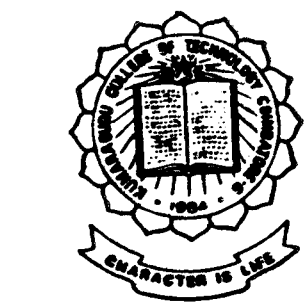
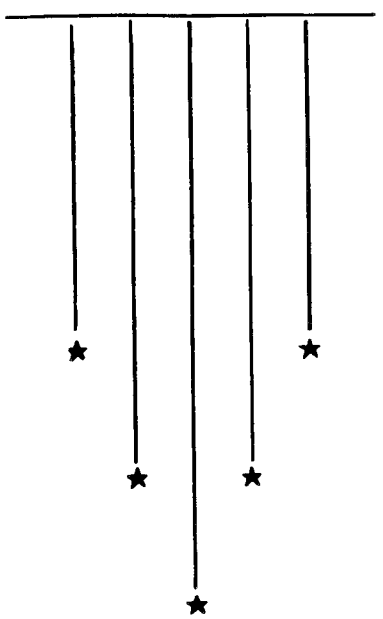
Design of Fuzzy Logic Controller for Wind Energy Conversion Scheme

P-1322

PROJECT REPORT 1996 - 97





Submitted by

C. CHANDRA SEKARAN

N. RAVIKKUMAR

T. SIVA KUMAR

N. VENKATESH

Guided by

Prof. K. RAMPRAKASH, M.E.,

Mrs. DEVI, B.E.,

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING IN ELECTRONICS AND COMMUNICATION ENGINEERING

OF THE BHARATHIAR UNIVERSITY, COIMBATORE

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Kumaraguru College of Technology

COIMBATORE - 641 006.

Kumaraguru College of Technology

Coimbatore - 641 006.

Department of Electronics and Communication Engineering

Certificate

This is to Certify that this Project Entitled

DESIGN OF FUZZY LOGIC CONTROLLER FOR WIND ENERGY CONVERSION SCHEME

Has been submitted by

Mr. C.CHANDRASEKARAN, N.RAVIKUMAR, T.SIVAKUMAR, N.VENKATESH

in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in the Electronics and Communication Engineering Branch of the Bharathiar University, Coimbatore-641046 during the academic year 1996-'97.

(Guide) (Head of Department)

Certified that the Candidate was Examined by us in the Project Work.

Viva-Voce Examination held on 10.4.97

University Register Number 9327D021, 9327D024, 9327D026, 9327D027

(Internal Examiner) (External Examiner)

CONTENTS

ACKNOWLEDGEMENT

First of all, we wish to express our gratitude and indebtedness to our beloved principal **Dr.S.Subramanian , B.E.,Msc(Engg), Ph.D., S.M.IEE** for his kind patronage.

We sincerely express our thanks to our mentor **Prof.M.Ramasamy, M.E., C.Engg(I)., MISTE, M.IEEE (USA), M.I.E.**, Head of the Department, for the complete freedom and constant encouragement throughout the project.

With immense pleasure we express our heartfelt thanks to our respected guides **Prof. K. Ramprakash, M.E., Assistant Professor and Mrs.B.Devi., B.E., Lecturer** for their inspiration, valuable guidance and unflinching support.

Special thanks to **Mr.R.Hariharan, M.E., Electrical** Department, for his thought provoking suggestions and ideas.

Last but not the least we would like to thank all the faculty members of the ECE Department who had lend their helping hands for the completion of the project.

SYNOPSIS

The objective of this project work is to track and extract maximum power from the wind energy scheme (WES) and to transfer this power to the local isolated load. This avoids the design problems of the controller based on mathematical derivation and it also introduces artificial intelligence in the control action.

This fuzzy logic controller is designed for the wind energy conversion scheme which uses modulation index control technique. Fuzzy algorithm adopted in this controller are systematically formed according to intuition and experience about the wind energy conversion scheme. This wind energy conversion controller software based on the fuzzy control algorithms is realised using personal computer. Since the system parameters are not needed in the implementation of the drive system, and due inherent future of highly adaptive capability possessed by the fuzzy controller, the performance of the controlled system is quiet robust and insensitive to parameter and operating condition changes.

The computer simulation is carried out to prove the above characteristics

INTRODUCTION

As a method of encoding and using human knowledge in a form that is very close to the way experts think about difficult, complex problems, "FUZZY SYSTEM" provide facilities necessary to break through the computational bottlenecks associated with additional decision support and expert system.

This project uses fuzzy logic controller to extract maximum power from the wind energy uses a rule based fuzzy logic controller to control the output power of a wind energy conversion scheme.

The fuzzy principle states that a matter of degree in chapter I we discuss the basic ideas and concept of fuzzy logic.

The fuzzy logic controller is based on fuzzy logic which represent the thinking process that human operator might go through while controlling the system.

Chapter II presents design aspects of process control system, and various types of feedback controllers. The limitations of the conventional controllers are also discussed.

The wind energy convertion scheme (WECS) consists of self exciting induction motor a diode bridge rectifier and a PWM inverter.

Chapter III deals with fuzzy logic controller and PID controller is compared with the that of fuzzy logic controller. Chapter IV use the brief discussion of the parts of WECS. To overcome the difficulty in controller design a fuzzy logic controller is designed based on fuzzy control algorithm. This is explained in Chapter V.

The chapter VI explains procedure involved in simulation and how to represent fuzzy logic in software.

FUZZY LOGIC

Fuzzy logic is a calculus of compatibility. Unlike probability which is based on frequency distributions in a random population, fuzzy logic deals with describing the characteristics of properties. Fuzzy logic describes properties that have continuously varying values by associating partitions of these values with a semantic label. Much of these descriptive power of fuzzy logic comes from the fact that these semantic partitions can overlap. This overlap corresponds to the transition from one state to the next. these transitions arise from the naturally occurring ambiguity associated with the intermediate states of the semantic labels.

Just what is "fuzziness" Fuzziness is a measure of how well an instance (value) conforms to a semantic ideal or concept. Fuzziness describes the degree of membership in a fuzzy set. This degree of membership can be viewed as the level of compatibility between an instance from the sets domain and the concept overlaying set

1.1 VAGUENESS

Every day, the vagueness is used in the language expression and communication. For example, the expression, " this car is fast", "Temperature is very hot" etc. .. are all vague and do not indicate any quantitative measure. They

are different from saying "The car runs at 80 MPH" or "Temperature is 45 degrees" . But vagueness is accepted, understood processed and propagated. We accept imprecise data and vague rule and able to recognise, interpret and use them in every day life. Fuzzy sets are generalised version of conventional sets and show how the vagueness can be treated systematically in a mathematical framework.

1.2 MEMBERSHIP FUNCTION

It is a function describing the membership of a element "x" in a fuzzy set. The shape of the membership function depends upon the situation in question. The element of x corresponds to the highest grade of membership is called a model value of the fuzzy number. The triangular membership function is the most frequently used function and the most practical, but other shapes also used. For example trapezoid, beta curve etc. ..

1.3 LAW OF NON-CONTRADICTION

The law of Boolean logic, if you give it a moments thought, is a direct out growth of the law of non-contradiction sine it assumes that the sets in question contain completely separate and disjoint elements. In this way the law of the Excluded Middle is symmetrical with the Law of Non contradiction one operation insures the disjointedness of sets along an arbitrarily crisp boundary and the other

operation combines the disjoint sets to produce the conjoint universal domain. Since fuzzy logic does not adhere to the low of non-contradiction, it follows that it will not obey the law of Excluded Middle. These aspects of fuzzy logic are discussed in detail when we turn to the idea of fuzzy complements.

1.4 LINGUISTIC VARIABLE

Linguistic variable is a name of a fuzzy set a linguistic variable encapsulates the properties of approximate or imprecise concept in a systematic and computationally useful way. It reduces the apparent complexity of describing a system by matching a semantic tag to the underlying concept. A linguistic variable always represents a fuzzy space (another way of saying that when we evaluate a linguistic variable we come out with a fuzzy set).

1.5 DOMAIN

The total allowable universe of values is called the domain of the fuzzy set. The domain is a set of real numbers increasing monotonically from left to right. The values can be both positive and negative. You select the domain to represent the completely operating range of values for the fuzzy set within the context of your model.

1.6 THE UNIVERSE OF DISCOURSE

A model variable is often described in terms of its fuzzy space. This space is generally composed of multiple, overlapping fuzzy sets, each fuzzy set describing a semantic partition of the variables allowable problem state.

The figure 1.1 illustrates this concept. The model parameter temperature is broken into four fuzzy sets: cold, cool, warm, and hot.

This total problem space from the smallest to the largest allowable value, is called the universe of discourse. Note that the universe of discourse is associated with a model variable and not with a particular fuzzy set (the range of an individual fuzzy set is the domain). The variables allowable range of values constitutes its working problem space, this space is then decomposed into a number of overlapping fuzzy regions. Each region is assigned a term name so that it can be referenced in the model (You can only write rules associated with fuzzy regions in the variables universe of discourse). This collection of fuzzy sets associated with a variable is often called a term set.

1.7 FUZZY SETS

Fuzzy sets are actually functions that map the value that might be a member of the set to a number between 0 and 1 indicating its actual degree of membership.

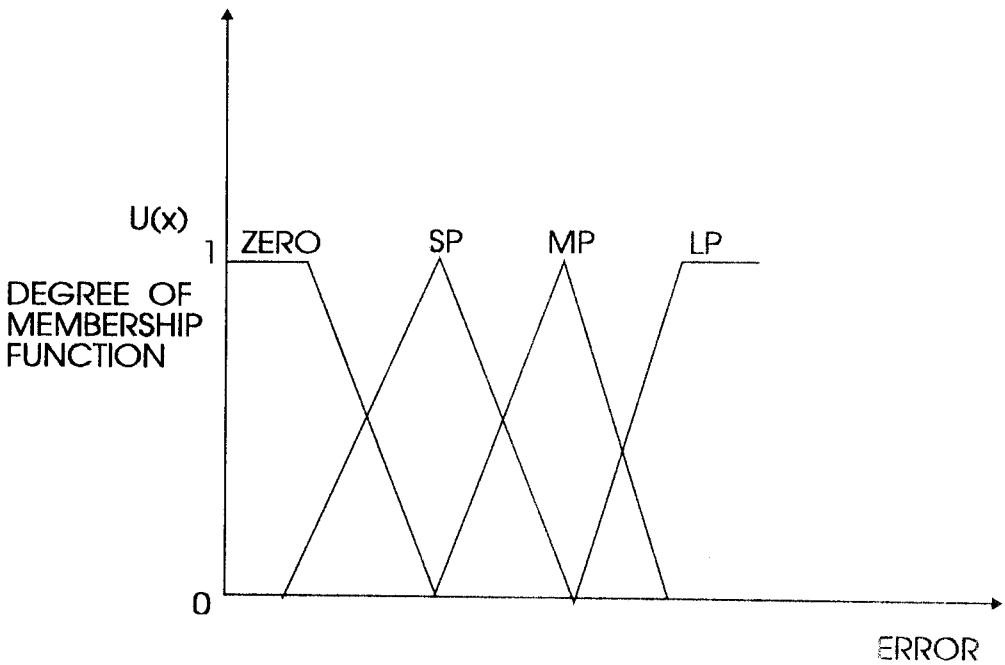


FIG 1.1 MEMBERSHIP FUNCTION

A degree of 0 means that the value is not in the set, and a degree of 1 means that the value is completely representative of the set. This produces a curve across the members of the sets as a simple example consider the idea of a LONG project.

The figure 1.2 shows how the concept might be represented. The members of the set are duration periods, in weeks for a project. The fuzzy set indicates to what degree a project of a specified duration is a member of a set of LONG project. As the number of weeks increases our belief that the project is LONG increases. A project two weeks in duration would not be considered LONG, a project ten weeks in duration will have a moderate membership in the set of LONG project, and a project of more that 16 weeks in duration is most certainly a LONG project. Ofcourse the actual definition of what is a long project depends on the context in which it is used.

An element is a member of a fuzzy set if

1. It falls within the underlying domain of the set
2. If its membership value is greater than zero

1.7.1 LINEAR REPRESENTATIONS

The linear proportionality surface is a straight line. This is perhaps the simplest fuzzy set and often a good choice when approximating an unknown or

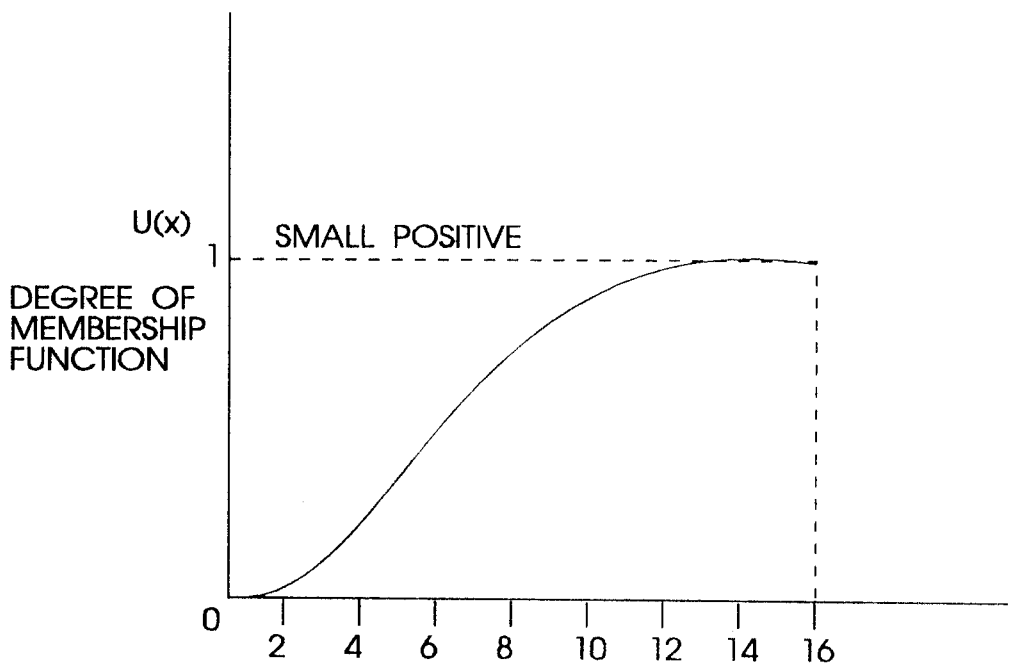


FIG 1.2 MEMBERSHIP FUNCTION

poorly understood concept that is not a fuzzy number. There are two states of a linear fuzzy set. The increasing values that have increasing set membership. The right-hand edge of the domain is the value with full membership as shown in figure 1.3.

The decreasing fuzzy set is the opposite of the increasing set. The value at the left handed edge of the domain has complete set membership while the value at the right hand side of the edge has no membership as shown in figure 1.4.

Since the linear fuzzy shape connects a square fuzzy spaces at a 45 degree angle, the truth of any domain value is proportional to its distance in the value axis.

1.7.2 TRIANGULAR REPRESENTATION

As shown in figure 1.5 triangular FUZZY SET there is a linear increasing portion and linear decreasing portion between which there is a single value of full membership. On the left side of the linear increasing portion and on the right side of the linear decreasing portion we have a value of zero membership.

