NEAR EAST UNIVERSITY

GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

FUZZY CONTROLLER DESIGN

Mohamed Elhag

Master Thesis

Department of Computer Engineering

Nicosia- 2002

Mohamed Elhag: Fuzzy Controller Design

Approval of the Graduate School of Applied and Social Sciences

Prof. Dr. Fakhraddin Mamedov
Director

We certify that this thesis is satisfactory for the award of the degree of Master of Sciences in Computer Engineering

Examining Committee in Charge:

Prof. Dr. Fakhraddin Mamedov, Chairman of Committee, Dean of Engineering Faculty, NEU

Assoc. Prof. Dr. Adnan Khashman, Chairman of the Electrical and Electronic Engineering Department, NEU

Assoc. Prof. Dr. Doğan İbrahim, Chairman of the Computer Engineering Department, NEU

ACKNOWLEDGEMENTS

"First, I would like to thank my supervisor Assoc. Prof. Dr. Rahib Abiyev for his valuable advice, encouragement and endless support.

Second, I would like to acknowledge special thank to the Near East University for offering me a suitable environment during my study. And also I will never forget the following teachers support and help Prof. Dr. Şenol Bektaş, Prof. Dr. Fakhraddin Mamedov, Assoc. Prof. Dr. Adnan Khashman, Assoc. Prof. Dr. Doğan İbrahim. And special thank to Mr. Tayseer Alshanableh.

Third, I would like to dedicate my research to my parents and the rest of the family who are always motivate me with all the love.

I gratefully acknowledge the role of my friends those who set behind me while I'm preparing this project..

Finally, Special thank to the Jury members who offering me this great opportunity to present my thesis"

ABSTRACT

Fuzzy logic is nowadays applied in almost all sectors of industry and science in the whole world, especially in the field of control and pattern recognition.

The aim of thesis is the development of the fuzzy controller for technological processes control. To achieve this aim the structures and operation principles of fuzzy PD-, PI-, PID-Like controllers are given. The functions of the main blocks-fuzzification, inference engine, defuzzification, fuzzy knowledge base are described.

The development of fuzzy PD-, PI-, PID-Like controllers are performed. Using time response characteristics of system and fuzzy model of the processes the fuzzy knowledge base for this controller are developed. The inference engine mechanism is realized by using max-min type fuzzy processing of Zadeh. Defuzzification mechanism is realized by using "Center of Gravity" algorithm.

The modeling of fuzzy PD-Like controller for control of temperature of heater is carried out. The simulation of system is realized in MATLAB Programming Language. In the result of simulation obtained time response characteristics of system show the efficiency of application of fuzzy controller in complicated processes.

TABLE OF CONTENTS

INTRODUCTION

Presently large class of industrial processes are characterized with non-linearity, time-variance, the overlapped presence of various disturbance and so on. As a result, it is difficult to develop sufficiently adequate models of these processes and, consequently, to design a control system using traditional methods of the control theory, even if sophisticated mathematical models are applied.

At the same time it is surprising that a skilled human-expert successfully performs his duties due to a great amount of qualitative information that he uses intuitively while elaborating a control strategy. Usually, he keeps in mind this information in the form of linguistic rules, which make up an intrinsic control algorithm. Furthermore, a human operator often is able to aggregate a great amount of quantitative information, to extract most essential peculiarities and interconnections as well as to define the most important qualitative control indices.

Fuzzy set theory was found to be a very effective mathematical tool for dealing with the modeling and control aspects of complex industrial and non industrial processes as an alternative to other much more sophisticated mathematical models. Further, the latter circumstance led to the appearance at the beginning of the 1970's of fuzzy logic computer controllers which became a powerfully tool for coping with the complexity and uncertainty with which we are faced in many real-world problems of industrial process control. The first investigations in this field had to answer the question: *Is it possible to realize a process controller which deals like a man with the involved linguistic information?* The results of these inquires led to the design of the first fuzzy control systems which implemented in hardware and software a linguistic control algorithm. Such a control algorithm was then formulated by a control engineer on the base of the interviews with human experts who currently work as process operators. The most simple fuzzy feedback control systems contain a fuzzy logic controller (FLC) in the form of a Table of linguistic rules, or fuzzy relation matrix and input-output interfaces.

Fuzzy logic has been successfully applied to many of industrial spheres, in robotics, in complex decision-making and diagnostic system, for data compression, in TV and others. Fuzzy sets can be used as a universal approximator, which is very important for modeling unknown objects. Fuzzy technology has such characteristics as

interpretability, transparency, plausibility, graduality, modeling, reasoning, imprecision tolerance.

The aim of the thesis is the development of a fuzzy controller for technological processes control. The thesis consists of introduction, five chapters and conclusion.

Chapter One describes the structure of the fuzzy system, the functions of it's main blocks. The structures of PD, PI and PID-Like fuzzy controllers and their operation principles are described.

Chapter Two presents the algorithms of fuzzy controllers. The linguistic variables, their fuzzy values and different fuzzification algorithms are describe. The steps of inference engine mechanism are also describe. Different types of fuzzy processing mechanisms are given in this chapter as well.

Chapter Three is devoted to the development of PD, PI and PID-Like fuzzy controllers for technological processes control. As a result, the fuzzy rule base controllers have been generated.

Chapter Four is devoted to the development of fuzzy controller for control temperature of heater. The description of the processes is given. The realization of each block of fuzzy controller are described.

Chapter Five describes the computer simulation of fuzzy system for control of temperature of heater by using Matlab package. The result of simulation is analyzed.

Conclusion presents the obtained important results and contributions in the thesis.

CHAPTER ONE

THE STRUCTURE OF FUZZY CONTROLLERS

1.1 Overview

In practice conventional controllers are often developed via simple models of the plant behavior that satisfy the necessary assumptions, and via the ad hoc tuning of relatively simple linear or nonlinear controllers. Regardless, it is well understood that heuristic enter the conventional control design process as long as we are concerned with the actual implementation of the control system. It must be acknowledged, moreover, that conventional control engineering approaches that use appropriate heuristics to tune the design have been relatively successful.

Fuzzy control provides a formal methodology for representing, manipulating, and implementing a human's heuristic knowledge about how to control a system.

In this chapter the structure of fuzzy systems and the functions of their main blocks are described. Section 1.2 describes the structure of general fuzzy system. Sections 1.3, 1.4 and 1.5 are devoted to the structures and operation principles of fuzzy PD-, PI- and PID-like fuzzy controllers.

1.2 Structure of General Fuzzy System

There are specific components characteristic of a fuzzy controller to support a design procedure. A general structure of fuzzy controller is described in the block diagram shown in Figure 1.1.

A fuzzy system is static nonlinear mapping between its inputs and outputs (i.e., it is not a dynamic system). It is assumed that the fuzzy system has inputs $u_i \in U_i$ where i =1,2,…..,n and outputs $y_i \in Y_i$ where i =1,2,…..,m. The inputs and outputs are "crisp" – that is, they are real numbers, not fuzzy sets.

The fuzzy controllers are composed of the following four elements:

1. *A rule Base* (a set of If-Then rules), which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.
2. *An Inference Mechanism* (also called an "inference engine" or "fuzzy inference" module), which emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant.

3. *A Fuzzification Inference*, which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules
4. *A defuzzification Inference,* which converts the conclusions of the inference mechanism into actual inputs for the process (converts fuzzy conclusion into crisp outputs).

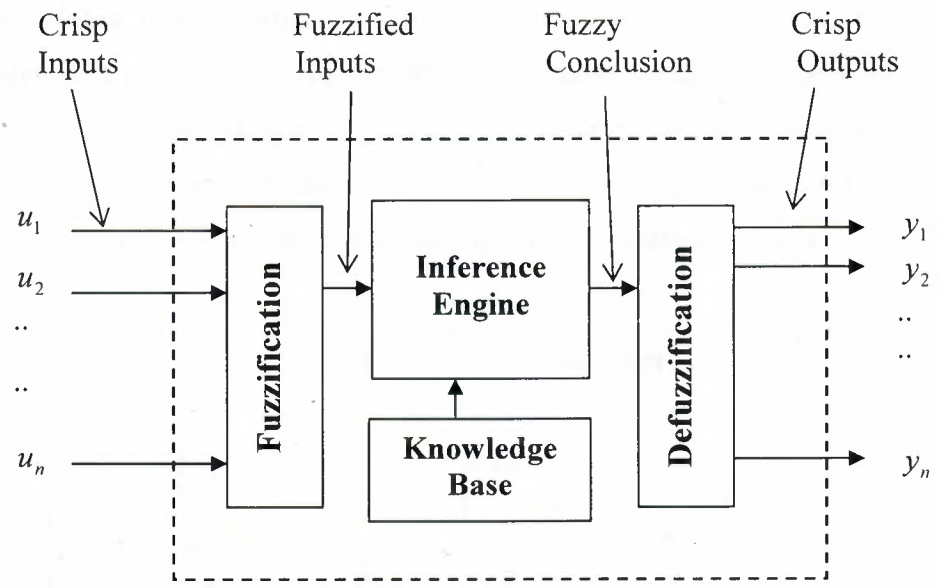


Figure 1.1. Structure of fuzzy controller.

1.3 Structure of The PD-Like Fuzzy Controller

The most simple fuzzy feedback control system contains a fuzzy logic controller (FLC) in the form of a Table of linguistic rules (or fuzzy relations matrix) and input-output interfaces. A linguistic rule consists of one or more premises and one or more consequences, i.e. in the form:

IF (premises:a and b and c...) hold

THEN (consequences:x and y and z...) hold too.

The structure of PD-Like fuzzy controller is shown in Figure 1.2.

A PD-Like fuzzy controller presents an information loop with:

- an input signal g as an advising set-point (for example, a quality control);
- a comparator which checks, if the emitted process output x is the correct reaction; to the set-point g, and which emits himself an error signal e as an input to the decision element Table of Linguistic Rules (TLR) (or Rule Base), in order to report him, how much the process output x deviates from the preset value of g;

- calculating change of errror
- a decision element TLR which emits for each value of *e* and change of error *e'* and output *u* which, on its side, becomes an input to a process with output *x* to be controlled.

A fuzzy logic controller is a synthesis of both, a controller's loop and a set of linguistic rules which are the content of the decision elements of the controller. The purpose of the input interface is to convert the non-fuzzy signals of error, either derivative (e`) or sum of error (or both) into those input fuzzy sets which serve as premises in the correspondent linguistic rule of the FLC. The output fuzzy set (or the consequent of the linguistic rule) is converted by the output interface to the non-fuzzy control action which is transferred to the input of an industrial process.

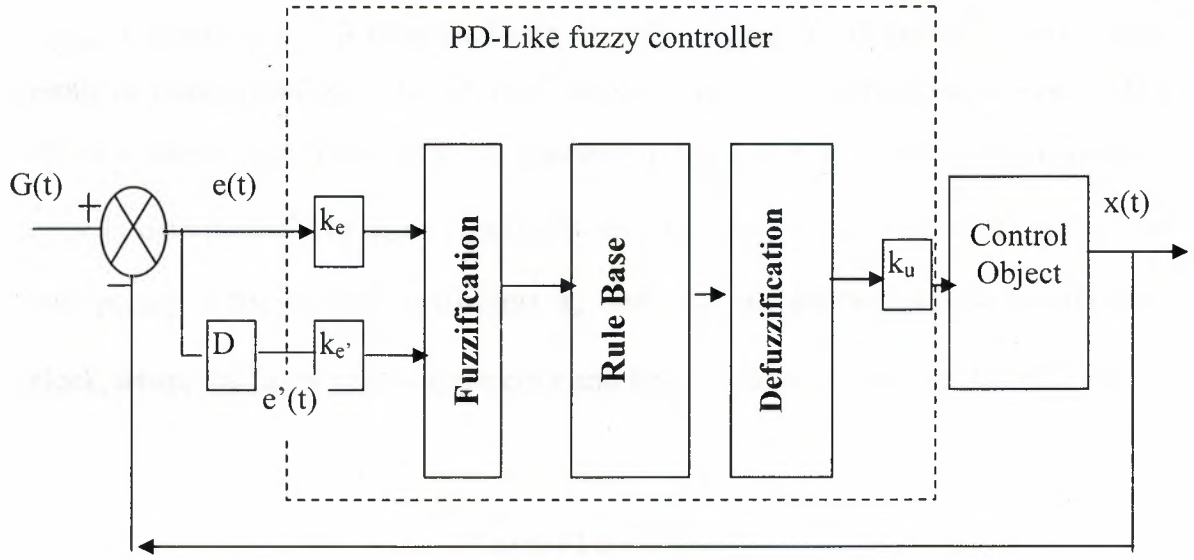


Figure 1.2. Structure of fuzzy PD control system.

The transient performance demonstrated by these controllers as well as the noise immunity and robustness were essentially better than that of usual PID (Proportional, Integral, Differential) controllers. At the same time, the practical use of fuzzy control systems revealed the following problems:

a. there is not yet a satisfactory approach to the construction of input-output interfaces being sufficiently supported by logical evidence;

b. there is no definitive agreement about how to proceed with an incomplete Table of linguistic rule (TLR). Thus, no actual rule in the TLR can be applied to a concrete decision case, if the features of parameters *p* of this case appear no where in

the TLR as premises. Then, a new consequent *c*, as the missing term of a new rule *r(p,c)* must be introduced (this is done, for instance, by interviewing the human process operator). On the other hand, the TLR demands an expensive study of the process and does not guarantee a desirable transient performance of the system in the case of a time variant process.

Moreover, the efficiency of fuzzy systems depends on the completess of the experts interviewed during the knowledge elicitation process. Therefore, a wide application of single-loop fuzzy control systems is restricted, because of their inability to cope with complex decision cases.

1.4 Structure of The PI-Like Fuzzy Controller

The structure of the fuzzy PI-Like controller is shown in Figure 1.3. The output signal of control object is compared with the target signal G(t) in the comparator. In the result of comparison the value of error between target and current signals of control object is determined. This signal e(t) is passed to integrator $\int$ and the integral value of error is determined. The error signal e(t) and the integral value of error $\int e(t)$ after multiplying to the scaling coefficients k_e and $k_{\int}$ are entered to the fuzzification block, where the fuzzy values of the error and integral value of error are determined.

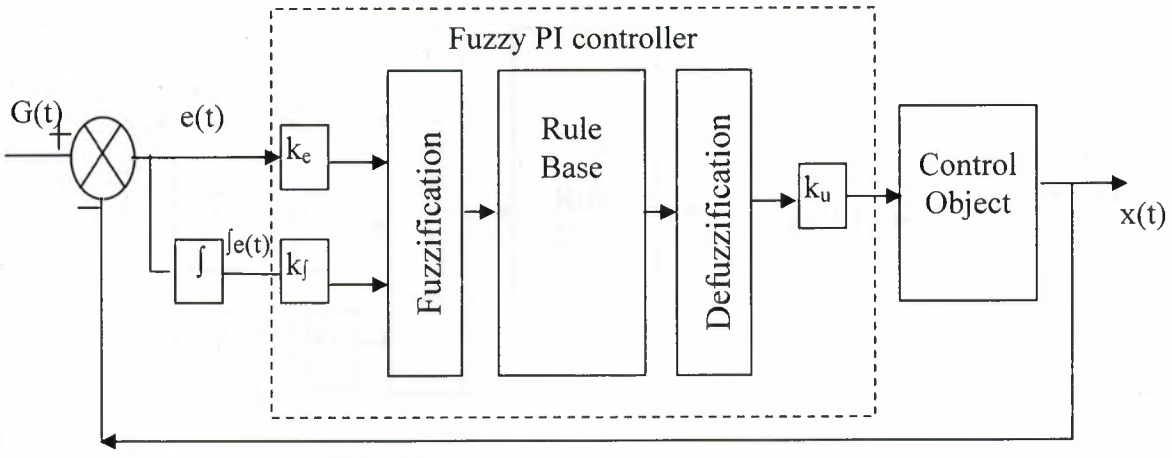


Figure 1.3. Structure of fuzzy PI control system.

Using rule base block the fuzzy output of the controller is determined. It is input for defuzzification block. Defuzzification block calculates the crisp output of the controller. This output signal after scaling is entered to the plant input.

1.5 Structure of The PID-Like Fuzzy Controller

The fuzzy PID-Like controller is a combination of fuzzy PD-Like controller and a fuzzy PI-Like controller. The structure of fuzzy PID-Like controller is shown in Figure 1.4. In the result of the comparison of output signal of control object with target signal of control system the value of error is determined. This signal e(t) is passed to the integrator and differentiator. On the output of integrator and differentiator the integral value of error and change of error are determined.

The error signal e(t), velocity of error e'(t) and the integral value of error $\int e(t)$ after multiplying to the scaling coefficients k_e , $k_{e'}$ and $k_{\int}$ are entered to the fuzzification block, where the fuzzy values of error, the velocity value of error and integral value of error are determined.

Using rule base block the fuzzy output of the controller is determined. This signal after defuzzification in the defuzzification block is scaled and entered to the plant input.

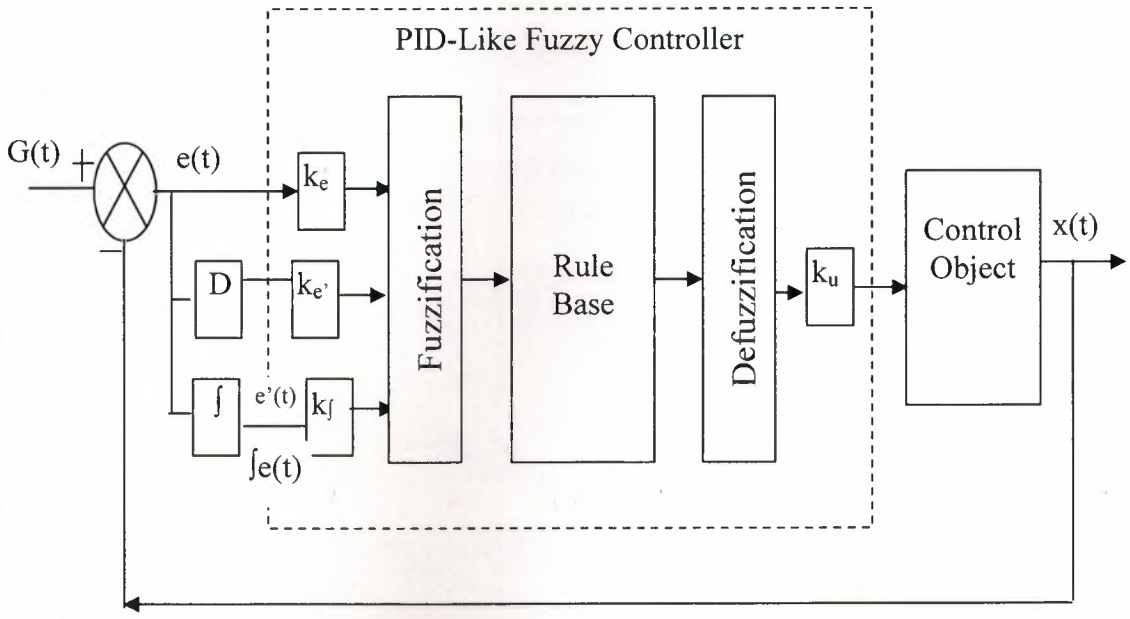


Figure 1.4. Structure of fuzzy PID control system.

1.6 Summary

A fuzzy system is a static nonlinear mapping between its inputs and outputs (i.e., it is not a dynamic system).

The fuzzy controller's are composed of the following four elements:

1. *A rule Base* (a set of If-Then rules), which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.
2. *An Inference Mechanism* (also called an "inference engine" or "fuzzy inference" module), which emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant.
3. *A Fuzzification Inference*, which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules
4. *A defuzzification Inference,* which converts the conclusions of the inference mechanism into actual inputs for the process (converts fuzzy conclusion into crisp outputs).

In this chapter a full description of the PD, PI, and PID-Like fuzzy controllers structures are given.

In next chapter, different operations of fuzzy controller will be described in details.

CHAPTER TWO

ALGORITHMS OF FUZZY CONTROLLERS

2.1 Overview

In order to construct a fuzzy application a sufficient knowledge on how to operate the system that is to be controlled is required. The performance of the fuzzy controller can be influenced by changing the shape and number of its membership functions, by changing its defuzzification method and its inference mechanism. These operations can be done in relatively easy manner without need for knowledge of all system parameters and without use of mathematical operations of any kind.

In this chapter, the operations of fuzzy controller are described in details. The entire operations inside the fuzzification, inference mechanism and defuzzification blocks are shown.

2.2 Operations of Fuzzy Controller

The inference engine is the heart of a fuzzy controller (and any fuzzy rules system) operation. The actual operation of the fuzzy controller can be divided into three steps as shown in Figure 2.1:

- Fuzzification: actual inputs are fuzzified and fuzzy inputs are obtained.
- Fuzzy processing: processing fuzzy inputs according to the rules set and producing fuzzy output.
- Defuzzification: producing a crisp real value for fuzzy output.

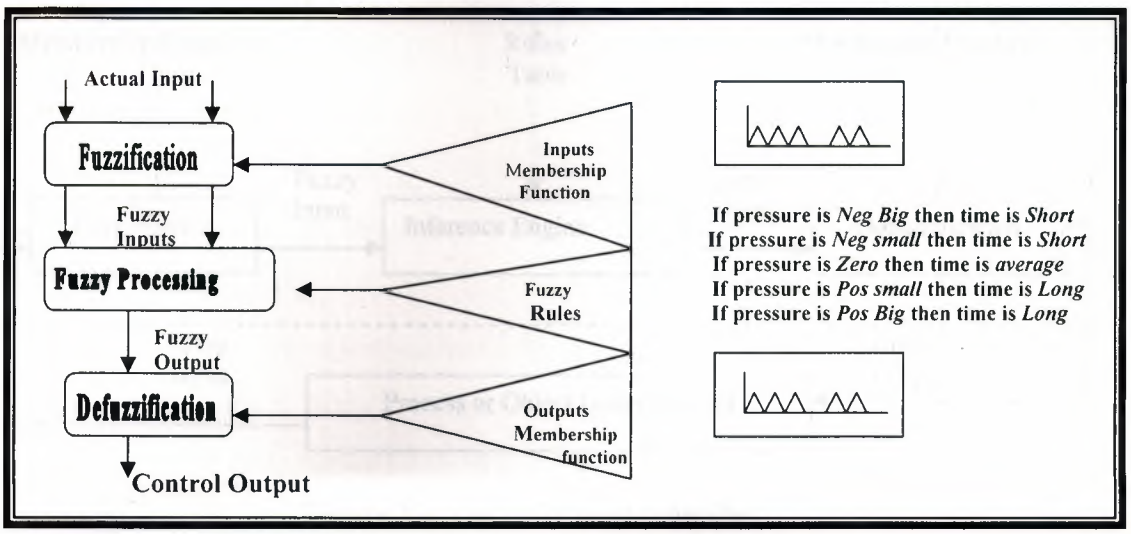


Figure 2.1. Operations of fuzzy controller.

