NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

FUZZY CONTROL SYSTEM DESIGN

Graduation Project

COM-400

Student: Bülent Özsobacı

Supervisor: ASST.PROF.DR RAHIB ABIYEV

Nicosia - 2001

ACKNOWLEDGEMENT

"First, I would like to pay my respect and thanks to my supervisor, Assist. Prof. Dr Rahib Abiyev for his primary advice and his full belief in my work over the course of this Graduation Project.

Second, I would like to express my gratefulness to Near East University for the erudition that helped me move my life closer to reality.

Third, I thank my family for their continuous support and encouragement during the grounding of this project and for the ardent environment they have provided all these years."

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 structure and operation principle of fuzzy control system are considered. 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 a 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 w*hich deals like* a man w*ith* the *involved linguistic information?* The results of these inquires led to the design of the first fuzzy control systems which implemented in hardware and software a linguistic control algorithm. Such a control algorithm was then formulated by a control engineer on the base of the interviews with human experts who currently work as process operators. The most simple fuzzy feedback control systems contain a fuzzy logic controller (FLC) in the form of a table of linguistic rules, or fuzzy relation matrix and input-output interfaces.

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

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

Chapter 1 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 fuzzy controller are described.

Chapter 2 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 3 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.

Chapter 4 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 ONE: ARCHİTECTURE OF FUZZY CONTROLLER

1.1 Structure Of General Fuzzy System

There are specific components characteristic of a fuzzy controller to support a design procedure, in the block diagram in Figl.1, the controller is between a preprocessing block and a post-processing block. The following explains the diagram block by block.

Input signals entering to the preprocessing unit after scaling and performing some operation are enter to the fuzzification block. On the output of fuzzification block the fuzzy values of input signals are determined. Inference engine using these fuzzy input signal and rule base block made decision. Obtained output signals after defuzzification are entered to the postprocessing unit , where the scaling of the output signal is carried out.

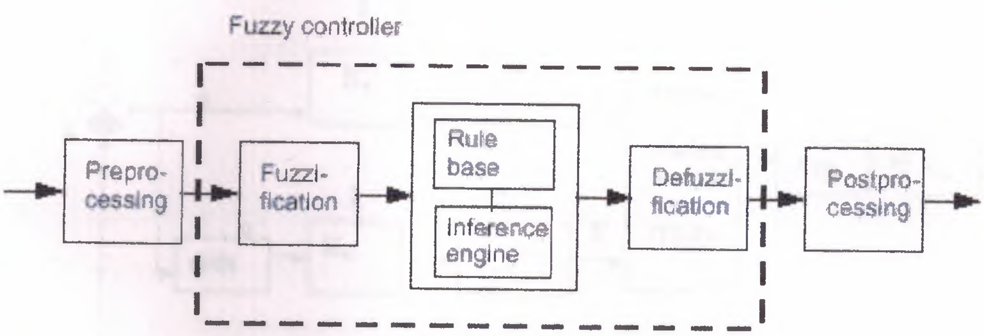


Figure 1.1. Blocks of a fuzzy controller

1.2 Structure of PD-Like Fuzzy controller

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

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

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

A controller (see Fig.1.2)presents and informationloop with:
-an input signal g as an advising set-point(for example, a quality control);
-a comparatorwhich checks, if the emitted process output x is the correct reaction to the set-point g,and which emits himself an error signal e as an input to the decision element TLR,in order to report him,how much the process output x deviatesfrom the preset value of g;
-a decision element TLR which emits for each value of e an output u which, on his side, becomes an input to a process with output x to be controlled.

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

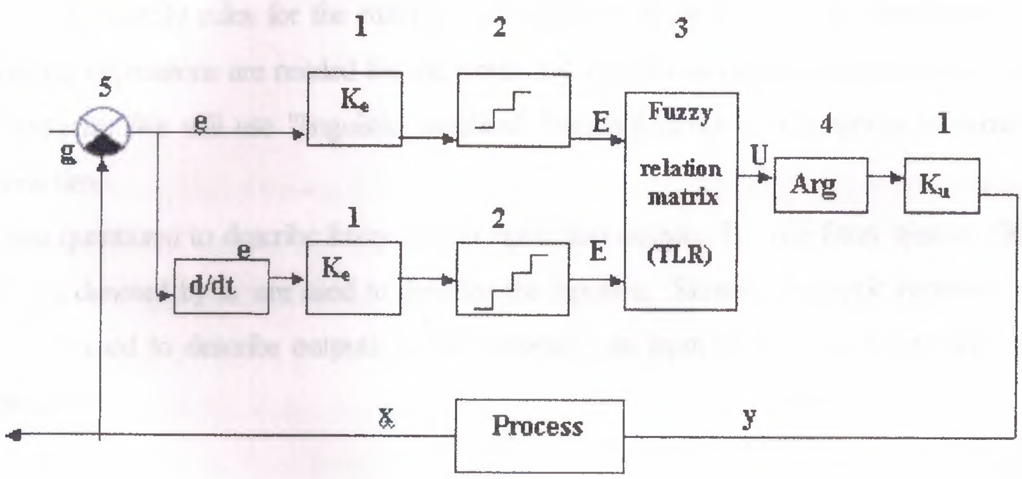


Fig.1.2 A structure of a fuzzy controller

The transient performance demonstrated by these controllers as well as the noise immunity and robustness were essentially better then that of usual PID (Proportional, Integral, Differential) controllers. At the same time,the practical use of fuzzy control systems revealed the following problems:
a.there is not yet a satisfactory approach to the construction of input-output interfaces being sufficientlysupported by logical evidence;

b.there is no definitive agreement about how to proceed with an incomplete table of linguistic rule(TLR).Thus, no actualrule in the TLR can be applied to a concrete decision case,if the features of parameters p of this caseappear nowhere in the TLR as premises.Then, a new consequent c, as the missing term of a new rule r(p,c)must be introduced(this is done,for instance,by interviewing the human process operator). On the other hand ,the broadened TLR demands an expensivestudy of the processand does not guarantee a desirable transient performance of the system in the case of a time variant process.

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

1.3 Fuzzy Analysis

1.3.1Fuzzy Analysis

To specify rules for the rule-base, the expert will use a "linguistic description"; hence, linguistic expressions are needed for the inputs and outputs and the characteristics of the inputs and outputs. We will use "linguistic variables" (constant symbolic descriptions of what are in general time-
varying quantities) to describe fuzzy system inputs and outputs. For our fuzzy system , linguistic variables denoted by u_i are used to describe the inputs u_i. Similarly linguistic variables denoted by y_i are used to describe outputs y_i. For instance , an input to the fuzz system may be y_l = "voltage in."

1.3.2 Linguistic Values

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

$$A_i = \{A_i^j : j=1,2,\ldots\ldots,N_i\}$$

(sometimes for convenience we will let the J indices take on negative integer values, as in the inverted pendulum example where we used the linguistic-numeric values). Similarly, let B_I^j denote the j^{th} linguistic value of
the linguistic variable y_i defined over the universe of discourse Y_i.The linguistic variable y_i takes on elements from the set of linguistic values denoted by

$$B_I=\{B^p_i : p=1,2,\ldots,M_i\}$$

(sometimes for convenience we will let the p indices take on negative integer values). Linguistic values are generally descriptive terms such as "positive large," "zero," and "negative big" (i.e., adjectives). For example, if we assume that u_l denotes the linguistic variable "speed," then we may assign A^1_l = "slow," A^2_l = "medium," and A^3_l = "fast" so that u_l has a value from

$$A_l= \{A_l^1, A_l^2, A_l^3\}.$$

1.3.3 Linguistic Rules

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

If premise **Then** consequent. (2.3)

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

if u_l is A_l^j **and** u_2 is A_2^k **and,....., and** u_n is A^l_n **Then** y_q is $B^p{}_q$ (2.4)

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

If u_1 is A_1^j **and** u_2 is A_2^k **and, . . . , and** u_n is A_n^l **Then** y_1 is B_1^r **and** y_2 is B_2^s is linguistically (logically) equivalent to the two rules

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

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

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

We assume that there are a total of R rules in the rule-base numbered 1, 2… R, and we naturally assume that the rules in the rule-base are distinct (i.e., there are no two rules with exactly the same premises and consequents); however, this does not in general need to be the case. For simplicity we will use tuples

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

to denote the i^{th} MISO rule of the form given in Equation (2.4). Any of the terms associated with any of the inputs for any MISO rule can be included or omitted. For instance, suppose a fuzzy system has two inputs and one output with u_1 = "position," u_2 = "velocity," and y_1 = "force." Moreover, suppose each input is characterized by two linguistic values A_i^1 = "small" and A_i^2 = "large" for $i = 1,2$. Suppose further that the output is characterized by two linguistic values B_1^1 = "negative" and B_1^2 = "positive." A valid If -Then rule could be

If position is large **Then** force is positive

even though it does not follow the format of a MISO rule given above. In this case, one premise term (linguistic variable) has been omitted from the If-Then rule. We see that we allow for the case where the expert does not use all the linguistic terms (and hence the fuzzy sets that characterize them) to state some rules.[6]

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

$$\prod_{i=1}^{n} Ni = N1 * N2 * .... * Nn$$

rules in the rule-base. For example, if $n = 2$ inputs and we have $N_i = 11$ membership functions on each universe of discourse, then there are 11x11 = 121 possible rules. Clearly, in this case the number of rules increases exponentially with an increase in the number of fuzzy controller inputs or membership functions.

