NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Fuzzy Control System Development

Graduation Project
COM- 400

Student: Ayyob A.M Dahdal (971334)

Supervisor: Assit. Prof. Dr Rhib Abiyev

Lefkosa - 2001

ACKNOWLEDGEMENT

First, I want to thank my parents for their endless support and their love for me, which without of them I could not be able to reach my aim to be an engineer, so thank you dad and thank you mama for every thing you did for me. And of course I did not forget my brothers' encourage for me to reach this. Thank you Mahmoud, thank you Marwan and thanks for my cosine Diya'a for his wishes me the best.

I also want to thank my friends in NEU: Mahmoud, Yaseer, Maher, Alaa, and special thanks to my friends Wael and Mohammad Imran.

Special thanks to **Assit.Prof.Dr Rahib Abiyev** for being my supervisor, for giving me a lot of his time and for his gaudiness to do this project and to solve any problem faced me while I am doing this project.

Special thanks to our chairman Assit.Prof.Dr Adnan Khashman and also I want to thank my advisor Mr.Tayseer Alshanableh.

ABSTRACT

In industry same technological processes are characterized by unpredictable and hard formulized factors, uncertainty and fuzziness of information. In this situation deterministic models is not enough adequately describe those processes and at the results control on their base begin difficult. In these conditions it is advisable to use fuzzy technology, which provide independency of the model to disturbance and adequacy of the model.

The aim of thesis is the development of the fuzzy control system for technological processes. To solve this problem the state of application problem of fuzzy control systems in real industry is considered. The structure and operation principle of fuzzy control system are described. Different fuzzy processing mechanisms are analyzed.

The development of fuzzy control system is performed. The one of main problem in synthesis of fuzzy system is the development fuzzy knowledge base. The synthesis of the fuzzy knowledge base for PD-like fuzzy controller is carried out. Processing mechanisms of fuzzy rules are described. By using max-min fuzzy processing of Zade the inference mechanism of fuzzy system is realized.

The fuzzy controller for control temperature of heater is modeled.

The simulation and obtained results satisfy the efficiency of application of fuzzy technology to industry.

TABLE OF CONTENTS

INTRODUCTION

Presently large class of industrial processes is 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, which 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 not 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. A control engineer on the base of the interviews then formulated such a control algorithm 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 approximate that is very important for modeling unknown objects. Fuzzy technology has such characteristics as interpretability, transparency, plausibility, gradualist, modeling, reasoning, imprecision

tolerance. In the thesis the development of fuzzy system for technological processes control is considered. The thesis consists of introduction, 4 chapters and conclusion.

Chapter 1 describes the state of application problems of fuzzy technology to solve control problems and the mile stone achievements to the problem.

Chapter 2 describes the architecture of fuzzy systems for technological processes control. The structure of fuzzy systems, the functions of its main blocks are described. The structures of PD-like, PI-like and PID-like fuzzy controller are described.

Chapter 3 presents the operations in fuzzy system. The description of linguistic rules, their characteristics, fuzzy rules firing, different types of fuzzy processing mechanisms are given. The representation of max-min processing of Zade is described.

Chapter 4 describes the development of fuzzy system for technological process control. Using fuzzy desired time response characteristic of the system, fuzzy model of the technological processes the synthesis of fuzzy control system is performed. Using these the development of the synthesis procedures and simulation of neural control system are performed.

Chapter 5 describes the simulation of the fuzzy system to control temperature of heater. The results of simulation of PD-like fuzzy control system are described. The efficiency of its application is analyzed. Conclusion presents the obtained important results and contributions in the thesis.

CHAPTER 1

STATE OF APPLICATION PROBLEM OF FUZZY SYSTEM FOR TECHNOLOGICAL PROCESSES CONTROL

Many decision-making and problem-solving tasks are too complex to be understood quantitatively, however, people succeed by using knowledge that is imprecise rather than precise. Fuzzy set theory, originally introduced by Lotfi Zadeh in the 1960's, resembles human reasoning in its use of approximate information and uncertainty to generate decisions. It was specifically designed to mathematically represent uncertainty and vagueness and provide formalized tools for dealing with the imprecision intrinsic to many problems. By contrast, traditional computing demands precision down to each bit. Since knowledge can be expressed in a more natural by using fuzzy sets, many engineering and decision problems can be greatly simplified. Fuzzy set theory implements classes or groupings of data with boundaries that are not sharply defined (i.e., fuzzy). Any methodology or theory implementing "crisp" definitions such as classical set theory, arithmetic, and programming, may be "fuzzified" by generalizing the concept of a crisp set to a fuzzy set with blurred boundaries. The benefit of extending crisp theory and analysis methods to fuzzy techniques is the strength in solving real-world problems, which inevitably entail some degree of imprecision and noise in the variables and parameters measured and processed for the application. Accordingly, linguistic variables are a critical aspect of some fuzzy logic applications, where general terms such a "large," "medium," and "small" are each used to capture a range of numerical values. While similar to conventional quantization, fuzzy logic allows these stratified sets to overlap (e.g., a 85 kilogram man may be classified in both the "large" and "medium" categories, with varying degrees of belonging or membership to each group). Fuzzy set theory encompasses fuzzy logic, fuzzy arithmetic, fuzzy mathematical programming, fuzzy topology, fuzzy graph theory, and fuzzy data analysis, though the term fuzzy logic is often used to describe all of these. Fuzzy logic emerged into the mainstream of information technology in the late 1980's and early 1990's. Fuzzy logic is a departure from classical Boolean logic in that it implements soft linguistic variables on a continuous range of truth values which allows intermediate values to be defined between conventional binary. It can often be considered a superset

of Boolean or "crisp logic" in the way fuzzy set theory is a superset of conventional set theory. Since fuzzy logic can handle approximate information in a systematic way, it is ideal for controlling nonlinear systems and for modeling complex systems where an inexact model exists or systems where ambiguity or vagueness is common. A typical fuzzy system consists of a rule base, membership functions, and an inference procedure. Today, fuzzy logic is found in a variety of control applications including chemical process control, manufacturing, and in such consumer products as washing machines, video cameras, and automobiles. Fuzzy logic is nowadays applied in almost all sectors of industry and science in the western world, especially in the field of control and in pattern recognition. The fuzzy logic has been until now widely accepted by the industrial world in Japan, Europe and USA as a valuable theory to rapidly develop prototypes, especially in control. Because of the ability of fuzzy logic to model imprecise and subjective notions makes it possible to mimic humanlike reasoning. This ability of systems in which fuzzy logic is included makes it possible to combine very different types of information and use these to control real life processes. According to many authors there are three main reasons for the present popularity and therefore many different applications of fuzzy logic control in industry. These reasons are:

1. Fuzzy logic can be easily combined with existing methods.
2. Process control often has to deal with intrinsic uncertainties, due to change in parameters and/or difficult and indirect measurements, these uncertainties can be handled by controller based on fuzzy logic, and not by human operators as in case in which classical controllers are used.
3. Fuzzy logic is suitable for rapid prototyping, It is reasonable to use fuzzy logic in control systems in the following cases:

    - If we are dealing with multi-variable systems.
    - In the presence of strong noise in the system and if a wide control range is demanded,
    - In the case of presence of nonlinear and/or dead time component in the system,
    - In the case of absence of the good mathematical model of the system.

It is known that in the design of any kind of controller three different methodologies can be followed:

1. Using the physical model of the system can control the simple processes. Unfortunately, in solving realistic control problems we deal very rarely with processes of that kind.
2. The process identification model is a second possible approach. This approach has disadvantages due to problems connected with identifications routines. especially in the case of nonlinear and non-minimum phase systems.
3. Fuzzy Logic can be exploit to model the control behavior of an expert operator.

In this case it is not necessary to have an in-depth understanding of the interrelations of all parameters or availability of the process for experiments. Inside the power system. a complex and multi-variable system by its nature, many different control circuits are contained. Many of them are complex and multi-variable themselves. Many of them are also nonlinear and / or nonminimum–phase systems. In addition, all systems inside the power system have to be robust and reliable. In some cases in power systems is also almost impossible to ensure knowledge of all parameters, e.g. in secondary control circuits. One of the cases in which fuzzy control can be used is load-frequency control. The fundamental purpose of automatic secondary load-frequency control in power systems is to maintain a constant equilibrium between production and consumption. In case there is such equilibrium, based on the power system nature, it is easy to conclude that:

1. Load exchange with neighboring power systems is within agreed values.
2. Frequency value is equal to the referable value.

Since power systems are complex and multi-variable, to realize automatic secondary load–frequency control is not always an easy task. Due to complexity and multi-variable conditions of the power system, classic methods of automatic control do not give good results and therefore do not represent good enough solutions. The facts that all state variables are well-known are aggravated by constant changes of system configuration while it functions and it represents a problem when classic control methods are considered to be used. Limitations and problems that appear when classic regulators are used may be surpassed if either adaptive control methods or procedures of fuzzy control are applied. These problems are even more transparent in the case of secondary voltage control. Because the fuzzy control is nonlinear by its nature it could be interesting t

explore possible applications in primary turbine control. An efficient corner-matching algorithm is developed to minimize the amount of calculation. To reduce the amount of calculation, all available information from a corner detector is used to make model. This information has uncertainties due to discretization noise and geometric distortion, and this is represented by fuzzy rule base, which can represent and handle the uncertainties. From fuzzy inference procedure, a matched segment list is extracted, and resulted segment list is used to calculate the transformation between object of model and scene. In order to reduce the fuzzy rule set, an overlapping cost to minimize the matched segment list is introduced. Also an auto-tuning scheme of the fuzzy rule base is developed to find out the uncertainties of features from recognized results automatically. To show the effectiveness of the developed algorithm, experiments are conducted for images of real electronic components. Incorporate learning mechanisms into electro-mechanical drives permits design systems with self-learning capabilities and produces more autonomous processes with some intelligence degree. The [5] presents a control system that uses a fuzzy learning methodology to design an inverse-model compensation controller. The controller shows generalization and learning capabilities to compensate non-linear terms that affect the system dynamics. To investigate the controller, an experimental system composed by an electro-hydraulic actuator is used. The system dynamic is marked by a dominant non-linear characteristic constituted by a dead-zone and hysteresys effects. The [5] shows experimental results describing the real-time controller capability in compensate the nonlinearities of the actuator and learning to improve its performance in trajectory following. One of the major problems of a proportional plus integral plus derivative (PID) type fuzzy control system is the requirement of a 3-dimenisional rule-base. This complicates the design and increases the real-time computational efforts. Another major problem of such an integral fuzzy controller is the excessive noise encountered in implementations. In [6] is developed a direct implementation oriented fuzzy PI and PID control architecture, whose designs are optimized and automated by genetic algorithm based evolutionary computing techniques. Using these architectures, a 1-D reduction of the rule-base is achieved. The proposed fuzzy control approach is illustrated through both fuzzy PI and fuzzy PID control of an asymmetrical nonlinear liquid-level regulation system. Comparisons are made with traditional fuzzy control architectures, showing superior performances in both simulations and implementations of the proposed scheme. Fuzzy logic is a powerful problem-solving methodology with a myriad of applications in

embedded control and information processing. Fuzzy provides a remarkably simple way to draw definite conclusions from vague, ambiguous or imprecise information. In a sense, fuzzy logic resembles human decision making with its ability to work from approximate data and find precise solutions. Unlike classical logic, which requires a deep understanding of a system, exact equations, and precise numeric values, Fuzzy logic incorporates an alternative way of thinking, which allows modeling complex systems using a higher level of abstraction originating from our knowledge and experience. Fuzzy Logic allows expressing this knowledge with subjective concepts such as very hot, bright red, and a long time, which are mapped into exact numeric ranges. Fuzzy Logic has been gaining increasing acceptance during the past few years. There are over two thousand commercially available products using Fuzzy Logic, ranging from washing machines to high-speed trains. Nearly every application can potentially realize some of the benefits of Fuzzy Logic, such as performance, simplicity, lower cost, and productivity. Fuzzy Logic has been found to be very suitable for embedded control applications. Several manufacturers in the automotive industry are using fuzzy technology to improve quality and reduce development time. In aerospace, fuzzy enables very complex real time problems to be tackled using a simple approach. In consumer electronics, fuzzy improves time to market and helps reduce costs. In manufacturing, fuzzy is proven to be invaluable in increasing equipment efficiency and diagnosing malfunctions. Fuzzy sets and logic must be viewed as a formal mathematical theory for the representation of uncertainty. Uncertainty is crucial for the management of real systems: if you had to park your car precisely in one place, it would not be possible. Instead, you work within, say, 10 cm tolerances. The presence of uncertainty is the price you pay for handling a complex system. Nevertheless, fuzzy logic is a mathematical formalism, and a membership grade is a precise number.

CHAPTER 2

THE STRUCTURES AND OPERATIONS OF FUZZY CONTROL SYSTEM

The primary goal of control engineering is to distill and apply knowledge about how to control a process so that the resulting control system will reliably and safely achieve high- performance operation. Fuzzy logic provides a methodology for representing and implementing our knowledge about how best to control a process.

2.1 The Structure of Fuzzy Control System:

A block diagram of a Fuzzy Control System is shown in Figure (2.1). The fuzzy controller is 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 interface, which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.
4. A defuzzification interface, which converts the conclusions of the inference mechanism into actual inputs for the process.

Consider each of the components of the fuzzy controller for a simple problem of balancing an inverted pendulum on a cart, as shown in Figure (2.1). Here, (y) denotes the angle that the pendulum makes with the vertical (in radians), (l) is the half-pendulum length (in meters), and (u) is the force input that moves the cart (in Newtons). (r) denotes the desired angular position of the pendulum. The goal is to balance the pendulum in the upright position (i.e., $r = 0$) when it initially starts with some nonzero angle off the vertical (i.e., $y \neq 0$). This is a very simple and academic nonlinear control problem, and many good techniques already exist for its solution..

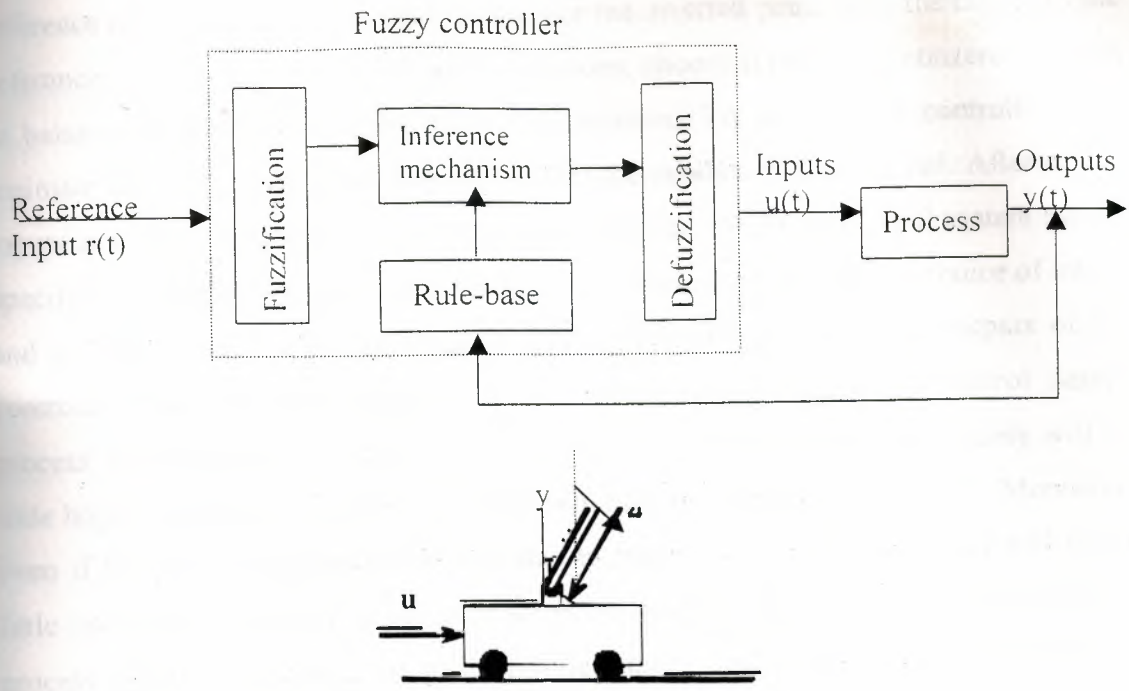


Fig (2.1) Inverted pendulum on a cart.

The fuzzy controller is to be designed to automate how a human expert who is successful at this task would control the system. First, the expert tells us (the designers of the fuzzy controller) what information she or he will use as inputs to the decision-making process. Suppose that for the inverted pendulum, the expert says that she or he will use:

$$e(t)=r(t)-y(t)$$

and

$$de(t)/dt.$$

As the variables on which to base decisions. Certainly, there are many other choices (e.g., the integral of the error (e) could also be used) but this choice makes good intuitive sense. Next, identify the controlled variable. For the inverted pendulum, it is allowed to control only the force that moves the cart, so the choice here is simple. For more complex applications, the choice of the inputs to the controller and outputs of the controller (inputs to the plant) can be more difficult. If the designer believes that proper information is not available for making control decisions, he or she may have to invest in another sensor that can provide a measurement of another system variable. Alternatively, the designer may implement some filtering or other processing of the plant outputs. Once the fuzzy controller inputs and outputs are chosen, what the

reference inputs are should be determined. For the inverted pendulum, the choice of the reference input (r =0) is clear. In some situations, choose (r) as some nonzero constant to balance the pendulum in the off-vertical position. To do this, the controller must maintain the cart at a constant velocity so that the pendulum will not fall. After all the inputs and outputs are defined for the fuzzy controller, the fuzzy control system can be specified. The fuzzy control system for the inverted pendulum, with our choice of inputs and outputs, is shown in Figure (2.2). The choice of the inputs and outputs of the controller places certain constraints on the remainder of the fuzzy control design process. If the proper information is not provided to the fuzzy controller, there will be little hope for being able to design a good rule-base or inference mechanism. Moreover, even if the proper information is available to make control decisions, this will be of little use if the controller is not able to properly affect the process variables via the process inputs. The choice of the controller inputs and outputs is a fundamentally important part of the control design process.

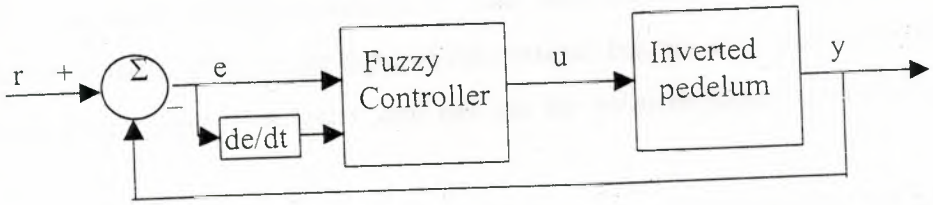


Fig. (2.2) Fuzzy controller for an inverted pendulum on a cart.

2.1.1 Putting Control Knowledge into Rule-Bases

The linguistic description provided by the expert can generally be broken into several parts. There will be "linguistic variables" that describe each of the time-varying fuzzy controller inputs and outputs. For the inverted pendulum,

"error" describes e(t)

"change-in-error" describes de(t)/dt;

"force" describes u (t),

The linguistic descriptions as short as possible (e.g., using "e(t)" as the linguistic variable for e(t)), yet accurate enough so that they adequately represent the variables. Suppose for the pendulum example that "error," "change-in-error," and "force" take on the following values:

"neglarge", "negsmall", "zero", "possmall", "poslarge",

Using "negsmall" as an abbreviation for "negative small in size" and so on for the other variables. Here "neg" is negative, "pos" is positive. Every linguistic value nicely represent that the varible has a numeric quality. The linguistic variables and values provide a language for the expert to express her or his ideas about the control decision-making process in the context of the framework established by our choice of fuzzy controller inputs and outputs. Recall that for the inverted pendulum (r = 0) and (e = r – y) so that:

$$(e = - y)$$

and

$$(de/dt = -dy/dt),$$

since (dr/dt = 0). For the inverted pendulum each of the following statements quantifies a different configuration of the pendulum:

- The statement "error is poslarge" can represent the situation where the pendulum is at a significant angle to the left of the vertical.
- The statement "error is negsmall" can represent the situation where the pendulum is just slightly to the right of the vertical, but not too close to the vertical to justify quantifying it as "zero" and not too far away to justify quantifying it as "neglarge."
- The statement "error is zero" can represent the situation where the pendulum is very near the vertical position (a linguistic quantification is not precise, hence we are willing to accept any value of the error around (e(t) = 0) as being quantified linguistically by "zero" since this can be considered a better quantification than "possmall" or "negsmali").
- The statement "error is poslarge and change-in-error is possmall" can represent the situation where the pendulum is to the left of the vertical and, since (dy/dt<0), the pendulum is moving away from the upright position (note that in this case the pendulum is moving counterclockwise).
- The statement "error is negsmall and change-in-error is possmall" can represent the situation where the pendulum is slightly to the right of the vertical and. since (dy/dt<0), the pendulum is moving toward the upright position (note that in this case the pendulum is also moving counterclockwise).

