FUZZY LOGIC BASED FRICTION COMPENSATOR FOR ROTARY MECHANICAL SYSTEMS

A Thesis submitted

in

the partial fulfillment of the requirement for the award of the degree of

MASTER OF ENGINEERING

in

CAD/CAM & ROBOTICS

Submitted by
Gurpreet Singh Chahil
Roll No. 8038109

Under the guidance of

Dr. Yaduvir Singh
Assistant Professor(EIED)
Thapar Institute of Engg. & Tech
Patiala-147004 (Punjab)

Sh. J. S. Saini
Lecturer(MED)
Thapar Institute of Engg. & Tech
Patiala-147004 (Punjab)

MECHANICAL ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA-147004 (PUNJAB)

2005

ACKNOWLEDGEMENT

I am deeply indebted to my guide **Dr. Yaduvir Singh, Assistant Professor, Electronics and Instrumentation Engineering Department** whose expert guidance helped me in completion of the entire work of thesis.

I am thankful to **Sh. J. S. Saini, Lecturer, Mechanical Engineering Department, TIET, Patiala,** for his valuable suggestions and kind encouragement at all stages of this thesis work.

I am also thankful to all my friends who helped me in all possible ways towards successful completion of my thesis work. I also take the opportunity to thank entire faculty and staff of **MED, TIET PATIALA,** for their help, inspiration and moral support, which went a long way in successful completion of my thesis.

(Gurpreet Singh Chahil)

ABSTRACT

Very accurate positioning is required in mechanical devices such as surgical tools, computers disk drives assembly robots, micro-mechanisms etc. For many of these devices the performance is limited by friction, which may induce undesirable tracking error, limit cycle and stick-slip motion. The negative effect of the friction is more significant at low velocities. Precise positioning, in particular, control of very small displacement is an especially difficult problem for direct drive mechanism. These devices have non transmission between drive and the output motion. The absence of transmission mechanism combined with the existence of non-linear friction often creates problems with precise positioning.

Friction parameters vary with load as well as with many environmental factors; therefore, it is always essential to detect the friction in the moving parts at very low velocity. In this thesis a fuzzy logic approach is presented to predict the non-linear (stick-slip) friction in the mechanical devices.

CONTENTS

CHAPTER 1

FRICTION

1.1 Introduction

The force of friction is a common but complex force. The exact method by which friction works is still a topic of great scientific interest but we can make some general statements about it. We do know that it arises from the electromagnetic forces between atoms and molecules at the surfaces of objects.

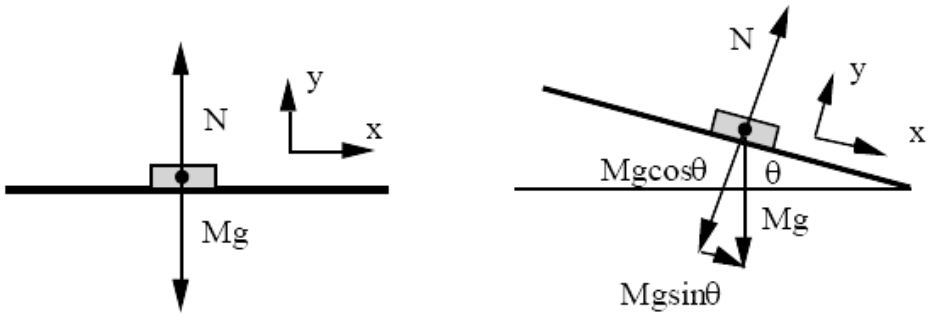
We can build a simple model of the friction force that is useful in many situations. The model friction force has the following properties:

- There are two types of frictional force. The force of static friction and the force of kinetic friction.
- The direction of the static frictional force is along the contact surface and opposite in direction of any applied force.
- The direction of the kinetic frictional force is opposite the direction of motion of the object it acts on.
- The coefficients of friction depend on the nature of the surface.
- The frictional force is nearly independent of the contact area between the objects.
- The kinetic friction force is usually less than the maximum static friction force.

1.2 Force of Friction:

We begin with an observation. We place an object with a mass M on a horizontal surface. Since the object doesn't move, the sum of the forces in the x and y directions must be zero. There are no forces on the object in the horizontal direction. In the y-direction we have N-Mg =0, that is the normal force is equal and opposite to the weight of the object. We now start to tip the surface at a small

angle θ. We keep our x-direction in the direction of the planar surface. Again, we write down the forces in the x and y directions



$$\sum F_Y = N - Mg\cos\theta = 0$$

$$\sum F_x = Mg\sin\theta \neq 0$$

1.3 Static Friction

The equation for F_x says that the force is nonzero so we should immediately observe the object to accelerate down the plane, no matter how small the angle θ. This is almost never observed. What we observe is that the object remains stationary on the plane. Therefore, there must be a force in the x-direction that we have not considered. This force is the force of friction. Our force equation in the x-direction becomes: -

$$\sum F_x = Mg\sin\theta - F_f$$

Where F_f is force of friction

Now, what happens as we make the angle steeper? By increasing the angle, we increase the force in the x-direction. (Remember, the force in the x-direction is Mg*sin*θ which gets larger as we increase the angle). However, the block remains stationary until we reach a critical angle θ_c. For angles greater than θ_c, the block will accelerate down the plane. What does this say about the force of friction for angles less than θ_c? This means that the force of friction must also increase as the angle increases in order to balance the component of the force in the positive x-direction.

If the object is stationary, the frictional force adjusts to balance the other horizontal forces acting on the object. This frictional force is usually called the static friction.

1.4 Kinetic Friction

We noted above that as we increase the inclination angle, there will be some point where the component of the force in the direction of motion becomes larger than the force of friction. This means that there must be some maximum value for the magnitude of static friction between two given objects. Once this maximum value is exceeded, the object begins to move down the plane.

You would think that at the critical angle, the force of friction would be balanced by the force acting down the plane. In this case, the object would move down the plane at a constant velocity. However, this is not what is usually observed. What is observed is that the object accelerates down the plane. This is because the force of friction actually decreases somewhat, once the object begins to move over the surface. We call the force of friction on a moving object its kinetic friction.

1.5 Relationship between the Kinetic Frictional Force and the Normal Force

The force of friction depends on the contact forces between the object and the surface that it moves upon. We call this force of contact the normal force. The normal force is always perpendicular to the surface of motion. While the force of kinetic friction can be quite complicated, it has been found that in most cases, the magnitude of the Kinetic Frictional Force is simply proportional to the magnitude of the Normal Force. Therefore, we can write

$$F_f = \mu_k N \quad (\textit{magnitude only})$$

where μ_k is the coefficient of kinetic friction. The coefficient of kinetic friction is simply a number, (i.e. it is dimensionless), and that depends on the nature of the surface and object. The Force of Kinetic Friction is always in the opposite direction of the motion.

1.6 Relationship between the Static Frictional Force and the Normal Force

The force of static friction also depends on the Normal force. Since the static force can change depending on the applied force, it is difficult to write an

equation that applies in all cases. However, there is a relationship between the normal force and the maximum force of static friction. This relationship is

$$F_s(\text{max.}) = \mu_s N$$

where F_s(max) means the maximum static frictional force and μ_s is the coefficient of static friction. Since this is the maximum force of static friction, we always know that

$$F_s \leq \mu_s N$$

1.7 The Coefficient of Kinetic and Static Friction

The coefficients of friction are numbers which are determined from experiment. They are material parameters, which means that the number we get depends on what the material we consider. We can't say that the coefficient of friction for steel is 0.53 or the coefficient for glass is 0.27. This is because μ_k depends on both the object and the planar surface. The coefficient of steel sliding on a piece of glass is not the same as for steel on wood. The coefficient of friction also depends on the surface textures of the object and plane.

The coefficient of static friction also depends on the materials and surface textures of the objects in contact. Since we know that the force of friction decreases somewhat when the object is in motion, we know that

$$\mu_k < \mu_s$$

Coulomb's laws of dry friction have been well known for over 200 years. They state that the friction force is given by a material parameter (friction coefficient) times the normal force. The coefficient of static friction (i.e. the force necessary to start sliding) is always equal to or larger than the coefficient of kinetic friction (i.e. the force necessary to keep sliding at a constant velocity). The dynamical behavior of a mechanical system with dry friction is nonlinear because Coulomb's laws distinguish between static friction and kinetic friction. If the kinetic friction coefficient is less than the static one, stick–slip motion occurs where the sliding surfaces alternately switch between sticking and slipping in a more or less regular fashion. This jerky motion leads to the everyday experience of squeaking doors and singing violins.

Even though Coulomb's laws are simple and well established (many calculations in engineering rely on these laws), they cannot be derived in a rigorous way because dry friction is a process which operates mostly far from equilibrium. It is therefore no surprise that deviations from Coulomb's laws have often been found in experiments. Typical deviations are as follows:

(i) Static friction is not constant but increases with the sticking time i.e. the time since the two sliding surfaces have been in contact without any relative motion.

(ii) Kinetic friction depends on the sliding velocity; for very large velocities, it increases roughly linearly with the sliding velocity like in viscous friction.

Coming from large velocities, the friction first decreases, goes through a minimum, and then increases. In the case of boundary lubrication (i.e. a few mono-layers of some lubricant are between the sliding surfaces) it decreases again for very low velocities (see figure). The coefficient of kinetic friction as a function of the sliding velocity therefore has at least one extremism. The kinetic friction can exceed the static friction, but in the limit of zero sliding velocity it is still less than or equal to the static friction.

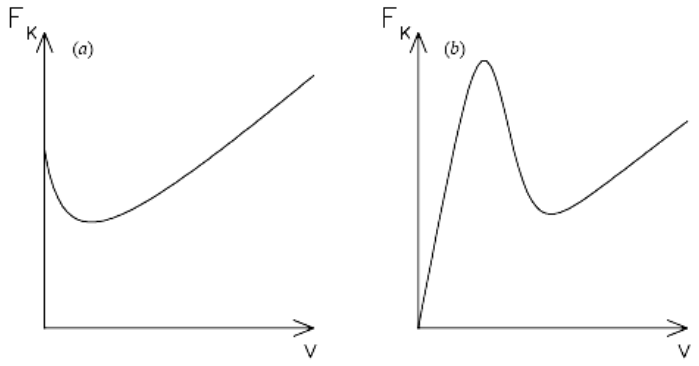


Figure: - schematically sketches of typical velocity-dependent kinetic friction laws for (a) systems without and (b) systems with boundary lubrication.

CHAPTER 2

LITERATURE SURVEY

In order to design the advanced fuzzy logic based friction predictor for rotational mechanical systems, first we have to study the influence of friction on the performance of rotational systems with slow speed, the literature related to stick-slip friction have been studied.

M R Popovic et. al. [1]; described a new approach to very accurate positioning of mechanical devices with nonlinear stick-slip friction is presented in this paper. The proposed controller applies narrow torque pulses to move a mechanism to a desired position despite nonlinear friction. The pulse shapes generated by the controller are computed using a fuzzy logic approximation of the dependence between the desired displacement and the torque pulse shape. The closed loop stability conditions for the proposed controller are derived taking into consideration a random variation of friction. A detailed experimental study of the system response to different torque pulse shapes and a detailed controller design are presented for a direct drive mechanism. In the experiments it was demonstrated that the proposed controller achieves positioning accuracy which is within the limits of the position encoder resolution. Very accurate positioning is required in mechanical devices such as surgical tools computer disk drives assembly robots micro-mechanisms etc. For many of these devices the performance is limited by friction caused positioning errors. Precise positioning in particular control of very small displacements is an especially difficult problem for direct drive mechanisms. These devices have no transmission between the drive and the output motion absence of a transmission mechanism combined with the existence of nonlinear stick-slip friction often creates problems with precise positioning.

Brian Armstrong [2]; Explained that how friction plays a large roll in mechanical motion. Brush type dc servo-motors typically have a break-away friction that is

4% to 6% of the full rated torque of the motor. This sets a lower bound on the friction observed in any mechanism employing these actuators. The friction behavior of a brush type d.c. servo-motor driven mechanism with gearing is explored. The standard kinetic plus viscous friction model is found to describe the dominant friction behavior. In addition, a dependence of friction upon position is identified and negative velocity dependence is observed at low velocities. A friction model is developed and used to pre-compute motion torques. The application of pre-computed torques in an open loop fashion results in motions accurate to within a few percent. Furthermore, hard non-linearities and possibly destabilizing friction effects at low velocity compound the problems of control.

Brian Armstrong et. al. [3]; Machines at low velocity exhibit stick-slip motion. This phenomenon is of particular importance in force control of robot manipulators because much of the force correcting action can occur in the low velocity regime of stick-slip. Experimental work has mapped out the structure of Stribeck friction, a non-linear low-velocity friction effect that contributes to and perhaps dominates stick-slip. In this paper the implications of Stribeck friction for feedback control are explored. Through the use of dimensional analysis, the following are examined:

- The minimum velocity below which stick-slip will occur;
- Accuracy of sensing required eliminating stick-slip
- The slip distance during stick-slip motion;
- Scaling of Stribeck friction and stick-slip.

Pierre E. Dupont [4]; discusses the importance of appropriately modeling friction for the simulation and control of high performance robot systems. Incorporating Coulomb type friction in the dynamic equations introduces two difficulties in the forward dynamic solution. We show that the differential equations become discontinuous in the highest order derivative terms. In addition, the load dependency of this type of friction generally causes the equations to be implicit in the joint accelerations. For the important case of load-dependent

transmission friction, the equations are shown to be explicit. Techniques for the forward solution are described through the example of a roller screw transmission. Experimental and simulation results are used to show the importance of load-dependent friction in a particular robot. Implementation issues are discussed as well as implications for robot control.

H. Rachoor [5]; developed a friction model for lubricated contacts. A few analysis models have been developed to investigate the friction under dynamic velocity conditions. In this study, two different tribological situations such as conformal and non-conformal contacts have been chosen. Friction modeling covers boundary, mixed and fluid film friction regions. A new theory based on the elastic properties of the surface materials, and fluid film properties of the lubricant at the contact has been developed to determine the dynamic friction in the boundary, mixed and full hydrodynamic regions. In the full fluid film lubrication regions, friction has been determined from the lubrication principals based on the tribological situations, i.e., hydrodynamic lubrication theory for a short journal bearing and elasto-hydrodynamic lubrication theory for a line contact. A conformal contact formed by a short journal bearing operating in region where hydrodynamic lubrication theory is valid has been considered to develop a model.

C. Canudas et. al. [6]; described a new dynamic model for friction. The model captures most of the friction behavior that has been observed experimentally. This includes the Stribeck effect, hysteresis, spring-like characteristics for stiction, and varying break away force. Properties of the model that are relevant to control design are investigated by analysis and simulation. New control strategies, including a friction observer, are explored, and stability results are presented. Friction is an important aspect of many control systems both for high quality servo mechanisms and simple pneumatic and hydraulic systems. Friction can lead to tracking errors, limit cycles, and undesired stick-slip motion. Control strategies that attempt to compensate for the effects of friction, without resorting to high gain control loops, inherently require a suitable friction model to predict and to compensate for the friction. These types of schemes are therefore named model-

based friction compensation techniques. A good friction model is also necessary to analyze stability, predict limit cycles, find controller gains, perform simulations, etc. Most of the existing model-based friction compensation schemes use classical friction models, such as Coulomb and viscous friction. In applications with high precision positioning and with low velocity tracking, the results are not always satisfactory. A better description of the friction phenomena for low velocities and especially when crossing zero velocity is necessary. Friction is a natural phenomenon that is quite hard to model, and it is not yet completely understood. The classical friction models used are described by static maps between velocity and friction force. Typical examples are different combinations of Coulomb friction, viscous friction, and Stribeck effect. The latter is recognized to produce a destabilizing effect at very low velocities. The classical models explain neither hysteretic behavior when studying friction for non-stationary velocities nor variations in the break-away force with the experimental condition or small displacements that occur at the contact interface during stiction. The latter very much resembles that of a connection with a stiff spring with damper and is sometimes referred to as the Dahl effect. Studies have shown that a friction model involving dynamics is necessary to describe the friction phenomena accurately.

Jean Claude et. al. [8]; proposes an algorithm for joint friction in robot manipulator simulations. The model includes an appropriate representation of the behavior at low speed and, specially, the stick-slip process. The concept of the model is introduced through two simple examples. The method is then extended to more complex multi-body systems and, in particular, to robot manipulators. It is further synthesized into a numerical algorithm for computer simulations. The algorithm is experimentally validated using a two-degrees-of-freedom planar robot moving in the vertical plane. Free motions resulting from known initial conditions were successfully reproduced.

CHAPTER 3
FUZZY LOGIC

3.1 Introduction

Fuzzy logic is a powerful problem-solving methodology with a myriad of applications in embedded control and information processing. Fuzzy provides a remarkably simple way to draw definite conclusions from vague, ambiguous or imprecise information. In a sense, fuzzy logic resembles human decision making with its ability to work from approximate data and find precise solutions.

Unlike classical logic which requires a deep understanding of a system, exact equations, and precise numeric values, Fuzzy logic incorporates an alternative way of thinking, which allows modeling complex systems using a higher level of abstraction originating from our knowledge and experience. Fuzzy Logic allows expressing this knowledge with subjective concepts such as very hot, bright red, and a long time which are mapped into exact numeric ranges.

Fuzzy Logic has been gaining increasing acceptance during the past few years. There are over two thousand commercially available products using Fuzzy Logic, ranging from washing machines to high speed trains. Nearly every application can potentially realize some of the benefits of Fuzzy Logic, such as performance, simplicity, lower cost, and productivity.

Fuzzy Logic has been found to be very suitable for embedded control applications. Several manufacturers in the automotive industry are using fuzzy technology to improve quality and reduce development time. In aerospace, fuzzy enables very complex real time problems to be tackled using a simple approach. In consumer electronics, fuzzy improves time to market and helps reduce costs. In manufacturing, fuzzy is proven to be invaluable in increasing equipment efficiency and diagnosing malfunctions.

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth values between "completely true" and "completely false". As its name suggests, it is the logic underlying modes of reasoning which are approximate rather than exact. The importance of fuzzy logic derives from the fact that most modes of human reasoning and especially common sense reasoning are approximate in nature.
The essential characteristics of fuzzy logic as founded by Zader Lotfi are as follows.

- In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning.
- In fuzzy logic everything is a matter of degree.
- Any logical system can be fuzzified.
- In fuzzy logic, knowledge is interpreted as a collection of elastic or, equivalently , fuzzy constraint on a collection of variables
- Inference is viewed as a process of propagation of elastic constraints.

The third statement hence, defines Boolean logic as a subset of Fuzzy logic.

3.2 Fuzzy Sets

Fuzzy Set Theory was formalized by Professor Lofti Zadeh at the University of California in 1965. What Zadeh proposed is very much a paradigm shift that first gained acceptance in the Far East and its successful application has ensured its adoption around the world.

A paradigm is a set of rules and regulations which defines boundaries and tells us what to do to be successful in solving problems within these boundaries. For example the use of transistors instead of vacuum tubes is a paradigm shift - likewise the development of Fuzzy Set Theory from conventional bivalent set theory is a paradigm shift.

Bivalent Set Theory can be somewhat limiting if we wish to describe a 'humanistic' problem mathematically. For example, Figure 3.1 illustrates bivalent sets to characterize the temperature of a room.

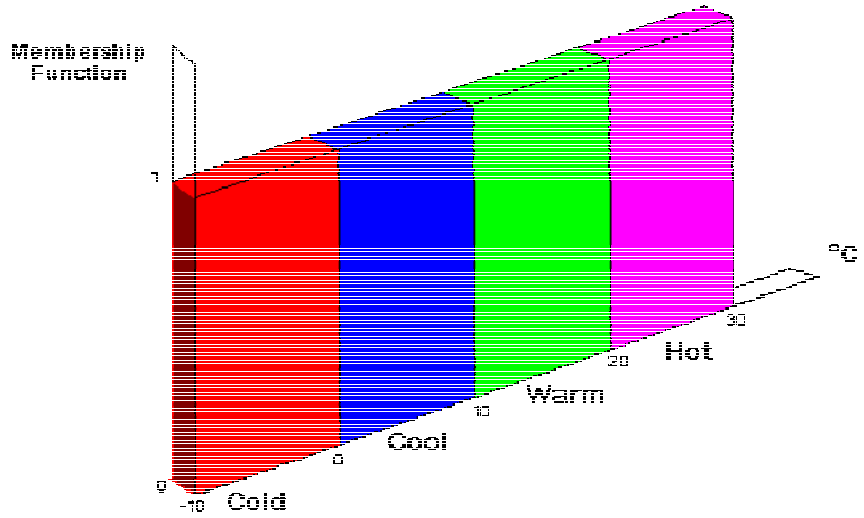


Figure 3.1

The most obvious limiting feature of bivalent sets that can be seen clearly from the diagram is that they are mutually exclusive - it is not possible to have membership of more than one set (opinion would widely vary as to whether 50 degrees Fahrenheit is 'cold' or 'cool' hence the expert knowledge we need to define our system is mathematically at odds with the humanistic world). Clearly, it is not accurate to define a transition from a quantity such as 'warm' to 'hot' by the application of one degree Fahrenheit of heat. In the real world a smooth (unnoticeable) drift from warm to hot would occur.