CHAPTETR TWO: OPERATİON OF FUZZY CONTROLLER

2.1 Preprocessing

The inputs are most often hard or crisp measurements from some measuring equipment, rather than linguistic. A preprocessor, the first block in Fig. 4, conditions the measurements before they enter the controller Examples of preprocessing are:

- Quantisation in connection with sampling or rounding to integers;
- normalisation or scaling onto a particular, standard range;
- filtering in order to remove noise;
- averaging to obtain long term or short term tendencies;
- a combination of several measurements to obtain key indicators; and
- differentiation and integration or their discrete equivalences.

A quantiser is necessary to convert the incoming values in order to find the best level in a discrete universe. Assume, for instance, that the variable error has the value 4.5, but the universe is :
u = (-5, -4,..,0,..,4,5). The quantiser rounds to 5 to fit it to the nearest level. Quantisation is a means to reduce data, but if the quantisation is too coarse the controller may oscillate around the reference or even become unstable.
Nonlinear scaling is an option (Fig. 5). In the FL Smidth controller the operator is asked

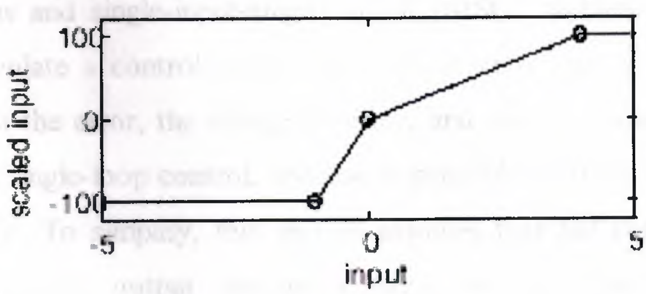


Figure 2.1: Example of nonlinear sealing of an input measurement.

to enter three typical numbers for a small, medium and large measurement respectively (Holmblad & 0stergaard. 1982). They become break-points on a curve that scales the incoming measurements (circled in the figure). The overall effect can be interpreted as a distortion of the primary fuzzy sets. It can be confusing with both scaling and gain factors in a controller, and it makes tuning difficult.

When the input to the controller is error, the control strategy is a static mapping between input and control signal. A dynamic controller would have additional inputs, for example derivatives, integrals, or previous values of measurements backwards in time. These are created in the preprocessor thus making the controller multi-dimensional, which requires many rules and makes it more difficult to design.

The preprocessor then passes the data on to the controller.

2.2 Fuzzification

The first block inside the controller is fuzzification, which converts each piece of input data to degrees of membership by a lookup in one or several membership functions. The fuzzification block thus matches the input data with the conditions of the rules to determine how well the condition of each mie matches that particular input instance. There is a degree of membership for each linguistic term that applies to that input variable.

2.3 Rule Base

The rules may use several variables both in the condition and the conclusion of the niles. The controllers can therefore be applied to both multi-input-multi-output (MIMO) problems and single-input-single-output (SISO) problems. The typical 5150 problem is to regulate a control signal based on an error signal. The controller may actually need both the error, the change in error, and the accumulated error as inputs, but we will call it single-loop control, because in principle all three are formed from the error measurement. To simplify, this section assumes that the control objective is to regulate some process output around a prescribed setpoint or reference. The presentation is thus limited to single-loop control.

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

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

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

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

The top row is the heading. with the names of the variables. It is understood that the two leftmost columns are inputs, the rightmost is the output, and each row represents a rule. This format is perhaps better suited for an experienced user who wants to get an overview of the rule base quickly The relational format is certainly suited for storing in a relational database. It should be emphasised, though. that the relational format implicitly assumes that the connective between the inputs is always logical **and** — or logical **or** for that matter as long as it is the same operation for all rules — and not a mixture of connectives. Incidentally, a fuzzy rule with an or combination of terms can be converted into an equivalent **and** combination of terms using laws of logic (DeMorgan's laws among others). A third format is the tabular linguistic format.

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

This is even more compact. The input variables are laid out along the axes, and the output variable is inside the table. In case the table has an empty cell, it is an indication of a missing rule, and this format is useful for checking completeness. When the input variables are error and change in error, as they are here, that format is also called a linguistic phase plane. in case there are n > 2 input variables involved, the table grows to an n-dimensional array; rather user-un friendly.

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

Lastly, a graphical format which shows the fuzzy membership curves is also possible (Fig. 3). This graphical user-interface can display the inference process better than the other formats, but takes more space on a monitor.

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

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

a and b = min (a. b) minimum

a or b = max (a. b) maximum

or (1)

a and b= a * b algebraic product

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

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

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

$$\textbf{very a} = \mathbf{a}^2 \tag{2}$$

Some examples of other modifiers are

extremely a = $\mathbf{a}^3$
slightly a =$\mathbf{a}^{1/3}$
somewhat a = moreorless a and not slightly a

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

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

If error is Neg and change in error is Pos then output is 0

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

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

Example 1 (standard universes) Many authors and several commercial controllers use standard universes.

- *The FL Smidth controller, for instance, uses the real number interval [-1,1].*
- *Authors of the earlier papers on fuzzv control used the integers in [-6.6].*
- *Another possibiliti' is the interval* [-100, 100] *corresponding to percentages of full scale.*
- *Yet another is the integer range* [0, 4095] *corresponding to the output from a 12 bit analog to digital converter*
- *A variant is* [-2047. 2048], *where the interval is shifted in order to accommodate negative numbers.*

The choice of datatypes may govern the choice of universe. For example, the voltage range [-5,5] *could be represented as an integer range* [-50,50], *or as a floating point range* [-5.0, 5.0], *a signed byte datatvpe has an allowable integer range* [-128, 127].

A way to exploit the range of the universes better is scaling. If a controller input mostly uses just one term, the scaling factor can be turned up such that the whole range is used. An advantage is that this allows a standard universe and it eliminates the need for adding more terms.

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

The designer is inevitably faced with the question of how to build the term sets. There are two specific questions to consider:

(i) How does one determine the shape of the sets? and **(ii)** How many sets are necessary and sufficient? For example, the error in the position controller uses the family of terms

Neg, Zero, and Pos. According to fuzzy set theory the choice of the shape and width is subjective, but a few rules of thumb apply.

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

A preliminary answer to questions (i) and (ii) is that the necessary and sufficient number of sets in a family depends on the width of the sets, and vice versa. A solution could be to ask the process operators to enter their personal preferences for the membership curves; but operators also find it difficult to settle on particular curves. The manual for the TIL Shell product recommends the following (Hill, Horstkotte & Teichrow, 1990).