AN INCREASING FUZZY SET

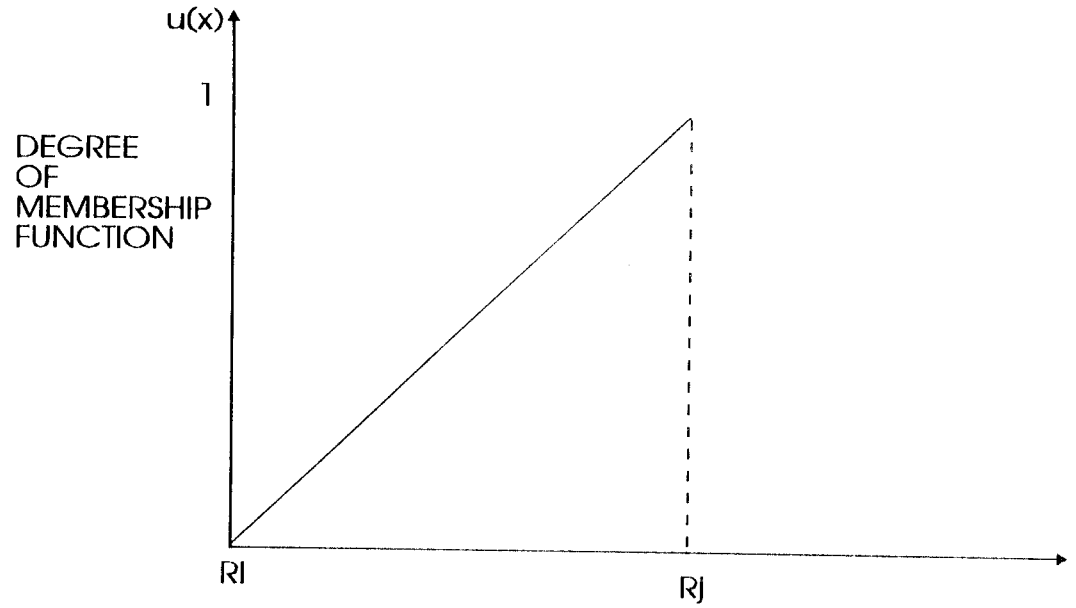
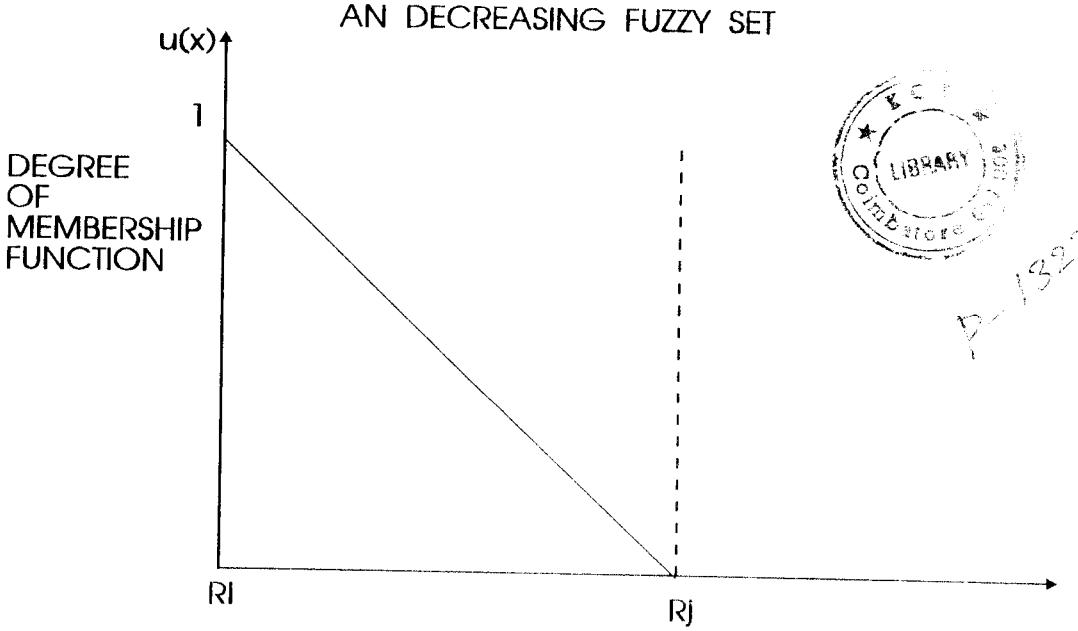


Fig 1.3 AN INCREASING FUZZY SET

AN DECREASING FUZZY SET



LIBRARY
P-1822

Fig 1.3 AN DECREASING FUZZY SET

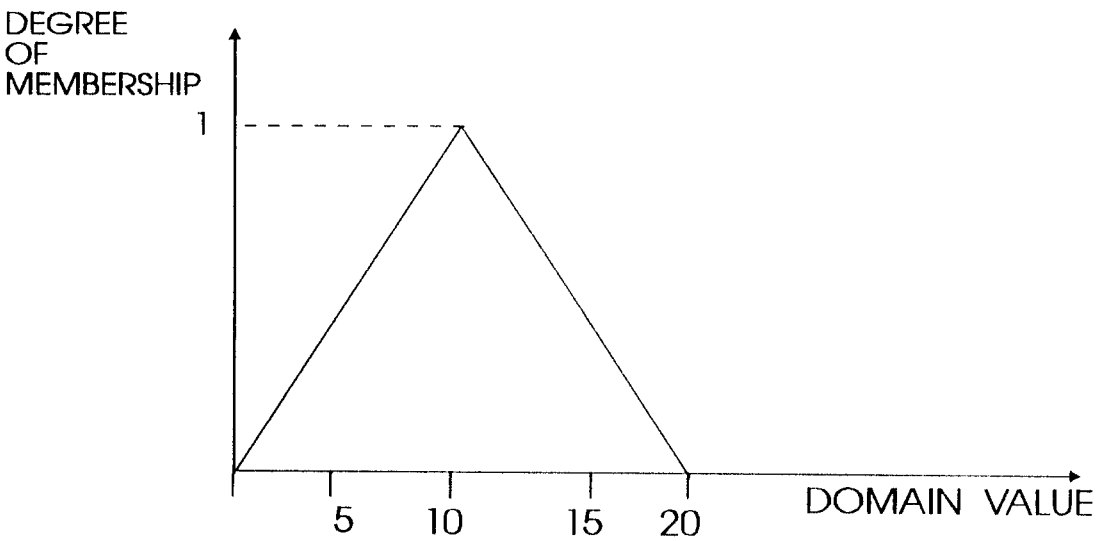


FIG. 1.5
TRIANGULAR MEMBERSHIP FUNCTION

1.7.3 INTERSECTION OF FUZZY SETS

In a crisp system the intersection of two sets contain the elements that are common to both sets. This is equivalent to the arithmetic or logical AND operation. In conventional fuzzy logic, the AND operator is supported by taking the minimum of the truth membership grades.

The intersection operator is the most common form of restriction used in fuzzy rule antecedents. When we evaluate rules in a form such as

If x is y and z is w then m is p;

the elastic membership strength between the consequent m and the fuzzy region p is determined by the strength of the premise or antecedent. The truth of this antecedent is determined by taking the min (M[x is y], M[z is w]).

1.8 FUZZY PROPOSITIONS

A fuzzy model consists of a series of conditional and propositions. A proposition or statement establishes a relationship between a value in the underlying domain and fuzzy space. Propositions are expressed in the form,

X is Y

where x is a scalar from the domain and y is a linguistic variable. The effect of evaluating a fuzzy proposition is a degree or grade of membership derived from the transfer function,

$$\mu_A \text{ ---- } (x\ y)$$

This is the essence of an approximate statement. The derived truth member value establishes a compatibility between x and the generated fuzzy space y. Such propositions answer the question "How compatible is x with y" or "to what degree is x is a member of set y?" This truth value is used in the correlation and implication transfer functions to create or update the output solution space. The final solution fuzzy space is created by aggregating the collection of correlated fuzzy propositions. The correlation is based on the truth of each fuzzy proposition. (When multiple propositions are connected by AND, OR connectors, the truth used in the process is the truth produced by applying these operators).

1.8.1 CONDITIONAL FUZZY PROPOSITIONS

A conditional proposition is one that is qualified by an IF statement. These are analogous to the rules of a conventional symbolic expert systems. The proposition has the general form, where w and x are model scalar values and z and y are linguistic variables. The proposition following the IF term is the antecedent or predicate and is any arbitrary fuzzy proposition. The proposition following THEN

term is the consequent and is also any arbitrary fuzzy proposition. The statement x IS y is conditional on the truth of the predicate. We interpret this statement as,

x is member of y to the degree that w is a member of z,

that is, the consequent is correlated with the truth of antecedent. The fundamental proposition can be extended with fuzzy connectors,

if (w is z) . (y is w) . ... (u is s) then x is y

where "." is some form of the AND or operator. In the case of multiple antecedent proposition, the position of x within y is determined by the composite truth of the complete antecedent.

This is the basis for fuzzy monotonic reasoning. In actual fuzzy models x is a temporary fuzzy region (indicated by the X) containing the results of each proposition that specifies an elastic space for y. Thus, we need to read the conditional fuzzy proposition as

(X is a powerset of Y) to the degree that (W is member of Z)

these powerset is formed by first correlating the fuzzy set Y according to the truth of the antecedent proposition (W is Z). The solution fuzzy space is then updated by taking union of the solution set and the newly correlated fuzzy set.

1.8.2 UNCONDITIONAL FUZZY PROPOSITIONS

An unconditional fuzzy proposition is one that is not qualified by an IF statement. The propositions has these general form,

X is Y

Where X is a scalar from the domain and Y is the linguistic variable. Unconditional statements are always applied within the model and, depending on how they are applied , serve either to restrict the output space or to define a default solution space (if none of the conditional rules execute). We interpret an unconditional fuzzy proposition as

X IS THE MINIMUM SUBSET OF Y

when the output fuzzy set x is empty, then x is restricted to y, otherwise, for the domain of y,x becomes the MIN(X,Y). Since these propositions are unconditional, they are never correlated, that is, their truth values are never reduced before they are applied to the output space. The solution fuzzy space is updated by taking intersection of the solution set and the target fuzzy set.

1.8.3 THE ORDER OF PROPOSITION EXECUTION

For models containing only conditional or unconditional propositions the order in which propositions (that is, rules) is executed is not important. However, if a model contains mixture of these two types, then the order of executions

becomes important. The effect of applying unconditional propositions changes the nature of the model solution space depending on whether the propositions are applied before or after the set of conditionals.

Unconditional propositions are generally used to establish the default support set for a model. If none of the conditional rules executes, then a value for the solution variable is determined from the space bounded by the unconditionals. For this reasons they must be executed before any of conditionals. If none of the conditional rules that fall within the same underlying domain as the unconditionals has an antecedent strength greater than the maximum intersection of the unconditionals, they will not contribute to the model solution.

Although much less commonly used, the unconditionals can also be used to restrict the final solution space of a model to the maximum truth of their intersection. This is done by applying the unconditionals after all the conditional proposition have been evaluated.

1.9 ADVANTAGES OF FUZZY LOGIC

* Fewer values,rules and decisions are required.
* More observed variables can be evaluated.

* Linguistic, not numerical variables are used, making it similar to the way human thinks.

* It relates output to input, without having to understand all the variables, permitting the design of a system that may be more accurate and stable than one with a conventional control system.

* Simplicity allows the solution of previously unsolved problems.

* Rapid prototyping is possible because a system designer does not have to know everything about the system before starting work.

* They are cheaper to make than conventional systems because they are easier to design.

* They have increased robustness.

* They simplify knowledge acquisition and representation.

* A few rules encompass great complexity.

1.10 DRAWBACKS OF FUZZY LOGIC

* It is hard to develop a model from the fuzzy system.

* Though they are easier to design and faster to prototypethan conventional control system, fuzzy system require more simulation and fine tuning before they are operational.

* Perhaps and biggest drawback is the cultural bias in the united states in favour of mathematically precise or crisp system and linear model for control systems.

1.11 FUZZY LOGIC APPLICATIONS

In the recent years, fuzzy logic has found several applications in fields ranging from finance to earthquake engineering. In particular, the fuzzy control

has emerged as one of the most active and fruitful areas for research for the application of fuzzy set theory. In many applications the FLC based systems have proved to be superior in performance to conventional systems.

Notable applications of FLC include heat exchange, warm water process, activated sludge process, traffic junctions, cement kiln, aircraft flight control, turning process, robot control, model car parking and turning, automobile speed control, water purification process, elevator control, automobile transmission control, power systems and nuclear reactor control, fuzzy memory devices and the fuzzy computer. In this connection it should be noted that the first successful industrial application of the FLC was the cement kiln control system developed in 1979.

Commercial applications include

* Vacuum cleaner.

* Washing machine.

* Air conditioners etc ..

SUMMARY

This chapter use a brief description about fuzzy logic, its concept, the various terminology used in fuzzy sets and the applications of fuzzy logic.

PROCESS CONTROL SYSTEM

Automatic process control is concerned with maintaining process variables like temperature, pressure, flow etc., at some desired operating value.

The basic block diagram of a process control system is shown in fig 2.1. The basic component of the system are

1. Sensor, often called the primary element.
2. Transmitter, also called the secondary element.
3. Controller, the brain of the control system.
4. Final control element.

These components perform the following basic operations.

1. ***Measurement*:** Measuring the variable to be controlled and it is usually done by the combination of sensor and transmitter.

2. ***Decision*:** Based on the measurement, the controller should decide about the action to be taken to maintain the variable at a desired value.

3. ***Action*:** As a result of the controllers decision, the final control element takes an action.

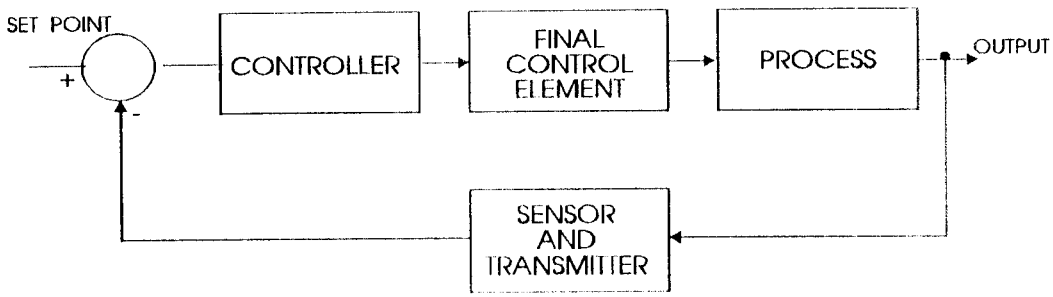


FIG 2.1
BLOCK DIAGRAM OF A PROCESS CONTROL SYSTEM

The objective of an automatic system is to use the manipulated variable to maintain the controlled variable at its set point inspite of the disturbances. After designing the control system, the closed loop response is examined and the inference is made in terms of performance indices.

2.1 TIME RESPONSE

In order to analyse the transient and steady state behaviours of a control system, the first step is always to obtain the mathematical model of the system. After obtaining the closed loop transfer function of the system, the performance of the control system is measured by computing several time response performance indices as well as steady state accuracy.

The step response characterised by the performance indices such as the delay time, rise time, peak overshoot, settling time and steady state error.

For a good performance, rise time, overshoot and settling time should be less and the steady state error should be zero.

2.2 NEED FOR MAINTAINING CONTROLLED VARIABLE AT SET POINT

1. To maintain the product quality on a continuous basis and with minimal cost.
2. To optimise plant production cost.
3. To prevent injury to plant personnel or damage to equipment.

2.3 DESIGN ASPECTS OF PROCESS CONTROL SYSTEM

2.3.1 CONTROL OBJECTIVES

A. To ensure stability of the process.

B. To suppress the influence of external disturbances.

C. To optimise the performance of a plant and a combination of the above.

2.3.2 DESIGN OF CONTROLLERS

The controller is the device that performs the decision in the control system its function are

1. To compare the process signals from the transmitter, the controller variables, with the set point and
2. To send an appropriate signal to the control valve or any other final control element, in order to maintain the controlled variable at its set point.

To determine the action of the controller, the designer must know,

1. The process requirement for control

2. The action of control valve or any other final control element

The design requirement of the feedback controllers are

1. To select the controller type for a specific purpose

2. To select the controller coefficients-known as tuning of a controller.

3. To verify whether the resulting performance indices satisfy the required values.

2.4 TYPES OF FEEDBACK CONTROLLERS

An automatic controller compares the actual value of the plant output with reference input, determines the deviations and produces the control signal that will reduce the deviation to zero or to a small value. The manner in which the controller produces the control signals is called the control action.

Given below are the six basic modes of analog controllers:

1. Two position or ON-OFF control
2. Proportional
3. Integral
4. Proportional plus integral
5. Proportional plus derivative
6. Proportional plus integral plus derivative.

2.4.1 TWO-POSITION CONTROL

Two position control has the widest industrial and domestic use on process having not more than two energy storage elements. In operation, two position control is very simple, but in theory the action is difficult to analyse because of the discontinuous nature of the changes in the manipulated variable. Nevertheless, problems of two position control can be solved by quantitative consideration.

2.4.2 CONTINUOUS CONTROLLER MODES

1. Proportional control mode:

The concept of proportional control mode is the smooth linear relationship exist between the controller output and error. Thus over some range of errors about the set point each value of error has a unique way of controller output in one-one correspondence. The range of error to cover the 0% to 100% controller output is proportional band.

This can be represented by

$$P = K_p + P_o$$

Where k_p is the proportional constant, e_p is the value between the error and output and po is the output when there is new error.

CHARACTERISTICS

1. If the error is zero, the output is a constant and equal to po.
2. If there is error for every 1 % about kp is added to or subtracted from p_o depending on the reverse or direct action of the controller.
3. There is a band of controller about zero of magnitude p_o within which the output is not subtracted at 0% and 100%.

INTEGRAL CONTROL MODE

It represents a natural extra vision of the principle of floating control in the limit of the infinitesimal changes in error instead of single speed, we have continuous changes in the speeds depending on error.

$$ie., \; dp/dt \quad K_i E_p$$

Where dp/dt is the rate of change of controller output ki is the constant relating the rate with error.

In some cases inverse of time called the integral time $t_i = 1/k_i$ expressed in second or minutes is used to describe the integral mode. The controller output at any time is given by equation

$$p(t) \quad K_i \; integral \; E_p(t)dt + P(o)$$

CHARACTERISTICS

1. If the error is not zero, output will begin to increase or decrease at a rate of ki% for every +/- 1% of error.
2. If the error is zero, output stages fixed at a value equal to what it was when the error went to zero

DERIVATIVE CONTROL MODE

This mode of control action provides an output proportional to the rate of change of error .This is also known as rate of anticipatory control. this mode cannot be use d alone because, when the error is zero or constant controller output will be zero.

$$P = K_d * dE_p/dt$$

where kd is the derivative gain constant, dE_p/dt is the rate of change of error.

CHARACTERISTICS

1. If the error is zero the mode provides no output.
2. If the error is constant in time, the mode provides no output 3. If the error is changing in time ,mode controls an output of kp% for every 1% per second rate of change of error.

4. For direct action the position rate of change of error produce a positive derivative mode output.

COMPOSITE CONTROL MODE

It is very common to find control requirements that do not fit the application of any previously controlled modes. It is possible to combine several basic modes, thereby gaining the advantages of each mode.