This natural phenomenon can be described more accurately by Fuzzy Set Theory. Figure 3.2 shows how fuzzy sets quantifying the same information can describe this natural drift.

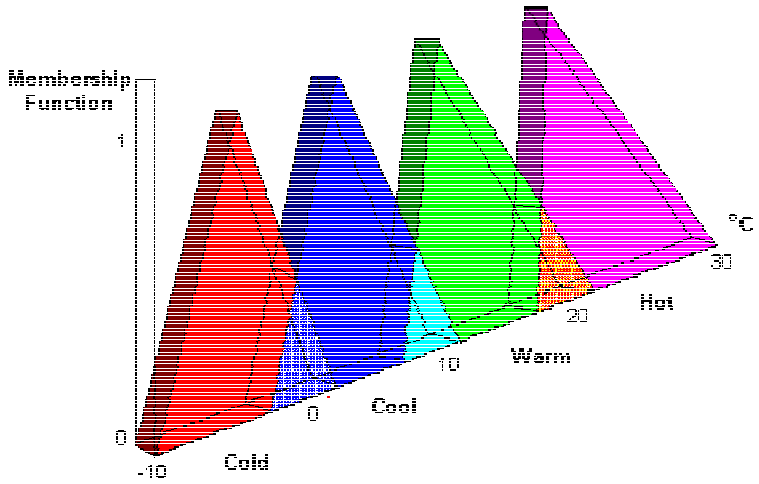
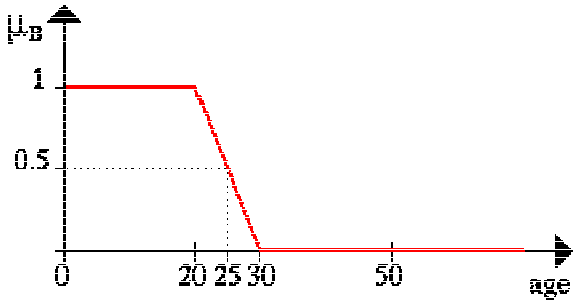


Figure 3.2

The whole concept can be illustrated with this example. Let's talk about people and "youth ness". In this case the set S (the universe of discourse) is the set of people. A fuzzy subset YOUNG is also defined, which answers the question "to what degree is person x young?" To each person in the universe of discourse, we have to assign a degree of membership in the fuzzy subset YOUNG. The easiest way to do this is with a membership function based on the person's age.

$$Young(x) = \begin{cases} 1, & if\ age(x) <= 20; \\ (30 - age(x))/10, & if\ 20 < age(x) <= 30; \\ 0, & if\ age(x) > 30; \end{cases}$$

A graph of this kind looks like:



Given this definition, here are some example values:

Person	Age	Degree of Youth
Anil	10	1.00
Sachin	21	0.90
Rajeev	25	0.50
Kulbhushan	26	0.40
Ashwani	28	0.20
Yashpal	83	0.00

So given this definition, we'd say that the degree of truth of the statement "Rajeev is YOUNG" is 0.50.

Membership functions almost never have as simple a shape as age(x). They will at least tend to be triangles pointing up, and they can be much more complex than that. Furthermore, a membership function so far is discussed as if they always are based on a single criterion, but this isn't always the case, although it is the most common case. One could, for example, want to have the membership function for YOUNG depend on both a person's age and their height (Kulbhushan's short for his age). This is perfectly legitimate, and occasionally used in practice. It's referred to as a two-dimensional membership function. It's also possible to have even more criteria, or to have the membership function depend on elements from two completely different universes of discourse.

3.3 Fuzzy Set Operations

3.3.1 Union

The membership function of the Union of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the maximum of the two individual membership functions. This is called the maximum criterion.

$$\mu_{A \cup B} = \max(\mu_A, \mu_B)$$

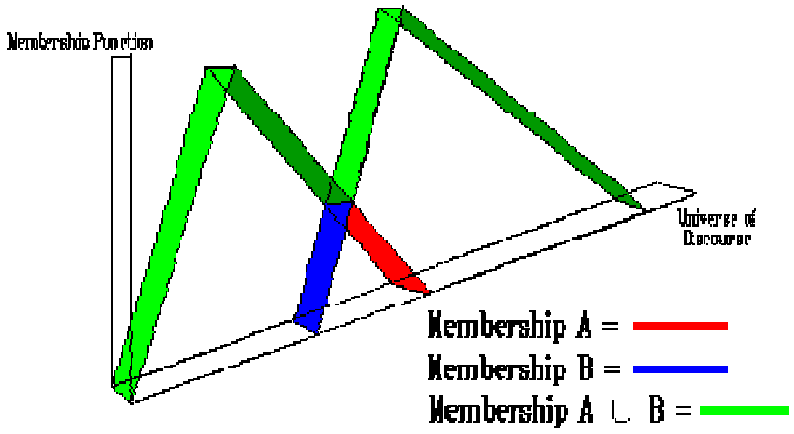


Figure 3.3

The Union operation in Fuzzy set theory is the equivalent of the OR operation in Boolean algebra.

3.3.2 Intersection

The membership function of the Intersection of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the minimum of the two individual membership functions. This is called the minimum criterion.

$$\mu_{A \cap B} = \min(\mu_A, \mu_B)$$

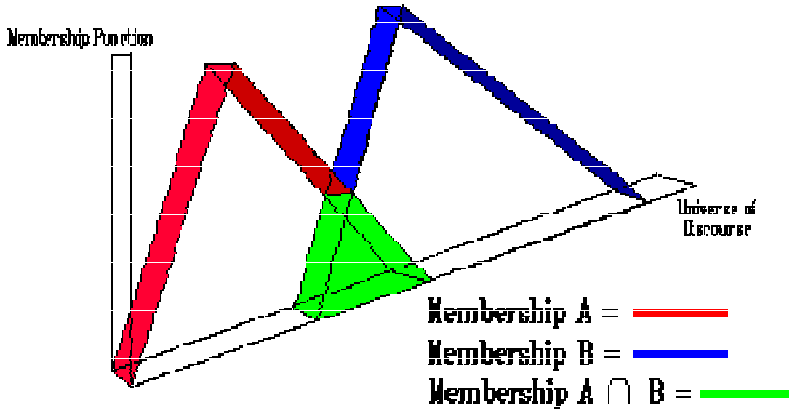


Figure 3.4

The Intersection operation in Fuzzy set theory is the equivalent of the AND operation in Boolean algebra.

3.3.3 Complement

The membership function of the Complement of a Fuzzy set A with membership function μ_A is defined as the negation of the specified membership function. This is called the negation criterion.

$$\mu_{\overline{A}} = 1 - \mu_A$$

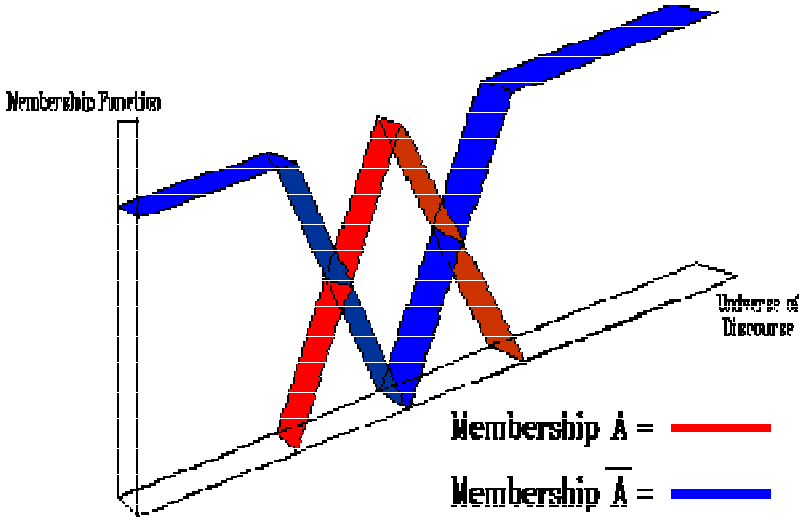


Figure 3.5

The Complement operation in Fuzzy set theory is the equivalent of the NOT operation in Boolean algebra.

3.4 Fuzzy Rules

Human beings make decisions based on rules. Although, we may not be aware of it, all the decisions we make are all based on computer like if-then statements. If the weather is fine, then we may decide to go out. If the forecast says the weather will be bad today, but fine tomorrow, then we make a decision not to go today, and postpone it till tomorrow. Rules associate ideas and relate one event to another.

Fuzzy machines, which always tend to mimic the behavior of man, work the same way. However, the decision and the means of choosing that decision are replaced by fuzzy sets and the rules are replaced by fuzzy rules. Fuzzy rules also operate using a series of if-then statements. For instance, if X then A, if y then b, where A and B are all sets of X and Y.

The following rules which are common in classical set theory also apply to Fuzzy set theory.

De Morgan's law

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$$

Associative Law

$$(A \cap B) \cap C = A \cap (B \cap C)$$

$$(A \cup B) \cup C = A \cup (B \cup C)$$

Commutative Law

$$A \cap B = B \cap A \, , \, A \cup B = B \cup A$$

Distributive Law

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

3.5 Fuzzy Control

Fuzzy control, which directly uses fuzzy rules, is the most important application in fuzzy theory. Using a procedure originated by Ebrahim Mamdani in the late 70s, three steps are taken to create a fuzzy controlled machine:

1. Fuzzification (Using membership functions to graphically describe situation).
2. Rule evaluation (Application of fuzzy rules).
3. Defuzzification (Obtaining the crisp or actual results).

3.6 Advantages of Fuzzy Logic

In order to appreciate why a fuzzy based design methodology is very attractive in embedded control applications let us examine a typical design flow Fig 3.6. Illustrates a sequence of design steps required to develop a controller using a conventional and a Fuzzy approach.

Using the conventional approach our first step is to understand the physical system and its control requirements. Based on this understanding, our second step is to develop a model which includes the plant, sensors and actuators.

Conventional Design methodology	**Fuzzy Based Design Methodology**

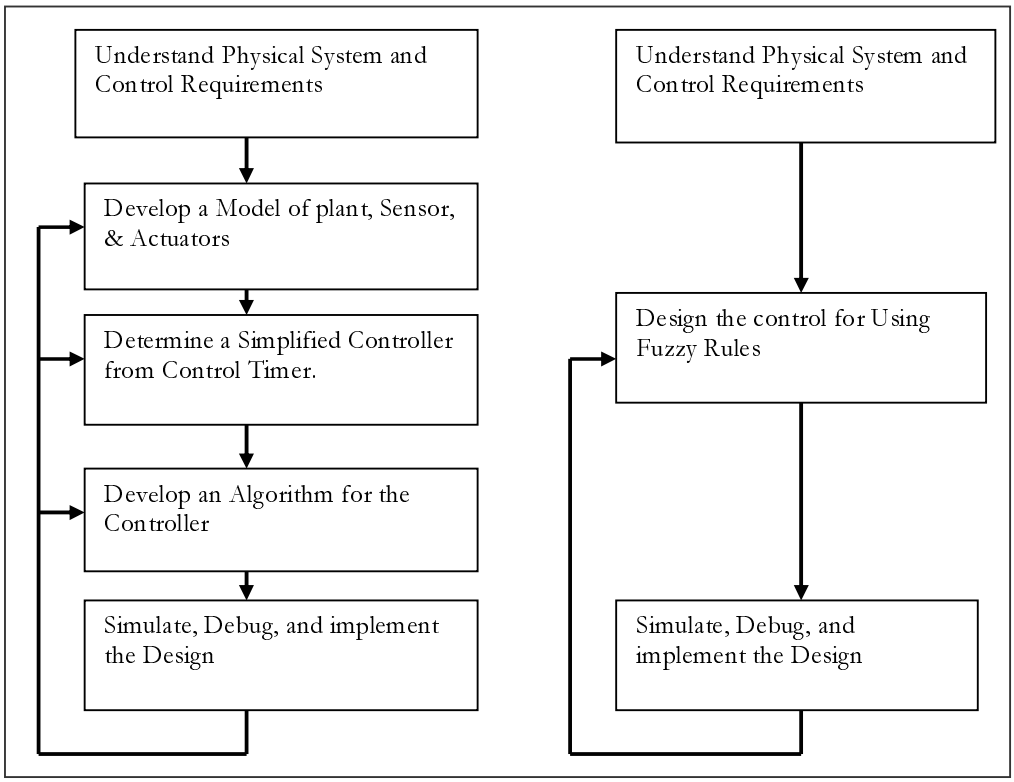


Figure 3.6

The third step is to use linear control theory in order to determine a simplified version of the controller, such as the parameters of a PID controller. The fourth step is to develop an algorithm for the simplified controller. The last step is to simulate the design including the effects of non-linearity, noise, and parameter variations. If the performance is not satisfactory we need to modify our system modeling, re-design the controller, re-write the algorithm and re-try.

With Fuzzy Logic the first step is to understand and characterize the system behavior by using our knowledge and experience. The second step is to directly design the control algorithm using fuzzy rules, which describe the principles of the controller's regulation in terms of the relationship between its inputs and outputs. The last step is to simulate and debug the design. If the performance is not satisfactory we only need to modify some fuzzy rules and re-try.

Although the two design methodologies are similar, the fuzzy-based methodology substantially simplifies the design loop. This results in some significant benefits, such as reduced development time, simpler design and faster time to market:

3.6.1 Fuzzy Logic reduces the design development cycle

With a fuzzy logic design methodology some time consuming steps are eliminated. Moreover, during the debugging and tuning cycle you can change your system by simply modifying rules, instead of redesigning the controller. In addition, since fuzzy is rule based, you do not need to be an expert in a high or low level language which helps you focus more on your application instead of programming. As a result, Fuzzy Logic substantially reduces the overall development cycle.

3.6.2 Fuzzy Logic simplifies design complexity

Fuzzy logic lets you describe complex systems using your knowledge and experience in simple English-like rules. It does not require any system modeling or complex math equations governing the relationship between inputs and outputs. Fuzzy rules are very easy to learn and use, even by non-experts. It typically takes only a few rules to describe systems that may require several of lines of conventional software. As a result, Fuzzy Logic significantly simplifies design complexity.

3.6.3 Fuzzy Logic improves time to market

Commercial applications in embedded control require a significant development effort a majority of which is spent on the software portion of the project. Development time is a function of design complexity, and the number of iterations required in a debugging and tuning cycle. As we explained above, a fuzzy based design methodology addresses both issues very effectively. Moreover, due to its simplicity the description of a fuzzy controller not only is transportable across design teams, but also provides a superior media to preserve, maintain, and upgrade intellectual property. As a result, Fuzzy Logic can dramatically improve time to market.

3.7 Artificial Neural Networks

An Artificial Neural Network ANN is an information-processing paradigm inspired by the way the brain processes information. ANNs are collections of mathematical models that emulate some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. An ANN is composed of a large number of highly interconnected processing elements that are analogous to neurons and are tied together with weighted connections that are analogous to synapses.

ANNs are applied to control problems where the input variables are measurements used to drive an output actuator, and the network learns the control function. The advantages of ANNs lie in their resilience against distortions in the input data and their capability of learning. They are often good at solving problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found.

3.8 Overview

Neural Networks are an information processing technique based on the way biological nervous systems, such as the brain, process information. The fundamental concept of neural networks is the structure of the information processing system. Composed of a large number of highly interconnected processing elements or neurons, a neural network system uses the human-like technique of learning by example to resolve problems. The neural network is configured for a specific application, such as data classification or pattern recognition, through a learning process called training. Just as in biological systems, learning involves adjustments to the synaptic connections that exist between the neurons. Neural networks can differ on: the way their neurons are connected; the specific kinds of computations their neurons do; the way they transmit patterns of activity throughout the network; and the way they learn including their learning rate. Neural networks are being applied to an increasing large number of real world problems. Their primary advantage is that they can solve problems that are too complex for conventional technologies -- problems

that do not have an algorithmic solution or for which an algorithmic solution is too complex to be defined. In general, neural networks are well suited to problems that people are good at solving, but for which computers generally are not. These problems include pattern recognition and forecasting which requires the recognition of trends in data.

3.9 What Exactly Is A Neural Network?

A neural network is an interconnected group of artificial or biological neurons. It is possible to differentiate between two major groups of neural networks: Biological neural networks, for example the human brain or parts thereof. Artificial neural networks originally referred to electrical, mechanical or computational simulations or models of biological neural networks. The field has expanded so that some applications do not clearly resemble any existing biological counterpart.

A neural network is mans crude way of trying to simulate the brain electronically. Our brains are made up of about 100 billion tiny units called Neurons. Each neuron is connected to thousands of other neurons and communicates with them via electrochemical signals. Signals coming into the neuron are received via junctions called synapses; these in turn are located at the end of branches of the neuron cell called Dendrites. The neuron continuously receives signals from these inputs and then performs a little bit of magic. What the neuron does (this is over simplified I might add) is sum up the inputs to itself in some way and then, if the end result is greater than some threshold value, the neuron fires. It generates a voltage and outputs a signal along something called an Axon.

A neuron can have any number of inputs from one to n, where n is the total number of inputs. The inputs may be represented therefore as x1, x2, x3… xn. And the corresponding weights for the inputs as w1, w2, w3… wn. Now, the summation of the weights multiplied by the inputs we talked about above can be written as x1w1 + x2w2 + x3w3 …. + xnwn, is the activation value. So

a = x1w1+x2w2+x3w3... +xnwn

Fortunately there is a quick way of writing this down which uses the Greek capital letter sigma. Which is the symbol used by mathematicians to represent summation.

$$a = \sum_{i=0}^{i=n} w_i x_i$$

3.10 The Perceptron or a Network for Decision Making

An artificial neural network which attempts to emulate this pattern recognition process is called the Perceptron. In this model, the nodes representing artificial neurons are arranged into layers. The signal representing an input pattern is fed into the first layer. The nodes in this layer are connected to another layer (sometimes called the "hidden layer"). The firing of nodes on the input layer is conveyed via these connections to this hidden layer. Finally, the activity on the nodes in this layer feeds onto the final output layer, where the pattern of firing of the output nodes defines the response of the network to the given input pattern. Signals are only conveyed forward from one layer to a later layer - the activity of the output nodes does not influence the activities on the hidden layer. In contrast to the Hopfield network, this network produces its response to any given input pattern almost immediately - the firing pattern of the output is automatically stable. There is no relaxation process to a stable firing pattern, as occurs with the Hopfield model.

To try to simplify things, we can think of a simple model in which the network is made up of two screens - the nodes on the first (input) layer of the network are represented as light bulbs which are arranged in a regular pattern on the first screen. Similarly, the nodes of the third (output) layer can be represented as a regular array of light bulbs on the second screen. There is no screen for the hidden layer - that is why it is termed "hidden"! Instead we can think of a black box which connects the first screen to the second. Of course, the magic of how the black box will function depends on the network connections between hidden

nodes which are inside. When a node is firing, we show this by lighting its bulb. We can now think of the network functioning in the following way: a given pattern of lit bulbs is set up on the first screen. This then feeds into the black box (the hidden layer) and results in a new pattern of lit bulbs on the second screen. This might seem a rather pointless exercise in flashing lights except for the following crucial observation. It is possible to "tweak" with the contents of the black box (adjust the strengths of all these internodes connections) so that the system can produce any desired pattern on the second screen for a very wide range of input patterns. For example, if the input pattern is a triangle, the output pattern can be trained to be a triangle. If an input pattern containing a triangle and a circle is presented, the output can be still arranged to be a triangle. Similarly, we may add a variety of other shapes to the network input pattern and teach the net to only respond to triangles. If there is no triangle in the input, the network can be made to respond, for example, with a zero.

