FUZZY LOGIC CONTROLLER FOR CONTROLLING DC MOTOR SPEED USING MATLAB APPLICATIONS

NUR AZLIZA ALI

This Thesis is Part Fulfillment of the Requirement for a Bachelor

Degree of Electrical Engineering (Power System)

Faculty of Electrical & Electronic Engineering

University Malaysia Pahang

NOVEMBER, 2008

DECLARATION

“I declare that this thesis entitled ‘Fuzzy Logic Controller for Controlling DC motor speed using Matlab Application’ is the result of my own research except as cited in references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree”,

Signature :……………………………

Name of candidate: Nur Azliza Bt Ali

Date : November 8, 2008

DEDICATION

Dedicated specially to my beloved family,friends

and and to all my faculty members.

For you care, support and belive in me

Sincerely,

Nur Azliza Ali

ACKNOWLEDGEMENT

Alhamdullilah... Finally I have finished my “Projek Sarjana Muda”. I would like to express my sincere thankful and appreciation to my supervisor Mr. Ahmad Nor Kasruddin Nasir for his guidance, encouragement and advice throughout the preparation of this thesis. His influence has helped me learn the practicalities of this project. I also would like to express my gratitude to my academic advisor, Prof Madya Shaikh Nasir Shaikh Ab Rahman for his support and believe in me during my studies.

I am very thankful to Universiti Malaysia Pahang and Fakulti Kejuruteraan Elektrik & Elektronik (FKEE) for providing good facilities in campus and laboratory specifically. A very big thank you dedicated to all the staff of Faculty of Electrical and Electronics. In addition, I would like to acknowledge Mr. Mohd Salmizan for his expertise and assistance with the implementation of the lab equipment.

Finally, I would like to thank to my family and all members for their support and encouragement.

ABSTRACT

The purpose of this project is to control the speed of DC motor by using fuzzy logic controller with MATLAB applications. The scopes includes the simulation and modeling of DC motor, implementation of fuzzy logic controller to actual DC motor and comparison between MATLAB simulation and experimental result. This research was about to introduce the new ability of estimating speed and control the DC motor. By using the controller, the speed can be tuned until it get similar to the desired output that user need. Data will be transferred from the controller to the DC motor using the DAQ card. Encoder will be used to detect speed error between the desired output and the measured output.

ABSTRAK

Tujuan projek ini dilaksanakan adalah untuk mengawal kelajuan *DC motor* dengan menggunakan *Fuzzy Logic Controller* sebagai pengawal utama dan diaplikasikan dengan menggunakan MATLAB. Skop kajian adalah merangkumi simulasi dan *modeling DC motor*, perlaksanaan *FLC* ke atas motor sebenar dan membandingkan keputusan antara simulasi daripada MATLAB dan keputusan daripada eksperimen. Kajian ini adalah untuk memperkenalkan keupayaan baru dalam menaksir dan mengawal kelajuan *DC motor*. Dengan menguunakan pengawal, kelajuan motor dapat ditentukan sehingga mendapat keputusan seperti yang telah ditetapkan oleh pengguna. Data akan dihantar daripada pengawal kepada *DC motor* dengan menggunakan kad DAQ. Pengesan akan mengesan sebarang kesilapan pada bacaan kelajuan antara nilai yang ditetapkan dengan nilai yang diperolehi daripada eksperimen.

TABLE OF CONTENT

ii)	IN4148
iii)	IRF740

LIST OF TABLE

LIST OF FIGURES

LIST OF SYMBOL/ABBREVIATIONS

AC- Alternating Current

DAQ – Data Acquisition

DC – Direct Current

FLC – Fuzzy Logic Controller

MATLAB

RTC – Real Time Computing

CHAPTER 1

INTRODUCTION

1.1 Introduction

1.1.1 Fuzzy Logic Controller for Controlling DC Motor speed using MATLAB applications.

DC motor is designed to run on a DC electric power which is used electrical energy and produce mechanical energy. There are two types of DC motor which is brushed and brushless DC motor. Brushless DC motor is chosen in this project because of brushless DC motor (BLDC) is a synchronous electric motor which is powered by DC electricity and which has an electronically controlled commutation system, instead of a mechanical commutation system based on brushes In such motors, current and torque, voltage and rpm are linearly related.

Other than that, brushless DC motor offer several advantages over the brushed DC motor. The advantages that provided by the brushless DC motor are higher efficiency and reliability, reduced noise, longer lifetime and elimination of ionizing sparks from the commutator.

Fuzzy Logic Controller (FLC) is chosen as a controller for this project because it consist several advantages compared to the other classical controller. The advantages of FLC are such as simplicity of control, low cost and the possibility to design without knowing the exact mathematical model of the process. It is suitable for applications such as the speed control of DC motor which has nonlinerities.

The structure of FLC consists of the following 3 major components which the first one is fuzzifier that used for measurement of the input or definition of the fuzzy sets that will applied. The second one is fuzzy control or rule base which provides the system with the necessary decision making logic based on the rule base that determine the control policy. The third method is defuzzifier which combines the action that have been decided and produce single non-fuzzy output that is the control signal of the systems.

1.2 Problem Statement

There are some problem occur while controlling the DC motor, The problem occur such as losses and efficiency of the motor. To encounter the problem, the controller is needed and for this project Fuzzy Logic Controller will be used.

There are too many controllers nowadays but FLC is chosen to interface with the DC motor because it suitable for application which has nonlinearities such speed of the DC motor. Either than that, it has several advantages such as low cost and simplicity of control

1.3 Problem encountered and solutions

Problem encountered

i. Control DC motor speed
ii. Interface DC motor with MATLAB simulink diagram
iii. To acquire data from the DC motor.

