The perceptron was a three layered architecture consisting of an input layer, middle layer and output layer. The middle layer was also known as the hidden layer or the associate layer. The perceptron had the ability to learn and associate a set of inputs to output. The other important development during this initial period of interest in neural network was adaptive linear element (ADALINE) which was developed in 1960 by Widrow and Hoff. The ADALINE used an LMS (Least-Mean-Square) learning rule and was an analog system [15],[16].

2.2.2 Period of Frustration and Disrepute

The period of initial enthusiasm and high funding was followed by a period of frustration and disrepute. Minsky and Papert published ”Perceptrons: An Introduction to Computational Geometry” (1969) in which they highlighted the limitations of single layer Perceptrons to multilayer systems, in other word a perceptrons could only solve linear separable functions but not in particular like XOR or XNOR logic. The publication disenchanted researchers in the field of neural network and was followed by a period of corresponding low funding [15],[16].

2.2.3 Re-Emergence

Currently significant process has been made and works have surpassed the limitations highlighted by Minsky and Papert. The adaptive resonance theory (ART) networks by Steve Grossberg and Gail Carpenter in 1988, the associative techniques of Anderson and Kohonen and the seminal error back propagation learning algorithm by Paul Werbos 1974, infused life into neural networks. Since then this field has enjoyed a period of high funding and significant commercial applications for industry and a financial solution have been developed [15],[16].

2.3 Human Brain Working Model

The human brain is still a mystery and it is unclear how the human nervous system functions and process a information. A typical neuron consists of three main parts - dendrites, axons and synapses. A neuron has the amazing capability of transferring electrochemical signals. A branch like projections from the cell body collects information in the form of signals known as dendrites. These signals are carried over the length of cell through a long cable like projection known as axon. The axon which carries the signal end into numerous branch like projection called synapse. The synapse converts the signal into electrical activity. When the electrical activity is of sufficient strength the electrical activity is conveyed to the next neuron. Learning is involved by the adjustment of these synaptic connections [15],[16].

2.4 Artificial Neuron

Neural network is composed of a large number of highly interconnected processing elements (neurons) working together to solve specific problems. A single neuron could be modeled by the equation (1). It is diagrammatically represented as shown in Fig 2.1. The output of the neuron is given by the sigmoid transfer function. Neural networks cannot be programmed to a specific task but trained to solve by using various algorithms

$$net = \sum_{i=1}^{n} W_i x_i + W_{n+1} \qquad (1)$$

An artificial neuron has many inputs and one output. The neuron operates in two modes the using mode and the training mode. In training mode the neuron is trainied to either fire or not for a particular specified input. In using mode when a specified pattern is detected at the input the neuron responds by firing the associated output of the specified value. If the input pattern detected does not match any of the one pattern in the specified list the firing rule is used to determine whether the neuron fires or not [15],[16].

2.4.1 Components of Neuron

The behavior of a neural network depends on the two components, the weights and the input-output function (transfer function) that is specified for the units. The transfer function mainly falls into three categories:

- Linear Transfer Function

  For linear units, the output activity is proportional to the total weighted output.

- Threshold

  For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

- Sigmoid

  For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, it must be choosen how the units are connected to one another and how to set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

A three-layer network can be trained to perform a particular task by using the following procedure:

1. The network is fed with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.

2. The difference between the actual output of the network and the desired output is found.

3. The weights of each connection is changed accordingly so that the network produces a better approximation of the desired output.

2.5 Types of Neural Network

Neural Network types can be classified based on following attributes [15],[16]:

2.5.1 Applications

Neural network applications can be grouped in the following categories [15],[16]:

- Clustering:

  A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining [19].

- Classification/Pattern recognition:

  The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory.

- Function approximation:

  The tasks of function approximation is to find an estimate of the unknown function f() subject to noise. Various engineering and scientific disciplines require function approximation.

- Prediction:

  The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems. Prediction differs from function approximation by considering time factor.

2.5.2 Architecture

- Feed-forward networks

Artificial neural networks that has signals that travel only from inputs to output are called feed-forward networks. The signals travel only in one direction and there is no feedback (loops). The output of any layer does not affect that same layer. Feed-forward artificial neural networks are straight forward networks that associate inputs with outputs [15],[16].

- Feedback networks

  Artificial neural networks that have loops in the network making the signals traveling in both directions are referred as feed back networks. Feedback networks are very powerful and can get extremely complicated. The loops make the feedback network very dynamic. The feedback network changes their state continuously until they reach an equilibrium point. The network then remains in the equilibrium point until the input changes and a new equilibrium is found. Single layer feedbacks architectures are also referred to as recurrent networks; multilayer feedback are referred to as interactive networks [15],[16].

2.5.3 Learning Methods

- Supervised:

  Supervised training has both inputs and the desired outputs. The output of the training network for the input is compared against the desired outputs to adjust weights which control the network. The network is adjusted till the resultant outputs of the network match the desired network.

- Unsupervised:

  In unsupervised training the desired outputs are not provided. The system then on its own must decide to self organize to group the input data. Unsupervised networks are referred to as self-organizing or adaptive networks.

2.6 Error Back-Propagation Algorithm

The most common method of training a neural network is error back-propagation algorithm. The main technique of error back-propagation is to gradually adjust the neural network to produce the correct result for an input. This is accomplished through an iterative process by which the weights between the various nodes are altered until the network as a whole performs correctly.

In order to train a neural network to perform some task, the weights of each unit must be adjusted in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW.

The basic steps involved in the algorithm are [15],[16]:

1. Assertion of random weights to the network.
2. An input is applied to the network and the results for all nodes are obtained.
3. The errors for each node are calculated by comparing the desired result with obtained result(starting from the last node and propagating the error backward).
4. The weights of each node are updated based on the error calculated.
5. The steps 1 to 4 are repeated until the desired output is gotten.

2.7 Advantages of Neural Network

A neural network can be assumed to be as expert in the category of information that it has been trained. The neural network has the ability to interpret meaningful information from imprecise data, interpret patterns and trends which are very hard and too complex to be done by other computing techniques. The following are the important features of artificial neural networks [15],[16].

- Parallel Processing:

  The parallel processing capability of neural networks allows it to solve problems where multiple constraints have to be met, which is very much similar to the real biological nervous system.

- Self-Organization:

  A neural network has the ability to self organize while learning. The self-organization helps in the visualization of low dimensional view of high dimensional data.

- Graceful Degradation:

  An important feature of neural network is that partial removal of components only causes corresponding degradation in the accuracy of output rather than the failure of the system.

- Continuous Adaptivity:

  The neural network has the capability to learn from the sample space of the input vectors.

2.8 Limitations of Neural Network

Neural network is the not the perfect solution of every problem. Neural network can only be considered as the expert only in the category of information it has been trained and tasked. Neural networks are limited by problems like the following:

- Accuracy: Neural network is not an hundred percent accurate system. The accuracy of neural network depends on the amount of training and the size of the network. Systems which require 100 percent accuracy cannot depend on neural networks.

- Black Box:

  Akin to the nervous system the neural network system is like a black box. The system learns based on training and experience but cannot justify the decision.

- Training:

  The amount of training needed by neural network is an obstacle, but this problem has been overcome with the vast amount of data available in the digital world.

  Neural network is more of art rather than science, but an art whose results can be observed and measured.

2.9 Conclusion

The computing world has a lot to gain from neural networks. The ability of neural networks to learn by example makes them very flexible and powerful. The self-learning and self-tuning capabilities makes it ideal for industrial application. Furthermore there is no need to write an algorithm in order to perform a particular task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

Neural networks also contribute to other areas of research such as neurology and psychology. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain.

Perhaps the most exciting aspect of neural networks is the possibility that some day conscious networks might be produced. There is a number of scientists arguing that consciousness is a mechanical property and that conscious neural networks are a realistic possibility.

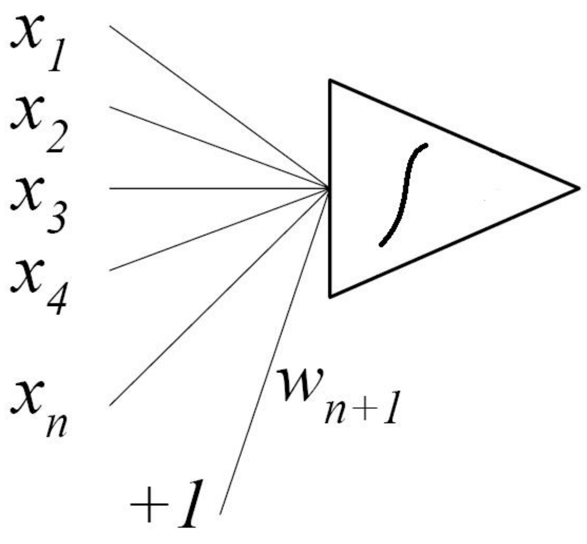

Figure 2.1: A neuron with n inputs, weights and threshold value

Chapter 3

Fuzzy Logic

3.1 Introduction

"Fuzzy logic is basically a multi-valued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/white, etc. Notions like rather warm or pretty cold can be formulated mathematically and processed by computers." - Bauer et al. Fuzzy logic provides a means of calculating intermediate values between absolute true and absolute false with resulting values ranging between 0 and 1. With fuzzy logic, it is possible to calculate the degree to which an item is a member [13],[14],[17].