3.11 The Multilayer Perceptron Model

The most common neural network model is the multilayer perceptron (MLP). This type of neural network is known as a supervised network because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. The MLP and many other neural networks learn using an algorithm called back propagation. With back propagation, the input data is repeatedly presented to the neural network. With each presentation the output of the neural network is compared to the desired output and an error is computed. This error is then fed back (back propagated) to the neural network and used to adjust the weights such that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output. This process is known as "training". Of course character recognition is not the only problem that neural networks can solve. Neural networks have been successfully applied to broad spectrum of data-intensive applications such as:

Process Modeling and Control - Creating a neural network model for a physical plant then using the modal to get the best settings of the plant.

Machine Diagnostics - Detect when a machine has failed so that the system can automatically shut down machines when this occurs.

Portfolio Management - Allocate the assets in a portfolio in a way that maximizes return and minimizes risk.

Target Recognition - Military application which uses video and/or infrared image data to determine if enemy target is present.

Medical Diagnosis - Assisting doctors with their diagnosis by analyzing the reported symptoms and/or image data such as MRIs or X-rays.

Credit Rating - Automatically assigning a company's or individual's credit rating based on their financial condition.

Targeted Marketing - Finding the set of demographics which have the highest response rate for a particular marketing campaign.

Voice Recognition - Transcribing spoken words into ASCII text.

Financial Forecasting - Using the historical data of a security to predict the future movement of that security.

Quality Control - Attaching a camera or sensor to the end of a production process to automatically inspect for defects.

Intelligent Searching - An internet search engine that provides the most relevant content and banner ads based on the users' past behavior.

Fraud Detection - Detect fraudulent credit card transactions and automatically decline the charge.

3.12 Neuro-control

The use of neural networks in control applications has recently experienced rapid growth. The basic objective of control is to provide the appropriate input signal to a given physical process in order to obtain a desired output. In control theory, the physical process to be controlled is referred to as a plant. If the input signal supplied to the plant is generated by a neural network based controller, the case can be termed "neural control" or briefly "Neuro-control". Neuro-control is part of the much broader concept of "intelligent control".

The intelligent control paradigm emerged quite naturally from the field of conventional control. As control methods have found their way into practice, they have opened the door to a wide spectrum of complex applications. Such complex systems are characterized by poor models, high dimensionality of the decision space, high noise levels, multiple performance criteria, complex behavior, etc. We can broadly classify the difficulties that arise in these systems into three categories for which established methods are insufficient. The first is computational complexity, the second is the presence of nonlinear processes with many degrees of freedom, and the third is uncertainty (presence of noise, disturbances, etc.). The greater the ability to deal with these difficulties, the more intelligent is the control system.

Because many living systems do implement some sort of intelligent control, it has been natural to look into computational paradigms used by nature. Artificial neural networks represent such a biologically inspired paradigm. At the present time, neural networks represent an important paradigm for classifying patterns, or generating signals to control their environment.

The main reasons for this are:

- Neural networks can be trained. This means that, given a set of input-output patterns (called the training set), the connection weights of the neural network can be adapted in order to approximate, according to some predefined criterion, the input-output patterns provided in the training set. After training, the neural network can be used to predict a new output patterns, based on the input pattern only. The adaptation law that allows adjusting the connection weights is called the learning algorithm. This is of course reminiscent of the adaptive control strategy.

- Neural networks can approximate nonlinear functions. Multi-layer neural networks (units or neurons are organized in layers: the input layer, the hidden layers, and the output layer) are special architectures for which powerful learning algorithms exist. Once the connection weights have been trained, the neural network approximates the input-output mapping provided by the

training set, provided that there are a sufficient number of hidden units in the hidden layers.

- Neural networks can be trained in noise. For instance, a neural network trained with a least mean square criterion approximates the conditional expectation of the output pattern, given the input pattern. There exist robust learning algorithms that guarantee convergence in the presence of uncertainties.
- Neural networks can easily be implemented on parallel hardware for fast computation. Indeed, they realize, by nature, a parallel processing of information.
- These properties clearly indicate that neural networks exhibit some intelligent behavior, and are good candidate models for the control of nonlinear processes, for which no perfect mathematical model is available.

3.13 Application of Artificial Neural Network

1. Game playing
2. Speech recognition
3. Understanding natural language
4. Computer vision
5. Expert systems

3.14 Advantages of Artificial Neural Networks (ANN)

Computer techniques can use neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, to extract patterns and detect trends that are too complex to be noticed. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest.

3.15 Limitation to Neural Networks

The major issues of concern today are the scalability problem, testing, verification, and integration of neural network systems into the modern environment. Neural network programs sometimes become unstable when applied to larger problems. The Defense, nuclear and space industries are concerned about the issue of testing and verification. The mathematical theories used to guarantee the performance of an applied neural network are still under development.

CHAPTER 4

MODELLING & SIMULATION

This chapter presents modeling approach in fuzzy tool box of MATLAB.

4.1 What is the Fuzzy logic toolbox?

The fuzzy logic toolbox is a collection of functions built on the MATLAB numeric computing environment. It provides tools for you to create and edit fuzzy inference systems within the framework of MATLAB, or if prefer you can integrated your fuzzy systems into simulations with SIMULINK, or you can even build standalone C Programs that call on Fuzzy you build MATLAB. This toolbox relies heavily on graphical user interface (GUI) tools to help you accomplish your work, although you can work entirely from the command line if you prefer.

4.2 The toolbox categories

- Command line functions.
- Graphical, interactive tools
- SIMULINK blocks and examples.

The first category of tools is made up of functions that you can call from the command line or from your own application.

Many of these functions are MATLAB m-files, series of MATLAB statements that implement specialized fuzzy logic algorithms. You can view the MATLAB code for these functions using the statement "Type function name".

You can change the way any toolbox function works by copying and renaming the m-file, then modifying your copy. You can also extend the toolbox by adding your own m-files.

Secondly, the toolbox provides a number of interactive tools that let you access many of ten functions through a GUI together, the GUI based tools provide an environment for fuzzy inference systems design, analysis, and implementation.

The third category of tools is a set of blocks for use with SIMULINK simulation software. These are specifically designed for high-speed fuzzy Logic inference in the SIMULINK environment.

4.3 What can fuzzy logic toolbox do?

The fuzzy logic toolbox allows you to do several things, but the most important thing it lets you do is create and edit fuzzy inference systems. You can create these systems using GUI or command line functions or you can generate them automatically using either clustering or adaptive neuro-fuzzy techniques.

If you have access to SIMULINK you can easily test your fuzzy system in block diagram simulation environment

The toolbox also lets you run your own standalone C programs directly, without the need of SIMULINK. This is made possible by a standalone fuzzy Inference Engine that reads the fuzzy systems saved from a MATLAB session. You can customize the standalone engine to build fuzzy inference into your own code. ALL provide code is ANSI compliant.

Because of the integrated nature of MAT Lab's environment, you can create your own tools to customize the Fuzzy Logic Toolbox or harness it with another toolbox, such as Control System, Neural, or Optimization Toolbox, to mention only a few of the possibilities.

4.4 Fuzzy Inference

It is method that interprets the values in the input vector and, based on some set of rules, assigns values to the output vector. These three areas are

- **Foundations of fuzzy logic**, which is an introduction to the general concepts.
- **Fuzzy inference system**, which explains the specific methods of fuzzy inference used in fuzzy logic toolbox. The field of fuzzy uses many terms that do not yet have standard interpretations.
- **Building systems with fuzzy logic toolbox**, this introduces the GUI tools available in fuzzy logic toolbox and guides you through the constructions of a complete inference system from start to finish. This also tells about how you build and edit a fuzzy system using this toolbox.

4.5. Foundations of fuzzy logic

Fuzzy Sets: Fuzzy logic starts with concept of a fuzzy set, which is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership.

Membership Functions: A membership function is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as Universe of Discourse, a fancy name for a simple concept.

1) Membership functions In a Fuzzy Logic Toolbox: The only condition a membership function must really satisfy is that it must vary between 0 & 1. The function itself can be an arbitrary curve whose shape weak and define as a function that suits us from the point of view of simplicity, convenience, speed, and efficiency.

2) If Then Rules: *Fuzzy sets and fuzzy operators are the subjects and the verbs of fuzzy logic. These if then rules statements are used to formulate the conditional statements that comprise fuzzy logic. A single fuzzy if then rule assumes the form,*

If X is A then Y is B

where A & B are linguistic values, defined by fuzzy sets on the ranges (universe of discourse) x and y, respectively. The If part of the rule "X is A" is called the Antecedent or premise, while the then part of the rule "Y is B "is called Consequent or conclusion.

4.6. Interpreting If-Then rules

It is a three-part process:

1) **Fuzzify inputs:** Resolve all fuzzy statements in the antecedent to a degree of membership between 0 & 1. If there is only one part to the antecedent this is the degree of support for the rule.
2) **Apply fuzzy operator to multiple part antecedent:** If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 &1. This is the degree of support for the rule.

3) **Apply implication method:** Use the degree of Support for the entire rule to the shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. This fuzzy set is represented by membership function that is chosen to indicate the qualities of the consequent. If the antecedent is only partially true, (i.e. is assigned a value less than 1), then the output fuzzy set is truncated according to implication method.

4.7. Fuzzy Inference Systems

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. There are two types of fuzzy inference systems that can be implemented in the Fuzzy Logic Toolbox:

- Mamdani-type.
- Sugeno-type.

These two types of inference systems vary somewhat in the way outputs are determined.

Mamdani-type fuzzy inference methods: It is the most commonly seen methodology. Mamdani-type inference as defined for the Fuzzy Logic Toolbox expects the output membership functions to be the fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs Defuzzification. It's possible, and in many cases much more efficient, to use single spikes as output membership functions rather than a distributed fuzzy set.
This is sometimes known as a *singleton output membership function, and* it enhances the efficiency of the Defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of two dimensional functions. Rather than integrating across the two-dimensional function to find the centroid, we use the weighted average of a few data points.

Sugeno-type systems: In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant.

4.8 Graphics User Interface (GUI)

Graphics User Interface tools is provided by the Fuzzy Logic Toolbox. There are five primary GUI tools for building, editing, and observing fuzzy inference systems in Fuzzy Logic Toolbox:

(i) The Fuzzy Inference System or FIS Editor

(ii) The Membership Function Editor

(iii) The Rule Editor

(iv) The Rule Viewer

(v) The Surface Viewer

I. The FIS Editor

The FIS Editor handles the high level issues for the system:

How many input and output variables? What are their names? The Fuzzy Logic Toolbox doesn't limit the number of inputs

II. The Membership Function Editor

The Membership Function Editor is used to define the shape of all the membership functions associated with each variable.

III. The Rule Editor

The Rule Editor is for editing the list of rules that defines the behavior of the system.

IV. The Rule Viewer

V. The Surface Viewer

The Rule Viewer and The Surface Viewer are used for looking at, as opposed to editing, the FIS. They are strictly read-only tools.

4.9. MODELLING

To start this system from scratch, type **'fuzzy'** at the MATLAB prompt.

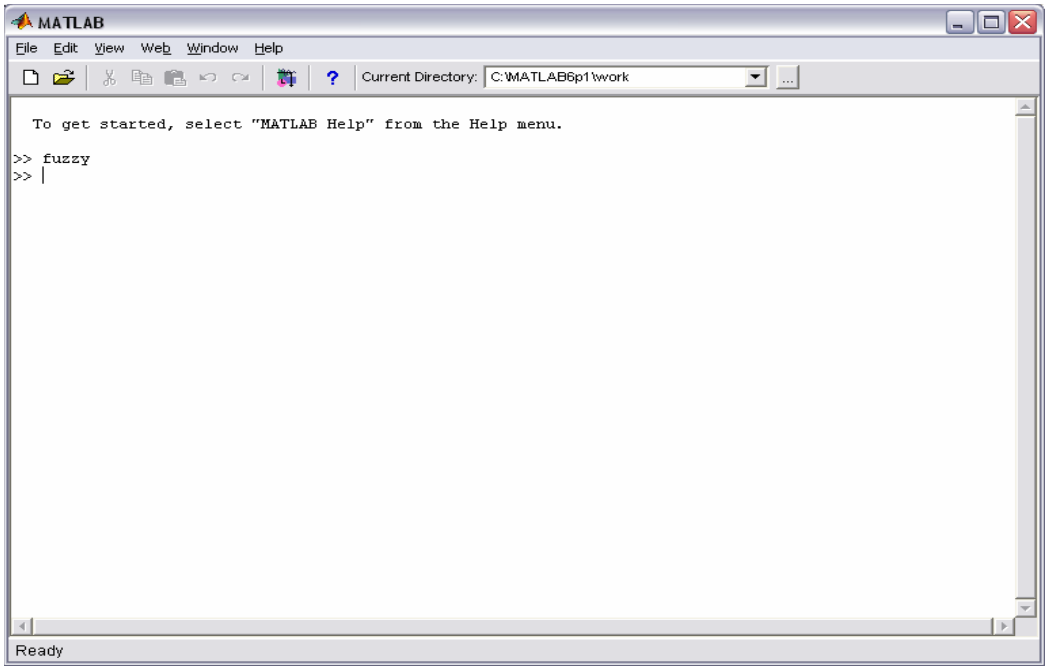


Fig 4.1: MATLAB Prompt

A) The generic untitled FIS Editor opens, with one input, labeled input1, and the one output, labeled output1.

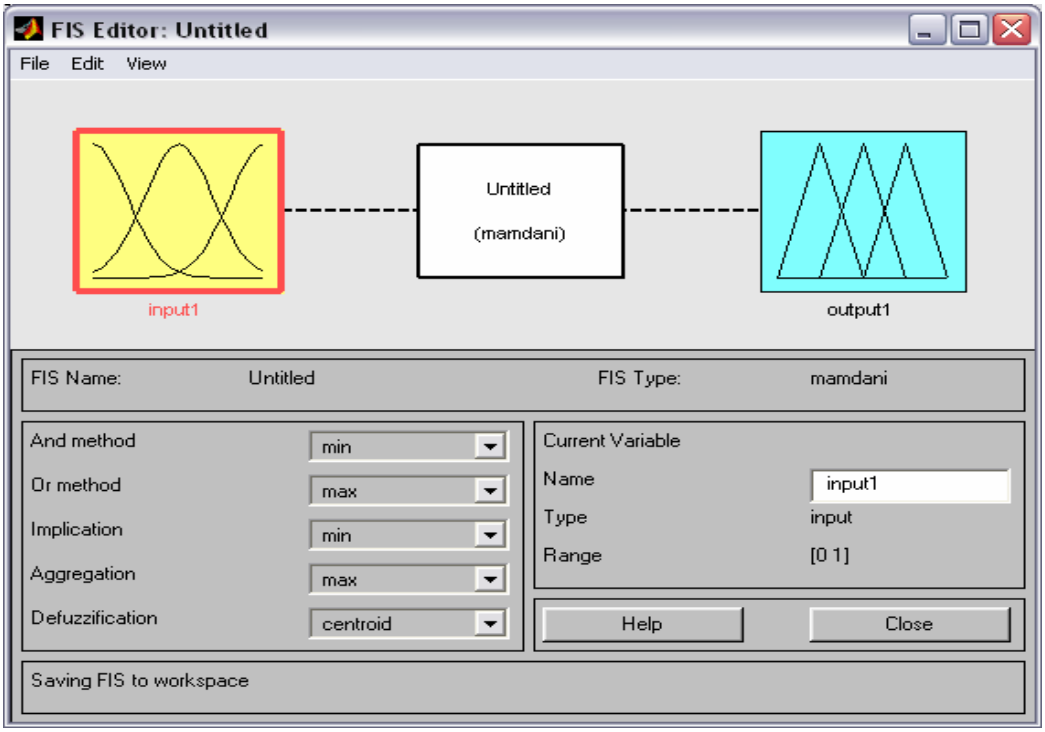


Fig 4.2: FIS Editor

We will construct a one input- one output system, so go to the Edit menu and select Add input. A second yellow box labeled input 2 will appear.

1. Click once on the left –hand (yellow) box marked input1 (the box will be highlighted in red).
2. In the white Edit field on the right, change input1 if you want to change.
3. Click once on the left-hand (yellow) box marked input2 (the box will be highlighted in red) if you want to add another input.
4. In the white edit field on the right, change input2 and press Return.
5. Click once on the right-hand(blue)box marked output1
6. In the white edit field on the right, change output1 if you want to change.
7. From the File menu select save to workspace as…..
8. Enter the variable name untitled and click on OK.

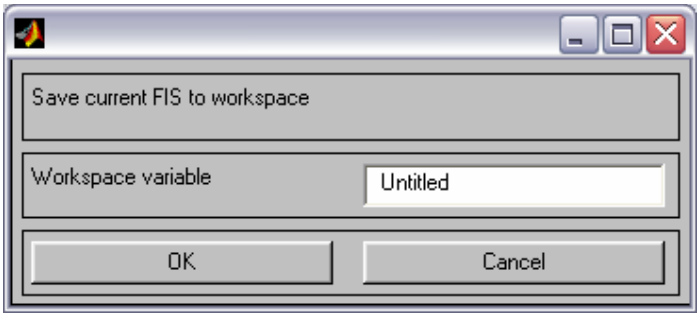


Fig 4.3: workspace save window

The FIS Editor for Friction Compensator handles the high level issues for the system. How many input and output variables? What are their names? The Fuzzy Logic Toolbox doesn't limit the number of inputs. Here the numbers of input variables are two which are velocity (V) and sampler (dv/dt) and only one output variable or parameter as coefficient of friction (μ) as shown in FIS Editor shown below.

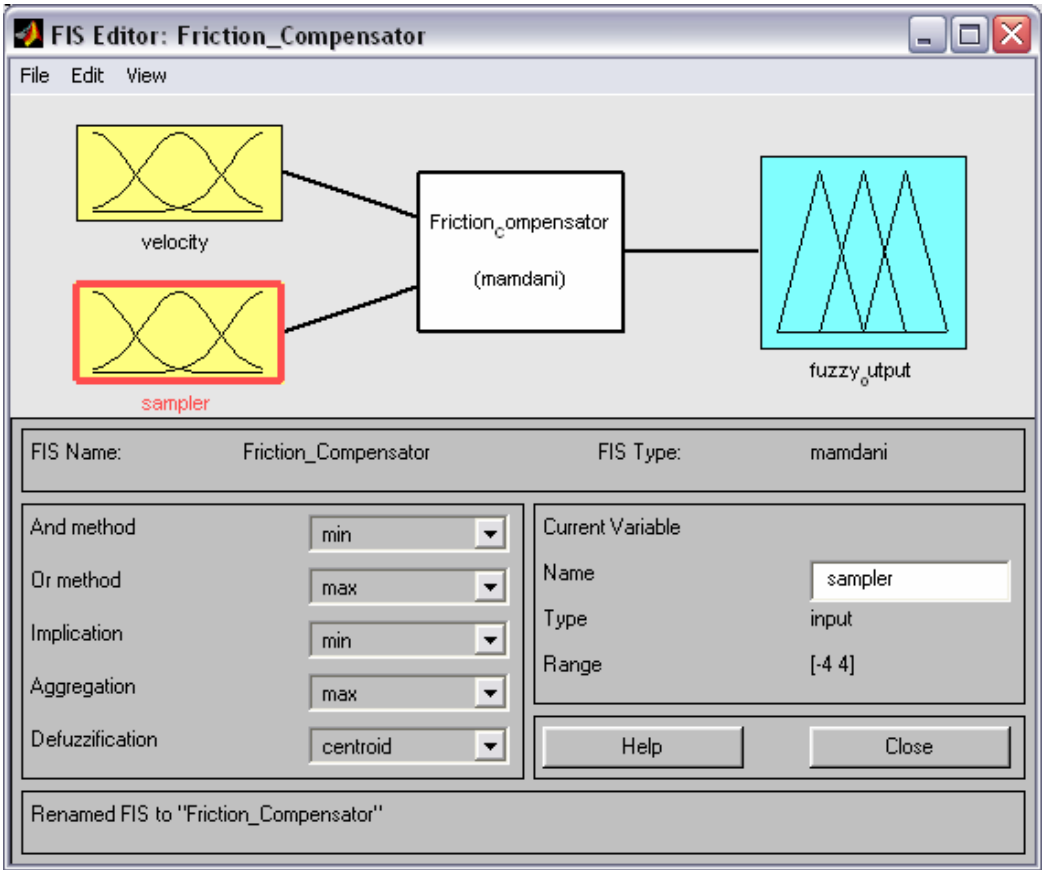


Fig 4.4: FIS editor

B) Define the membership functions associated with each variable. To do this, open the Membership Function Editor in one of three ways:

1. Pull down the **view** menu item and select **Edit Membership Function…**
2. Double click on the icon for the output variable.
3. Type mfedit at the command line.