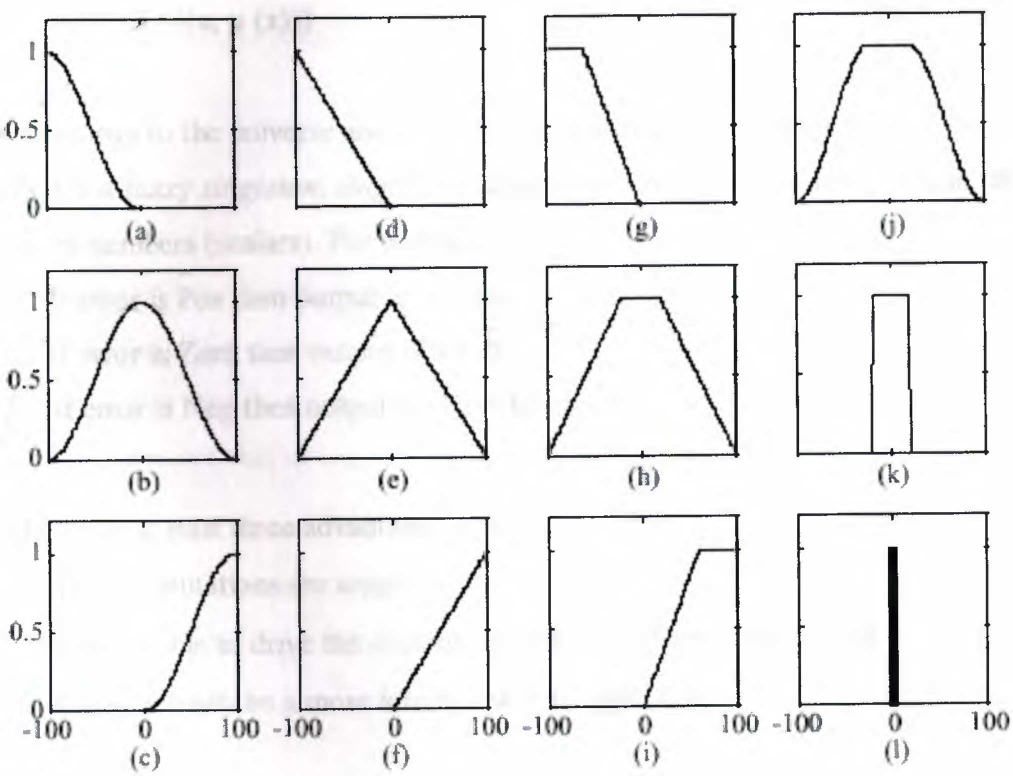


Figure 2.2: Examples of membership functions. Read from top to bottom, left to right: (a) s—function, (b) π— function, (c) z—function, (d-f) triangular versions, (g-i) trapezoidal versions, (j) flat π— function. (k) rectangle. (I) singleton.

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

Membership functions can be flat on the top, piece-wise linear and triangle shaped, rectangular, or ramps with horizontal shoulders. Fig. 2 shows some typical shapes of membership functions.

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

$$A=\{(x, \mu (x))\} \tag{3}$$

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

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

There are at least three advantages to this:

- The computations are simpler;
- it is possible to drive the control signal to its extreme values; and
- it may actually be a more intuitive way to write rules.

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

Example 2 (membership functions) *Fuzzy controllers use a variety of membership functions. A common example of a function that produces a bell curve is based on the exponential function,*

$$\mu(x) = \exp\left[\frac{-(x-x_0)^2}{2\sigma^2}\right] \tag{4}$$

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

$$\mu(x) = \left[1+\left(\frac{x-x_0}{\sigma}\right)^2\right]^{-1} \tag{5}$$

The FL Smidth controller uses the equation

$$\mu(x) = 1-\exp\left[-\left(\frac{\sigma}{x-x_0}\right)^a\right] \tag{6}$$

The extra parameter a controls the gradient of the sloping sides. It is also possible to use other functions, for example the sigmoid known from neural networks.
A cosine function can be used to generate a variety of membership functions. The s-curve can be implemented as

$$s(x_l, x_r, x) = \begin{cases} 0 & x < x_l \\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-x_r}{x_r-x_l}\right) & x_l \le x \le x_r \\ 1 & x > x_r \end{cases} \tag{7}$$

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

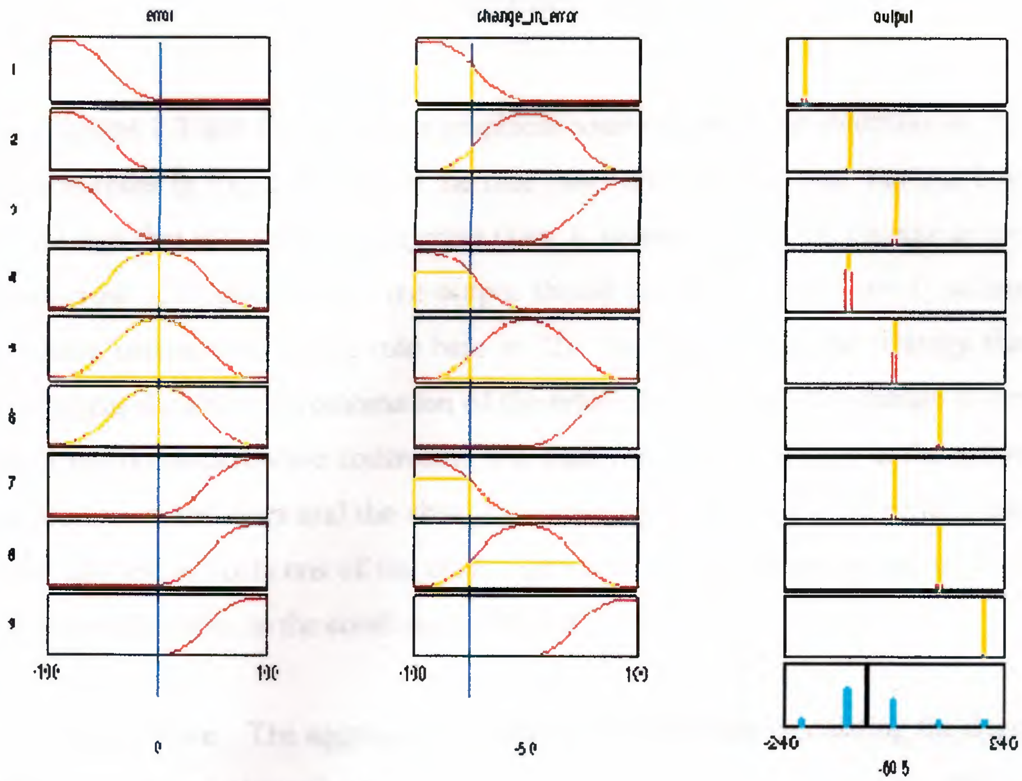


Figure2.3 :Graphical construction of the control signal in a fuzzy PD controller(generated in the Matlab Fuzzy Logic Toolbox).

$$z(x_l, x_r, x) = \left\{ \begin{array}{ll} 1 & x < x_l \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x - x_l}{x_r - x_l} * \pi\right) & x_l \le x \le x_r \\ 0 & x > x_r \end{array} \right\} \qquad (8)$$

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

$$\pi(x_1, x_2, x_3, x_4, x) = \min((x_1, x_2, x), z(x_3, x_4, x)) \qquad (9)$$

2.4 Inference Engine

Figures 2.3 and 2.4 are both a graphical construction of the algorithm in the core of the controiler In Fig. 2.3. each of the nine rows refers to one rule. For example, the first row says that if the error is negative (row 1, column 1) and the *change in error* is negative (row 1, column 2) then the output should be negative big (row 1, column 3). The picture corresponds to the rule base in (2). The rules reflect the strategy that the control signal should be a combination of the reference error and the change in error, a fuzzy proportional-derivative controller. We shall refer to that figure in the following. The instances of the error and the change in error are indicated by the vertical lines on the first and second columns of the chart. For each rule, the inference engine looks up the membership values in the condition of the rule.

Aggregation The aggregation operation is used when calculating the degree of *fulfillment or firing strength ak* . of the condition of a rule *k.* A rule, say *rule 1, will* generate a fuzzy membership value µe1 coming from the error and a membership value µce1 coming from the *change* in error measurement. The aggregation is their combination,

$$\mu_{e1} \textbf{ and } \mu_{ce1} \tag{10}$$

Similarly for the other rules. Aggregation is equivalent to fuzzification, when there is only one input to the controller. Aggregation is sometimes also called *fulfilment* of the rule or *firing strength.*

Activation The *activation* of a rule is the deduction of the conclusion, possibly reduced by its firing strength. Thickened lines in the third column indicate the firing strength of each rule. Only the thickened part of the singletons are activated, and **min** or product (*) is used as the *activation operator.* It makes no difference in this case, since the output membership functions are singletons, but in the general case of s—. π—, and z— functions in the third column, the multiplication scales the membership curves, thus preserving the initial shape, rather than clipping them as the **min** operation does. Both methods work well in general, although the multiplication results in a slightly smoother control signal. In Fig. 2.3, only rules four and five are active.