In real control system, the controller output should be used to control a real object or process. It is important to know a crisp value for every output signal. Defuzzification produces this value on the basis of output membership functions.

Fuzzy control gives us a rather simple to use method for producing high quality controller with complicated input/output characteristics. In order to construct a fuzzy controller, it is needed just to write some rules.

The classical design scheme contains the following steps:

1. Define the input and control variable: determine which states of the process shall be observed and which control action are to be considered.
2. Define the condition interface: fix the ways in which observation of the process are expressed as fuzzy sets.
3. Design the rule base: determine which rules are to be applied under which conditions.
4. Design the computational unit: supply algorithms to perform fuzzy computations. That unit will generally lead to fuzzy outputs.
5. Determine rules according to which fuzzy control statement can be transformed into crisp control actions.

The typical structure of control system based on fuzzy controller is given in Figure 2.2.

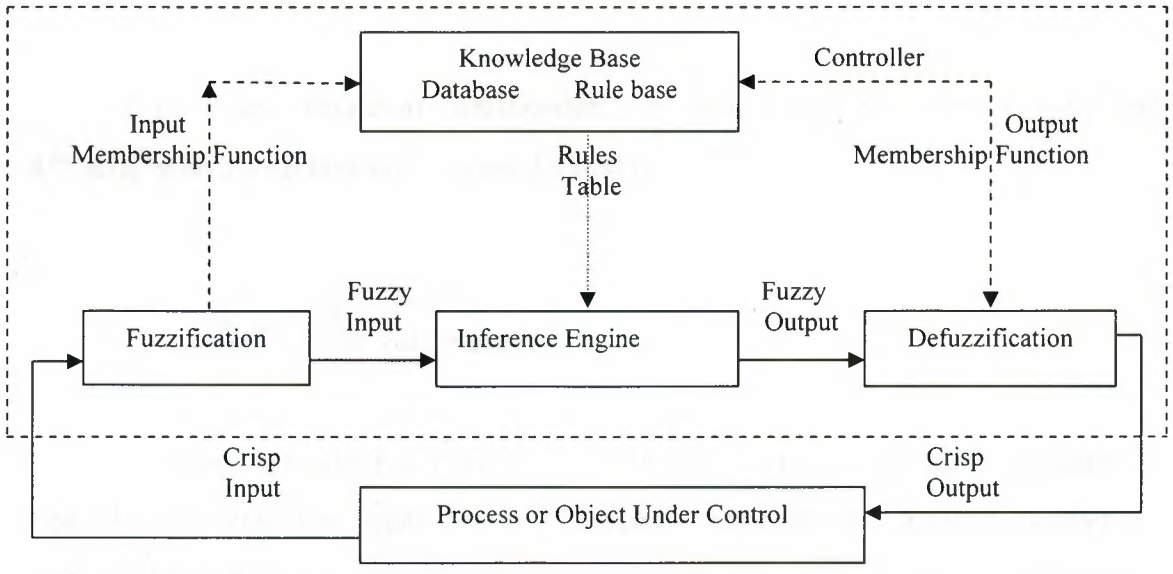


Figure 2.2. The fuzzy logic controller.

The first usual step in the design process of any controller is choosing variables that can be measured. These variables become the inputs of the controller. Step 2 represents the fuzzification process, step 4 fuzzy inference process and step 5 represents defuzzification process.

However, the heart of a fuzzy controller design is a formulation of the rules. To get these rules the main basis is an expert's experience, his/her understanding how a fuzzy controller should operate and what it should do.

2.3 Fuzzification

Fuzzy sets are used to quantify the information in the rule-base, and the inference mechanism operates on fuzzy systems to produce fuzzy sets; hence, we must specify how the fuzzy system will convert its numeric inputs $u_i \in U_i$ into fuzzy sets (a process called "fuzzification") so that they can be used by the fuzzy system.

Let U_i^*; denote the set of all possible fuzzy sets that can be defined on U_i. Given $u_i \in U_i$, fuzzification transforms u_i to a fuzzy set denoted by A_i^{fuzz} defined on the universe of discourse U_i. This transformation is produced by the fuzzification operator F defined by

$$F: U_i \Rightarrow U_i^*$$

where

$$F(u_i) = A_i^{fuzz},$$

Quite often "singleton fuzzification" is used, which produces a fuzzy set $A_i^{fuzz} \in U_i^*$ with a membership function defined by

$$\mu_{A_i^{fuz(x)}} = \begin{cases} 1 & x = u_i \\ 0 & \text{otherwise} \end{cases}$$

Any fuzzy set with this form for its membership function is called a "singleton." Basically, the singleton fuzzy set is a different representation for the number u_i. Singleton fuzzification is generally used in implementations since, without the presence of noise, we are absolutely certain that u_i takes on its measured value (and no other value), and since it provides certain savings in the computations needed to implement a

fuzzy system (relative to, for example, "Gaussian fuzzification," which would involve forming bell-shaped membership functions about input points, or triangular fuzzification, which would use triangles).

The reasons other fuzzification methods have not been used very much are they add computational complexity to the inference process, the need for them has not been that well justified.

This is partly due to the fact that very good functional capabilities can be achieved with the fuzzy system when only singleton fuzzification is used.

It is actually the case that for most fuzzy controllers, the fuzzification block in Figure 2.1 can be ignored since this process is so simple. Generally the fuzzification process is the act of obtaining a value of an input variable (e.g., e(t)) and finding the numeric values of the membership function(s) that are defined for that variable.

Some think of the membership function values as an "encoding" of the fuzzy controller numeric input values. The encoded information is then used in the fuzzy inference process that starts with "matching."

2.4 Linguistic Variables

The research lately have shown that conventional analysis methods for systems analysis and computer modeling, based on precise processing of numerical data, are not capable of dealing with huge complexity of real technological processes. This leads to the fact that in order to get decisions affecting the behavior of those processes we need to reject of traditional requirements to measurement accuracy, which are necessary for mathematical analysis of precisely defined mechanical systems.

The necessity to sacrifice the precision and determinate is dictated also by the appearance of some classes of control problems that are connected with decision making by operator in the “man-computer” interface. Implementation of the dialog in such interface is impossible without application of languages close to natural ones and capable of describing fuzzy categories near to human notions and imaginations. In this connection, it is valuable to use the notion of linguistic variable first introduced by L.Zadeh[5]. Such linguistic variables allow an adequate reflection of approximate in-word descriptions of objects and phenomena in the case if there is no any precise deterministic description. It should note as well that many fuzzy categories described linguistically even appear to be more informative than precise descriptions.

To specify rules for the rule base, the expert will use a linguistic description; hence, linguistic expressions are needed for the inputs and the outputs and the characteristics of the inputs and outputs. Here the linguistic variables (constant symbolic descriptions of what are in general time-varying quantities) will be used to describe fuzzy system inputs and outputs. In the fuzzy system that described in Figure 1.1 a linguistic input variables are denoted by u_i. Similarly, linguistic output variables are denoted by y_i. For instance, an input to the fuzzy system may be described as u_1= "position error" or u_2="velocity error," and an output from the fuzzy system may be y_1="voltage in."

2.4.1 Linguistic Values

Linguistic variables u_i and y_i take on linguistic values that are used to describe characteristics of the variables. Let A_i^j denote the j^{th} linguistic value of the linguistic variable u_i. Defined over universe of discourse U_i. If we assume that there exist many linguistic values defined over U_i, then the linguistic variable u_i takes on the elements from the set of linguistic values denoted by

$$A_i = \{A_i^j : j = 1,2,.....,N\}$$

(sometimes for convenience we will let the j indices take on negative integer values). Similarly, let B_i^j denote the j^{th} linguistic variable y_i defined over the universe of discourse Y_i. The linguistic variable y_i takes on elements from the set of linguistic values denoted by

$$B_i = \{B_i^p : p = 1,2,.....,M_i\}$$

(sometimes for convenience we will let the p indices take on negative integer values). Linguistic values are generally descriptive terms such as "positive large", "zero" and "negative big". For example, assume that u_1 denotes the linguistic variable "speed", then it is possible to assign A_1^1 = "slow", A_1^2 ="medium", A_1^3 ="fast" so that u_1 has a value from A_1 = { A_1^1, A_1^2, A_1^3 }.

Another important aspect of the notion of linguistic variable that a linguistic variable is associated with the two rules: the syntactic rule, which can be set as a grammar, generating names for the variable; and the semantic rule, which determines an algorithmic procedure for calculating the meaning of each value. Thus these rules make the essential part of the description of the structure of linguistic variable.

Definition . A linguistic variable is characterized by the set *(u, T, X, G, M)*, where u is the name of variable; *T* denotes the term-set of u that refer to a base variable whose values range over a universem *X; G* is a syntactic rule (usually in form of a grammar) generating linguistic terms; *M* is a semantic rule that assigns to each linguistic term its meaning, which is a fuzzy set on *X*.

A certain $t \in T$ generated by the syntactic rule *G* is called *A* term. *A* term consisting of one or more words, the words being always used together, is named an atomary term. *A* term consists of several atomary terms is named a composite term. The concatenation of some components of a composite term (i.e. the result of linking the chains of components of the composite term) is called a subterm. Here t_1, t_2, ... are terms in

$$T= t_1 + t_2 +...$$

The meaning of *M(t)* of the term *t* is defined as a restriction *R(t; x)* on the basis variable x conditioned by the fuzzy variable X:

$$M(t) \equiv R(t; x),$$

It is assumed here that *R(t; x)* and, consequently, *M(t)* can be considered as a fuzzy subset of the set *X* named as *t*.

The assignment equation in case of linguistic variable takes the form in which *t*-term in *T* are name generated by the grammar *G*, where the meaning assigned to the term *t* is expressed by the equality

$$M(t)=R(term\ in\ T)$$

In other words the meaning of the term *t* is found by the application of the semantic rule *M* to the value of term t assigned according to the right part of equation.

Moreover, it follows that *M(t)* is identical to the restriction associated with the term *t*.

It should be noted that the number of elements in *T* can be unlimited and then for both generating elements of the set *T* and for calculating their meaning, the application of the algorithm, not simply the procedure for watching term-set, is necessary.

We will say that a linguistic variable *u* is structured if its term-set *T* and the function *M*, which maps each element from the term-set into its meaning, can be given by means of algorithm. Then both syntactic and semantic rules connected with the structured linguistic variable can be considered algorithmic procedures for generating elements of the set *T* and calculating the meaning of each term in *T*, respectively.

However in practice we often encounter term-sets consisting of a small number of terms. This makes it easier to list the elements of term-set *T* and establishes a direct mapping from each element to its meaning.

2.5 Rule Base

The mapping of the inputs to the outputs for a fuzzy system is in part characterized by a set of condition → action rules, or in *modus ponens* (If-Then) form,

If premise **Then** consequent (2.1)

Usually, the inputs of the fuzzy systems are associated with the premise, and the outputs are associated with the consequence. These If-Then rules can be represented in many forms. Two standard forms, multi-input multi-output (MIMO) and multi-input single-output (MISO), are considered here. The MISO form of a linguistic rule is

If u_1 is A_1^j and u_2 is A_2^k and ,....., and u_n is A_n^l Then y_q is B_q^p (2.2)

It is an entire set of linguistic rules of this form that the expert specifies on how to control the system. Note that if u_1="velocity error" and A_1^j = "positive large", then "u_1 is A_1^j", a single term in the premise of the rule, means "velocity error is positive large". It can be easily shown that the MIMO form for a rule (i.e. one with consequents that have terms MISO rules using simple rules from logic. For instance, the MIMO rule with *n* inputs and *m* =2 outputs

If u_1 is A_1^j and u_2 is A_2^k and ,....., and u_n is A_n^l Then y_1 is B_1^r and y_2 is B_2^s

Is linguistically (logically) equivalent to the two rules

If u_1 is A_1^j and u_2 is A_2^k and ,....., and u_n is A_n^l Then y_1 is B_1^r

If u_1 is A_1^j and u_2 is A_2^k and ,....., and u_n is A_n^l Then y_2 is B_2^s

This is the case since the logical "and" in the consequent of the MIMO rule is still represented in the two MISO rules since it still assert that the both the first "and" second rule are valid. For implementation, then two fuzzy systems should be specified, one with output y_1 and the other with the output y_2. The logical "and" in the consequent of the MIMO rule is still represented in the MISO case since by implementation two fuzzy systems asserting that the ones set of rules is true "and" another it true.

Assume that there are a total of *R* rules in the rule base numbered 1,2,.....,*R*, and we naturally assume that the rules in the rule base are distinct (i.e. there are no two rules with exactly the same premises and consequent); however, this does not general need to be the case. For simplicity let use tuples to denote the i^{th} MISO rule of the form given in equation (2.2). any of the terms associated with any of the inputs for any MISO rule

$$(j, k, \ldots, l; p, q)I$$

can be included or omitted. For instance, suppose a fuzzy system has two inputs and one output with u_1="position", u_2="velocity", and y_1="force". Moreover, suppose each input is characterized by two linguistic values A_i^1="small" and A_i^2="large" for i=1,2. suppose further that the output is characterized by two linguistic values B_1^1="negative" and B_1^2="positive". A valid If-Then rule could be

If Position is large **Then** force is positive

Even though it does not follow the format of a MISO rule given above. In this case, one premise term (linguistic variable) has been omitted from the If-Then rule. It is clearly seen that it is allows for the case where the expert does not use all the linguistic terms (and hence the fuzzy sets that characterize them) to state some rules.

Finally, note that if all premise terms are used in every rule and a rule is formed for each possible combination of premise elements, then there are

$$\prod_{i=1}^{n} N_i = N_1.N_2. \quad .... \quad .N_n$$

rules in the rule base. For example, if n=2 inputs and we have N_i=11 membership functions on each universe of discourse, then there are 11*11=121 possible rules. Clearly, in this case the number of rules increases exponentially with an increase in the number of fuzzy controller inputs or membership functions.

The rules may use several variables both in the premise and the consequent of the rules. The controllers can therefore be applied to both multi-input multi-output (MIMO) problems and single-input single-output (SISO) problems. The typical S1S0 problem is to regulate a control signal based on an error signal. The controller may actually need both the error, the change in error, and the accumulated error as inputs, but we will call it single-loop control, because in principle all of the three are formed from the error measurement. To simplify, this section assumes that the control objective is to regulate some process output around a prescribed set-point or reference. The presentation is thus limited to single-loop control.

2.5.1 Rule formats

Basically a linguistic controller contains rules in the If-Then format, but they can be presented in different formats. In many systems, the rules are presented to the end-user in a format similar to the one below,

1. If error is **Neg** and change in error is **Neg** then output is **NB**
2. If error is **Neg** and change in error is **Zero** then output is **NM**
3. If error is **Neg** and change in error is **Pos** then output is **Zero**
4. If error is **Zero** and change in error is **Neg** then output is **NM**
5. If error is **Zero** and change in error is **Zero** then output is **Zero**
6. If error is **Zero** and change in error is **Pos** then output is **PM**
7. If error is **Pos** and change in error is **Neg** then output is **Zero**
8. If error is **Pos** and change in error is **Zero** then output is **PM**
9. If error is **Pos** and change in error is **Pos** then output is **PB**

The names Zero, Pos, Neg are labels of fuzzy sets as well as NB, NM, PB and PM (negative big, negative medium, positive big, and positive medium respectively). The same set of rules could be presented in a relational format, a more compact representation.

Consider Table 2.1, The top row is the heading, with the names of the variables. It is understood that the two leftmost columns are inputs, the rightmost is the output, and each row represents a rule. This format is perhaps better suited for an experienced user who wants to get an overview of the rule base quickly. The relational format is certainly suited for storing in a relational database. It should be emphasized that the relational format implicitly assumes that the connective between the inputs is always logical AND — or logical OR for that matter as long as it is the same operation for all rules — and not a mixture of connectives.

Table 2.1. Relation between input and output variables.

Error	Change in Error	Output
Neg	Pos	Zero
Neg	Zero	NM
Neg	Neg	NB
Zero	Pos	PM
Zero	Zero	Zero
Zero	Neg	NM
Pos	Pos	PB
Pos	Zero	PM
Pos	Neg	Zero

Incidentally, a fuzzy rule with an or combination of terms can be converted into an equivalent and combination of terms using laws of logic (DeMorgan's laws among others). A third format is the tabular linguistic format.

Consider Table 2.2, this is even more compact. The input variables are laid out along the axes, and the output variable is inside the table. In case the table has an empty cell, it is an indication of a missing rule, and this format is useful for checking completeness. When the input variables are error and change in error, as they are here,

that format is also called a linguistic phase plane. in case there are n > 2 input variables involved, the table grows to an n-dimensional array; rather user-unfriendly.

To accommodate several outputs, a nested arrangement is conceivable. A rule with several outputs could also be broken down into several rules with one output.

Lastly, a graphical format which shows the fuzzy membership curves is also possible. This graphical user-interface can display the inference process better than the other formats, but takes more space on a monitor.

Table 2.2. Complete description of relation between input and output variables.

		Change In Error		
		Neg	**Zero**	**Pos**
Error	**Neg**	NB	NM	Zero
	Zero	NM	Zero	PM
	Pos	Zero	PM	PB

2.5.2 Connectives

In mathematics, sentences are connected with the words and, or, if- then (or implies), and if and only if, or modifications with the word not. These five are called connectives. It also makes a difference how the connectives are implemented. The most prominent is probably multiplication for fuzzy and instead of minimum. So far most of the examples have only contained and operations, but a rule like "If error is very Neg and not Zero or change in error is Zero then …'' is also possible.

The connectives "and" and "or" are always defined in pairs, for example,

a *and* b = min (a. b) minimum

a *or* b = max (a. b) maximum

or

a *and* b= a * b algebraic product

a *or* b = a + b - a * b algebraic or probabilistic sum

There are other examples (e.g., Zimmermann. 1991, 31 32), but they are more complex.

2.5.3 Modifiers

A linguistic modjfier, is an operation that modifies the meaning of a term. For example, in the sentence "very close to 0". the word very modifies Close to 0 which is a fuzzy set. A modifier is thus an operation on a fuzzy set. The modifier very can be defined as squaring the subsequent membership function, that is

$$very\ a = a^2 \tag{2.3}$$

Some examples of other modifiers are

$$extremely\ a = a^3$$

$$slightly\ a = a^{1/3}$$

$$somewhat\ a = more or less\ a\ and\ not\ slightly\ a$$

A whole family of modifiers is generated by a^p where p is any power between zero and infinity With $p = \infty$ the modifier could be named exactly, because it would suppress all memberships lower than 1.0.

2.5.4 Universes

Elements of a fuzzy set are taken from a universe old discourse (aorist universe). The universe contains all elements that can come into consideration. Before designing the membership functions it is necessary to consider the universes for the inputs and outputs. Take for example the rule

If error is ***Neg*** *and change in error is* ***Pos*** *then output is* ***Z***

Naturally, the membership functions for *Neg* and *Pos* must be defined for all possible values of *error* and *change in error,* and a standard universe may be convenient.

Another consideration is whether the input membership functions should be continuous or discrete. A continuous membership function is defined on a continuous universe by means of parameters. A discrete membership function is defined in terms of a vector with a finite number of elements. In the latter case it is necessary to specify the range of the universe and the value at each point. The choice between fine and coarse resolution is a trade off between accuracy, speed and space demands. The quantiser takes time to execute, and if this time is too precious, continuous membership functions will make the quantiser absolute.

2.5.5 Membership Functions

Every element in the universe of discourse is a member of a fuzzy set to some grade, maybe even zero. The grade of membership for all its members describes a fuzzy set, such as Neg. In fuzzy sets elements are assigned a *grade of membership*, such that the transition from membership to non-membership is gradual rather than abrupt. The set of elements that have a non-zero membership is called the *support* of the fuzzy set. The function that ties a number to each element x of the universe is called the *membership function* (μ (x)).

The designer is inevitably faced with the question of how to build the term sets. Then two specific questions should be considered:
(i) How does one determine the shape of the sets? and (ii) How many sets are necessary and sufficient? For example, the error in the position controller uses the family of terms Neg, Zero, and Pos. According to fuzzy set theory the choice of the shape and width is subjective, but a few rules of thumb apply.

- A term set should be sufficiently wide to allow for noise in the measurement.
- A certain amount of overlap is desirable; otherwise the controller may run into poorly defined states, where it does not return a well defined output.

A preliminary answer to questions (i) and (ii) is that the necessary and sufficient number of sets in a family depends on the width of the sets, and vice versa. A solution could be to ask the process operators to enter their personal preferences for the membership curves; but operators also find it difficult to settle on particular curves.

- *Start with triangular sets*. All membership functions for a particular input or output should be symmetrical triangles of the same width. The leftmost and the rightmost should be shouldered ramps.
- *The overlap should be at least 50%.* The widths should initially be chosen so that each value of the universe is a member of at least two sets, except possibly for elements at the extreme ends. If, on the other hand, there is a gap between two sets no rules fire for values in the gap. Consequently the controller function is not defined.

  Membership function can be flat on the top, piece-wise linear and triangle shaped, rectangular, or ramps with horizontal shoulders.

Figure 2.3 shows some typical shapes of membership functions.