Solutions

i. Use FLC as a controller
ii. Implementation of DAQ card to the control board
iii. Use encoder from the DC motor to the control board.

Objective

Objective of the project is to control the speed of DC motor with the Fuzzy Logic Controller using MATLAB applications and to compare the result of the simulation with the experiment.

1.4 Scope

Scopes of the project are:-

i. Simulation and modeling of DC motor
ii. Implement of fuzzy logic controller to actual DC motor
iii. Comparison of the simulation from MATLAB simulation with the experimental result.

CHAPTER 2

LITERITURE REVIEW

2.1 DAQ card

Data acquisition is the sampling of the real world to generate data that can be manipulated by a computer. Data acquisition typically involves acquisitions of signals and waveform and processing the signals to obtain desired information. The components of data acquisition systems include appropriate sensors that convert any measurement parameter to an electrical signal, which acquired by data acquisition hardware.[1]

Acquired data are displayed, analyzed and stored on a computer and control can be developing using various general purposes programming language such as Pascal, Basic C and etc. Specialized programming language used for data acquisitions include EPICS used to build large scale data acquisition systems, LabView which offers a graphical programming environment optimized for data acquisition and MATLAB provides a programming language but also built-in-graphical tools and libraries for data acquisition and analysis.[1]

2.1.1 DAQ toolbox at MATLAB applications

Data Acquisition Toolbox provides a complete set of tools for analog input, analog output, and digital I/O from a variety of PC-compatible data acquisition hardware. The toolbox lets you configure your external hardware devices, read data into MATLAB and Simulink for immediate analysis, and send out data. Figure 9 shows connection between the DAQ toolbox, MATLAB and the hardware. [2]

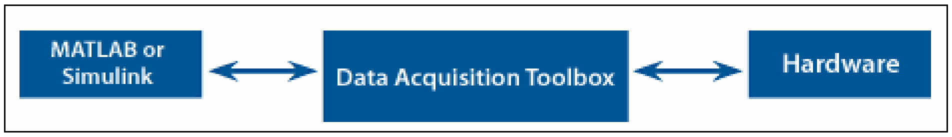


Figure 1: Connection between DAQ toolbox, hardware and MATLAB

The toolbox supports three device objects:-

1. **Analog Input**

-The analog input functions let to acquire signals from the hardware. The analog input object, add channels to the object, acquire data to memory, read data into the workspace, and preview the most recently acquired data can be created by the programmer.[2]

2. **Analog Output**

-Analog output functions let to send signals out to the hardware. The analog output object, add channels, queue data sets to be output, and generate analog signals can be created by the programmer.[2]

Digital I/O

-Digital I/O functions enable to generate or read digital signals using the hardware. The digital I/O objects, add lines, send data to the hardware, and read data into the workspace can be created by the programmer.[2]

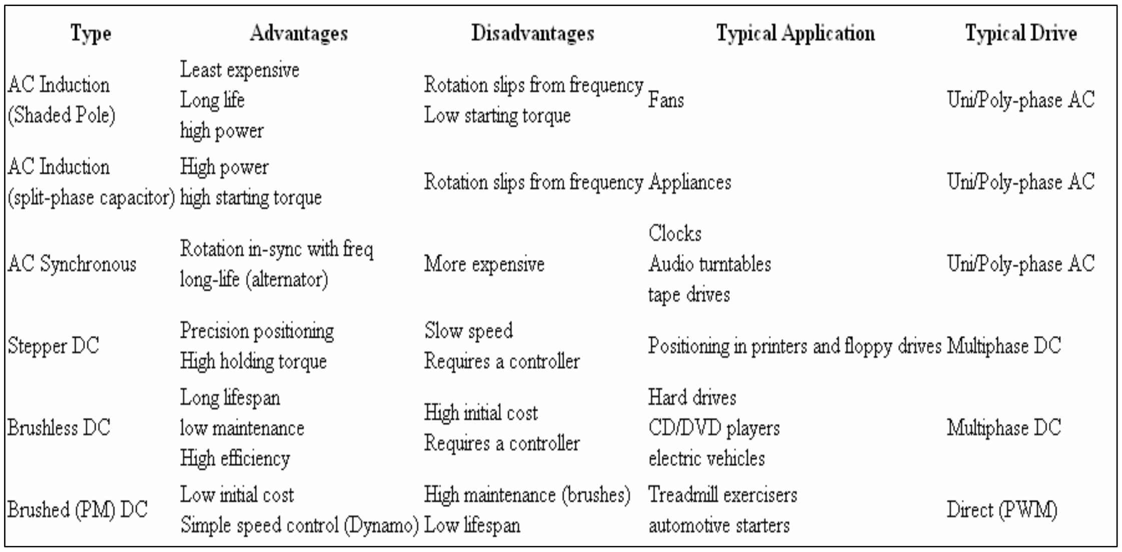
2.2 DC Motor

The electric motor uses an electrical energy to produce a mechanical energy. The principles of electrical energy into mechanical energy by electromagnetic means was demonstrated by the British scientist Michael Faraday and consisted of a free hanging wire dipping into a pool of mercury. A permanent magnet was placed in the middle of the pool of mercury. When a current passé through the wire, the wire rotated around the magnet, showing that the current gave rise to a circular magnetic field around the wire.[3]

There are two divisions of electric motors which is Direct Current (DC) and Alternating Current (AC). The ongoing trend toward electronic control further muddles the distinction, as modern drivers moved the commutator out of the motor shell. For this new breed of motor, driver circuits are relied upon to generate sinusoidal AC drive currents. The two best examples are: the brushless DC motor and the stepping motor which both being polyphase AC motors requiring external electronic control.[3]