1. Proportional-Integral Control

This is a control mode that results from accombination of the proportional mode and integral mode.

$$p = K_p E_p + E_p K_I \int E_p dt + P_I(o)$$

where $p_I(0)$ is the integral term value at t = 0.

•

The main advantage of this mode is one-to-one correspondence is available at the integral mode eliminates the inherent offset. Notice that the proportional gain design also change the integral gain though ki can be independently changed

CHARACTERISTICS

1. When the error is zero, the controller output is fixed at the value that the integral term had when the error went to zero. This output is given by p1(0). In equation ,simply because we choose to define the timer at which observations starts at t =0.
2. If the error is not zero, proportional term contributes a correction and integral term increases or decreases the accumulated value.

2. *PROPORTIONAL- DERIVATIVE CONTROL MODE*

A second combination of control modes has many industrial applications. It involves several uses of proportional and derivative mode.

$$P = K_p E_p + K_p k_d (dE_p/dt) + p(0)$$

It is clear that the system cannot eliminate the offset of proportional controller,it however handle faster load changes

THREE MODE CONTROLLER

PID CONTROLLER

The best known controllers used in industrial process controls the PID controller because of their simple structure and robust performance in a wide range of operating conditions. Hence a PID controller is chosen for tuning purposes.

4. Proportional-integral control has no offset because of the integral action. The unstabilizing influence of the integral response is reflected in the large maximum deviation and the present deviation.

5. Integral control is best suited for the control of the process having little or no energy storage and the result of the comparisons are not representative of all integral control. However, on this process, the results indicate a large maximum deviation and a long stabilization time.

STABILITY

Stability is an important characteristic of the transient performance of the system. The value of the proportional gain Kp, that makes the system marginally stable so that sustained oscillations occur can be obtained by the use of Routh's stability criterion.

2.6 LIMITATION OF CONVENTION CONTROL

1. A major cause of difficulty in feedback controllers is the time lag. There is always some lag in controlled process before its output responds as desired to the changes in the input there may be also be significant lag in the action of the controlled device itself, or in its sensors for the actuators.

2. Classical PID controllers do not work well for the case of non-linear control

and, even for linear control, they must be designed new whenever one resets the basic system parameters.

3. Conventionally designed automatic controlled devices have relatively narrow performance bands.

2.7 DIGITAL EQUIVALENT TO CONVENTIONAL CONTROLLER

The operation of an ideal PID controller is described by equation

$$V = V_o + K_c[e + 1/T_I \int e\,dt + T_d\, de/dt] \quad \ldots (2.1)$$

In conventional control applications, a controller, whose output approximates the right side of equation 2.1 can be built through the use of pneumatic components of op-amps, integrators and summers. In computer-controlled applications a discrete equivalent in equation 2.1 is employed. In the development of algorithms that are based on Z-transforms we specify the nature of the response to be achieved, whereas in the digital equivalent of the PID controller we "adjust" the constants Kc,Ti and Td so as to achieve the desired response. The computer controlled system containing the PID control algorithm can be simulated and the constants are adjusted so as to minimise the value of the integral of the type.

To obtain the digital equivalent of the PID controller the derivative and the integral terms of the equation 2.1 are numerically approximated to give an expression for the output of the algorithm the nth sampling instant.

Thus

$$V_n = V_o + K_c [e_n + (T/T_i) \sum e_i + (T_d/T) (e_n - e_{(n-1)})] \quad \ldots (2.2)$$

Where V_n controller output at nth sampling instant

e_n error(setpoint-measurement at the nth sampling instant)

V_o Steady state output of the control algorithm that gives zero error.

Equation 2.2 is referred to as the "position" form of the control algorithm, since the actual controller output is computed. To give an alternate form of algorithm we write the expression for the controller output at the (n-1)th sampling instant as

$$V_{n-1} = V_o + K_c [e_{n-1} + (T/T_i) \sum e_i + (T_d/T) (e_{n-1} - e_{(n-2)})] \quad \ldots (2.3)$$

Then we subtract eq 2.3 from eq 2.2 to obtain the

$$V_n = V_{n-1} + K_c [(e_n - e_{n-1}) + (T/T_i) e_n + (T_d/T) (e_n - 2e_{n-1} + e_{(n-2)})] \quad \ldots (2.4)$$

equation 2.4 is referred to as the velocity form of the PID algorithm, because it computes the incremental output instead of the actual output of the controller. The velocity form of the algorithm also provides some protection against reset wind-up, because it does not incorporate sums of error sequences.

SUMMARY

The basic components of the process control system and their functions have been discussed in this chapter. The performance indices and the design aspects of the process control system has been dealt with. The types of feedback controllers, the concept of stability, limitations of conventional controllers and the digital equivalent of controller are also presented.

FUZZY LOGIC CONTROLLER

In order to design a controller, the control algorithm should be described by some means. A classical control theory gives differential equations or transfer functions and a modern control theory gives a first-order vector matrix differential equation based on the state-space method. In these approach a controller designer has to possess knowledge about mathematics and the system under control.

However, human experts of ripe experience can skillful control plants, machines, vehicles, etc., even though the systems under control are very complex. These human experts mostly utilise know-how which have been summarised from a long experience including successes and faults, and are represented with IF-THEN rules including fuzzy linguistic terms. They very often succeed control the systems reasonably with these inexact information and without any calculation such as sin(wt), cos(wt), exp(x). This suggests that there is another algorithm which facilitates to control a complicated system by a simple and inexact knowledge base. One candidate is a fuzzy inference which produces a conclusion from a knowledge base and a fact.

In figure 3.1 we see the process flow logic for a typical fuzzy logic controller. The process starts at the bottom of the figure. The input is read from the sensors as an electrical signal. The signal is converted into a meaningful representation and then "fuzzified", that is, the values are converted to their fuzzy representations. These sensor values execute all the rules in the knowledge repository that have the fuzzified input in their premise resulting in a new fuzzy set representation for each output variable. Centroid defuzzification is used, in the majority of the cases, to develop the expected value for each of the output variables.The output value adjusts the setting of an actuator, which adjusts the state of the physical system. The new state is picked up by the sensors and the entire process begins once more.

3.1 BASIC CONFIGURATION OF FUZZY LOGIC CONTROLLER

An FLC basically comprise of four parts

(1) Fuzzification interface.

(2) Knowledge base.

(3) Decision making knowledge.

(4) Defuzzification interface.

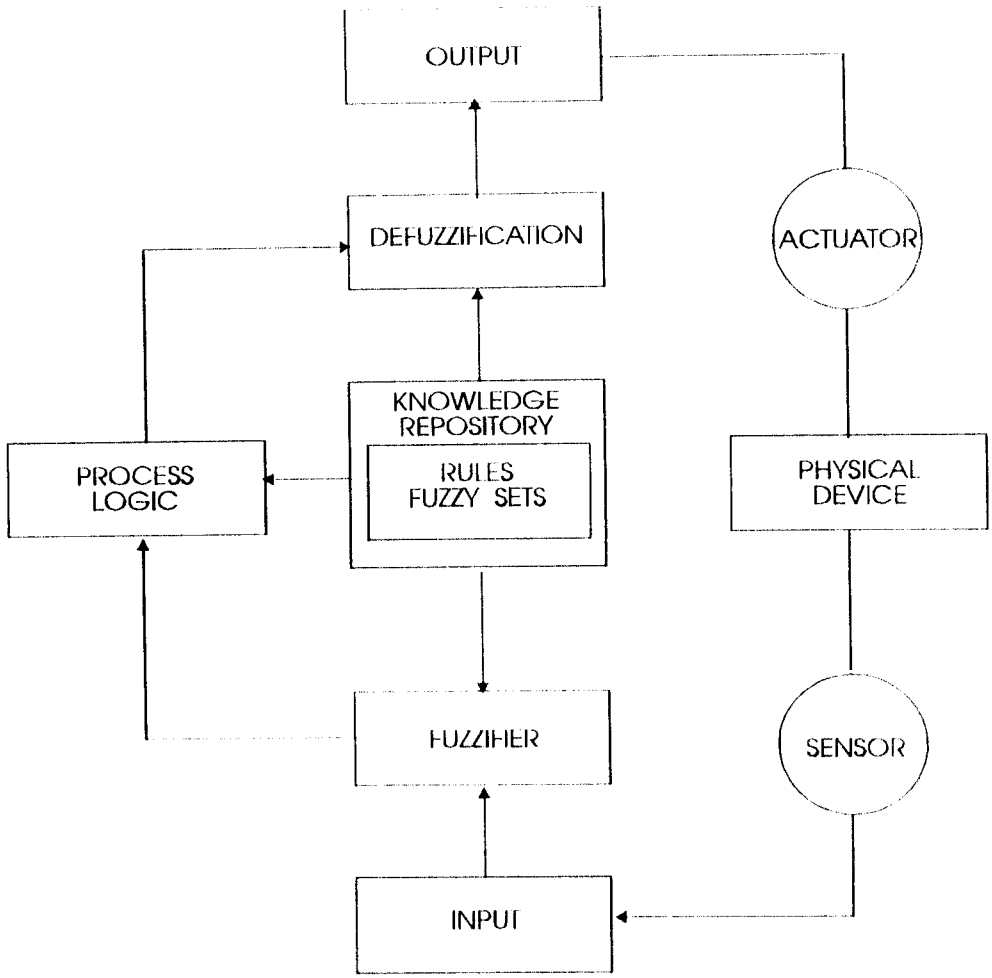


FIG 3.1

BLOCK DIAGRAM OF FUZZY LOGIC CONTROLLER

3.1.1 FUZZIFICATION INTERFACE

This involves the following function

* Measures the values of the input value.

* Performs a scale of mapping the transforms the range of input variable into corresponding universe of discourse.

* Performs the function of fuzzification that converts the input data into suitable linguistic values that may be viewed as labels of fuzzy sets.

3.1.2 KNOWLEDGE BASE

The knowledge base comprises knowledge of the application domain and the attendant control rules. It consists of a "data base" and a "linguistic control base".

(a) The database provided necessary definitions which are used to define linguistic data rules and fuzzy data manipulations in an FLC.

(b) The rule base characterises a control goals and control policies of the domain experts by means of a set of linguistic control rules.

3.1.3 DECISION MAKING LOGIC

The decision making logic is the kernel of the FLC, it has the capability of simulating human decision making based on fuzzy concepts and of inferring fuzzy control actions employing fuzzy implications and the rule of inference in fuzzy logic.

3.1.4 DEFUZZIFICATION INTERFACE

The defuzzification interface performs the following functions.

(a) A scale mapping, which converts a range of values of output variable into corresponding universe of discourse.

(b) Defuzzification, which yields a non-fuzzy action from an inferred fuzzy control action.

3.1.5 DEFUZZIFICATION PROCEDURE

The most often used defuzzification method are

* Center of Area/Gravity method
* Center of sums method
* Height defuzzification

3.1.6 CENTRE OF AREA/GRAVITY

The centre of area method is the best well known defuzzification method. In the discrete case the centre of gravity is calculated as

$$\sum W_i \mu_i / \sum \mu_i$$

Where μ_i is a grade of membership and

W_i is the weight of the ith element of the universe of discourse. In the case of centre of gravity method the overlapping area is not reflected in the formula. The operation is computationally more complex and therefore results in quite slow inference cycles.

3.1.7 CENTRE OF SUMS METHOD

Centre of sums method is a faster defuzzification method in which overlapping areas are reflected more than once. We obtain the centre of sums by

$$\sum W_i \sum \mu_i \mu_i$$

where μ_i – Membership value of clipped fuzzy set.

3.1.8 HEIGHT DEFUZZIFICATION METHOD

The height method is both a very simple and very quick method. let C_k be the peak value of the membership function and f_k is the height of the clipped fuzzy set.

$$= \sum C_k f_k / \sum f_k$$

3.2 FUZZY KNOWLEDGE BASED CONTROLLERS (FKBC)

3.2.1 PD LIKE FKBC

The equation giving the conventional PD controller is

$$u = K_p * e + K_d * \Delta e$$

Where K_p and K_d are the proportional and differential gain co efficients. Then the PD like FKBC consists of rules, the symbolic description of each rules given as

If e (k) is <property symbol> and ^e(k) is <property symbol> THEN u(k) is <property symbol>

Where <property symbol> is the symbolic name of the linguistic value. The natural language equivalent of the above symbolic description reads as follows:

For each sampling time k, IF the value of error has the property of being<linguistic value> and the value of the change of error has the property of being <linguistic value>

THEN the value of the control output has the property of being <linguistic value>.

Thus the final symbolic representation of the above rule is

IF e is <property symbol> and ^e is <property symbol>

THEN u is <property symbol>

3.2.2 PI like FKBC

The equation giving a conventional PI controller is

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

Where K_p and K_i are the proportional and integral gain co-efficient. When the derivative, with respect to time, of the above expression is taken, it is transformed into an equivalent expression.

$$\Delta u = K_p * \Delta e + K_i * e$$

The PI like FKBC consists of rules of the form.

If e is property symbol and Δe is property symbol

Then Δu is property symbol .

In this case, to obtain the value of the control output variable u(k), change of control output u(k) is added to u(k-1). It is to be stressed here that this takes place outside the PI like FKBC and is not reflected in the rules themselves.

3.2.3 P LIKE FKBC

A symbolic representation of a rule for a P like FKBC is given as

If e is property symbol Then u is property symbol .

3.2.4 PID LIKE FKBC

The equation describing a conventional PID controller is

$$u = K_p * e + K_d * \Delta e + Ki * \int (e\, dt)$$

Thus the symbolic expression for a rule of a PID like FKBC is

If e is property symbol and ^e is property symbol

Then u is property symbol

Where ^e is the sum of errors computed as

^e(k) = e(i), where i varies from 1 to k-1

3.3 THE CONVENTIONAL PID CONTROLLER

In contrast to an FLC, a conventional proportional-integral derivative PID controller is based on a rigorous mathematical model of some linear process. These models use a set of equations that describes the stable equilibrium state of the control surface through coefficients assigned to the proportional, integral and derivative aspects of the system as figure 3.1 illustrates, a conventional controller reads a sensor value, applies the mathematical model, and produces an output from the mathematical algorithm.

The PID model may appear simpler and thus, perhaps, more economical but we should not easily make this assumption. In fact, fuzzy controllers are often more easily prototyped and implemented, generally have an equal performance profile with PID systems, are simpler to describe and verify, can be maintained and extended with a higher degree of accuracy in less time, and, due to their reliance

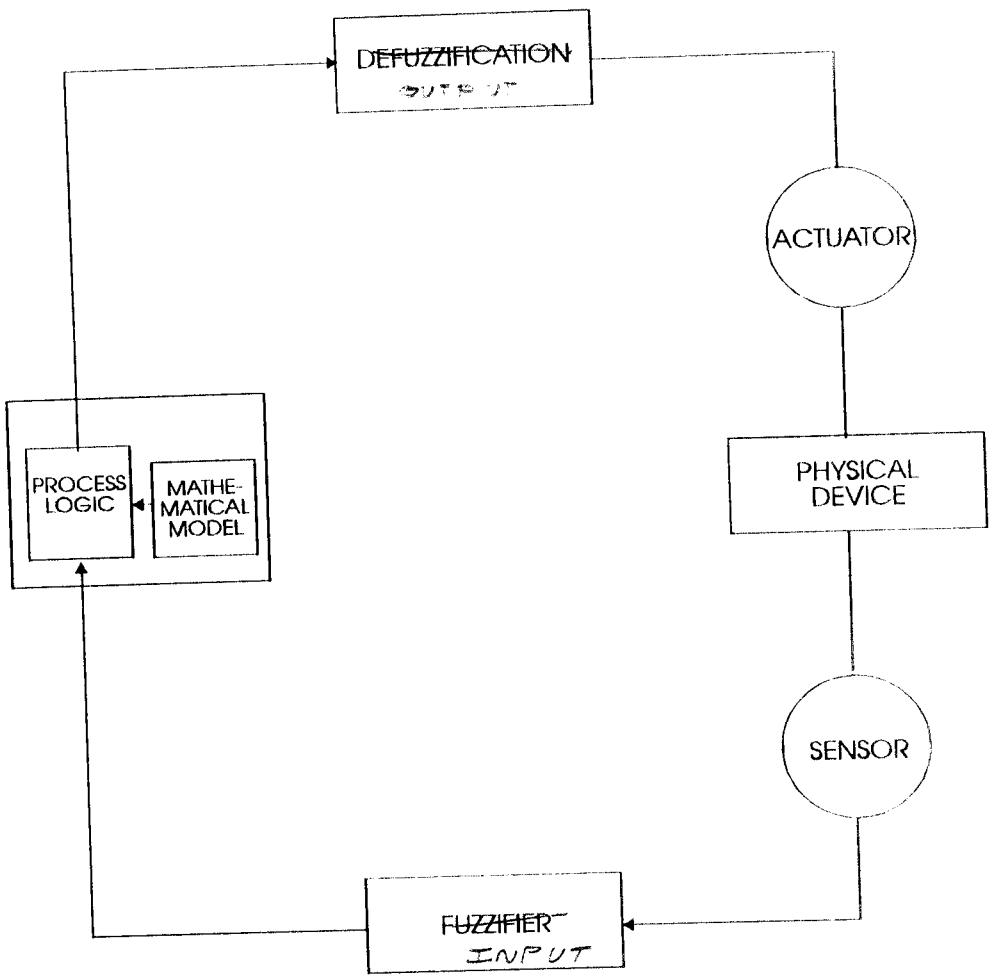


FIG 3.2

BLOCK DIAGRAM OF PID CONTROLLER

on rules and knowledge, give their environments a higher machine intelligence quotient.