4.9.1 The Membership Function Editor

The Membership Function Editor is used to define the shape of all the membership functions associated with each variable. Here all the membership functions used are of triangular type and trapezoidal. The membership functions for both inputs velocity and sampler are defined in separate membership function editor. The membership function for output parameter coefficient of friction in this case is also defined in separate membership function editor. The entire three separate membership function editor for input and output variables have different ranges which depends on the velocity Vs coefficient of friction.

4.9.1.1. Input member function velocity (V)

The Input Membership Function Editor for velocity is used to define the shape of all the membership functions associated with input velocity.

The window screen in figure shows "Input membership function editor velocity" of Mat lab version 6.1.The velocity range of 0.1 to 4 and we need to regulate the velocity. The velocity is set in different ranges by using membership functions.

In this case [-0.875 0.1 0.301] are set as vlow or mf1,

[0.1 0.301 0.497] are set as low or mf2,

[0.312 0.508 0.807] as medium or mf3,

[0.497 0.807 3.99] as high or mf4,

[0.817 4 4.97] as vhigh or mf5,

In fuzzy logic fuzzy quantifier are treated as fuzzy member that represents imprecisely the absolute or relative count of elements in fuzzy sets.

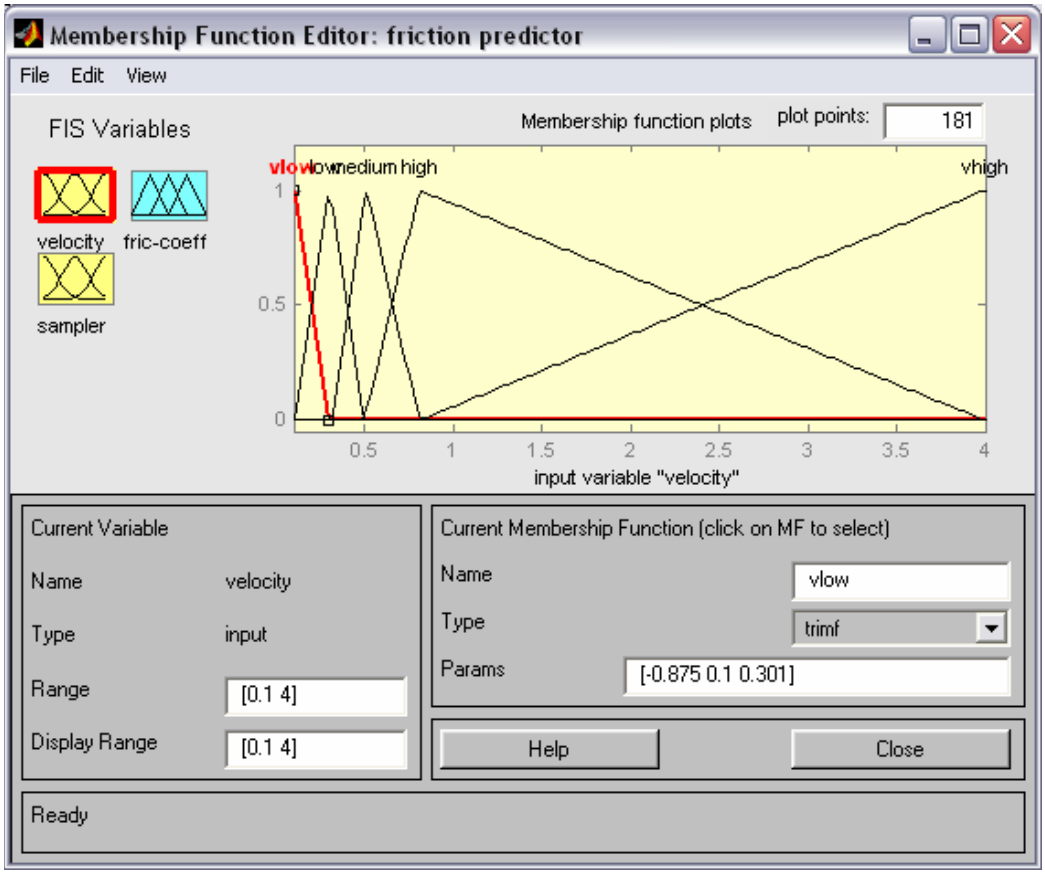


Fig 4.5: Membership function editor (input 1)

4.9.1.2. Input member function sampler (dv/dt)

The Input Membership Function Editor for sampler is used to define the shape of all the membership functions associated with input sampler.

The window screen in figure shows "Input membership function editor sampler" of Mat lab version 6.1.The sampler range of -0.1 to 0.1.

In this case [-0.15 -0.1 0.059] are set as -ve or mf1,

[-0.101 0.0595 0.0749] are set as v-low or mf2,

[0.0595 0.0734 0.0849] as +low or mf3.

[0.0743 0.0902 0.1] as +mid or mf4.

[0.0854 0.1 0.15] as high or mf4.

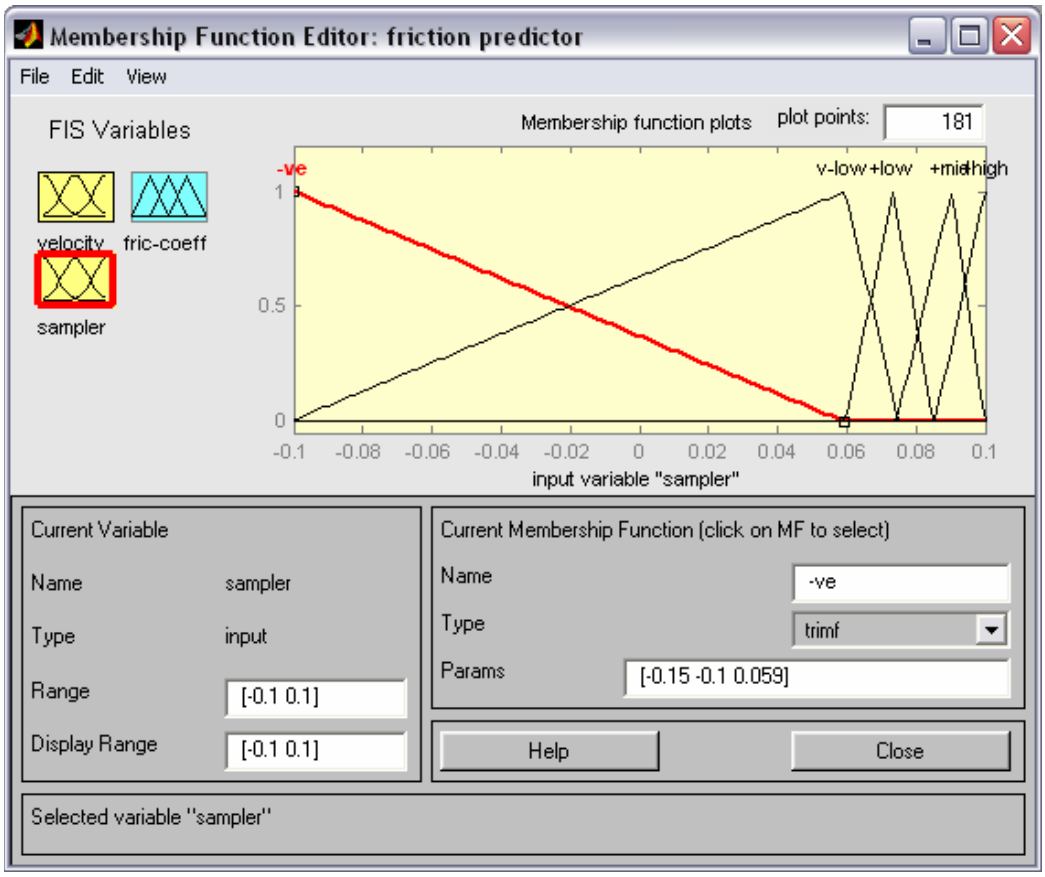


Fig 4.6: Membership function editor (input 2)

4.9.1.3. Output member function coefficient of friction (μ)

The output Membership Function Editor for coefficient of friction is used to define the shape of all the membership functions associated with output.

The window screen in figure shows "Output membership function editor" of Mat lab version 6.1.The coefficient of friction range of 0 to 0.4.

In this case [-0.00875 0.001 0.00477] are set as vlow or mf1,

[0.000897 0.00353 0.0149] are set as low or mf2,

[0.00487 0.01 0.0198] as mid or mf3.

[0.0149 0.02 0.04] as high or mf4.

[0.0199 0.04 0.0498] as vhigh or mf5.

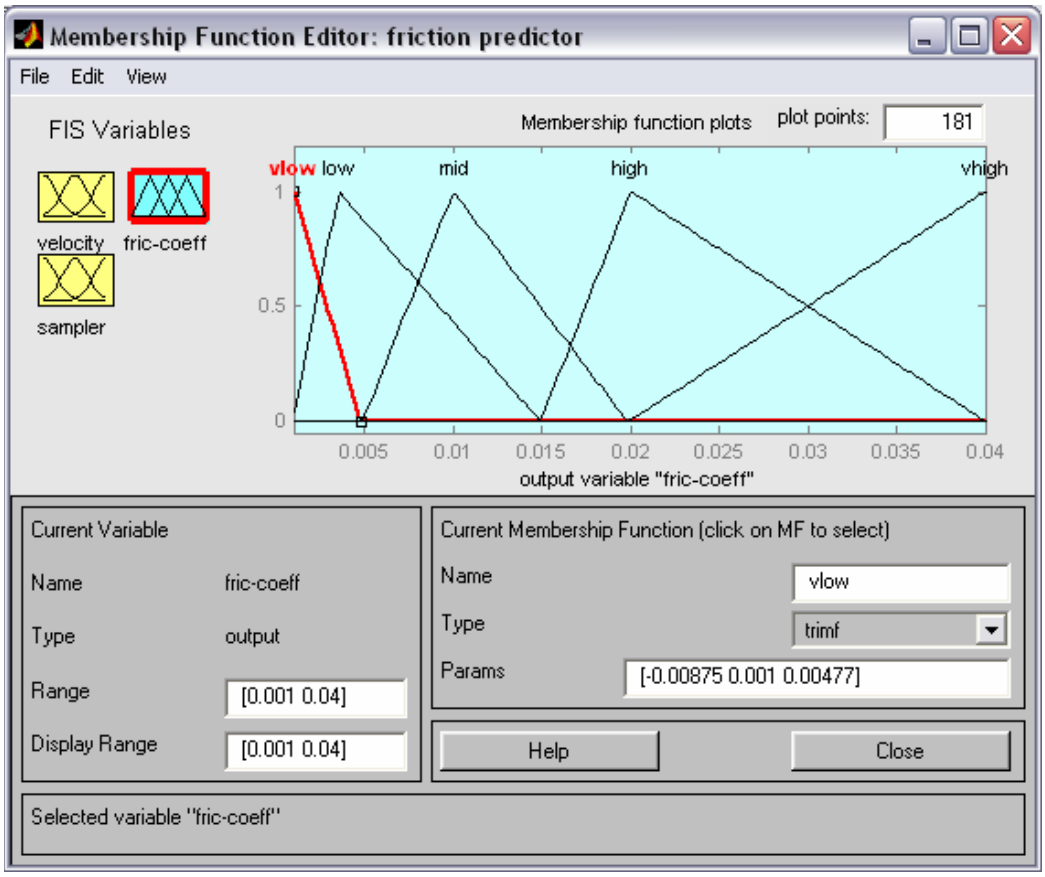


Fig 4.7: Membership function editor (output)

(C) Constructing rules using the Graphical Rule Editor interface allows you to construct the rule statements automatically, by clicking on and selecting one item

in each input variable box, one item in each output variable box, and one connection item.

4.9.2. Rule Editor The Rule Editor is for adding or editing the list of rules that defines the behavior of the system. These are basically computer programs as "if then statements". Here the 25 rules are used to define the behavior of controller. In the rule editor shown above we can increase or decrease the rules depends upon the requirement. We can delete, add and change the rules in the rule editor. The correction AND or OR can be used.

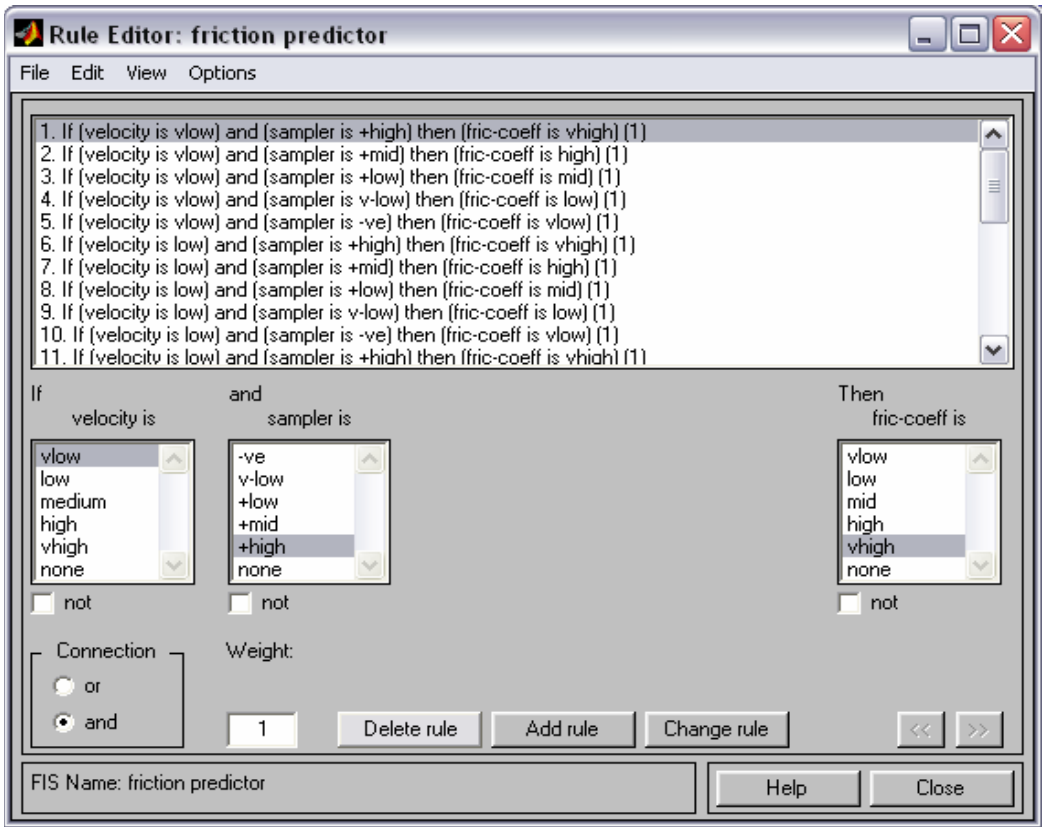


Fig 4.8: Rules editor

4.9.3. Rule Viewer

The Rule Viewer is used for looking at, as opposed to editing, the FIS. This is strictly read-only tool. The window screen shown in figure below shows the rule evaluation for nine rules used in rule editor. In the rule viewer corresponding to both the inputs the velocity and sample we get the output in terms of coefficient of friction.