3.2 Historical Background

The concept of Fuzzy logic was introduced by Professor Zadeh at the University of California at Berkeley in the 1960's. His goal was to develop a model that could more closely describe the natural language process. Fuzzy logic can be compared to the human decision making process. Conventional or classic logic is more of boolean conditions (true/false). Fuzzy logic is a superset of boolean logic, having its own similarities and differences with boolean algebra [13],[14],[17].

3.3 Fuzzy System

The structure of a Fuzzy system consists of:

- Rule base: selects the set of fuzzy rules
- Database: defines the membership functions used in the fuzzy rules

- Reasoning mechanism: performs the inference procedure (derive a conclusion from facts and rules)

3.4 Membership Function

The membership function represents the degree of truth as an extension of valuation. The degree of truth is often confused with probabilities. Probability is the likelihood that something is true. Fuzzy logic is the degree to which something is true (or within a membership set)

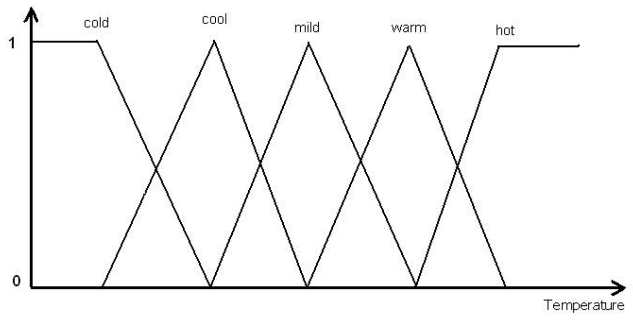

Figure 3.1: Membership Function

The three major types of membership functions are gaussian, triangular and trapezoidal.

The basic rules of membership functions are

- Each point of the input should at least belong to one membership function.
- The sum of two overlapping membership function should not exceed 1.
- The accuracy of the fuzzy system can be raised by increasing the number of membership functions, but this affects the stability of the control system.

3.5 Mamdani and TSK Fuzzy Controllers

The most important application of fuzzy logic is the fuzzy control system which directly uses the fuzzy theory. The Mamdani fuzzy controller originally used three steps to create a fuzzy controlled system.

1. Fuzzification: The process of using membership functions to graphically describe a situation
2. Rule evaluation: The process of applying the fuzzy rules to fuzzified inputs.
3. Defuzzification: The process of obtaining the crisp or actual results.

The Madmani fuzzy controller consists of the following block shown in fig 3.3,

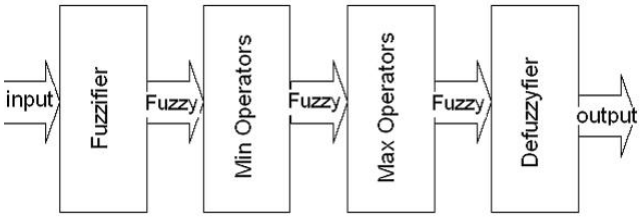

Figure 3.2: Fuzzy System

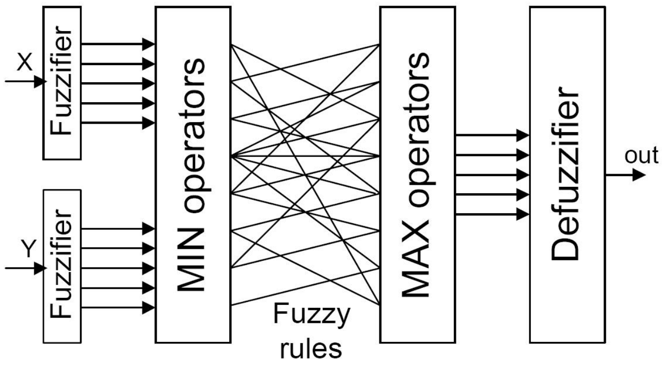

Figure 3.3: Block Diagram Mamdani Fuzzy Controller

A membership function specifies the effect of the particular variable on the final output. The block diagram of a typical fuzzy system as proposed by Takagi-Sugeno-Kang (TSK) is shown in Fig 3.4.

The analog inputs are converted to a set of n fuzzy values by the fuzzifier block. This process of mapping a single value into an n dimensional space is called fuzzification. Fuzzification is done by using membership functions; n membership functions are required to map to n dimensional space. Each analog input belongs to at least one and preferably two membership functions (overlapping).

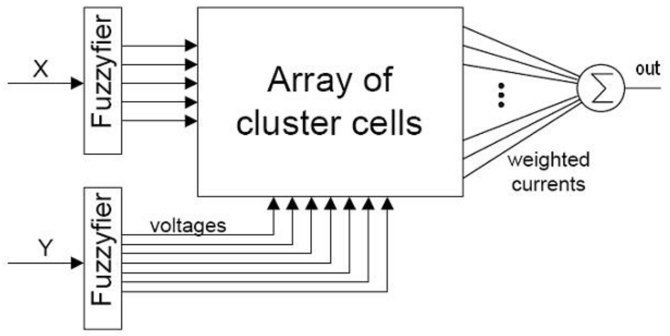

Figure 3.4: TSK Fuzzy Controller

The accuracy of the system can be increased by increasing the number of membership functions. But, very dense functions can lead to frequent controller action (also known as hunting) and sometimes may lead to instability. The fuzzified values are processed by fuzzy logic blocks which implement MIN and MAX operations. MIN and MAX operations are analogous to AND and OR operations in Boolean logic. Boolean logic can be considered as a subset of fuzzy logic since fuzzy logic is applicable for all the values from 0 to 1, unlike Boolean logic which operates on 0 or 1 only. TSK architecture does not require a MAX operator. The MAX operator is replaced by a weighted average which is applied directly to the values of MIN operation [13],[14],[17].

3.6 Advantages of Fuzzy Logic

- Fuzzy logic mimics human decision making to handle vague concepts.
- Computation is faster due to intrinsic parallel processing nature.
- Fuzzy logic has the ability to deal with imprecise information [13],[14],[17].
- Modeling of complex, non-linear problems.
- Fuzzy logic is popular in engineering because of its ability to deal with real-world, ambiguous problems. Fuzzy logic is mathematical oriented, but also at the same time emphasizes ambiguity and uncertainty. The main popularity is due to its practicality, not because of its mathematical rigor. It was developed to address practical problems, and the math came later. Fuzzy was developed in an atmosphere of application [13],[14],[17].
- Fuzzy logic does not preclude the idea of absolutes - after all, membership functions can be equal to 0 or 1. Fuzzy rather adds to the idea of absolutes more levels of belonging [13],[14],[17].
- Fuzzy logic uses membership functions and rules to approximate any continuous function to a precision of any degree.
- Fuzzy logic simplifies design complexity by using experience and knowledge in simple linguistic like rules. The input - output relations are not modeled by complex math equations and can be even used by non-experts [13],[14],[17].

3.7 Limitations of Fuzzy Logic

- Highly abstract and heuristic.
- Need experts for rule discovery (data relationships).
- lack of self-organizing and self-tuning mechanisms of neural network [13],[14],[17].

- The most important drawback of fuzzy logic is that it's not always accurate. The results are perceived as guesses, so the results cannot be as widely trusted as that of classical logic.

- Fuzzy logic is often confused with probability theory, and the terms are used interchangeably. While they are similar concepts, they do not say the same things. Probability is the likelihood that something is true. Fuzzy logic is the degree to which something is true [13],[14],[17].

- The traditional low respectability of the fuzzy logic is the biggest problem. Though fuzzy logic is the superset of all logic, it's not been widely accepted as classical logic because of the lack of precision.

3.8 Conclusion

Fuzzy logic is an alternative design methodology for control system which is simpler and faster. Fuzzy logic reduces the complexity of the design for control system and reduces the design time. The tuning of the control system is done by adjusting the membership function rather than redesigning the control system. Fuzzy logic is a better solution to the non-linear control system because it has better control performance than linear, peiceise linear and lookup table techniques.