3.4 COMPARISON WITH A TRADITIONAL PID CONTROL

It is necessary for a designer to know the difference between a traditional PID control and a fuzzy logic control.

A typical control system employing a PID controller is shown in figure 3.2.

The output state of the system under control is detected by a sensor. It is compared with the input signal to derive an error signal e(t), where the input signal is externally given and represents a desired state of the system. The error signal is delivered to a controller to produce an appropriate manipulating signal m(t), which change the state of the system under control.

In a control system, there are two main objectives. The first one is to make the state (or output) of the system to be very close or equal, if possible, to the set point (or reference input). In other words, a small steady-state error e(t) or a high steady-state accuracy is desired. The second one is to maintain the transient performance of the system within reasonable limits.

In order to design a controller of high steady-state accuracy and high speed settling, we need a linear combination of three control actions, (i.e.)., proportional control action, integral control action and derivation control (PID) action. It is called a PID control and characterised by the following equation.

$$m(t) \quad K_p\, e(t) + K_I \int e(t)\, dt + K_D\, (de(t)\, dt)$$

This controller has three inputs and one output. Let us consider a controller of two inputs and one output to visualise the relationship between input and output. The first and the second terms in the above equation is considered and differentiated to

$$m\,(t) \quad K_p\, e\,(t) + K_I e(t).$$

Which is the linear description of a traditional PI controller. Three-dimensional control surface obtained using a PI controller can describe only a simple plane. In contrast to the PI controller the fuzzy controller exhibits a very complicated and curved surface which can never be described by a linear combination of input variables such as equation1.

Furthermore, it is very easy to change the control surface of the fuzzy logic controller in accordance with the change of the system under control. For example if we want to pull up the center of the control surface, we have only to change the label of the consequent in the corresponding rule. If you want to change exclusively a small part of the control surface, you have to narrow the support of

the corresponding membership function and increase the number of labels to achieve the effective assignment of rules. This means the exceptional or irregular points on the control surface can be easily described in fuzzy logic controller. Thus fuzzy rules are very suitable for describing a sophisticated system by fuzzy linguistic terms. Of course, the fuzzy inference can be easily modified by the concept of learning, self organising, adaptational etc.

SUMMARY

This chapter discuss about the fuzzy logic controllers, a brief comparison made between the controllers.

WIND ENERGY CONVERSION SCHEME (WECS)

In remote locations where the utility grid does not exist, standalone wind energy scheme (WECS) can be used to feed the local electrical load. The WECS consist of a self excited induction generator driven by a emulated wind turbine, a diode bridge rectifier and a PWM inverter feeding a load as shown in figure 4.1. By controlling the pulse width of the PWM inverter it is possible to convert its output voltage and the power transferred to the local load, indirectly controlling the power extracted form the WECS.

The WECS consists of

1. Vertical axis wind turbine.
2. Self excited induction generator.
3. Uncontrolled rectifier.
4. DC link and input filter.
5. PWM inverter.

4.1 VERTICAL AXIS WIND TURBINE

The vertical axis wind turbine acts as a prime mover to drive the self excited induction generator. In practice, the wind turbine is coupled to the

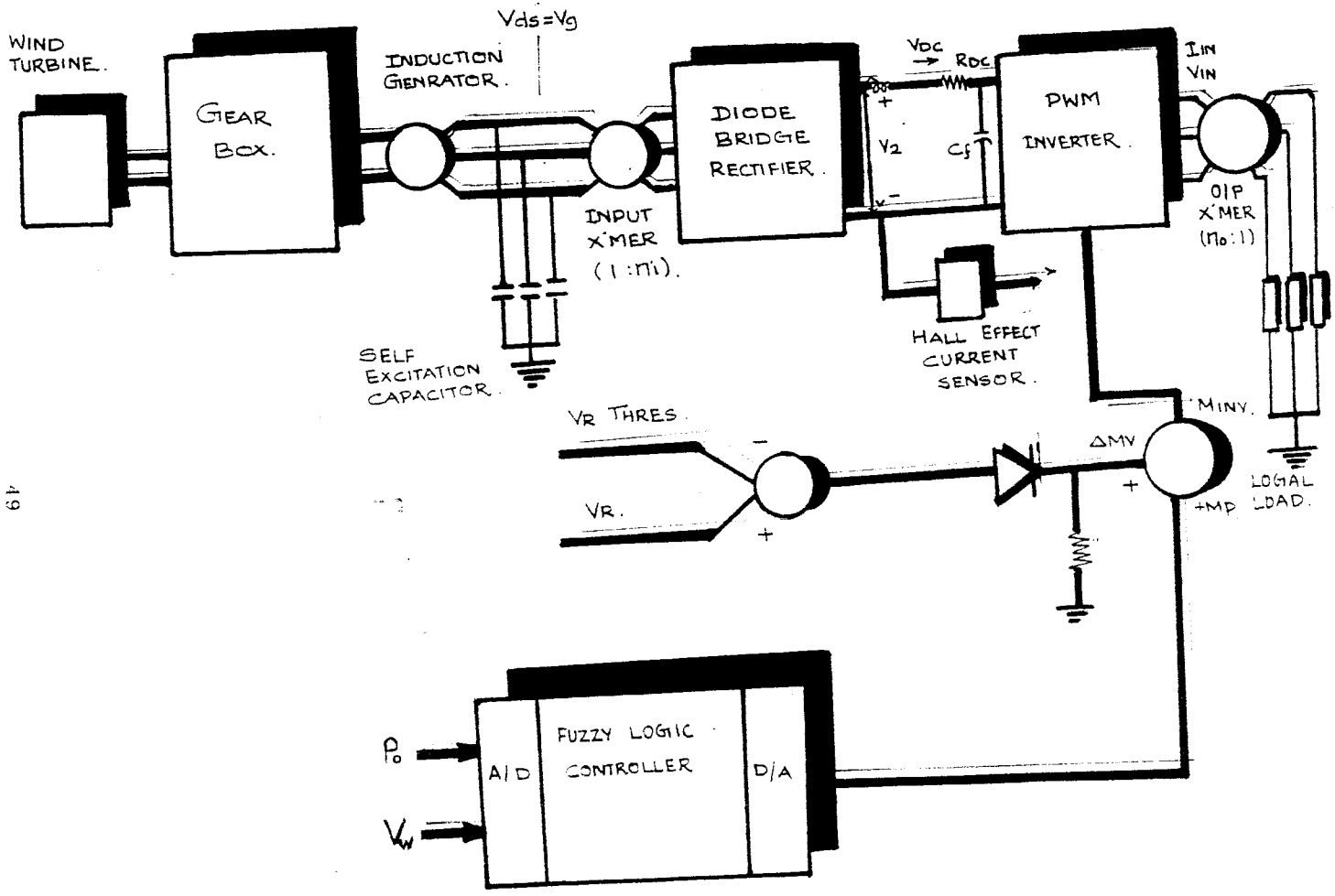


FIG 4.1 BLOCK DIAGRAM OF WECS

generator shaft through a set-up gear box(1:$n_{gearbox}$) so that the generator runs at a higher rotational speed W_m inspite of the low speed W_t of the wind turbine. The torque T_t at the turbine shaft is a function of its rotor speed and wind velocity and can be expressed as

$$T_t \; n_{gearbox}*T_m \quad 0.5*\rho*R*A*C_t*V_w*V_w$$

Where C_t is the average torque conversion co-efficient.C_t is averaged over one revolution of the wind turbine rotor and is a non linear function of the tip speed ratio $Y=W_t*R/V_w$

where, R is radius of wind turbine

The torque at the generator shaft T_m can be related to T_t by the gear box ratio. The vertical axis wind turbine characteristics are emulated using a separately excited D.C. machine fed from a power amplifier controlled by an analog computer.

4.2 SELF EXCITED INDUCTION GENERATOR

It is essentially an induction machine driven by a prime mover (wind turbine) and having a source of reactive power (self excitation capacitor bank).

4.3 UNCONTROLLED RECTIFIER

It is a three phase diode bridge rectifier which is used to convert the variable magnitude, variable frequency voltage at the induction generator terminal to D.C. The D.C. voltage V_r at its output can be expressed in terms of the peak phase voltage $V_{ds}=V_p$ of the generator and the input transformers turns ratio 1:ni.

$$V_r = (3\sqrt{2}/\pi)(\sqrt{3}/\sqrt{2})V_{ds} * n_i$$

4.4 D.C. LINK AND INPUT FILTER

The input filter consists of a series reactor and a shunt capacitor. The series reactor reduces the current ripple content in the rectifier output current and the shunt capacitor reduces the ripple content in the D.C.link voltage providing a relatively stiff voltage source for the PWM inverter. The dc link current is governed by the following differential equation

$$P_i dc \quad I I_{dc} * (V_r - V_i - R_{dc} * I_{dc})$$

where,

R_{dc} and L_{dc} are the dc link reactors resistance and inductance respectively.

I_{dc} is the dc link current (neglecting the small charging/discharging filter capacitor).

V_i is the dc voltage at the inverter input.

The dc power transferred over the dc link to the local load and is given by,

$$P_o = V_i * I_{dc}$$

4.5 PWM INVERTER

DC to AC conveters are known as inverters. The function of inverter is to change a DC input to symmetrical AC voltage of desired magnitude and frequency. The output voltage could be fixed or variable at a fixed or variable frequency. A variable output voltage can be obtained by varying the input DC voltage and maintaining the gain of the inverter constant. If the DC input voltage is fixed and it is not controllable, a variable output voltage can be obtained by varying the gain of the inverter which is normally accomplished by pulse width modulation (PWM) control within the inverter. The inverter gain may be obtained as the ratio of AC output to DC input voltage. The output voltage waveform of the ideal inverters should be sinusoidal. The waveform of the practical inverters are non-sinusoidal and contain certain harmonics.

Inverters are classified into

(1) Single phase inverters

(2) Three phase inverters

(1) Single phase inverter

It consists of two choppers. It consists of two transistors and is designed such that two transistors are not turned on at the same time. This is also called as half-bridge inverter.

(2) Three phase inverters

Three phase inverters are normally used for high power applications. Three single phase (half or full) bridge inverter. The gating signals of single phase inverters should be advanced (or) delayed by 120 degree with respect to each other in order to obtain three phase balanced voltages. If the output voltages of single phase inverters are not perfectly balanced in magnitude and phases, the three phases output voltages will be unbalanced.

VOLTAGE CONTROL OF SINGLE PHASE INVERTERS.

In many applications, it is often required to control the output voltage of inverters.

1. To cope with the variations of DC input voltage.
2. For voltage regulation of inverters.
3. For the constant volts/frequency control requirements

There are varies techniques to vary the inverter gain. The most efficient method of controlling the gain is to incorporate pulse-width modulation control within the inverters.

The commonly used techniques are

1. Single pulse-width modulation
2. Multiple pulse width modulation

3. Sinusoidal pulse width modulation

4. Modified sinusoidal pulse width modulation

5. Phase displacement control

VOLTAGE CONTROL OF THREE PHASE INVERTERS

The three phase inverter may be considered as three single phase inverters and the output of each single phase inverter is shifted by 120 degree. The above voltage controlled techniques also applicable to three phase inverters.

SUMMARY

In this chapter basic components used in the wind energy conversion scheme are discussed, and their operation in the entire system are described.

FUZZY LOGIC CONTROLLER FOR WECS

The conventional PID controller to track and extract maximum energy, requires quite a bit of tuning to obtain a fast and dynamically acceptable response. The chief reason is that, it is generally implemented using operational amplifier circuits whose parameters are adjusted for an operation point based on a piece wise linear model of the non-linear system these circuits has a tendency to shift with age and temperature causing degradation of the system performance. To overcome this difficulty a controller based on the fuzzy control algorithms is developed and implemented and it does not require a detailed mathematical model of the system and its operation is governed by a set of rules. It is easy to implement and same performance is ensured over the years. Having tested the effectiveness of the fuzzy controller by computer simulation, the hardware of the energy conversion scheme is given and the software realisation of the fuzzy control algorithms is carried out using a personal computer. The experimental result indicates that good dynamic response can be achieved by the fuzzy controller. Moreover, since the system parameters are not need in the implementation of the system, and due to the inherent feature of the highly adaptive capability possessed by the fuzzy controller, the performance of the

controlled scheme is quite robust and insensitive to the parameters and operating condition changes.

5.1 FUZZY CONTROL ALGORITHMS FOR WIND ENERGY CONVERSION SCHEME

Generally, the requirements for high performance energy conversion scheme are:

1. Fast tracking of maximum power changes without overshoot.
2. The steady state error in the command tracking must be zero. In order to achieve this, the outer feedback loops for the controlled variables of the system must be added.

Since dynamic system model is not necessary for the fuzzy controller design, and the performance of the fuzzy controller is insensitive to the parameter changes, it is very suitable in the application. Theoretical basis for the fuzzy set theory have been introduced in the chapter1. However, unfortunately there is no mature guidance for fuzzy control algorithm determination in the following, the development of the fuzzy controller for the energy conversion system is described in detail.

5.2 DYNAMIC SIGNAL ANALYSIS

For convenience of incorporating the intuition and experience into the fuzzy control algorithm, the behaviour of the dynamic response is first investigated. The error and change in error of the system are defined as,

$$e(k) \quad reference - u(k)$$

$$\hat{}e(k) \quad e(k) - e(k-1)$$

Where u(k) is the system response in the kth sampling interval, e(k) is the error in the kth sampling interval and ^e(k) is the error change in the k th sampling interval.

The general waveforms of the system response and the command are drawn in the figure 5.1. According to the magnitude of e and sign of ^e , the response plane is roughly divided into four areas. The index used for identifying the response area is defined as

$$a1: e \quad 0 \text{ and } \hat{}e \quad 0$$

$$a2: e \quad 0 \text{ and } \hat{}e \quad 0$$

$$a3: e \quad 0 \text{ and } \hat{}e \quad 0$$

$$a4: e \quad 0 \text{ and } \hat{}e \quad 0$$

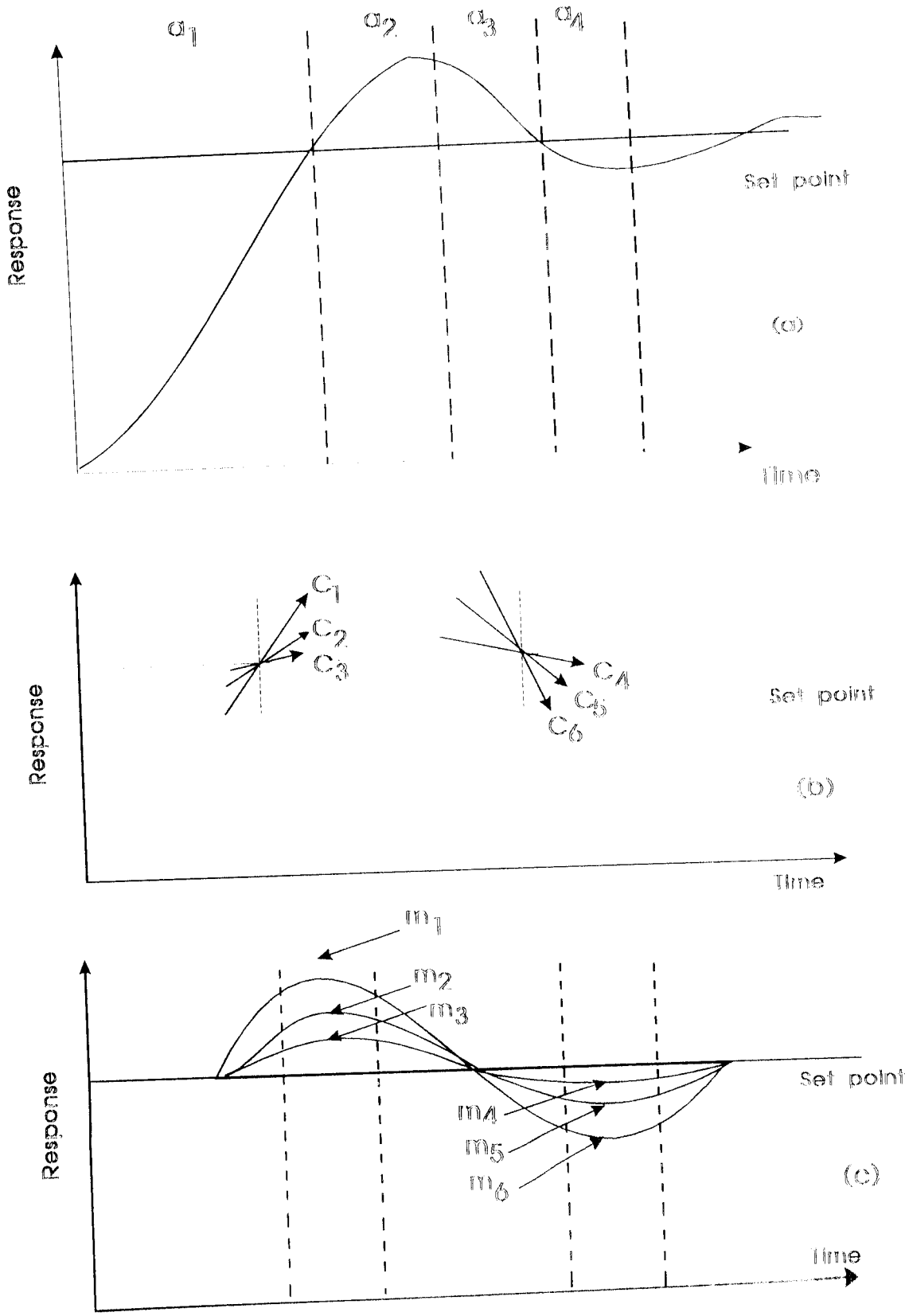


Figure 5.1

THE DYNAMIC SYSTEM RESPONSE

Table 5.2

THE STATE PLANE OF ERROR AND ERROR CHANGE

_e e	NB	NM	NS	ZE	PS	PM	PB
NB	..	..	..	c1	..	..	..
NM	..	a2	..	c2	..	a1	..
NS	..	..	..	c3	..	..	..
ZE	m6	m5	m4	ZE	m3	m2	m1
PS	..	..	..	c4	..	..	..
PM	..	a3	..	c5	..	a4	..
PB	..	..	..	c6	..	..	..