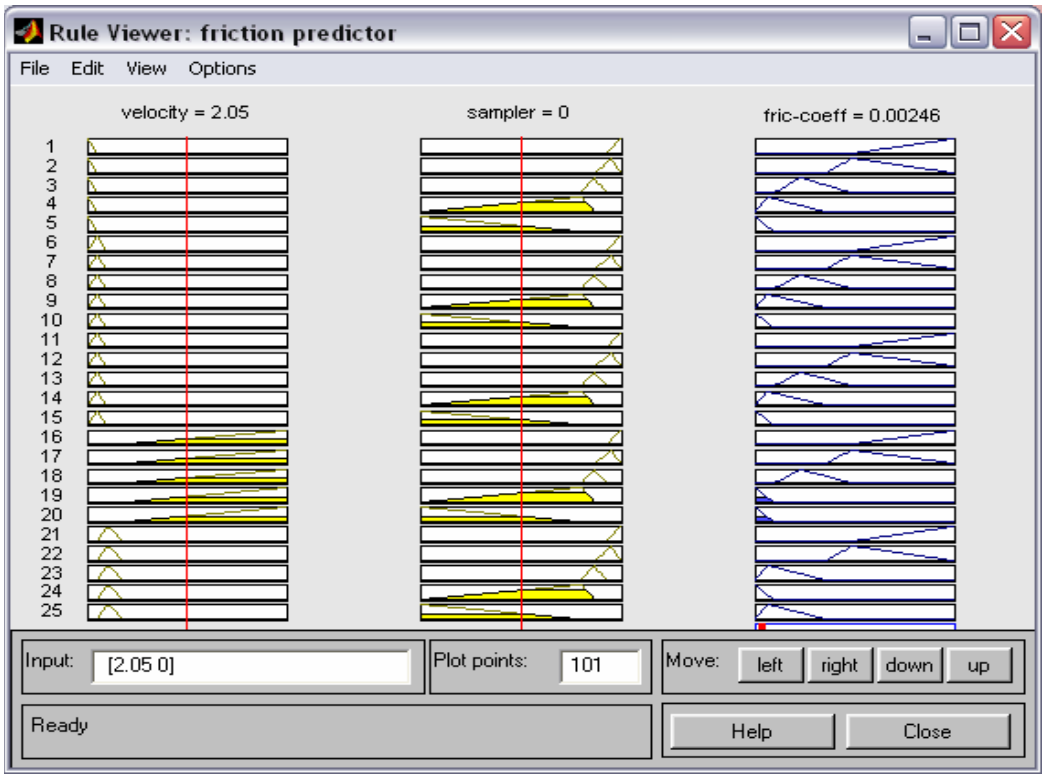


Fig 4.9: Rules viewer

Surface Viewer

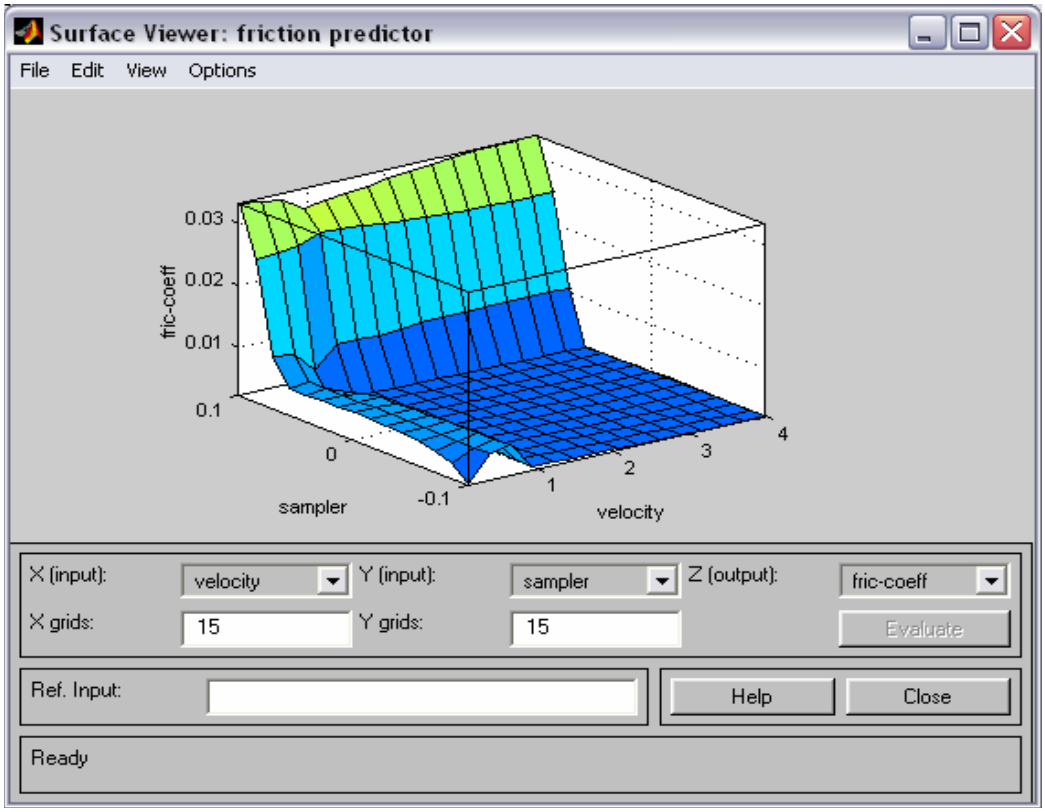
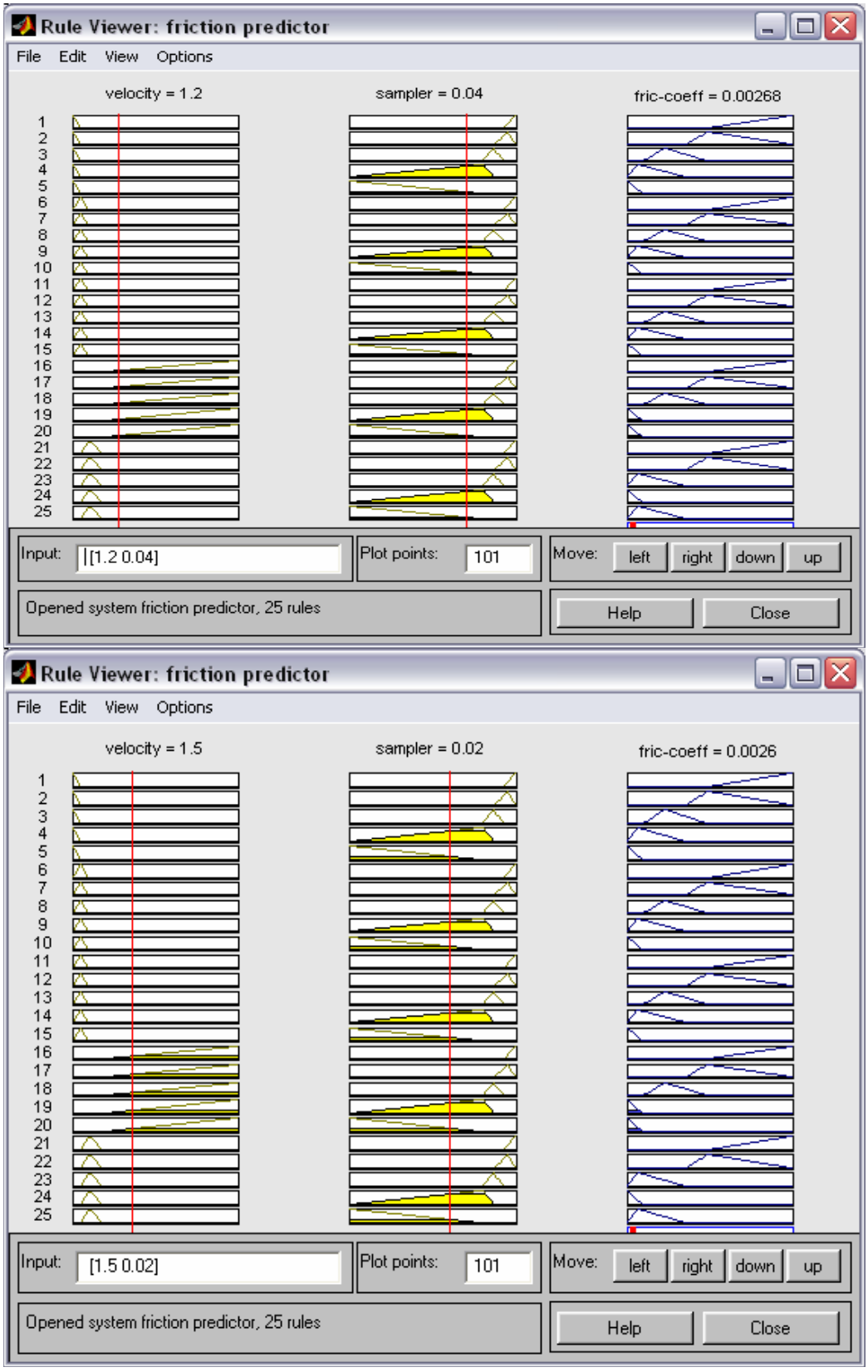
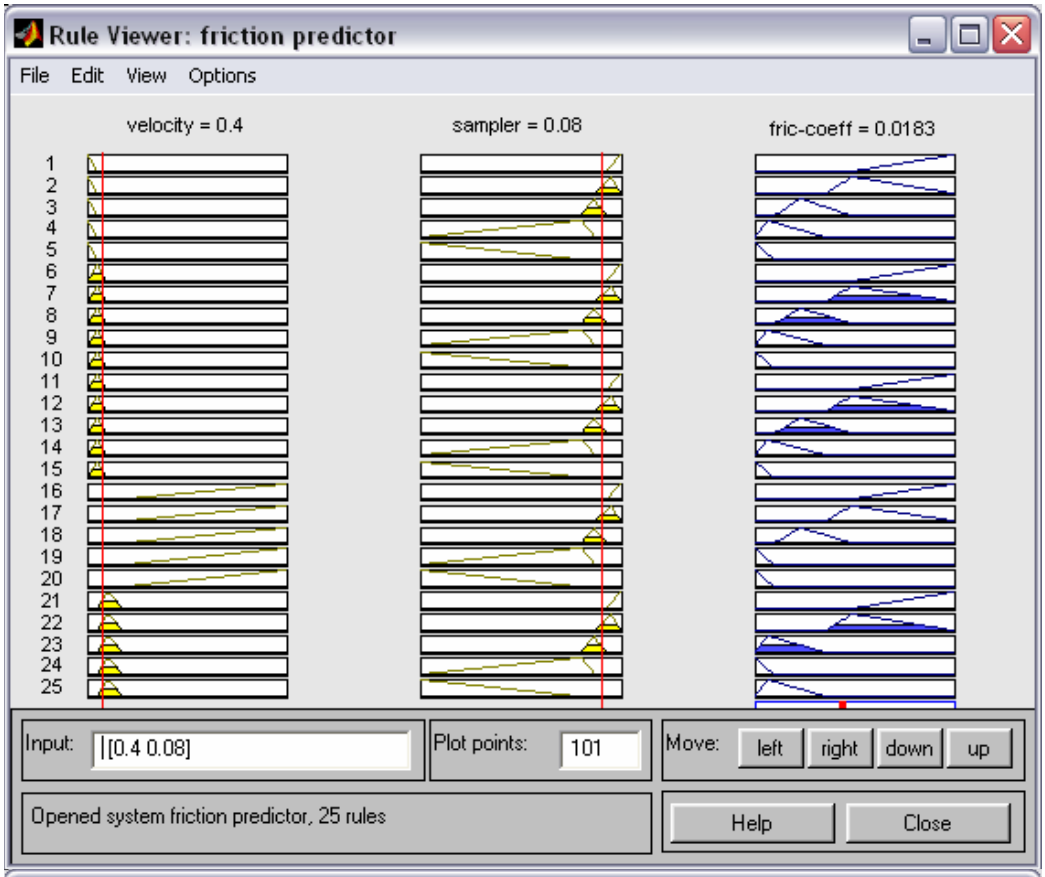
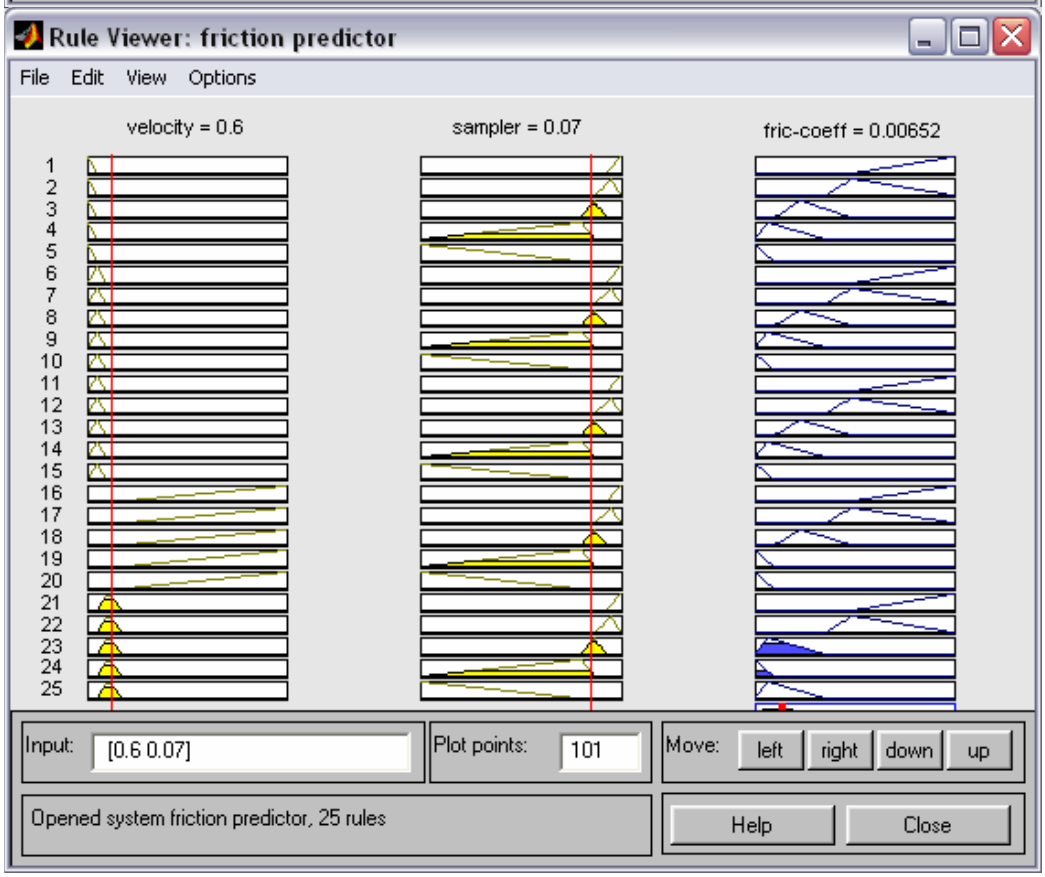


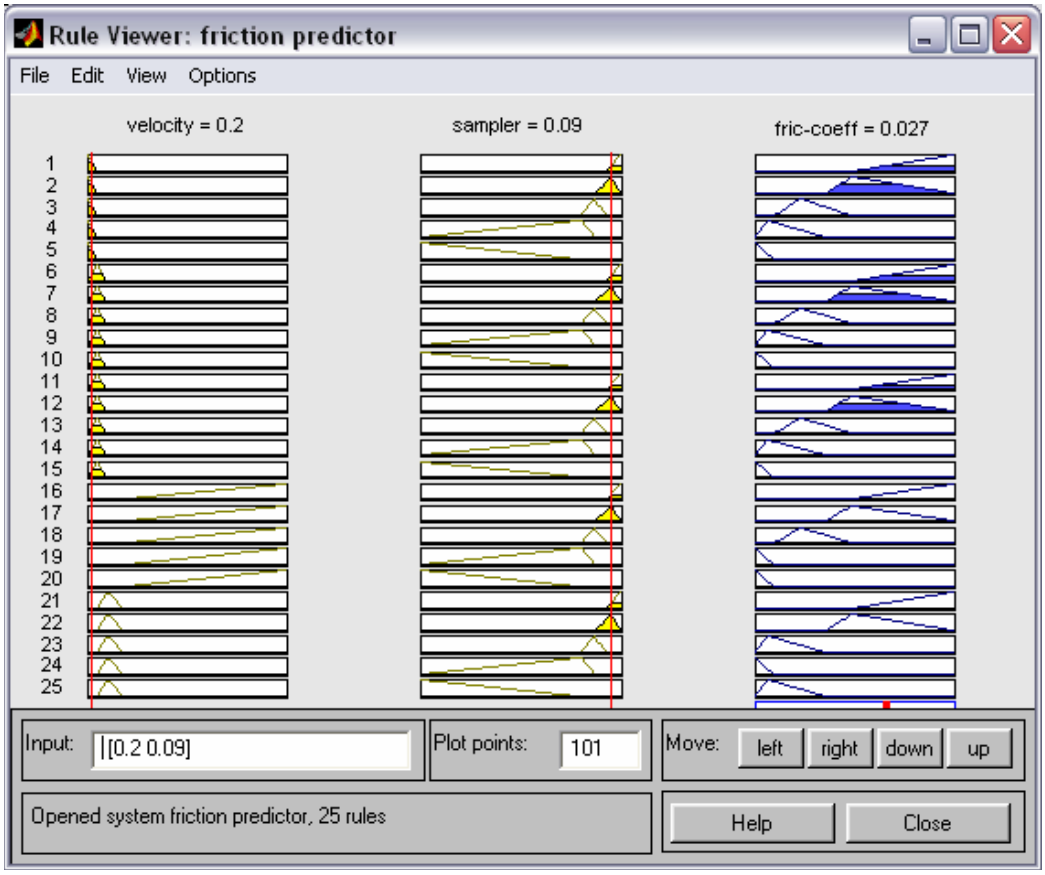
Fig 4.10: Surface viewer

4.9.5. Snapshots: - Few snapshots of friction compensator.




Rule Viewer: friction predictor
File Edit View Options
velocity = 0.4
sampler = 0.08
fric-coeff = 0.0183
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Input: [0.4 0.08]
Plot points: 101
Move: left right down up
Opened system friction predictor, 25 rules
Help Close


Rule Viewer: friction predictor
File Edit View Options
velocity = 0.6
sampler = 0.07
fric-coeff = 0.00652
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Input: [0.6 0.07]
Plot points: 101
Move: left right down up
Opened system friction predictor, 25 rules
Help Close



4.10. ANFIS GUI for Advanced Fuzzy Logic Friction Compensator

The way to proceed is to use Anfis GUI. First open the GUI for Fuzzy Logic Toolbox by typing command Fuzzy or to start this system from scratch pad, type 'fuzzy' at the MATLAB prompt as shown below. We will construct a two input-one output system for advanced fuzzy logic fault detector which uses the Anfis GUI.

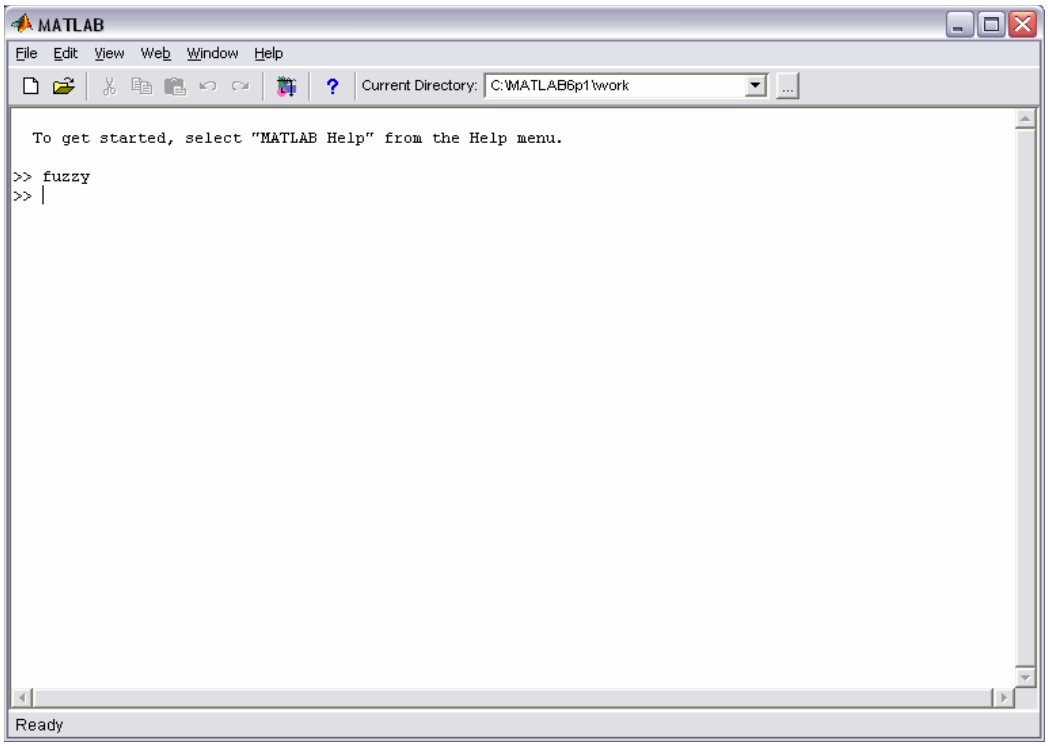


Fig 4.11: MATLAB Prompt

Then press Enter. Here we will see that the generic untitled FIS Editor opens, with one input, labeled input1, and the one output, labeled output1.

The Neuro fuzzy model is constructed with two inputs and single output (TISO). The velocity and sample are considered as inputs and coefficient of friction is chosen as output for the Neuro fuzzy model. The input variables are classified into three membership functions such as low, medium and high. The output variable is classified into three membership functions such as low, medium and high. The velocity range is chosen from 0 to 0.4, sampler range from -4 to 4. The relationship between input and output variables is established through fuzzy rules.

Then in the FIS Editor choose New Sugano system from File menu.

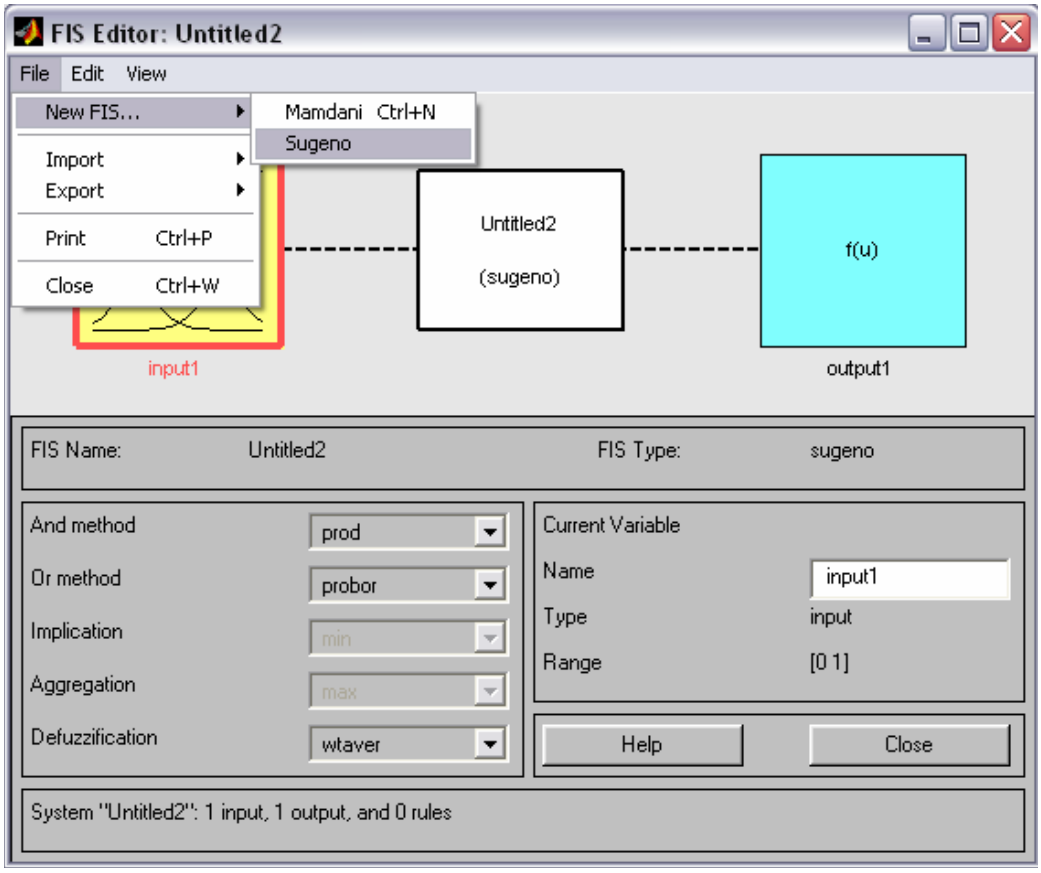


Figure 4.12: FIS Editor for New Sugeno Type System

The new FIS Editor for New Sugeno type system will appear. We have two input model, so go to the Edit menu and select Add input. A second yellow box labeled input 2 will appear.

1. Click once on the left –hand (yellow) box marked input1 (the box will be highlighted in red).
2. In the white Edit field on the right, change input1 if you want to change. Change input1 as velocity.
3. Click once on the left-hand (yellow) box marked input2 (the box will be highlighted in red) if you want to add another input. Change the input2 as sampler
4. In the white edit field on the right, change input2and press Return.
5. Click once on the right-hand (blue) box marked output1 and change output1 as coefficient of friction which is shown here as f (u).