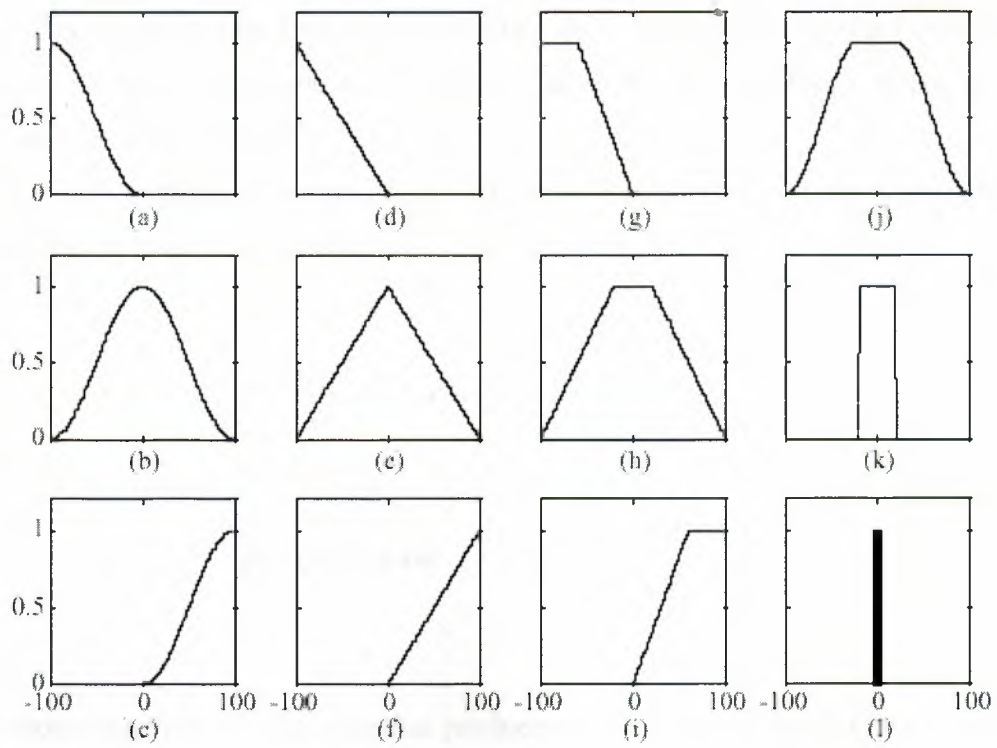


Figure 2.3. Examples of membership functions. Read from top to bottom, left to right: (a) s— function, (b) π— function, (c) z—function, (d-f) triangular versions, (g-i) trapezoidal versions, (j) flat π— function. (k) rectangle. (I) singleton.

Strictly speaking, a fuzzy set A is a collection of ordered pairs

$$A=\{(x, \mu(x))\} \tag{2.4}$$

Item x belongs to the universe and $\mu(x)$ is its grade of membership in A. A single pair $(x, \mu(x))$ is a fuzzy *singleton*; *singleton output* means replacing the fuzzy sets in the conclusion by numbers (scalars). For example

1. If error is Pos then output is 10 volts
2. If error is Zero then output is 0 volts
3. If error is Neg then output is -10 volts

There are at least three advantages to this:

- The computations are simpler;
- It is possible to drive the control signal to its extreme values; and

- It may actually be a more intuitive way to write rules.

The scalar can be a fuzzy set with the singleton placed in a proper position. For example 10 volts, would be equivalent to the fuzzy set (0,0,0,0,1) defined on the universe (-10,-5,0,5,10) volts.

Fuzzy controllers use a variety of membership functions. Membership function for the triangle form is calculated as

$$\mu(x) = \begin{cases} 1 - \dfrac{\bar{x} - x}{\alpha}, \bar{x} - \alpha \le x \le \bar{x} \\ 1 - \dfrac{\bar{x} - x}{\beta}, \bar{x} \le x \le \bar{x} + \beta \\ 0, \text{in other case} \end{cases} \tag{2.5}$$

A common example of a function that produces a bell curve is based on the exponential function,

$$\mu(x) = \exp\left[\frac{-(x - x_0)^2}{2\sigma^2}\right] \tag{2.6}$$

This is a standard Gaussian curve with a maximum value of 1, x is the independent variable on the universe, x is the position of the peak relative to the universe, and σ is the standard deviation. Another definition which does not use the exponential is

$$\mu(x) = \left[1 + \left(\frac{x - x_0}{\sigma}\right)^2\right]^{-1} \tag{2.7}$$

The FL Smidth controller uses the equation

$$\mu(x) = 1\text{-}\exp\left[-\left(\frac{\sigma}{x - x_0}\right)^a\right] \tag{2.8}$$

The extra parameter a controls the gradient of the sloping sides. It is also possible to use other functions, for example the sigmoid known from neural networks.

A cosine function can be used to generate a variety of membership functions. The s-curve can be implemented as

$$s(x_l, x_r, x) = \begin{cases} 0 & x \prec x_l \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x - x_r}{x_r - x_l}\right) & x_l \le x \le x_r \\ 1 & x \succ x_r \end{cases} \qquad (2.9)$$

where x_l is the left breakpoint, and x_r, is the right breakpoint. The z-curve is just a reflection,

$$z(x_l, x_r, x) = \begin{cases} 1 & x \prec x_l \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x - x_l}{x_r - x_l} * \pi\right) & x_l \le x \le x_r \\ 0 & x \succ x_r \end{cases} \qquad (2.10)$$

Then the π-curve can be implemented as a combination of the s-curve and the z-curve, such that the peak is fiat over the interval $[x_2, x_3]$

$$\pi(x_1, x_2, x_3, x_4, x) = \min((x_1, x_2, x), z(x_3, x_4, x)) \qquad (2.11)$$

2.6 Inference Mechanism

The inference mechanism has two basic tasks:

I. Determining the extent to which each rule is relevant to the current situation as characterized by the inputs u_i, i = 1, 2, ..., n (this task called "matching");

II. Drawing conclusions using the current inputs u_i and the information in the rule-base (we call this task an "inference step").

For matching note that $A_1^j \times A_2^k \times \ldots \times A_n^l$ is the fuzzy set representing the premise of the i^{th} rule (j, k, . . . , l; p, q); (there may be more than one such rule with this premise).

Suppose that at some time we get inputs u_i, $i = 1, 2, \ldots, n$, and fuzzification produces

$$A_1^{fuz}\ ,\ A_2^{fuz}\ ,\ \ldots\ldots\ ,\ A_n^{fuz}$$

the fuzzy sets representing the inputs. There are then two basic steps to matching.

Step 1: Combine Inputs with Rule Premises: The first step in matching involves finding fuzzy sets A_1^j, A_2^k, . . . , A_n^l, with membership functions

$$\mu\ A_1^j\ (u_1) = \mu\ A_1^j\ (u_1) * \mu\ A_1^{fuz}\ (u_1)$$
$$\mu\ A_2^k\ (u_2) = \mu\ A_2^k\ (u_2) * \mu\ A_2^{fuz}\ (u_2)$$
$$\vdots$$
$$\mu\ A_n^l\ (u_n) = \mu\ A_n^l\ (u_n) * \mu\ A_n^{fuz}\ (u_n)$$

(for all j, k, . . . , l) that combine the fuzzy sets from fuzzification with the fuzzy sets used in each of the terms in the premises of the rules. If singleton fuzzification is used, then each of these fuzzy sets is a singleton that is scaled by the premise membership function (e.g. $\mu\ A_1^{fuz}\ (u_1) = \mu\ A_1^j\ (u_1)$). That is, with singleton fuzzification we have $\mu\ A_1^{fuz} = 1$, for all i = l, 2, ..., n for the given u_i inputs so that

$$\mu\ A_1^{fuz}\ (u_1) = \mu\ A_1^j\ (u_1)$$
$$\mu\ A_2^{fuz}\ (u_2) = \mu\ A_2^k\ (u_2)$$
$$\vdots$$
$$\mu\ A_n^{fuz}\ (u_n) = \mu\ A_n^i\ (u_n)$$

We see that when singleton fuzzification is used, combining the fuzzy sets that were created by the fuzzification process to represent the inputs with the premise membership functions for the rules is particularly simple. It simply reduces to computing the membership values of the input fuzzy sets for the given inputs $u_1, u_2, \ldots, u_n$.

Step 2: Determine Which Rules Are On: In the second step, we form membership values $\mu_i(u_1, u_2, \ldots, u_n)$ for the i^{th} rule's premise that represent the certainty that each rule premise holds for the given inputs. Define

$$\mu_i = (u_1, u_2, \ldots, u_n) = \mu A_1^j (u_1) * \mu A_2^k (u_2) * \mu A_n^i (u_n)$$

which is simply a function of the inputs u_i

We use to represent the certainty that the premise of rule (i) matches the input information when we use singleton fuzzification. This $\mu_i(u_1, u_2, \ldots, u_n)$ is simply a multidimensional certainty surface. It represents the certainty of a premise of a rule and thereby represents the degree to which a particular rule holds for a given set of inputs.

Finally, we would remark that sometimes an additional "rule certainty" is multiplied by μ_i. Such a certainty could represent our a priori confidence in each rule's applicability and would normally be a number between zero and one. If for rule (i) its certainty is (0.1), we are not very confident in the knowledge that it represents; while if for some rule (j) we let its certainty be (0.99), we are quite certain that the knowledge it represents is true.

This concludes the process of matching input information with the premises of the rules.

There are two standard alternatives to performing the inference step, one that involves the use of implied fuzzy set and the other that uses the overall implied fuzzy set.

Alternative 1: Determine Implied Fuzzy Sets: Next, the inference step is taken by computing, for the i^{th} rule (j, k, . . . , l; p,q)$_i$, the "implied fuzzy set" Bq with membership function

$$\mu B_q^i(y_q) = \mu_i(u_1, u_2, \ldots, u_n) * \mu B_q^i(y_q) \qquad (2.12)$$

The implied fuzzy set (μB_q^1) specifies the certainty level that the output should be a specific crisp output y_q within the universe of discourse y^q, taking into consideration only rule (i). Note that since $\mu_i(u_1, u_2, \ldots, u_n)$ will vary with time, so will the shape of the membership functions $\mu B_q^i(y_q)$ for each rule.

Alternative 2: Determine the Overall Implied Fuzzy Set: Alternatively, the inference mechanism could, in addition, compute the "overall implied fuzzy set" B_q with membership function

$$\mu B_q(y_q) = \mu B_q^1(y_q) \oplus \mu B_q^2(y_q) \oplus \ldots \oplus \mu B_q^R(y_q) \qquad (2.13)$$

that represents the conclusion reached considering all the rules in the rule-base at the same time (notice that determining B_q can, in general, require significant computational resources).

Instead, COG or centeraverage defuzzification method performed the aggregation of the conclusions of all the rules that are represented by the implied fuzzy sets.

Using the mathematical terminology of fuzzy sets, the computation of $\mu B_q(y_q)$ is said to be produced by a "sup-star compositional rule of inference". The "sup" in this terminology corresponds to the $\oplus$ operation, and the "star" corresponds to *. "Zadeh's compositional rule of inference" is the special case of the sup-star compositional rule of inference when maximum is used for $\oplus$ and minimum is used for (*). The overall justification for using the above operations to represent the inference step lies in the fact that we can be no more certain about our conclusions than we are about our premises.

The operations performed in taking an inference step added here to this principle. To see this, we should study equation (2.5) and note that the scaling from $\mu_i(u_1, u_2, \ldots, u_n)$ that is produced by the premise matching process will always ensure that $\sup_{y_q} \{\mu B_q(y_q)\} \leq \mu_i(u_1, u_2, \ldots, u_n)$. The fact that we are no more certain of our consequents than our premises is shows that the heights of the implied fuzzy sets are always less than the certainty values for all the premise terms.

2.7 Defuzzification

The resulting fuzzy set must be converted to a number that can be sent to the process as a control signal. This operation is called *defuzzification*. The resulting fuzzy set is thus defuzzified into a crisp control signal. There are several defuzzification methods.

2.7.1 Centre Of Gravity (COG)

The crisp output value u is the abscissa under the centre of gravity of the fuzzy set,

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \tag{2.14}$$

Here x_i is a running point in a discrete universe, and $\mu(x_i)$ is its membership value in the membership function. The expression can be interpreted as the weighted average of the elements in the support set. For the continuous case, replace the summations by integrals. It is a much used method although its computational complexity is relatively high. This method is also called *centroid of area.*

2.7.2 Center Of Gravity Method for Singletons (COGS)

If the membership functions of the conclusions are singletons (Figure 2.4), the output value is

$$u = \frac{\sum_i \mu(s_i) s_i}{\sum_i \mu(s_i)} \tag{2.15}$$

Here s_i is the position of singleton i in the universe. and $\mu(s_i)$ is equal to the firing strength α_i of rule i. This method has a relatively good computational complexity and u is differentiable with respect to the singletons s_i, which is useful in neuro-fuzzy systems.

2.7.3 Bisector Of Area (BOA)

This method picks the abscissa of the vertical line that divides the area under the curve in two equal halves. In the continuous case,

$$u = \left\{ x \left| \int_{Min}^{x} \mu(x)dx = \int_{x}^{Max} \mu(x)dx \right. \right\} \tag{2.16}$$

Here x is the running point in the universe, $\mu(x)$ is its membership.

Min is the leftmost value of the universe, and *Max* is the rightmost value. Its computational complexity is relatively high, and it can be ambiguous. For example, if the fuzzy set consists of two singletons any point between the two would divide the area in two halves; consequently it is safer to say that in the discrete case, BOA is not defined.

2.7.4 Center Of Average

A crisp output y_q^{Crisp} is chosen using the centers of each of the output membership functions and the maximum certainty of each of the conclusions represented with the implied fuzzy sets, and is given by

$$y_q^{Crisp} = \frac{\sum_{i=1}^{R} b_i^q \sup_{yq}\{\mu B_q^i(y_q)\}}{\sum_{i=1}^{R} \sup_{yq}\{\mu B_q^i(y_q)\}}$$

where "sup" denotes the "supermum" (i.e., the least upper bound which can often be thought of as maximum value). Hence, $\sup_x\{\mu(x)\}$ can be simply thought as the highest value of $\mu(x)$.

2.7.5 Max Criterion

A crisp output y_q^{Crisp} is chosen as the point on the output universe of discourse y_q for which the overall implied fuzzy set B_q achieves a maximum-that is,

$$y_q^{Crisp} \in \left\{ \arg\sup_{y_q} \{\mu B_q(y_q)\} \right\}$$

Here, "$\arg\sup_x\{\mu(x)\}$" returns the value of x that results in the supermum of the function $\mu(x)$ being achieved. For example, suppose that $\mu_{Overall}(u)$ denotes the membership function for the overall implied fuzzy set that is obtained by taking the maximum of the certainty values of $\mu(1)$ and $\mu(2)$ over all u.

2.7.6 Mean Of Maxima (MOM)

An intuitive approach is to choose the point with the strongest possibility i.e. maximal membership. It may happen, though, that several such points exist, and a common practice is to take the *mean of maxima* (MOM). This method disregards the shape of the fuzzy set, but the computational complexity is relatively good.

2.7.7 Leftmost (LM), and Rightmost Maxima (RM)

Another possibility is to choose Leftmost Maxima (LM), or Rightmost Maxima (RM). In the case of the robot, for instance, it must choose between left and right to avoid an obstacle in front of it.

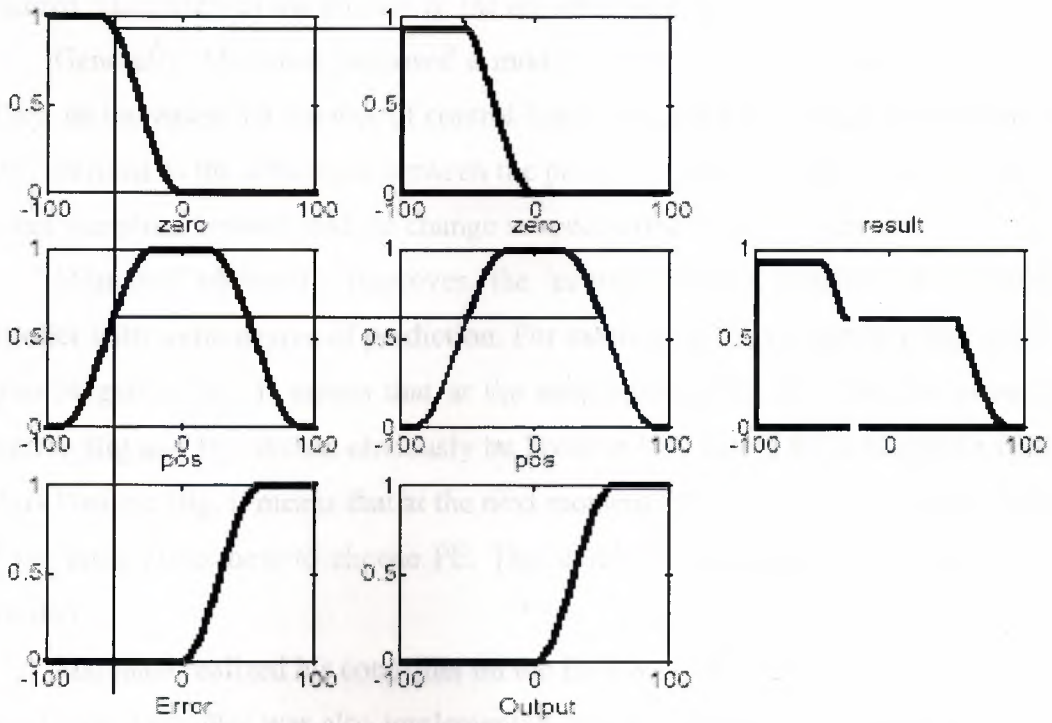


Figure 2.4. One input, one output rule base with non-singleton output sets.

The defuzzifier must then choose one or the other, not something in between. These methods are indifferent to the shape of the fuzzy set, but the computational complexity is relatively small.

2.8 Mamdani-Type Fuzzy Processing

Mamdani[19] proposed to control the plant by realizing some fuzzy rules or fuzzy conditional statements, for example:

If pressure error (PE) is Negative Big (NB)
Then heat change (HC) is Positive Big (PB)

So the outputs of a plant can easily be measured and control action can be calculated according to this rules Table.

The pressure error is the difference between the current value of the pressure and the set point. And the speed error (SE) is again the difference between the current speed and the set point.

A set point is usually a desired value for the plant output: the values which some measured parameters of the process or the object should have at a particular time.

Generally, Mamdani proposed a modification. In order to improve the control quality, he increased the number of control inputs and used the change in pressure error (CPE), defined as the difference between the present PE and the last one (corresponding to a last sampling instant), and the change in speed error (CSE) as well.

Mamdani obviously improves the control quality, because it provides a controller with some degree of prediction. For example, if PE is Negative Big and CBE is also Negative Big, it means that at the next moment PE will become even more Negative Big and HC should obviously be Positive Big. But if PE is Negative Big and CPE is Positive Big, it means that at the next moment PE will become Negative Smaller and we think about how to choose PE. This condition increases the sensitivity of the controller.

Mamdani realized his controller on the PDP-8 computer. It contained 24 rules. A fixed digital controller was also implemented on the computer and applied to the same plant for a comparison. For the fixed controller, many runs were required to tune the controller for the best performance. This tuning was done by trial-and-error process.

The quality of the fuzzy controller was found to be better than the best result of the fixed controller each time, so opening a new era in a controller design.

Mamdani used fuzzy theory to calculate the output according to the rules set and gained a solid theoretical base. It means that we can use this result to construct other fuzzy controllers. Some of these controllers, illustrating different possible applications, are given in Table 2.3

To process these rules the process is called the inference mechanism or the inference engine. Let us write Mamdani rules in a general case. Mathematically a rule will look like

$$R^i: \quad \text{IF} \quad A_{i1}(x_1), A_{i2}(x_2), \ldots\ldots, A_{im}(x_m) \quad \text{THEN Y is } B_i .$$

Here x_1, x_2, .. x_m stand for input variables, for example, pressure, temperature, error, etc., $A_{ij}(xj)(j = 1,2,\ldots.m)$ is a fuzzy set on X_j, Y is an output variable, B_i is a fuzzy set on Y.

So in the rule:

if pressure error (PE) is Negative Big (NB) then heat change (HC) is Positive Big (PB) pressure error is X_1 heat change is Y, $A_{i1}(x_1)$ is Negative Big, and B_i is Positive Big.

Table 2.3 Fuzzy controller applications designed using Mamdani logic.

Date	Application	Designers
1975	Laboratory steam engine	Mamdani and Assilian
1976	Warm water plant	Kikert and Van Nauta Lemke
1977	Ship course control	Van Amerongen
1978	Rolling steel mill	Tong
1979	Iron ore sinter plant	Rutherford
1980	Cement kiln control	Umbers and king
1985	Aircraft landing control	Larkin
1989	Autonomous guided vehicle	Harris et al.
1991	Ship yaw control	Sutton and Jess

2.9 Sugeno-Type Fuzzy Processing

We have considered how fuzzy processing was realized by Mamdani, and the method is called after him. Another method was proposed by Sugeno, who changed a part of the rules. In his method, the consequence part is just a mathematical function of the input variables. The format of the method is:

if $A_1(x_1)$, $A_2(x_2)$, … , $A_n(x_n)$ then $Y=f(x_1, x_2, \ldots, x_n)$.

You see that the antecedent part is similar to the Mamdani method. The function *f* in a consequence is usually a simple mathematical function, linear or quadratic:

$$f = a_0 + a_1 * x_1 + a_2 * x_2 + \ldots + a_n * x_n$$

The antecedent part in this case is processed in exactly the same way as the Mamdani method, and then an obtained degree of applicability is assigned to the value of Y calculated as the function of real inputs.
Let us again consider the rule:

If pressure is NegBig and temperature is High then time is Short

But now replace it with:

If pressure is NegBig and temperature is High then time is 0.3 x pressure + 0.5 x temperature.

Suppose 0.3 and 0.5 are factors expressing how the necessary time depends on pressure and temperature. Then again for crisp inputs of –22kPa for a pressure error and 22^0C for temperature we calculate the applicability degree of the rule, which is 0.6. Now we calculate the value for Y as a real function of crisp inputs. The result is 0.3 x (-22) + 0.5 x 22 = 4.4. And the membership degree obtained earlier is assigned to this result. So the output of the inference process will be (4.4, 0.6) where 4.4 is a real result and 0.6 is its membership degree. This is the result of the application of one rule. The final result will be obtained after applying all the rules. Then one will have a fuzzy set as a result. Once again if any element of the universe has two or more different

membership degrees as a result of different rules processing, one can choose the maximum value or apply another s-norm calculation method.

Finally there are some questions presenting, which method is the best? When to apply a Mamdani or Sugeno controller?

Mamdani fuzzy controller (each rule output is described by a membership function) is good for capturing the expertise of a human operator. But it is awkward to design if you have the plant model but don't have a working controller (for instance, a human operator). Sugeno fuzzy controller (each rule's output is linear equation) is good for embedding linear controller and continuous switching between these output equations.

This becomes very effective when the plant model is known. Also an adaptive capability and mathematical tractability make this type of fuzzy controller a primary choice for nonlinear and/or adaptive control design that is subject to rigorous analysis.

To summarize our discussion about fuzzy inference, we should say that procedure can be roughly divided into three steps:

- Calculation of the degree of applicability of the antecedent (condition) part of each control rule.
- Calculation of the inference result (solution fuzzy set) for each control rule.
- Synthesis of the solution set for each control rule into the fuzzy output set.