For further increases in the resolution of the behaviour representation, the response around the set point and the extremes in figure 5.1A are emphasised in figure 5.1B and 5.1C, respectively.

The cross over index Ci , for identifying the slope of the response across the set point is defined as

$$C1:(e\ 0\ --\ e\ 0)\ and\ \hat{}e\ \ 0$$

$$C2:(e\ 0\ --\ e\ 0)\ and\ \hat{}e\ \ 0$$

$$C3:(e\ 0\ --\ e\ 0)\ and\ \hat{}e\ 0$$

$$C4:(e\ 0\ --\ e\ 0)\ and\ \hat{}e\ 0$$

$$C5:(e\ 0\ --\ e\ 0)\ and\ \hat{}e\ \ 0$$

$$C6:(e\ 0\ --\ e\ 0)\ and\ \hat{}e\ \ 0$$

Also, the magnitude index for representing the extent of overshoot and undershoot is defined as

$$M1:\hat{}e\ \sim 0\ and\ e\ \ 0$$

$$M2:\hat{}e\ \sim 0\ and\ e\ \ 0$$

$$M3:\hat{}e\ \sim 0\ and\ e\ 0$$

$$M4\ \hat{}e\ \sim 0\ and\ e\ 0$$

$$M5\ \hat{}e\ \sim 0\ and\ e\ \ 0$$

$$M6\ \hat{}e\ \sim 0\ and\ e\ \ 0$$

TABLE 5.2

e \ ^e	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZE
NM	NB	NB	NB	NM	NS	ZE	PS
NS	NB	NB	NB	NM	NM	ZE	PS
ZE	NB	NM	NS	ZE	PM	PM	PB
PS	NM	NS	ZE	PS	PM	PB	PB
PM	NS	ZE	PS	PM	PB	PB	PB
PB	ZE	PS	PM	PB	PB	PB	PB

5.3 LINGUISTIC CONTROL RULES

The three type of index previously mentioned can be combined and shown in the state plane table 5.1 for reference, where the qualitative statements are quantized using the linguistic set defined.

$$\{NL, NM, NS, ZE, PS, PM, PL\}$$

Where 'N' is negative, 'P' is positive, 'L' is large,'M' is medium, 'S' is small, and 'ZE' is zero. The linguistic control rules defined according to the figure 5.1 and table 5.2 are listed in table 5.1. The conditional rules are implied in the table, see for example, the element of 1 row and 7th column, that implies that

If e is PL and ^e is NL

Then control input is PL

5.4 RULE BASE

The design parameters of the rule base include:

1. Choice of process state and control output variables.
2. Choice of the content of the rule antecedent and the rule consequent.
3. Choice of term set for the process state and control output variables.
4. Derivation of the set of rules.

5.4.1 CHOICE OF VARIABLES AND CONTENT OF RULES

The choice of designing a P-,PD-,PI- or PID like Fuzzy Knowledge Based control (FKBC) implies the choice of process state and control output variables as well as content of the rule antecedent and the rule consequent for each of the rules. The process state variables representing the contents of the rule antecedent (IF part of a rule) are selected amongst the

1. Error denoted by e.
2. Change of error denoted by ^e.

The control output (process input) variables representing the contents of the rule consequent (THEN part of the rule are selected amongst

1. Change of control output, denoted by ^u.
2. Control output denoted by u.

Furthermore, by analogy with a conventional controller we have that

1. e(k)=reference -u(k).
2. ^e(k)=e(k)-e(k-1).

In the above expressions, reference is the desired process output, u(k) is the process output variable (controlled variable), k is the sampling time.

5.4.2 CHOICE OF TERM SET

The linguistic values, members of the term set are expressed as tuples of the form <value sign, value magnitude>, example (positive big), (negative small) etc.. The value sign component of such a tuple takes on either one of the two values positive/negative. The value magnitude component can take on any number of linguistically expressed magnitudes, examples:- {zero, small, medium, big} or {zero, small, big} or {zero, very small, small, medium, big, verybig} etc.. The meaning of a tuple in the case of PI like FKBC, can be summarised as follows

1. Linguistic values of e with a negative sign mean that the current process output y has a value below the set point since e(k) = set point - y(k) < 0. The magnitude of a negative value describes the magnitude of the difference reference - y. On the other hand linguistic values of e with a positive sign mean that the current value of y is above the setpoint. The magnitude of such a positive value is the magnitude of the difference (set point -y).

2. Linguistic values of ^e with a negative sign mean that the current process output y(k) has increased when compared with its previous value y(k-1) since ^e(k) = -(y(k)-y(k-1))<0.

The magnitude of such an negative value is given by the magnitude of this increase. Linguistic values of ^e(k) with a positive sign mean that y(k) has decreased its value when compared to y(k-1). The magnitude of such a value is the magnitude of the decrease.

3. A linguistic value of "zero" for e means that the current process output is at the set point. A "zero" of ^e means that the current process output has not changed from its previous value (ie) -(y(k)-y(k-1)) =0.

4. Linguistic values of ^u(k) with a positive sign mean that the value of the control output u(k-1) has to be increased to obtain the value of the control output for the current sampling time k. A value with a negative sign means a decrease in the value of u(k-1). The reason for this is that u(k) = u(k-1) + ^u(k). The magnitude of a value is the magnitude of increase/decrease of the value of u(k-1).

Suppose now that the term set of e,^e,^u all have been chosen equal in size and contain the same linguistic expressions for the magnitude part of the linguistic values, i.e., LE=L^E=L^U={NB,NM,NS,ZE,PS,PM,PP} where, for example, NB reads "negative big" and PS reads "positive small". The rule base of pi like FKBC

and describe the operational meaning of the if-then rules. The rule base is shown in table 5.2.

The cell defined by the intersection of the first row and the first column represents a rule such as,

if e(k) is NB and ^e(k) is NB then ^u(k) is NB

Now the set of rules can be divided in the following five groups:

Group 0: In this group of rules both e and ^e are (positive or negative)small or zero. This means that the current value of the process output variable y has deviated from the set point but is still closed to it. The amount of change ^u(k), in the previous control output u(k-1) prescribed by this rules is also small or zero in magnitude and is intended to correct the small deviations from the set point. Therefore, the rules in this group are related to the steady-state behaviour of the process.

Group 1: For this group of rules e is negative big or medium which implies that y(k) is significantly above the setpoint. At the same time, since ^e(k) is positive, this means that y is moving towards set point. The amount of change ^u which the rules of this group introduce to the previous control output u(k-1) is

intended to either speed up or slow down the approach to the set-point. For example, if y(k) is much above the set-point(e(k) is NB) and it is moving towards the set-point with a small step(^e(K) is PS) then the magnitude of this step has to be significantly increased (^u(k) is NM).

Group 2 : For this group of rules is either close to the set-point(PS,ZE,NS) or is significantly below it (PM,PB). At the same time, since ^e is positive, y is moving away from the set-point. Thus a positive change ^u(k) in the previous control output u(k-1) is intended to reverse this trend and make y, instead of moving away from the set-point, to start moving away towards it.

Group 3: For this group of rules e is positive medium or big which means that y(k) is significantly below the set-point. The amount of change ^u which the rules of this group introduce to the previous control output u(k-1) is intended to either speed up or slow down the approach to the set-point. For example, if y(k) is much below the set-point (e(k) is PB) and it is moving towards the set-point with a somewhat large step (e(k) is NM then the magnitude of this step has to be only slightly enlarged (^u(k) is PS).

Group 4: For this group of rules e is either close to the setpoint (PS,ZE,NS) or is significantly above it (NM,NB). At the same time, since ^e is negative, y is moving away from the setpoint. Thus a negative change ^u(k) in the previous control output u(k-1) is intended to reverse this trend and make y, instead of moving away from the set-point,

start moving towards it.

In the context of the above five groups of rules, we can see that the size of the term set determines the granularity of the control action of the FKBC. If one desires better control resolution around the set_point then one can consider a larger range of linguistic values for NS, ZE, and PS, e.g., PZE for "positive zero",NZE for "negative zero",PVS for "positive very small". However, one should be aware of the fact that the use of term sets with large size leads to the increase in the number of rules. For example, if each of the term sets of e and ^e has 10 elements then the maximum number of rules in the rule base is zero.

5.4.3 DERIVATION OF RULES

There are three major approaches to the derivation of the rules of a FKBC. These three approaches complement each other and it seems that combination of

them would be necessary to construct an effective approach to the derivation of rules.

APPROACH 1

This approach is the one that is most widely used today. It is based on the derivation of rules from the experience-based knowledge of the process operator and/or control engineer. The approach is realised using two type of techniques.

1. An introspective verbalisation of experience-based knowledge
2. Interrogation of a process operator and/or control engineer using a carefully organised questionnaire.

Both the techniques help in providing an initial prototype version of the rule base, specific for a particular application domain. Consequent tuning of membership function and rules is necessary next step. For example in the case of a PI- like FKBC, an initial set of rules, selected using either one of the above two techniques, is implemented on the process under control. The subsequent response of the closed-loop system to non-aperiodic disturbance is monitored in the phase plane. The later is composed of error and change of error, and obtained trajectory is used to update the set of initial rules. This procedure is then repeated as often as required until an acceptable response is obtained.

APPROACH 2.

This approach still in its early stages uses a linguistic description viewed as a fuzzy model of the process under control, to derive the set of rules of a FKBC.

The general controller design ideas using such a fuzzy process model a already described in the previous subsection. Based on the fuzzy process model, either a set of rules or an explicit fuzzy relation describing the FKBC is derived.

APPROACH 3

The approach, also in its early development stages release on the existence of a conventional process model, usually a non-linear one.

5.5 CHOICE OF MEMBERSHIP FUNCTION

The linguistic values taken by the variables in the rules antecedent and the rule- consequent, and symbolic representation of the rules are good enough to allow some quality analyses concerning the stability of the closed-loop system. However, for needs of quantitative description of the behaviour of the closed loop system, involving the computation of quantitative control output, one needs a quantitative interpretation of the meaning of the linguistic values.

The most popular choices for the shape of the membership function include triangular, trapezoidal, linear and bell-shaped functions as shown in figure 5.2. These three choices can be explained by the ease with which a parametric, functional description of a membership function can be obtained, stored with minimal use of memory, and manipulated efficiently, in terms of real-time requirements, by the inference engine.

5.6 CROSSPOINTS

Let M1 and M2 be two membership functions representing the meaning of two different linguistic values of x which are elements of the term set Lx. A cross point between M1 and M2 is that value Xcross in X such that

$$M1_{(Xcross)} \quad M2_{(Xcross)} \cdot 0.$$

Furthermore a cross point level is defined by the degree of membership of Xcross, that is M1(Xcross) which, by the definition the crosspoint, is the same as M2(Xcross). It may be the case that two membership function defining the meaning of two different linguistic values may have more than one cross-point. The number of cross points between two membership functions is called the cross-point ratio. Fig 5.3 illustrates the above notions in the case of triangular membership functions.