Chapter 4
Proposed Neuro-Fuzzy Model

4.1 Introduction

Control systems for nonlinear system [1],[18] still remains a challenge in modern control theory, when compared to control system for linear systems. An adaptive nonlinear system that has nonlinear characteristics which change with time is better managed by methods of computational intelligence such as neural network and fuzzy systems. Fuzzy systems utilize expert knowledge and perception based information in the form of a set of rules. The dynamics of a system is generally complicated, but sometimes its behavior can be defined more conveniently using linguistic terms in which case fuzzy logic is the best option to model the system. Moreover, fuzzy systems are easy to design and implement in hardware. But, the major drawback of fuzzy controllers is that the control surfaces obtained from these systems are rough, which can cause unstable operation [18].

Fuzzy set system theory was introduced by Zadeh [9]. The fact that a decision (output) might depend on several factors (variables) is implemented using membership functions.

A membership function specifies the effect of the particular variable on the final output. The block diagram of a typical fuzzy system as proposed by Takagi-Sugeno-Kang (TSK) is shown in Fig 3.4.

The analog inputs are converted to a set of n fuzzy values by the fuzzifier block. This process of mapping a single value into an n dimensional space is called fuzzification. Fuzzification is done by using membership functions; n membership functions are required to map to n dimensional space. Each analog input belongs to at least one and preferably two membership functions (overlapping) [18].

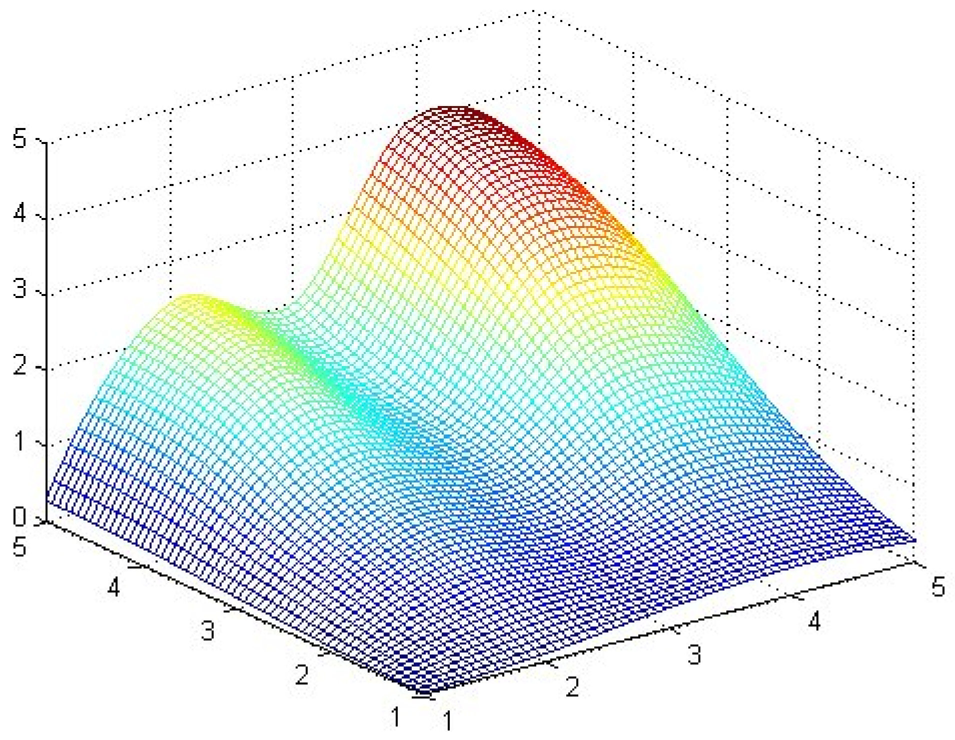

Figure 4.1: Required Control Surface

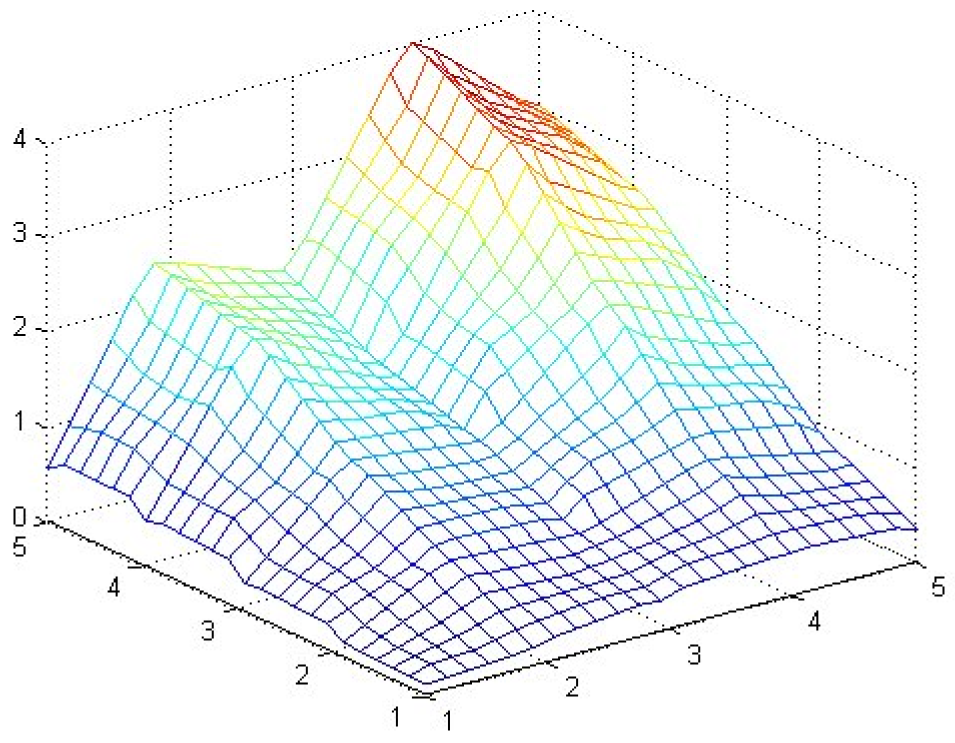

Figure 4.2: Control Surface Using Triangular TSK Fuzzy system

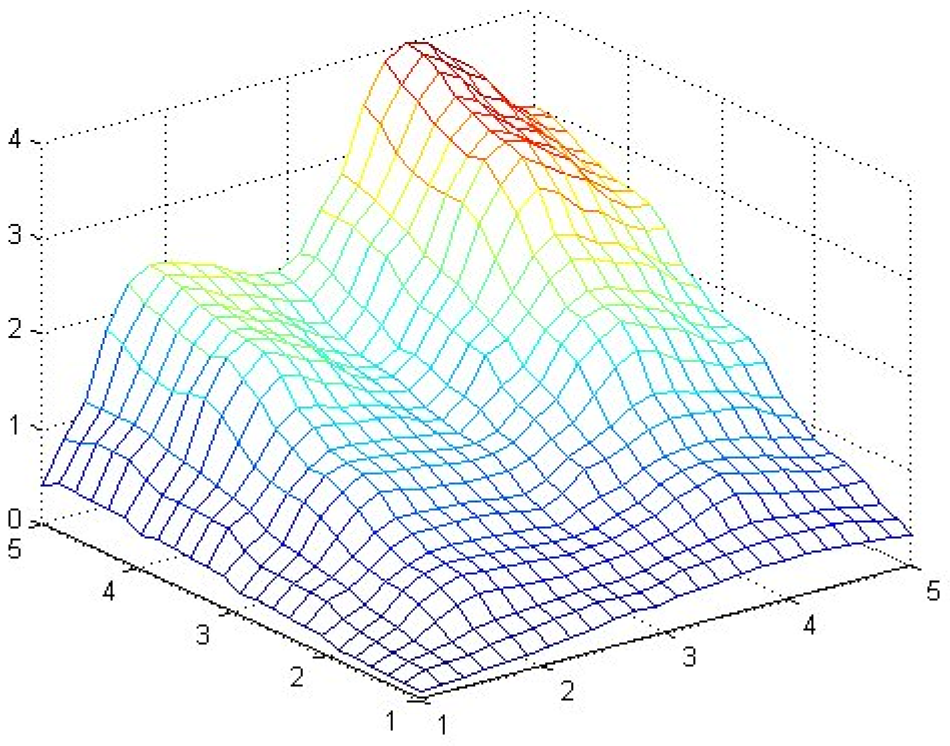

Figure 4.3: Control Surface Using Guassian Memberhsip TSK Fuzzy system

The accuracy of the system can be increased by increasing the number of membership functions. But, very dense functions can lead to frequent controller action (also known as hunting) and sometimes may lead to instability.