2.10 Summary

The fuzzy controller operations can be divided into three processes:

- Fuzzification: converts the crisp actual inputs into a fuzzy set that can be use by the inference mechanism.
- Fuzzy Processing: processing fuzzy inputs according to the rules set and producing fuzzy output.
- Defuzzification: producing a crisp real value for fuzzy output.

The necessity to sacrifice the precision and determinate is dictated also by the appearance of some classes of control problems that are connected with decision-making by operator in the "man-computer" interface. Implementation of the dialog in such interface is impossible without application of languages close to natural ones and

capable of describing fuzzy categories near to human notions and imaginations. In this connection, it is valuable to use the notion of linguistic variable first introduced by L.Zadeh[5]. Such linguistic variables allow an adequate reflection of approximate in-word descriptions of objects and phenomena in the case if there is no any precise deterministic description. It should note as well that many fuzzy categories described linguistically even appear to be more informative than precise descriptions.

Rules in the rule base can be formed in different formats:

- If-Then Format: a set of condition → action rules.
- Relational Format: this format is perhaps better for an experienced user who wants to get an overview of the rule base quickly.
- Tabular Linguistic Format: the input variable are laid out along the axes, and the output variable is inside the table.
- Graphical Format: the graphical user-interface can display the inference process better than other formats, but takes more space on a monitor.

In chapter Three the development of PD, PI and PID-Like fuzzy controllers are described.

CHAPTER THREE

DEVELOPMENT OF FUZZY CONTROLLERS

3.1 Overview

Fuzzy systems provide a rich and meaningful addition to standard logic. In the fuzzy control design methodology, we are concerned to write down a set of rules on how to control the process, then we incorporate these rules into fuzzy controller that emulates the decision-making process. These rules are taken from a linguistic tables of the PD, PI and PID-like fuzzy controllers. In fact, there are slight changes between these tables, that means even small deviation errors will be lowed by large control signals. However, usually it is needed to make the control surface smoother in the vicinity of a set-point.

Generally a controller is made to be less reactive to the large errors, In this case the solution is to modify the top and the bottom rows. It means that to make significant modifications, it is best to make changes to the regions than to change an individual cells.

In this chapter the development of PD, PI and PID-Like fuzzy controllers are described.

3.2 Development of PD-Like Fuzzy Controller

The equation giving a conventional PD-controller is

$$u(k) = K_p * e(t) + K_D * \Delta e(t), \qquad (3.1)$$

Where K_P and K_D are the proportional and the differential gain factors. The structure of Fuzzy PD-Like controller is given in Figure 1.2.

To describe this equation with the help of rules, what inputs and outputs should be used for this rules table. The PD controller for any pair of the values of error (e) and change-of-error (Δe) calculate the control signal (u). The fuzzy controller should do the same thing. For any pair of error and change-of-error, it should work out the control signal. Then a PD-like fuzzy controller consists if rules, and a symbolic description of each are given as:

If e(t) is <property symbol> and Δe(t) is <property symbol> then u(t) is

<property symbol>,

Where *<property symbol>* is the symbolic name of a linguistic value.

The natural language equivalent of the above symbolic description reads as follows. For each sampling time (t):

If the value of error is <linguistic value> and the value of change-of error is < linguistic value > then the value of control output is < linguistic value >.

Consider the explicit reference to sampling time (t) is being omitted, since such a rule expresses a casual relationship between the process state and control output variables, which holds for any sampling time (t).

This is one of the linguistic qualifiers, determine for the proper variable: error, change-of-error or control signal, for example: high, low, medium, etc.

So, it is needed to have membership functions, describing all these qualifiers for all our variables: error, change-of-error or control.

Definitely, these variables might be measured in different units. So the rules when the PD controller used can be like:

If error is positive big and change-of-error is negative big then control is negative small.

It is needed to describe an error signal. Because the actual process output (y) can be higher than the desired one as well as lower, the error can be negative as well as positive. Values of error (e) with a negative sign mean that the current process output $y(t)$ has a value below that set-point (y_{sp}) since [$e(t) = y_{sp} - y(t) < 0$]. A negative value describes the magnitude of the difference ($y_{sp} - y$). On the other hand, linguistic value of (e) with a positive sign means that the current value of (y) is above the set-point. The magnitude of such a positive value is the magnitude of the difference ($y_{sp} - y$).

The change-of-error (Δe) with a negative sign means that the current process output $y(t)$ has increased when compared with its previous value $y(t-1)$, since

$$\Delta e(t) = e(t) - e(t-1) = -y(t) + y(t-1) < 0.$$

The magnitude of this negative value given by the magnitude of this increase. Linguistic value of $\Delta e(t)$ with a positive sign means that $y(t)$ has decreased its value when compared to $y(t-1)$. The magnitude of this value is the magnitude of the decrease.

A linguistic values of e with a negative sign mean that the current process output y has a value below the set-point y_{sp} since $e(t) = y_{sp} - y(t) < 0$. The magnitude of a negative value describes the magnitude of the difference $y_{sp} - y$. On the other hand, linguistic values of e with a positive sign mean that the current value of y is above the set-point. The magnitude of such a positive value is the magnitude of the difference $y_{sp} - y$.

A Linguistic value of (Δe) with a negative sign mean that the current process output $y(t)$ has increased when compared with its previous value $y(t-1)$ since $\Delta e(t) = -(y(t) - y(t-1)) < 0$. The magnitude of such a negative value is given by the magnitude of this increase. A Linguistic value of $\Delta e(t)$ with a positive sign means that $y(t)$ has decreased its value when compared to $y(t-1)$. The magnitude of such a value is the magnitude of the decrease.

A linguistic value of 'zero' for e means that the current process output is about the set-point. A 'zero' for Δe means that the current process output has not changed significantly from its previous value i.e. $-(y(t)-y(t-1))=0)$. The sign and the magnitude for u constitute the value of the control signal.

The Table (3.1) is suitable when we have two inputs and one output. On the topside of the table it should be written the possible linguistic values for the change-of-error Δe and on the left side the error e.

The cell of the table at the intersection of the row and the column will contain the linguistic value for the output corresponding to the value of the first input written as the beginning of the row and the value of the second input written on the top of the column.

Let us consider the example where both inputs and an output have a set of possible linguistic values {NB, NM, NS, Z, PS, PM, PB} where NB stands for Negative Big, NM stands for Negative Medium, NS stands for Negative Small, Z stands for Zero, PS strands for Positive Small, PM stands for Positive Medium and PB stands for Positive Big. The cell defined by the intersection of the first row and the first column represents a rule such as:

if e(t) is NB and Δe(t) is NB then u(t) is NB.

Table 3.1. PD-Like fuzzy controller linguistic rules.

e \ Δe	PB	PM	PS	Z	NS	NM	NB
PB	NB	NB	NB	NB	NM	NS	Z
PM	NB	NB	NB	NM	NS	Z	PS
PS	NB	NB	NM	NS	Z	PS	PM
Z	NB	NM	NS	Z	PS	PM	PB
NS	NM	NS	Z	PS	PM	PB	PB
NM	NS	Z	PS	PM	PB	PB	PB
NB	Z	PS	PM	PB	PB	PB	PB

Gr. 0	Gr. 1	Gr. 2	Gr. 3	Gr. 4

The table includes 49 rules. It is taking into account now not just the error but the change-of-error as well. It allows describing the dynamic of the controller.

To explain how this rules set works and how to choose the rules, let us divide the set of all rules into the following five groups:

Group 0: in this group of rules both (e) and (Δe) are (positive or negative) small or zero. This means that the current value of the process output variable (y) has divided from the desired level (the set-point) but is still close to it. Because of this closeness the control signal should be zero or small in magnitude and is intended to correct small deviation from the set-point. Therefore, the rules in this group are related to the steady-state behavior of the process. The change-of-error, when it is negative small or positive small, shifts the output to negative or positive region, because in this case, for example, when e(t) and Δe(t) are both negative small the error is already negative and, due to the negative change-of-error, tends to become more negative. To prevent this trend, one needs to increase the magnitude of the control output.

Group 1: for this group of rules e(t) is positive big or medium which implies that y(t) is significantly above the set-point. At the same time since Δe(t) is negative, this means that (y) is moving towards the set-point. The control signal is intended to either speed

up or slow down the approach to the set-point. For example, if $y(t)$ is much below the set-point (e(t) is positive big) and it's moving toward the set-point with small step ($\Delta e(t)$ is negative small) then the magnitude of this step has to be significantly increased (u(t) is negative medium). However, when $y(t)$ is still much below the set-point (e(t) is positive big) but it is moving towards the set-point very fast ($\Delta e(t)$ is negative big) no control action can be recommended because the error will be compensated due to the current trend.

Group 2: for this group of rules $y(t)$ is either close to the set-point(e(t) is positive small, zero, negative small) or significantly above it (negative medium, negative big).

At the same time, since $\Delta e(t)$ is negative, $y(t)$ is moving away from the set-point. The control here is intended to reverse this trend and make $y(t)$, instead of moving away from the set-point, start moving toward it. So here the main reason for the control action choice is not just the current error but also the trend in its change.

Group 3: for this group of rules e(t) is negative medium or big, which means that $y(t)$ is significantly below the set-point. At the same time, since $\Delta e(t)$ is positive, $y(t)$ is moving towards the set-point. The control is intended to either speed up or slow down the approach to the set-point. For example, if $y(t)$ is much above the set-point (e(t) is negative big) and its moving towards the set-point with a some what large step ($\Delta e(t)$ is positive medium), then the magnitude of this step has to be only slightly enlarged (u(t) is negative small).

Group 4: the situation here is similar to the group (2) in some sense. For this group of rules e(t) is either close to the set-point (positive small, zero, negative small) or significantly above it (positive medium, positive big). At the same time since (Δe) is positive $y(t)$ is moving away from the set-point. This control signal is intended to reverse this trend and make $y(t)$ instead of moving away from the set-point start moving towards it.

So, to design a PD-like controller it is needed just to create a rules table like Table (4.1). The contents of the table can be different. For example, we may replace the rule:

If e is PS and Δe is PM then u is NB

With the rule:

If e is PS and Δe is PM then u is NM

3.3 Development of PI-Like Fuzzy Controller

The equation given a conventional PI-controller is

$$u(k) = K_p * e(t) + K_1 * \int e(t)dt \qquad (3.2)$$

Where K_P and K_I are the proportional and the integral gain coefficients. A block diagram for a fuzzy control system is given in Figure 1.3.

It seems in this diagram to have a different form from the previous one. Differentiation with integration and a change-of-error with an integral error are replaced. Now the fuzzy controller and the rule table have other inputs. It means that the rules themselves should be reformulated. Sometimes it is difficult to formulate rules depending on an integral error, because it may have the very wide universe of discourse.

Move the integration from the part proceeding to a fuzzy controller to the part following it. It may have the error and the change of error inputs and still realize the PI-control.

When the derivative, with respect to time, of the equation (3.2) is taken, it is transformed into an equivalent expression

$$du(t)/dt = K_p * de(t)/dt + K_I * e(t)$$

Or in the discrete form:

$$\Delta u(t) = K_p * \Delta e(t) + K_I * e(t)$$

One can see here that one has the error and the change-of-error input and one needs just to integrate the output of a controller. One may consider the controller output not as a control signal, but as a change in the control signal. The gain factor K_I is used with the error input and K_P with the change-of-error.

The rule can be written as:

If e is<property symbol> and Δe is < property symbol >then Δu is < property symbol >.

In this case, to obtain the value of the control output variable u(t), the change – of-control output $\Delta u(t)$ is added to $u(t-1)$. It is necessary to stress here that this take place outside the PI-like fuzzy controller, and is not reflected in the rule themselves. Now let's compile the rules table for this type of a controller.

The output is not a control signal but the change-of-control.

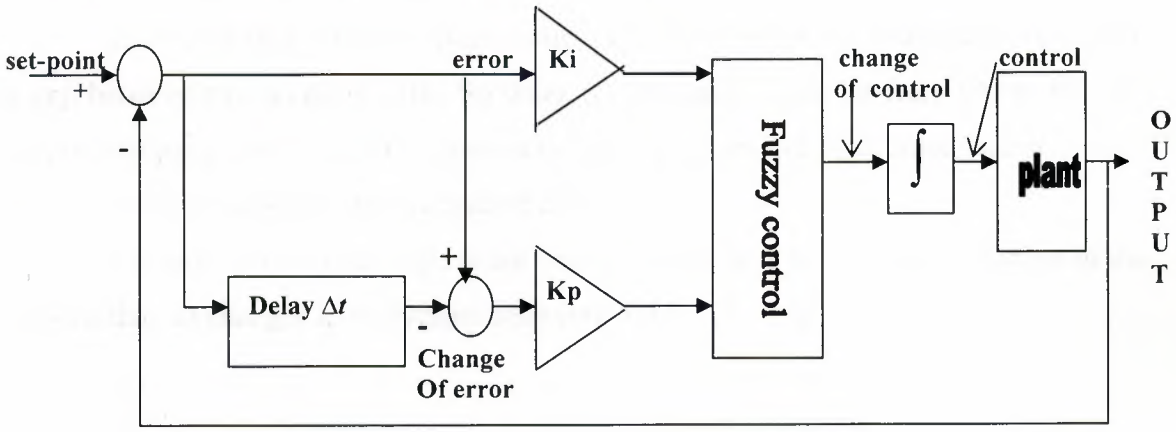


Figure 3.1. A block diagram of PI fuzzy control system.

If e is Z and Δe is NS then Δu is PM

And:

If e is Z and Δe is PS then Δu is NM.

Table 3.2. PI-Like fuzzy controller linguistic rules.

e \ Δe	PB	PM	PS	Z	NS	NM	NB
PB	NB	NB	NB	NB	NM	NS	Z
PM	NB	NB	NB	NM	NS	Z	PS
PS	NB	NB	NM	NS	Z	PS	PM
Z	NB	NM	NM	Z	PM	PM	PB
NS	NM	NS	Z	PS	PM	PB	PB
NM	NS	Z	PS	PM	PB	PB	PB
NB	Z	PS	PM	PB	PB	PB	PB

The correction will lead to the change of the control surface. The controller will become more reactive in the neighborhood of the set-point. It means that even small deviation errors will be lowed by larger control signals. It is difficult to say if it will become better in a general case. However, usually it is needed to make the control surface smoother in the vicinity of a set-point.

Generally a controller is made to be less reactive to the large errors, In this case the solution is to modify the top and the bottom rows.

In the case that the left bottom corner is changed to PS, for example there will be a gap between two adjacent cells. So when (e) changed a little bit from PM to PB, the output will jump from NS to PS. Generally, these gaps should be avoided and perform a smooth transformation between adjacent cells.

It means that to make significant modifications, it is best to make changes to the regions than to changes in individual cells (see Table 3.3).

Table 3.3. PI-Like fuzzy controller linguistic rules (after some modifications).

e \ Δe	PB	PM	PS	Z	NS	NM	NB
PB	NB	NB	NB	NM	NS	Z	Z
PM	NB	NB	NB	NM	NS	Z	PS
PS	NB	NB	NM	NS	Z	PS	PM
Z	NB	NM	NS	Z	PS	PM	PB
NS	NM	NS	Z	PS	PM	PB	PB
NM	NS	Z	PS	PM	PB	PB	PB
NB	Z	Z	PS	PM	PB	PB	PB

3.4 Development of PID-Like Fuzzy Controller

The equation for conventional PID-controller is as follows:

$$u = K_p * e + K_d * e + K_i * \int e dt,$$

Thus, in the discrete case of a PID-like fuzzy controller one as an additional process variables, namely sum-of-errors, denoted (σe) and computed as:

$$\sigma e(t) = \sum_{i=1} e(i) ,$$

The symbolic expression for a rule of a PID-like fuzzy controller is:

If e is < property symbol > and Δe is < property symbol > and σe is < property symbol > then u is < property symbol >

The last one has three conditions in the antecedent part but the previous ones had just two. So it will need to formulate many more rules to describe the PID-controller.

If any input is described with seven linguistic values, as it was before, then because the PID-controller has three inputs and any rule has three conditions, it will need [7*7*7=343] rules. Previously it had just [7*7=49] rules.

It is too much work to write [343] rules. The PID-like fuzzy controller can be constructed as a parallel structure of a PD-like fuzzy controller and a PI-like fuzzy controller with the output approximated as:

$$u = (Kp / 2 * e + Kd * de/dt) + (Kp / 2 * e + Ki * \int e dt).$$

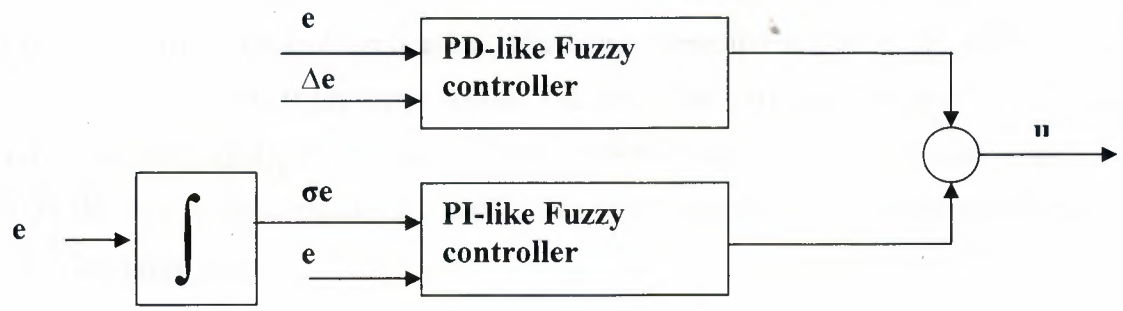


Figure 3.2. The structure for PID-like fuzzy controller.

When information about the object or process under control and its structure is available, one may not want to be confined to using error, change of error, and sum of errors as process state variables, but rather use the actual process state variables. The symbolic expression for a rule in the case of multiple inputs and a signal output (MISO) system is as follows:

If X_1 is < property symbol > and ... and X_n is < property symbol > then u is < property symbol >,

The fuzzy controller has been designed to control the turbine speed and pressure. The block diagram of fuzzy control system for a turbine speed control is given in Figure 3.3.

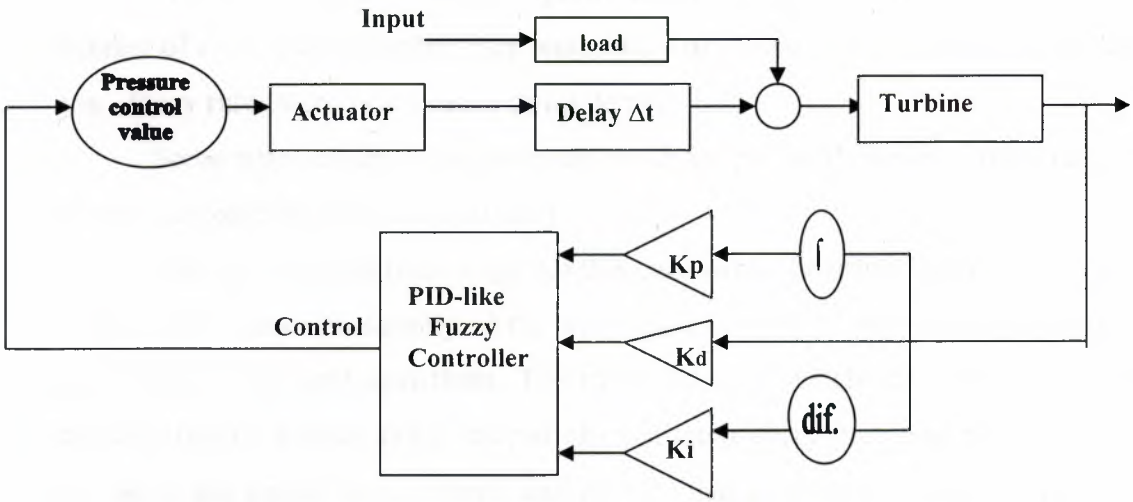


Figure 3.3. The block diagram of fuzzy controller system for a turbine speed control.

All blocks on this Figure demonstrate a nonlinear behavior, which is the main reason for a fuzzy control application. The fuzzy controller has been designed as PID-like fuzzy controller. It has three inputs: the error (the difference between a set-point and an actual output), the change of error, and the integral-error, and one output. To fuzzy the inputs, three classes are applied for each input with the membership function given in Figure 3.4.a.

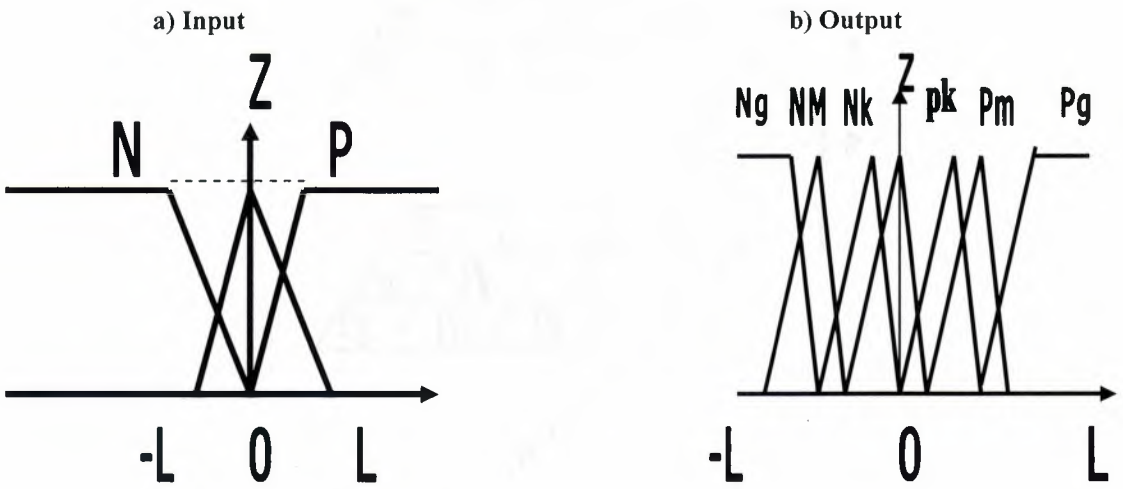


Figure 3.4. Membership functions for the inputs and outputs.

In order to defuzzify the output, seven classes are applied with the membership functions presented in Figure 3.4b.

Because the fuzzy controller has three inputs, its rules table has a three-dimensional image given in Figure 3.5.

The best way to improve the performance of fuzzy controller is to increase the number of rules used. Whether they were fuzzy or classical didn't really matter, but the non –fuzzy rules were much easier to implement.

Some types of servo operators are much simpler with standard rules (e.g., if the sensor goes past this line, then actuate).