A rule k can he weighted a priori by a weighting factor $\omega k \in [0,1]$. which is its *degree of confidence*. In that case the firing strength is modified to

$$\alpha_k = \omega_k * \alpha_k \tag{11}$$

The degree of confidence is determined by the designer, or a learning program trying to adapt the rules to some input-output relationship.

Accumulation All activated conclusions are *accumulated*, using the **max** operation, to the final graph on the bottom right (Fig. 2.3). Alternatively, **sum** accumulation counts overlapping areas more than once (Fig. 2.4). Singleton output (Fig. 2.3) and **sum** accumulation results in the simple output

$$\alpha_1 * s_1 + \alpha_2 * s_2 + \dots\dots\dots + \alpha_n * s_n \tag{12}$$

The alpha's are the firing strengths from the n rules and $s_1, \ldots s_n$, are the output singletons. Since this can be computed as a vector product, this type of inference is relatively fast in a matrix oriented language.

There could actually have been several conclusion sets. An example of a one-input-two-outputs rule is "if $é_a$ is **a** then o_1 is **x** and o_2 is **y**" The inference engine can treat two (or several) columns on the conclusion side in parallel by applying the firing strength to both conclusion sets. In practice, one would often implement this situation as two rules rather than one, that is, "If $é_a$ is **a** then o_1 is **x**", "If $é_a$ is **a** then o_2 is **y**".

2.5 Defuzzyfication

The resulting fuzzy set (Fig. 2.3, bottom right; Fig. 2.4, extreme right) must be converted to a number that can be sent to the process as a control signal. This operation is called *defuzzification*, and in Fig. 2.4 the x-coordinate marked by a white, vertical dividing line becomes the control signal. The resulting fuzzy set is thus defuzzified into a crisp control signal. There are several defuzzification methods.

Centre of gravity (COG) The crisp output value u (white line in Fig.2.4) is the abscissa under the centre of gravity of the fuzzy set,

$$u = \frac{\sum_i \mu(x_i)x_i}{\sum_i \mu(x_i)} \quad (13)$$

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

Centre of gravity method for singletons (COGS) If the membership functions of the conclusions are singletons (Fig. 2.3), the output value is

$$u = \frac{\sum_i \mu(s_i)s_i}{\sum_i \mu(s_i)} \quad (14)$$

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

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

$$u = \left\{ x \,\middle|\, \int_{Min}^{x} \mu(x)dx = \int_{x}^{Max} \mu(x)dx \right\} \quad (15)$$

Here x is the running point in the universe, μ (x) is its membership.

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

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

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

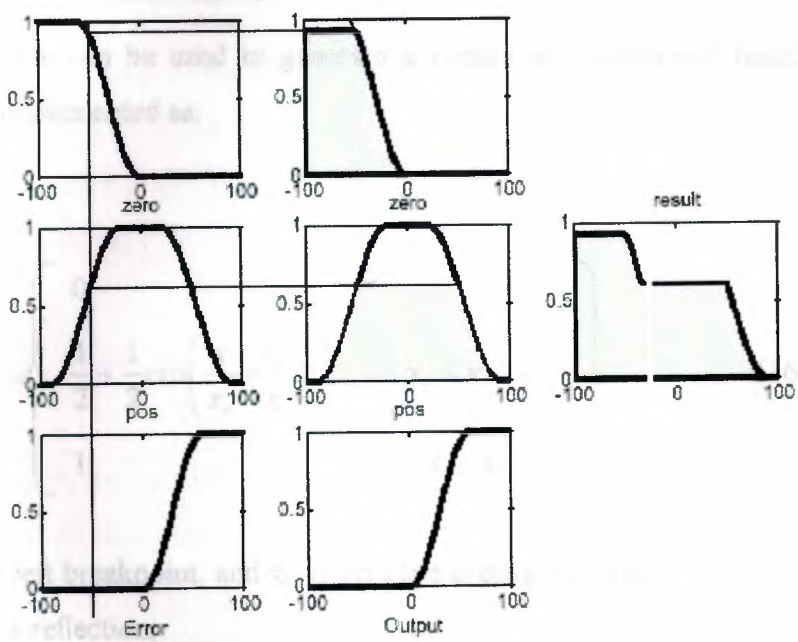


Figure2.4:One input, one output rule base with non-singleton output sets.

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

2.6 Postprocessing

Output scaling is also relevant. In case the output is defined on a standard universe this must be scaled to engineering units, for instance,volts meters, or tons per hour. An example is the scaling from the standard universe [-1,1] to the physical units [-10,10] volts.

The postprocessing unit often contains an output gain that can be tuned , and sometimes also an integrator.

Example 3(inference) *How is the inference in Fig. 8 implemented using discrete fuzzy sets?*

Behind the scene all universes were divided into 201 points from −100 to100. but for brevity , let us just use five points. Assume the universe ***u***, common to all variables, is the vector

U=	*-100*	*-50*	*0*	*50*	*100*

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

$$s(x_l, x_r, x) = \begin{cases} 0 & x < x_l \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x - x_r}{x_r - x_l}\pi\right) & x_l \le x \le x_r \\ 1 & x > x_r \end{cases} \qquad (16)$$

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

$$z(x_l, x_r, x) = \begin{cases} 1 & x < x_l \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x - x_l}{x_r - x_l} * \pi\right) & x_l \le x \le x_r \\ 0 & x > x_r \end{cases} \qquad (17)$$

Then the π-curve (see for example Fig. 2.2 (j)) can be implemented as a combination of the s-curve and the z-curve, such that the peak is flat over the interval [x2,x3]

$$\pi(x_1,x_2,x_3,x_4,x)= \min\left(s(x_1,x_2,x),z(x_3,x_4,x)\right) \tag{18}$$

A familiv of terms is defined bv means of the π—function, such that

$\mathbf{neg} = \pi(-100,-100,-60,10,\mathbf{u}) =$

1	0.95	0.05	0	0

$\mathbf{zero} = \pi(-90,-20,20,90,\mathbf{u}) =$

0	0.61	1	0.61	0

$\mathbf{pos} = \pi(-10,60,100,100,\mathbf{u}) =$

0	0	0.05	0.95	1

Above we inserted the whole vector **u** *in place of the running point x; the result is thus a vector. The figure assumes that error = -50 (the unit is percentages of full range). This corresponds to the second position in the universe, and the first rule contributes with a membership* **neg***(2) = 0.95. This firing strength is propagated to the conclusion side of the rule using* **min***, such that the contribution from this rule is*

$0.95\ \mathbf{min}\ \mathbf{neg} =$

0.95	0.95	0.05	0	0

The activation operation was **min** *here. Apply the same pmcedure to the two remaining rules, and stack all three contributions on top of each other,*

0.95	0.95	0.05	0	0
0	0.61	0.61	0.61	0
0	0	0	0	0

To find the accumulated output set, perform a **max** *operation down each column. The result is the vector*

$0.95\ \mathbf{min}\ \mathbf{neg} =$

0.95	0.95	0.05	0	0

The centre of gravity method yields

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \tag{19}$$

$$= \frac{0.95*(-100)+0.95*(-50)+0.61*0+0.61*50+0*100}{0.95+0.95+0.61+0.61+0} = \tag{20}$$

$$= -35.9$$

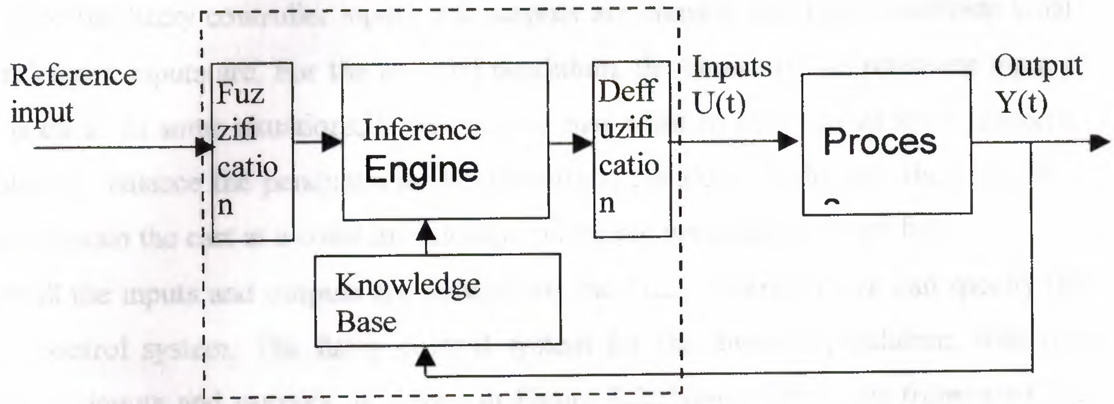
(21)

which is the control signal (before postprocessing).