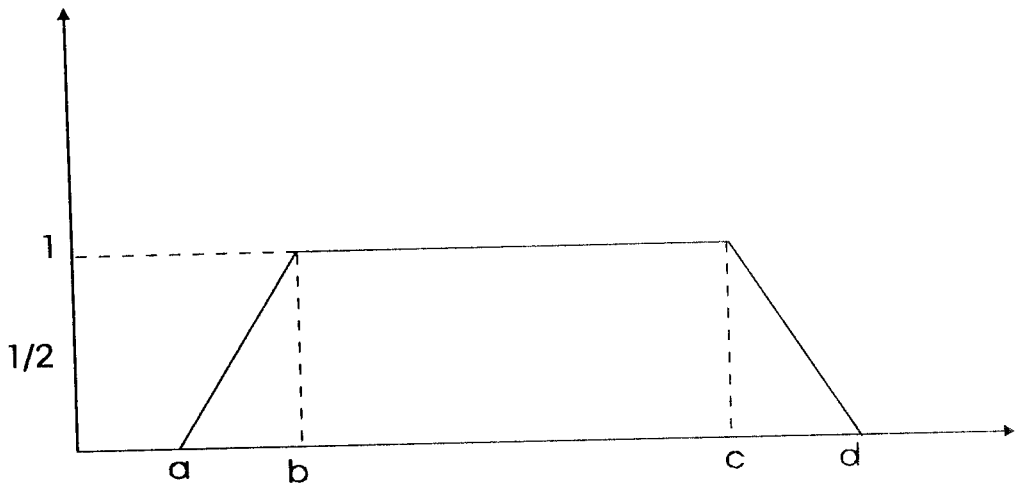


FIG. 5.2 a TRAPEZOIDAL MEMBERSHIP FUNCTION

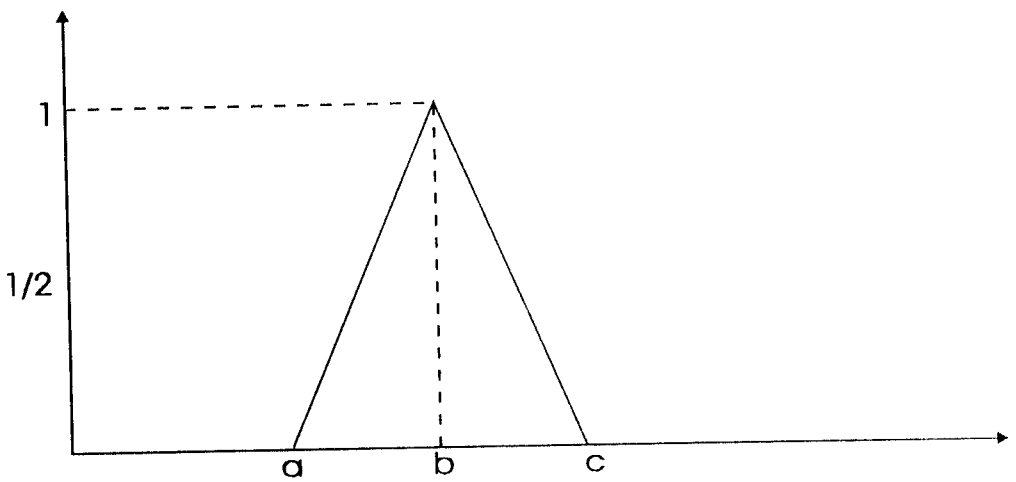


FIG. 5.2 b TRIANGULAR MEMBERSHIP FUNCTION

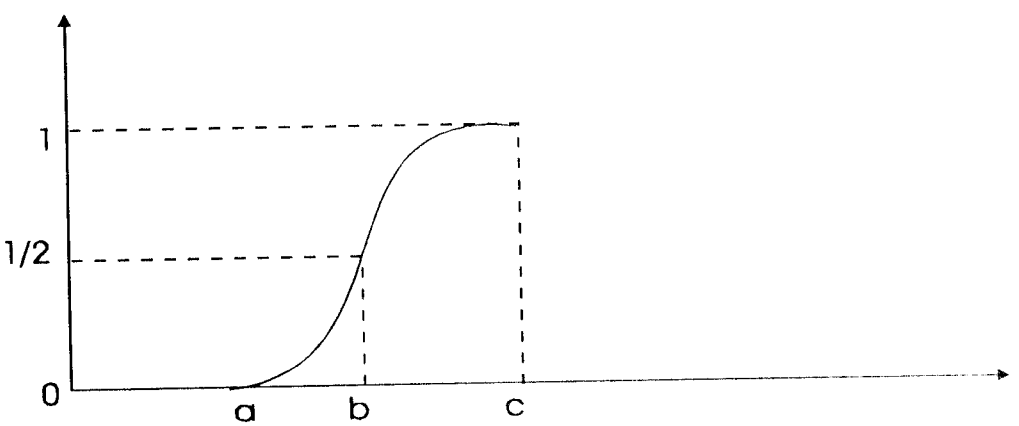


FIG. 5.2 c S.SHAPED MEMBERSHIP FUNCTION

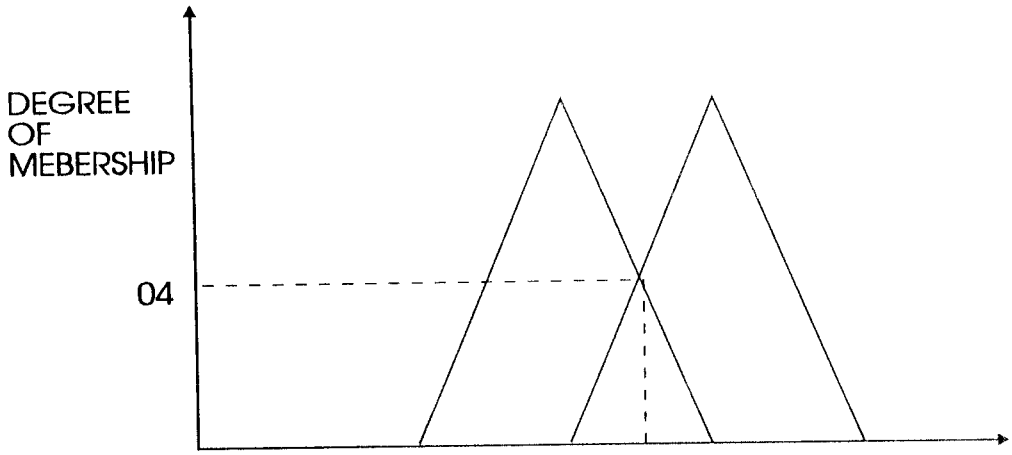


FIG. 5.3a XCROSS

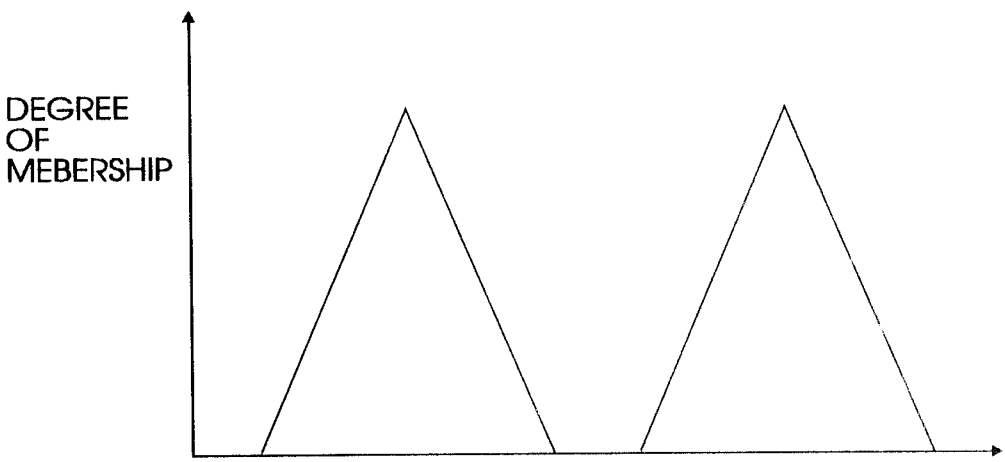


FIG. 5.3b NO CROSS POINT

5.7 INFLUENCE OF CROSS-POINT LEVEL

First the mapping from a term-set, say LX to membership functions on the domain of error 'e' has to be such that the crosspoint level for every two membership function is greater than zero. This means that every crisp value of error belong to atleast to one membership function with degree of membership strictly greater than zero. If this is not the case then there will be a crisp, input value of error which cannot be matched during the fuzzification phase, to a rule_antecedent. Thus, none of the rules will fire and consequently, no value for the control output will be computed. This in turn leads to discontinuities in the control output. Furthermore, if the crosspoint ratio between every two membership functions is zero then only one rule at a time can fire. This situation is depicted in figure 5.4

If each two adjacent membership function have a higher crosspoint level of 0.5 and a cross-point ratio of 1, then this provides for significantly less overshoot, faster rise time, and less undershoot. The shape of the membership function does not play a significant role but trapezoid functions are responsible for a slower rise time. Though these results have an empirical character, the choice of the cross-point level of 0.5 and a crosspoint ratio of 1 is the usual choice.

2.5 COMPARISON OF PROPORTIONAL AND DERIVATIVE CONTROLLER

Each of the modes of control is applicable to process having certain characteristics and the importance of designing the controlled system must not be overlooked. If all process consisted only one capacitance without dead time there will be no necessity for any complex control actions. The difficulty is that few, if any, industrial process are so simple in dynamic structure.

The following comments applied to each type of control

1. Proportional derivative control provides the smallest maximum error because the derivative part of the response allows the proportional sensitivity to be increased to a high value. The stabilisation time is the smallest because of the derivative action. The offset is allowed but is only half that experienced without derivative action.
2. Proportional-Integral-derivative control has the next smallest maximum derivation and offset is eliminated because of the integral action. However, that the addition of the integral action markedly increases the stabilisation time.
3. Proportional control has the largest maximum deviation than controllers with derivative action because of the absence of this stabilising influence. Offset is also larger.

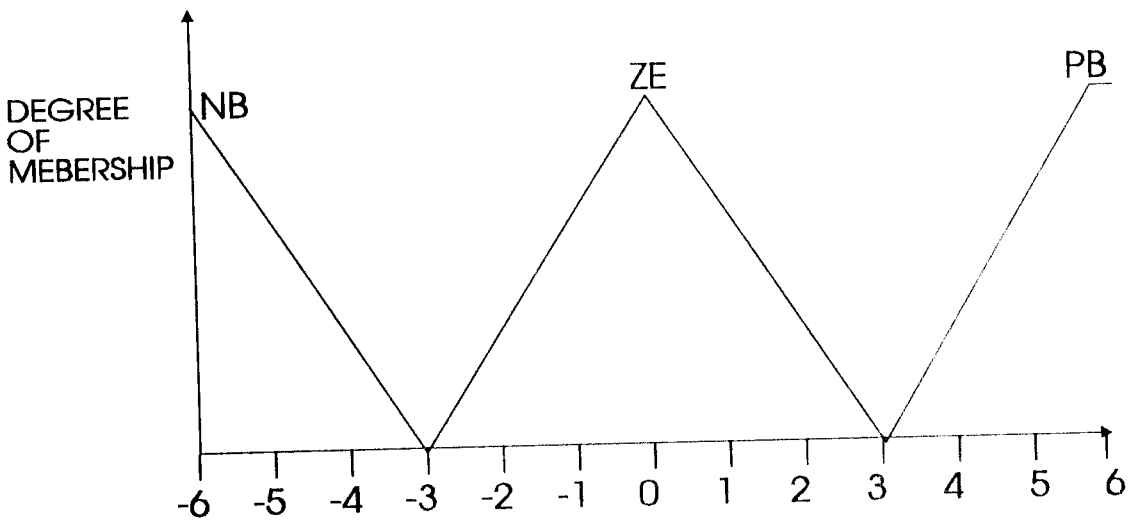


FIG. 5.4
MEMBERSHIP FUNCTION
WITH NO CROSS POINT

SUMMARY

This chapter explains the development of fuzzy algorithm and its design implementation and discuss how a rule base is designed for wind energy conversion scheme.

SIMULATION

Simulation is the representation of a real life system by another system, which depicts the important characteristic of a real system and allows experimentation on it. In other words simulation is an imitation of reality.

The technique of simulation has become an important role of decision making. Its popularity is increasing continuously with the increase in popularity of computers.

This chapter deals with the procedure for the simulation process used for the fuzzy logic controller for wind energy conversion scheme.

PROCEDURE

The output of the controller is the modulation index. For the simulation the modulation index should be related to the voltage delivered to the load.

$$\textit{Output voltage } V_o = [2/(2*\pi)\int(Vr*Vr\, d(wt))]^{1/2} = Vr * (delta/\pi)$$

Where 'delta' is the pulse width.

By varying V_r from '0' to 'V_L', the pulse width, 'delta' varies from '0' to '180' degrees. The ratio of V_r to V_L is the control variable and is defined as the amplitude modulation index.

If M_o is the old modulation index, then the new modulation index can be calculated by,

$$M_l = M_o + \Delta M$$

Where ^m is the change in the modulation index which is the defuzzified output. This is equivalent to,

$$V_{nl} = V_{nol} + \Delta V_l$$

Where V_{nl} is the new output voltage, V_{nol} is the old output voltage, ΔV_l is the change in the output voltage.

This new output voltage is given to the FLC for the error and change in error calculation.

The error value is constantly monitored for a present value and the output is made to settle at the reference level. The plotting has been done by taking time along the x-axis and output value along the y-axis. Error<vs>time and Changeinerror<vs>time are also plotted.

The software for the above simulation process has been presented in the appendix with the software listing of the FLC. The graphs of the simulation are also presented with it.

SUMMARY

In this chapter, the simulation method which was followed in the simulation process of the FLC for Wind energy conversion process has been presented.

CONCLUSION

The software realization of the fuzzy logic controller for wind energy conversion scheme has been performed using C++.

An alternative controller based on fuzzy set theory can thus be used effectively to extract maximum energy from the wind energy system.

Unlike it's counter part, the PID controller, no detailed mathematical model is required by the FLC. Also, each rule addresses a wider scenario of operating conditions and ensures change in temperature, variation in system parameter with ageing or with change in operating conditions.

Having tested the performance of the controller by simulations, the results show that good dynamic response characteristics are obtained over the other conventional controllers.

BIBLIOGRAPHY

1. "FUZZY SYSTEM HANDBOOK" by *Earl Cox*

2. "FUZZY THINKING" by *Bart Kosko*

3. "EXPERT SYSTEMS AND FUZZY SYSTEMS" by *Constantin Virgil Negoita*

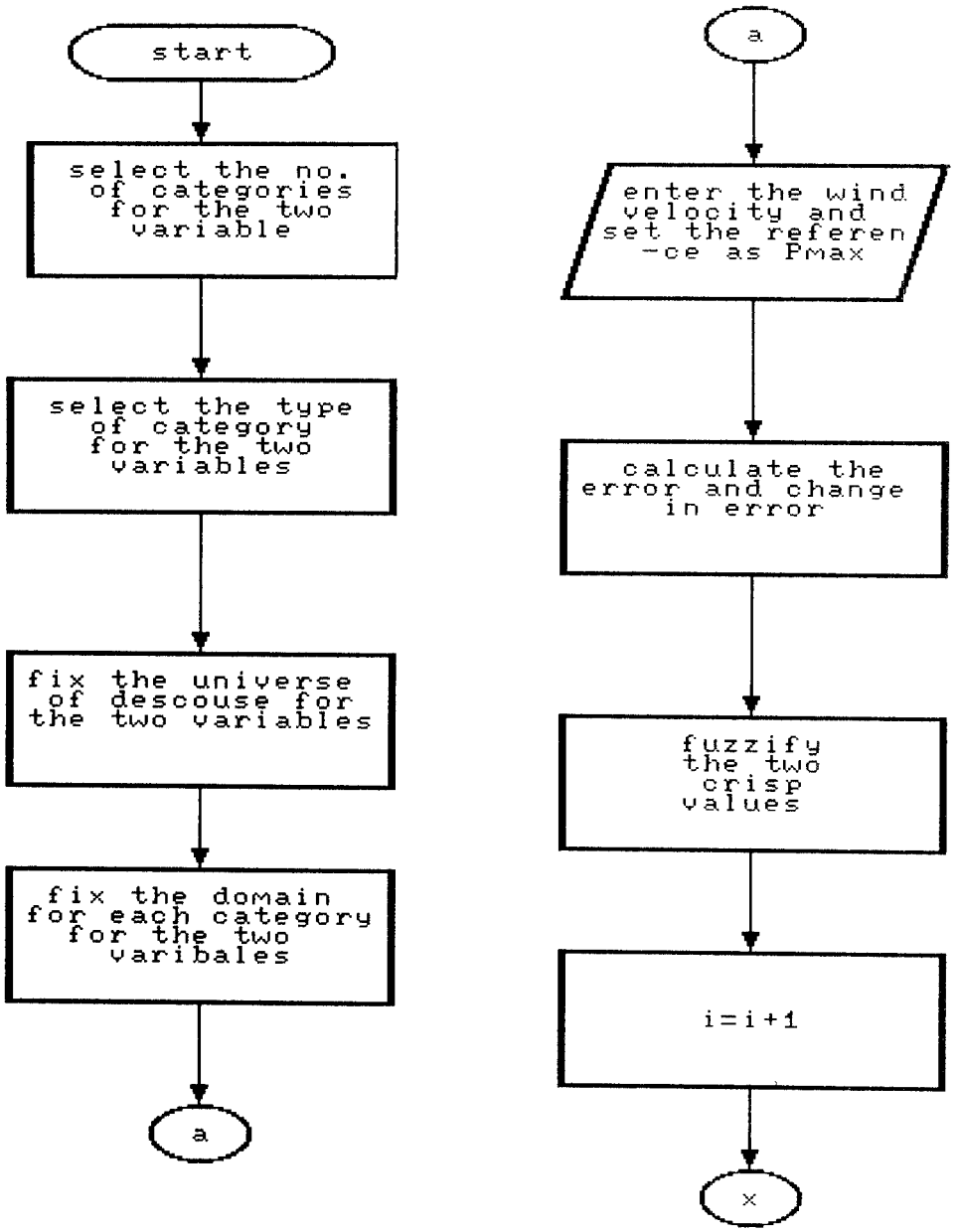
4. " INTRODUCTION TO FUZZY CONTROL" by *D.Driankov, H.Hellndoorn, M.Reinfrank*

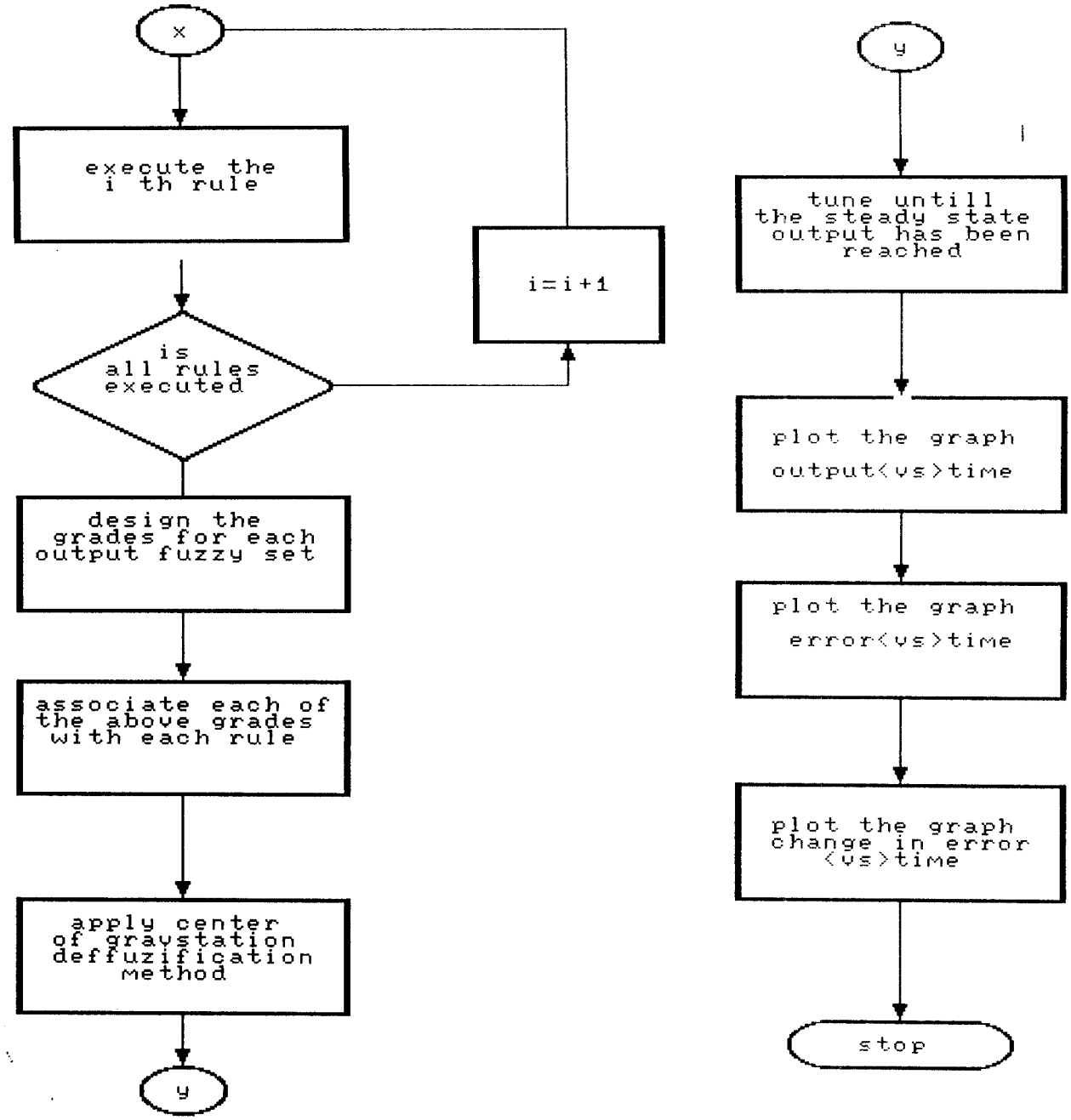
5. "FUZZY LOGIC IN CONTROL SYSTEMS FUZZY LOGIC CONTROLLER", PART I, IEEE Transactions on Systems,Man and Cybernetics, Vol.20 by *C.C.Lee*,