Now, using the above linguistic quantification to specify a set of rules that captures the expert's knowledge about how to control the plant. In particular, for the inverted pendulum in the three positions shown in Figure (2.3), the following rules are applied:

1. "If error is neglarge and change-in-error is neglarge then force is poslarge".

This rule quantifies the situation in Figure (2.3a) where the pendulum has a large positive angle and is moving clockwise; hence it is clear that we should apply a strong positive force (to the right) so that we can try to start the pendulum moving in the proper direction.

2. "If error is zero and change-in-error is possmall then force is negsmall".

This rule quantifies the situation in Figure (2.3b) where the pendulum has nearly a zero angle with the vertical (a linguistic quantification of zero does not imply that e(t) = 0 exactly) and is moving counterclockwise; hence we should apply a small negative force (to the left) to counteract the movement so that it moves toward zero (a positive force could result in the pendulum overshooting the desired position).

3. "If error is poslarge and change-in-error is negsmall then force is negsmall".

This rule quantifies the situation in Figure (2.3c) where the pendulum is far to the left of the vertical and is moving clockwise: hence we should apply a small negative force (to the left) to assist the movement, but not a big one since the pendulum is already moving in the proper direction.

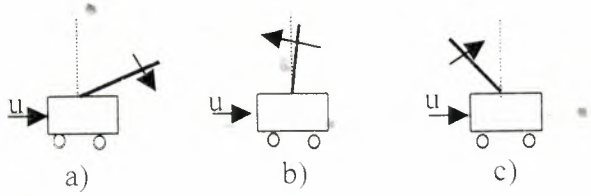


Fig (2.3) Inverted pendulum in various position.

Each of the three rules listed above is a "linguistic rule" since it is formed solely from linguistic variables and values. Since linguistic values are not precise representations of the underlying quantities that they describe, linguistic rules are not precise either. They are simply abstract ideas about how to achieve good control that could mean somewhat

different things to different people. The general form of the linguistic rules listed above is

If premise Then consequent,

From the three rules listed above, the premises (which are sometimes called "antecedents") are associated with the fuzzy controller inputs and are on the left-hand-side of the rules. The consequents (sometimes called "actions") are associated with the fuzzy controller outputs and are on the right-hand-side of the rules. For the pendulum problem, with two inputs and five linguistic values for each of these, there are at most ($5^2 = 25$) possible rules. A tabular representation of one possible set of rules for the inverted pendulum is shown in Table (2.1). The body of the table lists the linguistic-numeric consequents of the rules, and the left column and top row of the table contain the linguistic-numeric premise terms. Then, for instance, the (2, -1) position has a "-1" ("negsmall") in the body of the table and represents the rule

If error is poslarge and change-in-error is negsmall Then force is negsmall.

Table (2.1) represents abstract knowledge that the expert has about how to control the pendulum given the error and its derivative as inputs.

"force" u		"Change-in-error" e'				
		-2	-1	0	1	2
"error" e	-2	2	2	2	1	0
	-1	2	2	1	0	-1
	0	2	1	0	-1	-2
	1	1	0	-1	-2	-2
	2	0	-1	-2	-2	-2

Table (2.1) Rule table for the inverted pendulum.

2.1.2 Fuzzy Quantification

The meaning of linguistic descriptions may automate, in the fuzzy controller, the control rules specified by the expert.

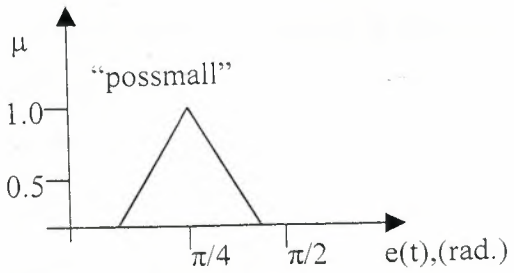
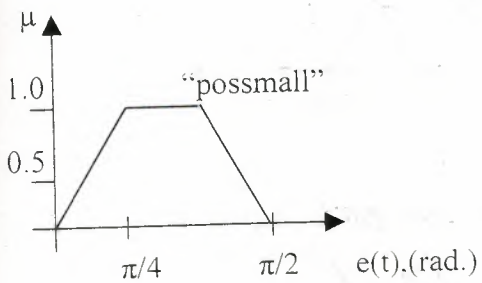
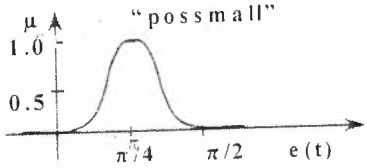


Fig (2.4) Membership function for linguistic value "possmall".

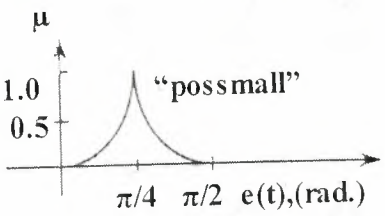
The membership function quantifies, in a continuous manner, whether values of e(t) belong to (are members of) the set of values that are "possmall," and hence it quantifies the meaning of the linguistic statement "error is possmall." This is why it is called a membership function. It is important to recognize that the membership function in Fig (2.4) is only one possible definition of the meaning of "error is possmall".



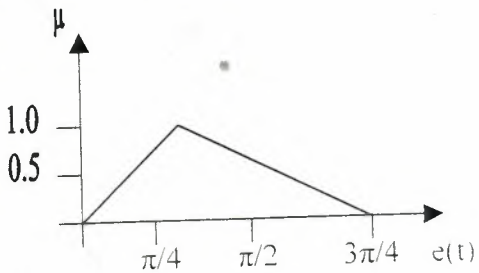
a)Trapezoid



b)Gaussian



c)Sharp peak



d) Skewed triangle

Fig (2.5) A few membership function choices for representing "error is possmall".

Depending on the application and the designer (expert), many different choices of membership functions are possible. A "crisp" (as contrasted to "fuzzy") quantification of "possmall" can also be specified. but via the membership function shown in Figure(2.6). This membership function is simply an alternative representation for the interval on the real line. and it indicates that this interval of numbers represents "possmall." Clearly, this characterization of crisp sets is simply another way to represent a normal interval (set) of real numbers.

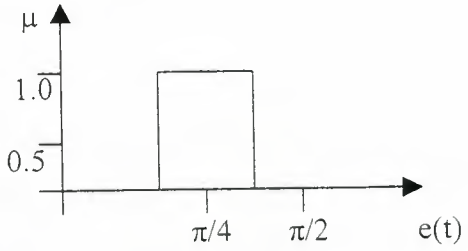


Fig (2.6) membership function for a crisp.

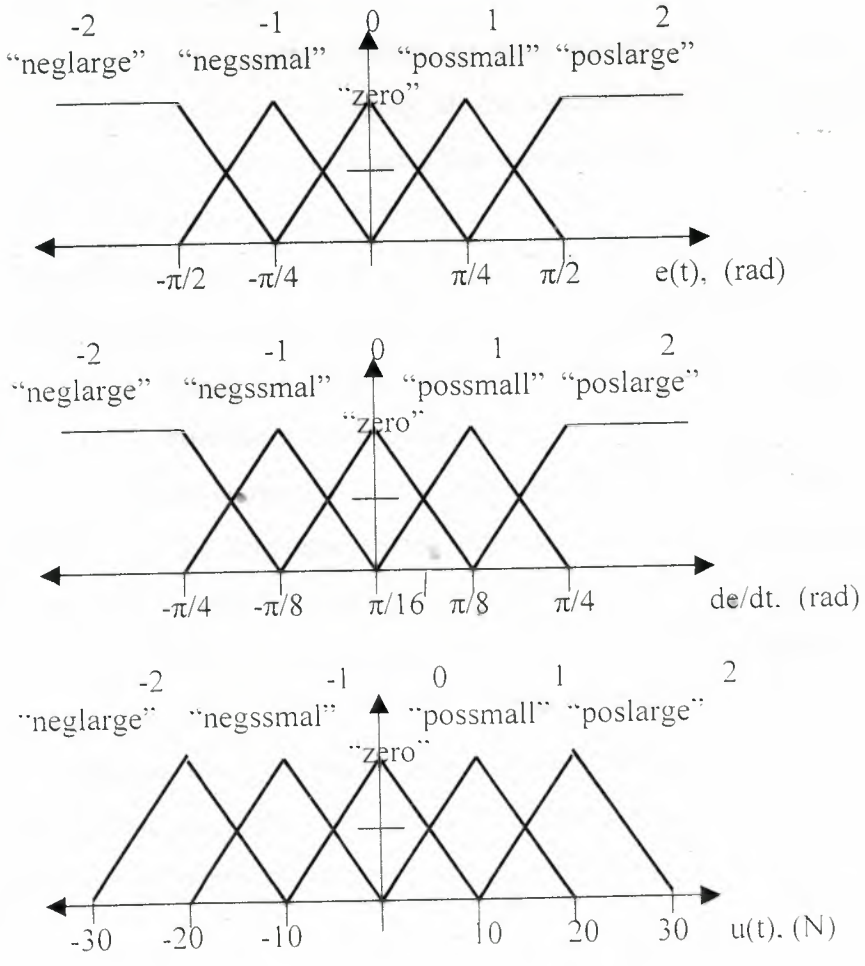


Fig (2.7) membership functions for an inverted pendulum on a cart.

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, "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 (2.1), so, the description of the simple inverted pendulum is completed.

2.1.3 Matching

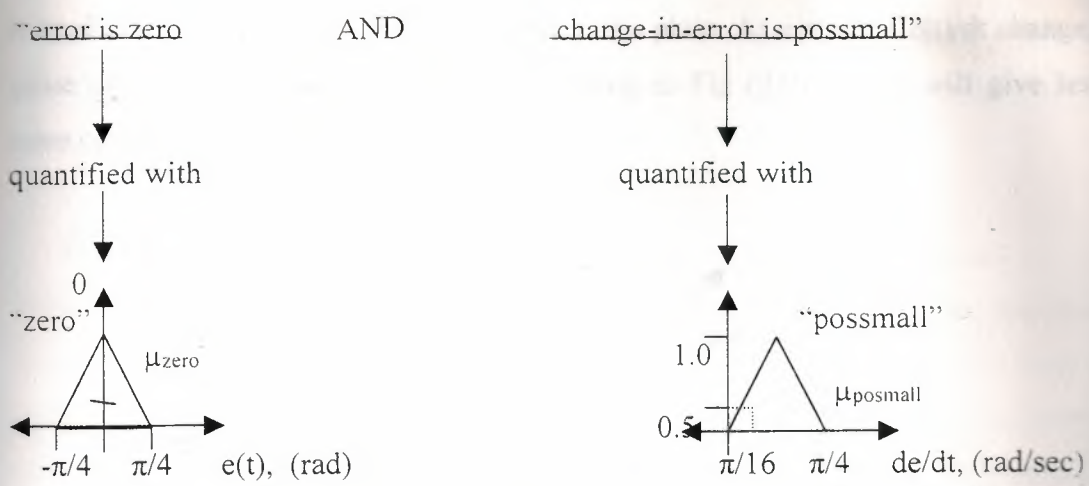
The inference process generally involves two steps:

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 applies, and typically we will more strongly take into account the recommendations of rules that we are more certain 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.

To perform inference we must first quantify each of the rules with fuzzy logic. To do this we first quantify the meaning of the premises of the rules that are composed of several terms, each of which involves a fuzzy controller input. Consider Fig (2.8), where we list two terms from the premise of the rule

If error is zero and change-in-error is possmall Then force is negsmall

Above, the meaning of the linguistic terms "error is zero" and "change-in- error is possmall" via the membership functions shown in Fig(2.7) had been quantified. Now seek to quantify the linguistic premise "error is zero AND change-in-error is possmall.". Hence, the main item to focus on is how to quantify the logical "AND" operation that combines the meaning of two linguistic terms. Use standard Boolean logic to combine these linguistic terms; since it has quantified them more precisely with fuzzy sets (i.e., the membership functions), so using these.



Fig(2.8). Membership functions of premise terms.

To see how to quantify the "AND" operation, begin by supposing that e(t)=n/8 and de(t)/dt=n/32, so that using Fig (2.7) (or Fig (2.8):

$$\mu_{zero}(e(t)) = 0.5$$

and

$$\mu_{possmall}(de/dt) = 0.25$$

It will need to denote this certainty 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 there are no more certain about the conjunction of two statements than they are about the individual terms that make them up (note that $0 \leq \mu_{premise} \leq 1$ for either case). The 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, that gives a multidimensional membership function $\mu_{premise}(e(t), de(t)/dt)$ that is a function of e(t) and de(t)/dt for each rule. For our example, if by choosing the minimum operation to represent the "AND" in the premise. then that will give the multidimensional membership function $\mu_{premise}(e(t).de(t)/dt)$.

Notice that the values for e(t) and de(t)/dt, the value of the premise certainty $\mu_{premise}(e(t).de(t)/dt)$ represents how certain it is that the rule:

If error is zero and change-in-error is possmall Then force is negsmall

is applicable for specifying the force input to the plant. As e(t) and de(t)/dt change, the value of $\mu_{premise}(e(t),de(t)/dt)$ changes according to Fig (2.9), and it will give less or more certain of the applicability of this rule.

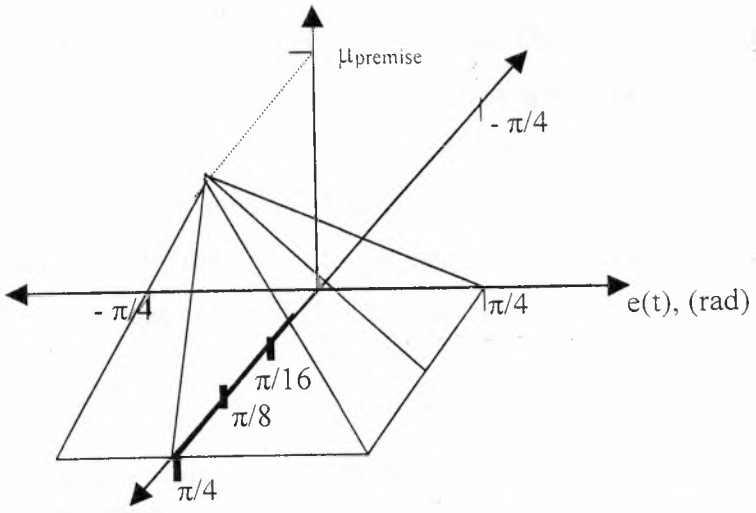


Fig (2.9) Membership function of the premise for a single rule.

Determining the applicability of each rule is called "matching." The 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 inverted pendulum example. Suppose that

$$e(t) = 0$$

and

$$de(t)/dt = n/8 - n/32(= 0.294)$$

Fig (2.10) 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.that is $\mu_{zero}(de(t)/dt)=0.25$ and $\mu_{possmall}(de(t)/dt) = 0.75$ and that all the other membership functions are off.

"error is zero"

"change-in-error is zero"

"change-in-error is possmall"

This implies that rules that have the premise terms are on (all other rules have $\mu_{premise}(e(t),de(t)/dt)) = 0$. So, which rules are these? Using Table (2.1), the rules that are on are the following:

1. If error is zero and change-in-error is zero Then force is zero
2. If error is zero and change-in-error is possmall Then force is negsmall.

Note that since for the pendulum example, which gives at most two membership functions over-lapping, it will never give more than four rules on at one time (this concept generalizes to many inputs). Actually, for this system it will either give 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=n/8 so that only rule (2) above is on.

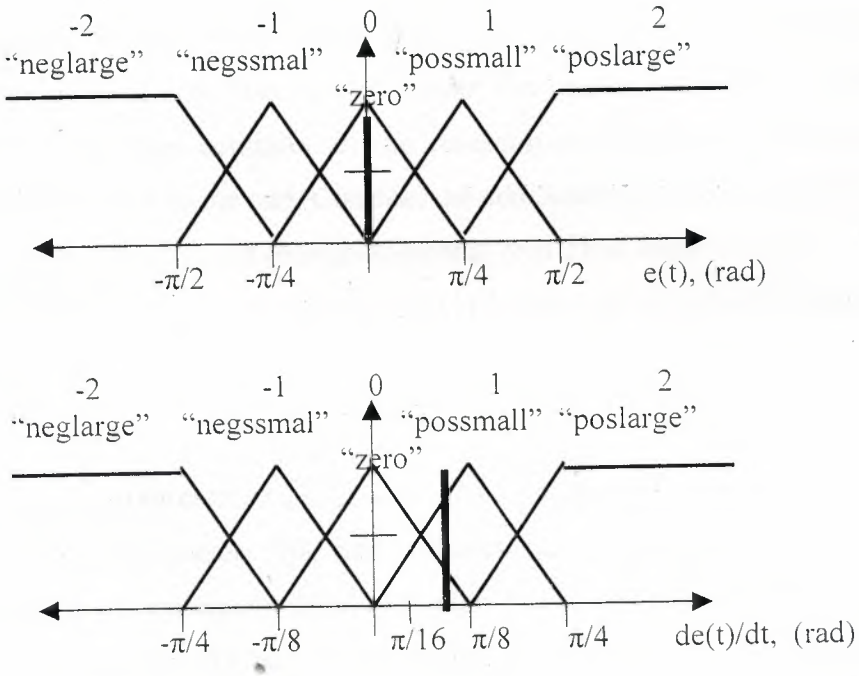


Fig (2.10) input member functions with input values.

It is useful to consider pictorially which rules are on. Consider Table (2.2), which is a copy of Table (2.1) with boxes drawn around the consequents of the rules that are on (notice that these are the same two rules listed above). Notice that since e(t) = 0(e(t) is directly in the middle between the membership functions for "possmall" and "negsmall") both these membership functions are off. If it is perturbed e(t) slightly positive (negative), then it would give the two rules below (above) the two highlighted ones on also.

"force" u		"Change-in-error" e'				
		-2	-1	0	1	2
"error" e	-2	2	2	2	1	0
	-1	2	2	1	0	-1
	0	2	1	0	-1	-2
	1	1	0	-1	-2	-2
	2	0	-1	-2	-2	-2

Table (2.2) rule table for the inverted pendulum with rules
Those are "ON" highlighted.

2.1.4 Inference Step

Consider how to determine which conclusions should be reached when the rules that are on are applied to deciding what the force input to the cart carrying the inverted pendulum should be. To do this, first consider the recommendations of each rule independently. Then later combine all the recommendations from all the rules to determine the force input to the cart. Consider the conclusion reached by the rule

If error is zero and change-in-error is zero Then force is zero

which for convenience it will refer to as "rule (1)." Using the minimum to represent the premise, it gives:

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

(the notation $\mu_{premise1}$ represents $\mu_{premise}$ for rule (1)) so that, 0.25 certain 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 "force is zero" (this makes sense, for here the pendulum is balanced, so any force should not be applied since this would tend to move the pendulum away from the vertical). The membership function for this consequent is shown in Fig (2.11a). The membership function for the conclusion reached by rule (1), which denoted by μ_1, is shown in Fig (2.11b) 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 there would be no more certain about our 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). See that $\mu_1(u)$ is in general a time-varying function that quantifies how certain rule (1) is that the force input u should take on certain values. It is most certain that the force input should lie in a region around zero (see Fig.(2.11b), 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 the linguistic meaning of the rule is considered. 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 to be able to picture how the shape of the implied fuzzy set changes as the rule's premise certainty changes over time.