The advantage of fuzzy logic for this case would be using fewer, simpler rules to handle all the cases reasonably. With fuzzy logic it needs to write three to six rules for size, motion, etc., and sum them. The rules follow more closely what a truck driver standing behind a truck does; instead of examining every item and plugging in every rule he or she knows about objects and decides that something is either 'a problem' or 'can be ignored' or ' Keep an eye on it' as the driver is guided backwards.

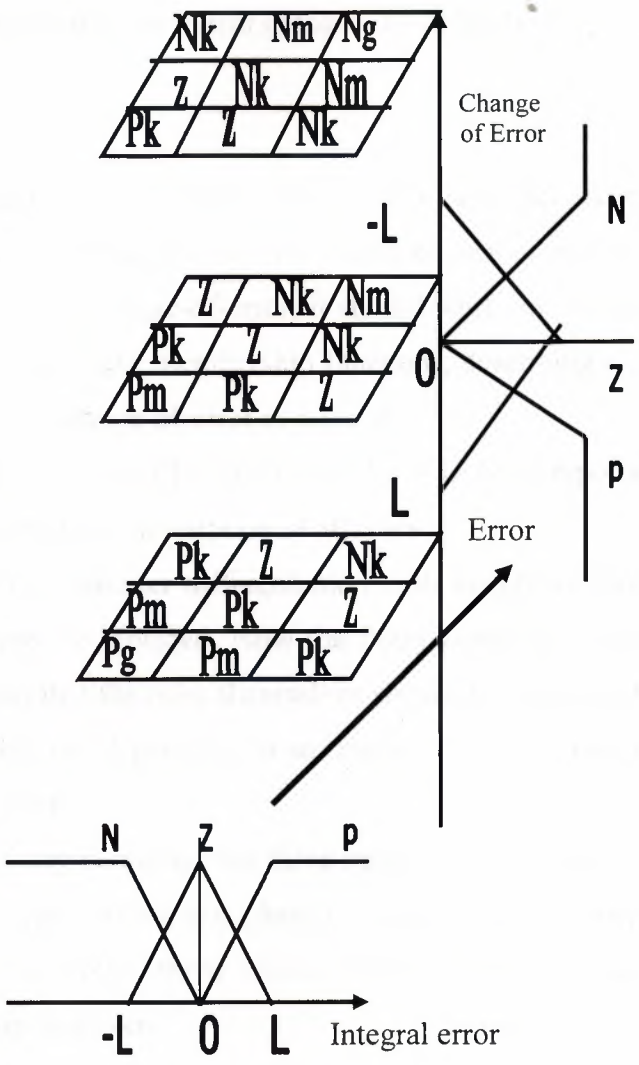


Figure 3.5. The rules table for the fuzzy controller.

This allows them to ignore cats, birds, open lunch boxes, paper bags, ignore pedestrians who obviously see the truck and are moving at safe distance form it without having to make a special rule for each of them, thinks like drunks stumbling under the wheels, broken glass, other trucks or unaware pedestrians receive more attention because they fit into the class of 'problem item'.

In one sense all measured inputs have some fuzziness even in classical control all have limits to precision. As a result no rule will be 'absolutely accurate', since the error terms, physical limitation (e.g., phase-shift and attenuation) and input noise will normally be a noticeable part of the input. Also filtering inevitably reduces the legitimate input, attenuating the original single. At the point, fuzzy logic simply

recognizes what have been doing all along and 'hard' rules are ignored: approximations onto the response that makes the useful outcome more likely.

3.5 Summary

The PD controller for any pair of the values of error (e) and change-of-error (Δe) should calculate the control signal (u). The fuzzy controller should do the same thing. For any pair of error and change-of-error, it should work out the control signal. At the same time it is needed to have membership functions, describing all these qualifiers for all our variables: error, change-of-error or control.

To design a PD, PI or PID-like controller it is very important to create a rules table that holds the linguistic description of all rules.

In the PI-Like controller differentiation with integration and a change-of-error with an integral error are replaced. Now the fuzzy controller and the rule table have other inputs. It means that the rules themselves should be reformulated. Sometimes it is difficult to formulate rules depending on an integral error, because it may have the very wide universe of discourse.

A PID-like fuzzy controller has three inputs: the error (the difference between a set-point and an actual output), the change of error, and the integral-error, and one output. To fuzzy the inputs, three classes should applied for each input with the different membership functions.

In chapter four development of fuzzy controller for control temperature of a heater is described. The general steps to develop and implement a fuzzy controller are shown.

CHAPTER FOUR

DEVELOPMENT OF FUZZY CONTROLLER FOR CONTROL TEMPERATURE OF A HEATER

4.1 Overview

A heater control has been designed that uses a fuzzy controller to sense the temperature, compare it to the user-selected temperature and control heater current. The inexpensive, dedicated fuzzy logic device is an adaptive controller that prevents thermal instability, providing consistent performance under all conditions.

This chapter describes all the steps of designing a fuzzy controller in general and a temperature of heater controller specifically.

4.2 Description of The Process

Household appliances that use heaters, such as ovens, rice cookers, toasters, should come quickly to the desired temperature and maintain it regardless of changes in conditions such as load or air-flow. Heating elements, because of thermal inertia, require a certain amount of time to change temperature. This between control and response causes the heater to overshoot and oscillate about the desired temperature. Once the temperature is achieved, varying environmental conditions often throw the heater back into oscillation. In addition, most heaters have poor temperature control due to the use of crude, on-off switches. The result is an inefficient and inconsistent operation.

To overcome this drawback, many industrial and consumer products as diverse as industrial chambers, ovens, rice cookers, toasters and irons use heating elements that could benefit advantages and feature enhancements of fuzzy logic control. A heater control has been designed that uses a fuzzy controller to sense the temperature, compare it to the user-selected temperature and control heater current. The inexpensive, dedicated fuzzy logic device is an adaptive controller that prevents thermal instability, providing consistent performance under all conditions.

We will describe both the fuzzy logic rules and design for the heater control. Most appliance heaters are controlled by bimetallic strips or capillary tubes that expand with heat and switch the heater on or off. These crude mechanical controllers merely react to temperature fluctuations and cannot anticipate when the heater is approaching the selected temperature. When the element passes through the operating point, the

switch opens, cutting power to the heater. But by this time, the heater has enough energy to carry the system temperature far above the selected range and it takes a while for it to return.

The switch stays open until the heater cools to the correct temperature. At that point the switch closes, but some time is required before the heater can again provide sufficient heat and the system cools well below the correct temperature. The overshoot and undershoot process can continue for minutes or hours. A change in the selected temperature or in environment (such as changing air conditions in a heating system or opening an oven door) may cause the heater to go into oscillation again.

Heaters have widely varying characteristics. They are specified in terms of their form including length, shape, thickness and material composition. Heating elements may add instability to a system because of their slow response time and thermal inertia.

The heater specifications are based on the requirements of the end product. The end product also has a range of thermal characteristics that influence the behavior of the heater. Many variables of heating systems make the design of a controller for different systems a difficult task. A control system using a fuzzy controller brings the temperature of the heater to the selected temperature quickly and keeps it there regardless of any changes in the load or environment. This results is a more stable and reliable operating temperature.

There are three external inputs monitored by the controller. The first comes from a thermistor to monitor the temperature. The second is the user-selected, desired temperature setting. Input three is, again, the measured thermistor value signal only delayed by a small amount of time. This last input enables the controller to know the direction and magnitude of the temperature change in addition to the absolute temperature. The controller samples the input data, processes it and outputs a pulse width modulated (PWM) output signal that switches a triac controlling the current through the heater.

4.3 Rule Base Formulation

Using knowledge of experts the fuzzy IF-THEN rules for controller are generated. The Table 4.1 will represent abstract knowledge of human-expert to control the temperature of the heater.

The main part of any fuzzy controller is implemented as a set of rules. It performs the control algorithm. By studying the rules, one can see the criteria for taking actions such as switching the heater on or off. These rules make decisions based on adjustable membership function definitions. The rules are easily modified to respond to different criteria. The following describes the rule's purposes in relation to the inputs, their associated fuzzy variables, and the action taken when a rule is fired.

Table 4.1. Expert knowledge for controlling the temperature of a heater.

e \ e'	**PB**	**PM**	**PS**	**Z**	**NS**	**NM**	**NB**
PB	NB	NB	NB	NB	NM	NS	Z
PM	NB	NB	NB	NM	NS	Z	PS
PS	NB	NB	NM	NS	Z	PS	PM
Z	NB	NM	NM	Z	PM	PM	PB
NS	NM	NS	Z	PS	PM	PB	PB
NM	NS	Z	PS	PM	PB	PB	PB
NB	Z	PS	PM	PB	PB	PB	PB

Here NB is Negative Big, NM is Negative Medium, NS is Negative Small, N is Normal, PS is Positive Small, PM is Positive Medium, PB is Positive Big.

Using Table 4.1 we can form IF-THEN rules. In Figure 4.2 fragment of rule base is given,

1. **If** e = Z and e' = Z **Then** U = Z.
2. **If** e = NL and e' = Z **Then** U = PB.
3. **If** e = PL and e' = Z **Then** U = NB.
4. **If** e = NM and e' = PB **Then** U = NS.
5. **If** e = NM and e' = PS **Then** U = NS.
6. **If** e = PM and e' = NS **Then** U = PS.
7. **If** e = PM and e' = NB **Then** U = PS.
8. **If** e = NM and e' = Z **Then** U = PB.
9. **If** e = PM and e' = Z **Then** U = NS.
10. **If** e = NS and e' = NB **Then** U = PB.
11. **If** e = NS and e' = PS **Then** U = Z.
12. **If** e = NS and e' = Z **Then** U = PS.
13. **If** e = PS and e' = Z **Then** U = NS.

Figure 4.1. Some of the If-Then rules taken from Tabe 4.1.

Let consider the implementation of the blocks of fuzzy controller to control the temperature of heater. As described above linguistic variables are represented by membership functions.

Many different choices of membership functions are possible, for our case to describe linguistic terms, in rule base shown in Figure 4.1, the triangle membership function is used. The membership function quantities wheather values of e(t) error belong to the set of values that are "positive small", and hence it quantities the mean using of the linguistic statement "error is positive small".

To specify the meaning of a linguistic value via a membership function we can specify the membership functions for all linguistic values (seven for each input and seven for the output) in Table 4.1. Figure 4.2 demonstrates the choice of membership function for input variables error, change of error and output variable voltage.

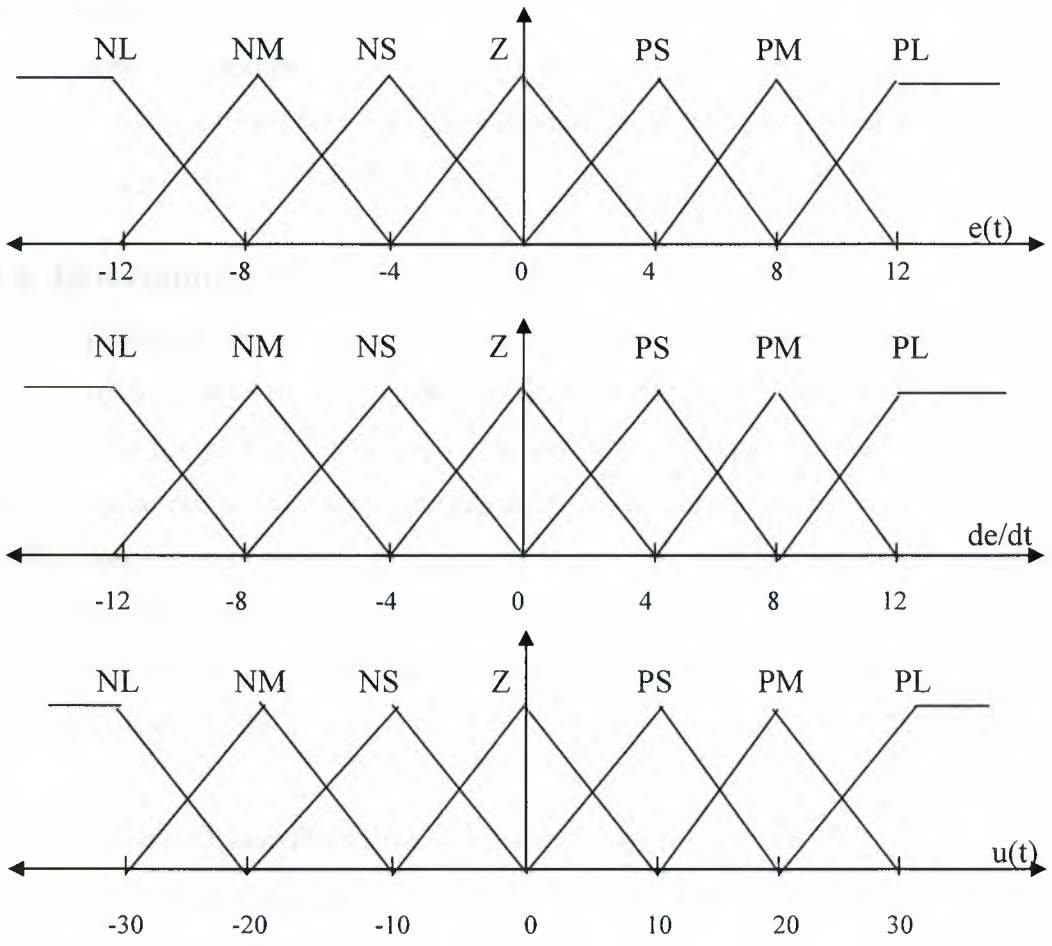


Figure 4.2. Choices of membership functions for input and output variables.

For the output u, the membership functions at the outermost edges cannot be saturated for the fuzzy system to be properly defined. The basic reason for this takes actions an exact value for the process input. Generally cannot indicate to a process actuator, "any value bigger than, say, 10, is acceptable."

The rule-base of the fuzzy controller holds the linguistic variables, linguistic values, their associated membership functions, and the set of all linguistic rules (shown in Table 4.1) describing the simple heater temperature controller.

4.4 Fuzzification

The fuzzification process as the act of obtaining a value of an input variable (e.g., $e(t)$) and finding the numeric values of the membership function(s) that are defined for that variable error. For example, if $e(t) = 8$ and $de(t)/dt = 2$, the fuzzification process amounts to finding the values of the input membership functions for these. In this case $\mu_{ps}=1$ (with all others zero) and $\mu_{zero}(de(t)/dt)= \mu_{ps}(de(t)/dt)=0.5$.

Here the membership function values as an "encoding" of the fuzzy controller numeric input values. The encoded information is then used in the fuzzy inference process that starts with "matching."

4.5 Determining Which Rules to Use

1. The premises of all the rules are compared to the controller inputs to determine which rules apply to the current situation. This "matching" process involves determining the certainty that each rule applied, and typically take into account the recommendations of rules that certainly apply to the current situation.
2. The conclusions (what control actions to take) are determined using the rules that have been determined to apply at the current time. The conclusions are characterized with a fuzzy set (or sets) that represents the certainty that the input to the plant should take on various values.

4.6 Premise Quantification via Fuzzy Logic

To perform inference mechanism first each of the rules must be quantified with fuzzy logic. To do this first quantify the meaning of the premises of the rules that are composed of several terms, each of which involves a fuzzy controller input.

If error is Z ***and*** *change-in-error is PS* ***Then*** *output is NS*

Above, the meaning of the linguistic terms "error is zero" and "change-in- error is Positive Small" via the membership functions had quantified. Now quantifing the linguistic premise "*error is zero* ***and*** *change-in-error is Positive Small.*" Hence, the main item to focus on is how to quantify the logical "and" operation that combines the meaning of two linguistic terms. While standard boolean logic used to combine these linguistic terms, since they quantified more precisely with fuzzy sets (i.e., the membership functions).

To see how to quantify the "and" operation, begin by supposing that e(t) = 4 and *de(t)/dt*=1, so that using Figure 4.2 (or Figure 4.3) so

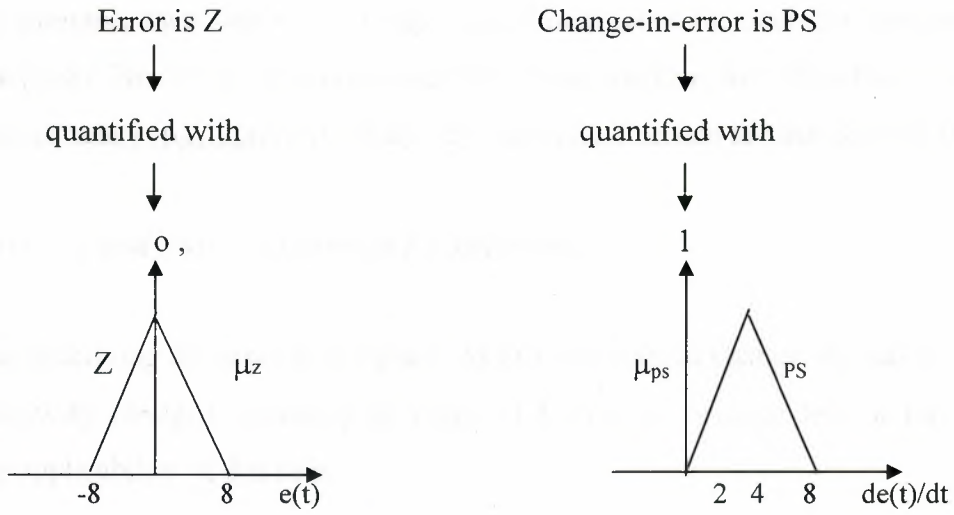


Figure 4.3. Quantifying AND operation.

$\mu_z(e(t))=0.5$ and $\mu_{ps}(de/dt)=0.5$

What, for these values of *e(t)* **and** *de(t)/dt,* is the certainty of the statement

"error is Z ***and*** *change-in-error is PS"*

that is the premise from the above rule. This certainty will be denoted by $\mu_{premise}$. There are actually several ways to define it:

Minimum: Define $\mu_{premise}=\min\{0.5,0.25\}=0.25$, that is, using the minimum of the two membership values.

Product: Define $\mu_{premise}=(0.5)(0.25)=0.125$, that is, using the product of the two membership values.

Notice that both ways of quantifying the "and" operation in the premise indicate that you can be no more certain about the conjunction of two statements than you are about the individual terms that make them up (note that $0\leq\mu_{premise}\leq1$ for either case).

While this is simply shown how to quantify the "and" operation for one value of *e(t)* and *de(t)/dt,* consider all possible e(t) and *de(t)/dt* values, the multidimensional membership function $\mu_{premise}(e(t), de(t)/dt)$ will be obtained that is a function of *e(t)* and d*e(t)/dt* for each rule. For this example, choose the minimum operation to represent the "and" in the premise, then this result to the multidimensional membership function $\mu_{premise}(e(t),de(t)/dt)$. Notice that picking values for *e(t)* and *de(t)/dt* will allow the value of the premise certainty $\mu_{premise}(e(t),de(t)/dt)$ represents how certain are that the rule is

If *error is Z* ***and*** *change-in-error is PS* ***Then*** *output NS*

applicable for specifying the input to the plant. As *e(t)* and d*e(t)/dt* change, the value of $\mu_{premise}(e(t),de(t)/dt)$ changes according to Figure 4.4, and we become less or more certain of the applicability of this rule.

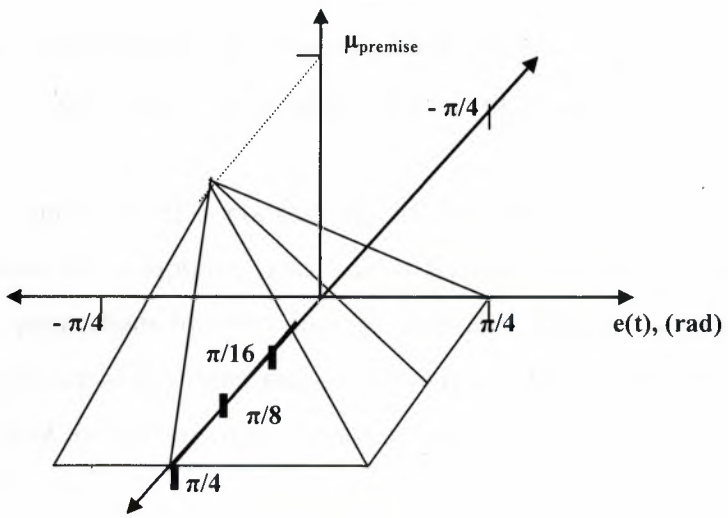


Figure 4.4. Membership function of the premise for a single rule.

4.7 Determining Which Rules Are On

Determining the applicability of each rule is called "matching." It said that a rule is "on at time *t*" if its premise membership function $\mu_{premise}(e(t),de(t)/dt) > 0$. Hence, the inference mechanism seeks to determine which rules are on to find out which rules are relevant to the current situation.

Consider, for the Heater example, how to compute the rules that are on. Suppose that

$$e(t)=0 \quad and \quad de(t)/dt=4-1=3$$

Figure 4.5 shows the membership functions for the inputs and indicates with thick black vertical lines the values above for $e(t)$ and $de(t)/dt$. Notice that $\mu_{zero}(e(t))=1$ but that the other membership functions for the $e(t)$ input are all "off" (i.e., their values are zero). For the $de(t)/dt$ input it is clearly seen that $\mu_{zero}(de(t)/dt)=0.25$ *and* $\mu_{possmal}(de(t)/dt) = 0.75$ and that all the other membership functions are off. This implies that rules that have the premise terms are on (all other rules have $\mu_{premise}(e(t),de(t)/dt) = 0$.

"Error is Z"

"Change-in-error is Z"

"Change-in-error is PS"

So, which rules are these? Table 4.1 is shows that the rules that are on are the following:

1. ***If*** *error is Z* ***and*** *change-in-error is Z* ***Then*** *output is Z*
2. ***If*** *error is Z* ***and*** *change-in-error is PS* ***Then*** *output is NS*

Note that since for the controlling of heater temperature example two membership functions over- lapping, it will never have more than four rules on at one time (this concept generalizes to many inputs). Actually, for this system either it has one, two, or four rules on at any one time. To get only one rule on choose, for example, $e(t) = 0$ and $de(t)/dt=4$ so that only rule 2 above is on.