The fuzzified values are processed by fuzzy logic blocks which implement MIN and MAX operations. MIN and MAX operations are analogous to AND and OR operations in Boolean logic [18]. Boolean logic can be considered as a subset of fuzzy logic since fuzzy logic is applicable for all the values from 0 to 1, unlike Boolean logic which operates on 0 or 1 only. TSK architecture does not require a MAX operator. The MAX operator is replaced by a weighted average which is applied directly to the values of MIN operation. Fig 4.1 depicts a required control surface, Fig 4.2, 4.3 gives a control surface from traditional triangular and Gaussian membership function [18].

4.2 Limitations of FPGA Implementaion

The TSK defuzzification architecture is not suitable for FPGA implementation because it requires normalization, which is very computational intensive. Mamdani architecture is also not suitable because its gives a very rough control surface and the architecture requires division in defuzzification which makes it a tough task to implement [18].

4.3 Neuro-Fuzzy Model

The proposed neuro-fuzzy architecture consists of two blocks. The first block does the fuzzification and is implemented using a neural network. The second block which does the processing of fuzzified values and defuzzification is done through a LUT based sinusoidal approximation algorithm, which makes a FPGA or microcontroller implementation a simple and easy task [18].

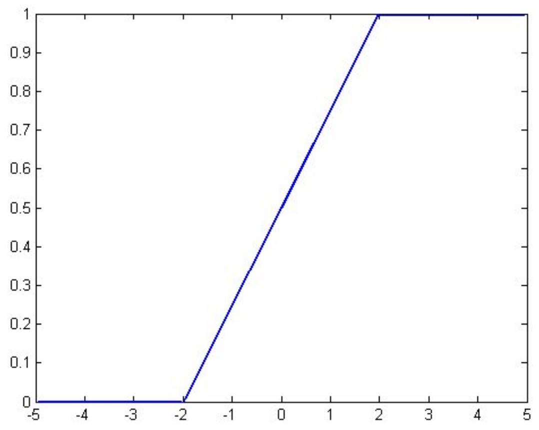

Figure 4.4: Piecewise Linear

4.4 Neural Fuzzification Using Monotonic Functions

The control surface obtained using the triangular, trapezoidal and Gaussian membership functions are not smooth and accurate. And moreover, these membership functions are difficult to be modeled and trained in a neural network. So, a neuro-fuzzy model with monotonic membership functions is proposed and used in this work which is easier to implement in a neural network [18].

Three monotonic membership functions are used in this model. One of them is a exponential sigmoid function Fig 4.5. and is given by the formula:

$$f(net) = 1/(1 + exp(-net)) \qquad (2)$$

The other two monotonic functions are piecewise linear Fig 4.4. and parabolic sigmoid Function Fig 4.6.

The reasons for choosing monotonic functions are:

- A sigmoid with a exponential function is the characteristic transfer function of bipolar differential transistor pair.
- A MOS differential transistor has similar characteristics as first half of Fig 4.6.

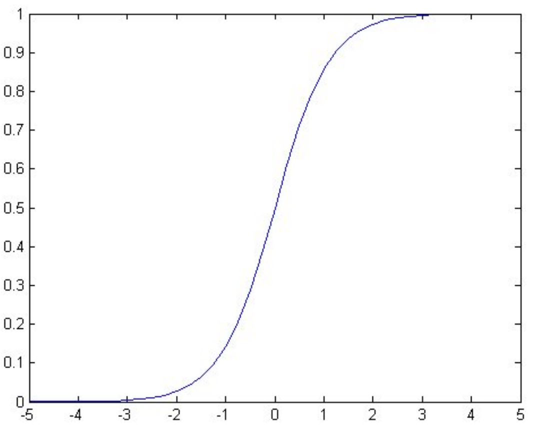

Figure 4.5: Exponential Sigmoid

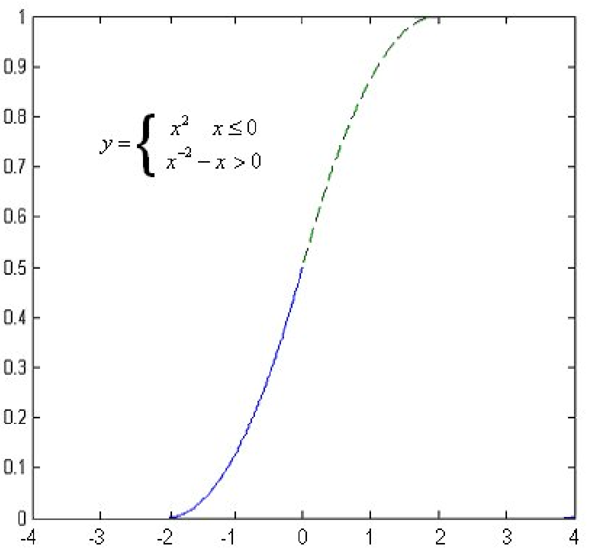

Figure 4.6: Parabolic

- A piecewise linear function could be easily generated in a microcontroller and DSP chips.

Hence a single neuron could be could be easily modeled in hardware using these monotonic functions. Now membership functions can be constructed by mirror imaging the sigmoid monotonic functions as shown in Fig 4.8. The membership function is gotten by subtracting two functions F1 and F2 , where F1 and F2 are two sigmoid monotonic functions [18].

The membership functions are constructed from these basic functions. This membership function can be implemented using two neurons. One of the neurons output gives the first half of the membership function and the negated output of the second neuron implements the second half of the function. Extending this idea a set of neurons is trained to give the required number of sigmoid membership functions. The output of a trained neural network for a sample application is shown in 4.9.

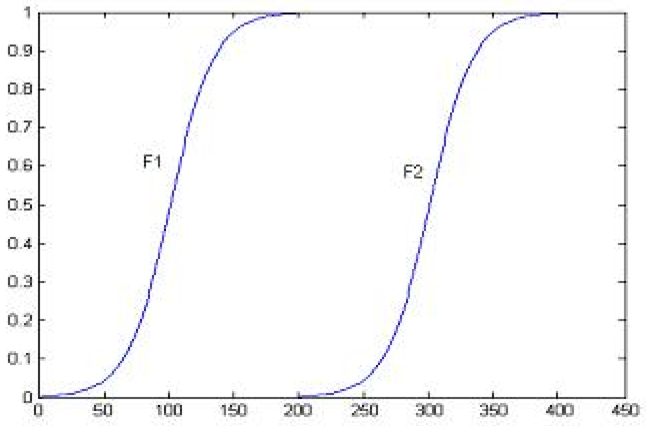

Figure 4.7: A Membership Function Implemented Using Exponential Functions F1 and F2

All the neurons use unipolar activation function and if designed properly any fuzzy system can be easily modeled. It can be observed that there was no training process required for this architecture. If training is allowed the network architecture is significantly simplified. LM (Lavenberg-Marquardt) [6] is used in this work to train the network. This algorithm is implemented in MATLAB. Cross layer weights have been introduced to make

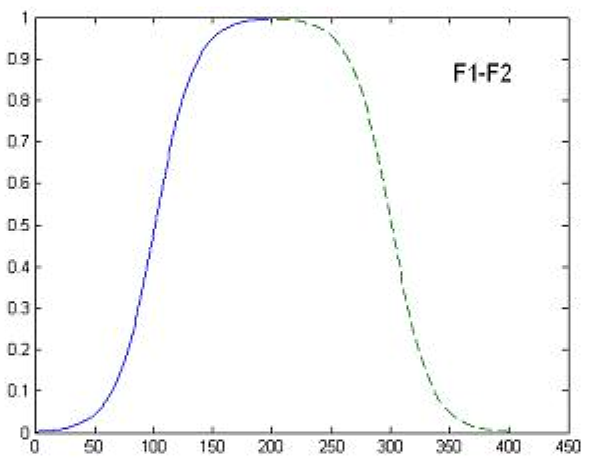

Figure 4.8: A Membership Function Implemented Using Exponential Functions F1 and F2

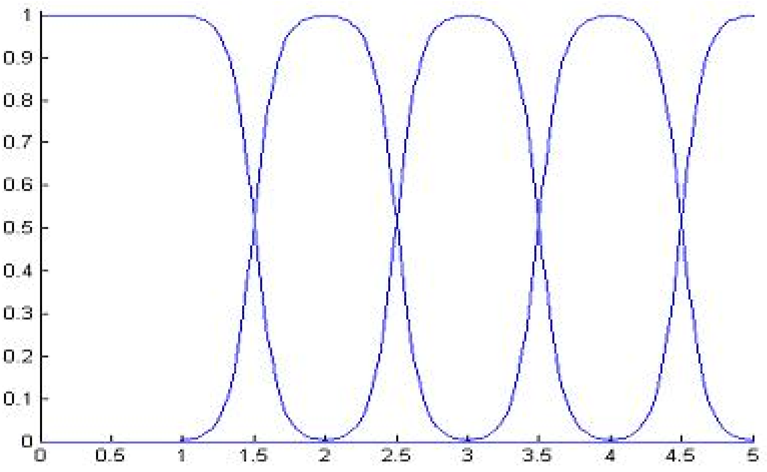

Figure 4.9: Exponential Sigmoid Membership Functions

the feed forward network [6] more accurate. Thus the above idea shows, how a fuzzifier block can be implemented through neural network by consciously choosing the membership functions [18].