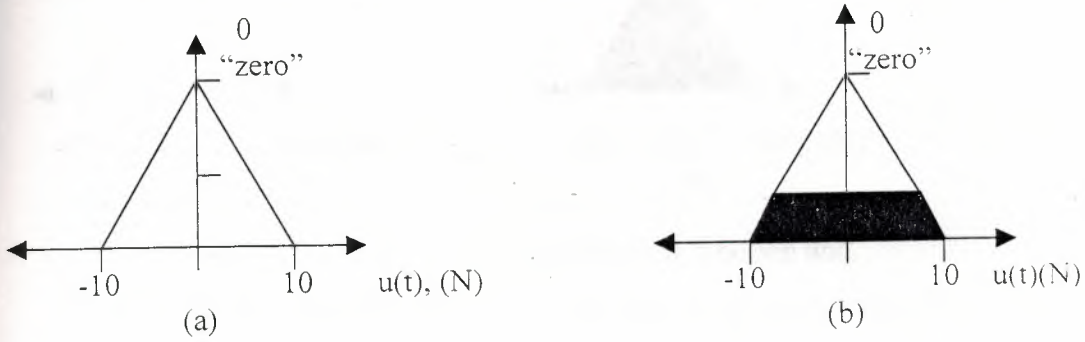


Fig (2.11). (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_1(u)$ for rule (1).

Then. consider the conclusion reached by the other rule that is on.

If error is zero and change-in-error is possmall Then force is negsmall

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

$$\mu_{premise2}(u) = \min\{0.75.1\} = 0.75$$

Now. 0.75 certain that this rule applies to the current situation. For rule (2) the consequent is "force is negsmall" (this makes sense. for here the pendulum is perfectly balanced but is moving in the counterclockwise direction with a small velocity). The membership function for this consequent is shown in Fig (2.12a). The membership function for the conclusion reached by rule (2). denoted by $\mu_2(u)$. is shown in Fig (2.12b) (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. This makes sense since if the pendulum has some counterclockwise velocity then it will need to apply a negative force (i.e., one to the left). As rule (2) has a premise membership function that has higher certainty than for rule (1), and it would be more certain of the conclusion reached by rule (2).

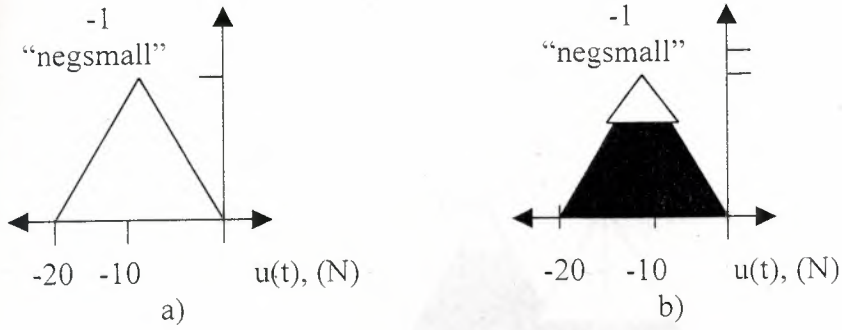


Fig (2.12) (a) Consequent membership function and,
(b) Implied fuzzy set with membership function $\mu_{(2)}(u)$ for rule (2).

This completes the operations of the inference mechanism in Fig (2.1). 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).

2.1.5 Converting Decisions into Actions

Consider the defuzzification operation, which is the final component of the fuzzy controller shown in Fig (2.1). 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. To understand defuzzification, it is best to first draw all the implied fuzzy sets on one axis as shown in Fig (2.13). We want to find the one output, which we denote 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. 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 (i.e., where it reaches its peak for our example) of the consequent of rule (i). For our example we have

$$b_1=0$$

and

$$b_2=-10$$

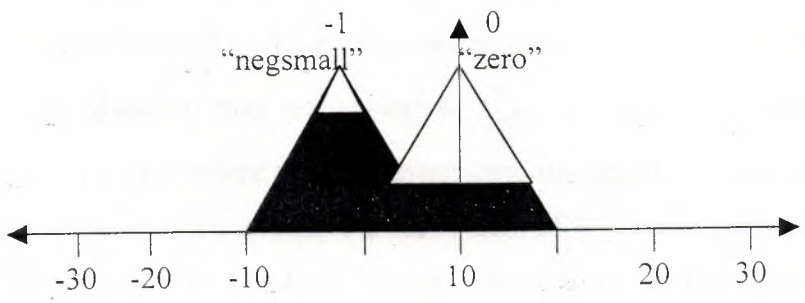


Fig (2.13) Implied fuzzy sets.

as shown in Fig (2.13). Let

$$\int \mu_{(i)}$$

denote the area under the membership function $\mu_{(i)}$. The COG method computes u^{crisp} to be

$$u^{crisp} = \frac{\Sigma_i\, b_i \int \mu_{(i)}}{\Sigma_i \int \mu_{(i)}} \quad (1)$$

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 (1) is important to note:

1. Practically, we cannot have 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 (e.g., we did not allow the saturated membership functions at the outermost edges as we had for the inputs shown in Fig (2.7).
2. You must be careful to define the input and output membership functions so that the sum in the denominator of Equation (1) is not equal to zero no matter what the inputs to the fuzzy controller are. Essentially, this means that we must have some sort of conclusion for all possible control situations we may encounter.
3. While at first glance it may not appear so, $\int \mu_i$ is easy to compute for our example. For the case where we have 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 Fig (2.11 and Fig (2.14)) is equal to

$$w(h - h^2/2)$$

Given this, the computations needed to compute u^{crisp} are not too significant. We see that 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. As another example, it is interesting to consider how to compute, by hand, the operations that the fuzzy controller takes when we use the product to represent the implication or the "center-average" defuzzification method. First, consider the use of the product. Consider Fig (2.15), where we have drawn the output membership functions for

"negsmall" and "zero" as dotted lines. The implied fuzzy set from rule (1) is given by the membership function

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

shown in Fig (2.15) as the shaded triangle; and the implied fuzzy set for rule (2) is given by the membership function

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

shown in Fig (2.15) as the dark triangle. Notice that computation of the COG is easy since we can use (1 wh) as the area for a triangle with base width w and height h. When we useproduct to represent the implication, we obtain

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

which also makes sense

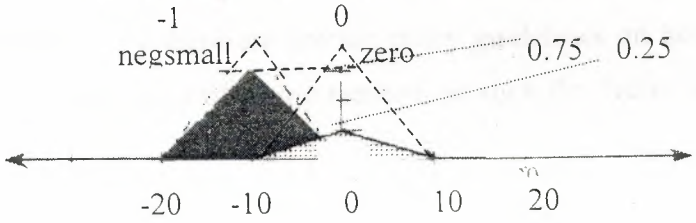


Fig (2.15) Implied fuzzy sets when the product is used to represent the implication.

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

$$u^{crisp} = \frac{\sum_i b_i \mu_{premise_i}}{\sum_i \mu_{premise_i}} \qquad (2)$$

where to compute $\mu_{premise_i}$ we use, for example, minimum.We call it the "center-average" method since Equation (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 $\mu_{premise_i}$ .This is a valid replacement since the area of the implied fuzzy set is generally proportional to $\mu_{premise_i}$ since $\mu_{premise_i}$ is used to chop the top off (minimum) or scale (product) the

triangular output membership function when COG is used for our example. For the above example, we have

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

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 we discuss 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.

2.1.6 Graphical Depiction of Fuzzy Decision Making

For convenience, let us summarize the procedure that the fuzzy controller uses to compute its outputs given its inputs in Fig (2.16). Here, we use the minimum operator to represent the "AND" in the premise and the implication and COG defuzzification.The reader is advised to study each step in this diagram to gain a fuller understanding of the operation of the fuzzy controller. To do this, 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.

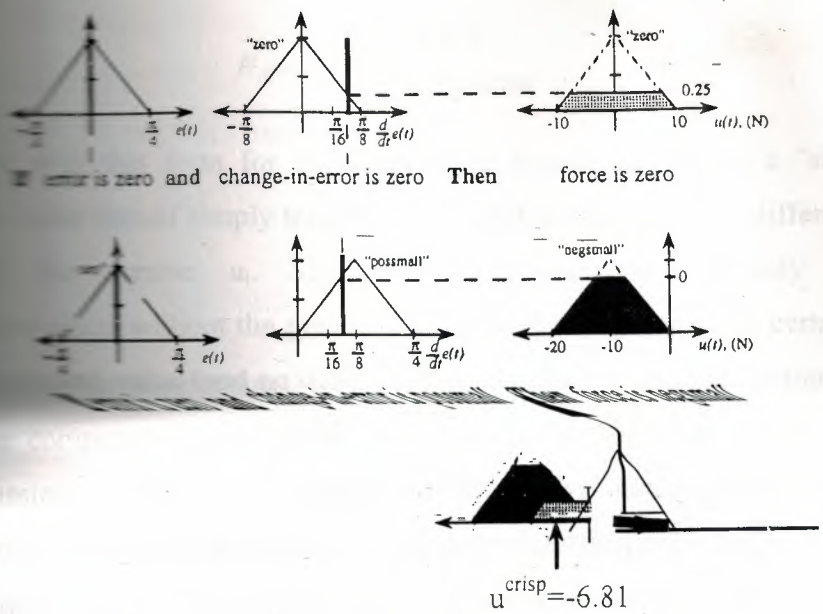


Fig (2.16) Graphical representation of fuzzy controller operations.

This completes the description of the operation of a simple fuzzy controller. You will find that while we will treat the fully general fuzzy controller in the next section, there will be little that is conceptually different &om this simple example. We simply show how to handle the case where there are more inputs and outputs and show a fuller range of choices that you can make for the various components of the fuzzy controller.

2.2 Fuzzification

Fuzzy sets are used to quantify the information in the rule-base, and the inference mechanism operates on fuzzy sets 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 $\hat{A}_i^{fuz}$ 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

$$F(u_i) = \hat{A}_i^{fuz}.$$

Quite often "singleton fuzzification" is used. which produces a fuzzy set $\hat{A}_i^{fuz} \in U_i^*$ 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 reader should simply think of the singleton fuzzy set as 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

1. They add computational complexity to the inference process,
2. 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 Figure1 can be ignored since this process is so simple. For now, the reader should simply think of 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. For example, if e(t) = n/4 and de(t)/dt = n/16, the fuzzification process amounts to finding the values of the input membership functions for these. In this case $\mu_{possmall}=1$ (with all others zero) and $\mu_{zero}(de(t)/dt)= \mu_{possmall}(de(t)/dt)=0.5$.

2.3 Inference Mechanism

The inference mechanism has two basic tasks:

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

$$\hat{A}_1^{fuz}, \hat{A}_2^{fuz}, \ldots, \hat{A}_n^{fuz}$$

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

Step 1: Combine Inputs with Rule Premises: The first step in matching involves finding fuzzy sets $\hat{A}_1^j$, $\hat{A}_2^k$,. . ., $\hat{A}_n^l$, with membership functio

$$\mu_{\hat{A}_1^j}(u_1) = \mu_{A_1^j}(u_1) * \mu_{\hat{A}_1^{fuz}}(u_1)$$

$$\mu_{\hat{A}_2^k}(u_2) = \mu_{A_2^k}(u_2) * \mu_{\hat{A}_2^{fuz}}(u_2)$$

$$\vdots$$

$$\mu_{\hat{A}_n^l}(u_n) = \mu_{A_n^l}(u_n) * \mu_{\hat{A}_n^{fuz}}(u_n)$$

(for all j, k, . . . , l) that combine the fuzzy sets from fuzzification with the fuzzy sets used in each of the teims 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., $\mu_{\hat{A}_1^j}(\hat{u}_1) = \mu_{A_1^j}(\hat{u}_1)$) for $\hat{u}_1 = u_1$ and $\mu_{\hat{A}_1^j}(\hat{u}_1) = 0$ for $\hat{u}_1 \neq u_1$). That is, with singleton fuzzification we have $\mu_{\hat{A}_1^{fuz}}(u_1) = 1$, for all i = l, 2, ..., n for the given u_i inputs so that

$$\mu_{\hat{A}_1^j}(u_1) = \mu_{A_1^j}(u_1)$$

$$\mu_{\hat{A}_2^k}(u_2) = \mu_{A_2^k}(u_2)$$

$$\vdots$$

$$\mu_{\hat{A}_n^l}(u_n) = \mu_{A_n^l}(u_n)$$

It is clear 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,..... , 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 (what we called $\mu_{premise}$ in the last section on the inverted pendulum) that represent the certainty that each rule premise holds for the given inputs.

Define

$$\mu_i = (u_1, u_2, \ldots, u_n) = \mu_{\hat{A}_1^j}(u_1) * \mu_{\hat{A}_2^k}(u_2) * \mu_{\hat{A}_n^l}(u_n)$$

which is simply a function of the inputs u_i. When singleton fuzzification is used, we have

$$\mu_i = (u_1, u_2, \ldots, u_n) = \mu_{A_1^j}(u_1) * \mu_{A_2^k}(u_2) * \mu_{A_n^l}(u_n)$$

We use to represent the certainty that the premise of rule (i) matches the input information when we use singleton fuzzification. This $\mu_i(u_1,u_2,...,u_n)$ is simply a multidimensional certainty surface, a generalization of the surface shown in Fig (2.9) for the inverted pendulum example. 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 sets (as we did for the pendulum earlier) 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 $(j, k, \ldots, l; p,q)_i$, the "implied fuzzy set" Bq with membership function

$$\mu_{B'^i_q}(y_q) = \mu_i(u_1, u_2, .., u_n) * \mu_{B_q^i}(y_q) \qquad (3)$$

The implied fuzzy set (B'^i_q) 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 $\mu_{B'^i_q}(y_q)$ for each rule. An example of an implied fuzzy set can be seen in Fig (2.11b) for the inverted pendulum example.

Alternative 2: Determine the Overall Implied Fuzzy Set: Alternatively, the inference mechanism could, in addition, compute the "overall implied fuzzy set" $B\hat{}_q$ with membership function

$$\mu_{B\hat{}_q}(y_q) = \mu_{B\hat{}_q^1}(y_q) \oplus \mu_{B\hat{}_q^2}(y_q) \oplus \ldots \oplus \mu_{B\hat{}_q^R}(y_q) \qquad (4)$$

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). Notice that we did not consider this possibility for the inverted pendulum example for reasons that will become clearer in the next subsection. Instead, our 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 $\mu_{B\hat{}_q}(y_q)$ 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 adhere to this principle. To see this, we should study Equation (3) 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 $\sup_{y_q}\{\mu_{B\hat{}_q}(y_q)\} \leq \mu_i(u_1, u_2,.., u_n)$. The fact that we are no more certain of our consequents than our premises is shown graphically in Fig (2.17) where the heights of the implied fuzzy sets are always less than the certainty values for all the premise terms.

2.4 Defuzzification

A number of defuzzification strategies exist, and it is not hard to invent more. Each provides a means to choose a single output (which we denote with (y_q^{crisp}) based on either the implied fuzzy sets or the overall implied fuzzy set (depending on the type of inference strategy chosen, "Alternative 1 or 2," respectively, in the previous section).

As they are more common, we first specify typical defuzzification techniques for the implied fuzzy sets $\mu_{B'_q{}^i}$:

- Center of gravity (COG): A crisp output y_q^{crisp} is chosen using the center of area and area of each implied fuzzy set, and is given by

$$Y_q^{crisp} = \frac{\sum_{i=1}^{R} b_i^q \int y_q \mu_{B'_q{}^i}(y_q) dy_q}{\sum_{i=1}^{R} \int y_q \mu_{B'_q{}^i}(y_q) dy_q}$$

where R is the number of rules, b_i^q is the center of area of the membership function of B_q^p associated with the implied fuzzy set $B'_q{}^i$ for the i^{th} rule (j, k, . . . , l; p, q)$_i$, and

$$\int y_q \mu_{B'_q{}^i}(y_q) dy_q$$

Denotes the area under denotes the area under $\mu_{B'_q{}^i}(y_q)$. Notice that COG can be easy to compute since it is often easy to find closed-form expressions for $\int y_q \mu_{B'_q{}^i}(y_q) dy_q$. which is the area under a membership function. Notice that the area under each implied fuzzy set must be computable, so the area under each of the output membership functions (that are used in the consequent of a rule) must be finite (this is why we cannot "saturate" the membership functions at the outermost edges of the output universe of discourse). Also, notice that the fuzzy system must be defined so that

$$\sum_{i=1}^{R} \int y_q \mu_{B'_q{}^i}(y_q) dy_q \neq 0$$

for all u_i or y_q^{crisp} will not be properly defined. This value will be nonzero if there is a rule that is on for every possible combination of the fuzzy system inputs and the consequent fuzzy sets all have nonzero area.

- Center-average: A crisp output yqrisp 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

$$Y_q^{crisp} = \frac{\sum_{i=1}^{R} b_i^q \, sup_{y_q} \{ \mu_{B'_q{}^i}(y_q) \}}{\sum_{i=1}^{R} sup_{y_q} \{ \mu_{B'_q{}^i}(y_q) \}}$$

where "sup" denotes the "supremum" (i.e., the least upper bound which can often be thought of as the maximum value). Hence, $\sup_x\{\mu(x)\}$ can simply be thought of as the highest value of $\mu(x)$ (e.g., $\sup_u\{\mu_1(u)\} = 0.25$ for $\mu_1(x)$ when product is used to represent the implication, as shown in Fig(2.15). Also, b_i^q is the center of area of the membership function of B_q^p associated with the implied fuzzy set $\hat{B}_q^i$ for the i^{th} rule (j, k, . . . , l; p, q)$_i$. Notice that the fuzzy system must be defined so that

$$\sum_{i=1}^{R} \sup_{y_q} \{\mu_{\hat{B}_q^i}(y_q)\} \neq 0$$

for all u_i . Also, note that $\sup_{y_q} \{\mu_{\hat{B}_q^i}(y_q)\}$ is often very easy to compute since if $\mu_{\hat{B}_q^i}(y_q)=1$ for at least one y_q (which is the normal way to define consequent membership functions), then for many inference strategies, using Equation (3), we have $\sup_{y_q} \{\mu_{\hat{B}_q^i}(y_q)\} = \mu_i(u_1, u_2,.., u_n)$, which has already been computed in the matching process. Moreover, the formula for defuzzification is then given by

$$Y_q^{crisp} = \frac{\sum_{i=1}^{R} b_i^q \mu_i(u_1, u_2,.., u_n),}{\sum_{i=1}^{R} \mu_i(u_1, u_2,.., u_n)}$$

where we must ensure that

$$\sum_{i=1}^{R} \mu_i(u_1, u_2,.., u_n) \neq 0$$

for all u_i. Also note that this implies that the shape of the membership functions for the output fuzzy sets does not matter; hence, you can simply use singletons centered at the appropriate positions. Next, we present typical defuzzification techniques for the overall implied fuzzy set $\hat{B}_q$:

- 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 $\hat{B}_q$ achieves a maximum that is,

$$y_q^{crisp} = \in \left\{ \arg \sup_{y_q} \left\{ \mu_{\hat{B}_q}(y_q) \right\} \right\}$$

Here, "$\text{argsup}_x\{\mu(x)\}$" returns the value of (x) that results in the supremum 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 (μ_1) and (μ_2) over all (u) in Fig (2.16) (i.e., $\mu_{overall}(u)=\max_u\{\mu_1(u),\mu_2(u)\}$ per Equation (4)). In this case, $\text{argsup}_u\{\mu_{overall}(u)\}= -10$, which is the defuzzified value via the max criterion. Sometimes the supremum can occur at more than one point in Y_q (e.g.,consider the use of the max criterion for the case where minimum is used to represent the implication, and triangular membership functions are used on the output universe of discourse, such as in Fig (2.14). In this case you also need to specify a strategy on how to pick only one point for y_q^{crisp} (e.g., choosing the smallest value). Often this defuzzification strategy is avoided due to this ambiguity; however, the next defuzzification method does offer a way around it.

- Mean of maximum: A crisp output y_q^{crisp} is chosen to represent the mean value of all elements whose membership in is a maximum.