There is a clearer distinction between a synchronous motor and asynchronous types. In the synchronous types, the rotor rotates in synchrony with the oscillating field or current. In contrast, an asynchronous motor is designed to slip; the most ubiquitous example being the common AC induction motor which must slip in order to generate torque.[3]

The classic DC motor design generates an oscillating current in a wound rotor with a split ring commutator or permanent magnet stator. A rotor consists of a coil wound around a rotor which is then powered by any type of battery. There are three types of DC motor which is stepper DC motor, brushed DC motor and brushless DC motor. Figure 8 below shows the comparison of motor types.[3]



Type	Advantages	Disadvantages	Typical Application	Typical Drive
AC Induction (Shaded Pole)	Least expensive Long life high power	Rotation slips from frequency Low starting torque	Fans	Uni/Poly-phase AC
AC Induction (split-phase capacitor)	High power high starting torque	Rotation slips from frequency	Appliances	Uni/Poly-phase AC
AC Synchronous	Rotation in-sync with freq long-life (alternator)	More expensive	Clocks Audio turntables tape drives	Uni/Poly-phase AC
Stepper DC	Precision positioning High holding torque	Slow speed Requires a controller	Positioning in printers and floppy drives	Multiphase DC
Brushless DC	Long lifespan low maintenance High efficiency	High initial cost Requires a controller	Hard drives CD/DVD players electric vehicles	Multiphase DC
Brushed (PM) DC	Low initial cost Simple speed control (Dynamo)	High maintenance (brushes) Low lifespan	Treadmill exercisers automotive starters	Direct (PWM)

Figure 2: Comparison of motor types

These projects choose permanent magnet DC motor to control by the controller because it provides high torque at low speeds. Specific benefits of this type of motor include simplified power supply requirements, less cooling, and reduced frame size for a given power output. The custom permanent magnet motors can be controlled to very precise speeds, making it especially useful in servo applications.[4]

2.3 Fuzzy Logic Controller

Fuzzy Logic Controller (FLC) is based on fuzzy logic controller and constitutes a way of converting linguistic control strategy into an automatic by generating a rule base which controls the behavior of the system. Fuzzy control is control method based on fuzzy logic. Fuzzy provides a remarkably simple way to draw definite conclusions from vague ambiguous or imprecise information. It suitable for applications such as the speed control of dc motor which is has non linearities. [5]

FLC have some advantages compared to other classical controller such as simplicity of control, low cost and the possibility to design without knowing the exact mathematical model of the process. Fuzzy logic incorporates an alternative way of thinking which allows modeling complex systems using higher level of abstraction originating from the knowledge and experience. Fuzzy logic can be described simply as "computing words rather than numbers" or "control with sentence rather than equations."[5]

The applications of fuzzy logic are usually for household appliance such as washing machine and rice cooker. Fuzzy also been used in industrial process such as cement kilns, underground trains and robots. [5]

CHAPTER 3

METHODOLOGY

3.1 Fuzzy Logic Controller

3.1.1 Structure of Fuzzy Logic

There are specific components characteristic of a fuzzy controller to support a design procedure. Figure 3 shows the controller between the preprocessing block and post processing block.[6]

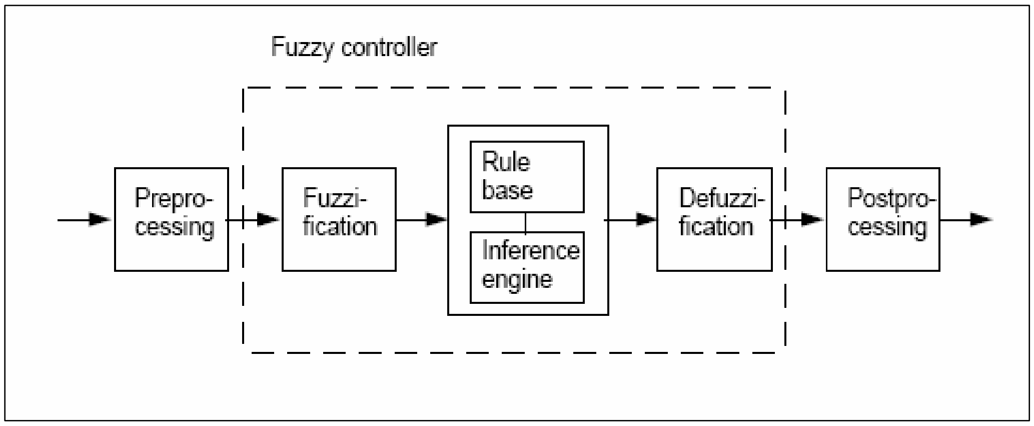


Figure 3: Structure of fuzzy logic controller

3.1.1.1 Preprocessing

The inputs are most often hard or crisp measurement from some measuring equipment rather than linguistic. A preprocessor, the first block in Figure 1 shows the conditions the measurements before enter the controller.[6]

3.1.1.2 Fuzzification

The first block inside the controller is fuzzification which converts each piece of input data to degrees of membership by a lookup in one or several membership functions. The fuzzification block matches the input data with the conditions of the rules to determine. There is degree of membership for each linguistic term that applies to the input variable.[6]

3.1.1.3 Rule Base

The collection of rules is called a rule base. The rules are in “*If Then”* format and formally the *If side* is called the *conditions* and the *Then side* is called the *conclusion.* The computer is able to execute the rules and compute a control signal depending on the measured inputs *error* (e) and *change in error*.(dE). In a rule based controller the control strategy is stored in a more or less natural language. A rule base controller is easy to understand and easy to maintain for a non- specialist end user and an equivalent controller could be implemented using conventional techniques.[6]