4.5 Conclusion

Thus the limitation of implemnting a fuzzy control system identifed can be succesfully negotiated by using a neural network to implement a fuzzy system by consciously choosing the membership Function.

Chapter 5
LUT Based Nonlinear Mapping

5.1 Introduction

Using the established methods for defuzzification results in rougher control surface also hardware implementation of defuzzification by a neural network is very expensive and requires more processing capability. Micro controllers like Motorola HC 11 which is 8-bit processor is incapable of doing such processing. These limitations warrant the use of a LUT based solution for high precision and cost effective control system [18].

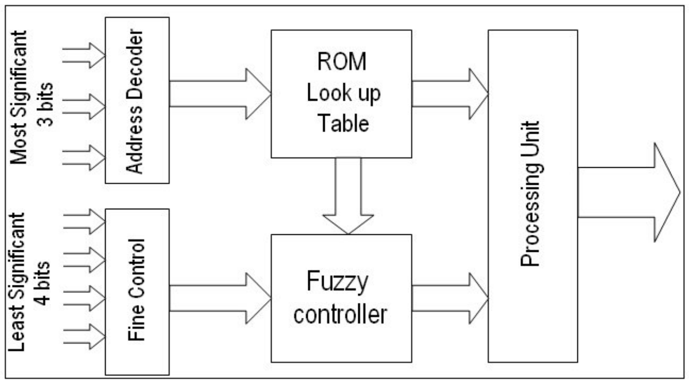

Figure 5.1: Defuzzification Block

5.2 Defuzzification with Trignometric Approximation Algorithm

A LUT based model which is suitable for FPGA implementation is shown in Fig 5.1. The inputs to the system are the digital values of the fuzzified values from the first neural fuzzification block. The digital values are obtained using an analog to digital (A/D) converters in the FPGA. In this approach, the model consists of a ROM and address of which is determined by the 3 most significant bits of the fuzzified values. Weighted average is done using the values stored in the LUT [18].

This method results in higher precision and smoother control surface. However the major disadvantage is if the number of input is increased the size of LUT increases exponentially. To handle this problem the size of LUT is kept constant and least significant bits are used for finer calculation, but essentially the size of LUT is kept constant. In this model we use a (4,4) LUT for the simulation.

The advantage of this method is the efficient usage of a very small lookup table (LUT), which minimizes the hardware requirement; hence, easier implementation and also the error produced by this approach is very minimal resulting in a very smooth control surface [18].

The values from the LUT are defuzzified by a second order sinusoidal approximation algorithm which is better than the Second order defuzzification algorithm (SODA) [12][13]. The algorithm is explained by Fig 5.2. The single space approximation is a linear approximation; the double space approximation needs the value of 'y' from the LUT, but gives higher resolution output. The single space sinusoidal approximation is the one which this thesis deals with. The value of 'y' is found by fixing a sine curve.

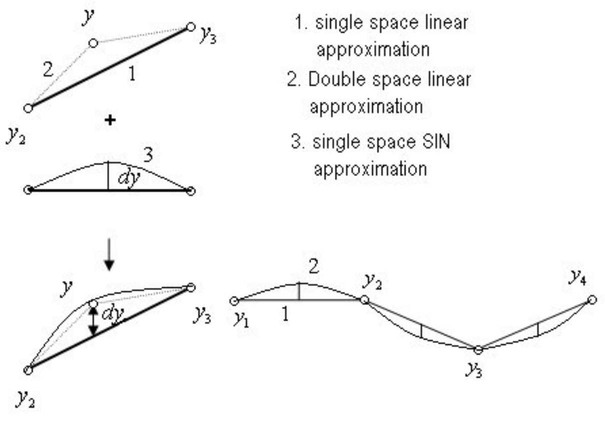
1. single space linear approximation
2. Double space linear approximation
3. single space SIN approximation
y
y2
y3
y1
y4
dy
1
2
3
+

Figure 5.2: Defuzzification with Sinusoidal Approximation

Consider the equation

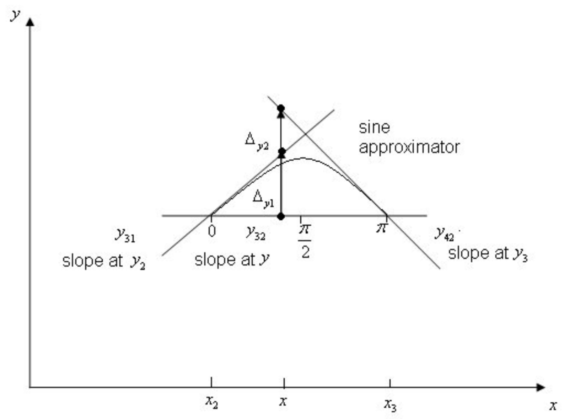

Figure 5.3: Illustration of Approximation Algorithm

$$y(x) = ax + b + csin(\pi\Delta x/x) \qquad (3)$$

Considering 4 points

separated by x_1, x_2, x_3 and x_4 distance x. Suppose x is a point between x_2 and x_3 at a distance Δx from x_2 .The corresponding Y values are y_1, y_2, y_3 and y_4 which are gotten respectively from the LUT

At x_2 and x_3 the value of sine is zero

$$y(x_2) = ax_2 + b \qquad (4)$$

$$y(x_3) = ax_3 + b \qquad (5)$$

Solving equation (4) and equation (3) gives $'a'$ and $'b'$

The first derivative is

$$\delta y/\delta x = a + ccos(\pi\Delta x/x) \qquad (6)$$

At x_2 and x_3 the value of cos is +1 and -1 respectively

Therefore,

$$y'(x_2) = a - c \qquad (7)$$

$$y'(x_3) = a + c \qquad (8)$$

Equation(8) - equation(7) gives:

$$y'(x_3) - y'(x_2) = 2c \qquad (9)$$

The slopes are determined by

Slope of y at x_2

$$y_{31} = y_3 - y_1/2x \qquad (10)$$

Slope of y at x_3

$$y_{42} = y_4 - y_2/2x \qquad (11)$$

Slope of y at x

$$y_{32} = y_3 - y_2/x \qquad (12)$$

Substituting equation (10) and equation (11) in equation (9)

$$c = 0.25(y_4 + y_1 - y_2 - y_3) \qquad (13)$$

The second derivative

$$\Delta_{y1} = y_{31} - y_{32}/2 \qquad (14)$$

$$\Delta_{y2} = y_{42} - y_{32}/2 \qquad (15)$$

Substituting equation(13), equation (14), equation (15) in equation(3)

$$y(x) = y_2 + (y_3 - y_2)\Delta x + sin(\pi\Delta x/x)\Delta_{y1} + sin(\pi\Delta x/x)\Delta_{y2} \qquad (16)$$

The first two terms of equation (16) gives the linear approximated value. These sin values are also retrieved from the LUT.

5.3 Application of Algorithm to 1-Dimension Problem

To compare the performance of the new algorithm with SODA, consider the equation:

$$y(x) = xsin(x) \qquad (17)$$

Linear approximation is applied and the result is shown in Fig 5.4. Results obtained by the new algorithm are depicted in Fig 5.6.and Fig 5.7. Error pattern observed by the new algorithm is shown in Fig 5.8. The total error is less than the total error of SODA by 25

The difference between the single space approximation and the double space approximation is that the double space algoritm uses a (4,4) LUT while the single space approximation uses a (8,8) LUT.