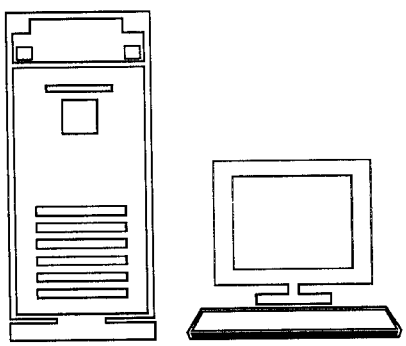
6. "FUZZY LOGIC IN CONTROL SYSTEMS: FUZZY LOGIC CONTROLLER, PART II", IEEE Transactions on Systems,Mman and Cybernetics, Vol.20 by *C.C.Lee*

7. "MODERN CONTROL ENGINEERING", PHI, 1987 by *Katsuhiko Ogata*

8. "ELECTRICAL TECHNOLOGY", Vol.2, S.Chand & Co., 1984. by *B.L.Thera*


start
select the no. of categories for the two variable
select the type of category for the two variables
fix the universe of descouse for the two variables
fix the domain for each category for the two varibales
a
a
enter the wind velocity and set the referen -ce as Pmax
calculate the error and change in error
fuzzify the two crisp values
i=i+1
x


x
execute the i th rule
is all rules executed
i=i+1
design the grades for each output fuzzy set
associate each of the above grades with each rule
apply center of gravstation deffuzification method
y
y
tune untill the steady state output has been reached
plot the graph output<vs>time
plot the graph error<vs>time
plot the graph change in error <vs>time
stop



Source Code

```
#include <math.h>
#include <string.h>
#include <iostream.h>
#include "plot.cpp"

#define mean 0
#define product 1
#define  bounded 2
#define yager 3
#define zadeh 4

#define LN -.5
#define MN -0.45
#define SN -0.46
#define ZE 0
#define SP 0.45

#define MP 0.46
#define LP .5

class category
        {
       private:

         char name[30];
      float x,y,z,left,beta,right,low,high,l,ml,m,mh,h;

    public:
                category() {};
                void setname(char *);
                char * getname();
                float sshare( float,float,float,float);
                float tshare( float,float,float,float,float );
                float lshare( float,float,float);
                ~category(){};

         };


void category :: setname(char* n)
{
       strcpy(name,n);
 }
```

```
char * category :: getname()
{
 return name;
}
float category :: sshare( float input,float left,
                          float beta,float right)
{
 float output,temp1;
 if (input >= right ) output =1.00;

 else if(input>beta)
{temp1=(input - right)/(right-left);
  output= 1 - (2 * (pow(temp1,2)));
        }
  else if(input >left)
   {
          temp1=(input -left)/(right-left);
          output=2*(pow(temp1,2));
          }
                else output=0.0;


         return output;
 } //end of sshare

 float category :: tshare( float  input,float x,
                      float y,float z,float h )

 { float output;

 if((input<=x) || (input >=h)) output=0.0;
  if ((input > x) && (input <y)) output =(input -x)/(y-x);
  if((input >z) && (input <h)) output =(h-input)/(h-z);
  if((input >=y) && (input<=z))output=1.00;
        return output;

  }//end of tshare

 float category :: lshare (float input,float low,float high)
  {
         float output =0.0;
         if (input >high) output =1.0;
         else
         if(input > low) output =(input-low)/(high-low);
```

```
        return output;
 }


float memberfun(double inval,char curve, double l,
                double ml,double mh,double h, double m,char *name )
{
 int i=0;
 double membership;

  category categobj;
 categobj.setname(name);


//close the function

 switch(curve)
        {
   /*   case 's' :
                        membership=categobj.sshare(inval,l,m,h);
                                                break;

        */      case 't' :
                        membership=categobj.tshare(inval,l,ml,mh,h);
                        break;

                case 'n' :
                         ml=mh;
                        membership=categobj.tshare(inval,l,ml,mh,h);
                        break;
                case 'l' :
                        membership=categobj.lshare(inval,l,h);
                        break;
                case 'd':
                                m=categobj.lshare(inval,l,h);
                                membership=1-m;
                                break;
               default: cout << "irrelevant input"<<endl;

                }//end switch

        return(membership);

        }//end of function loop
```

```
double min(double a,double b)
{
return((a>=b)? b:a);
}
double max(double a,double b)
{
return((a>=b)? a:b);
}


double minimum(double x,double y,int type,double coeff)

{

 double temp;

 switch(type)
 {

case mean:
                        {       temp=(x+y)/2;
                       if (coeff!=0&&coeff<2) temp=temp*coeff;
                       return (temp);
                        }
case product:
                                { temp=x*y;
                         if (coeff!=0&&coeff<2) temp=temp*coeff;
                       return(temp);
                                }

                        case bounded:

                        { temp=max(0,(x+y)-1);
                  if (coeff!=0&&coeff<2) temp=temp*coeff;
                return(temp);
                  }

        case yager:

                        {
                                double exp=(1/coeff);
                                double x1=pow((1-x),coeff);
                                double y1=pow((1-y),coeff);
                                temp=1-min(1.0,pow(x1+y1,exp));
```

```
                        return (temp);
                    }

        case zadeh:
                    {
                        return((double)min(x,y));
                     }

        default :  cout <<"error has occured"<<endl;

 }
}


main()
{ clrscr();

 int i,j,r=1,v;
 double w[50],rule[50],out[100],x[100],delout[100];
 int q=0; x[0]=0.0;
 int values=100;
 double error[100],temp1,temp2,changeinerror[100],mem1[10],mem2[10];
 double l=0.0,m =0.0,ml=0.0,mh=0.0,h=0.0;
 char *name1[7]={  "largenegative",
                   "mediumnegative",
                   "smallnegative",
                   "zero",
                   "smallpositive",
                   "mediumpositive",
                   "largepositive"
                  };

 char *name2[7]={   "largenegative",
                    "mediumnegative",
                    "smallnegative",
                    "zero",
                   "smallpositive",
                   "mediumpositive",
                    "largepositive"
                   };


         char *graphname[4]={"output <vs> time",
                             "error <vs> time",
                             "changeinerror <vs> time",
```

```
                        "changeinoutput <vs> time"
                        };
w[0]=LN;
w[1]=LN; w[2]=LN; w[3]=LN; w[4]=MN; w[5]=SN; w[6]=ZE;
w[7]=LN; w[8]=LN; w[9]=MN; w[10]=MN; w[11]=SN; w[12]=ZE; w[13]=SP;
w[14]=LN; w[15]=MN; w[16]=SN; w[17]=SN; w[18]=ZE; w[19]=SP; w[20]=MP;
w[21]=MN; w[22]=MN; w[23]=SN; w[24]=ZE; w[25]=SP; w[26]=MP; w[27]=MP;
w[28]=MN; w[29]=SN; w[30]=ZE; w[31]=SP; w[32]=SP; w[33]=MP; w[34]=LP;
w[35]=SN; w[36]=ZE; w[37]=SP; w[38]=MP; w[39]=MP; w[40]=LP; w[41]=LP;
w[42]=ZE; w[43]=SP; w[44]=MP; w[45]=LP; w[46]=LP; w[47]=LP; w[48]=LP;


        double reference=3;
        out[0]=0;
        error[0]=0;
cout<<"*******"<<"\t"<<"****"<<"\t"<<"*****"<<"\t"<<"*****************"
        <<endl;

cout <<"delout"<<"\t"<<"out"<<"\t"<<"error"<<"\t"<<"changeinerror"<<endl;

cout<<"*******"<<"\t"<<"****"<<"\t"<<"*****"<<"\t"<<"*****************"
        <<endl;

for(j=1;j<values;j++)
{

 error[j]=(reference-out[j-1])/reference;
changeinerror[j]=(error[j]-error[j-1]);

 mem1[0]=memberfun(error[j],'d', -0.6,-.7,-.68,-0.3,0,name1[0]);
 mem1[1]=memberfun(error[j],'n', -0.45,-0.2,-0.2,-0.1,0,name1[1]);
 mem1[2]=memberfun(error[j],'n', -0.25,-0.1,-0.1,0.08,0,name1[2]);
 mem1[3]=memberfun(error[j],'n', -0.08,0,0,0.08,0,name1[3]);
 mem1[4]=memberfun(error[j],'n',- 0.08,0.1,0.1,0.25,0,name1[4]);
 mem1[5]=memberfun(error[j],'n', 0.1,0.2,0.2,0.45,0,name1[5]);
 mem1[6]=memberfun(error[j],'l', 0.3,0.5,7,.6,0,name1[6]);

 mem2[0]=memberfun(changeinerror[j],'d', -0.6,-.7,-.68,-0.3,0,name2[0]);
 mem2[1]=memberfun(changeinerror[j],'n', -0.45,-0.2,-0.2,-0.1,0,name2[1]);
 mem2[2]=memberfun(changeinerror[j],'n', -0.25,-0.1,-0.1,0.08,0,name2[2]);
 mem2[3]=memberfun(changeinerror[j],'n', -0.08,0,0,0.08,0,name2[3]);
 mem2[4]=memberfun(changeinerror[j],'n',- 0.08,0.1,0.1,0.25,0,name2[4]);
 mem2[5]=memberfun(changeinerror[j],'n', 0.1,0.2,0.2,0.45,0,name2[5]);
 mem2[6]=memberfun(changeinerror[j],'l', 0.3,0.5,7,.6,0,name2[6]);
```

```
//rule base
        rule[0]  = minimum(mem1[0],mem2[0],4,0);
        rule[1]  = minimum(mem1[0],mem2[1],4,0);
        rule[2]  = minimum(mem1[0],mem2[2],4,0);
        rule[3]  = minimum(mem1[0],mem2[3],4,0);
        rule[4]  = minimum(mem1[0],mem2[4],4,0);
        rule[5]  = minimum(mem1[0],mem2[5],4,0);
        rule[6]  = minimum(mem1[0],mem2[6],4,0);
        rule[7]  = minimum(mem1[1],mem2[0],4,0);
        rule[8]  = minimum(mem1[1],mem2[1],4,0);
        rule[9]  = minimum(mem1[1],mem2[2],4,0);
        rule[10]= minimum(mem1[1],mem2[3],4,0);
        rule[11]= minimum(mem1[1],mem2[4],4,0);
        rule[12]= minimum(mem1[1],mem2[5],4,0);
        rule[13]= minimum(mem1[1],mem2[6],4,0);
        rule[14]= minimum(mem1[2],mem2[0],4,0);
        rule[15]= minimum(mem1[2],mem2[1],4,0);
        rule[16]= minimum(mem1[2],mem2[2],4,0);
        rule[17]= minimum(mem1[2],mem2[3],4,0);
        rule[18]= minimum(mem1[2],mem2[4],4,0);
        rule[19]= minimum(mem1[2],mem2[5],4,0);
        rule[20]= minimum(mem1[2],mem2[6],4,0);
        rule[21]= minimum(mem1[3],mem2[0],4,0);
        rule[22]= minimum(mem1[3],mem2[1],4,0);
        rule[23]= minimum(mem1[3],mem2[2],4,0);
        rule[24]= minimum(mem1[3],mem2[3],4,0);
        rule[25]= minimum(mem1[3],mem2[4],4,0);
        rule[26]= minimum(mem1[3],mem2[5],4,0);
        rule[27]= minimum(mem1[3],mem2[6],4,0);
        rule[28]= minimum(mem1[4],mem2[0],4,0);
        rule[29]= minimum(mem1[4],mem2[1],4,0);
        rule[30]= minimum(mem1[4],mem2[2],4,0);
        rule[31]= minimum(mem1[4],mem2[3],4,0);
        rule[32]= minimum(mem1[4],mem2[4],4,0);
        rule[33]= minimum(mem1[4],mem2[5],4,0);
        rule[34]= minimum(mem1[4],mem2[6],4,0);
        rule[35]= minimum(mem1[5],mem2[0],4,0);
        rule[36]= minimum(mem1[5],mem2[1],4,0);
        rule[37]= minimum(mem1[5],mem2[2],4,0);
        rule[38]= minimum(mem1[5],mem2[3],4,0);
        rule[39]= minimum(mem1[5],mem2[4],4,0);
        rule[40]= minimum(mem1[5],mem2[5],4,0);
        rule[41]= minimum(mem1[5],mem2[6],4,0);
        rule[42]= minimum(mem1[6],mem2[0],4,0);
```

```
        rule[43]= minimum(mem1[6],mem2[1],4,0);
        rule[44]= minimum(mem1[6],mem2[2],4,0);
        rule[45]= minimum(mem1[6],mem2[3],4,0);
        rule[46]= minimum(mem1[6],mem2[4],4,0);
        rule[47]= minimum(mem1[6],mem2[5],4,0);
        rule[48]= minimum(mem1[6],mem2[6],4,0);

/*      int k=0;
        for(int i=0;i<7;i++)
        for(int j=0;j<7;j++)
        {
        rule[k]= minimum(mem1[i],mem2[j],4,0);
        k++;
        }

*/

//APPLY DEFUZZIFICATION

for(i=0;i<49;i++)
{
temp1+=rule[i]*w[i];
temp2+=rule[i];
}

 delout[j]=temp1/temp2;
out[j]=out[j-1]+delout[j];

cout <<"("<<j<<")"<<"\t"<<delout[j];
cout<<"\t"<<out[j]<<"\t"<<error[j]<<"\t"<<changeinerror[j]<<endl;

if(j>30&&r==1){getch();r=j; v=1;}
if(j>60&&v==1){getch();v=j;}
if(r>90)getch();

}//for the for loop

 for( q=0;q<values;q++)
{ x[q]=x[q-1]+0.1;
}
cout <<"****************************************"<<endl;
cout <<"press enter to show the output response"<<endl;
cout <<"****************************************"<<endl;
getch();
```

```
plot(x,out,values,graphname[0]);
cout <<"************************************"<<endl;
cout <<"press enter to show the  error graph"<<endl;
cout <<"************************************"<<endl;
getch();
plot(x,error,values,graphname[1]);
cout <<"********************************************"<<endl;
cout <<"press enter to show the  changeinerror graph"<<endl;
cout <<"********************************************"<<endl;
getch();
plot(x,changeinerror,values,graphname[2]);
getch();
cout <<"*********************************************"<<endl;
cout <<"press enter to show the  change in output graph"<<endl;
cout <<"*********************************************"<<endl;
plot(x,delout,values,graphname[3]);
getch();

}
```

```
//plot
void plot(double *point1,double *point2,int values,char * graphname)
{
 double x1,x2,y1,y2,xmax,ymax,scalx,scaly,setpt;
 int gd=DETECT,gm,i=0;
 double d,e,r;
  setpt=0.8;
 initgraph(&gd,&gm,"");
 xmax=getmaxx()-100;
 ymax=getmaxy()-100;
 scalx=xmax/10;
 d=scalx;
 line(50,ymax+75,xmax+50,ymax+75);
 for(scalx=0;scalx<=xmax;scalx+=d)
 line(scalx+50,77+ymax,50+scalx,75+ymax);
 outtextxy(38,ymax+80,"0.0");
 outtextxy(1*d+38,ymax+80,"1.0");
 outtextxy(2*d+38,ymax+80,"2.0");
 outtextxy(3*d+38,ymax+80,"3.0");
 outtextxy(4*d+38,ymax+80,"4.0");
 outtextxy(5*d+38,ymax+80,"5.0");
 outtextxy(6*d+38,ymax+80,"6.0");
 outtextxy(7*d+38,ymax+80,"7.0");
 outtextxy(8*d+38,ymax+80,"8.0");
 outtextxy(9*d+38,ymax+80,"9.0");
 outtextxy(10*d+38,ymax+80,"10.0");
 outtextxy(xmax/2,getmaxy()-10,"TIME IN SECONDS");

scaly=ymax/(setpt*2.0);
line(50,ymax+75,50,10);
e=ymax/5.0;
r=e;
for(e=0;e<=ymax+60;e+=r)
line(48,ymax+75-e,52,75-e+ymax);
line(50,ymax+75-3*r,xmax+50,ymax+75-3*r);
outtextxy(point1[0]*d+50,ymax+75-point2[0]*r,"");
for(i=1;i<values;i++)
{lineto(point1[i]*d+50,ymax+75-point2[i]*r);}
outtextxy(20,ymax+75-0*r,"0");
outtextxy(20,ymax+75-1*r,"1");
outtextxy(20,ymax+75-2*r,"2");
outtextxy(20,ymax+75-3*r,"3");
outtextxy(20,ymax+75-4*r,"4");
outtextxy(20,ymax+75-5*r,"5");
```

```
outtextxy(20,ymax+75-6*r,"6");
outtextxy(20,ymax+75-7*r,"7");
outtextxy(4,ymax/2+4,"O");
outtextxy(4,ymax/2+14,"U");
outtextxy(4,ymax/2+24,"T");
outtextxy(4,ymax/2+34,"P");
outtextxy(4,ymax/2+44,"U");
outtextxy(4,ymax/2+54,"T");
outtextxy(4,ymax/2+70,"V");
outtextxy(4,ymax/2+78,"A");
outtextxy(4,ymax/2+86,"L");
outtextxy(4,ymax/2+94,"U");
outtextxy(4,ymax/2+102,"E");
settextstyle(4,0,4);
outtextxy(150,50,graphname);

}
```

```
#include <iostream.h>
#include <stdio.h>
#include <io.h>
#include <bios.h>
#include <stdlib.h>
#include <stdarg.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>
#include <dos.h>
#include <process.h>
 void snd()
          {
          sound(1000);
          delay(100);
          sound(6000);
          delay(100);
          nosound();
          delay(500);
          }
void plot()

{
 float px1,px2,px3,x1,x2,x3,xmax,ymax,scalx,scaly,setpt;
 float x11,x22,x33;
 px1=-7;
 px2=-6;
 px3=-5;
 x1=8+px1;
 x2=8+px2;
 x3=8+px3;

 int gd=DETECT,gm,i=0;
 float t=0.0,d,e,r;
  setpt=0.8;
 initgraph(&gd,&gm,"c:\\bc4\\bgi");
 xmax=getmaxx()-100;
 ymax=getmaxy()-100;
 scalx=xmax/16;
  outtextxy(200,10/*xmax/2-50,getmaxy()-12*r,*/," RULE EXECUTION");
 d=scalx;
  scaly=ymax/(setpt*2.0);
 e=ymax/20;
 r=e;
```

```
line(50,ymax+75,7*d+50,ymax+75);
line(50,ymax+72-12*r,7*d+50,ymax+72-12*r);
line(50+10*d,ymax+72-12*r,17*d+50,ymax+72-12*r);
 for(scalx=0;scalx<6*d+38;scalx+=d)
 line(scalx+50,78+ymax,50+scalx,75+ymax);
 for(scalx=0;scalx<6*d+38;scalx+=d)
 line(scalx+50,ymax+72-12*r,50+scalx,ymax+75-12*r);
 for(scalx=0;scalx<6*d+38;scalx+=d)
 line(scalx+50+10*d,ymax+72-12*r,scalx+50+10*d,ymax+75-12*r);

 outtextxy(48,ymax+80,"0");
 outtextxy(1*d+48,ymax+80,"1");
 outtextxy(2*d+48,ymax+80,"2");
 outtextxy(3*d+48,ymax+80,"3");
 outtextxy(4*d+48,ymax+80,"4");
 outtextxy(5*d+48,ymax+80,"5");
 outtextxy(6*d+48,ymax+80,"6");
 outtextxy(7*d+48,ymax+80,"7");
 outtextxy(xmax/2-50,getmaxy()-12*r,"--->VARIABLE 1");

 outtextxy(48,ymax+78-12*r,"0");
 outtextxy(1*d+48,ymax+78-12*r,"1");
 outtextxy(2*d+48,ymax+78-12*r,"2");
 outtextxy(3*d+48,ymax+78-12*r,"3");
 outtextxy(4*d+48,ymax+78-12*r,"4");
 outtextxy(5*d+48,ymax+78-12*r,"5");
 outtextxy(6*d+48,ymax+78-12*r,"6");
 outtextxy(7*d+48,ymax+78-12*r,"7");
 outtextxy(xmax/2-50,getmaxy()-10,"--->VARIABLE 2");

 outtextxy(48+10*d,ymax+78-12*r,"0");
 outtextxy(11*d+48,ymax+78-12*r,"1");
 outtextxy(12*d+48,ymax+78-12*r,"2");
 outtextxy(13*d+48,ymax+78-12*r,"3");
 outtextxy(14*d+48,ymax+78-12*r,"4");
 outtextxy(15*d+48,ymax+78-12*r,"5");
 outtextxy(16*d+48,ymax+78-12*r,"6");
 outtextxy(17*d+48,ymax+78-12*r,"7");
 outtextxy(xmax/2-50+9*d,getmaxy()-12*r,"---> OUTPUT ");

line(50,ymax+75,50,ymax+75-10*r);
line(50,ymax+72-12*r,50,30);
line(50+10*d,ymax+72-12*r,50+10*d,30);
```

```
for(e=0;e<=ymax+75-13*r;e+=r)
line(46,ymax+75-e,50,ymax+75-e);
for(e=ymax+75-11*r;e<=ymax+75-2*r;e+=r)
line(46,ymax+75-e,50,ymax+75-e);
for(e=ymax+75-11*r;e<=ymax+75-2*r;e+=r)
line(46+10*d,ymax+75-e,50+10*d,ymax+75-e);


outtextxy(20,ymax+72-1*r,"0.1");
outtextxy(20,ymax+72-2*r,"0.2");
outtextxy(20,ymax+72-3*r,"0.3");
outtextxy(20,ymax+72-4*r,"0.4");
outtextxy(20,ymax+72-5*r,"0.5");
outtextxy(20,ymax+72-6*r,"0.6");
outtextxy(20,ymax+72-7*r,"0.7");
outtextxy(20,ymax+72-8*r,"0.8");
outtextxy(20,ymax+72-9*r,"0.9");
outtextxy(20,ymax+72-10*r,"1.0");

outtextxy(20,ymax+72-13*r,"0.1");
outtextxy(20,ymax+72-14*r,"0.2");
outtextxy(20,ymax+72-15*r,"0.3");
outtextxy(20,ymax+72-16*r,"0.4");
outtextxy(20,ymax+72-17*r,"0.5");
outtextxy(20,ymax+72-18*r,"0.6");
outtextxy(20,ymax+72-19*r,"0.7");
outtextxy(20,ymax+72-20*r,"0.8");
outtextxy(20,ymax+72-21*r,"0.9");
outtextxy(20,ymax+72-22*r,"1.0");

outtextxy(20+10*d,ymax+72-13*r,"0.1");
outtextxy(20+10*d,ymax+72-14*r,"0.2");
outtextxy(20+10*d,ymax+72-15*r,"0.3");
outtextxy(20+10*d,ymax+72-16*r,"0.4");
outtextxy(20+10*d,ymax+72-17*r,"0.5");
outtextxy(20+10*d,ymax+72-18*r,"0.6");
outtextxy(20+10*d,ymax+72-19*r,"0.7");
outtextxy(20+10*d,ymax+72-20*r,"0.8");
outtextxy(20+10*d,ymax+72-21*r,"0.9");
outtextxy(20+10*d,ymax+72-22*r,"1.0");
line(x1*d+50,ymax+80,x2*d+50,ymax+75-10*r);
line(x2*d+50,ymax+75-10*r,x3*d+50,ymax+80);
line(x1*d+50,ymax+72-12*r,x2*d+50,ymax+75-22*r);
line(x2*d+50,ymax+75-22*r,x3*d+50,ymax+80-12*r);
```

```
line(50+11.5*d,ymax+76-17*r,50+12.5*d,ymax+76-17*r);
line(x1*d+50+10*d,ymax+72-12*r,50+11.5*d,ymax+76-17*r);
line(50+12.5*d,ymax+76-17*r,x3*d+50+10*d,ymax+80-12*r);
setfillstyle(8,15);
floodfill(50+11.5*d+10,ymax+76-17*r+10,getmaxcolor());

float inval1=1.8,inval2=1.4;
 for(int j=ymax+70;j>ymax+75-10*r;j-=20)
{
outtextxy(inval1*d+50,j,"|");//input line
 snd();
 }
 delay(2000);
  for(  j=ymax+75-12.5*r;j>ymax+75-22.5*r;j-=20)
{
outtextxy(inval2*d+50,j,"|");//input line 2
 snd();
 }

 for(i=20;i<20+8*d;i+=30)
 {
outtextxy(i,ymax+72-9*r,"-");//fin  line
snd();
}
for(i=ymax+72-17*r;i<ymax+72-8*r;i+=30)
{outtextxy(7*d+48,i,"|");
snd();//vertical line
}

for(i=20;i<16*d+48;i+=30)
{
outtextxy(i,ymax+72-17*r,"-");//.5line
snd();
}
}
void main()
{
 plot();

 getch();
 }
```

```
#include <iostream.h>
#include <stdio.h>
#include <io.h>
#include <bios.h>
#include <stdlib.h>
#include <stdarg.h>
#include <conio.h>
#include <math.h>
#include <graphics.h>
#include <dos.h>
#include <process.h>
void snd()
        {
        sound(1000);
        delay(100);
        sound(6000);
        delay(100);
        nosound();
        delay(500);
        }


void plot(float *px,float inval)
{
 inval+=8;
 float x[30],xmax,ymax,scalx,scaly,setpt;
 for(int j=0;j<28;j++)
 x[j]=8+px[j];

 int gd=DETECT,gm,i=0;
 float t=0.0,d,e,r;
 static struct
 {int ovval;}ov[50];
  setpt=0.8;
 initgraph(&gd,&gm,"c:\\bc4\\bgi");
 xmax=getmaxx()-100;
 ymax=getmaxy()-100;
 scalx=xmax/16;
 d=scalx;
line(50,ymax+75,xmax+50,ymax+75);//x axis
 for(scalx=0;scalx<=xmax;scalx+=d)
 line(scalx+50,77+ymax,50+scalx,75+ymax);
 outtextxy(38,ymax+80,"-8");
 outtextxy(1*d+38,ymax+80,"-7");
```

```
 outtextxy(2*d+38,ymax+80,"-6");
 outtextxy(3*d+38,ymax+80,"-5");
 outtextxy(4*d+38,ymax+80,"-4");
 outtextxy(5*d+38,ymax+80,"-3");
 outtextxy(6*d+38,ymax+80,"-2");
 outtextxy(7*d+38,ymax+80,"-1");
 outtextxy(8*d+38,ymax+80,"0");
 outtextxy(9*d+38,ymax+80,"1");
 outtextxy(10*d+38,ymax+80,"2");
 outtextxy(11*d+38,ymax+80,"3");
 outtextxy(12*d+38,ymax+80,"4");
 outtextxy(13*d+38,ymax+80,"5");
 outtextxy(14*d+38,ymax+80,"6");
 outtextxy(15*d+38,ymax+80,"7");
 outtextxy(16*d+38,ymax+80,"8");

outtextxy(xmax/2,getmaxy()-10,"I variable");
 scaly=ymax/(setpt*2.0);
e=ymax/15.0;
r=e;
line(8*d+50,ymax+75,8*d+50,ymax+75-12*r);
for(j=0;j<28;j+=4)
{line(x[j]*d+50,ymax+80,x[j+1]*d+50,ymax+75-10*r);
 line(x[j+1]*d+50,ymax+75-10*r,x[j+2]*d+50,ymax+75-10*r);
line(x[j+2]*d+50,ymax+75-10*r,x[j+3]*d+50,ymax+80);
 }
//for(int k=0;k<10;k++)

for(j=ymax+80;j>100;j-=20)
{
outtextxy(inval*d+50,j,"|");//input line
 snd();

}
for(e=0;e<=ymax+75-5*r;e+=r)
line(8*d+46,ymax+75-e,8*d+50,75-e+ymax); //small line

outtextxy(8*d+20,ymax+75-0*r,"");
outtextxy(8*d+20,ymax+75-1*r,"0.1");
outtextxy(8*d+20,ymax+75-2*r,"0.2");
outtextxy(8*d+20,ymax+75-3*r,"0.3");
outtextxy(8*d+20,ymax+75-4*r,"0.4");
outtextxy(8*d+20,ymax+75-5*r,"0.5");
outtextxy(8*d+20,ymax+75-6*r,"0.6");
```

```
outtextxy(8*d+20,ymax+75-7*r,"0.7");
outtextxy(8*d+20,ymax+75-8*r,"0.8");
outtextxy(8*d+20,ymax+75-9*r,"0.9");
outtextxy(8*d+20,ymax+75-10*r,"1.0");
 outtextxy(2*d+60,ymax+75-8*r,"LN");
  outtextxy(4*d+60,ymax+75-8*r,"MN");

        outtextxy(6*d+60,ymax+75-8*r,"SN");
         outtextxy(8*d+60,ymax+75-8*r,"ZE");
          outtextxy(10*d+60,ymax+75-8*r,"SP");
                outtextxy(12*d+60,ymax+75-8*r,"MP");
                 outtextxy(14*d+60,ymax+75-8*r,"LP");

outtextxy(4,ymax/2+4,"M");
outtextxy(4,ymax/2+14,"E");
outtextxy(4,ymax/2+24,"M");
outtextxy(4,ymax/2+34,"B");
outtextxy(4,ymax/2+44,"E");
outtextxy(4,ymax/2+54,"R");
outtextxy(4,ymax/2+70,"S");
outtextxy(4,ymax/2+78,"H");
outtextxy(4,ymax/2+86,"I");
outtextxy(4,ymax/2+94,"P");

}


void main()
{
 float px[]={-7,-6,-5,-4,
                                  -5,-4,-3,-2,
                                  -3,-2,-1,0,
                                   -1,0,1,2,
                                   1,2,3,4,
                                   3,4,5,6,
                                   5,6,7,8  };

float inval=-4;
 plot(px,inval);
 getch();
 }
```

```
#include <graphics.h>
#include <iostream.h>
#include <math.h>
#include <conio.h>

 main()

{
int i=0,j;
 char *arrptrs[56]={ "LN  :","LN","LN","LN","LN","MN","SN","ZE",
                     "MN  :","LN","LN","LN","MN","SN","ZE","SP",
                     "SN  :","LN","LN","MN","SN","ZE","SP","MP",
                     "ZE  :","LN","MN","SN","ZE","SP","MP","LP",
                     "SP  :","MN","SN","ZE","SP","MP","LP","LP",
                     "MP  :","SN","ZE","SP","MP","LP","LP","LP",
                     "LP  :","ZE","SP","MP","LP","LP","LP","LP",
                     };
 cout <<"**********************"<<endl;
 cout <<"THIS IS THE FAM MATRIX"<<endl;
 cout <<"**********************"<<endl<<endl;
 cout <<"    : LN\tMN\tSN\tZE\tSP\tMP\tLP"<<endl;
 cout <<"-------------------------------------------------------"<<endl;
do{
        j=0;

        do{

                cout <<arrptrs[i]<<"\t";
                j++;
                i++;
          }while(j<8);
  cout <<"\n";
  }while(i<56);
getch();

 }
```