We define $\hat{b}_q^{max}$ as the supremum of the membership function of $\hat{B}_q$ over the universe of discourse Y_q. Moreover, we define a fuzzy set $\hat{B}_q^*$ with a membership function defined as

$$\hat{B}_q^*(y_q) = \begin{cases} 1 & \mu_{\hat{B}_q}(y_q) = \hat{b}_q^{max} \\ 0 & \text{Otherwise} \end{cases}$$

then a crisp output, using the mean of maximum method, is defined as

$$y_q^{crisp} = \frac{\int_{y_q} y_q \mu_{\hat{B}_q^*}(y_q)dy_q}{\int_{y_q} \mu_{\hat{B}_q^*}(y_q)dy_q}$$

where the fuzzy system must be defined so that $\int_{y_q} \mu_{\hat{B}_q^*}(y_q)dy_q \neq 0$ for all u_i. As an example. suppose that for Fig (2.17) the two implied fuzzy sets are used to form an overall implied fuzzy set by taking the maximum of the two certainty values over all of u (i.e.,$\mu_{overall}(u)= \max_u\{\mu_1(u), \mu_2(u)\}$ per Equation (4)). In this case there is an interval of (u) values around (-10) where the overall implied fuzzy set is at its maxi- mum value. and hence there is an ambiguity about which is the best defuzzified value. The mean of the maximum method would pick the value in the middle of the interval as the

defuzzified value, so it would choose (-10). This can require excessive computational resources for continuous universes of discourse. For some types of membership functions, simple ideas from geometry can be used to simplify the calculations; however, for some choices of membership functions, there may be many subintervals spread across the universe of discourse where the maximum is achieved. In these cases it can be quite difficult to compute the defuzzified value unless the membership functions are discretized. Complications such as these often cause designers to choose other defuzzification methods.

- Center of area (COA): A crisp output y_q^{crisp} is chosen as the center of area for the mem- bership function of the overall implied fuzzy set B'_q. For a continuous output universe of discourse Y_q, the center of area output is denoted by

$$y_q^{crisp} = \frac{\int_{y_q} y_q \mu_{B'_q{}^*}(y_q)dy_q}{\int_{y_q} \mu_{B'_q{}^*}(y_q)dy_q}$$

The fuzzy system must be defined so that $\int_{y_q} \mu_{B'_q{}^*}(y_q)dy_q \neq 0$ for all u_i . Note that, similar to the mean of the maximum method, this defuzzification approach can be computationally expensive. For instance, we leave it to the reader to compute the area of the overall implied fuzzy set $\mu_{overall}(u) = \max_u\{\mu_1(u), \mu_2(u)\}$ for Fig16. Notice that in this case the computation is not as easy as just adding the areas of the two chopped- off triangles that represent the implied fuzzy sets. Computation of the area of the overall implied fuzzy set does not count the area that the implied fuzzy sets overlap twice; hence, the area of the overall implied fuzzy set can in general be much more difficult to compute in real time. It is important to note that each of the above equations for defuzzification actually provides a mathematical quantification of the operation of the entire fuzzy system provided that each of the terms in the descriptions are fully defined. We discuss this in more detail in the next section. Overall, we see that using the overall implied fuzzy set in defuzzification is often unde- sirable for two reasons:

1. The overall implied fuzzy set B'_q is itself difficult to compute in general,
2. The defuzzification techniques based on an inference mechanism that provides B'_q are also difficult to compute.

CHAPTER 3

THE DEVELOPMENT OF FUZZY CONTROL SYSTEM

The inference engine is the heart of a fuzzy controller (and any fuzzy rules system) operation can be divided into three steps (Fig 3.1):

- 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.

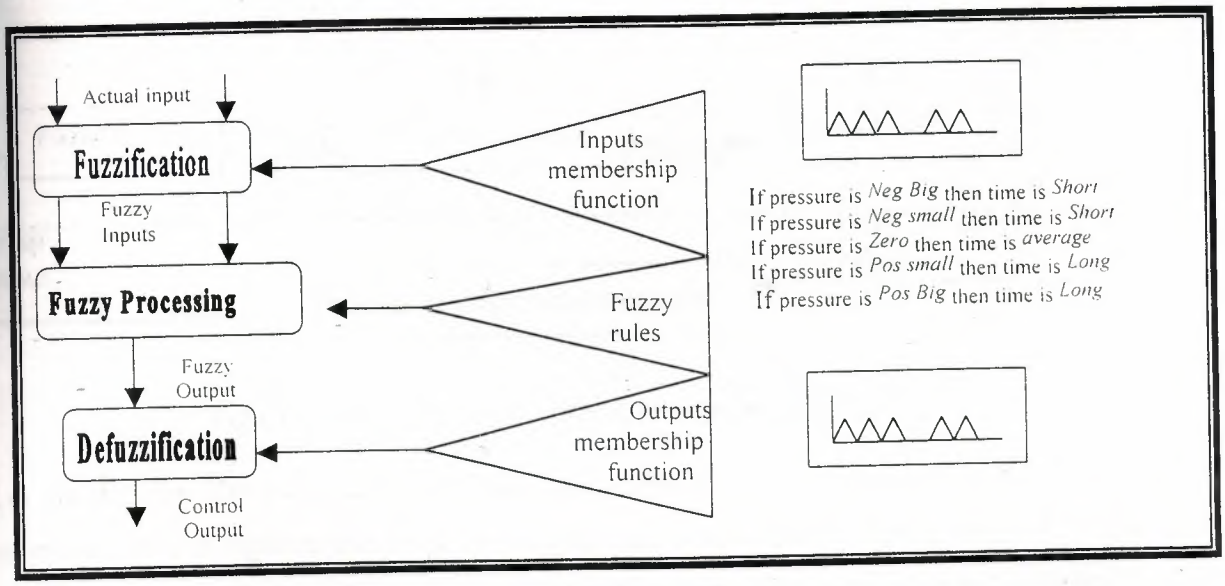


Fig (3.1) Operating of a fuzzy controller

In real control system, the controller output should be used to control a real object or process. It needed to know a scrip 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. Define the condition interface-fix the ways in which observation of the process are expressed as fuzzy sets.

2. Design the rule base-determine, which rules are to be applied under which conditions.
3. Design the computational unit-supply algorithms to perform fuzzy computations. That unit will generally lead to fuzzy outputs.
4. Determine rules according to which fuzzy control statement can be transformed into crisp control actions.

The typical structure of fuzzy controller is given in (Fig 3.2)

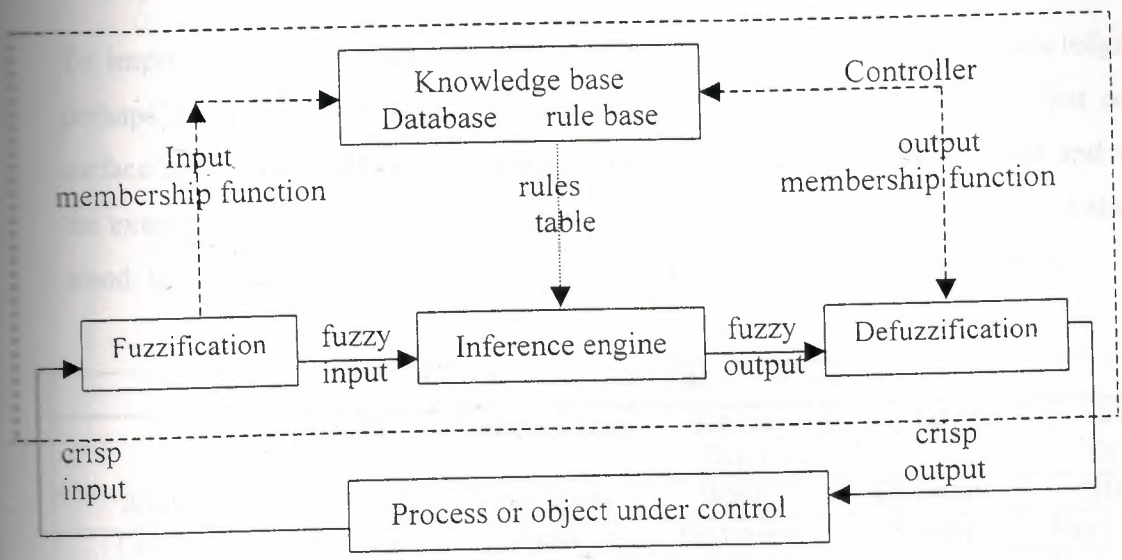


Fig (3.2) The fuzzy logic controller (a basic structure)

Let us develop the rules table for the fuzzy controller of a vacuum cleaner. This controller should regulate the force of sucking dust from a surface being cleaned. This force can be described as a linguistic variable with values: very strong, strong, ordinary. week, very weak. The input of this controller should ordinary consider an amount of dust on the surface. The surface can be very dirty, dirty, rather dirty, almost clean. clean. The controller can change the force depending on how dirty the surface is. One can propose the following set of rules to describe the controller operation:

If surface is very dirty	then force is very strong.
If surface is dirty	then force is strong.
If surface is rather dirty	then force is ordinary.
If surface is almost clean	then force is weak.
If surface is clean	then force is very weak.

It is more convenient to write this rules set in a table form.

Surface	Force
Very dirty	Very strong
Dirty	Strong
Rather dirty	Ordinary
Almost clean	Weak
Clean	Very weak

Table (3.1) Rules table for a fuzzy vacuum cleaner

To improve the performance, one should apply some extra expert's knowledge. So perhaps a driving force should depend not only on an amount of dust, but on the surface texture and fabric also. There is some difference in cleaning wood and wool, for example. So let us introduce another input: surface type with linguistic values of wood, tatami, and carpet. This will help to implement the following rules table.

Table (3.4)

	Clean	Almost Clean	Rather Dirty	Dirty	Very Dirty
Wood	Very Weak	Very Weak	Weak	Ordinary	Strong
Tatami	Very Weak	Weak	Ordinary	Strong	Very Strong
Carpet	Weak	Ordinary	Ordinary	Strong	Very Strong

Table (3.2) Rules table for surface type and dust amount.

These names, or linguistic labels, have a symbolic sense only. They just mark different membership function which should describe how easy it is to clean a particular surface, i.e. to mark a different degree of 'easyness'. These degrees are fuzzy, in that fabric can be considered as 'a little tatami and mainly wood'. This could be placed between tatami and carpet and considered as a little of this and a little of that. The dust sensor includes a phototransistor, which is mounted opposite an infrared light-emitting diode. Infrared rays are emitted in a beam. When they pass through the dust, some rays are lost; causing the amount that reaches the phototransistor to decrease. The varying component is amplified and used to evaluate the amount of dust on the surface being cleaned.

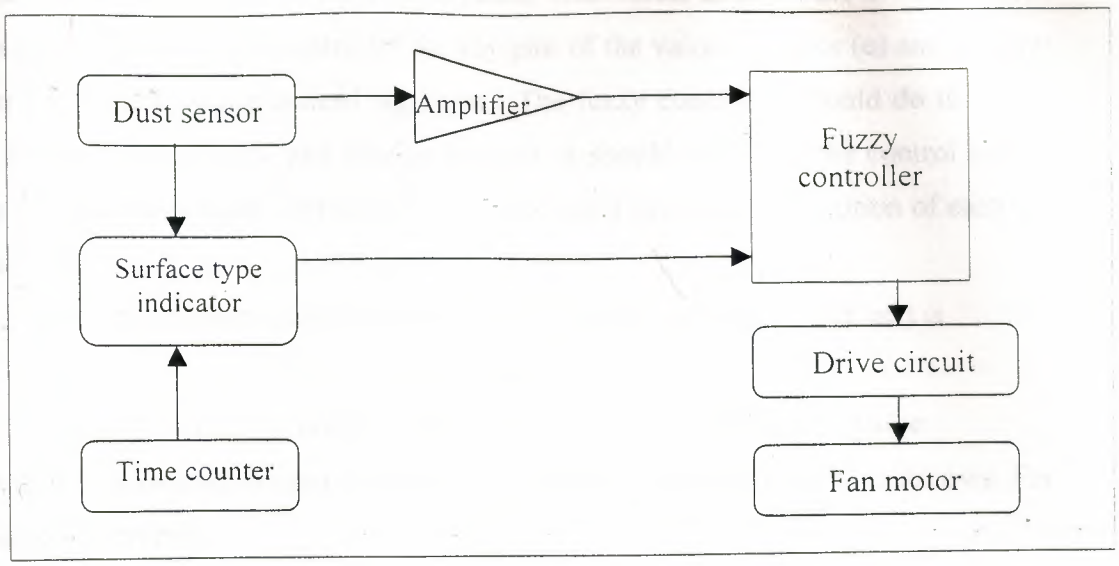


Fig (3.3) The structure of the vacuum cleaner fuzzy cleaner.

As cleaning proceeds, the amount of dust decreases, but the speed of decreasing depends on the surface type. If the surface is smooth like wood, it is cleaned very fast because it is easy to pickup dust from such a surface. On the other hand, it is hard to clean wool carpet surface. Thus, by evaluation of the change of the dust amount collected during a time unit, we can judge the type of the surface being cleaned. This controller works well under different conditions. So in this controller, the first input is the amount of dust collected during a time unit. And the second input is the change in this amount. Then the second input can be considered as a derivative of the first one. So our fuzzy controller is an analogy to a classical PD-controller.

3.1 PD-Like Fuzzy Controller

The equation giving a conventional PD-controller is

$$u(k) = K_P * e(t) + K_D * \Delta e(t), \qquad \text{Equation (3.1)}$$

Where K_P and K_D are the proportional and the differential gain factors.

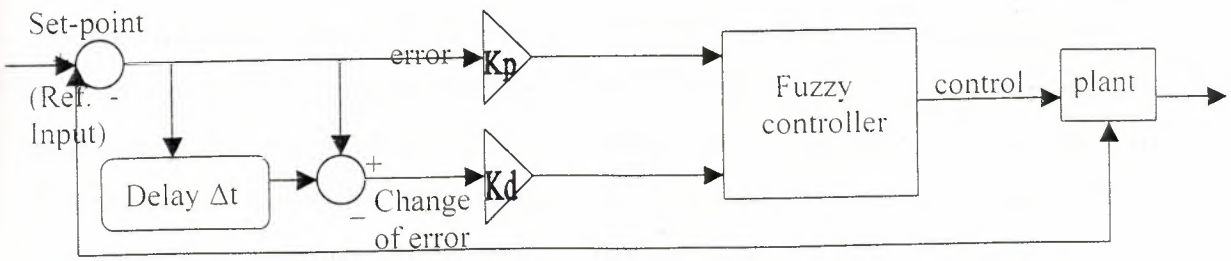


Fig (3.4) A block-diagram of a PD-like fuzzy control system.

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 $\Delta e(t)$ 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. And these variables might be measured in different units. So the rules if the PD controller 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 mean 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. Linguistic values of (e) with a negative sign mean that the current process output y has a value below the set-point ysp 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 (3.3) 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 [drian93] where both inputs and an output have a set of possible linguistic values {NB, NM, NS, Z, PS, PM, PB} where NB stand 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 $\Delta e(t)$ is NB then u(t) is NB.

Table (3.3)

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

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 (Δe(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 (Δe(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 somewhat 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 (3.3). 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.

3.2 PI-like fuzzy controller

The equation given a conventional PI-controller is

$$u(t) = K_P * e(t) + K_I * \int e(t)dt, \qquad \text{Equation (3.2)}$$

Where K_P and K_I are the proportional and the integral gain coefficients. A block diagram for a fuzzy control system looks like Fig (3.5).

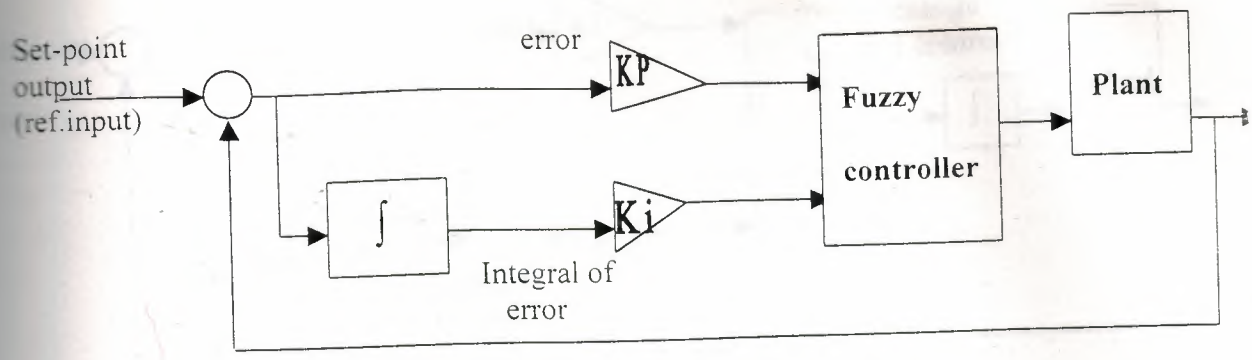


Fig (3.5) A block-diagram of a PI fuzzy control system

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 Δu(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 they.

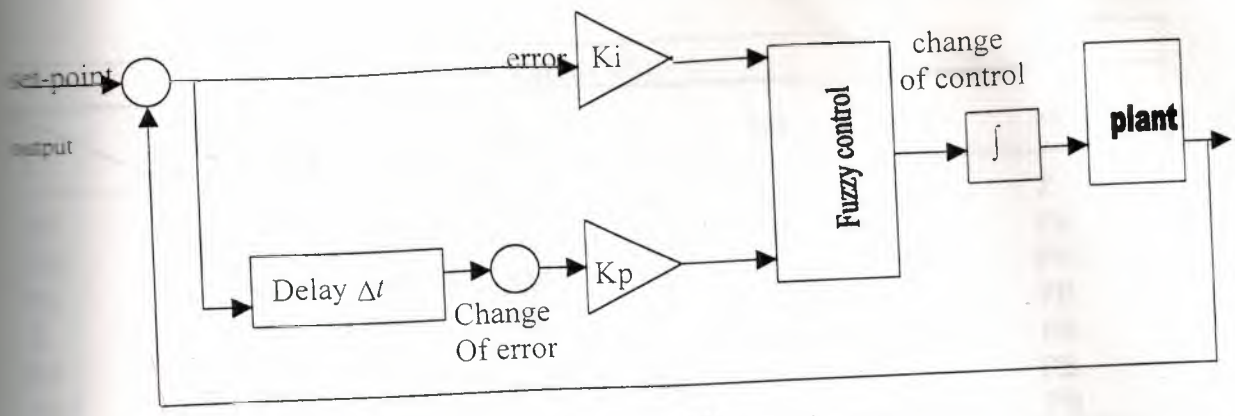


Fig (3.6) a block diagram of PI fuzzy control system.

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

Table (3.4)

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	NB	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

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.

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 a designer tries to make the control surface smoother in the vicinity of a set-point. If it needed to make our controller less reactive to the large errors. In this case we need to modify the top and the bottom rows

Table (3.5)

e \ Δe	PB	PM	PS	Z	NS	NM	NP
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

If it needed to change the left bottom corner 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.

3.3 PID-like fuzzy controller

The equation for a 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 state variables, namely sum-of-errors, denoted (σe) and computed as:

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

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:

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

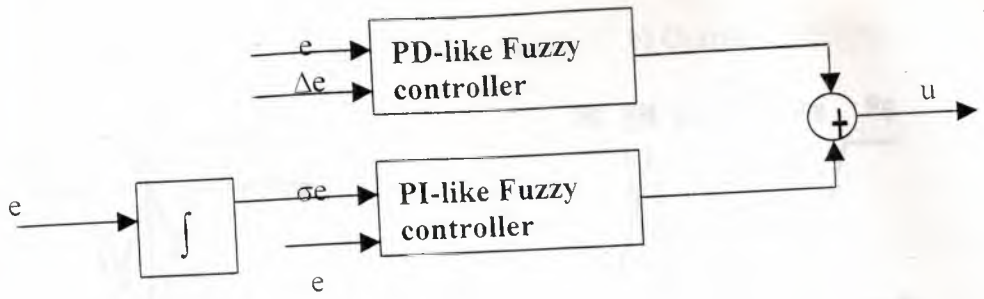


Fig (3.7) The structure for PID-like fuzzy controller.

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.