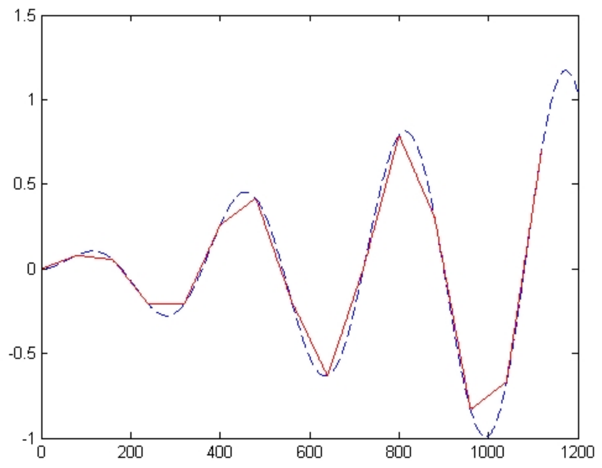

Figure 5.4: Double Space Linear Approximation

5.4 Extension of Algorithm to 2-Dimension and more

The same approach is extended for the 2-dimensional problem. 4 values from the LUT are required to find the linear approximation shown in Fig 5.9.

The algorithm is extended to 2-dimension by solving both x-axis and y-axis. The Fig 5.10 depicts the approximation is done along the four direction and the average is found. 12 adjacent points are required to find the value of $'y'$. In the same manner, the algorithm could also be extended to more than 2-dimension problems [18].

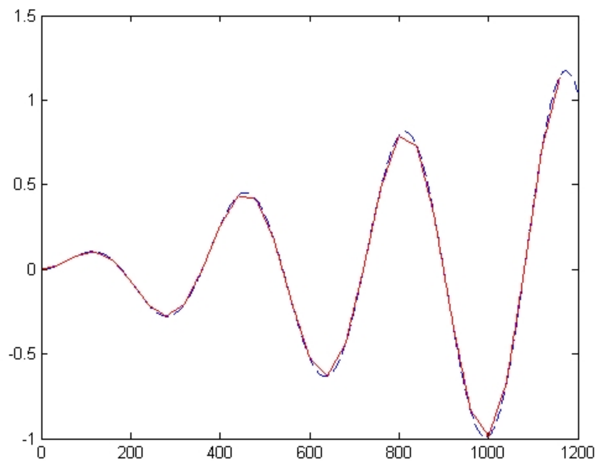

Figure 5.5: Single Space Linear Approximation

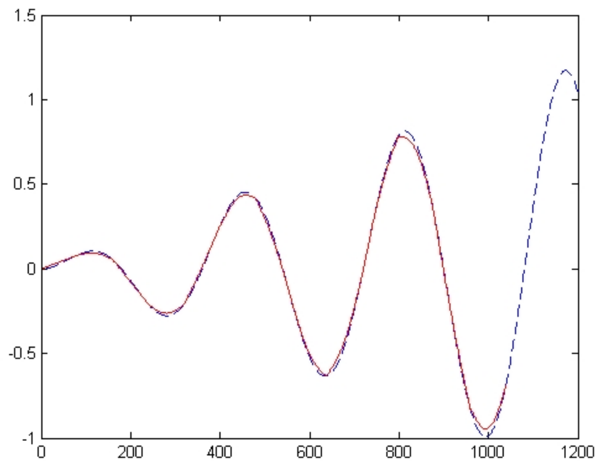

Figure 5.6: Double Space Sine Approximation

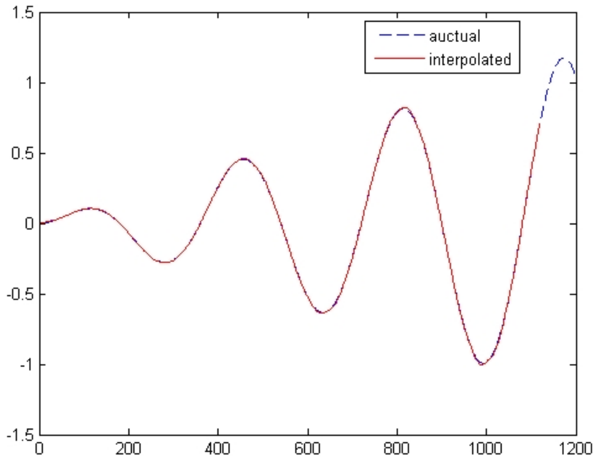

Figure 5.7: Single Space Sine Approximation

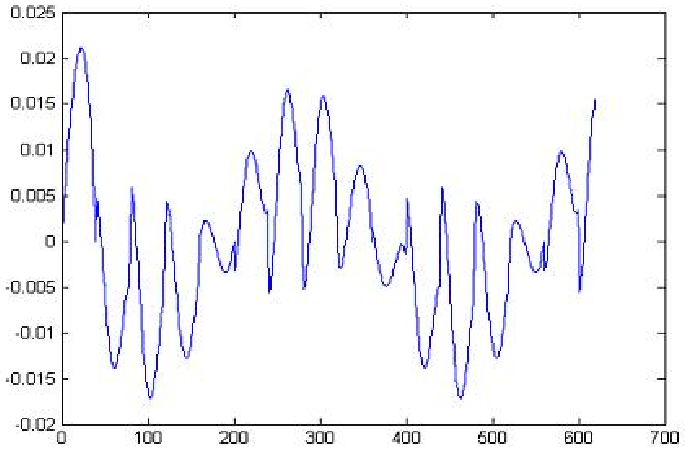

Figure 5.8: Error Observed Using LUT Algorithm in 1-D

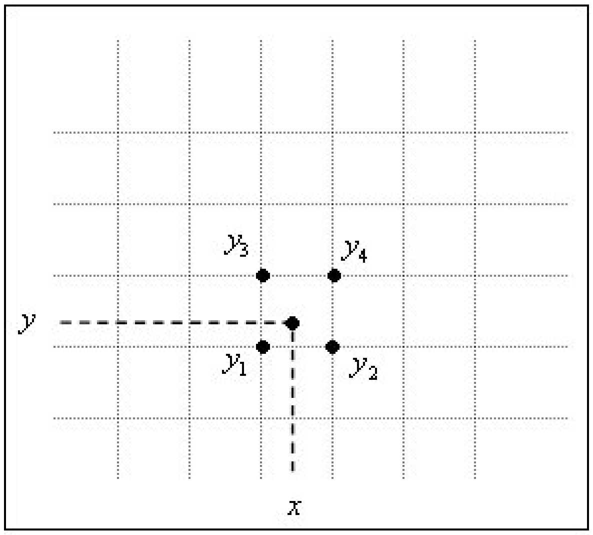

Figure 5.9: Application of Algoritm to 2-Dimensional Problem

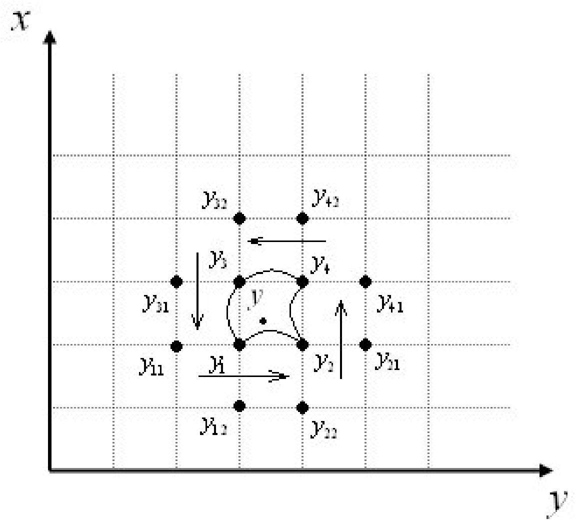

Figure 5.10: Application of Algoritm to 2-Dimensional Problem

Chapter 6
Simulation Results

In this section the performance of the proposed membership functions are compared with traditional triangular and gaussian membership functions. Fig 6.1 shows the given data points from which the control surface has to be constructed. Fig 6.2- Fig 6.5 shows the control surfaces constructed using various methods. The TSK with the parabolic monotonic functions has better control surface Fig 6.2. It can be observed that the exponential Fig 6.3 gives an even more smoother control surface when compared to triangular, Gaussian, piecewise linear and parabolic membership functions.

Fig. 6.5 shows the control surface constructed using trigonometric approximation algorithm. This algorithm produces a surface which is very close to the expected one and demonstrated to be better than all the other membership functions [18].