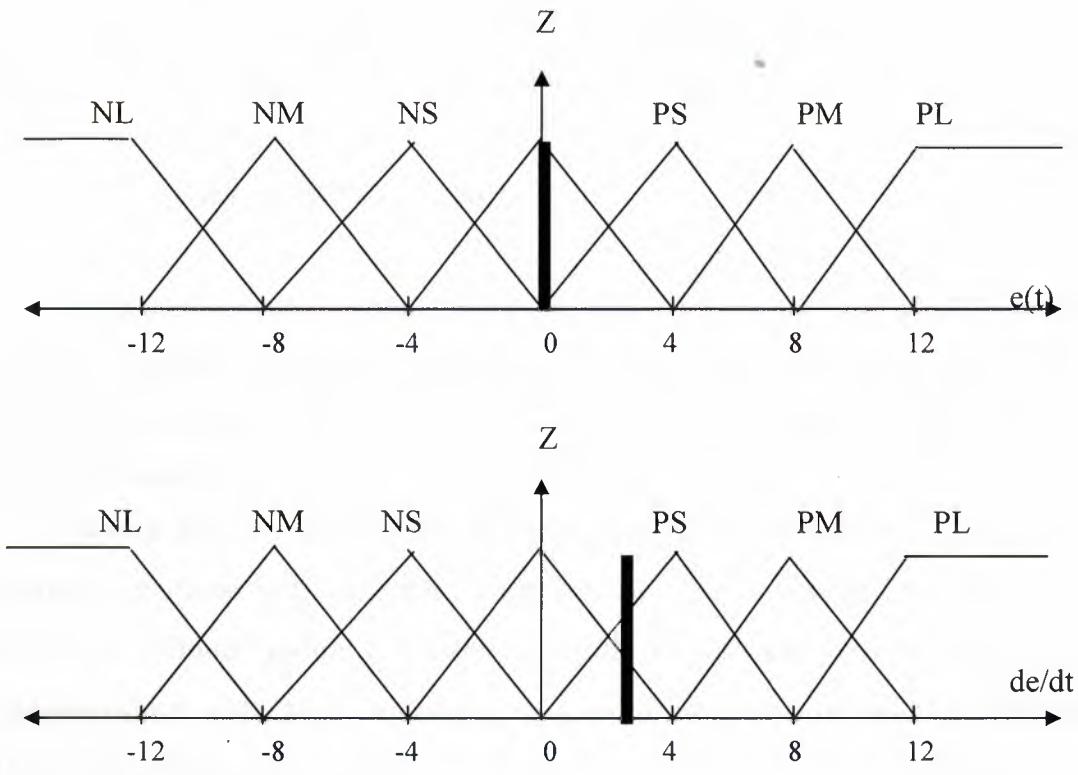


Figure 4.5. Input membeship functions with input values.

4.8 Inference Step: Determining Conclusions

Next, the question is how to determine which conclusions should be reached when the rules that are on are. To do this, first consider the recommendations of each rule independently. Then later combine all the recommendations from all the rules.

4.9 Recommendation from One Rule

Consider the conclusion reached by the rule

If *error is Z* ***and*** *change-in-error is Z* ***Then*** *T is Z*

which for convenience we will refer to as "rule (1)." Using the minimum to represent the premise, so

$$\mu_{premise1}=min\{0.25,1\}=0.25$$

The notation $\mu_{premise1}$ represents $\mu_{premise}$ for rule (1) so that the certainty are 0.25 that this rule applies to the current situation. The rule indicates that if its premise is true then the action indicated by its consequent should be taken. For rule (1) the consequent is "temperature is zero". The membership function for this consequent is shown in

Figure 4.6 (a). The membership function for the conclusion reached by rule (1), which we denote by μ_1, is shown in Figure 4.6 (b) and is given by

$$\mu_1(u) = min\{0.25, \mu_{zero}(u)\}$$

This membership function defines the "implied fuzzy set" for rule (1) (i.e., it is the conclusion that is implied by rule (1)). The justification for the use of the minimum operator to represent the implication is that can be no more certain about the consequent than our premise.

Notice that the membership function $\mu_1(u)$ is *a function* of u and that the minimum operation will generally "chop off the top" of the $\mu_{zero}(u)$ membership function to produce $\mu_1(ut)$. For different values of $e(t)$ and $de(t)/dt$ there will be different values of the premise certainty $\mu_{premise}(e(t), de(t)/dt)$ for rule (1) and hence different *functions* $\mu_1(u)$ obtained (i.e., it will chop off the top at different points).

It is clearly seen that $\mu_1(u)$ is in general a time-varying function that quantifies how certain rule (1) is that the temperature input u should take on certain values. It is most certain that the temperature input should lie in a region around zero (see Figure 4.6(b)), and it indicates that it is certain that the force input should not be too large in either the positive or negative direction-this makes sense if you consider the linguistic meaning of the rule. The membership function $\mu_1(u)$ quantifies the conclusion reached by only rule (1) and only for the current $e(t)$ and $de(t)/dt$. It is important that the reader be able to picture how the shape of the implied fuzzy set changes as the rule's premise certainty changes over time.

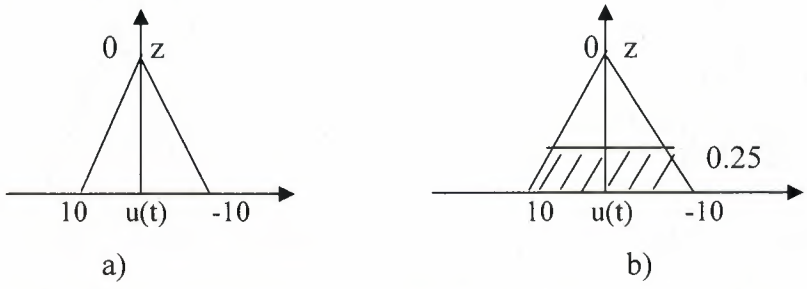


Figure 4.6. (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_1(u)$ for rule (1).

4.10 Recommendation From Another Rule

Next, consider the conclusion reached by the other rule that is on,

If error is Z and change-in-error is PS Then u is NS

which for convenience will refer to as "rule (2)." Using the minimum to represent the premise, then

$$\mu_{premise2}(u)=min\{0.75,1\}=0.75$$

so that it is 0.75 certain that this rule applies to the current situation. Notice that it is much more certain that rule (2) applies to the current situation than rule (1). For rule (2) the consequent is "T is negative small". The membership function for this consequent is shown in Figure 4.7 (a). The membership function for the conclusion reached by rule (2), which we denote by $\mu_2(u)$, is shown in Figure 4.7 (b) (the shaded region) and is given by

$$\mu_2(u)=min\{0.75,\ \mu_{negsmall}(u)$$

this membership function defines the implied fuzzy set for rule (2) (i.e., it is the conclusion that is reached by rule (2)). Once again, for different values of $e(t)$ and $de(t)/dt$ there will be different values of $\mu_{premise2}(e(t),de(t)/dt)$ for rule (2) and hence different functions $\mu_2(u)$ obtained. Rule (2) is quite certain that the control output (process input) should be a small negative value. As rule (2) has a premise membership function that has higher certainty than for rule (1), note that it is more certain of the conclusion reached by rule (2).

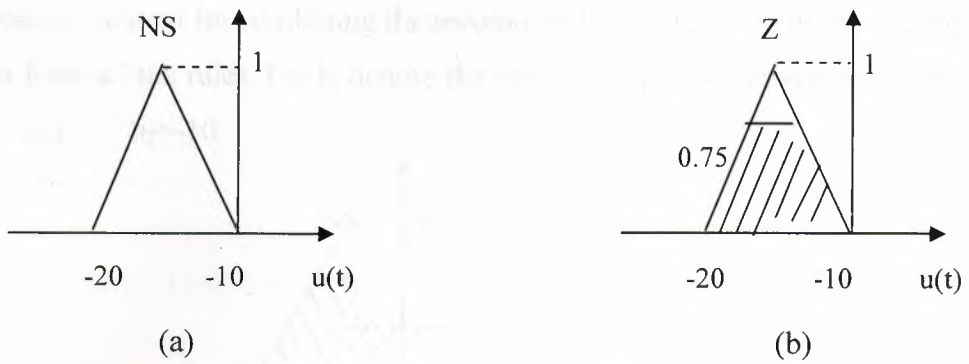


Figure 4.7. consequent membership function (a), and implied fuzzy set with membership function $\mu_2(u)$ for rule (2)

This completes the operations of the inference mechanism. While the input to the inference process is the set of rules that are on, its output is the set of implied fuzzy sets that represent the conclusions reached by all the rules that are on. For our example, there are at most four conclusions reached since there are at most four rules on at any one time. (In fact, you could say that there are *always* four conclusions reached for our example, but that the implied fuzzy sets for some of the rules may have implied membership functions that are zero for all values).

4.11 Converting Decisions Into Actions

Next, consider the defuzzification operation, which is the final component of the fuzzy controller. Defuzzification operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the "most certain" controller output (plant input). Some think of defuzzification as "decoding" the fuzzy set information produced by the inference process (i.e., the implied fuzzy sets) into numeric fuzzy controller outputs. These are a number of defuzzification algorithms they are described in chapter 2.

To demonstrate defuzzification, it is best to first draw all the implied fuzzy sets on one axis as shown in Figure 4.8. Then find the one output, which denoted by $"u^{crisp}"$ that best represents the conclusions of the fuzzy controller that are represented with the implied fuzzy sets. There are actually many approaches to defuzzification.

4.11 Combining Recommendations

Due to its popularity, first consider the "center of gravity" (COG) defuzzification method for combining the recommendations represented by the implied fuzzy sets from all the rules. Let b_i denote the center of the membership function, then $b_1=0$ and $b_2=-10$

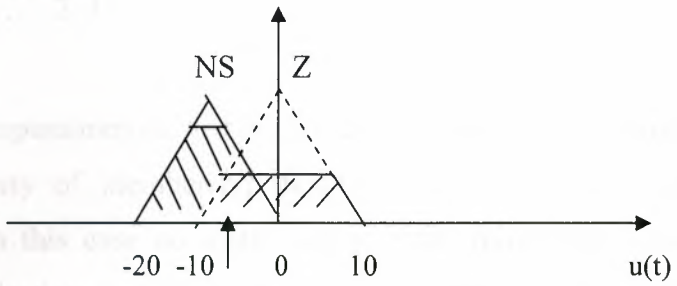


Figure 4.8. Combining recommendations.

as shown in Figure 4.8. Let $\int \mu_i$ denote the area under the membership function μ_i. The COG method computes u^{crisp} to be

$$u^{crisp} = \frac{\sum_i b_i \int \mu_i}{\sum_i \int \mu_i} \tag{4.1}$$

This is the classical formula for computing the center of gravity. In this case it is for computing the center of gravity of the implied fuzzy sets. Three items about equation (4.1) are important to note:

1. Practically, there are no output membership functions that have infinite area since even though they may be "chopped off' in the minimum operation for the implication (or scaled for the product operation) they can still end up with infinite area. This is the reason we do not allow infinite area membership functions for the linguistic values for the controller output.

2. You must be careful to define the input and output membership functions so that the sum in the denominator of equation (4.1) is not equal to zero no matter what the inputs to the fuzzy controller are. Essentially, this means that there must be some sort of conclusion for all possible control situations that may encounter.

3. While at first glance it may not appear so, $\int \mu_i$ is easy to compute. For the case of symmetric triangular output membership functions that peak at one and have a base width of *w*, simple geometry can be used to show that the area under a triangle "chopped off' at a height of *h* (such as the ones in Figures 4.6 and 4.7) is equal to

$$\omega\left(h - \frac{h^2}{2}\right).$$

Given this, the computations needed to compute u^{crisp} are not too significant.

The property of membership functions being symmetric for the output is important since in this case no matter whether the minimum or product is used to represent the implication, it will be the case that the center of the implied fuzzy set will be the same as the center of the consequent fuzzy set from which it is computed. If the output membership functions are not symmetric, then their centers, which are needed in

the computation of the COG, will change depending on the membership value of the premise. This will result in the need to recompute the center at each time instant.

Using Equation (4.1) with Figure 4.8 result to the following equation

$$u^{crisp} = \frac{(0)(4.375) + (-10)(9.375)}{4.375 + 9.375} = -6.81$$

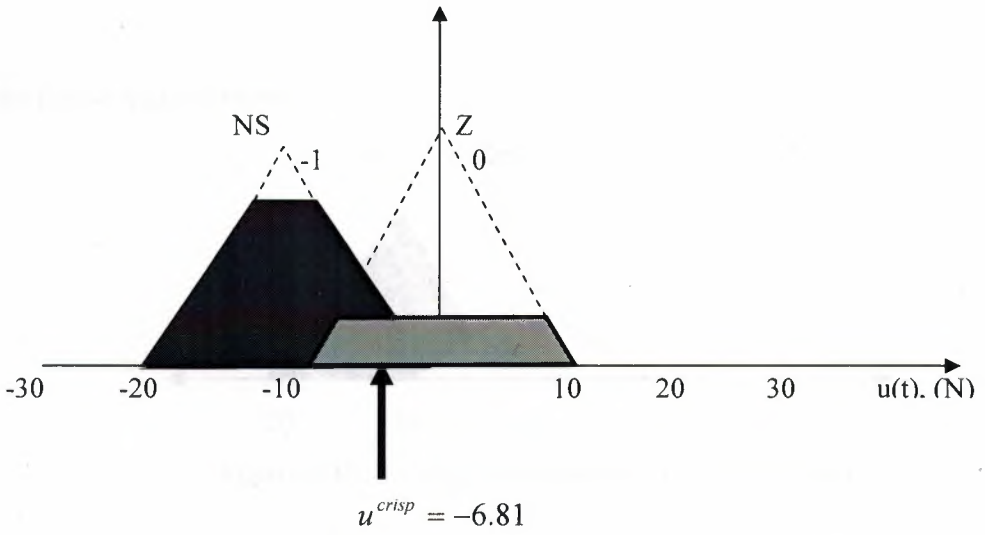


Figure 4.9. Represents the Center Of Overage Defuzzification process

4.12 Other Ways to Compute and Combine Recommendations

As another example, it is interesting to consider how to compute, by hand, the operations that the fuzzy controller takes when using the product to represent the implication or the "center-average" defuzzification method.

First, consider the use of the product. Consider Figure 4.9, where the output membership functions have been drawn for "negative small" and "zero" as dotted lines. The implied fuzzy set from rule (1) is given by the membership function shown in Figure 4.10.

$$\mu_1(u) = 0.25\mu_{zero}(u)$$

as the shaded triangle; and the implied fuzzy set for rule (2) is given by the membership function shown in Figure 4.10 as the dark triangle.

$$\mu_2(u)=0.75\mu_{NS}(u)$$

Notice that computation of the COG is easy since it can use *wh* as the area for a triangle with base width *w* and height *h*. When using product to represent the implication, the following equation will be obtained

$$u^{crisp} = \frac{(0)(2.5)+(-10)(7.5)}{2.5+7.5} = -7.5$$

which also makes sense

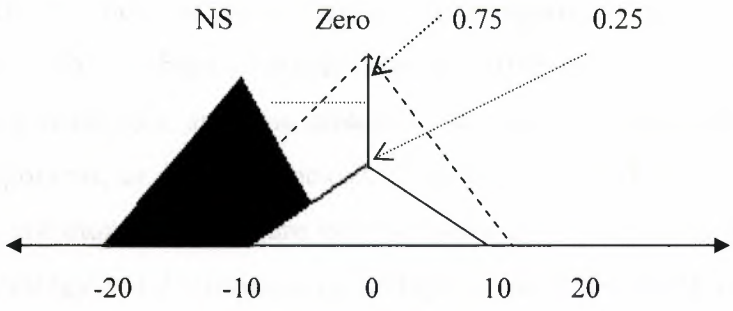


Figure 4.10. Example of combining recommendations.

Next, as another example of how to combine recommendations, we will introduce the "Center-Average" method for defuzzification. For this method we let

$$u^{crisp} = \frac{\sum_i b_i \mu_{premise_i}}{\sum_i \mu_{premise_i}} \qquad (4.2)$$

where to compute $\mu_{premisei}$ we use, for example, minimum. This is called the "center-average" method since equation (4.2) is a weighted average of the center values of the output membership function centers. Basically, the center-average method replaces the areas of the implied fuzzy sets that are used in COG with the values of $\mu_{premisei}$. This is a valid replacement since the area of the implied fuzzy set is generally proportional to $\mu_{premisei}$ since $\mu_{premisei}$ is used to chop the top off (minimum) or scale (product) the triangular output membership function when COG is used. For the above example,

$$u^{crisp} = \frac{(0)(0.5) + (-10)(0.75)}{0.25 + 0.75} = -7.5$$

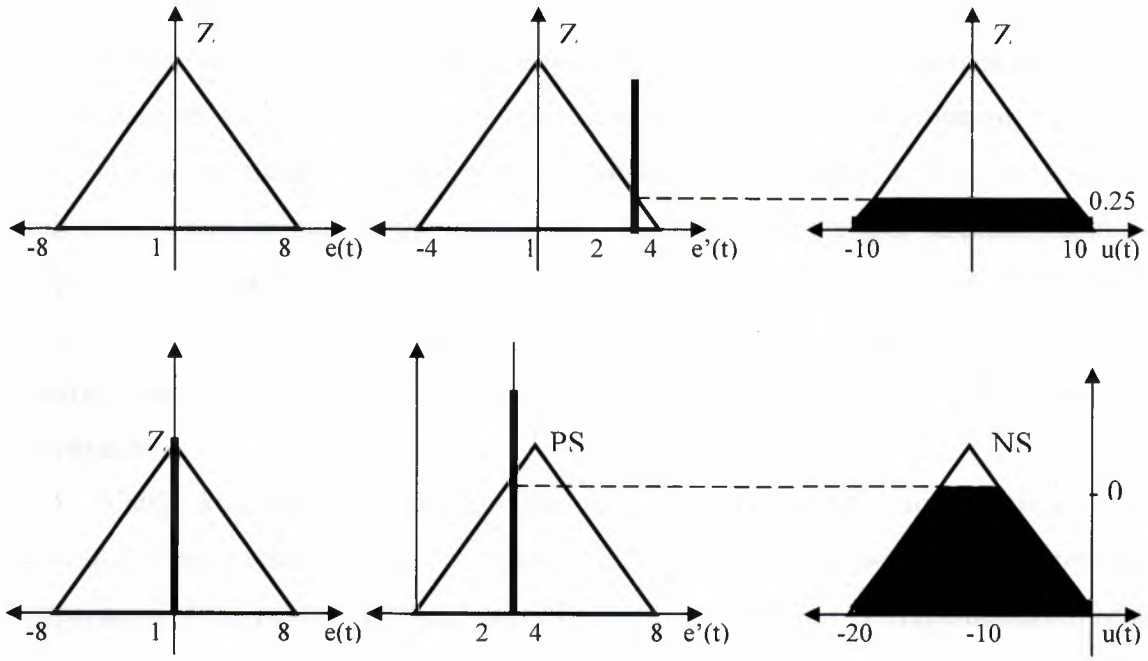
which just happens to be the same value as above. Some like the center-average defuzzification method because the computations needed are simpler than for COG and because the output membership functions are easy to store since the only relevant information they provide is their center values (B_i) (i.e., their shape does not matter, just their center value).

Notice that while both values computed for the different inference and defuzzification methods provide reasonable command inputs to the plant, it is difficult to say which is best without further investigations (e.g., simulations or implementation). This ambiguity about how to define the fuzzy controller actually extends to the general case and also arises in the specification of all the other fuzzy controller components, as discussed below. Some would call this "ambiguity" a design flexibility, but unfortunately there are not too many guidelines on how best to choose the inference strategy and defuzzification method, so such flexibility is of questionable value.

4.13 Graphical Depiction of Fuzzy Decision Making

Let summarize the procedure that the fuzzy controller uses to compute its outputs given its inputs in Figure 4.11. Here, let use the minimum operator to represent the "and" in the premise and the implication and COG defuzzification. To develop a similar diagram for the case where the product operator is used to represent the "and" in the premise and the implication, and choose values of *e(t)* and *de(t)/dt* that will result in four rules being on. Then, repeat the process when center-average *de(t)/dt* defuzzification is used with either minimum or product used for the premise. Also, learn how to picture in your mind how the parameters of this graphical representation of the fuzzy controller operations change as the fuzzy controller inputs change.

This completes the description of the operation of a simple fuzzy controller



If error is Z **and** change-in-error is PS **Then** output is NS

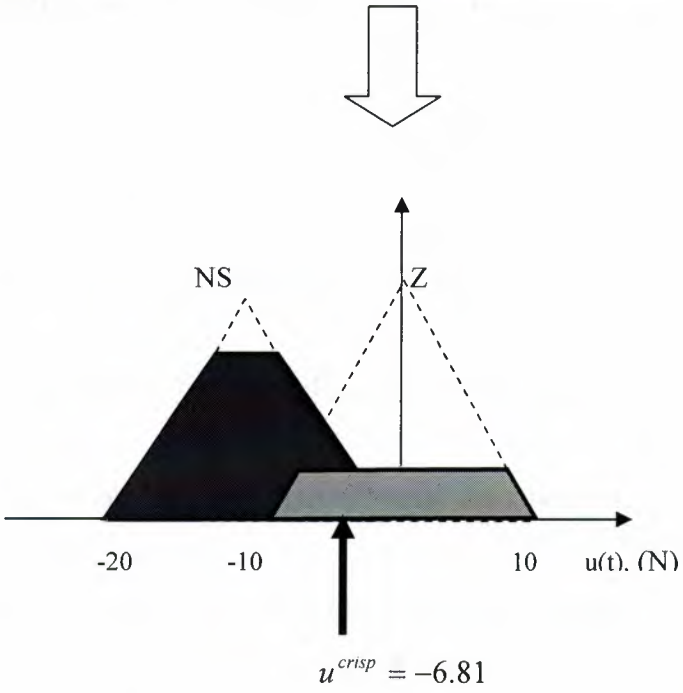


Figure 4.11. Graphical representation of fuzzy controller operations.

4.14 Summary

A control system using a fuzzy controller brings the temperature of the heater to the selected temperature quickly and keeps it there regardless of any changes in the load or environment. This results is a more stable and reliable operating temperature.

There are three external inputs monitored by the controller. The first comes from a thermistor to monitor the temperature. The second is the user-selected, desired temperature setting. Input three is, again, the measured thermistor value signal only delayed by a small amount of time. This last input enables the controller to know the direction and magnitude of the temperature change in addition to the absolute temperature.

Using knowledge of experts the fuzzy IF-THEN rules for controller are generated. Then we should represent abstract knowledge of human-expert to control the temperature of the heater. the main part of any fuzzy controller is implementation of a set of rules. It performs the control algorithm. By studying the rules, one can see the criteria for taking actions such as switching the heater on or off. These rules make decisions based on adjustable membership function definitions. The rules are easily modified to respond to different criteria.

The fuzzification process as the act of obtaining a value of an input variable (e.g., *e(t))* and finding the numeric values of the membership function(s) that are defined for that variable error.