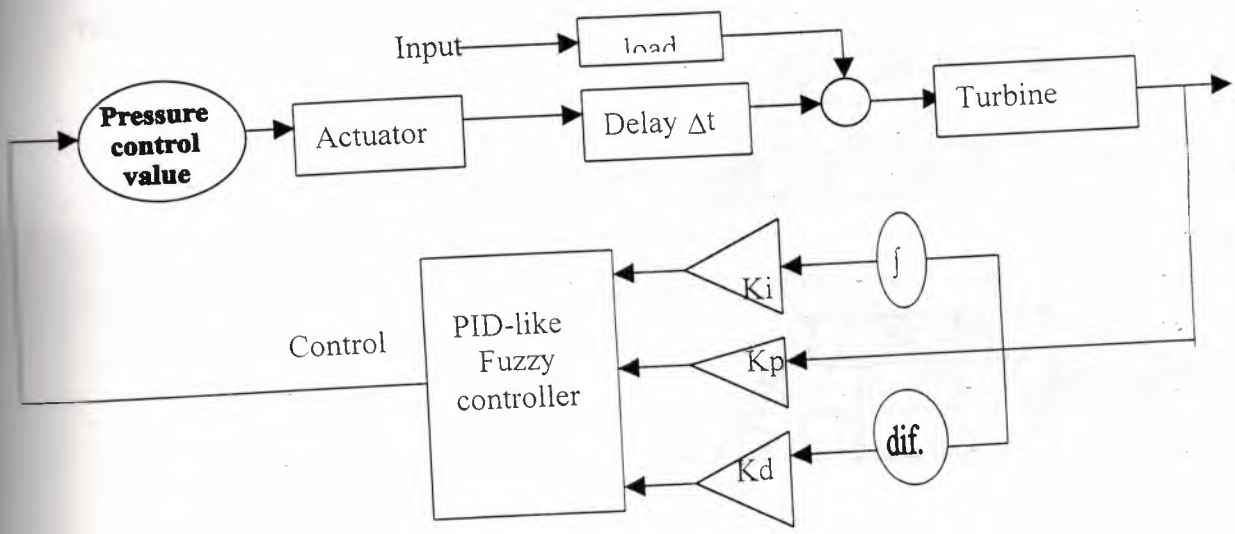


Fig (3.8) The block diagram of fuzzy controller system for a turbine speed control.

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 Fig (3.8). 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 Fig (3.9a).

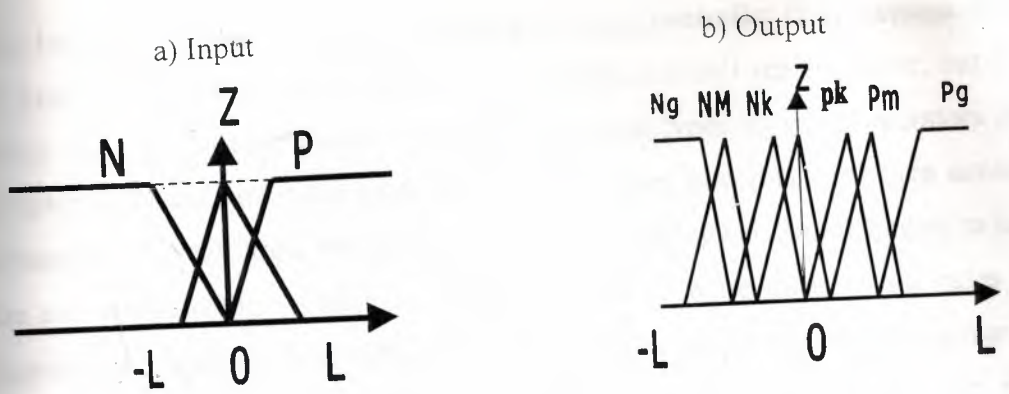


Fig (3.9) Membership functions for the inputs and outputs.

In order to defuzzify the output, seven classes are applied with the membership functions presented in Fig (3.9b). Because the fuzzy controller has three inputs, its rules table has a three-dimensional image given in Fig (3.10).

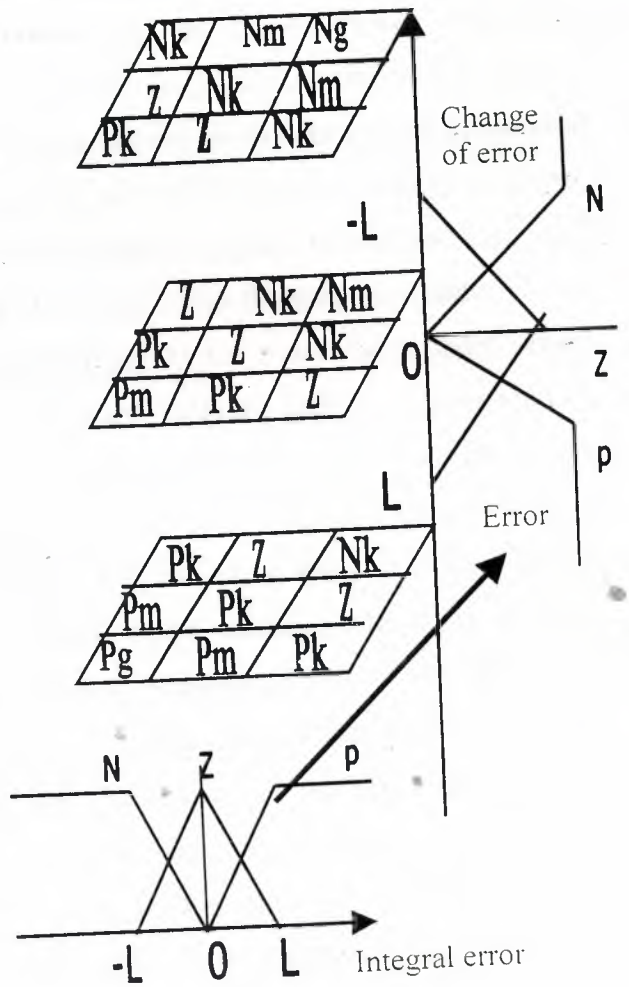


Fig (3.10) the rules table for the fuzzy controller.

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. 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. Fuzzy controller can be particularly good as an operator replacement. However, an operator in process control system usually controls different technical devices including PID-controllers. In this case a fuzzy controller is placed on a higher, more intelligent level. It produces command signals for conventional controllers. Another possible task for fuzzy controller in this structure is a future prediction. The famous Japanese subway and helicopter control system are based on these principles. A modern aircraft is well equipped with conventional control techniques and, in particular, various PID controllers that demonstrate a good performance and successfully solve different guidance problems. Guidance control is the PID controllers producing the control signals, which are applied, to ailerons and elevators perform a modern aircraft. The necessary reference inputs for PID controller are usually supplied by the aircraft crew based on different data, first of all the current position provided through the global positioning system (GPS). Some piloted aircraft classes are to be replaced with

autonomous vehicles, which are cheaper in operation and have some other advantages. The hybrid guidance control system, incorporating conventional PID controllers and fuzzy controller, is proposed for this aircraft. A fuzzy controller takes the place of a pilot (on operator) in developing signals for a PID controller. The navigation of the self-piloted vehicle is organized by the onboard (GPS) receiver tied to a PC-based flight director. Flight-planning software generates a list of consecutive points necessary to track the mission-determined flight path. Onboard autopilots keep the aircraft stable, while the flight director (guidance system) interprets the point positions to determine the course, speed, climb rate and turns of the aircraft. The object of the guidance system is to bring the aircraft to the next operational point at a specified altitude and to stabilize the vehicle to allow for the operation of the onboard photographic and/or measurement equipment. The aircraft is assumed to be guided to an initial position during the take-off stage before the guidance system takes over a control. The designed control structure is shown schematically in Fig (3.11). The altitude of the aircraft is controlled by low-level conventional PID feedback controller through aerodynamic ailerons and elevator with mechanical limits of their deflection angles. The throttle setting controls a speed of the aircraft. The guidance fuzzy controller has to provide reference signals for the PID controller, which are required roll and pitch angles (θ_{ref} respectively) for a leveled flight, and also to produce the throttle setting command T_{th} if change of altitude is required.

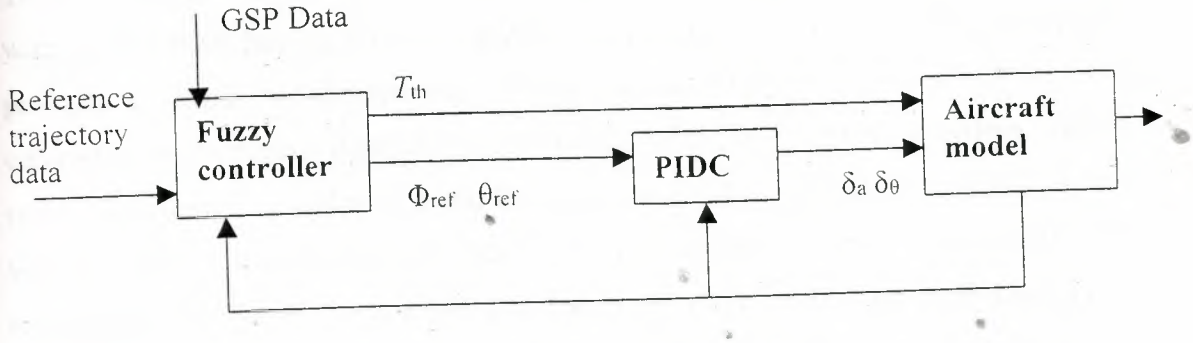


Fig (3.11) Structure of the combined control system.

An operation of the fuzzy controller developed is illustrated in Fig (3.12). Coordinates of the aircraft current position and of the next operational point are used to estimate an offset angle, δ, between the direction to the operational point and the current velocity vector, $\dot{\upsilon}$, and the rate of change of the offset angle, $\dot{\delta}$, as well as an altitude difference between the current position of the aircraft and the operational altitude, h, and the rate

of change of the altitude difference $\hat{h}$. These estimates become the input signals for the fuzzy controller and are subject to fuzzification.

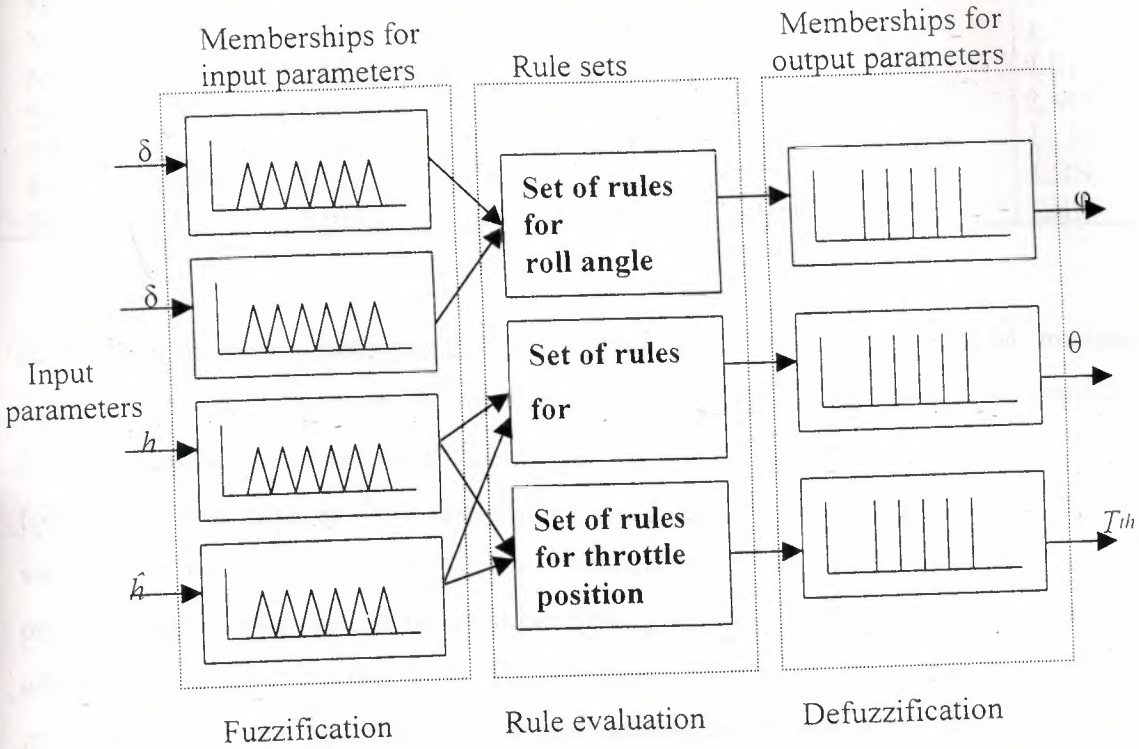


Fig (3.12) Operational block diagram of the fuzzy control system.

The whole control structure consists of three fuzzy controllers operating independently with each of them having two input and one-output signals. This input-output mapping provides a simple two-dimensional structure of the linguistic rules sets. The fuzzy controller output mainly depends on a definition of the membership functions and the rules. The control variables δ, δ and domains are divided into seven linguistic values with the relative memberships, and the control variable $\hat{h}$ into five, respectively. The rule definition is subjective and based on the expert's knowledge and experience. For a system with two control variables and seven membership functions in each range it may lead to a (7x7) decision table. The total of three rules sets is used in this fuzzy controller design: for the roll angle control, the pitch angle control and the throttle position control. These rules sets can be viewed as (7x7, 7x5 and 7x5) decision tables, respectively. As an illustration, the rules set for roll angle control is given in table (3.6), where equally shaded cells produce the same fuzzy output.

Table (3.6)

δ \ δ	NB	NM	NS	SM	PS	PM	PB
NB	LB	LB	LB	LB	LB	LB	LL
NM	LB	LL	LL	LL	LL	LL	LM
NS	LL	LM	LM	LM	LM	LM	LLIT
SM	LM	LLIT	LLIT	LLIT	LLIT	LLIT	LMN
PS	LLIT	LMN	LMN	LMN	LMN	LMN	LLN
PM	LMN	LLN	LLN	LLN	LLN	LLN	LBN
PB	LLN	LBN	LBN	LBN	LBN	LBN	LBN

In this table linguistic tables for roll angle denoted: LB, big; LL, large, LM, medium; LLIT, little; the character N denotes negative. Simplicity and low hardware implementation cost determine a choice of the membership functions of a singleton type for the output parameters (roll angle, pitch angle and throttle position). The linguistic variables of the fuzzy outputs that are evaluated by cycling through the rule sets are projected onto output sets of the memberships. The defuzzification process takes place after the generation of the fuzzy control signals is completed using the inference mechanism. As more than one fuzzy output variable can be assigned a non-zero degree. the contribution of each variable into the physical output should be taken into account. The defuzzification method was based on calculating the center of gravity of all fuzzy outputs for each system physical output.

CHAPTER 4

IMPLEMENTATION OF FUZZY CONTROL SYSTEM

The simplest and the most usual way to implement a fuzzy controller is to realize it as a computer program on a general-purpose processor. However, a large number of fuzzy control applications require a real-time operation to interface high-speed external devices. For example, automobile speed control, electric motor control robot control are characterized by severe speed constraints, software implementation of fuzzy logic on general purpose processors can not be considered as a suitable design solution for this type of application. In such cases, specialized fuzzy processors can match design specifications. The requirements to the hardware implementation are:

- High-speed performance.
- Low complexity.
- High flexibility.

These conditions contradict each other. So it is not easy to choose the right way, especially if one takes into account some other factors, such as manufacturing cost (very important for consumer product fuzzy controllers) or design cost (important in research and development). Low complexity means that algorithms for fuzzy processing, fuzzification and defuzzifications have to be very simple and demand as small an amount of memory as possible for their realization. Flexibility means the ability of the hardware to be used successfully in different applications and configurations. During recent years, there has been an increasing interest in the development of efficient fuzzy controller hardware capable of coping with the requirements of real-time applications. The first fuzzy logic chip has been developed in 1985. Later, a fuzzy logic hardware using analog techniques has been developed. Since then, several chips have been proposed utilizing both analog and digital techniques. Generally, three different ways of implementating fuzzy controller hardware can be proposed. They are summarized, together with their advantages and disadvantages, in table (4.1).

Table (4.1)		
Class of hardware Implementation	Advantages	Disadvantages
Digital general purpose processor.	Flexibility in choice of hardware and software tools.	Low performance unless a very powerful one is used.
Digital specialized processor.	Increasing Performance.	Incrementing complexity and cost, as it must be coupled with a standard host processor. Lack of flexibility, as it may be applied to a limited class of problems. Higher cost
Analog processor	High performance. Low cost. Low power consumption.	Mainly the research topic. Low accuracy. Lack of flexibility.

One can see that any hardware type has its positive and negative sides for fuzzy controller application.

4.1 Implementation On a Digital General Purpose Processor

Nowadays, most fuzzy controllers are implemented as software programs on general-purpose processors and microprocessors. If there is a need for higher operation speed, a specialized fuzzy processor can be added. An example of such a processor is the FC110. The advantage of this processor is the powerful arithmetic logical unit. However, for defuzzification implementation, a fast multiplication and division are needed. If an 8-or 16-bit microprocessor is applied for a fuzzy controller realization, the FC110 speeds up the operation significantly. If a 32-bit or 64-bit microprocessor is used, the advantage of the FC110 processor is only the fast minimum and maximum operations. At the same time, other operations can be performed faster on a general-purpose processor. Some

advice can be provided on how to speed up a controller operation on a general processor. In this case, the optimization is based not on the features of a particular platform, but on the specific features of the fuzzy controller operation. Research has demonstrated that only two stages of a fuzzy controller operation take most of the processing time, about 83 per cent for rules processing and 16 per cent for defuzzification. These are why the main efforts should be and are concentrated on the inference engine implementation. On the other hand, the important features of modern general microprocessors should be considered as well. These features include the following:

- Addition, multiplication and division operations are about 10 to 100 times faster with integer numbers than with floating point numbers. Floating-point operations are not available on microcontroller.
- Usually minimum and maximum operations are not available.
- Jumps for short distances are fast.
- Data that is often used is held in the processor cache or register memory.

Based on this, the following recommendations can be provided:

- All calculations operations must be done with integer numbers. It can be easily realized as the input signal for the fuzzy controller comes from the AD converter outputs an integer code.
- The membership functions for the inputs are to be stored in look-up tables. In this case, the antecedent parts of the rules can be calculated very fast.
- Data dependencies, especially the property of the minimum operation, that min (X, 0) = 0, can reduce the number of operations drastically. It can cause a significant reduction in the rules calculation time. If an antecedent part is Zero and the *t* norm operation is interpreted as the min operation, then this rule is not activate. There is no need to calculate the consequent part. Moreover, other antecedent conditions need not to be checked.

All other rules with the same antecedent part do not need to be calculated either. Generally, if similar antecedent parts are used in different rules (that is quite common in fuzzy controllers), there is no need to recalculate them, and the result of the first calculation can be reused in the actual rule. As an example, the comparison between

PC80486, 33 MHz, the FC110 and the MC8052, 12 MHz is given, figure (4.1). The FC110 is nearly 20 times faster than MC8052 and needs less program memory.

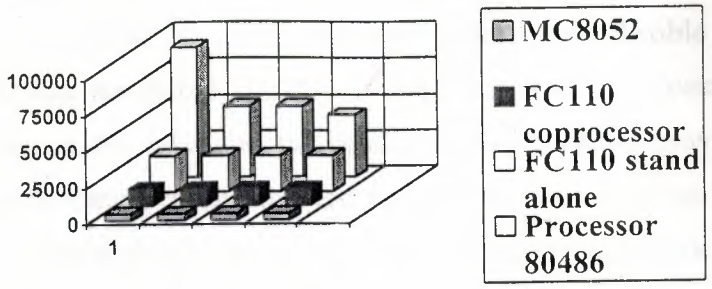


Fig (4.1) comparison results for a standard microcontroller MC8052, a specialized digital processor FC110, and a general digital processor PC80486: performance benchmark in iterations per second.

In general, microcontrollers like the MC8052, which is an 8-bit type, are suitable as a cheap realization of fuzzy controllers with a medium complexity and low or medium execution time. These and similar microcontrollers are well tested and often used. However, the PC80486 with an optimized C-code implementation is faster than the FC110 because of the improved algorithm. The generation and optimization of an assembler code can reduce the execution time for it reduced even further. So, for faster processing, the extension and optimization of some basic fuzzy operations are the most important.