CHAPTER THREE: DEVELOPMENT OF FUZZY CONTROL SYSTEM

3.1. Fuzzy Control System Architecture

We introduce each of the components of the fuzzy controller for a simple problem of balancing an inverted pendulum on a cart, as shown in Figure 3.1. 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). We will use r to denote 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<>0). This is a very simple and academic nonlinear control problem, and many good techniques already exist for its solution..



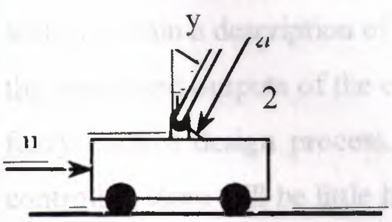


Fig 3.1 Inverted pendulm on a cart.

3.2 Choosing Fuzzy Controller Inputs and Outputs

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 (this could be you!) says that she or he will use

$$E(t)=r(t)-y(t) \text{ and } de(t)/dt$$

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, we must identify the controlled variable. For the inverted pendulum, we are 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, you must determine what the reference inputs are. For the inverted pendulum, the choice of the reference input r = 0 is clear. In some situations, however, you may want to 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, we can specify the fuzzy control system. The fuzzy control system for the inverted pendulum, with our choice of inputs and outputs, is shown in Figure 3.2. Now, *within this framework* we seek to obtain a description of how to control the process. We see then that 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. It must be understood that the choice of the controller inputs and outputs is a fundamentally important part of the control design process.

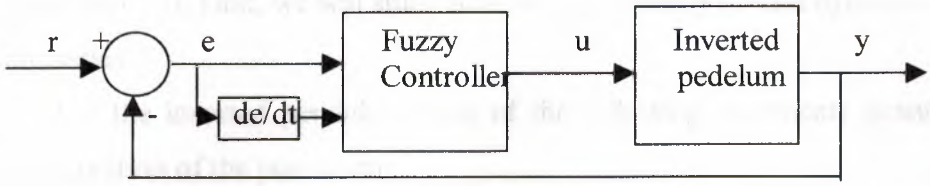


Fig. 3.2 Fuzzy controller for an inverted pendulum on a cart.

3.3 Linguistic Descriptions Of Knowledge

Suppose that the human expert shown provides a description of how best to control the plant in some natural language (e.g., English). We seek to take this "linguistic" description and load it into the fuzzy controller, as indicated by the arrow in Figure 3.2

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)/dtt;*

"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"

Note that we are using "negsmall" as an abbreviation for "negative small in size" and so on for the other variables. Such abbreviations help keep the linguistic descriptions short yet precise. 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. First, we will study how we can quantify certain dynamic behaviors with linguistics.

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 "negsmall").
- 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).

3.3.1 Rules

Next, we will use 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 3.3, we have the following rules

1. ***If*** *error is neglarge* ***and*** *change-in-error is neglarge* ***Then*** *force is poslarge*

This rule quantifies the situation in Figure 3.3(a) 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 Figure3.3(b) 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 3.3(c) 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.

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

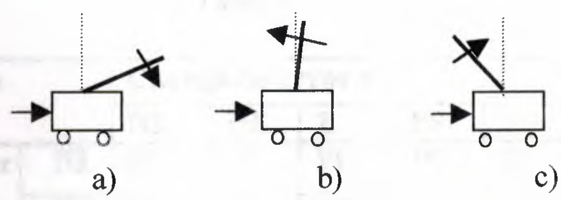


Fig 3.3

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

As you can see 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.

3.3.2 Rule-Bases:

Using the above approach, we could continue to write down rules for the pendulum problem for all possible cases. Note that since we only specify a finite number of linguistic variables and linguistic values, there is only a finite number of possible 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 1. Notice that 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 (where the "2" represents the row having "2" for a numeric-linguistic value and the "-1" represents the column having "-1" for a numeric-linguistic value) 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

which is rule 3 above. Table 1 represents abstract knowledge that the expert has about how to control the pendulum given the error and its derivative as inputs.

Table 1

force u		**Change-in-error e'**				
		NL	NS	Z	PS	PL
Error e	NL	PL	PL	PL	PS	Z
	NS	PL	PL	PS	Z	NS
	Z	PL	PS	Z	NS	NL
	PS	PS	Z	NS	NL	NL
	PL	Z	NS	NL	NL	NL

Notice the diagonal of zeros and viewing the body of the table as a matrix we see that it has a certain symmetry to it. This symmetry that emerges when the rules are tabulated is no accident and is actually a representation of abstract knowledge about how to control the pendulum; it arises due to a symmetry in the system's dynamics.

3.4 Fuzzy Quantification of Knowledge

3.4.1 Membership Functions

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

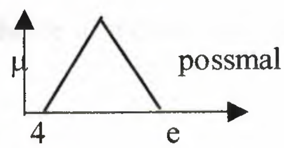


Fig 3.4

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 Figure3.4 is only one possible definition of the meaning of "error is possmall"; you could use a bell-shaped function, a trapezoid, or many others.

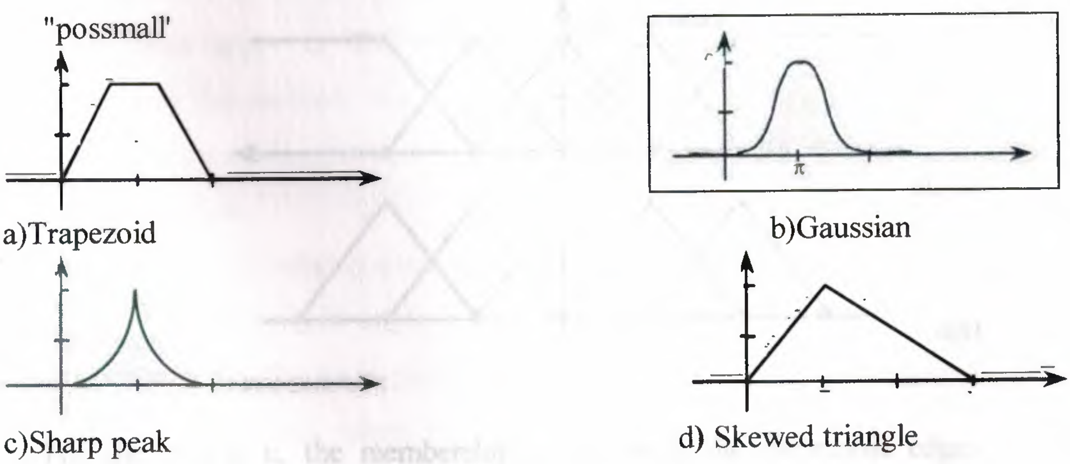


Fig3.5

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 3.6. 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.

Now that we know how to specify the meaning of a linguistic value via a membership function (and hence a fuzzy set), we can easily specify the membership functions for all 15 linguistic values (five for each input and five for the output) of our inverted pendulum example. Figure3.7 for one choice of membership functions.

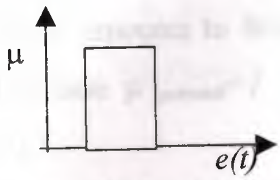


Fig.3.6 Membership function for a crisp set.

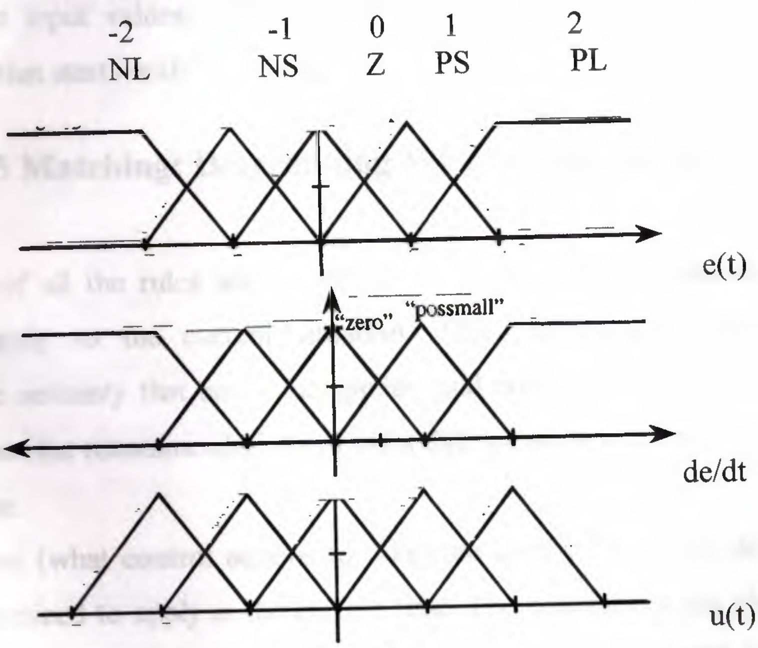


Fig3.7

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. We do not generally indicate to a process actuator, "any value bigger than, say, 10, is acceptable."