3.1.1.4 Defuzzification

Defuzzification is when all the actions that have been activated are combined and converted into a single non-fuzzy output signal which is the control signal of the system. The output levels are depending on the rules that the systems have and the positions depending on the non-linearities existing to the systems. To achieve the result, develop the control curve of the system representing the I/O relation of the systems and based on the information; define the output degree of the membership function with the aim to minimize the effect of the non-linearity.[6]

3.1.1.5 Postprocessing

The postprocessing block often contains an output gain that can be tuned and also become as an integrator.[6]

3.1.2 Fuzzy Logic Toolbox

There are five primary graphical user interface (GUI) tools for building, editing and observing fuzzy inference systems in the toolbox:-

- Fuzzy Inference System (FIS) editor

- Membership Function editor
- Rule Editor
- Rule Viewer
- Surface Viewer

These GUI are dynamically linked and if the changes make to the FIS to the one of the toolbox, the effect can be seen in other GUIs. In addition to these five primary GUIs, the toolbox includes the graphical ANFIS Editor GUI, which is used for building and analyzing Sugeno-types adaptive neural fuzzy inference systems [7]

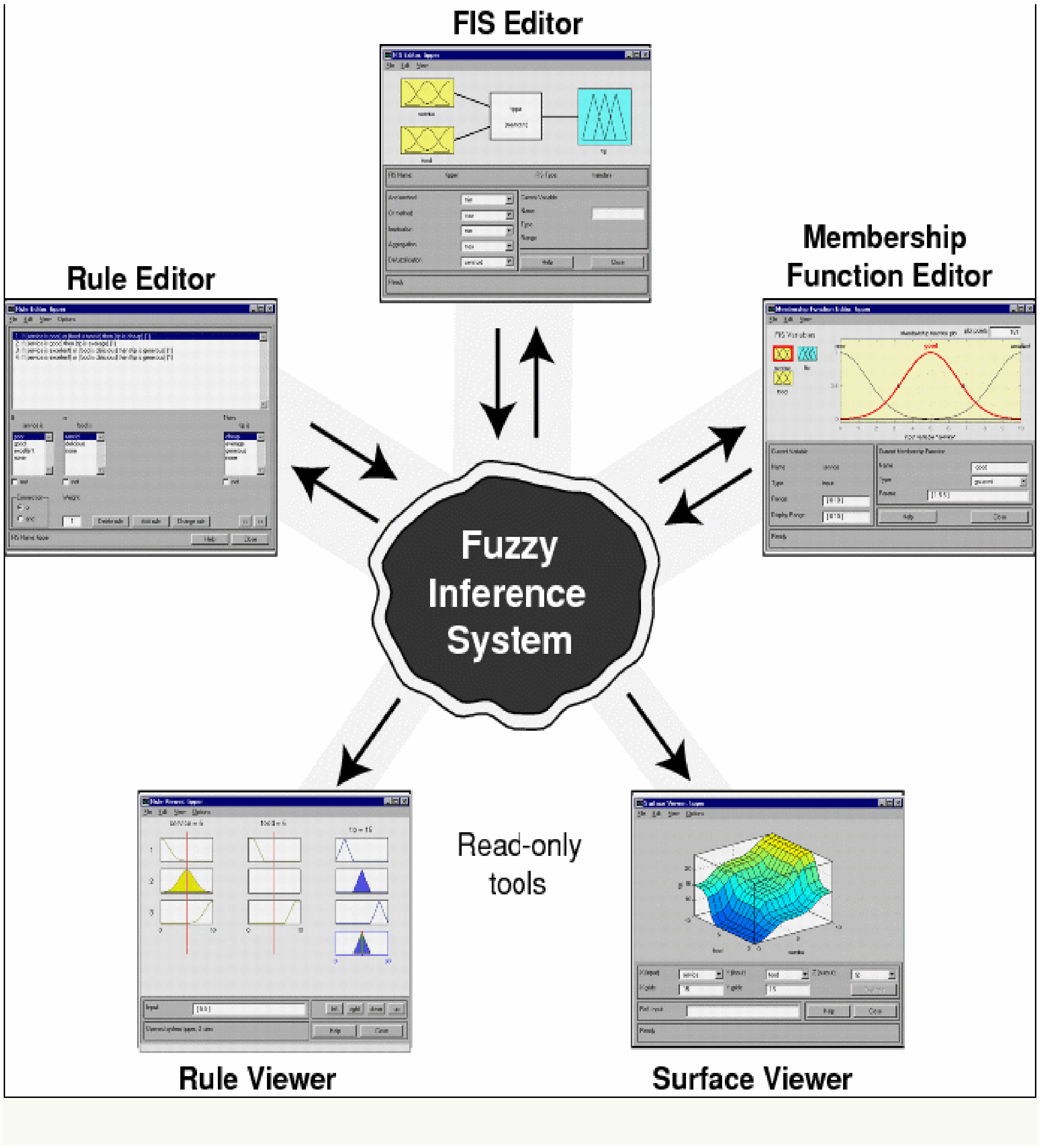


Figure 4: Fuzzy Inference System

3.1.2.1 FIS Editor

The FIS editor handles the high level issues of the system. Fuzzy Logic toolbox does not limit The FIS editor displays general information about fuzzy inference systems. There is a simple diagram at the top shows the name of each input variable on the left and the output on the right. The step below is show how to open the FIS editor:- [7]