4.2 Implementation on a Digital Specialized Processor

Research into the balance between performance and cost has recently led to the development of architecture solutions with a specific support, and several accelerator coprocessors dedicated to fuzzy logic and control have been proposed. Dedicated hardware may be considered as the best way in terms of performance, but it can only cover a limited range of applications. In spite of the lack of flexibility, the choice of the entire specialized hardware solutions may represent an effective way, in particular for the applications, which require a large number of rules. Specific fuzzy hardware allows us in many cases to reach a better cost-performance ratio because of the exploitation of parallelism in fuzzy processing and the introduction of special purpose units. To introduce this type of implementation, consider two specialized processors. The first one is FC110; it is rather old (a few years). The second one is AL220, which is very

new. The FC110 digital fuzzy processor was developed as a specialized fuzzy processor. It is a single chip small enough for sensitive embedded applications. Its architecture supposes high communication possibilities for working together with a host processor. It is not oriented to any particular host type and is flexible in possible applications. Variable data are stored in a 256-byte on-chip RAM. At least the low 64 bytes are shared between the host and the device with arbitration provided by the FC110. Special communication capabilities are assigned to two of the addresses. Off-chip data interfacing is also possible. Due to this architecture, the microprocessor allows the program and all the constant data to reside in an off-chip ROM and the variable data to be placed in an on-chip RAM that both the host and the device can access. The shared RAM is used for temporary storage and to transfer observations, commands, and conclusions. Additional RAM is provided in a 192-byte segment adjacent to the shared RAM. The AL220 is an inexpensive, high performance; stand-alone microcontroller utilizing fuzzy control. The device contains four 8-bit resolution analog inputs and four 8-bit analog outputs, and an internal clock generator. Inputs can be directly connected to sensors or switches. Outputs can be connected to analog devices or used to control a mechanism. The AL220 consumes very little power during normal operation and has a power-down mode. The AL220 diagram is given on figure (4.2).

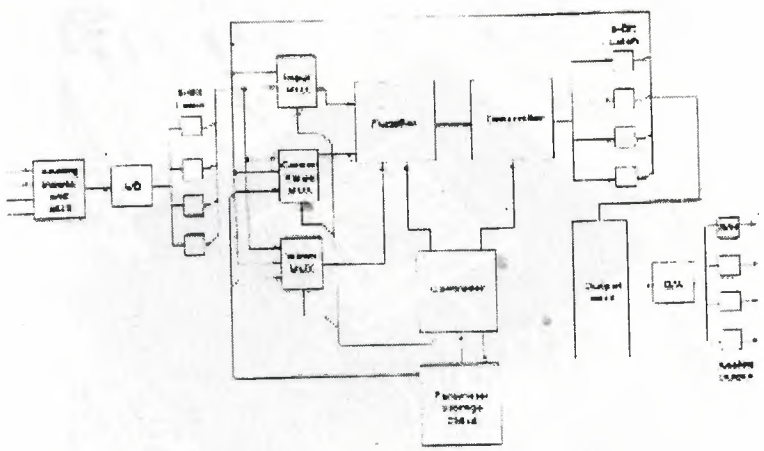


Fig (4.2) Detailed AL220 block diagram

The main elements are a fuzzier, a defuzzier and a controller, performing fuzzy processing. The knowledge base containing rules and membership functions is realized

as an EEPROM/ROM with the capacity of (256 * 8 bits). It is available in either an 18-pin DIP or 20-pin SOIC versions. One can see that this device includes on-chip A/D and D/A converters that eliminates a need in external devices and gives a designer a one-chip solution. The microcontroller reads voltage levels from its four analog inputs using an 8-bit A/D converter, processes the channel data according to fuzzy rules contained on the chip, and generates four analog inputs via its 8-bit D/A converters and four sample-and-hold output drivers. Fuzzy processing is performed at a decision rate of 500,000 rules per second that allows one to carry out first, second and third-order derivatives calculation and control, automatic calibration and rule-based timing at 10,000 samples per second for each of the four analog channels. Have a look at some fuzzy processor chips currently being developed. In order not to be accused of any biased approach, two fuzzy chips are presented. One of a Japanese design and another one of European design, table (4.2). data given is from the manufacturers. Omron is famous for the world's first high-speed controller, FZ-1000. FP-3000 is a new generation fuzzy processor that is applied in different Omron products. WARP (Weight Associative Rule Processor) by SGS-Thomson, figure (4.3), is claimed to be the technological state-of-the-art processor. Table (4.2) is not comparing these two chips, but presenting some real-life data and demonstrating how fast this area is being developed.

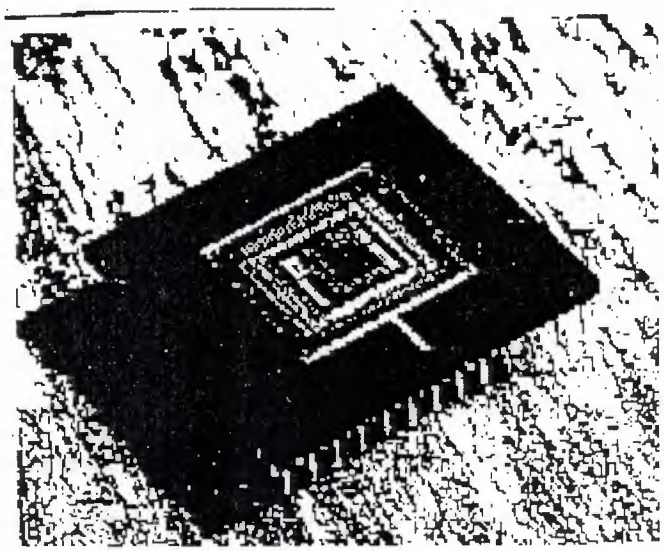


Fig (4.3) WARP fuzzy processor.

Table (4.2)		
Key features	FP-3000, Omron	WARP, SGS-Thomson
No. of rules processing inputs	8	16
No. of rules processing outputs	4	16
No. of possible membership functions for each input	7	16
No. of possible membership functions for each output	7	128
No. of types of membership functions supported	4 shapes (L, P, S, Z)	All
No. of rules	Single mode: 29 Expanded mode: 128 per group with 3 groups	Up to 256
Operating time	20 rules with 5 inputs and 2 outputs and defuzzification by center-of-gravity methods in 650 μs	32 rules with 5 inputs and 1 output are evaluated in 1.85 μs (1.5 MFLIPS)
Data resolution	Unsigned 12 bit	8 bit

A fuzzy coprocessor by Togai, the VY86C570, is a high-performance fuzzy coprocessor with a 12-bit FCA (Fuzzy Computational Acceleration) core, 4K * 12 OCTD (Observation, Conclusion and Temporary Data), RB (Rule Base), SMI (Shared Memory Interface), and host interface logic combined in a single chip. The VY86C570 is capable of executing simple to very complex fuzzy computation at high speeds, making it suitable for a wide range of fuzzy logic applications. Simple-to-medium complexity fuzzy logic rule bases can be directly downloaded by a host processor to 4K words (approximately 200 rules) of on-chip rule base memory. This allows designers to create fuzzy coprocessing systems without the need for an expensive on-board memory.

For larger fuzzy application requirements, the VY86C570 includes an external rule base interface that allows up to 64K words (over 1,000 rules) of rule base memory. In a typical fuzzy application, figure (4.4), a fuzzy rule base is downloaded by the host into RB memory prior to the start of any fuzzy computation. At the beginning of a fuzzy computation, 'crisp' input values, or observations, are downloaded by the host into OCTD SMI. The fuzzy core uses the rule base information stored in the RB memory to perform calculations and produce a set of 'crisp' output values or calculations. These values are stored by the fuzzy core in the OCTD SMI and are read by the host through the host interface. Control status registers are used to control the models of the chip and to provide status read-back.

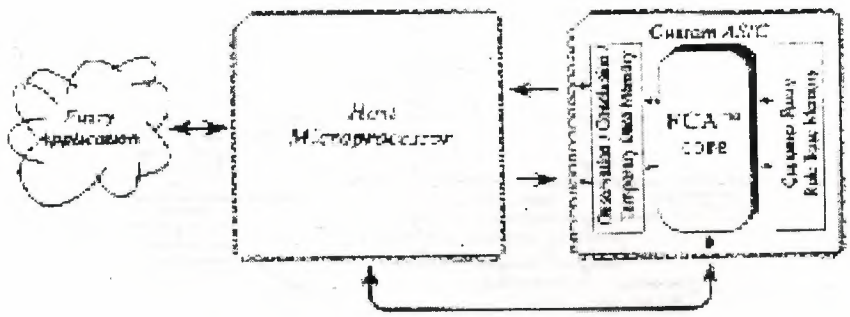


Fig (4.4) Typical application when a custom ASIC chip is used as a fuzzy processor.

4.3 Specialized Processor Development System

A special design environment is supplied to put all these data onto the microcontroller chip called a development system. FC110 has a development system, while AL220 has one called INSIGHT IIie™. This includes some software and a special hardware unit. All together this system provides an interface and tools for design development, simulation, real-time emulation and debugging. The development process is conducted on the personal computer in an MS-Windows environment and is pretty easy to perform. The block diagram for the FC110 development system is presented in figure (4.5), which explains the contents and operation of this system. In the Togai InfraLogic design package, the fuzzy controller rule base is written in FPL high-level language. The development system should translate this FPL description into an executable code that can be downloaded into the FC110 processor. The system includes a special Compiler, Assembler and Linker.

The Compiler translates the FPL code into the machine code optimized for the FC110 digital processor. The Assembler converts additional files written in assembler language into relocatable files. These files may contain information other than a knowledge base. The Linker combines all the parts of the code together. It utilizes specific information describing the board configuration. The Linker produces the code, which will be downloaded into the target board, if the knowledge base memory is implemented as RAM, burned in, if the memory is a PROM or permanently programmed into a ROM. It also outputs the C-code file which is used for the interface with the application software programs.

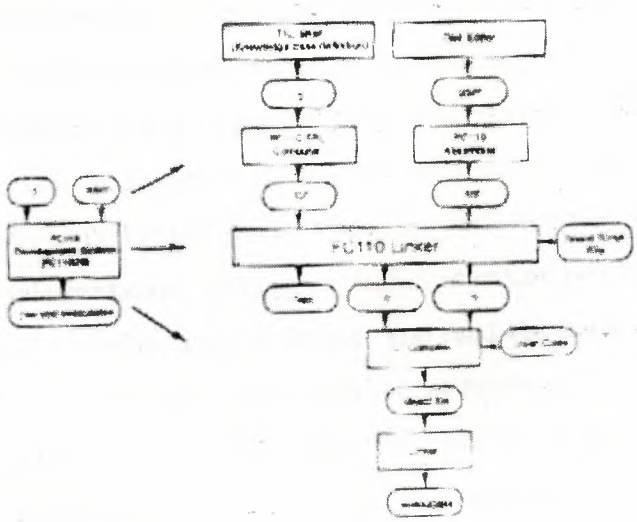


Fig (4.5) The FC110 development system block diagram.

4.4 Implementation On Analog Devices

During the last years, analog circuits attracted close attention as a good candidate for a fuzzy controller implementation. This implementation is characterized with a higher operation speed and lower power consumption. The functional efficiency is also much larger than for the digital realization because of the possibility of the versatile exploitation of small analog devices for a wide variety of low level linear and nonlinear processing required for fuzzy inference realization. The fuzzy controller application is a lucky exemption, which does not required high accuracy. Accuracy of 6-9 bits is enough and is quite affordable even for the cheapest analog implementation.

This makes analog circuits natural candidate for designing fuzzy controller chips with optimum speed-to-power ratio figures for low and medium precision applications, up to about 1 per cent. That's why some circuits and chips have been developed and implemented already. The whole fuzzy system is divided into two parts according to their functions, that is, the rule chip for fuzzy inference (FP9000) and the defuzzifier chip for defuzzification (FP9001). This functional division facilitates flexible system configuration. The distinctive features of these chips are: high-speed fuzzy logic operation in parallel mode, compact fuzzy systems (chip saving) suitable for built-in application and adaptability of the fuzzy system based on a rule set during execution of fuzzy inference. These design features have allowed an inference speed of more than 1 mega fuzzy logical inference per second, excluding defuzzification. The fuzzy engine is implemented in a parallel architecture where the consequents of all rules in response to the determined antecedents are defined and programmed internally, aggregated by an analog 'or' construction and combined to produce a defuzzified output value. The internal processing in both FP9000 and FP9001 is performed in an analog mode as opposed to other implementation, although a digital interface has been included in the latest version in order to define, modify (write) and read the parameters of each fuzzy rule quickly. The rule chip consists of an antecedent block, a consequent block and a rule memory to store fuzzy rule sets. Up to four fuzzy rules can be stored and processed simultaneously, each with three antecedent variables and one consequent variable. The (t) norm operation applied to the antecedent parts is performed by a min circuit. To describe input fuzzy variables, only S or Z functions are allowed as membership functions. Each membership function circuit can produce up to six different alternate function types, with a total of 31 different center positions within the universe of discourse. The consequent block has four demultiplexer circuits (one for each rule in memory) to decode the single consequent label defined for each fuzzy rule. A three-bit opcode provides seven possible combinations (code 000 is not assigned) labeled NL, NM, NS, Z ,PS, PM, PL for the defined of the consequent center values within the universe of discourse. The consequent block performs a max operation (t conorm). The rule memory supports a digital interface for fuzzy rules definition and application. It is a two-stage memory consisting of 24 8-bit registers (three duplicated antecedents and four inference engines) and four 3-bit registers for the consequent parts of each engine. The defuzzifier chip accepts a set of singletons, one from each rule consequent, which is considered as the centers of the fuzzy outputs. Singletons are applied to increase the

computation speed and decrease the complexity. They calculate a crisp output by the center of gravity method, using the assigned weights for each singleton. The three types of S and Z types membership functions are terms to describe the classes of membership given in figure (4.6), which are standard for hardware implementation.

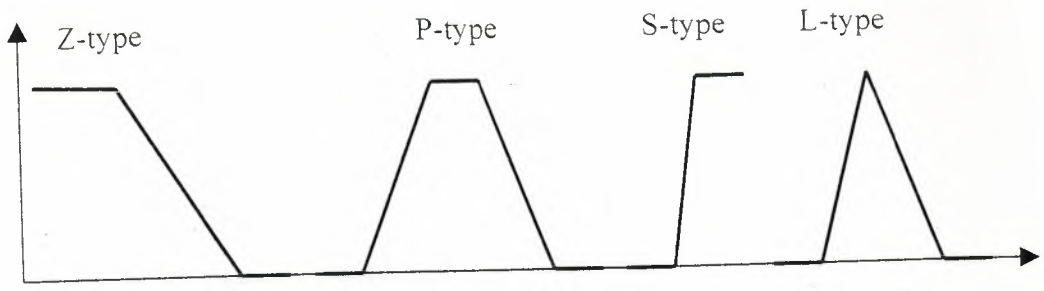


Fig (4.6) standard membership function types.

Both P-type and L-type are minimum combination of Z-type and S-type functions. It is not necessary to apply chips to develop a fuzzy controller. One may construct the whole circuit from simple elements, this can be done layer by layer. Because a fuzzy controller operation includes some stages like fuzzification, fuzzy rules processing. defuzzification, a circuit using it contains parts or layers corresponding to these operations. Each layer realizes one of the operations. For example, consider constructing the simple circuit based on the operational amplifier, which realizes the S-type of membership functions. This circuit is applied in fuzzification. Note that, the Z-type membership function has two parameters: a and b. then it can be defined as:

$$Z(a,b,x) = \min (1, \max (0, Z_0 (a,b,x))).$$

Where

$$Z_0(a,b,x) = \tfrac{1}{2} - (x-a) * b.$$

To realize this membership function the circuit given in figure (4.7) with the symmetric power supply V_{cc} can be proposed. In this case:

$$a = R_2/R_1$$

$$b = \frac{V_{cc}(a + 1)(R_2/R_1)}{a(R_2/R_1 + 1)}.$$

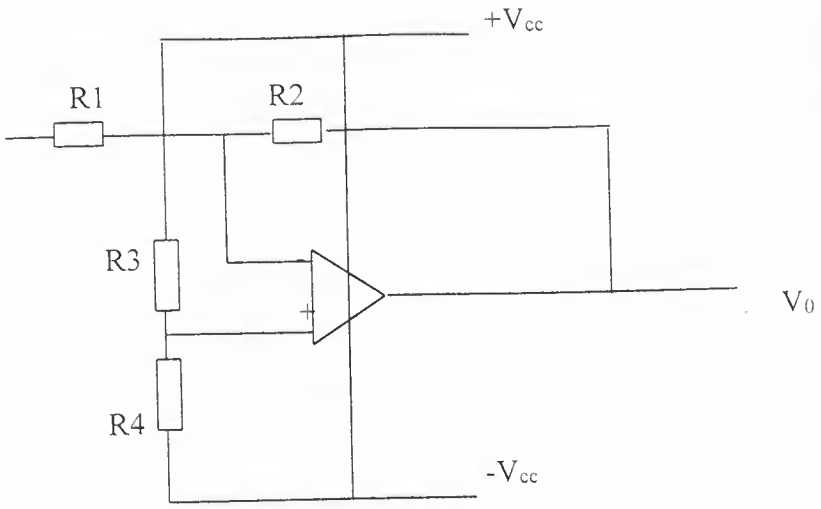


Fig (4.7) Circuit for the implementation of Z-type membership functions.

It is not so that difficult to design analog fuzzy controller if one only has to consider basic design principles. The example of analog reconfigurable fuzzy controller recently developed is given in [Guo94, Guo95]. This controller has a modular architecture and reconfigurable inference engine. It is a two-input and one-output fuzzy controller which implements Mamdani's min-max inference engine and center-of-area defuzzification method. Each input and output includes five membership functions. The controller implements 13 rules. It was designed with analog circuits working in a voltage mode. The design was implemented in 2.4 micron COMS technology.

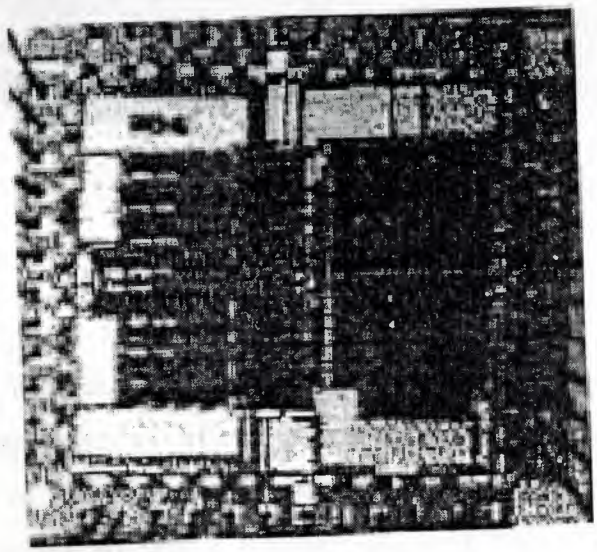


Fig (4.8) A reconfigurable fuzzy controller.

4.5 Integration of Fuzzy and Conventional Control

As fuzzy control has become popular and a number of successful real life applications have been developed, hardware developers have started proposing complex solutions, integrating fuzzy software and hardware with conventional PLC (Programmable Logic Control) and DCS (Distributed Control System) hardware. The aim is to provide an opportunity to complete a design and implement a controller for any particular application without any need for any extra software and hardware. Two ways of transforming fuzzy controller design methodology into real industrial applications are given below. The first one is called UNAC™, which was developed by CICS Automation. It proposes the combination of fuzzy and conventional technology on a 'macro' level. UNAC consists of a design package, including fuzzy control methodology, and a hardware part (SAAC™), figure (4.9), which is downloaded with a controller code developed by the software.

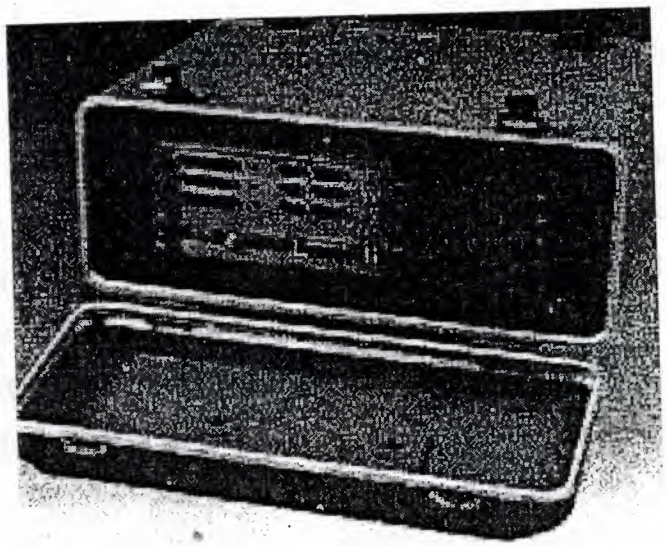


Fig (4.9) SAAC 1000™ portable controller.