The rule-base of the fuzzy controller holds the linguistic variables, linguistic values, their associated membership functions, and the set of all linguistic rules (shown in Table 1), so we have completed the description of the simple inverted pendulum.

3.4.2 Fuzzification

It is actually the case that for most fuzzy controllers the fuzzification block in Figure 1 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 $\mu_{possmal}=1$ (with all others zero) and $\mu_{zero}(de(t)/dt)= \mu_{possmal}(de(t)/dt)=0.5$.

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

3.5 Matching: Determining Which Rules to Use

1. The premises of all the rules are compared to the controller inputs to determine which rules apply to the current situation. This "matching" process involves determining the certainty that each rule 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.

3.6 Premise Quantification via Fuzzy Logic

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 Figure 9, 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, we had quantified the meaning of the linguistic terms "error is zero" and "change-in- error is possmall" via the membership functions shown in Figure 8. Now we 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. While we could use standard Boolean logic to combine these linguistic terms, since we have quantified them more precisely with fuzzy sets (i.e., the membership functions), we can use these.

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 Figure3.7 (or Figure3.8) we see that

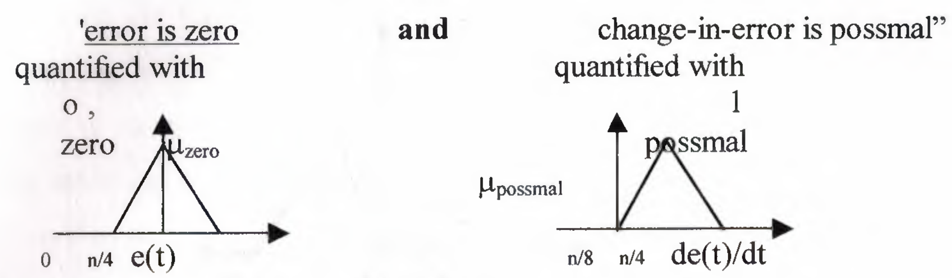


Fig.3.8

μ_{zero} (e(t))=0.5 **and** $\mu_{possmal}$(de/dt)=0.5

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

"error is zero **and** change-in-error is possmall"

that is the premise from the above rule? We will denote this certainty by $\mu_{premise}$. There are

actually several ways to define it:

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

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

Notice that both ways of quantifying the "and" operation in the premise indicate that you can be no more certain about the conjunction of two statements than you are about the individual terms that make them up (note that $0 \leq \mu_{premise} \leq 1$ for either case)..

While we have simply shown how to quantify the "and" operation for one value of *e(t)* and *de(t)/dt,* if we consider all possible e(t) and *de(t)/dt* values, we will obtain 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 we choose the minimum operation to represent the "and" in the premise, then we get the multidimensional membership function $\mu_{premise}(e(t),de(t)/dt)$. Notice that if we pick values for *e(t)* and *de(t)/dt,* the value of the premise certainty $\mu_{premise}(e(t),de(t)/dt)$ represents how certain we are 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 Figure3.9, and we become less or more certain of the applicability of this rule.

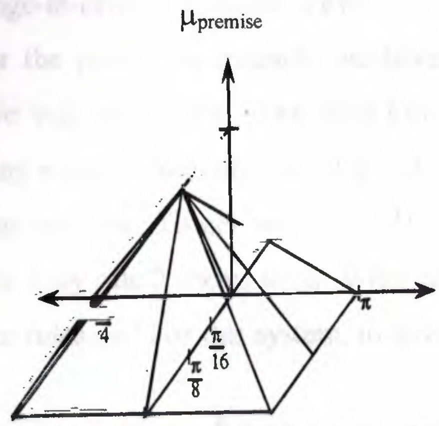


Fig.3.9 Membership function of the premise for a single rule.

3.7 Determining Which Rules Are On

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

Consider, for the inverted pendulum example, how we compute the rules that are on. Suppose that

e(t)=0 and de(t)/dt=n/8-n/32(=0.294)

Figure 11 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 we see that $\mu_{zero}(de(t)/dt)=0.25$ *and* $\mu_{possmal}(de(t)/dt) = 0.75$ and that all the other membership functions are off. This implies that rules that have the premise terms

"error is zero"

"change-in-error is zero"

"change-in-error is possmall"

are on (all other rules have $\mu_{premise}(e(t),de(t)/dt) = 0$. So, which rules are these? Using Table 1 on, we find that 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 we have at most two membership functions over- lapping, we will never have more than four rules on at one time (this concept generalizes to many inputs). Actually, for this system we will either have one, two, or four rules on at any one time. To get only one rule on choose, for example, $e(t) = 0$ and $de(t)/dt=\pi/8$ so that only rule 2 above is on. What values would you choose for e(t) and de(t)/dt to get four rules on? For this system, to have exactly three rules on?

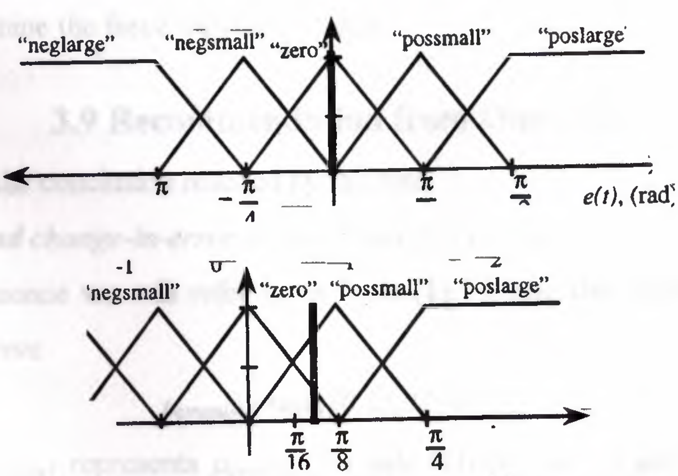


Fig 3.10.Input membeship functions with input values

It is useful to consider pictorially which rules are on. Consider Table 2, which is a copy of Table 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 we perturbed e(t) slightly positive (negative), then we would have the two rules below (above) the two highlighted ones on also.

Table 2

force u		Change-in-error e'				
		NL	NS	Z	PS	PL
Error e	NL	PL	PL	PL	PS	Z
	NS	PL	PL	PS	Z	NS
	Z	PL	PS	Z	NS	NL
	PS	PS	Z	NS	NL	NL
	PL	Z	NS	NL	NL	NL

3.8 Inference Step: Determining Conclusions

Next, we 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, we will first consider the recommendations of each rule independently. Then later we will combine all the recommendations from all the rules to determine the force input to the cart.

3.9 Recommendation from One Rule

Consider the conclusion reached by the rule

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

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

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

(the notation $\mu_{premise1}$ represents $\mu_{premise}$ for rule (1)) so that we are 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 we should not apply any force since this would tend to move the pendulum away from the vertical). The membership function for this consequent is shown in Figure 3.11(a). The membership function for the conclusion rea_ched by rule (1), which we denote by μ_l, is shown in Figure 3.11(b) and is given by

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

this membership function defines the "implied fuzzy set"[3] 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 we *can be no more certain about our consequent than our premise*.

Notice that the membership function $\mu_l(u)$ 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 $\mu_l(ut)$. 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* $\mu_l(u)$ obtained (i.e., it will chop off the top at different points).

We see that $\mu_l(u\ )$ 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 Figure 3.11(b)), and it indicates that it is certain that the force input should not be too large in either the positive or negative direction-this makes sense if you consider the linguistic meaning of the rule. The membership function $\mu_l(u)$ quantifies the conclusion reached by only rule (1) and only for the current $e(t)$ and $de(t)/dt$. It is important that the reader be able to picture how the shape of the implied fuzzy set changes as the rule's premise certainty changes over time.