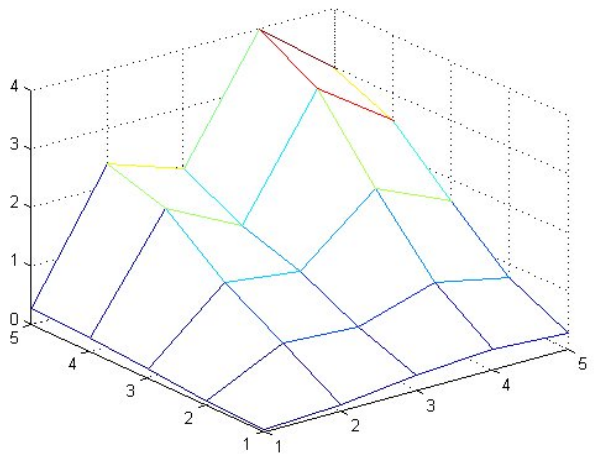

Figure 6.1: A plot of the Stored data in LUT

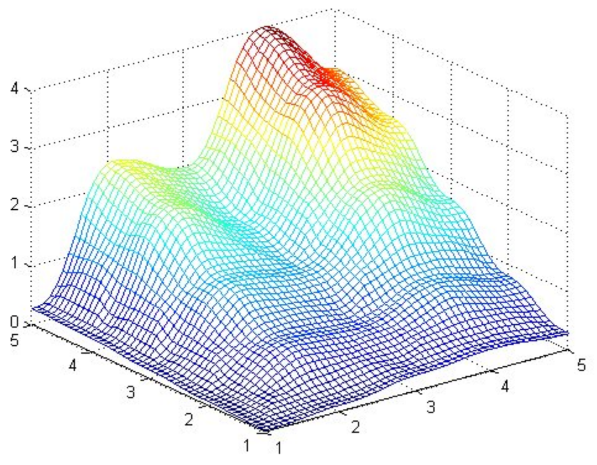

Figure 6.2: parabolic with Traditonal TSK Fuzzy system

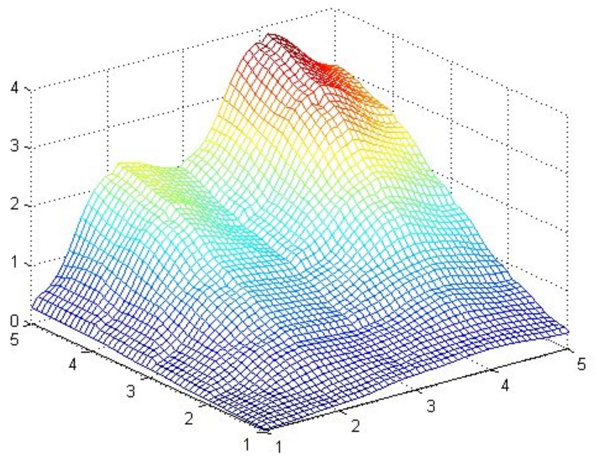

Figure 6.3: Exponential with Traditional TSK Fuzzy system

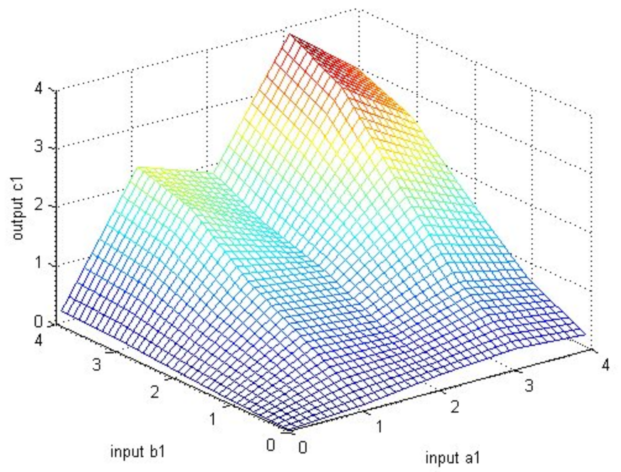

Figure 6.4: Nonlinear Mapping Using Single Space Linear Approximation

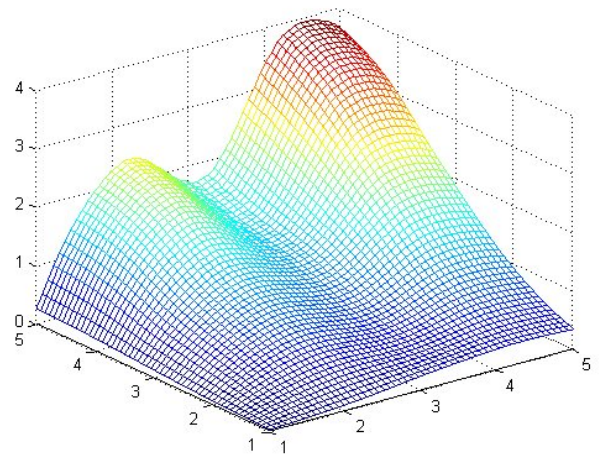

Figure 6.5: Nonlinear Mapping Using Sinusoidal Approximation

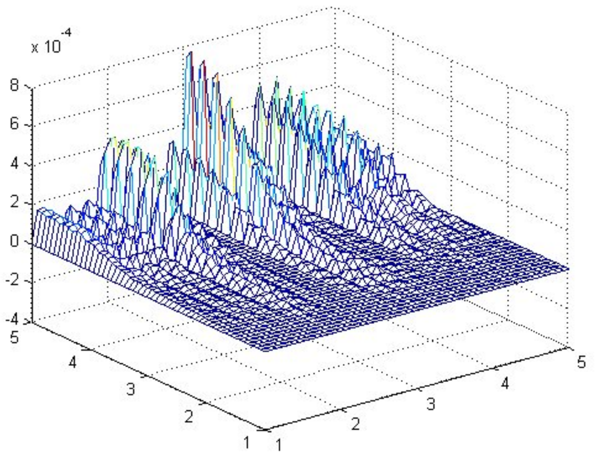

Figure 6.6: Error plot of Nonlinear Mapping Using Sinusoidal Approximation

Chapter 7
Conclusion

In this thesis a simple yet accurate implementation of a neuro-fuzzy system was proposed. Sigmoid membership functions were used in fuzzy logic so that they could be implemented using neural networks. A LUT (Lookup Table) algorithm was introduced to implement it in the FPGA. This LUT algorithm in itself can be used to model a non-linear control surface with sufficient accuracy. It was demonstrated using MATLAB simulations that the proposed neuro-fuzzy model produces accurate and smooth control surfaces avoiding the instability problem in fuzzy systems

Bibliography

[1] Shouling H. Relf, K. Unbehauen, R, Neural approach for control of nonlinear systems with feedback linearization Neural Networks, IEEE Transactions on On page(s): 1409-1421, Volume: 9, Issue: 6, Nov 1998

[2] Deutsch, S. and Deutsch, A., Understanding the NerwusSjmem: An Engineering Perspective, IEEE Press, Piscataway, NJ, 1993.

[3] Zurada, J., Intrudiiction to Artifkid Naira1 Systems, West Publishing Company, 1992.

[4] Wilamowski, B. M. Jaeger, R. C. and Kaynak, M. 0., "NewFuzzy Architecture for CMOS Implementation" IEEE Trun,sucrion on 1,xfuWial Electrunia, vol. 46, No. 6, pp. 1132-1136, Dec. 1999.

[5] Wilamowski, B. M. Chen, Y.,"Efficient Algorithm for Training Neural Networks with one Hidden Layer", in IJCNN. Proc. 1999 International Joint Conference Vol. 3, page(s): 1725-1728. Discovery," Decision Support Systems J., vol. 35, no. 1, pp. 129-147, 2003.

[6] Tan H. Sandige, R. and Wilamowski, B. M., "Hardware implementation of PLD based fuzzy logic controllers using Look-up table technique", proc ANNIE '94, Nov -1994, vol. 4, pp.89-94.

[7] Tikk, D. and Baranyi, P., "Comprehensive Analysis of a New Fuzzy Rule Interpolation Method IEEE Transaction on Fuzzy System, vol. 8, no.3 page(s). 281-296, JUNE 2000 281.

[8] Zadeh, L. A, Fuzzy sets. Information and control, New York, Academic Press vol 8, pp. 338-353, 1965.

[9] Takagi, T. and Sugeno, S, Derivation of Fuzzy Control Rules from Human Operator's Control Action . Proc. of the IFAC Symp. On Fuzzy Info Knowledg Representation and Decision Analysis, pp. 55-60, July 1989.

[10] Ying, H, " General SISO Takagi-Sugeno Fuzzy Systems with Linear Rule Consequent are Universal Approximators", IEEE Transaction on Fuzzy System, vol.6, no.4, Page(s):582-587 November.

[11] Dharia, Gownipalli, N. Kaynak, J. Wilamowski, B.M., Fuzzy controller with second order defuzzification algorithm., in IJCNN '02. Proc. 2002 International Joint Conference Vol. 3, page(s): 2327-2332.

[12] Dharia, Gownipalli, N. Wilamowski B.M, Kaynak O.Multi dimensional second order defuzzification algorithm (M-SODA), IECON ’02, Industrial Electronics Society, Nov. 2002,Vol. 4, page(s): 3215- 3220.