6. From the File menu select save to workspace as…

4.10.1. The FIS Editor for Advance Friction Compensator

The FIS Editor for Sugeno type system handles the high level issues for the system:

How many input and output variables? What are their names? The Fuzzy Logic Toolbox for Sugeno Type System or Model doesn't limit the number of inputs. In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant. Takagi-Sugeno (TS) Model Combines fuzzy sets in antecedents with crisp function in output Here the number of input variables are two which are velocity and sampler and only one output variable or parameter as coefficient of friction or output1 as shown in FIS Editor shown below Figure 4.13. Now define the membership functions associated with each variable. To do this, open the Membership Function Editor in one of three ways:

(1) Pull down the view menu item and select Edit Membership Function…

(2) Double click on the icon for the output variable.

(3) Type mfedit at the command line.

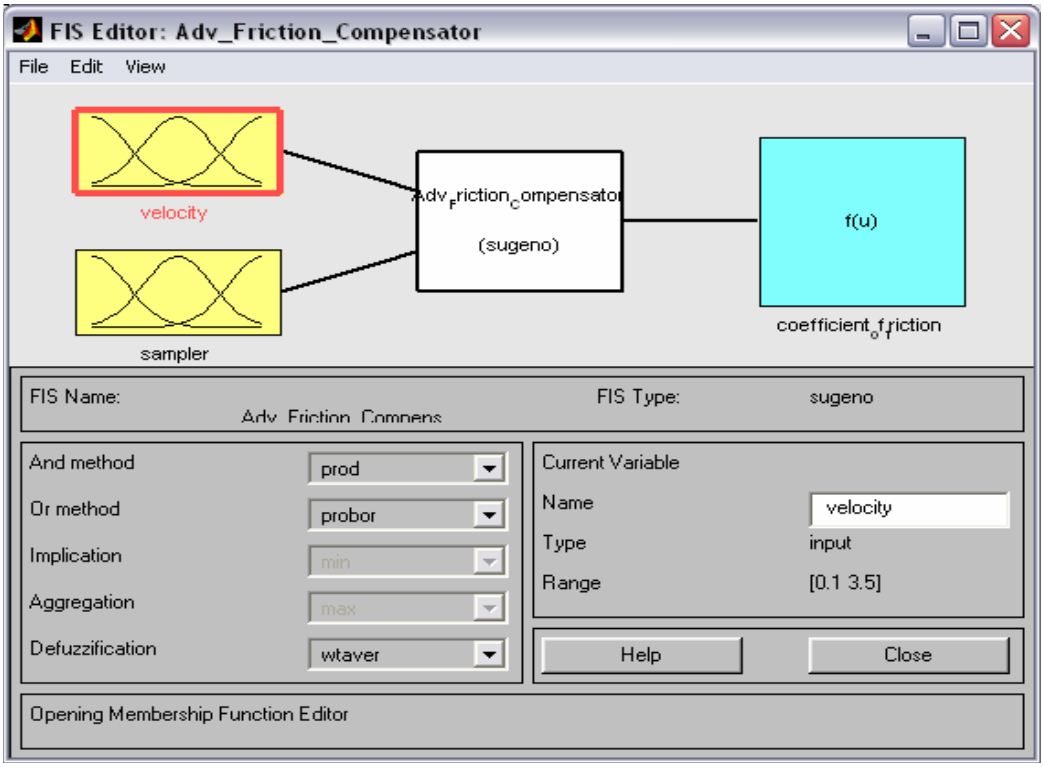


Figure 4.13: FIS Editor for Sugeno Type System

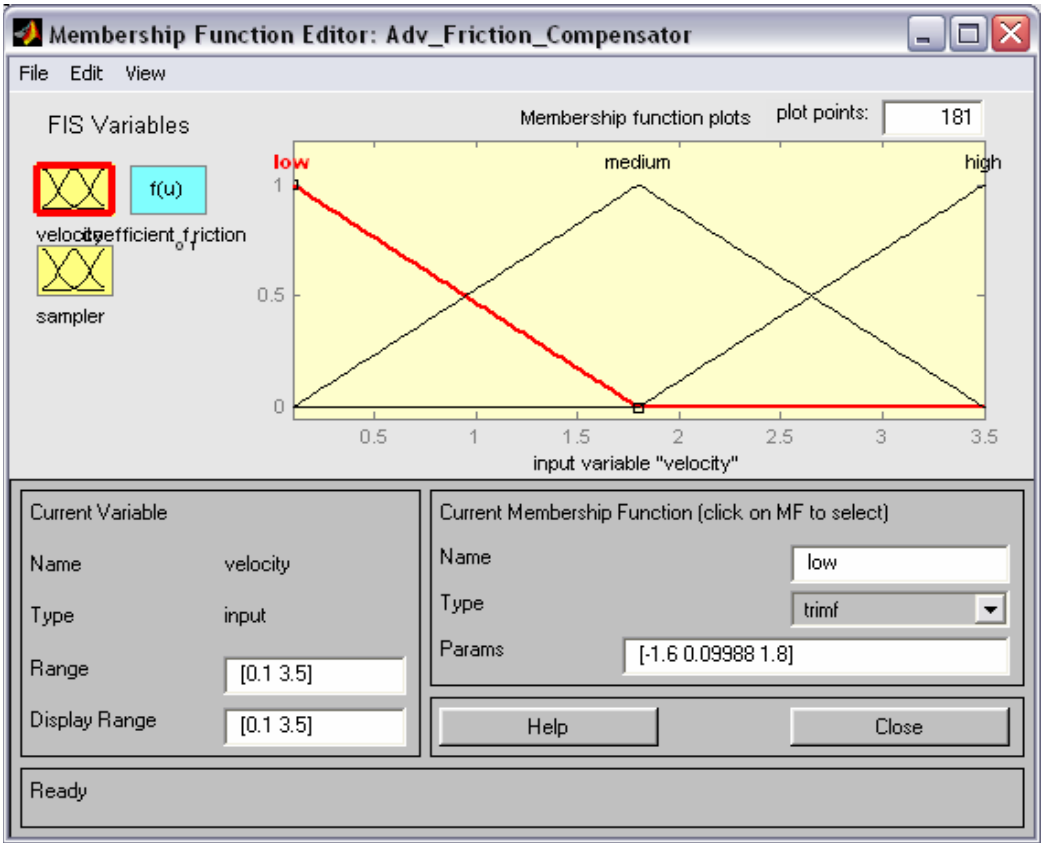


Fig 4.14: Membership function editor

Then in Edit menu choose Edit Anfis.

This will result in the following display.

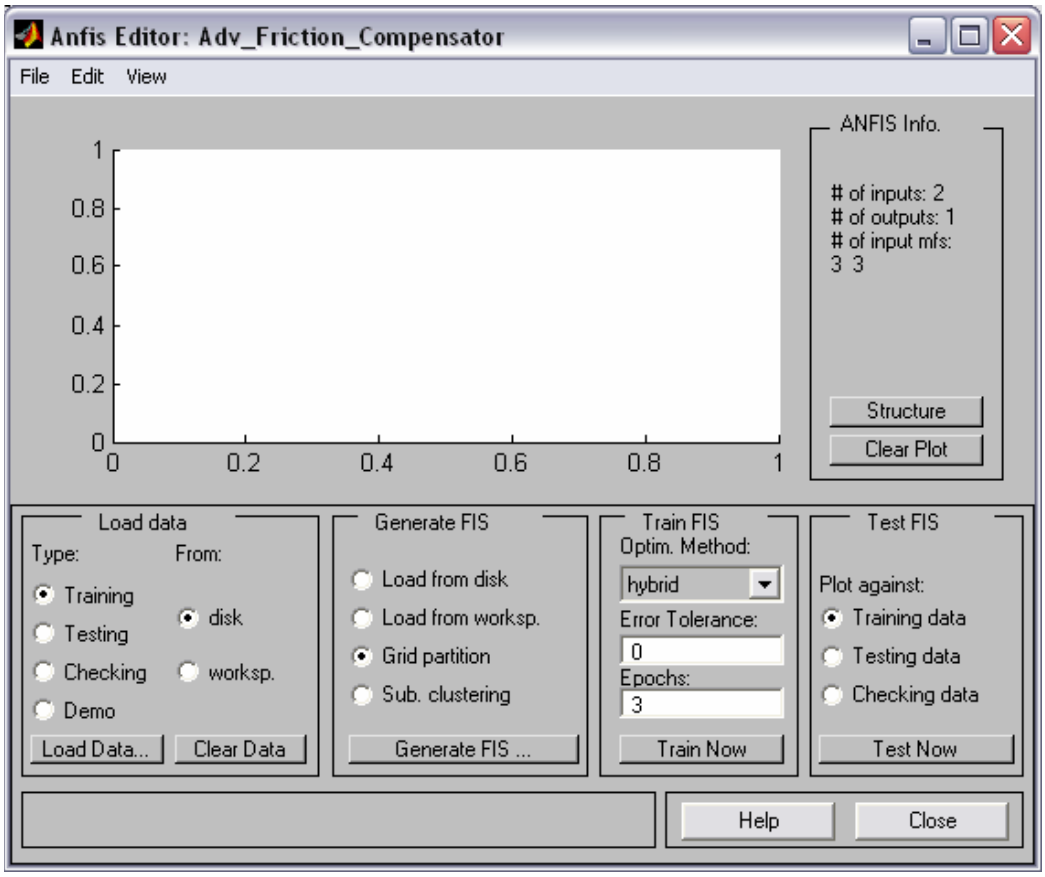


Figure 4.15: Anfis Editor for Friction Compensator

The Anfis Objective is to integrate the best features of Fuzzy Systems and Neural Networks. From Fuzzy Systems the representations of prior knowledge into a set of constraints (network topology) to reduce the optimization search space. From Neural Network the Adaptation of back propagation to structured network to automate FC parametric tuning. The ANFIS application to synthesize the controllers (automated FC tuning) and models (to explain past data and predict future behavior)

4.10.2. Anfis Editor

Display is divided into four main sub displays:

1. Load data
2. General FIS
3. Train FIS

4. Test FIS

Let us consider them in more detail.

4.10.2.1. Load data

Here you can choose Type of data according to the purpose:

Training (default choice), Testing, Checking, or Demo.

For your first effort stick with Training data.

Suppose you have generated data on MATLAB command side. It should be in a matrix form, where the first columns consist of input data and the last of output data. Open the MATLAB command side; go to the file-new-M-file. A new window will open and write the data on this window in matrix form.

Here you also have to be care of where the data are located. Data can be either stored on disk or it is in workspace only, because you just generated it. In our example it is in workspace so you have to change the default value disk to workspace. Click Load data and you are asked to type Input variable name to the space provided. Remember that the data are in matrix form.

Click Load data and you are asked to type Input variable name to the space provided

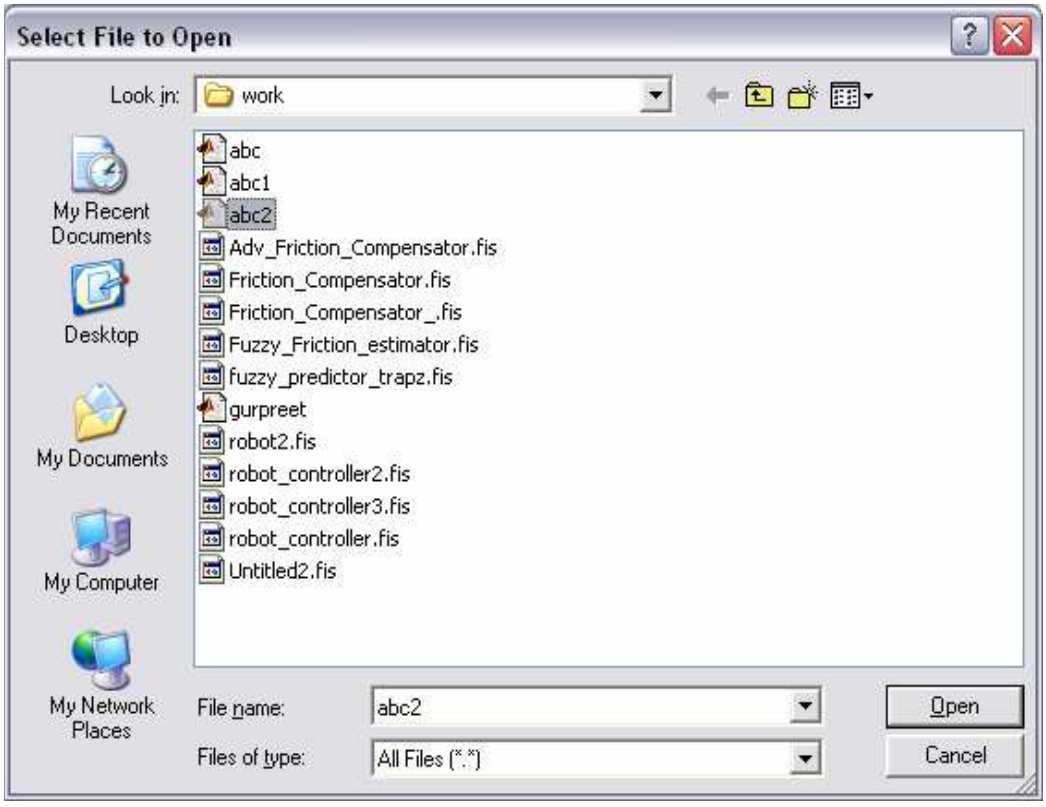


Fig 4.16: Open Window

Load the data in the Anfis Editor from the disk or workspace which is stored in matrix form as discussed above. The following window will appear. In this window the blue circle shows the data is loaded in Anfis Editor.

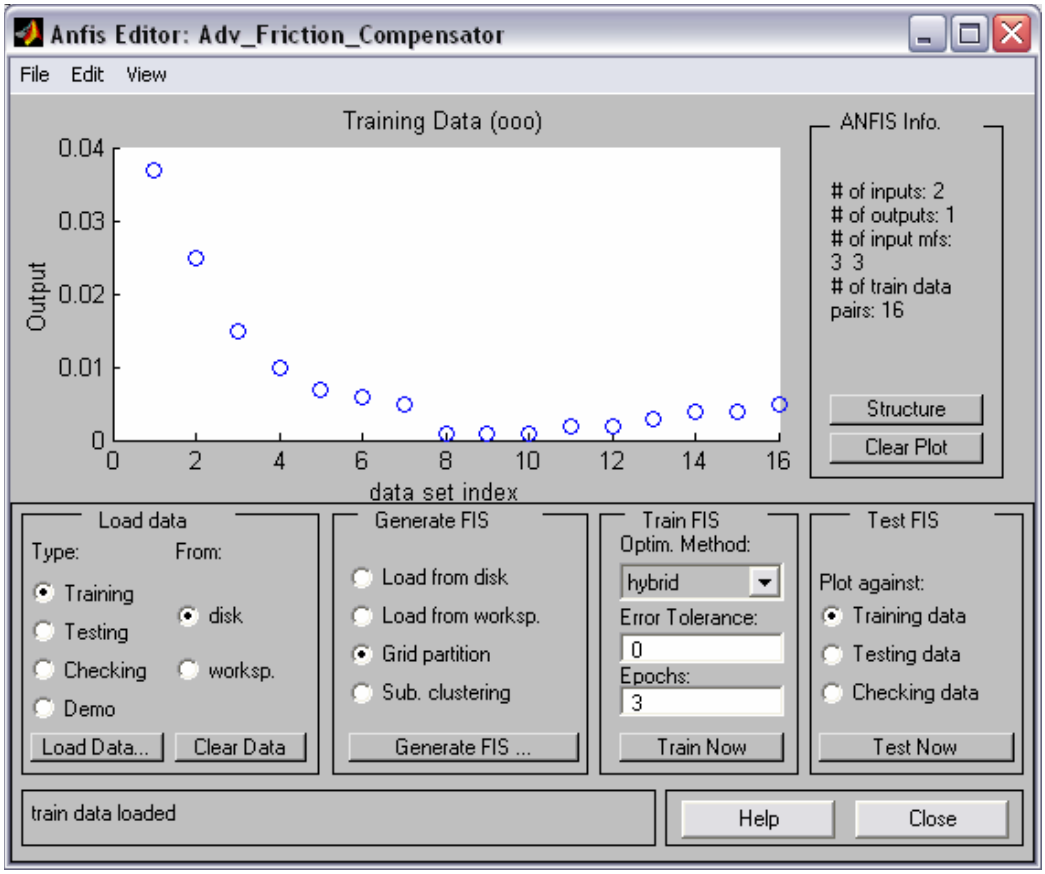


Figure 4.17: Shows Data Loaded on the Anfis Editor.

4.10.2.2. General FIS

Once data have been loaded, an initial fuzzy system can be generated using Generate FIS. Easiest option is to leave the default choice, Grid partition on. Click Generate FIS and the following window opens. In this window we can change the membership function type for inputs such as trimf, trapf gbellmf, gaussmf, gauss2mf, pimf, dsigmfp, sigmf and change the number of membership functions for each input. To assign a different number of membership functions to each input, use spaces to separate the numbers just like 3 3 . From this window it is clear that the two inputs membership functions are of triangular type and three membership functions are generated for each input. This window also

shows the output membership function type which is either liner or constant. You can assign the constant or linear depending upon your requirement of output membership function type. In this case set as linear. Click the Ok.

4.10.2.3. Train FIS

The main computing occurs in the Training block. Here hybrid algorithm is used. The Error tolerance is 0. In practice, a complete fit is not achieved, so change this to, and say 0.01. During training you can see how Training error develops. Epochs means number of iterations. The default value 3 is hardly ever enough. It is not unusual to have hundreds or even thousands of iterations. Pick 5. Now you are ready for training. Click Train now.

Training error

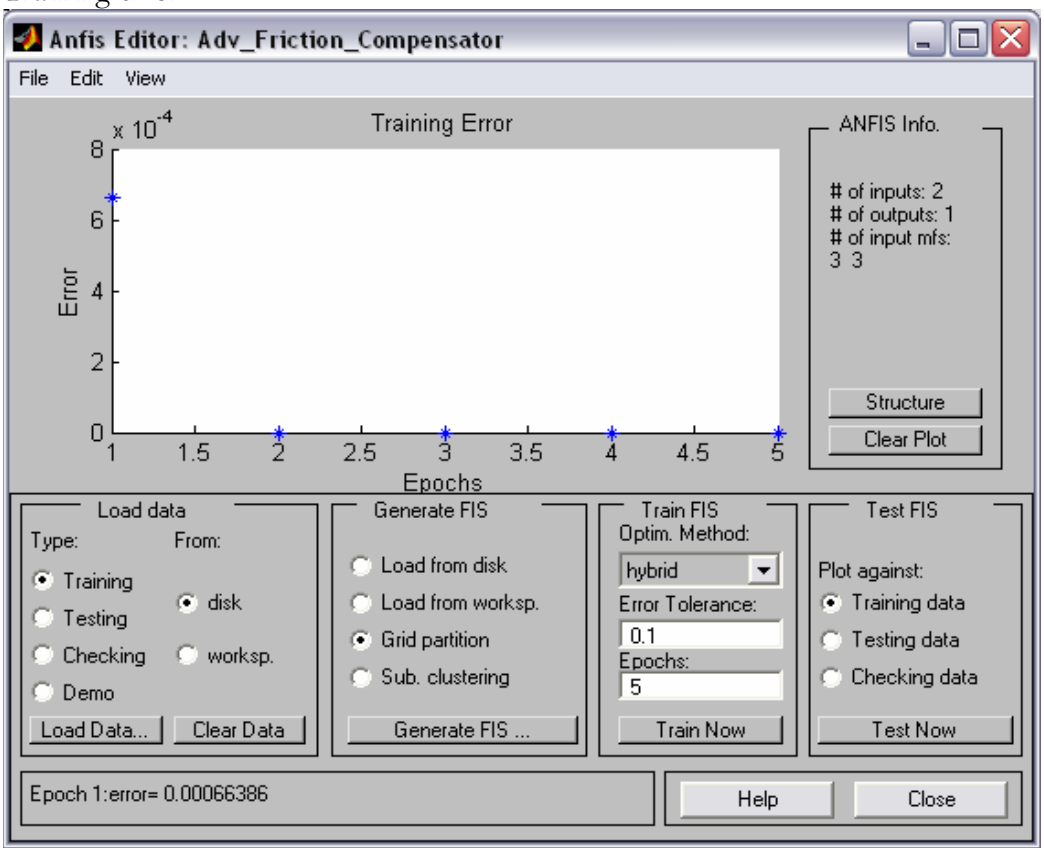


Figure 4.18: Anfis Editor for After Training Data

You can improve the situation by introducing more membership functions.