After fuzzification we should determine which rules to use, there are two ways to do so:

1. The premises of all the rules are compared to the controller inputs to determine which rules apply to the current situation.
2. The conclusions (what control actions to take) are determined using the rules that have been determined to apply at the current time.

To perform inference mechanism first each of the rules must be quantified with fuzzy logic. To do this first quantify the meaning of the premises of the rules that are composed of several terms, each of which involves a fuzzy controller input.

Determining the applicability of each rule is called "matching." It said that a rule is "on at time *t"* if its premise membership function $\mu_{premise}(e(t), de(t)/dt) > 0$. Hence, the inference mechanism seeks to determine which rules are on to find out which rules are relevant to the current situation.

Next, consider the defuzzification operation, which is the final component of the fuzzy controller. Defuzzification operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the "most certain" controller output (plant input).

Chapter five presents the computer simulation of fuzzy system for control temperature of heater using Matlab Programming Language.

CHAPTER FIVE

MODELING OF FUZZY CONTROLLER FOR CONTROLING TEMPERATURE OF HEATER

5.1 Overview

The computer simulation of temperature controller is realized using a graphical user interface (GUI) tools provided by the fuzzy logic toolbox in Matlab package.

The computer simulation of control system with PD-Like fuzzy controller have been carried out. Result of computer simulation of fuzzy control system is compared with the result of simulation of control system with PD-Like controller. Time response characteristic of control system with PD controller and fuzzy controller is shown in the same diagram which shows the efficiency of the fuzzy controller over the conventional PD controller.

5.2 Modeling Fuzzy Controller Using MATLAB Package

The computer simulation of fuzzy system for control temperature of a heater is performed in Matlab package. To control the temperature of heater the PD-like fuzzy controller is used.

During simulation the temperature of heater is described by second order differential equation:

$$a_0 y'' + a_1 y' + a_2 y = Bu \qquad (5.1)$$

Here y is controlled temperature variable, y' and y'' first and second order derivatives of controlled variable, u is control signal (the output of controller), a_0=0.075, a_1=0.64, a_2=1, B=40 are the parameters of control object.

The rule base for controller is given in Table 5.1.

e / e'	PB	PM	PS	Z	NS	NM	NB
PB	NB	NB	NB	NB	NM	NS	Z
PM	NB	NB	NB	NM	NS	Z	PS
PS	NB	NB	NM	NS	Z	PS	PM
Z	NB	NM	NM	Z	PM	PM	PB
NS	NM	NS	Z	PS	PM	PB	PB
NM	NS	Z	PS	PM	PB	PB	PB
NB	Z	PS	PM	PB	PB	PB	PB

Table 5.1. Linguistic rule base for the controller.

The computer simulation of temperature controller is realized using a Graphical User Interface (GUI) tools provided by the fuzzy logic toolbox in Matlab Package (Appendix I).

In general, there are five primary GUI tools for building, editing, and observing fuzzy inference systems in the fuzzy logic tool box: the Fuzzy Inference System or FIS Editor, the Membership Function Editor, the Rule Editor, The Rule Viewer, and the Surface Viewer. Lets consider each of these tools separately.

5.2.1 The FIS Editor

The FIS Editor for control temperature of a heater example is shown in Figure 5.1. In our example we have two inputs, error and change of error, and one output, the FIS Editor displays general information about a fuzzy inference system. There's a simple diagram at the top that shows the name of each input variables on the left and output variable on the right. The sample membership functions in the boxes are just icons and do not depict the actual shapes of the membership functions.

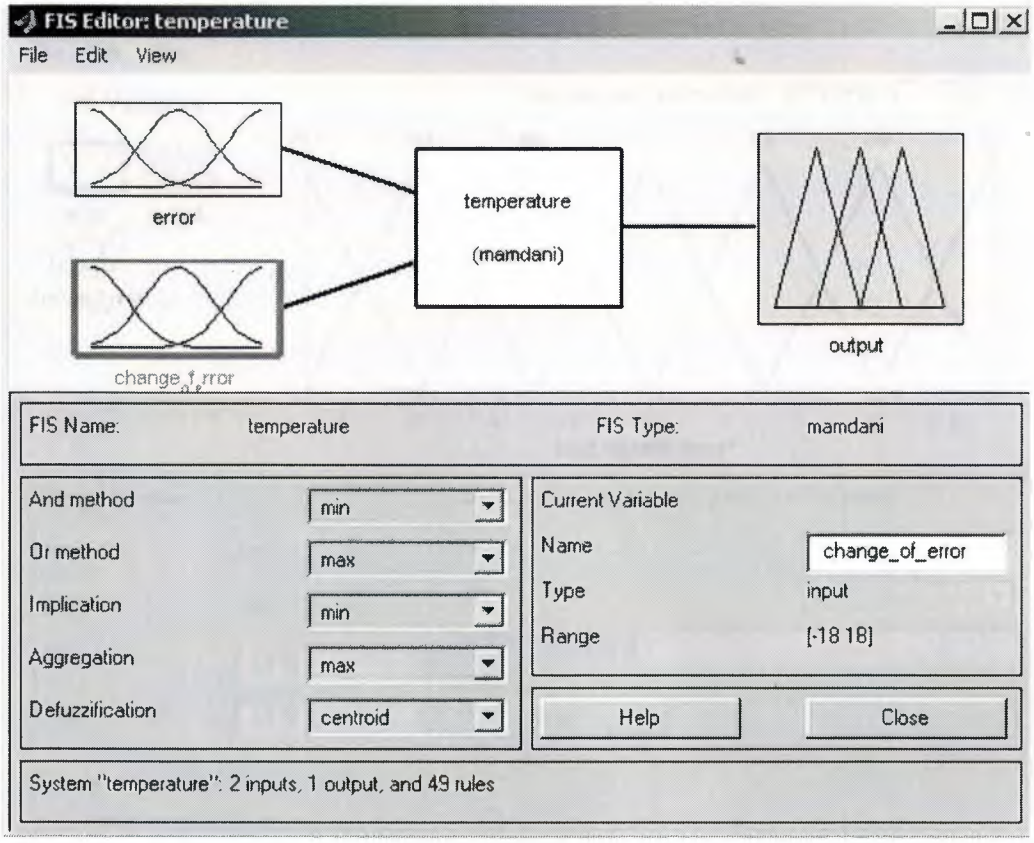


Figure 5.1. FIS Editor for the control of temperature of a heater.

Bellow the diagram is the name of the system and the type of inference used. Here we are using Mamdani-Type Inference. Bellow the name of the fuzzy inference system (temperature), on the left side of the Figure is the area that displays the name of either an input or output variable, its associated membership function type, and its range.

5.2.2 The Membership Function Editor

The Membership Function Editor (see Figure 5.2) shares some features with the FIS Editor. In fact, all of the five basic GUI tools have similar menu options, status lines, and Help and Close buttons. Membership Function Editor is the tool that displays and edits all of the membership functions associated with all of the input and output variables for the entire fuzzy inference system.

When we open the Membership Function Editor to work on a fuzzy inference system that does not already exist in the work space, there are not yet any membership functions associated with an inputs or an output variable for the FIS, select an FIS variable in this region by clicking on it.

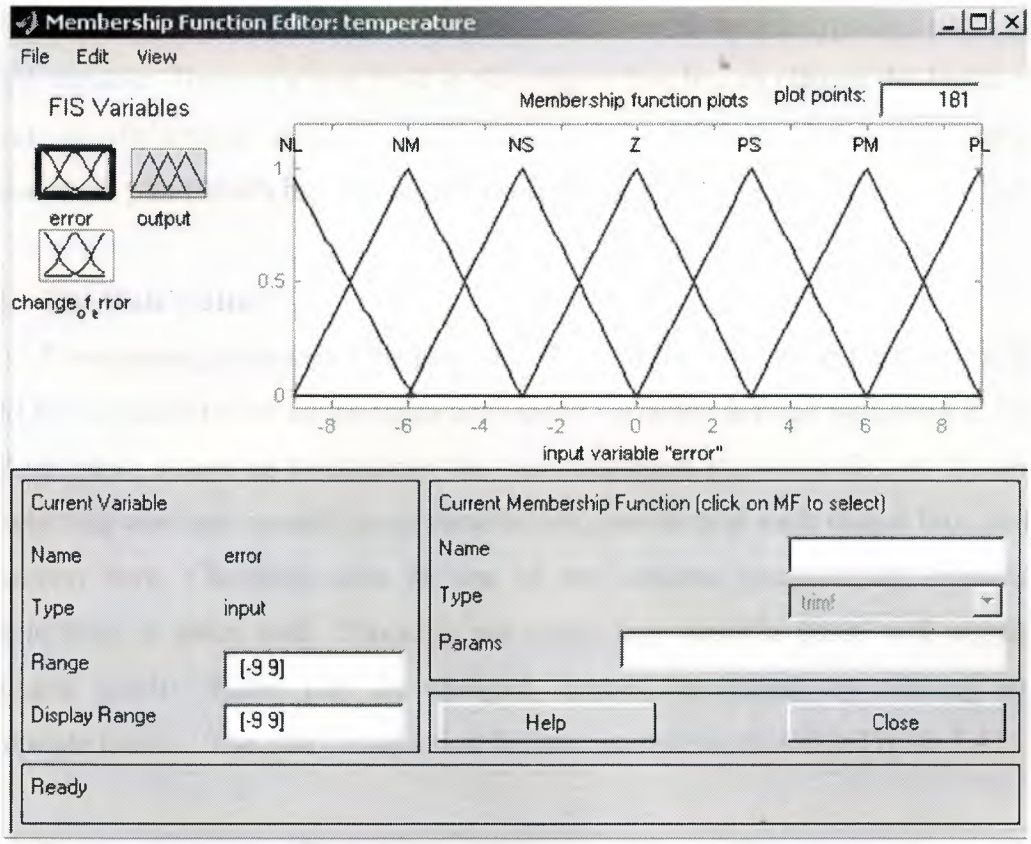


Figure 5.2. The Membership Function Editor.

Next select the Edit pull-down menu, and choose Add MFs. A new window will appear, which allows you to select both the type and a number of membership functions associated with the selected variable (see Figure 5.3), in our example we have 7 membership functions associated with each input and output (NL, NM, NS, Z, PS, PM, PL).

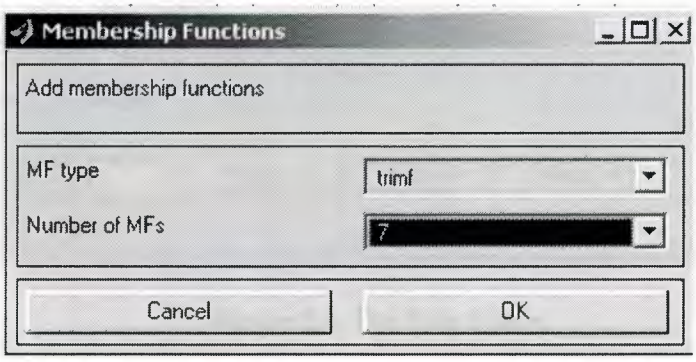


Figure 5.3. Adding membership function.

Bellow the Variable Plette is some information about the type and name of the current variable. There is a text field in this region that lets us change the limits of the current variable's range (universe of discourse) and another that lets you set the limits of the current plot (which has no real effect on the system).

5.2.3 The Rule Editor

Constructing rules using the graphical Rule Editor interface is fairly self-evident. Based on the descriptions of the input and output variables defined with the FIS Editor, the Rule Editor allows us to construct the rule statements automatically, by clicking on and selecting one item in each input variable box, one item in each output box, and one connection item. Choosing none as one of the variable qualities will exclude the variable from a given rule. Choosing not under any variable name will negate the associated quality. Rules may be changed, deleted, or added, by clicking on the appropriate button. The rule construction for our example is shown in Figure 5.4.

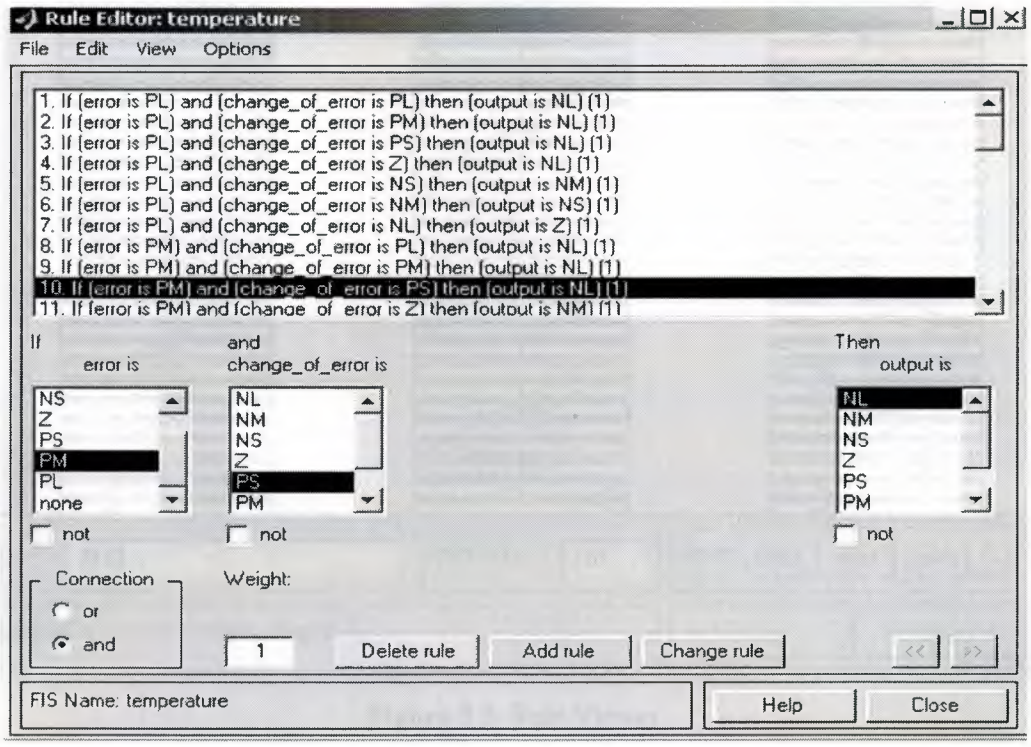


Figure 5.4. The Rule Editor.

The rules have been constructed using the Table 5.1. If we choose PM for error and PS for change of error, the output will be NL, so the rule will be looks like

If (error is PM) and (change of error is PS) then (output is NL) (1)

The number in the parentheses represents weights that can be applied to each rule if desired. You can specify the weights by typing in a desired number between zero and one under weight-setting.

5.2.4 The Rule Viewer

The Rule Viewer displays a roadmap of the whole fuzzy inference process. It's based on fuzzy inference diagram described in the previous section. In Figure 5.5 a single window with 49 small plots nested in it (Figure 5.5 shows the Rule Viewer).

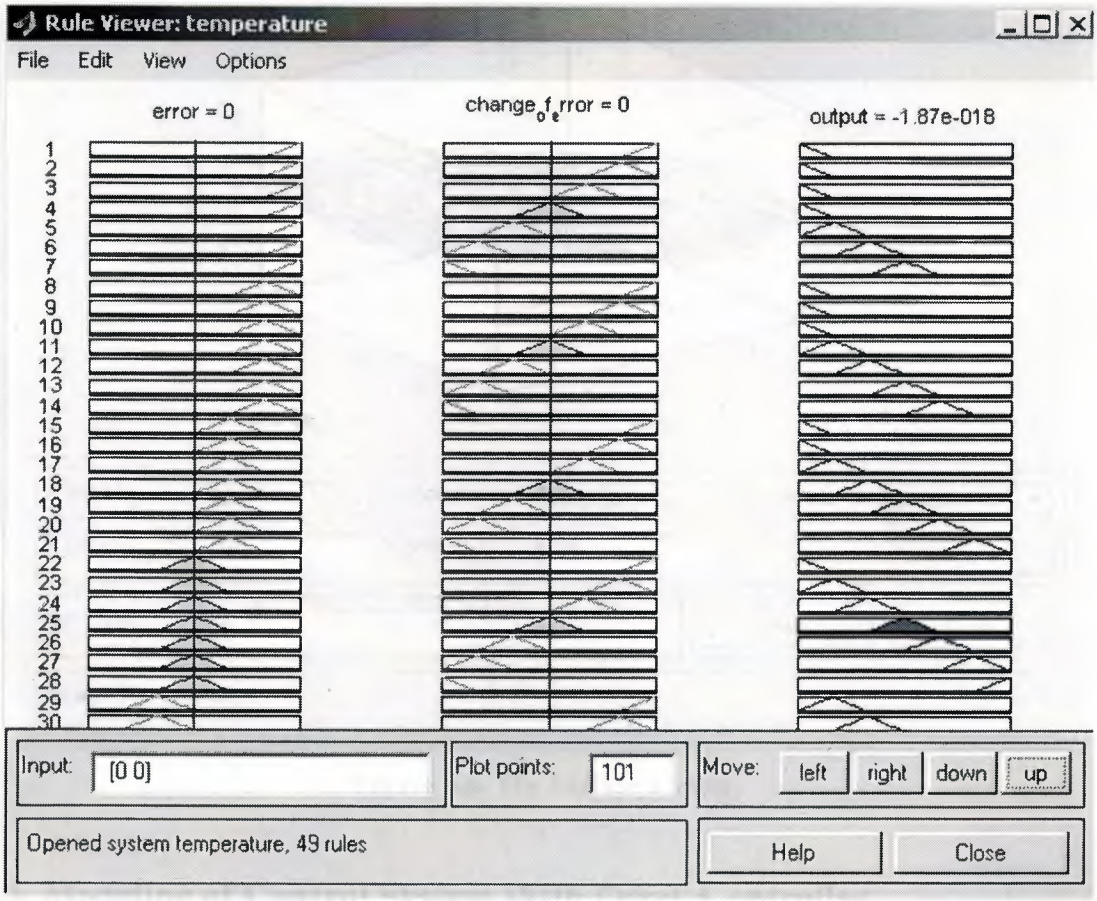


Figure 5.5. Rule Viewer.

The Rule Viewer allow us to interpret the entire fuzzy inference process at once. The Rule Viewer also shows how the shape of a certain membership functions influences the overall result. Since it plots every part of every rule, it can became

unwieldy for particular large systems, but, for a relatively small number of inputs and outputs, it performs well.

5.2.5 The Surface Viewer

Upon opening the Surface Viewer, we are presented with a two-dimensional curve that represents the mapping from service quality to temperature amount. The number of inputs and outputs is strongly affects the general result shape, since this two-input one-output system also work well, see Figure 5.6.

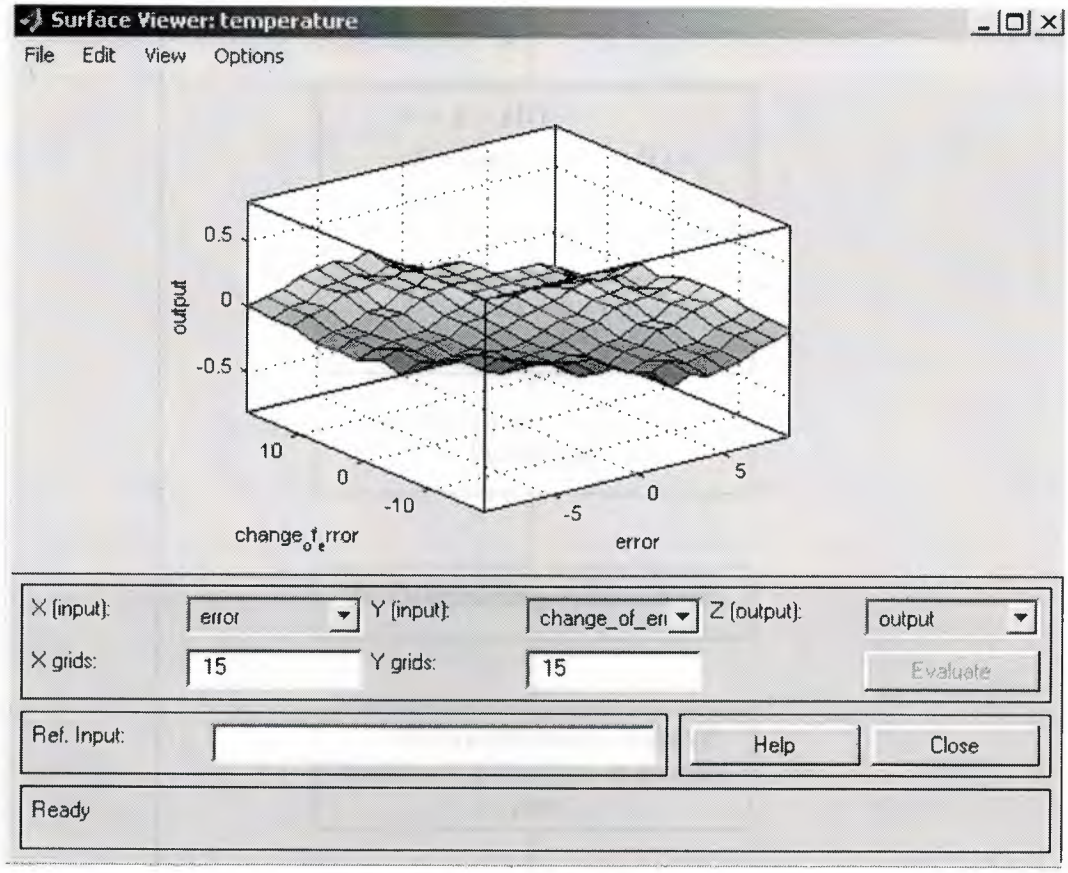
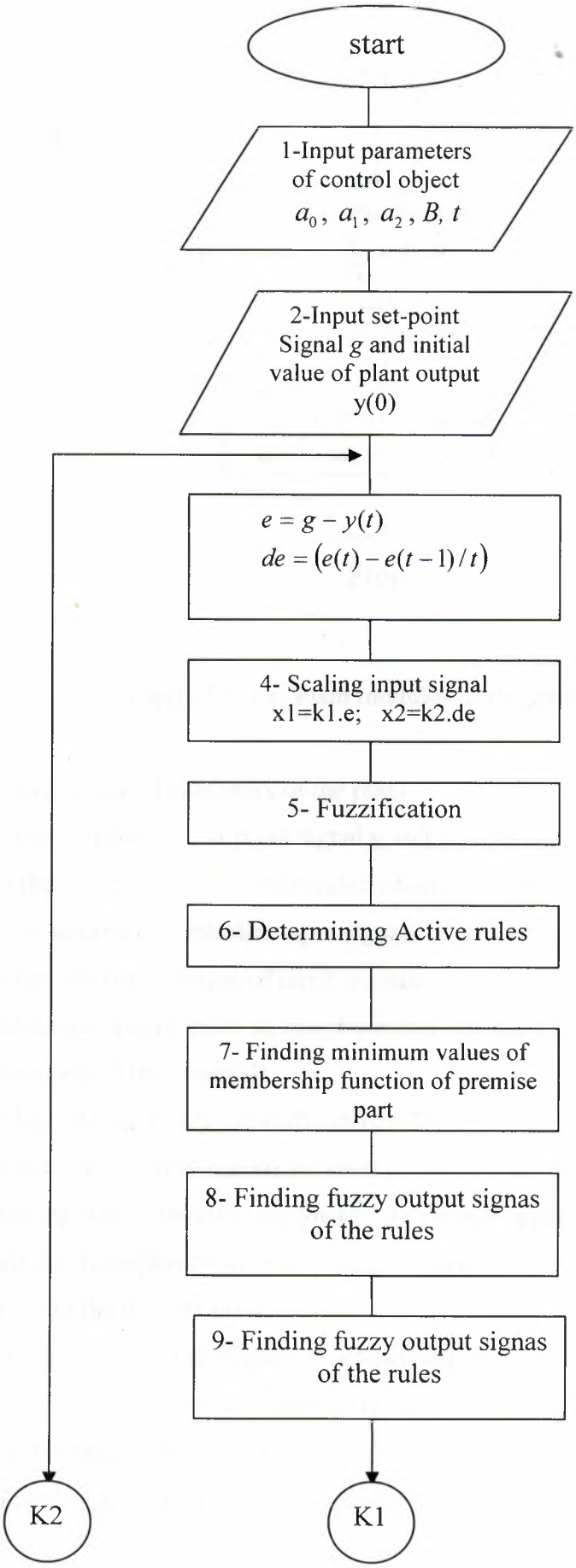


Figure 5.6. The Surface Viewer.