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

ii. Select **Edit** > **Add Variable** > **Input** (if the system need two inputs)

iii. Click the yellow box **input1**. This box is highlighted with a red outline

iv. Edit the **Name** field from input1 to service, and press **Enter.**

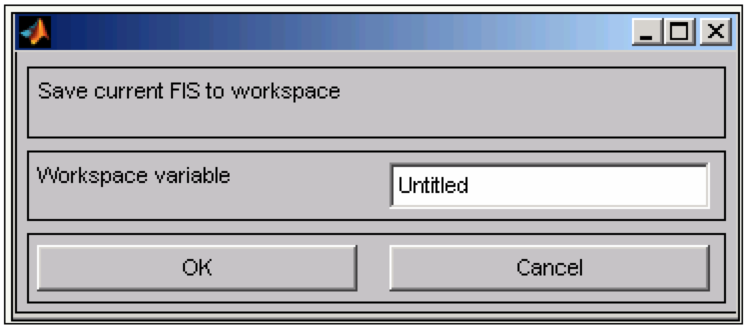
v. Click the yellow box **input2**. This box is highlighted with a red outline/

vi. Edit the **Name** field from input2 to food, and press **Enter.**

vii. Click the blue box **output1.**

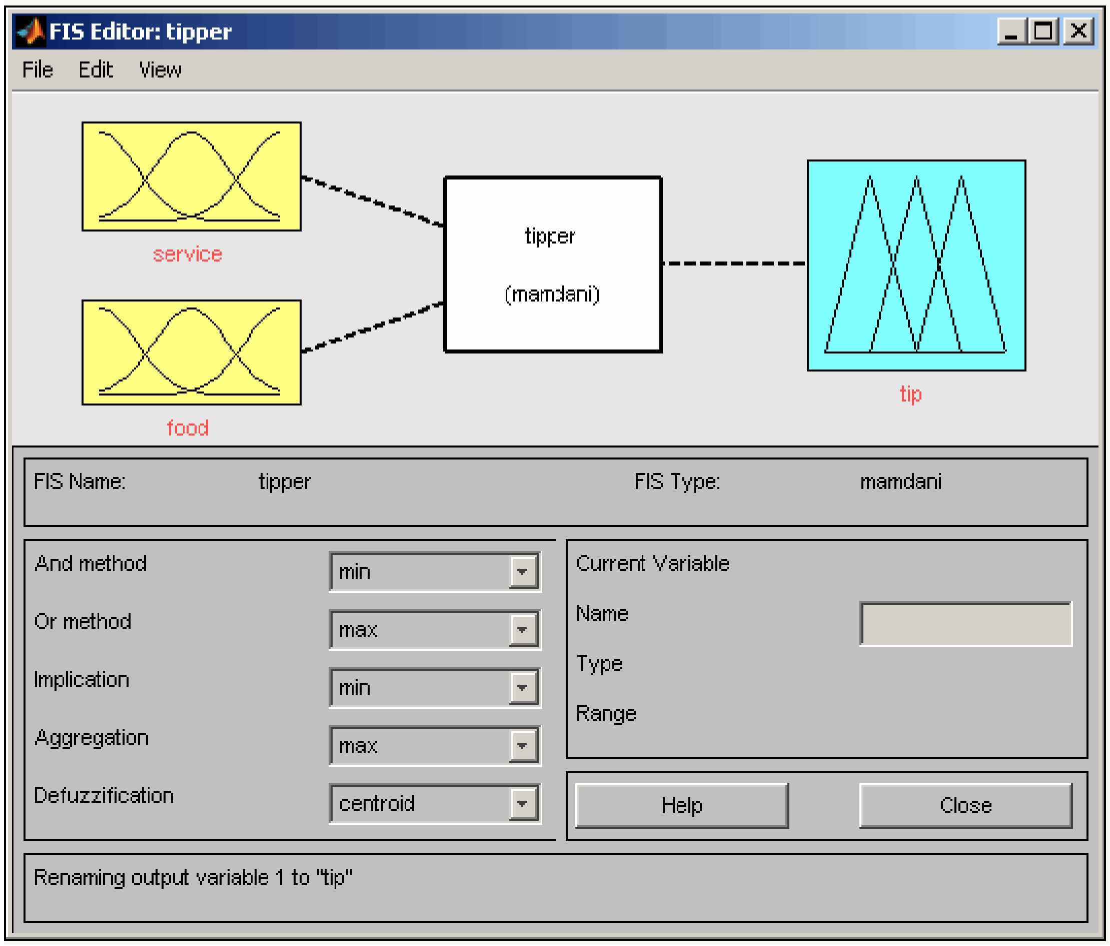
viii. Edit the **Name** field from output1 to tip, and press **Enter.**

ix. Select **File** > **Export** > **To workspace**


Save current FIS to workspace

Workspace variable: Untitled

x. Enter the **Workspace variable** name **tipper**, and click **OK**.



FIS Name: tipper

FIS Type: mamdani

And method: min

Or method: max

Implication: min

Aggregation: max

Defuzzification: centroid

Current Variable

Name

Type

Range

Renaming output variable 1 to "tip"

Figure 5: FIS editor

3.1.2.2 Membership Function Editor

The membership function editor shares some features with the FIS editor. The membership function editor is the tool that lets the programmer displays and edits all of the membership functions associated with all inputs and output variables for entire fuzzy inference system. [7]

The step below shows how to open the membership function editor:-

- Within the FIS editor windows, select **Edit** > **Membership functions**
- Within the FIS editor, double click the blue icon called **tip**
- At the command line, type **mfedit**