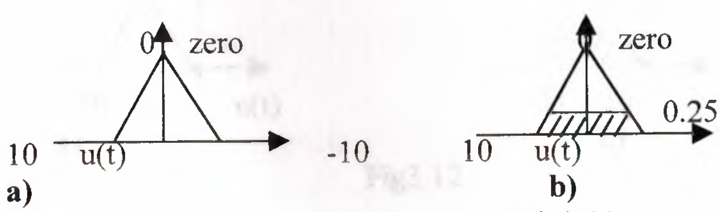


Fig3.11

(a) Consequent membership function and **(b)** implied fuzzy set with membership function $\mu_l(u)$ for rule (1). Recall that the units for $u(t)$ are Newtons (N).

3.10 Recommendation From Another Rule

Next, 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 we will refer to as "rule (2)." Using the minimum to represent the premise, we have

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

so that we are 0.75 certain that this rule applies to the current situation. Notice that we are much more certain that rule (2) applies to the current situation than rule (1). 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 Figure3.12 (a). The membership function for the conclusion reached by rule (2), which we denote by $\mu_2(u)$, is shown in Figure 3.12(b) (the shaded region) and is given by

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

this membership function defines the implied fuzzy set for rule (2) (i.e., it is the conclusion that is reached by rule (2)). Once again, for different values of $e(t)$ and $de(t)/dt$ there will be different values of $\mu_{premise2}(e(t),de(t)/dt)$ for rule (2) and hence different functions $\mu_2(u)$ obtained. Rule (2) is quite certain that the control output (process input) should be a small negative value. This makes sense since if the pendulum has some counterclockwise velocity then we would want 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), we see that we are more certain of the conclusion reached by rule (2).

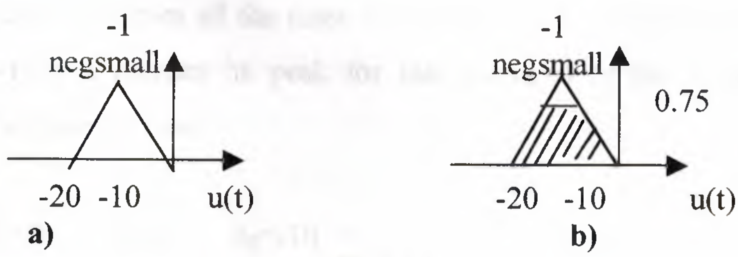


Fig3.12

This completes the operations of the inference mechanism in Figure 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.)

3.11 Converting Decisions into Actions

Next, we consider the defuzzification operation, which is the final component of the fuzzy controller shown in Figure 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 Figure3.13 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.

3.12 Combining Recommendations

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

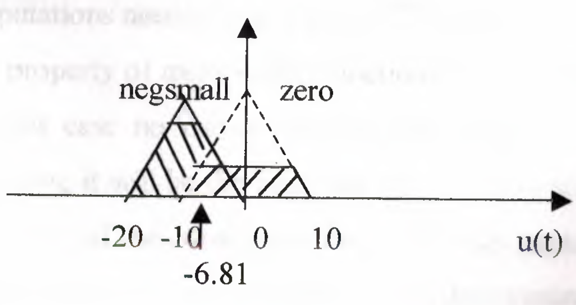


Fig.3.13

as shown in Figure3.13 Let $\int \mu_i$

denote the area under the membership function μ_i. The COG method computes u^{crisp} to be

$$u^{crisp} = \frac{\sum_i b_i \int \mu_i}{\sum_i \int \mu_i} \tag{1}$$

This is the classical formula for computing the center of gravity. In this case it is for com- puting the center of gravity of the implied fuzzy sets. Three items about Equation (1) are 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 Figure 3.7).
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 Figures3.11 and 3.12) is equal to

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.

Using Equation (1) with Figure 3.13 we have

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

as the input to the pendulum for the given *e(t)* and *de(t)/dt* *(see fig3.13)*.

3.13 Other Ways to Compute and Combine Recommendations

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

First, consider the use of the product. Consider Figure3.14, 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 Figure 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 Figure3.14 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

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

which also makes sense

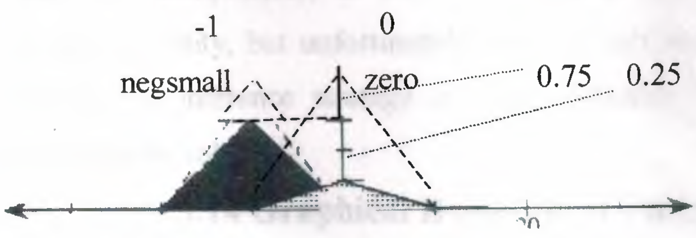


Fig3.14

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 $\mu_{premisei}$ 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 $\mu_{premisei}$. This is a valid replacement since the area of the implied fuzzy set is generally proportional to $\mu_{premisei}$ since $\mu_{premisei}$ is used to chop the top off (minimum) or scale (product) the triangular output membership function when COG is used for our example. For the above example, we have

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

which just happens to be the same value as above. Some like the center-average defuzzification method because the computations needed are simpler than for COG and because the output membership functions are easy to store since the only relevant information they provide is their center values (b;) (i.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.

3.14 Graphical Depiction of Fuzzy Decision Making

For convenience, let is summarize the procedure that the fuzzy controller uses to compute its outputs given its inputs in Figure3.15. 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.

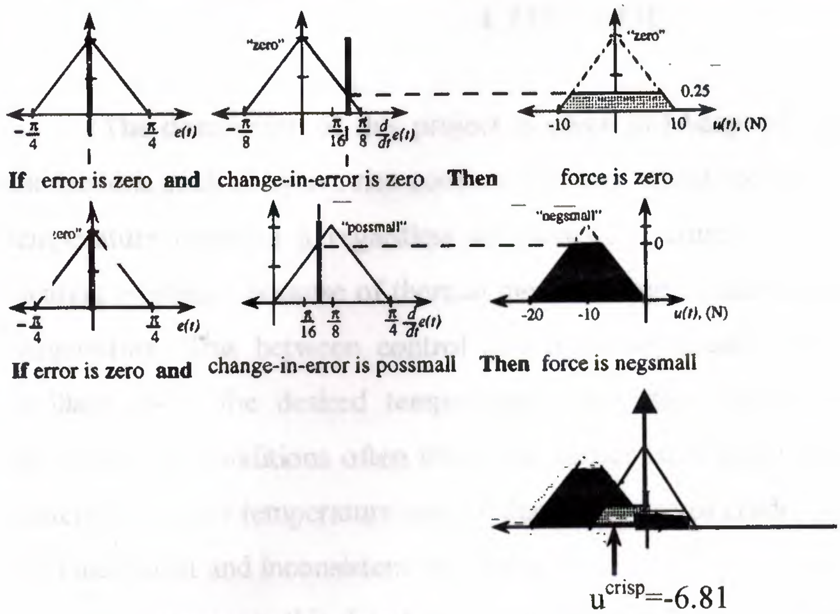


Fig.3.15 Graphical representation of fuzzy controller operations

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.

CHAPTER FOUR: FUZZY SYSTEM FOR SMART HEATER CONTROL

The description of this project is given in [Adap96]. Household appliances that use heaters, such as ovens, rice cookers, toasters, should come quickly to the desired temperature maintain it regardless of changes in conditions such as load or air flow. Heating elements, because of thermal inertia, require a certain amount of time to change temperature. This between control and response causes the heater to overshoot and oscillate about the desired temperature. Once the temperature is achieved, varying environmental conditions often throw the heater back into oscillation. In addition, most heaters have poor temperature control due to the use of crude, on-off switches. The result is an inefficient and inconsistent operation.

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

We will describe both the fuzzy logic rules and design for the heater control. Most appliance heaters are controlled by bimetallic strips or capillary tubes that expand with heat and switch the heater on or off. These crude mechanical controllers merely react to temperature fluctuations and cannot anticipate when the heater is approaching the selected temperature. When the element passes through the operating point, the switch opens, cutting power to the heater. But by this time, the heater has enough energy to carry the system temperature far above the selected range and it takes a while for it to return.