The SAAC 1000 uses a DEC Alpha AXPVME™ processor board utilizing a VxWorks™ real-time operating system. It incorporates a VME backplane and supports an extensive range of VME™ cards. Communication with the workstation running a software package is via 10 Mbit/s Ethernet using TCP/IP. High-speed sampling is provided via VME based I/O cards, and other systems can be accessed using RS-232, RS-485 and GPIB. SAAC 1000 may be integrated into DCS OR PLC systems via

communication protocols such as Modbus. This provides SAAC 1000 with access to the DCS/PLC operator interface and plant I/O. and in its turn, it provides the DCS/PLC system with a user-friendly advanced process control facility. The second methodology is the 'micro' level solution, developed by Inform software and Klockner-Moeller, and consists of two chips, field bus connections and interface. An analog ASIC handle the analog/digital interface at industry standard 12-bit resolution. Snap-on modules can extend the periphery for large applications of up to about 100 signals. An integrated field bus connection, based on RS485, provides further extension by networking. The conventional and the fuzzy logic computation are handled by a 16-bit RISC microcontroller. The operating system and communication routines, developed by Klockner-Moeller, are based on a commercial real-time multitasking kernel. The internal RAM of 256 kB can be extended by memory cards using flash technology. Thus the fuzzy PLC™, figure (4.10), is capable of solving real complex problems of industrial automation. The fuzzy PLC is programmed by an enhanced version of the standard fuzzy logic system development software fuzzy TECH™ of inform software. Unlike all other control design packages, fuzzy TECH has been enhanced with editors and functions to support the conventional programming of the PLC. Thus, a user only needs one tool to program both conventional and fuzzy logic parts of the solution. The software runs on a PC and is linked to the fuzzy PLC by a standard serial cable (RS232) or the field bus (RS485). Through this link, the developer downloads the designed system to the fuzzy PLC. Because fuzzy logic systems often require optimization 'on-the-fly', fuzzy TECH and the fuzzy PLC feature 'online debugging', where the system running on the fuzzy PLC is completely visualized by the graphical editors and analyzers of fuzzy TECH. Plus, in online-debugging modes, any modification of the fuzzy logic system is instantly translated to the fuzzy PLC without halting operation.

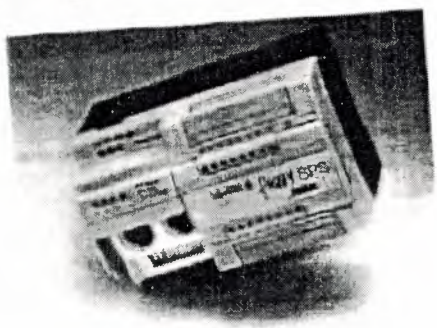


Fig (4.10) Fuzzy PLC™, an integration of fuzzy and automation hardware.

CHAPTER 5

SIMULATION OF FUZZY CONTROL SYSTEM

5.1 Simulation Of Fuzzy Control System For Active Noise Cancellation

There is no more popular problem in telecommunication engineering than one of noise cancellation. One of the methods explored which is called an active noise control (ANC) is producing an anti-noise signal. Adaptive logic, Inc. has developed a solution based on fuzzy controller design and realization. The solution achieves audio noise reduction of over (-20 dB). By adjusting phase and gain values from the measured noise level, the controller reduces significantly the dynamic error measured at the cancellation point.

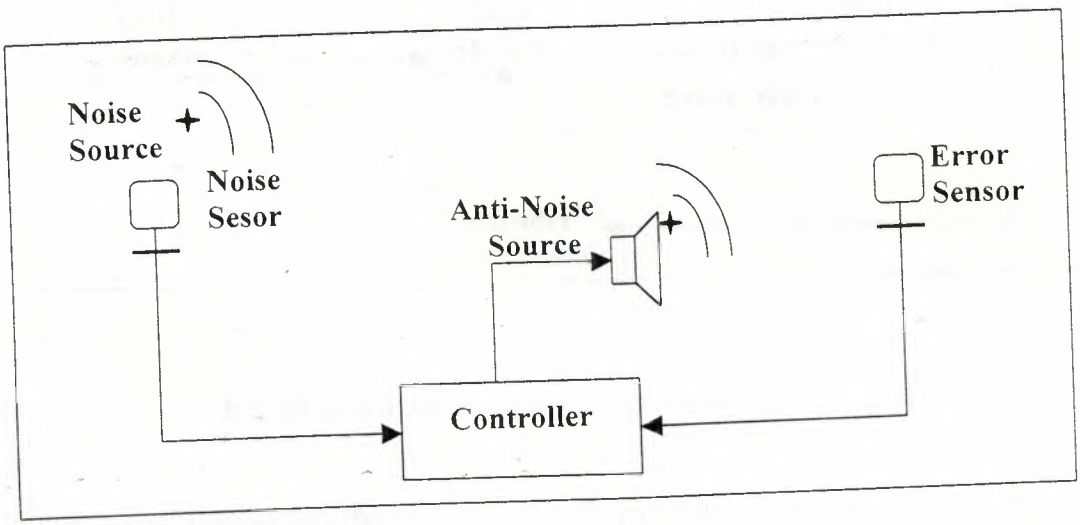


Fig. (5.1) A typical ANC system.

The idea of an ANC is to produce an anti-noise signal coinciding with the noise in phase and magnitude to cancel it out. Figure (5.1) shows a typical ANC system that contains a noise source, noise sensor, error sensor and the control or anti-noise source. The system can be modeled as a closed loop system, where the error is continuously being minimized. Though there are various methods for producing the anti-noise, they all rely on the principle that a signal, when added to the inverse of itself, will produce a null. The problem is that in a time variable world, producing the exact inverse signal

can be very difficult. Additionally, acoustic reflections can add to the original signal producing even more complexity in the system. The exact specifications for the anti-noise signal might be known for some very periodic systems. In these cases, only the phase and amplitude need to be adjusted. An example of such a system might be a large transformer or motor which being driven at (60 Hz), produces (60 Hz) noise and harmonics. For this system, the exact frequency is known so it can be regenerated, phase shifted and amplitude adjusted to make the anti-noise. Figure (5.2) shows a waveform synthesis system.

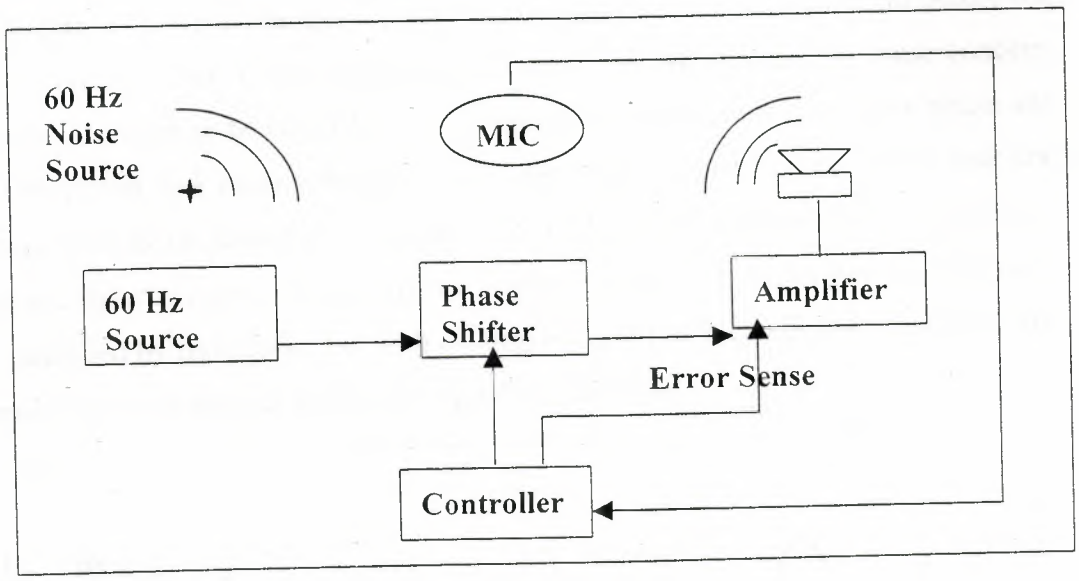


Fig. (5.2) Waveform synthesis for an ANC.

For simplicity in this example, the error signal is RMS value of the noise sensor signal. Response time is critical, because if set too slow, transients will not be seen, and if too fast, a null may never be reached. The controller monitors the error signal, and generates a derivative, so that the direction of the error can be measured. The derivative is simply produced by copying the error signal to an output and then on the following cycle comparing the old stored value at the output (internally feedback) with the new error signal. By monitoring the error and delayed error, the controller can determine if the overall error is improving or getting worse. Since this is the entire controller knows, it must try parameter changes to see how they affect the error. In other words, the controller does not know if the phase should be increased or decreased. It only knows that since an error exists, an adjustment must be made. If after making adjustment to the

phase or amplitude, the error has improved, then the controller makes the same adjustment again. When an increase in the error signal is measured. the controller 'takes back' the last change made, which caused the error to increase. It then begins to adjust the other parameter in the same way before again returning to readjust the first parameter. It will continually switch back-and- forth between parameter adjustments turning them for optimal noise cancellation. By making the adjustment larger when the error is large, and smaller when the error is small the time needed to reach a null can be shorted. Because the dynamics of this system are slow, audio frequencies, it needs to be sure that after making a change, it needs to wait for the system to respond before the change to error is read. If the remaining harmonics are high enough to cause concern, they too might need to be cancelled in a similar manner, but with a separate phase and amplitude circuit. For most systems of this type, only the primary frequency and first harmonic need to be cancelled. The phase accuracy of the anti-noise causes the most change and therefore error in this type of system. Figure (5.3) illustrates a single wave being cancelled by its inverse, but with (5) degrees of error. It can be seen that with only this small degree of error, a significant signal still remains.

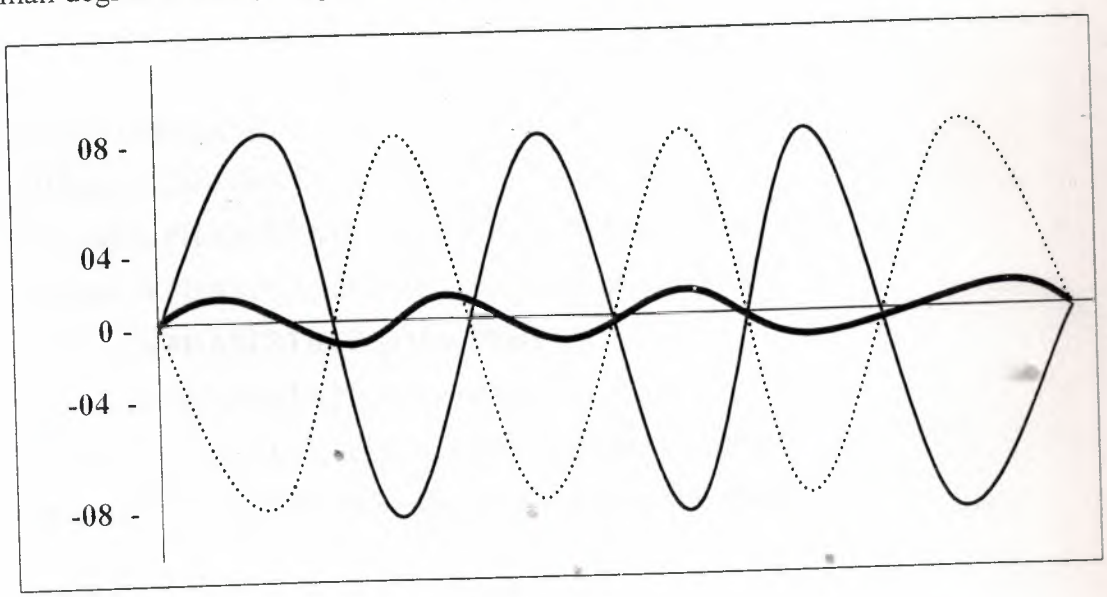


Fig. (5.3) Error caused by a small phase shift.

This problem is only worsened with more complex signals: therefore care must be taken in the phase shifter design. The delay in a given system remains fairly constant over time. This means that the phase shifting circuit need not vary more than the delay would change over a long term, typically less than (45) degrees. So concentrating it to the

narrower region can effectively increase the resolution of the controller. The rule set for this example developed by adaptive logic, Inc. is listed in figure (5.4). This example uses one analog input, Error, and tow analog outputs, Phase and Gain. Tow other outputs; state and ErrorDLY are used for internal variable storage. The state variable forms a state machine that controls the flow of the systems. As discussed earlier, ErrorDLY, is used to determine the direction of error. In other words, it tells the controller if error is getting better or worse. The rules are divided into four sections – one section for each of the tow outputs and tow variable registers. For each sample only one rule in each section can win. That rule then adjusts the output. If no rules are fired, the output does not change. The delayed error, ErrorDLY, is simply the error delayed by one sample time. To solve the terms 'Error is Worse' and 'Error is BetterSame, Error is compared with ErrorDLY. Otherwise, it is Better or the Same. There are four Phase and four Gain states. In either case, state (2) increases the parameter, and state (3) decreases it. States (1 and 4) are used to provide time for the parameter change to settle before checking the response of Error.

Phase control rules section

0. *If* State is Zero *then* Phase = 127.
1. *If* State is PHASESTATES2 and Error is High *then* Phase + 5.
2. *If* State is PHASESTATES3 and Error is High *then* Phase - 5.
3. *If* State is PHASESTATES2 *then* Phase + 1.
4. *If* State is PHASESTATES3 *then* Phase - 1.
5. *If* State is PHASESTATES1 and Error is Worse *then* Phase - 1.
6. *If* State is PHASESTATES4 and Error is Worse *then* Phase + 1.

Gain control rules section

7. *If* State is Zero *then* Gain = 127.
8. *If* State is GainState2 and Error is High *then* Gain + 5.
9. *If* State is GainState3 and Error is High *then* Gain - 5.
10. *If* State is GainState2 *then* Gain + 1.
11. *If* State is GainState3 *then* Gain - 1.
12. *If* State is GainState1 and Error is Worse *then* Gain + 1.

13. *If* State is GainState4 and Error is Worse *then* Gain - 1.

State control rules section

14. *If* State is Zero *then* State = PHASESTATES1.
15. *If* Error is Zero and State is NotZero *then* State = PHASESTATES1.
16. *If* State is PHASESTATES1 and Error is BetterSame *then* State = PHASESTATES2.
17. *If* State is PHASESTATES2 *then* State = PHASESTATES1.
18. *If* State is PHASESTATES1 and Error is Worse *then* State = PHASESTATES3.
19. *If* State is PHASESTATES3 *then* State = PHASESTATES4.
20. *If* State is PHASESTATES4 and Error is BetterSame *then* = PHASESTATES3.
21. *If* State is PHASESTATES4 and Error is Worse *then* State = GainState1.
22. *If* State is GainState1 and Error is BetterSame *then* State = GainState2.
23. *If* State is GainState2 *then* State = GainState1.
24. *If* State is GainState1 and Error is Worse *then* State = GainState3.
25. *If* State is GainState3 *then* State = GainState4.
26. *If* State is GainState4 and Error is BetterSame *then* State = GainState3.
27. *If* State is GainState4 and Error is Worse *then* State = PHASESTATE1.

Error delayed rules section

28. *If* anything *then* ErrorDLY = Error.

Fig. (5.4) Rule set.

To illustrate the efficiency of the fuzzy control methodology, figure (5.5) shows the frequency spectrum of a noise source with many spurious frequencies.

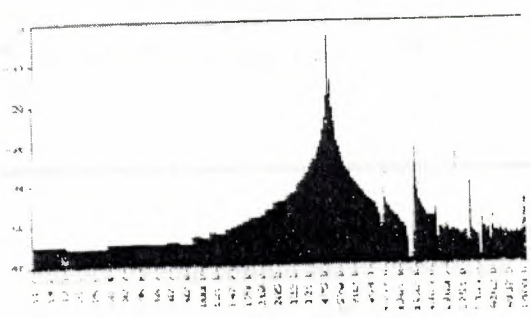


Fig (5.5) Complex noise source frequency spectrum.

The frequencies, though centered around (500 Hz), extend down to (0 Hz) and up over (6 KHz). The noise level at the peak is about (-2 dB). Figure (5.6) shows the cancelled spectrum for this noise source. With the active cancellation, the peak noise level is now at (-25 dB) and the energy contained in the other frequencies is dramatically reduced. With multimode cancellation other frequencies could be further reduced as well.

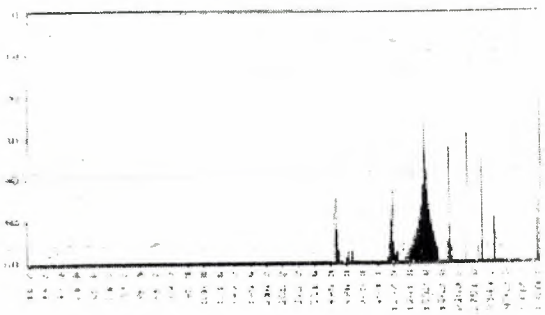


Fig (5.6) Cancelled complex noise frequency spectrum.

5.2 Development of Fuzzy Control System For Power Systems

Today many types of fuzzy controllers are available. Their robustness and reliability make them useful in solving wide range of control problems. For the single input-single output type of system the fuzzy controller shown on the Figure (5.7) can be used. In the Figure (5.7) *kp* and *ki* are the proportional and the integral gain respectively. The fuzzy controller input can be also the derivative of *e* together with signal *E* as in Figure (5.7). In Figure (5.7) the block "fuzzy controller" includes fuzzification of *E*, inference mechanism and defuzzification, so the output *Y* is a crisp value.

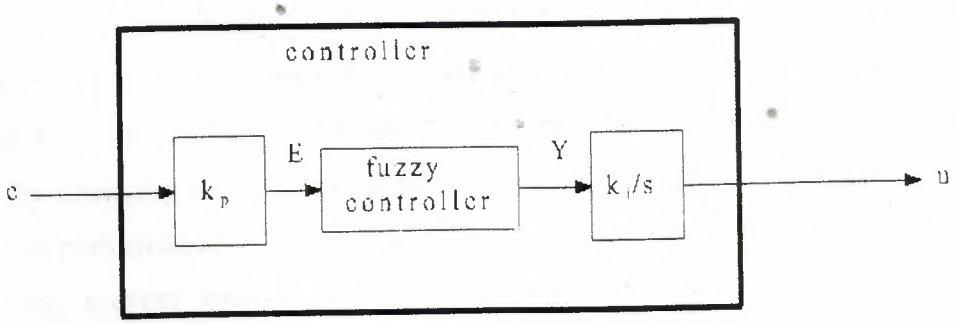


Fig (5.7) The simple fuzzy controller.

For the multi input-multi output system, the fuzzy state controller shown on the

Figure (5.8) can be used.

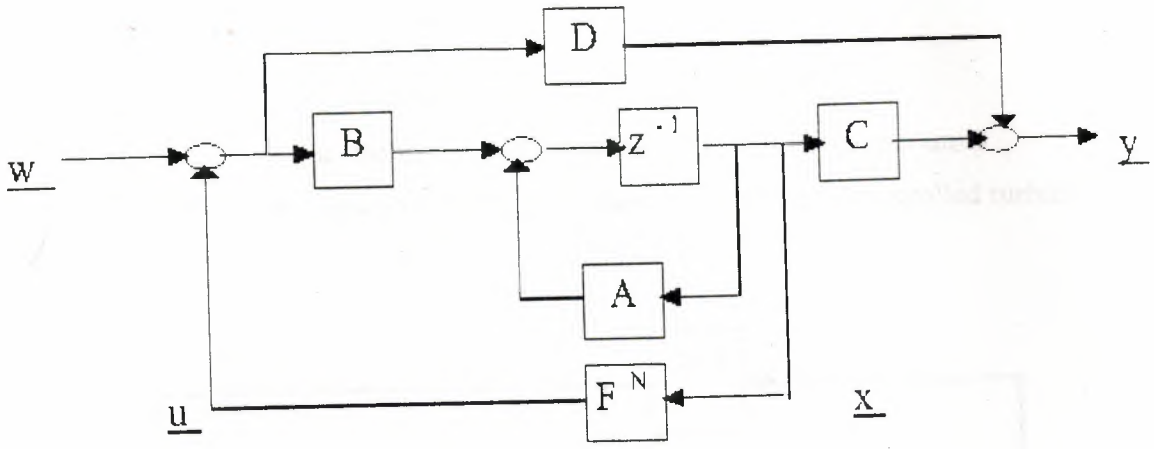


Fig (5.8) The state space fuzzy controller.