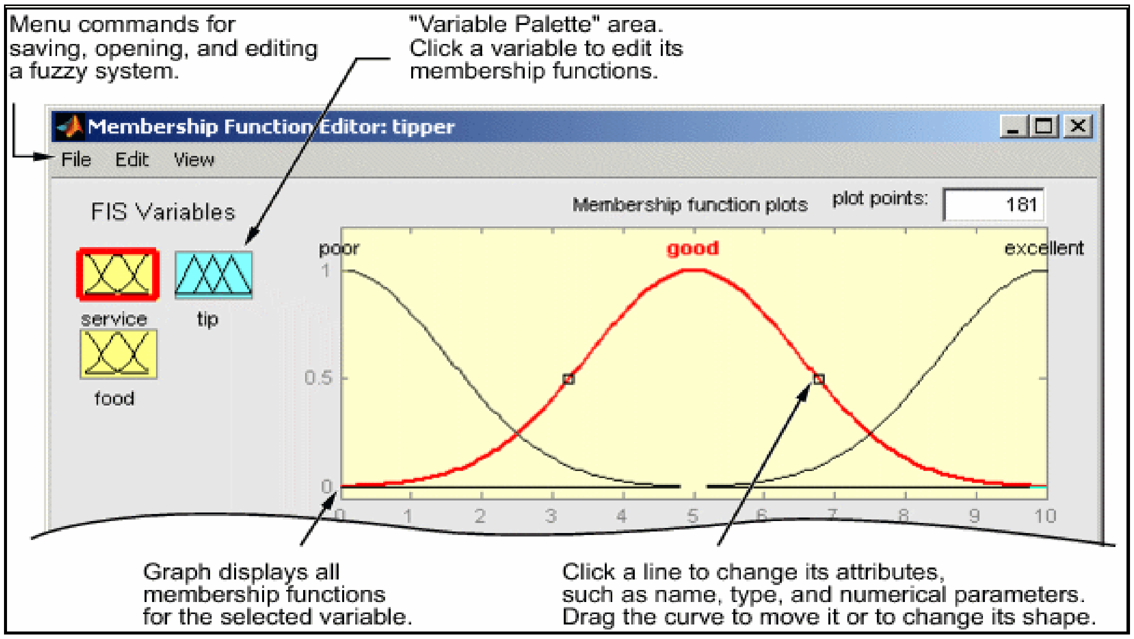

Menu commands for saving, opening, and editing a fuzzy system.
"Variable Palette" area. Click a variable to edit its membership functions.
Membership Function Editor: tipper
File
Edit
View
FIS Variables
service
tip
food
Membership function plots
plot points:
181
poor
good
excellent
1
0.5
0
0
1
2
3
4
5
6
7
8
9
10
Graph displays all membership functions for the selected variable.
Click a line to change its attributes, such as name, type, and numerical parameters. Drag the curve to move it or to change its shape.

Figure 6: Membership Function Editor

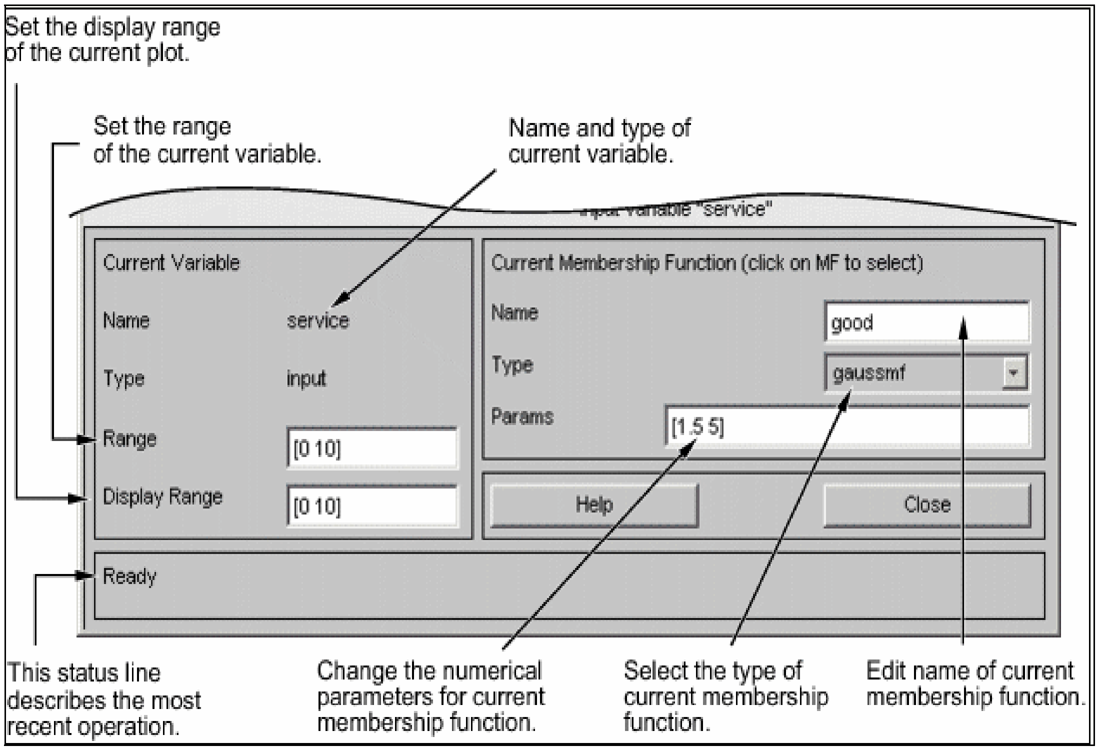
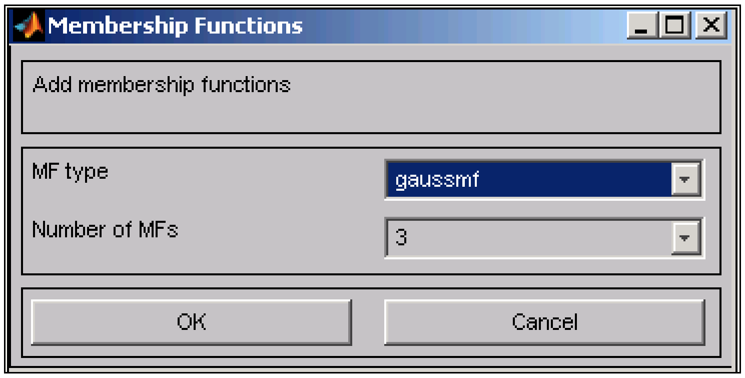