4.10.2.4. Test FIS

Once the result is satisfactory, you can test the performance of FIS against either training, testing or checking data. Default value is training data.

Click Test now and you will see the following display.

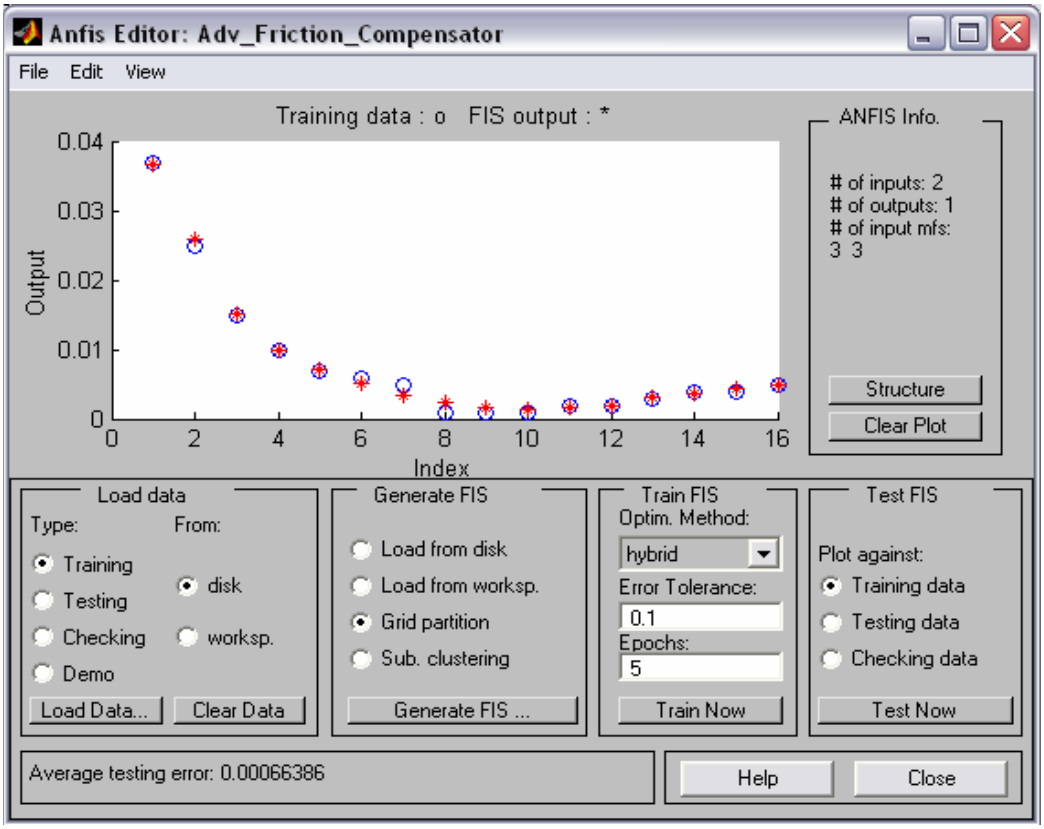


Fig 4.19: Anfis Editor after Testing the Data

MATLAB Command side

Now start the actual optimization procedure by calling **anfis**.

% determine number of iterations

Numepochs = 3;

% Start the optimization

[fismat1, trnerr, ss, fismat2, chkerr]= ...

Anfis (trndata, fismat, numepochs, NaN, chkdata);

ANFIS info:

Number of nodes: 35

Number of linear parameters: 9

Number of nonlinear parameters: 18

Total number of parameters: 27

Number of training data pairs: 16

Number of checking data pairs: 0

Number of fuzzy rules: 9

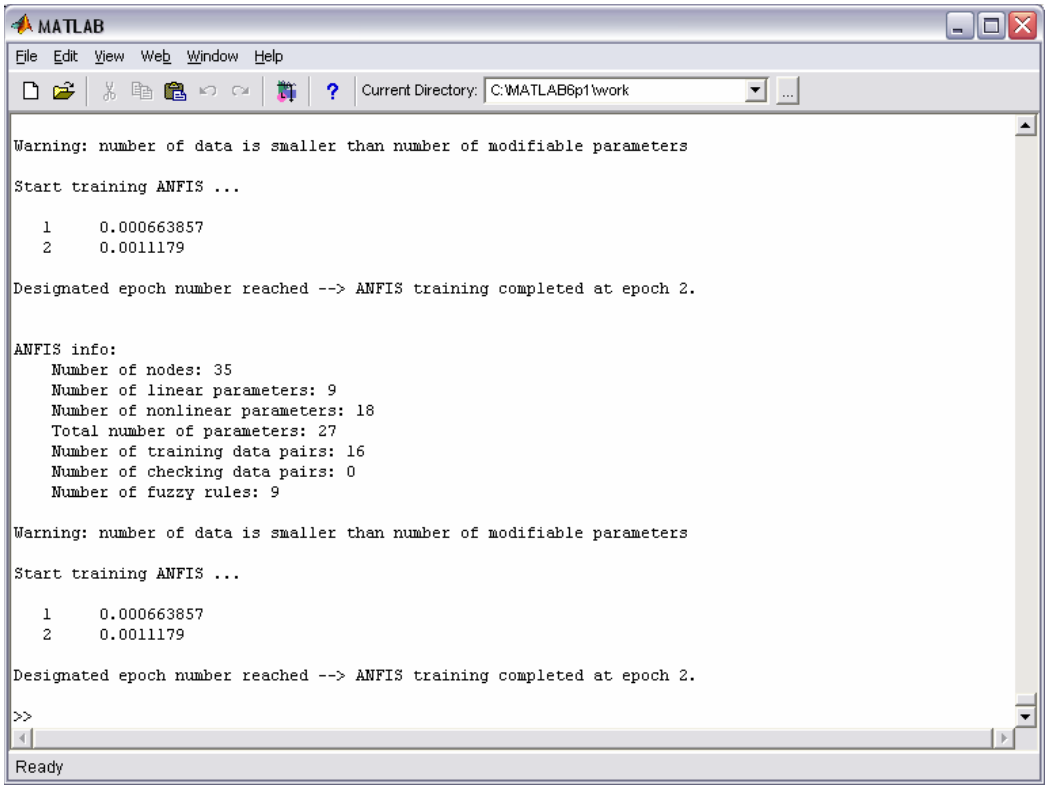


Figure 4.20: Shows the Anfis Info: Window

Start training ANFIS...

1. 0.0006638578
2. 0.0011179

Designated epoch numbers reached -> ANFIS training completed at epoch 3.

Next check the final membership functions again by plotmf or using FIS Editor

Plotmf (fismat1,'input', 1)

4.10.3. Membership function Editor

The details of the membership functions are given below.

4.10.3.1. Membership function Editor for Input1

The Membership Function Editor for Input1 generated here after testing the Anfis Editor defines the shape of all the membership functions associated with input1. Here all the membership functions generated are of triangular type because triangular type membership functions gives minimum error. The membership functions for both inputs are generated in separate membership

function editor. The membership function for output parameter temperature in this case is also generated in separate membership function editor. The entire three separate membership function editor for input and output variables have different ranges depends on experimental data in matrix form used for training the Anfis Editor.

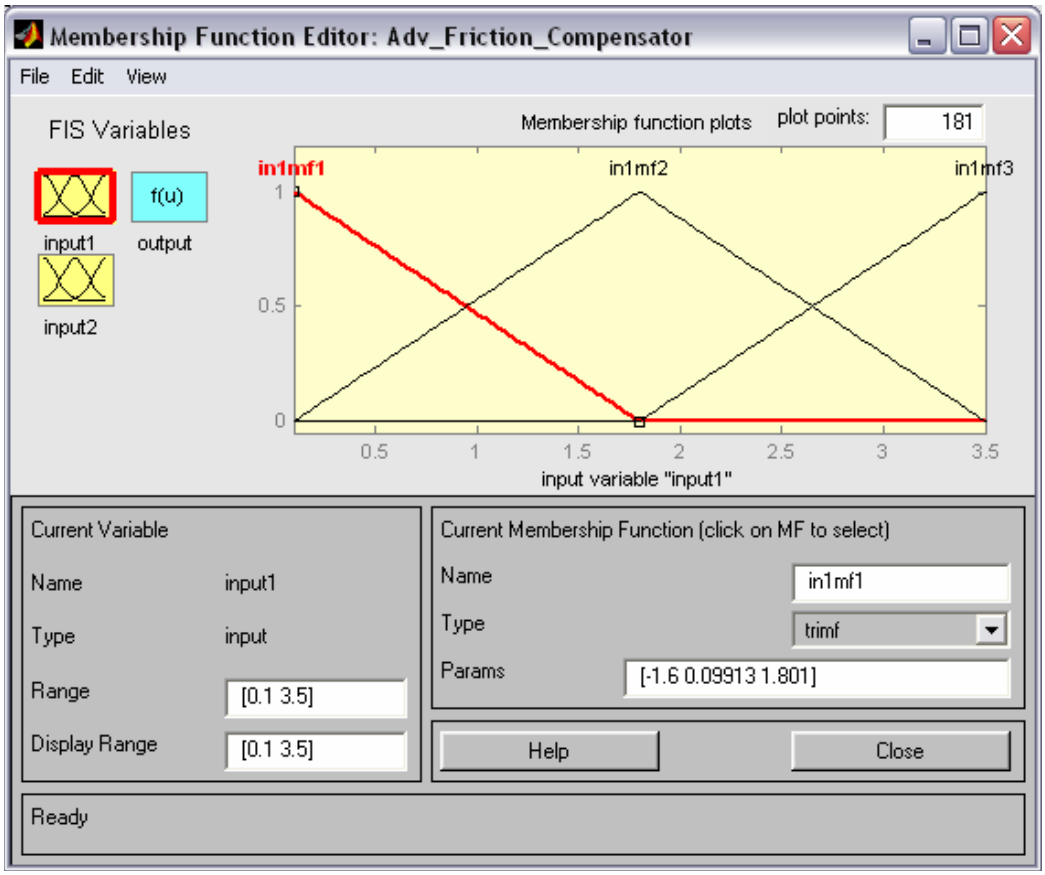


Figure 4.21: Membership function Editor Generated for Input1

The window screen in figure shows "Input membership function editor for input1" of Mat lab version 6.1. The total range generated is [0 3.5].The input 1 varies in the range of [-1.6 0.09913 1.801] for Low or for in1mf1, [0.1002 1.799 3.5] for medium or for in1mf2, [1.8 3.5 5.2] for in1mf3 or for high. All the three membership functions generated here are of triangular type. In practice it is found that the triangular shapes gives simple computation and good sense of fuzzy representation .In fuzzy logic fuzzy quantifiers are treated as fuzzy member that represents imprecisely the absolute or relative count of elements in fuzzy sets.

Next check the final membership functions again by plotmf or using FIS Editor

Plotmf (fismat1,'input', 2)

4.10.3.2. Membership function Editor for Input2

The Membership Function Editor for Input2 generated here after testing the Anfis Editor defines the shape of all the membership functions associated with input2. Here all the membership functions generated are of triangular type because triangular type membership functions gives minimum error. In practice it is found that the triangular shapes gives simple computation and good sense of fuzzy representation.

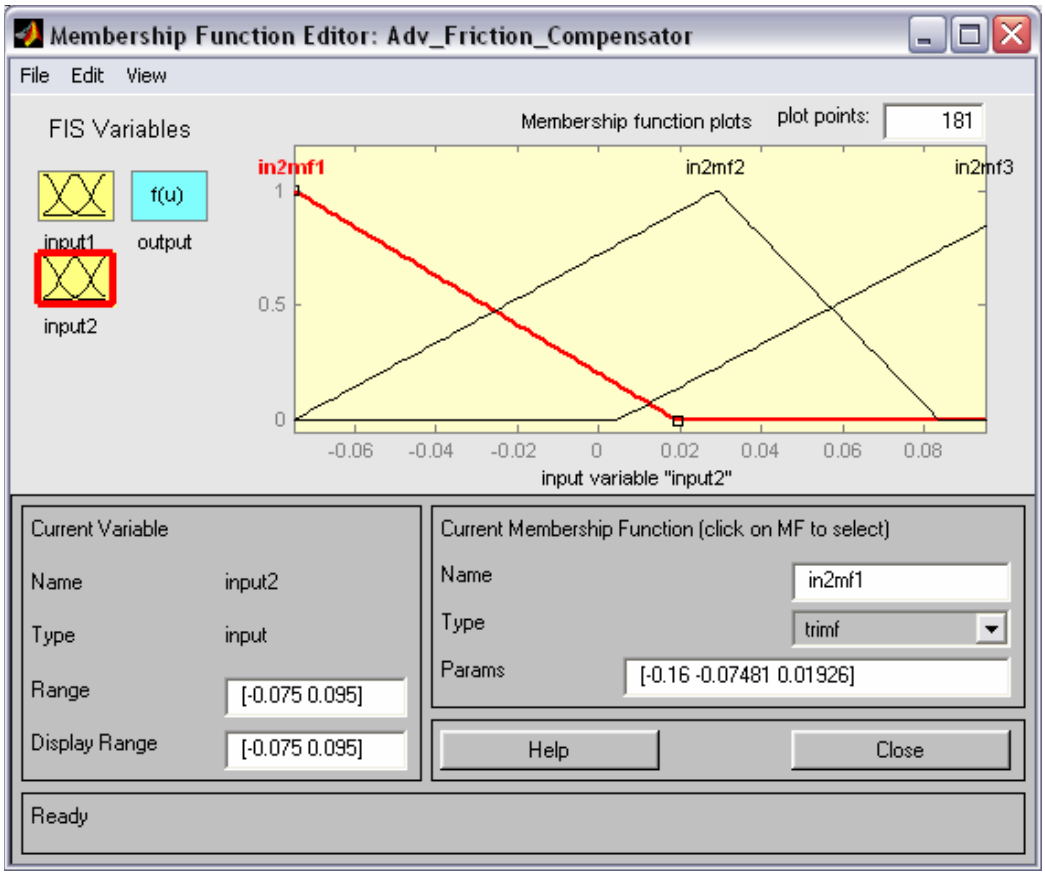


Fig 4.22: Membership function Editor generated for Input2

The window screen in figure shows "Input membership function editor for input2 or Speed (N)" of Mat lab version 6.1. The total range generated is [-0.075 0.095] and display range is same as the total range generated. The input 1 varies in the range of [-0.16 -0.07481 0.01926] for Low or for in2mf1, [-0.07514 0.02914 0.08388] for medium or for in2mf2, [0.004457 0.1119 0.18] for in2mf3 or for high.

Next check the final membership functions again by plotmf or using FIS Editor

Plotmf (fismat1,'output', 1)

4.10.3.3. Membership function Editor for Output

The Membership Function Editor for output generated here after testing the Anfis Editor defines the membership function plots for ouput1,output2,output3, output4,output5,output6, output7, output8,output9 which are associated with coefficient of friction.

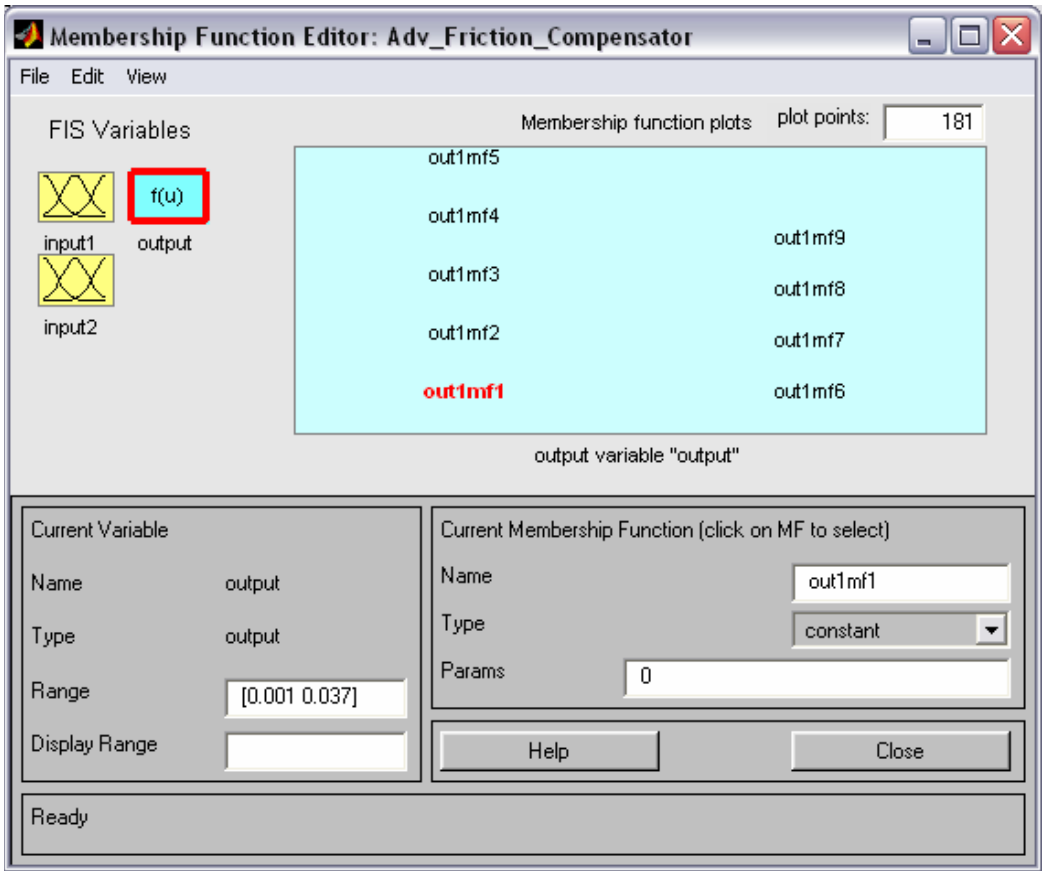


Fig 4.23: Membership function Editor generated for Output

The window screen in figure shows "membership function editor for Output" of Mat lab version 6.1. The total range generated is [0.001 0.037] and display range is same as the total range generated.

4.10.4. Rule Editor

The Rule Editor generated automatically after testing the Anfis Editor defines the list of rules that defines the behavior of the system or single phase induction motor. Anfis Editor generates nine rules in the rule editor corresponding to data matrix stored in workspace used for training and testing purpose. These nine rules are as shown below in the screen window named rule editor. The rules are

basically if then statements e.g., If (input1 is in1mf1) and (input2 is in2mf1) then (output is out1mf1).

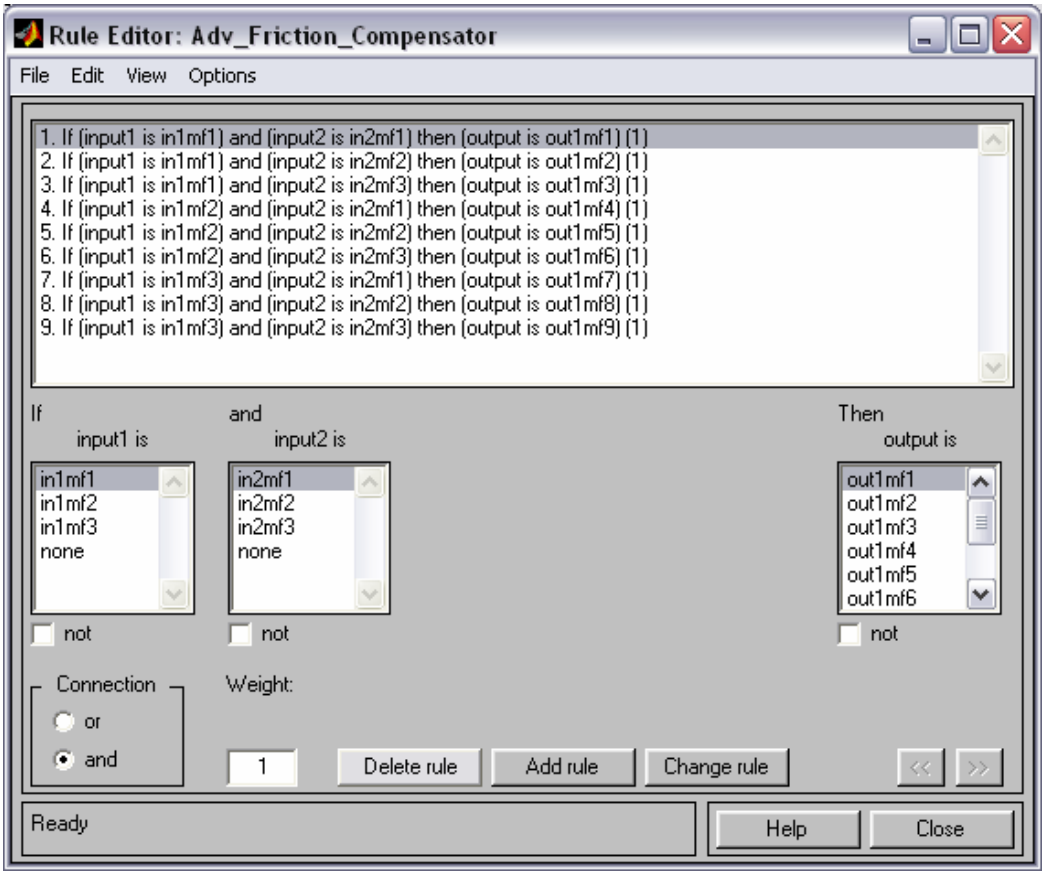


Fig 4.24: Rule Editor Generated Automatically by Sugeno Type System

These are basically computer programs as "if then statements". Here the nine rules are generated to define the behavior of single phase induction motor.