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

Heaters have widely varying characteristics. They are specified in terms of their form including length, shape, thickness and material composition. Heating elements may add instability to a system because of their slow response time and thermal inertia. The heater specifications are based on the requirements of the end product. The end product also has a range of thermal characteristics that influence the behavior of the heater. Many variables of heating systems make the design of a controller for different systems a difficult task. A control system using a fuzzy controller brings the temperature of the heater to the selected temperature quickly and keeps it there regardless of any changes in the load or environment. This results in a more stable and reliable operating temperature.

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

The fuzzy controller design parameters (inputs, outputs, fuzzy variables and rules) are given in (Fig. 4.5), followed by a brief description of the fuzzy controller operation. The main part of any fuzzy controller is implemented as a set of rules. It performs the control algorithm. By studying the rules, one can see the criteria for taking actions such as switching the heater on or off. These rules make decisions based on adjustable membership function definitions. The rules are easily modified to respond to different criteria. The following describes the rules' purposes in relation to the inputs, their associated fuzzy variables, and the action taken when a rule is fired.

Timer rules (Rules 1 and 2) are used to generate the timer for pulse switching the triac and to adjust the data processing rate. Rule one increments the ramp output if it is in the count membership function. When the ramp reaches the reset function, then the reset rule will be fired and return the ramp to zero to begin to increment through the count again. The rate of an increment is set to 12, but could be any non-zero value according to the requirements of the application. The increment and reset actions implement a timer that causes the ramp to sweep across an axis.

The heater output is used to define the center value of membership functions On and Off. On and Off are, in turn, used by later rules to switch the triac on and off. Rules 3-19 consider both the current temperature and the previous value of temperature whenever the time has reached reset. The winning rule in this group will add to or subtract from the value of the heater.

The fuzzy variables classify the temperature as matching the selected value, or being too hot or cold. The membership functions used the user selected temperature (TSET) as a floating center value to compare the selected temperature with the actual temperature. As the user vanes the desired temperature value, the membership function centers move left or right. Other fuzzy variables use the current temperature value to define their centers. As the temperature varies, the functions shift right or left. The fuzzy variables compare the current temperature with its time delayed value. The comparison is a calculation of the derivative value and sign of the temperature. By comparing both the current value and derivative of temperature with the desired temperature, one can calculate a temperature correction value based on both the absolute difference and the rate of change of temperature. The calculation allows for precise control and minimizes the overshoot.

Rules 3-19 use one fuzzy variable from each set to adjust the value of the heater. The actions of the rules are designed to move the heater towards the desired temperature without causing it to overshoot. (Table 4.2) summarizes the various conditions and actions of the heater control rules. Note, if all of these rules are not fired (i.e., the blank boxes scenarios in Table 4.2) then the value of the heater is not changed. Some of the rules are described below. Values indicated are added from the heater based upon die winning condition of TEMP and DELAY. No action is taken where there are blanks.

Td \ T	NL	NM	NS	N	PS	PM	PL
PL		-15					
PS		-8	-6	-1	-2		
N	25	5	1	0	-1	-5	-25
NS			2	1	6	8	
NL						15	

NL: Negative Large.

NM: Negative Medium.

NS: Negative Small.

PL: Positive Large.

N: Normal.

PS: Positive Small.

PM: Positive Medium.

Fragment of knowledge base:.

1. **If** T = N and Td = N **Then** U = + 0.
2. **If** T = NL and Td = N **Then** U = + 25.
3. **If** T = PL and Td = N **Then** U = - 25.
4. **If** T = NM and Td = PL **Then** U = - 15.
5. **If** T = NM and Td = PS **Then** U = - 8.
6. **If** T = PM and Td = NL **Then** U = + 15.
7. **If** T = PM and Td = NL **Then** U = + 8.
8. **If** T = NM and Td = N **Then** U = + 5.
9. **If** T = PM and Td = N **Then** U = - 5.
10. **If** T = NS and Td = NL **Then** U = + 2.
11. **If** T = NS and Td = PS **Then** U = - 6.
12. **If** T = NS and Td = N **Then** U = + 1.
13. **If** T = PS and Td = N **Then** U = - 1.
14. **If** T = PS and Td = NS **Then** U = + 6.
15. **If** T = PS and Td = PS **Then** U = - 2.
16. **If** T = N and Td = NS **Then** U = + 1.
17. **If** T = N and Td = PS **Then** U = - 1.

Rule 3, for example, is true if the temperature and its previous value are in the selected range, and leaves the heater unchanged. Rules 4 and 5 detect the temperature to be far from the selected value and not changing. These rules make large increases or decreases of the heat. Rules 6 and 7 both consider a medium-low value for the temperature (TMLOW), but they consider different derivatives and have different action values. Rule 6 considers a large positive delayed value (DLPOS) while Rule 7 considers a small one (DSPOS). At first glance, the correction actions for the rules may seem to be counterintuitive. In each rule the temperature is too low, yet the action is to reduce the value of the heater which decreases temperature when the heat is stable. The key is that in each rule the derivative is positive indicating that the temperature of the heater is increasing. The reduction in the value of heat prevents overshoot while still allowing the temperature to reach the set point.

The corrective action from Rule 6 is larger than that for Rule 7 because in the former rule the derivative value of the air temperature was larger indicating a faster increase in temperature. The larger previous value subtracts more from the heater (-15) than does the smaller (-8). The same reasoning holds in the remaining rules in the set. As the heater moves closer to the desired point, other rules will act to gradually hold it there.

Rules 20 and 22 guard against the heating unit from getting too hot. Rule 20 decreases the heater value in small increments when the temperature has gone above the safety level. During this process, at the same time, Rule 22 prevents the PWM control of System the triac and keeps it off until the temperature decreases to below the safety value. Note that Rule 23 (which turns on the pulse) will never be enacted while the temperature is greater than the safety value. Even if the fuzzy variables **Ramp** is **ON** and **TEMP** is **OverT** should evaluate to the same value, the precedence goes to the first rule (Rule 22) for determining the action. If the temperature is above the safety value, **TEMP is OverT** will always be evaluated as a maximum value. Rules 21-23 control the PWM output, triac, turning it on and off. The decision is based upon the comparison of the ramp feedback value to the heater value. The heater value serves as a center for membership functions ON and OFF. As the ramp moves across count and back to zero, it also moves between the functions ON and OFF. Depending on the value of the heater, the position of the center value for the two functions will move to the left or right. That will vary the amount of time during a given sweep that the heater is on or off. By moving the heater, one changes the width of the pulse controlling the triac and the heater temperature.

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 bloks- fuzzification, inference engine, defuzzification, 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 Process Control and Knowledge Engineering in Petrochemical and Robotic Manufacturing , Refik Aliev , Fuad Aliev and Memetgasan Babae

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

[3]Hill, G., Horstkotte, E. and Teichrow, J. (1990). Fuzzy-C *development system- users manual*, Togai Infralogic, 30 Corporate Park, Irvine, CA 92714, USA.

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

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

[6]Lee, C. C. (1990). Fuzzy logic in control systems: Fuzzy logic controller, *IEEE, Trans. System, Man &Cybernetics* **20**(2): 404–435.

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

[8]Mizumoto, M. (1992). Realization of PID controls by fuzzy control methods, in IEEE (ed.),*First Int.Conf.On. Fuzzy Systems*, number 92CH3073-4, The Institute of Electrical and Electronics Engineers, Inc, San Diego, pp.709-715.

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

[10]Pedrycz, W. (1993). Fuzzy controller fuzzy system, second edn, Wiley and Sons, New York. Qiao, W. and Mizumoto, M. (1996). PID type fuzzy controller and parameters adaptive method,*Fuzzy Sets and Systems* **78**:23-35

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

[12]Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control, IEEE *Trans.Systems,Man &Cybernetics* **15**(1): 116–132.

[13]. K.M.Kerimov, R.H.Abiyev, T.G.Melikov. Fuzzy system for determination of oil-gas fields productivity/ Geophysics News in Azerbaijan. Baku,N:1,1999.pp.13-15.

[14]. K.M.Kerimov, R.H.Abiyev, T.G.Melikov, E.S.Novruzov. Use of illegible logic to determine oil and gas bearing intervals in beds/ Geophysics News in Azerbaijan. Baku, N:2, 1999, pp.33-35.

[15]. K.M.Kerimov, R.H.Abiyev, T.G.Melikov. Determination of oil and gas field productivity in the condition of insufficiency and fuzziness of information// Second international symposium on mathematical and computational applications, Baku, september 1-3,1999.