[13] http://www.doc.ic.ac.uk/ nd/surprise96/journal/vol4/sbaa/report.html

[14] http://www.makhfi.com/tutorial/introduction.htm

[15] http://www.doc.ic.ac.uk/ nd/surprise96/journal/vol4/cs11/report.html

[16] http://www.zaptron.com/literature/neurofuzzy.htm

[17] http://www.aptronix.com/fide/whyfuzzy.htm

[18] Govindasamy, K. Neeli, S. Wilamowski, B.M. Fuzzy System with Increased Accuracy Suitable for FPGA implementaion, proc. 2008 IEEE, Intelligent Engineering Systems Confenrce (INES), 25-29 Feb, page(s): 133-138

[19] Neeli, S. Govindasamy, K. Wilamowski, B.M. Malinowski, A., Automated Data Mining from Web Servers Using Perl Script, proc. 2008 IEEE, Intelligent Engineering Systems Confenrce (INES), 25-29 Feb, page(s):191 - 196

Appendices

Appendix A

Matlab Code

A.1 1-Dimensional LUT algorithm

```
clc;
clear all;
total=1200;
t=40;
figure(1);clf;
for i=1:total
p(i)=i-1;
x(i)=(pi*p(i))/180;
y(i)=(i/1000)*sin(x(i));
end
plot(p,y);
figure(2);
stem(p,y);
for i=1:t:total
a(i)=p(i);
b(i)=y(i);
end
figure(3);
stem(a,b);
for i=1:total-t
c(i)=p(i);
modx=mod(i-1,t);
if (modx==0)
d(i)=b(i);
else
y1=b(i-modx);
temp=i+(t-modx);
y2=b(temp);
d(i)=y1+((y2-y1)*(1/t)*modx);
end
end
hold on;
figure(4);
plot(c,d,'g');
```

```
for i=1:t
j1(i)=i-1;
temp1=(i-1)*(180/t);
k1(i)=sin(pi*temp1/180);
end
figure(5);
stem(j1,k1);
for i=1:total-(2*t)
c1(i)=p(i);
modx=mod(i-1,t);
if i¡=t
if (modx==0)
d1(i)=b(i);
else
y1=b(i-modx);
temp=i+(t-modx);
y2=b(temp);
d1(i)=y1+((y2-y1)*(1/t)*modx);
end
else
if (modx==0)
d1(i)=b(i);
else
y1=b(i-(modx+t));
y2=b(i-modx);
temp=i+(t-modx);
y3=b(temp);
y4=b(temp+t);
y11=(y3-y1)/(2*t);
y21=(y3-y2)/t;
y31=(y4-y2)/(2*t);
dely=(((y11-y21)/2)-((y31-y21)/2));
dely1=(dely*k1(modx));
dely2=((((y11-y21)/2)+((y31-y21)/2))/2)*k1(modx);
d1(i)=y2+((y3-y2)*(1/t)*modx)+((t/4)+4)*dely1+dely2;
end
end
end
plot(c1,d1,'r');
for i=1:total-(2*t)
d2(i)=y(i)-d1(i);
d3(i)=y(i)-d(i);
end
figure(7);
plot(c1,d2);
```

```
figure(8);
plot(c1,d3);
```

A.2 2-Dimensional LUT algorithm

```
clc;
clear all;
[j,k] = meshgrid(-3:.1:3);
l = j.^2 + k.^2;
figure(1);clf;
mesh(j,k,l);
xlabel('x');
ylabel('y');
zlabel('z=x^2 + y^2');
total1=320;
t=40;
v=0;
for i=1:total1
x(i)=i-total1/2;
for j=1:total1
y(j)=j-total1/2;
z(i,j) = x(i).^2 - y(j).^2;
end
end
figure(1);clf;
mesh(x,y,z);
clc; clear all;
for i=1:20:500
a(i)=i-250;
for j=1:20:500
b(j)=j-250;
c(i,j) = a(i).^2 + b(j).^2;
end
end
figure(3);clf;
mesh(a,b,c);
for i=1:t:total1
a(i)=x(i);
v=v+1;
for j=1:t:total1
b(j)=y(j);
c(i,j) = z(i,j);
end
end
figure(2);clf;mesh(a,b,c);
```

```
for i=1:t
j1(i)=i-1;
temp1=(i-1)*(180/t);
k1(i)=sin(pi*temp1/180);
end
for i=1:t
j1(i)=i-1;
temp1=(i-1)*(180/t);
k2(i)=sin(2*pi*temp1/180);
end
figure(3);clf;
plot(j1,k1);
for i=1:(total1-(2*t))
d1(i)=x(i);
for j=1:(total1-(2*t))
e1(j)=y(j);
if i¡=t — j¡=t
flag=0;
modx=mod(i-1,t);
mody=mod(j-1,t);
if (modx =0)—(mody =0)
a1=c(i-modx,j-mody);
a2=c(i+(t-modx),j-mody);
a3=c(i-modx,j+(t-mody));
a4=c(i+(t-modx),j+(t-mody));
k=(1/t)*modx;
l=(1/t)*mody;
flag=1;
end
if flag==1
f1(i,j)=a1*(1-k)*(1-l) + a3*(l)*(1-k) + a4*(k)*(l) +a2*(k)*(1-l);
else
f1(i,j)=c(i,j);
end
else
flag=0;
modx=mod(i-1,t);
mody=mod(j-1,t);
if (modx =0)—(mody =0)
a1=c(i-modx,j-mody);
a2=c(i+(t-modx),j-mody);
a3=c(i-modx,j+(t-mody));
a4=c(i+(t-modx),j+(t-mody));
k=(1/t)*modx;
l=(1/t)*mody;
```

```
a11=c(i-modx-t,j-mody);
a12=c(i-modx,j-mody-t);
a21=c(i+(t-modx),j-mody-t);
a22=c(i+(t-modx)+t,j-mody);
a31=c(i-modx-t,j+(t-mody));
a32=c(i-modx,j+(t-mody)+t);
a41=c(i+(t-modx),j+(t-mody)+t);
a42=c(i+(t-modx)+t,j+(t-mody));
slop11=(a2-a11)/(2*t);
slop21=(a22-a1)/(2*t);
slop31=(a4-a31)/(2*t);
slop41=(a42-a3)/(2*t);
slop12=(a3-a12)/(2*t);
slop22=(a32-a1)/(2*t);
slop32=(a4-a21)/(2*t);
slop42=(a41-a2)/(2*t);
mslop12=(a2-a1)/t;
mslop34=(a4-a3)/t;
mslop13=(a3-a1)/t;
mslop24=(a4-a2)/t;
dely1=(((slop11-mslop12)/2)-((slop21-mslop12)/2))*k1(modx+1);
dely2=(((slop31-mslop34)/2)-((slop41-mslop34)/2))*k1(modx+1);
delz1=(((slop12-mslop13)/2)-((slop22-mslop13)/2))*k1(mody+1);
delz2=(((slop32-mslop24)/2)-((slop42-mslop24)/2))*k1(mody+1);
dely11=(((slop11-mslop12)/2)+((slop21-mslop12)/2))*k2(modx+1);
dely21=(((slop31-mslop34)/2)+((slop41-mslop34)/2))*k2(modx+1);
delz11=(((slop12-mslop13)/2)+((slop22-mslop13)/2))*k2(mody+1);
delz21=(((slop32-mslop24)/2)+((slop42-mslop24)/2))*k2(mody+1);
flag=1;
end
if flag==1
y1=a1+((a2-a1)*(1/t)*modx)+(t/4)*dely1;
y2=a3+((a4-a3)*(1/t)*modx)+(t/4)*dely2;
z1=a1+((a3-a1)*(1/t)*mody)+(t/4)*delz1;
z2=a2+((a4-a2)*(1/t)*mody)+(t/4)*delz2;
f1(i,j)=(y1+y2+z1+z2)/4;
avg1=(dely1+dely2+delz1+delz2)/4;
avg2=(dely11+dely21+delz11+delz21)/4;
f1(i,j)=(a1*(1-k)*(1-l) + a3*(l)*(1-k) + a4*(k)*(l) +a2*(k)*(1-l))+(((t/4)+10)*avg1);
else
f1(i,j)=c(i,j);
end
end
end
end
```

```
figure(4); clf;
mesh(d1,e1,f1);
error calculation........................................
for i=1:total1-2*t
for j=1:total1-2*t
err(i,j)=f1(i,j)-z(i,j);
end
end
figure(5);clf;
mesh(d1,e1,err);
```