1. If (input1 is in1mf1) and (input2 is in2mf1) then (output is out1mf1)
2. If (input1 is in1mf1) and (input2 is in2mf2) then (output is out1mf2)
3. If (input1 is in1mf1) and (input2 is in2mf3) then (output is out1mf3)
4. If (input1 is in1mf2) and (input2 is in2mf1) then (output is out1mf4)
5. If (input1 is in1mf2) and (input2 is in2mf2) then (output is out1mf5)
6. If (input1 is in1mf2) and (input2 is in2mf3) then (output is out1mf6)
7. If (input1 is in1mf3) and (input2 is in2mf1) then (output is out1mf7)
8. If (input1 is in1mf3) and (input2 is in2mf2) then (output is out1mf8)
9. If (input1 is in1mf3) and (input2 is in2mf3) then (output is out1mf9)

4.10.5. The Rule Viewer

The Rule Viewer is used for looking at, as opposed to editing, the FIS. This is strictly read-only tool. The window screen shown in figure below shows the rule evaluation for nine rules generated in rule editor. In the rule viewer corresponding to both the inputs the velocity and sampler we get the output in terms of coefficient of friction.

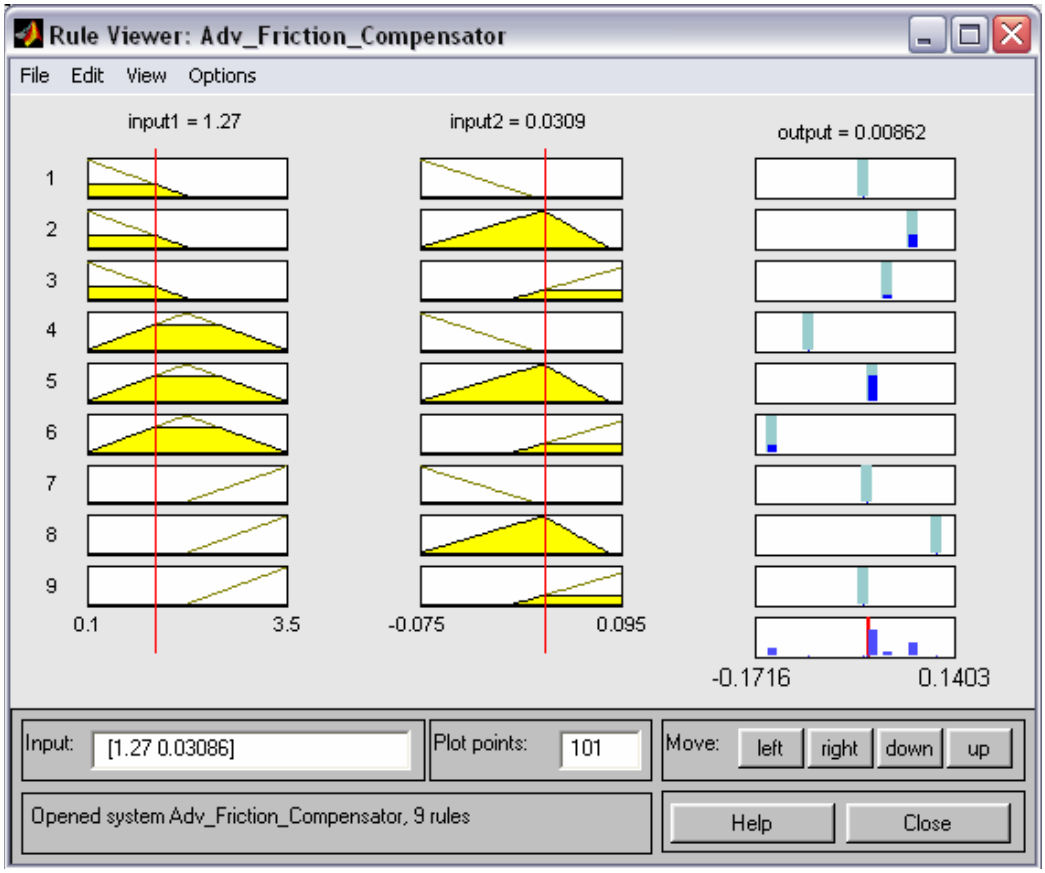


Fig 4.25: Rule Viewer Generated for Checking the Output

This window uses nine rules generated in rule editor. The velocity and sampler can be varied from [0.1 3.5] and [-0.75 0.095] respectively, gives the output in terms of coefficient of friction. Note down the output corresponding to different readings of both the inputs and compare the result with experimental readings, then calculate the error. In practice it is found that the triangular shapes gives simple computation and good sense of fuzzy representation. The output is in the range of [-0.1716 0.1403].

4.10.6. The surface viewer

The Surface Viewer is used for looking at, as opposed to editing, the FIS. This is strictly read-only tool. This gives the three dimensional image for the system.

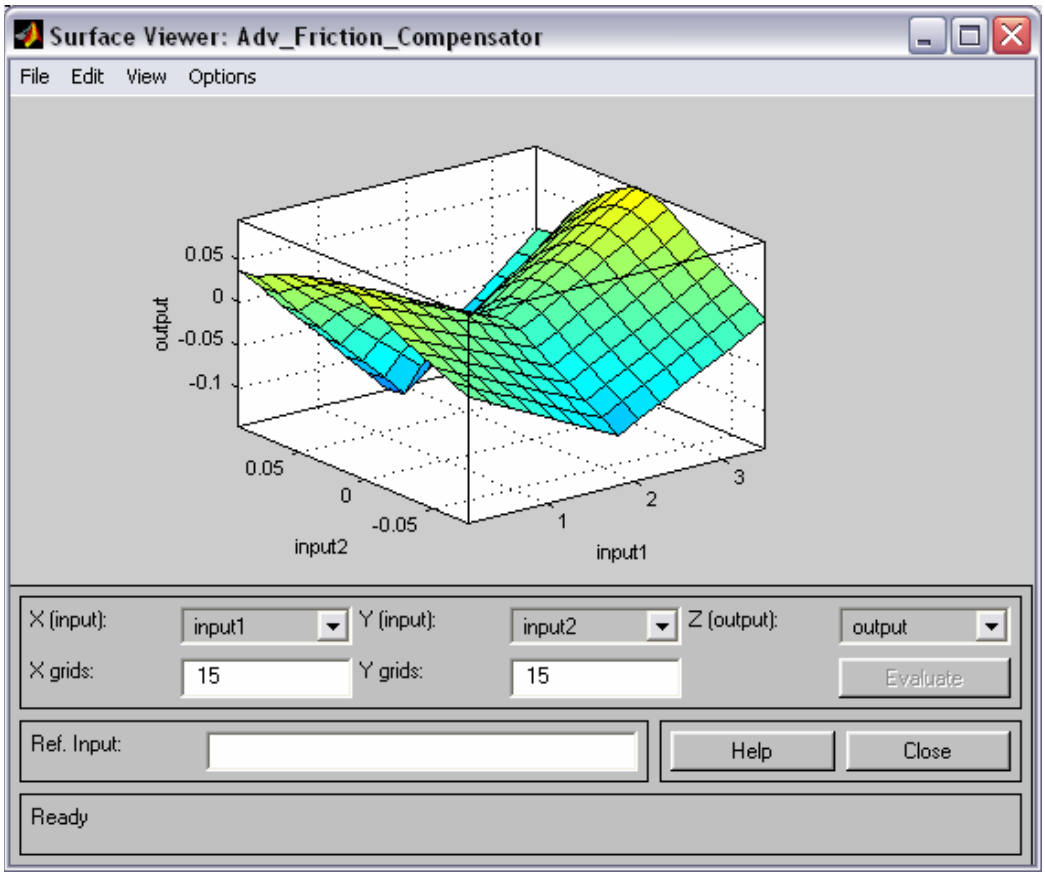


Fig 4.26: Surface Viewer Generated Showing the Input–Output Relationships

Figure 4.26 shows the 3-D map of the input–output relationships between the sideband components input1 and input2,outputThe above window screened shows the three dimensional view or structure for all the input and output variables.

4.10.7. Anfis Model Structure: The Anfis Structure is obtained after the loading, training and testing the Anfis Editor with the data stored in the workspace. It tells

about the input inputmf, rules, outputmf and the output. It shows that there are two inputs and one output; nine rules are generated for this system.

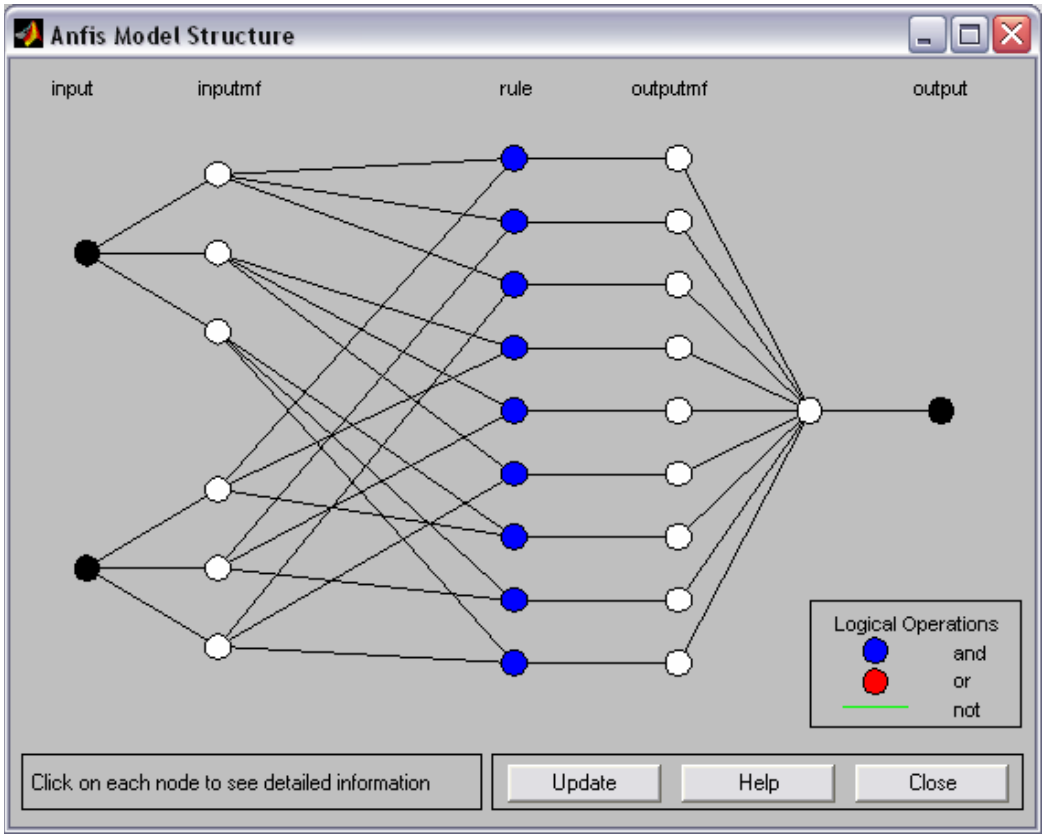


Fig 4.27: Anfis Modal Structure for AFLFD System

The window screened above shows the connection between the inputs and output through the inputmf, rule and outputmf. Blue circle shows the AND connection between the inputmf, outputmf. White circle indicates the inputmf and outputmf; they are also connected to the nine rules generated and with the inputs and outputs shown above. Red circle indicates or connection. The green line can be used for not connection. You can update the rules in the Anfis modal structure. This system or Anfis modal Structure contains the following three different layers:

- Fuzzification layer
- Fuzzy rule layer
- Defuzzification layer

In a Fuzzification layer each neuron represents an input membership function of the Antecedent of a fuzzy rule. In a fuzzy inference layer fuzzy rules are fired and the value at the end of each rule represents the initial weight of the rule, and will

be adjusted to its appropriate level at the end of training. In the defuzzification layer each neuron represents a consequent proposition and its membership function can be implemented by combining one or two sigmoid functions and linear functions. The weight of each output link here represents the centre of gravity of each output membership function of the consequent and is trainable. After getting the corresponding output the adjustment is made in the connection weights and the membership functions in order to compensate the error and produce a new control signal. ANFIS is one of the best tradeoff between neural and fuzzy systems, providing: Smoothness, due to the FC interpolation – adaptability, due to the NN Back propagation ANFIS however has strong computational complexity restrictions.

SUMMARY

The friction compensator is developed with the help of fuzzy tools with hands on the MATLAB. All the input and output screens are shows the destined target.

CHAPTER 5

RESULTS AND DISCUSSIONS

In this chapter, the results are evaluated which comes after the simulation of fuzzy logic friction compensator and advanced fuzzy logic friction compensator. Percentage error and average error are calculated in both cases.

Velocity	**Experimental Data (μ)**	**Fuzzy compensator**	**%Error**	**Advance Fuzzy Compensator**	**%Error**
0.1	0.037	0.038	2.702	0.037	0
0.2	0.025	0.027	8	0.0251	0.4
0.3	0.015	0.025	66.666	0.0148	1.333
0.4	0.01	0.009	10	0.0106	6
0.6	0.007	0.0065	7.142	0.0073	4.285
0.7	0.006	0.0068	13.333	0.0057	5
0.8	0.005	0.005	0	0.0048	4
0.9	0.002	0.0021	5	0.0021	5
1	0.001	0.0011	10	0.001	0
1.2	0.001	0.0011	10	0.001	0
1.3	0.002	0.0019	5	0.0019	5
1.4	0.002	0.002	0	0.002	0
1.5	0.002	0.002	0	0.0019	5
1.6	0.002	0.0021	5	0.0021	5
1.7	0.002	0.0022	10	0.0021	5
1.8	0.003	0.0025	16.66	0.0028	6.667
1.9	0.003	0.002	33.333	0.0028	6.667
2	0.003	0.0028	6.666	0.003	0
2.2	0.003	0.0029	3.3333	0.0031	3.333
2.3	0.003	0.0026	13.333	0.0032	6.667
2.4	0.003	0.0029	3.333	0.0033	10
2.5	0.004	0.0039	2.5	0.0037	7.5
2.6	0.004	0.0039	2.5	0.0036	10
2.7	0.004	0.0041	2.5	0.0039	2.5

2.8	0.004	0.0042	5	0.0039	2.5
2.9	0.004	0.0045	12.5	0.004	0
3	0.004	0.0046	15	0.0042	5
3.1	0.004	0.0045	12.5	0.0043	7.5
3.2	0.004	0.0039	2.5	0.0044	10
3.4	0.005	0.0047	6	0.0048	4
3.5	0.005	0.0048	4	0.005	0
			9.500		4.140

The fuzzy friction compensator and advanced fuzzy friction compensator uses the membership functions of triangular type for input and output. Such systems require "minimum configuration intelligence". There are a lot of advantages using the fuzzy compensator and advanced fuzzy compensator which are already discussed in last chapters.

Fuzzy logic provides heuristic reasoning, is used to simulate and implement the problem. Fuzzy logic can easily and systematically transfer heuristic, linguistic, and qualitative knowledge preferred by humans too and quantitative knowledge preferred by machines, and vice-versa. Based on the heuristics, fuzzy logic provides a general solution for a class of problems, thus making it insensitive to special cases or conditions. Unfortunately, the major drawback of fuzzy logic is that the technology is difficult to give an exact solution to the problems, while an exact solution is essential for some problems like friction compensation.

The results obtained from this illustration give evaluated heuristically by extracting the learned information from each module. This heuristic information indicates if any incorrect assumptions were made when initializing the neural/fuzzy friction compensator. After formulating the friction compensator with sets, we then use a heuristically constrained neural/fuzzy system to learn the exact input/output relation of the velocity and coefficient of friction for a specific rotary system. This

system provides updated membership functions of the sets which better describe the relation between them.

The average error in fuzzy friction compensator is more than that of average error in advanced fuzzy friction compensator. The average error in fuzzy logic friction compensator is 9.5% and 4.1% in advanced fuzzy logic friction compensator. The performance of advanced fuzzy friction compensator is analyzed through computer simulation and results were presented. From the simulation results, it is inferred that the advanced fuzzy friction compensator with three triangular membership functions well suited for this application since it gives minimum error.

CONCLUSION AND FUTURE SCOPE

Conclusion

Fuzzy is a good approach, but should be used with system damping model whose structure is chosen according to the behavior of the predominant friction contribution in a particular rotary system. The accuracy of the fuzzy compensator is increased by increasing the input parameters. The precision of the fuzzy compensator can be improved when it includes parameters which are determined experimentally.

Future Scope

It is not necessarily correct to use a same fuzzy compensator for each rotary system. With further study, it may be possible to identify general fuzzy compensator for different types of rotary systems. Simulation combined with experiment can be used to develop these fuzzy compensator models as well to refine them for a particular system.

REFERENCES

[1]. **M. R. Popovic, D. M. Gorinevsky, A. A. Goldenberg;** *High precision positioning of a mechanism with nonlinear friction using a fuzzy logic pulse controller,* Robotics and Automation Laboratory, University of Toronto.

[2]. **Brian Armstrong;** *Friction: Experimental determination, modeling and compensation,* 1988 IEEE

[3]. **Brian Armstrong-Helouvry;** *Stick-slip arising from stribeck friction,* 1990 IEEE.

[4]. **Pierre E Dupont;** *Friction modeling in Dynamics Robot Simulation,* 1990 IEEE.

[5]. **Hanuman Rachor;** *Investigation of dynamic friction lubricated surfaces,* Ph.D. Thesis, Department of Mechanical Engineering, New Jersey Institute of Technology. January 1996.

[6]. **C. Canudas de Wit, H. Olsson, K.J. Astrom, P.Lischinsky;** *A new model for control of systems with friction,* IEEE Transaction on Automation & Control Volume 40 No 3 March 1995.

[7]. **Paolo Dadone, Hugh F. Van Landingham;** *The use of fuzzy logic for controlling coulomb friction in crane swing alleviation,* 1999 ANNIE.

[8]. **Jean Claude Piedbceuf, Jean De Carufel, Richard Hurteau;** *Friction and stick slip in Robots: simulation and experimentation,* Multi-body System Dynamics 4: 341-354, 2000.

[9]. **James Vernon;** *Fuzzy Logic systems,* Fuzzy logic systems, www.control-systems-principal.co.uk.

[10]. **Pierre E. Dupont;** *The effect of coulomb friction on the existence and uniqueness of the forward dynamics problem,* Proc. Conference on Robotics and Automation, Nice France. May 1992 pp 1442-1447.

[11]. **J.ong Bae Lee, Tae Bin Im;** *A low cost speed control system of brush less motor using fuzzy logic.*

[12]. **Pierre E. Dupont;** *Avoiding stick slip in position and force control through feedback,* Department of Aerospace and Mechanical Engineering Boston University, IEEE 1990.

[13]. **Franz Josef Elmer;** *Nonlinear dynamics of dry friction,* J Phys. A: Math Gen volume 30, 1997, pp 6057-6063.