Set the display range of the current plot.
Set the range of the current variable.
Name and type of current variable.
input variable "service"
Current Variable
Name
service
Type
input
Range
[0 10]
Display Range
[0 10]
Current Membership Function (click on MF to select)
Name
good
Type
gaussmf
Params
[1.5 5]
Help
Close
Ready
This status line describes the most recent operation.
Change the numerical parameters for current membership function.
Select the type of current membership function.
Edit name of current membership function.

Figure 7: FIS Editor for Membership Function variable

The process of specifying the input membership function for two input tipper problem is a follows:

1. Select **input** variable, service, double clicking on it. Set both the **Range** and the **Display Range** to the vector [0 10]
2. Select **Remove All MFs** from **Edit** menu. It removes all the existing Membership Function from the Membership Function Editor**.**
3. Select **Add MFs** from **Edit** menu. The following window opens:



Add membership functions

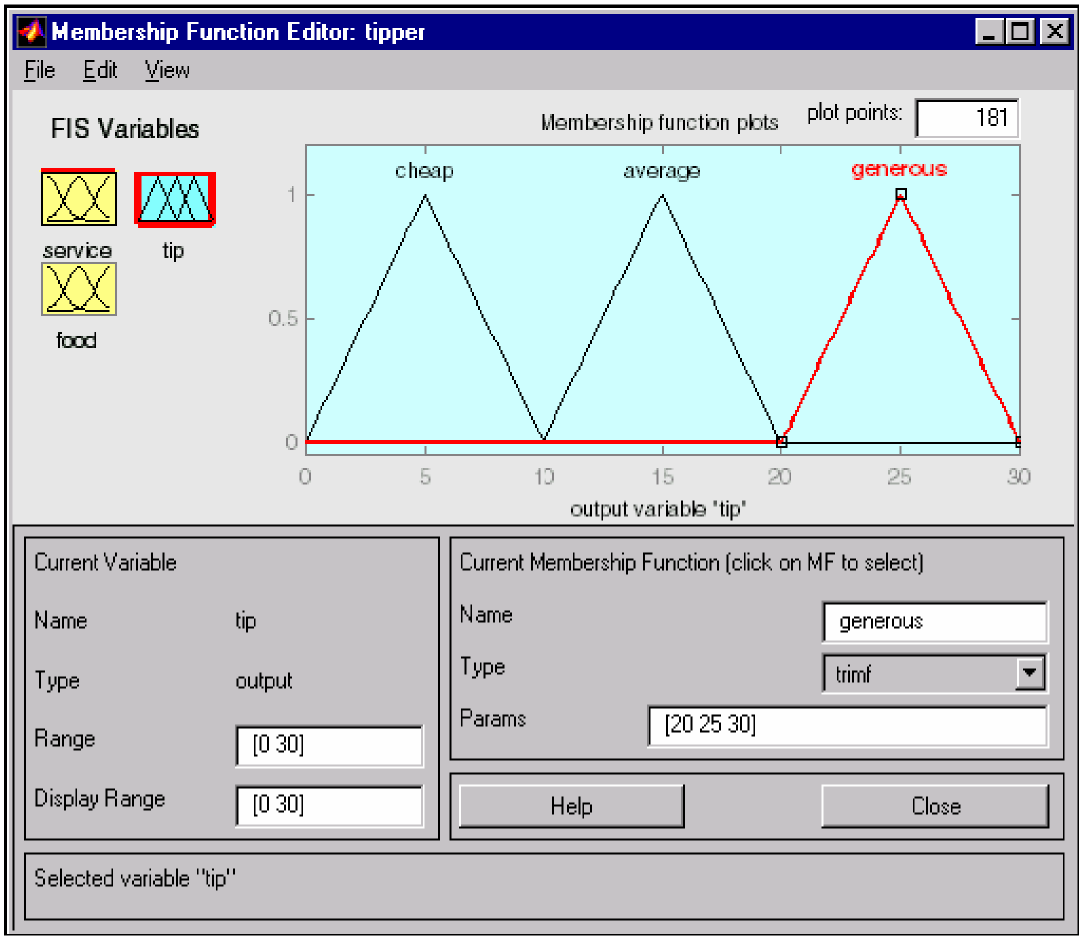
MF type: gaussmf

Number of MFs: 3

4. Use tab to choose “gaussmf” for **MF Type** and 3 **Number of MFs**. This choice adds three Gaussian curves to the input variable service.

5. Click once on the curve with the left-most *hump*. Change the name of the curve to poor. To adjust the shape of the membership function, click on the **membership function**. The desired parameter **Params** listing for this will appear. The two inputs of **Params** represent the standard deviation and center for the Gaussian curve.