5.3 Modeling of Control System With Fuzzy Controller

The computer simulation of control system with PD-Like fuzzy controller have been carried out. In Figure 5.7 the flow-chart of the program of fuzzy control system is given.


start
1-Input parameters of control object a_0, a_1, a_2, B, t
2-Input set-point Signal g and initial value of plant output y(0)
$e = g - y(t)$
$de = (e(t) - e(t-1)/t)$
4- Scaling input signal x1=k1.e; x2=k2.de
5- Fuzzification
6- Determining Active rules
7- Finding minimum values of membership function of premise part
8- Finding fuzzy output signas of the rules
9- Finding fuzzy output signas of the rules
K2
K1

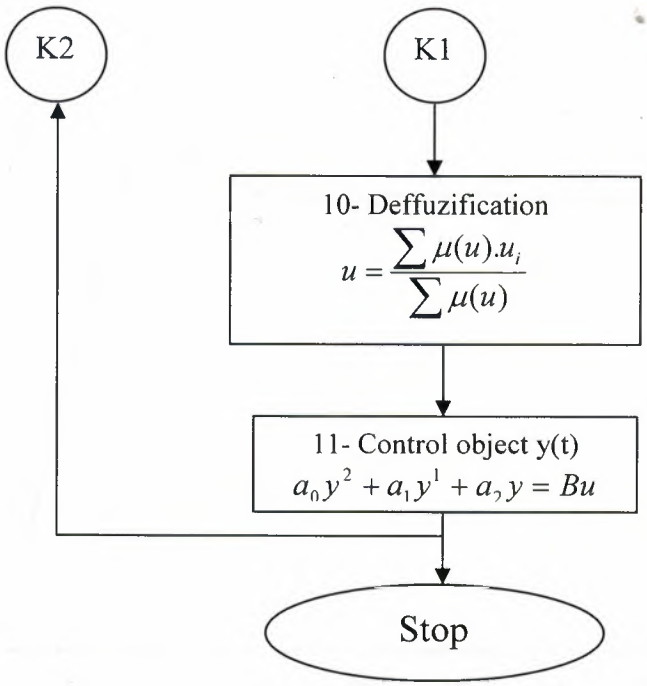


Figure 5.7. The main modules of the program.

Block 1 is used to enter parameters of the plant.

Block 2 is used to enter the set-point signal g and current value of plant output signal.

By Block 3 the values of error e and change of error e' are calculated.

In Block 4 the scaling of controller input signals is performed.

Block 5 realize the fuzzification of input signals.

Block 6 determine active rules in rule base and for each active rule it calculates the membership grade of input signals.

Block 7 realize min operation for each active rule.

By Block 8 the fuzzy output signals for each active rule are determined.

In Block 9 using union operation the total fuzzy output signal is determined.

In Block 10 the fuzzification of fuzzy output signal of controller is carried out. This signal is given to the input control.

In Block 11 the output signal of plant is determined.

As a result of computer simulation the fuzzy control system for controlling temperature of heater is developed.

In Figure 5.8 the time response characteristic of PD-like fuzzy control system is given.

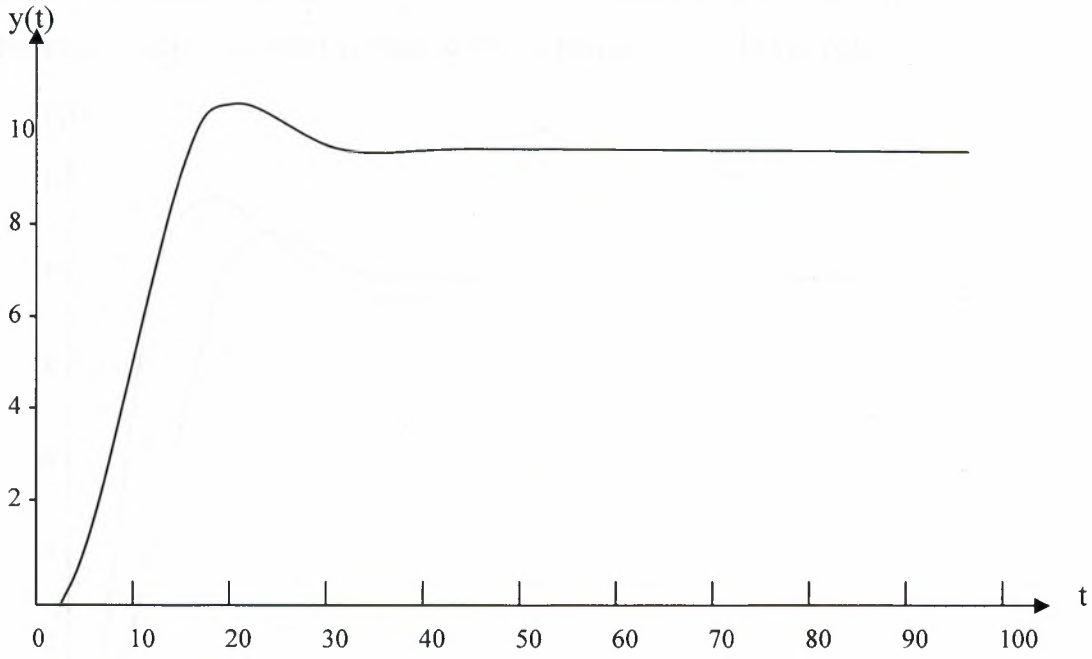


Figure 5.8. Time response characteristic of control system with fuzzy controller.

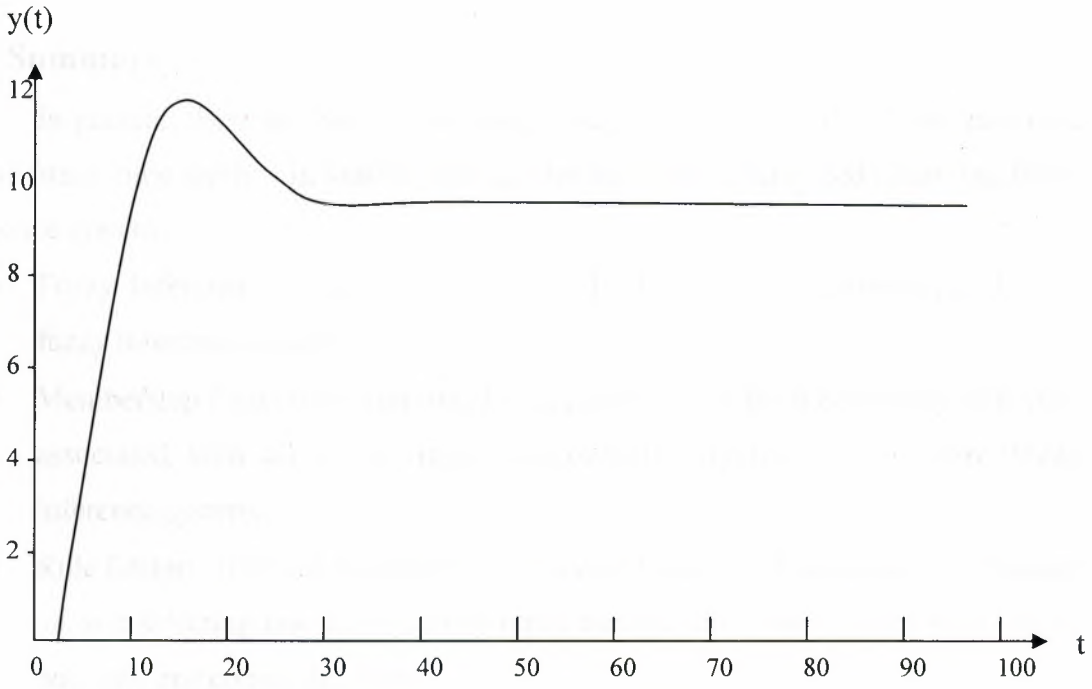


Figure 5.9. Time response characteristic of control system with PD controller.

Result of computer simulation of fuzzy control system is compared with the result of simulation of control system with PD-Like controller (Figure 5.9). In Figure 5.10 comparative results of control system based on fuzzy and PD-Like controllers is shown.

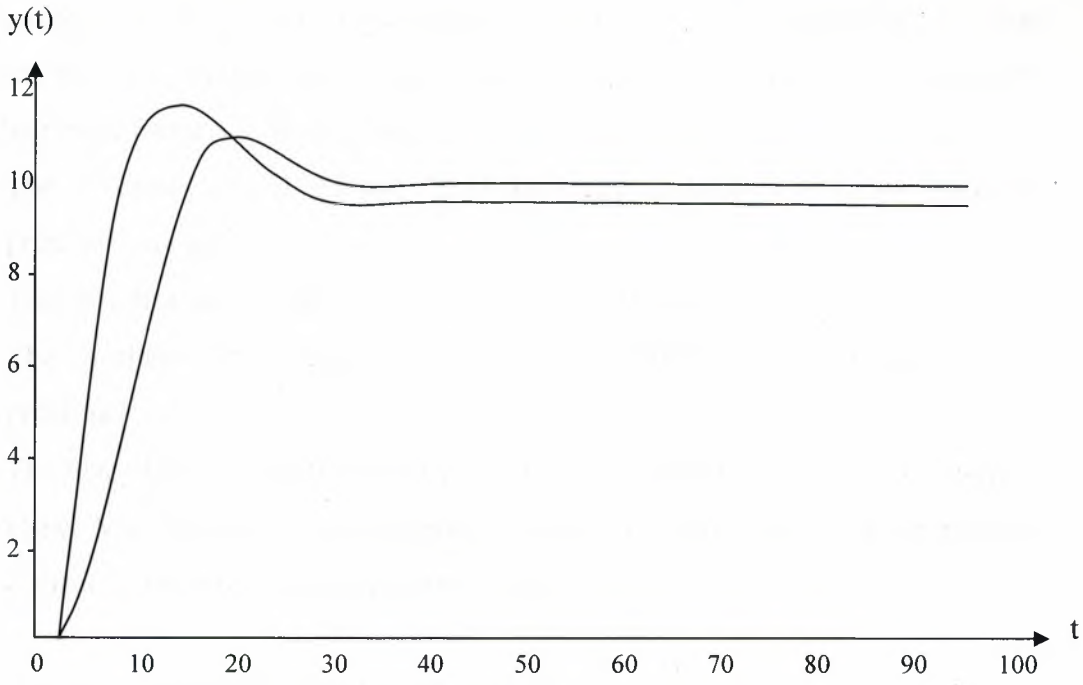


Figure 5.10. Comparative results of fuzzy PD-like and PD controllers.

5.4 Summary

In general, there are five primary graphical user interface (GUI) tools, provided by the fuzzy logic toolbox in Matlab package, for building, editing, and observing fuzzy inference systems:

- Fuzzy Inference System or FIS Editor: displays general information about a fuzzy inference system.
- Membership Function Editor: displays and edits all of the membership functions associated with all of the input and output variables for the entire fuzzy inference system.
- Rule Editor: allows us to construct the rule statements automatically, by clicking on and selecting one item in each input variable box, one item in each output box, and one connection item.
- Rule Viewer: displays a roadmap of the whole fuzzy inference process.
- Surface Viewer: a two-dimensional curve that represents the mapping from service quality to temperature amount.

CONCLUSION

The analysis of some industrial and non-industrial processes show, that they are characterized with uncertainty of environment, fuzziness of information. In these conditions the fuzzy system is the most effective mathematical tool for modeling and controlling these processes. In the thesis:

- The structures of PD, PI and PID-Like fuzzy controllers and their operation principles are given.
- The operation algorithms of fuzzy controllers are presented.
- The synthesis procedures for PD, PI and PID-Like fuzzy controllers are presented.
- The fuzzy PD-Like controller for controlling temperature of heater is developed.
- Using Matlab package the computer simulation of PD-Like fuzzy controller to control temperature of heater is performed.

The thesis consists of introduction, five chapters and conclusion.

Chapter One described the structure of the fuzzy system and the functions of it's main blocks. The structures of PD, PI and PID-Like fuzzy controllers and their operation principles are described.

Chapter Two presented the algorithms of fuzzy controllers. The linguistic variables, their fuzzy values and different fuzzification algorithms are described. The steps of inference engine mechanism are also described. Different types of fuzzy processing mechanisms are given in this chapter as well.

Chapter Three is devoted to the development of PD, PI and PID-Like fuzzy controllers for technological processes control. As a result, the fuzzy rule base controllers have been generated.

Chapter Four is described the development of fuzzy controller for control temperature of heater. The description of the processes is given. The realization of each block of fuzzy controller are described.

Chapter Five described the computer simulation of fuzzy system for control of temperature of heater by using Matlab package. The result of simulation is analyzed.

The thesis demonstrated how to design, develop and implement a fuzzy logic control applications and how to fine tune an existing fuzzy logic system to operate at its optimal point.

The result of simulation demonstrated the efficiency of the applied methodology.

REFERENCES

[1] Aliev R., Aliev F. and Babae M., Fuzzy Process Control and Knowledge Engineering in Petrochemical and Robotic Manufacturing, Verlarge Tüv Rheinland Gmbh, Köln Germany (1991).

[2] Driankov D., Hellendoorn H. and Reinfrank M. *An Introduction to Fuzzy Control*, second edn, Springer-Verlag, Berlin (1996).

[3] Hill G., Horstkotte E. and Teichrow J. Fuzzy-C *Development System- Users Manual*, Togai Infralogic, 30 Corporate Park, Irvine, CA 92714, USA (1990).

[4] Holmblad L. P. and Stergaard J. J., Control of a Cement Kiln By Fuzzy Logic, in Gupta and Sanchez (eds), *Fuzzy Information and Decision Processes*, North-Holland, Amsterdam, pp. 389–399. (Reprint in: FLS Review No 67, FLS Automation A/S, Høffdingsvej 77, DK-2500 Valby, Copenhagen, Denmark) (1982).

[5] Zadeh L. A., Fuzzy set, Theoretic International & Linguistic Hedges, Cybernetics vol.12,4-34.1972.

[6] Zadeh L. A., Fuzzy Sets, Information and Control, vol 8,338-353,1976.

[7] R. H. Abiyev, B. A. Abbasov. Adaptive Fuzzy Controller for Technological Processes Control, high Military Sea School, "N1, Baku, (Russia)(1998).

[8] IEC, Programmable Controllers: Part 7 fuzzy control programming, *Technical Report IEC 1131*, International Electrotechnical Commission. (Draft.) (1996).

[9] Lee C. C., Fuzzy Logic in Control Systems: Fuzzy Logic Controller, *IEEE, Trans. System, Man & Cybernetics* 20 (2): 404–435, (1990).

[10] MIT & CITE *Litareture and Products Database* , MIT GmbH/ELITE, Promenade 9, D-52076 Aachen, Germany (1995).

[11] Mizumoto M., Realization of PID Controls By Fuzzy Control Methods, in IEEE (ed.), *First Int.Conf.On. Fuzzy Systems*, number 92CH3073-4, The Institute of Electrical and Electronics Engineers, Inc, San Diego, pp.709-715, (1992).

[12] Passino K. M. and Yurkovich S., *Fuzzy Control*, Addison Wesley Longman, Inc, Menlo Park, CA, USA (1998).

[13] Pedrycz W., Fuzzy Controller Fuzzy System, second edn, Wiley and Sons, New York. Qiao W. and Mizumoto, M., PID Type Fuzzy Controller and Parameters Adaptive Method, *Fuzzy Sets and Systems* 78:23-35, (1996)

[14] Siler W. and Ying H., Fuzzy control theory: The Linear Case, *Fuzzy Sets and Systems* 33: 275–290, (1989).

[15] Takagi T. and Sugeno M., Fuzzy Identification of Systems and its Applications to Modeling and Control, IEEE *Trans.Systems,Man & Cybernetics* 15(1): 116–132, (1985).

[16] Kerimov K. M., Abiyev R. H., Melikov T. G., Fuzzy System for Determination of Oil-gas Fields Productivity/Geophysics News in Azerbaijan. Baku, N:1,pp.13-15, (1999).

[17] Kerimov K. M., Abiyev R. H., Melikov T. G., Novruzov E.S., Use of Illegible Logic to Determine Oil and Gas Bearing Intervals in Beds/Geophysics News in Azerbaijan. Baku, N:2, pp.33-35, (1999).

[18] Kerimov K. M., Abiyev R. H., Melikov T. G., Determination of Oil and Gas Field Productivity in The Condition of Insufficiency and Fuzziness of Information, Second International Symposium on Mathematical and Computational Applications, Baku, september 1-3, (1999).

[19] E. H. Mamdani and S. Assilian. An Experiment in Linguistic Synthesis with a Fuzzy logic Controller// Internat. J. Man-Machine Stud. 7/1-13, (1975).

[20] Bektaş Ş., Mamedov F. and Khashman A., The Graduate Studies: A Complete Guide, Near East University press, Nicosia (2001).

Appendix 1

```
[System]
Name='temperature'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=49
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='error'
Range=[-9 9]
NumMFs=7
MF1='NL':'trimf',[-14.85 -9.15 -5.85]
MF2='NM':'trimf',[-9 -6 -3]
MF3='NS':'trimf',[-6 -3 0]
MF4='Z':'trimf',[-3 5.551e-017 3]
MF5='PS':'trimf',[0 3 6]
MF6='PM':'trimf',[3 6 9]
MF7='PL':'trimf',[6 9 15]

[Input2]
Name='change_of_error'
Range=[-18 18]
NumMFs=7
MF1='NL':'trimf',[-30 -18 -12]
MF2='NM':'trimf',[-18 -12 -6.001]
MF3='NS':'trimf',[-12 -6.001 0]
```

MF4='Z':'trimf',[-6.001 0 6.001]

MF5='PS':'trimf',[0 6.001 12]

MF6='PM':'trimf',[6.001 12 18]

MF7='PL':'trimf',[12 18 30]

[Output1]

Name='output'

Range=[-0.9 0.9]

NumMFs=7

MF1='NL':'trimf',[-1.2 -0.905 -0.605]

MF2='NM':'trimf',[-0.9 -0.6 -0.3]

MF3='NS':'trimf',[-0.6 -0.3 0]

MF4='Z':'trimf',[-0.3 0 0.3]

MF5='PS':'trimf',[0 0.3 0.6]

MF6='PM':'trimf',[0.3 0.6 0.9]

MF7='PL':'trimf',[0.6 0.9 1.2]

[Rules]

7 7, 1 (1) : 1

7 6, 1 (1) : 1

7 5, 1 (1) : 1

7 4, 1 (1) : 1

7 3, 2 (1) : 1

7 2, 3 (1) : 1

7 1, 4 (1) : 1

6 7, 1 (1) : 1

6 6, 1 (1) : 1

6 5, 1 (1) : 1

6 4, 2 (1) : 1

6 3, 3 (1) : 1

6 2, 4 (1) : 1

6 1, 5 (1) : 1

5 7, 1 (1) : 1

5 6, 1 (1) : 1

5 5, 2 (1) : 1

5 4, 3 (1) : 1

5 3, 4 (1) : 1

5 2, 5 (1) : 1

5 1, 6 (1) : 1

4 7, 1 (1) : 1

4 6, 2 (1) : 1

4 5, 3 (1) : 1

4 4, 4 (1) : 1

4 3, 5 (1) : 1

4 2, 6 (1) : 1

4 1, 7 (1) : 1

3 7, 2 (1) : 1

3 6, 3 (1) : 1

3 5, 4 (1) : 1

3 4, 5 (1) : 1

3 3, 6 (1) : 1

3 2, 7 (1) : 1

3 1, 7 (1) : 1

2 7, 3 (1) : 1

2 6, 4 (1) : 1

2 5, 5 (1) : 1

2 4, 6 (1) : 1

2 3, 7 (1) : 1

2 2, 7 (1) : 1

2 1, 7 (1) : 1

1 7, 4 (1) : 1

1 6, 5 (1) : 1

1 5, 6 (1) : 1

1 4, 7 (1) : 1

1 3, 7 (1) : 1

1 2, 7 (1) : 1

1 1, 7 (1) : 1

Appendix 2

```
g=input ('Enter set-point signal  :  ');
l=3;
a0=0.42;
a1=0.72;
a2=1;
b=60;
t=0.15;
aa1=(2*a0+a1-t)/(a0+a1*t+a2*t*t);
aa2=-a0/(a0+a1*t+a2*t*t);
bb1=(b*t*t)/ (a0+a1*t+a2*t*t);
bb2=0;
x(1)=0;
x(2)=0;
x(3)=0;
e(1)=0;
e(2)=0;
e(3)=0;
out(1)=0;  out(2)=0;  out(3)=0;
k0=0.02;
k1=0.01;
while(1<100)
e(l)=g-x(1);
ee(l)=(e(l)-e(l-1))/t;
e1=k0*e(l);
e2=k1*ee(l);
output=e1+e2;
out(l)=output*bb1+aa1*out(l-1)+aa2*out(l-2);
l= l+1;
end;
plot(out)
```