In Fig (1.2) *A, B, C, D* are the system matrixes and *FN* is nonlinear function of the state space fuzzy controller. Other vectors in Fig (5.8) are:

$\underline{w}$ - reference input vector.

$\underline{y}$ - output vector.

$\underline{x}$ - state vector.

$\underline{u}$ - input control vector.

The output of the state space fuzzy controller is, similarly to linear state space controller, the vector u:

$$\underline{u} = F^N(\underline{x}) \qquad (1)$$

By using several different inputs into fuzzy controller it is possible to make logical control connection between different variables of the system. Of course, if one wants to make a fuzzy application he must have a sufficient knowledge on how operates the system that is to be controlled. Changing the shape and number of its membership functions, by changing its defuzzification method and its inference mechanism, can influence the performance of the fuzzy controller. 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. Industrial fuzzy controllers available today are also very robust and reliable. This makes them very useful in solving wide range of control problems in power systems. As an illustrative example,

simulation on mathematical model by using software FUZCONTR, as shown in Figure (5.9),

where are:

df - change of system frequency in Hz

u - control signal from the controller in pu (per unit)

R - measure of the static speed drop of the uncontrolled turbine generator in Hz / pu

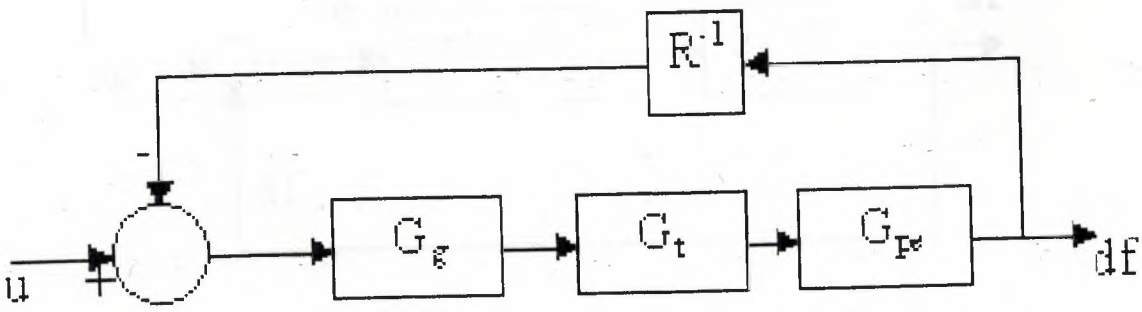


Fig (5.9) The block diagram of the sample power system.

The transfer functions of the system shown in Figure (5.9)

$$G_g = \frac{1}{1 + sT_g} \qquad (2)$$

$$G_t = \frac{1}{1 + sT_t} \qquad (3)$$

$$G_{ps} = \frac{K_{ps}}{1 + sT_{ps}} \qquad 4)$$

Where are:

T_g - time constant of the governor

T_t - time constant of the turbine

T_{ps} - time constant of the power system

K_{ps} - power system gain in Hz / pu

The controller on this power system is connected as shown in Figure (5.10). In Figure (5.10) the block 'system' represents the system shown in Figure (5.9). The block 'controller' represents controller, which will be, in simulation on this mathematical

model, a proportional-integral (PI) controller in one case and, in another, a fuzzy controller shown in Figure (5.7). The variables shown in Figure (5.10) are:

w - frequency reference input in Hz.

df - change of system frequency in Hz.

e - input of the controller in Hz.

u - control signal from the controller in pu.

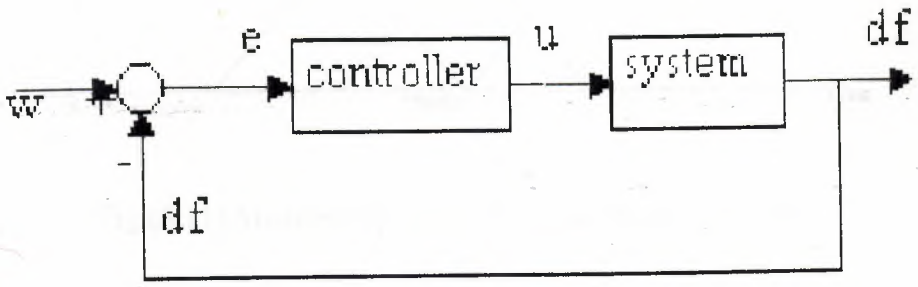


Fig (5.10) Implementation of the controller into system.

Simulations were performed with following set of parameters:

R=2.4 Hz / pu;

K_{ps} =120 Hz / pu;

T_g =0.08 s;

T_t =0.3 s;

T_{ps} =20 s;

The parameters of the conventional proportional-integral discrete controller used in simulation are:

- Sample time T = 0.1 s
- Proportional gain k_{pp} = 0.2
- Integral gain k_{pi} = 0.9

The proportional-integral control law is of course:

$$u(nT)= -k_{pp}e(nT) - k_{pi}\Sigma e(nT) \qquad (5)$$

The structure of fuzzy controller used in simulation is shown in Figure (5.7). The parameters of the fuzzy controller were:

- Proportional gain k_p = 2.2
- Integral gain k_i = 0.5

The membership functions of the fuzzy controller are shown in Figure (5.11).

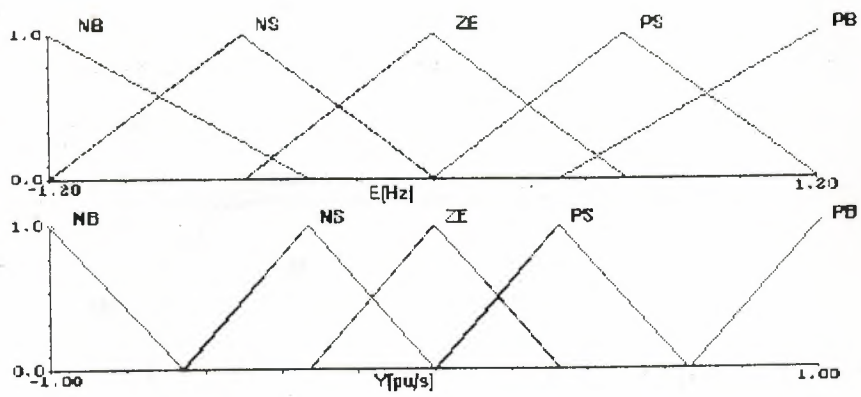


Fig (5.11) Membership functions of the fuzzy controller.

Inference mechanism is realized by five rules:

If E=NB Then Y=PB

If E=NS Then Y=PS

If E=ZE Then Y=ZE

If E=PS Then Y=NS

If E=PB Then Y=NB

Defuzzification is performed by center of gravity method. The response of the system with PI controller on the reference change of 1 Hz is shown on Figure (5.12).

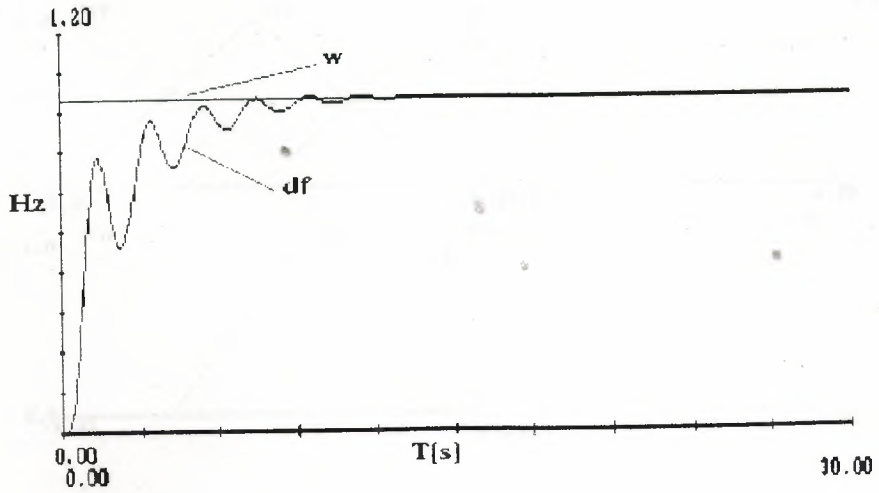


Fig (5.12) The response of the system with PI controller.

The response of the system with fuzzy controller on the reference change of (1 Hz) is shown on Figure (5.13).

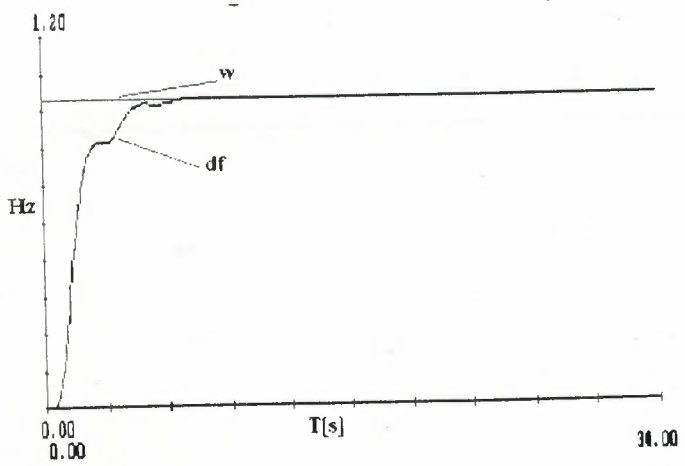


Fig (5.13) The response of the system with fuzzy controller.

If one replaces the membership functions shown in Figure (5.11) with new set of membership functions shown in Fig (5.14) the response of the system will look as in Figure (5.15).

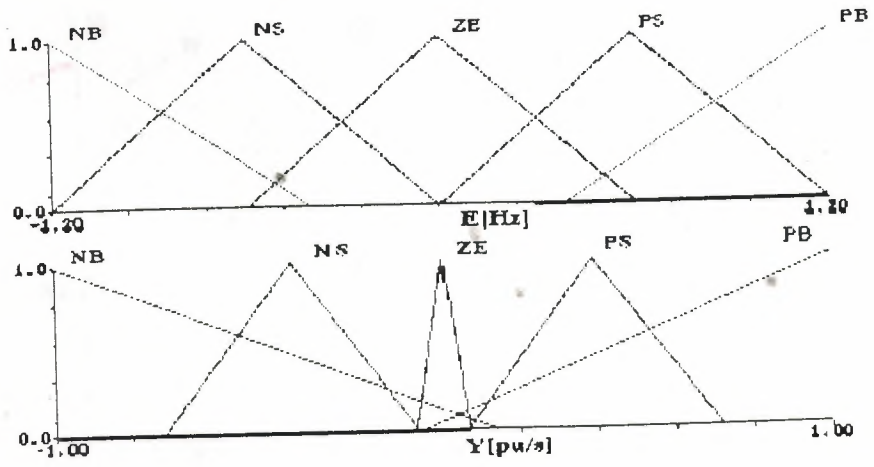


Fig (5.14) The new membership functions.

In Figure (5.16) and Figure (5.17) are shown responses of the system with PI and fuzzy controller respectively in case of reference changing by sinus law.

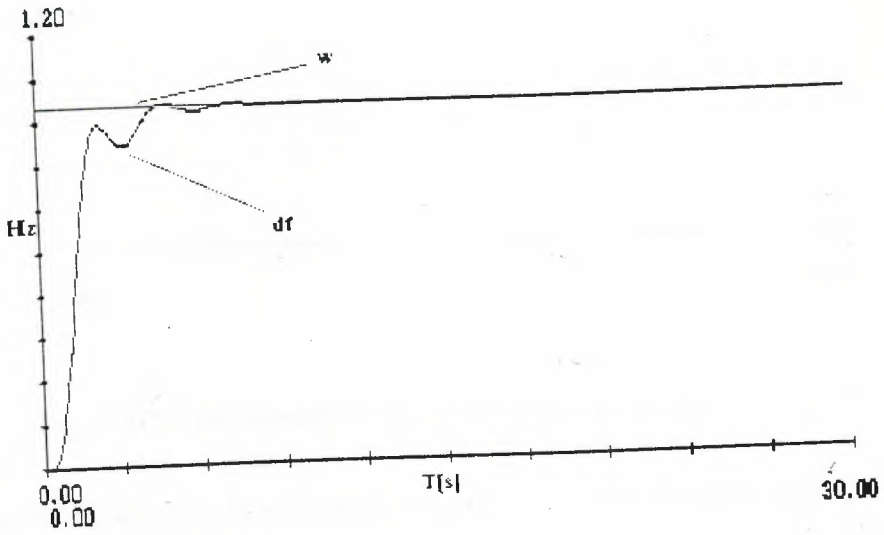


Fig (5.15) The response of the system with new fuzzy controller.

By comparing the responses of the system with PI and with fuzzy controller one can conclude that the fuzzy controller gives better performances. Its response is faster and more accurate than the response of the system with PI controller:

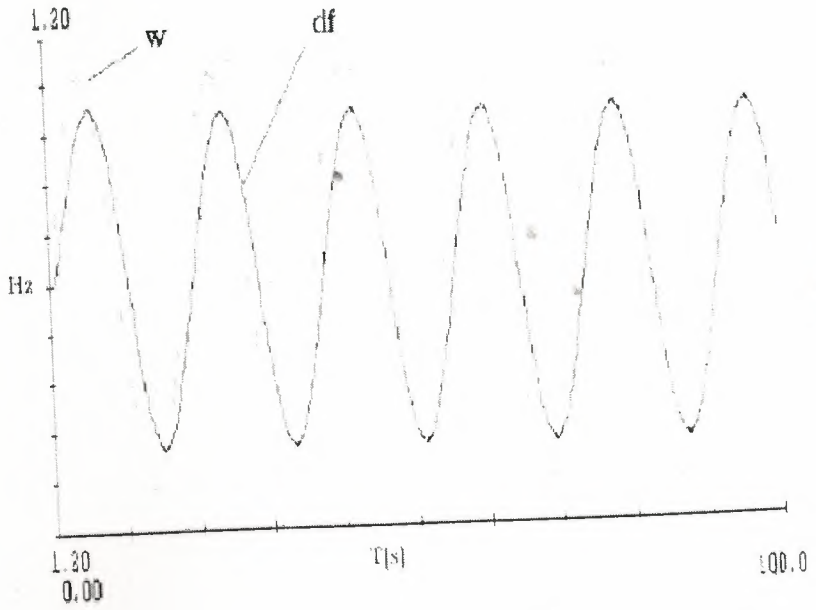


Fig (5.16) The response of the system with PI controller.

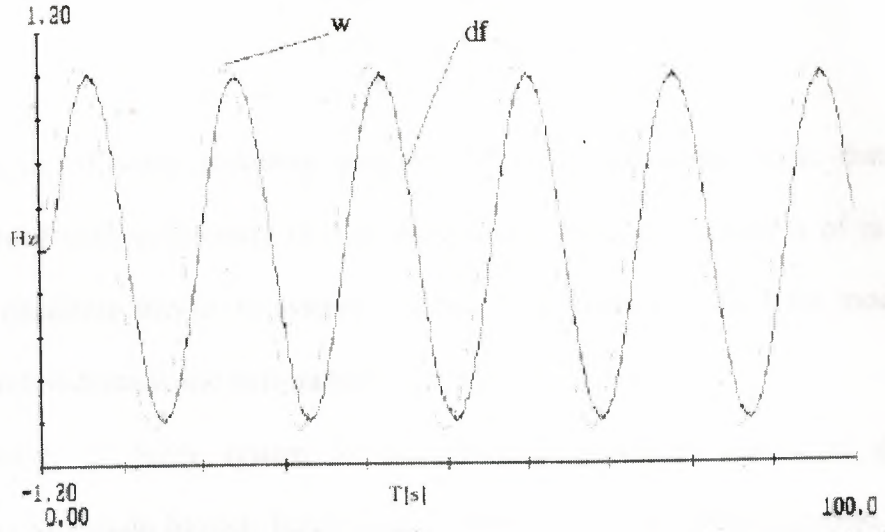


Fig (5.17) The response of the system with new fuzzy controller.

In a method of identifying a process model from plant input-output data has been developed. The model is in the form of qualitative linguistic relationships that are represented and evaluated using fuzzy set theory. This so-called fuzzy identification is used in the design of fuzzy model-based controllers. On-line identification is used to produce an adaptive fuzzy controller.

CONCLUSION

The analysis of some industrial and non-industrial processes show, that they are characterized with uncertainty of their functioning principle, fuzziness of information. In these condition the fuzzy system is effective mathematical tool for modeling and control both industrial and non-industrial processes.

The structure of fuzzy system for technological processes control is given. The functions of its main blocks- fuzzification, inference engine, defuzzification, and fuzzy knowledge base are described.

The development of fuzzy PD-like controller is performed. Using time response characteristics of system and fuzzy model of the processes the fuzzy knowledge base for this controller is developed. The inference engine mechanism is realized by using max-min type fuzzy processing of Zade. Defuzzification mechanism is realized by using "Center of Gravity" Algorithm.

The modeling of fuzzy controller for control of temperature of heater is carried out. The simulation of system is realized in C programming language. In the result of simulation obtained time response characteristics of system show the efficiency of application of fuzzy controller in complicated processes.

REFERENCES

1. Fuzzy Control and Knowledge Engineering in Petrochemical and Robitic Manfacturing, Rafik Aliev, Fuad Aliev and Mamedgassan Baba, Verlag TUV Reheinland.
2. Fuzzy Control, Kevin M. Passino and Stephen Yorkovich, 1998 Addison Wesley Longman, Inc.
3. Fuzzy Controllers, Leonid Reznik, Victoria University of Technology, Melourne< Australia.
4. Abiyev R.H and etc, Controller Based on Fuzzy Neural Network for Control of Technological Process, international conference on Application of Fuzzy System and Soft computing signed, Germany, 1996, June 25-27.
5. Abiyev R.H etc, Determination of Oil and Gas Field Productivity in the Condition of Unsufficient and Fuzziness of Information, second international symposium on mathematical and computational applications, Baku-September 1-3, 1999.
6. Comparison of Fuzzy, Rule-Based, and Conventional Process Control, W.BROCKMANN, Control 1994, IFAC Symposium Valencia, Spain, 3-5 October 1994.
7. The Real Issues in Fuzzy Logic Applications S.J.P, LAUBE, E.F, STARK, Artificial Intelligent in Real Time Control 1994, IFAC Symposium.
8. Tilli T., Automatisierung mit Fuzzy-Logic Franzis-Verlag, München, 1992.
9. Pan C.T., Liaw C.M., An Adaptive Controller for Power System Load-Frequency Control, IEEE, Transactions on Power Systems, 122-127,Vol.4, No. 1, February 1989.

10. Burce Graham and Robert Newell, An Adaptive Fuzzy Model-based Controller. Computer Aided Process Engineering Center, Dept. of Chemical Engineering, The university of Queensland, 4072, Australia, E-mail: bob@cape.uq.oz.au.

11. Burhan T. urk¢ sen _ Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ont., Canada M5S 3G8. Received July 1998.

12. A Corner Matching Algorithm Using Fuzzy Logic, Kilijae Lee, Zeungnam Bien, Korea Advanced Institute of Science and Technology (KAIST), lkj@ctrsys.kaist.ac.kr.

13. Learning From Examples to Compensate Nonlinearities in Electro-Mechanical, Drives: a Fuzzy-Logic Approach, P.J. Costa Branco, J.A. Dente Instituto Superior Tecnico CAUTL/Laboratorio de Mecatronica, pbranco@alfa.ist.utl.pt.

14. Towards a Neural-Based Theory of Emotional Dispositions J.G. Taylor, W.A. Fellenz, R. Cowie, E. Douglas-Cowie.

15. Hydrid Hierarchical Steps and Learning by Abstraction for an Emotion Recognition System B.Apolloni, C.Orovas, G.Palmas (Italy).

16. Emotion Recognition Using Feature Extraction and 3-D Models K. Karpouzis, G. Votsis, G. Moschovitis, S. Kollias (Greece).

17. Refining Rules of Emotion Recognition in Hybrid Systems.Piat, F. and Stamou, G. (Greece).

18. A Fuzzy System for Emotion Classification based on the MPEG-4 Facial Definition

19. Parameter. N. Tsapatsoulis, K. Karpouzis, G. Stamou, F. Piat and S. Kollias (Greece).