6. After editing all the value and adjusted the membership function, the system will look similar to the following figure :-



plot points: 181

Current Variable

Name tip

Type output

Range [0 30]

Display Range [0 30]

Current Membership Function (click on MF to select)

Name generous

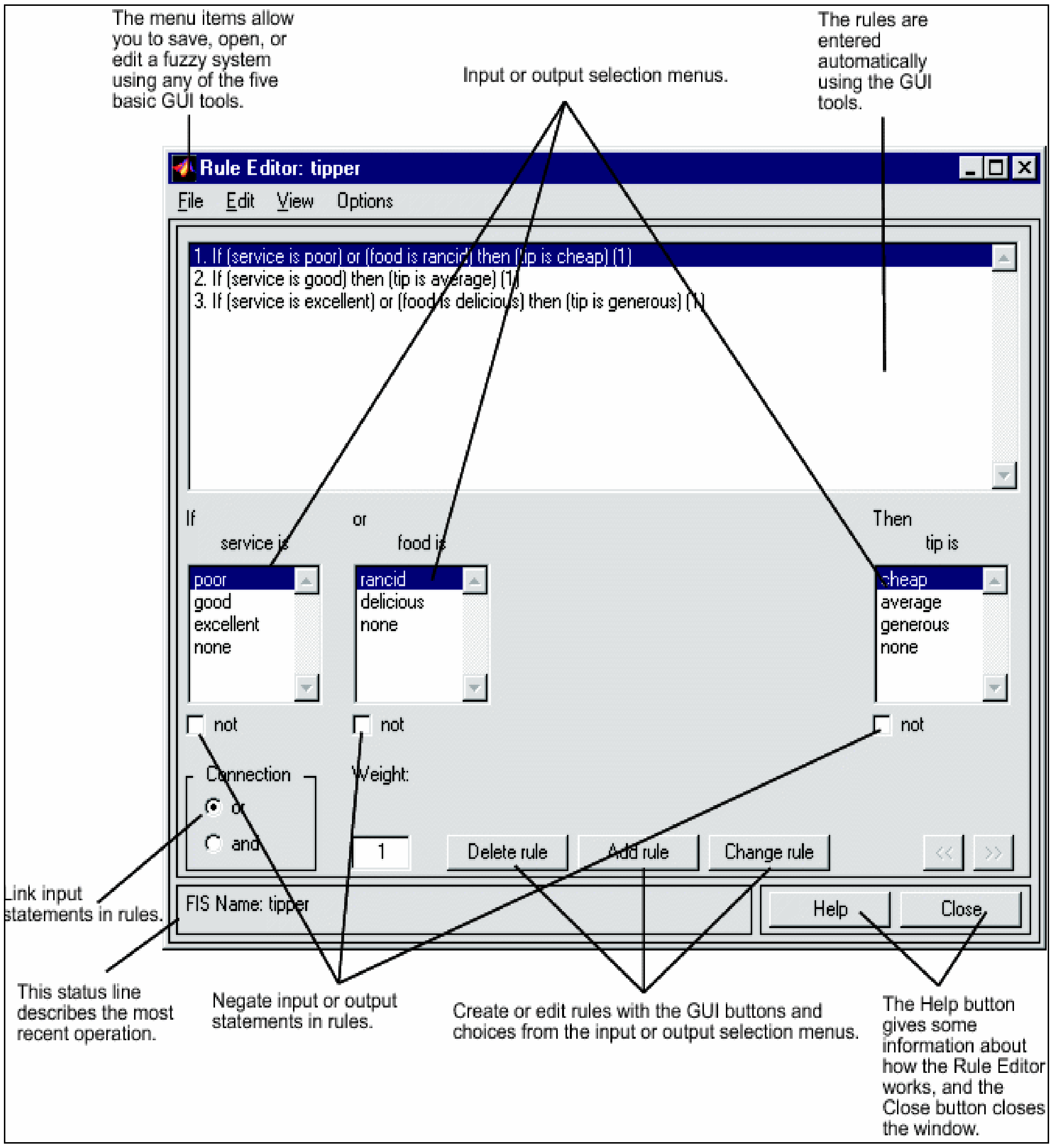
Type trimf

Params [20 25 30]

Selected variable "tip"

3.1.2.3 Rule Editor

To call up the **Rule Editor**, choose **Edit** menu and select **Rules** or type *ruleedit* at command line.[7]



The menu items allow you to save, open, or edit a fuzzy system using any of the five basic GUI tools.

Input or output selection menus.

The rules are entered automatically using the GUI tools.

1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is good) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is generous) (1)

FIS Name: tipper

Link input statements in rules.

This status line describes the most recent operation.

Negate input or output statements in rules.

Create or edit rules with the GUI buttons and choices from the input or output selection menus.

The Help button gives some information about how the Rule Editor works, and the Close button closes the window.

Figure 8: Rule Editor

Based on the description of the input and output variable defined with the FIS Editor, the Rule Editor allows to construct the rule statements automatically. From GUI:-

- Create rules by selecting an item in each input and output variable box and one **Connection** item and clicking **Add Rule**. You can choose none as one of the variable qualities to exclude that variable from a given rule and choose not under any variable name to negate the associated quality.
- Delete a rule by selecting the rule and clicking **Delete Rule**.
- Edit a rule by changing the selection in the variable box and clicking **Change Rule**.
- Specify weight to a rule by typing in a desired number between 0 and 1 in **Weight**. If you do not specify the weight, it is assumed to be unity (1).

3.1.2.4 Rule Viewer

The Rule Viewer displays a roadmap of the whole fuzzy inference process. It based on the fuzzy inference. The three plots across the top of the Figure 7 represent the antecedent and consequent of the first rule. Each rule is a row of plots, and each column is a variable. The rule numbers are displayed on the left of each row. To view the rule in the status line click the rule number.[7]