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <process.h>
#include "title2.cpp"
#include "title3.cpp"
main()
 { main2();
        main1();
        int gd=DETECT,gm,i,j;
        initgraph(&gd,&gm,"");
        for(i=500;i>100;i-=50)
                {
        cleardevice();
        settextstyle(4,0,6);
        outtextxy(50,50,"idea and realization");
        rectangle(i,410,i+325,460);
        settextstyle(3,0,5);
        outtextxy(i+10,410,"c.chandrasekar.");
        sound(250);
        delay(25);
        sound(460);
        delay(25);
        nosound();
                }
        for(i=500;i>100;i-=100)
                {
        cleardevice();
                settextstyle(4,0,6);
        outtextxy(50,50,"idea and realization");
                settextstyle(3,0,5);
        rectangle(100,350,425,400);
        outtextxy(110,350,"C.Chandrasekar.");
        rectangle(i-10,410,i+315,460);
        outtextxy(i,410,"N.Ravikumar");
        sound(250);delay(25);
        sound(460);delay(25);
        nosound();
                }
        for(i=500;i>100;i-=100)
                {
        cleardevice();
```

```
                    settextstyle(4,0,6);
outtextxy(50,50,"idea and realization");
          settextstyle(3,0,5);
 rectangle(100,290,425,340);
 outtextxy(110,290,"C.Chandrasekar.");
 rectangle(100,350,425,400);
 outtextxy(110,350,"N.Ravikumar");
 rectangle(i-10,410,i+315,460);
 outtextxy(i,410,"T.Sivakumar");
 sound(250);delay(25);
 sound(460);delay(25);
 nosound();
         }
         for(i=500;i>100;i-=100)
{
 cleardevice();
                    settextstyle(4,0,6);
outtextxy(50,50,"idea and realization");
          settextstyle(3,0,5);
  rectangle(100,230,425,280);
  outtextxy(110,230,"C.Chandrasekar.");
  rectangle(100,290,425,340);
  outtextxy(110,290,"N.Ravikumar.");
  rectangle(100,350,425,400);
  outtextxy(110,350,"T.Sivakumar");
  rectangle(i-10,410,i+315,460);
  outtextxy(i,410,"N.Venkatesh.");
  sound(250);delay(25);
  sound(460);delay(25);
  nosound();
}
  for(i=500;i>100;i-=100)
          {
 cleardevice();
                    settextstyle(4,0,6);
outtextxy(50,50,"idea and realization");
          settextstyle(3,0,5);
 rectangle(100,170,425,220);
 outtextxy(110,170,"C.Chandrasekar.");
 rectangle(100,230,425,280);
 outtextxy(110,230,"N.Ravikumar.");
 rectangle(100,290,425,340);
 outtextxy(110,290,"T.Sivakumar.");
 rectangle(100,350,425,400);
```

```
        outtextxy(110,350,"N.Venkatesh.");
        sound(250);delay(25);
        sound(460);delay(25);
        nosound();
                }
        getch();
        closegraph();
}
```

```
void snd(void);
void kumb3(void);
void main1()
{
int i,j,k;
int gd=DETECT,gm;
fflush(stdin);
initgraph(&gd,&gm,"");
cleardevice();
settextstyle(4,0,10);
outtextxy(100,100,"Opening");
outtextxy(250,200,"up");
delay(3000);
for(i=1;i<15;i++)
{
sound((i*40)+300);
cleardevice();
settextstyle(1,0,i);
outtextxy(145,150,"F");
delay(10);
}
nosound();
for(i=1;i<15;i++)
{
sound((i*40)+300);
settextstyle(1,0,15);
outtextxy(145,150,"F");
cleardevice();
settextstyle(1,0,i);
outtextxy(230,150,"L");
delay(10);
}
nosound();
for(i=1;i<15;i++)
{
cleardevice();
sound((i*40)+300);
settextstyle(1,0,15);
outtextxy(145,150,"F");
outtextxy(230,150,"L");

settextstyle(1,0,i);
outtextxy(300,150,"C");
delay(10);
```

```
nosound();
nosound();
delay(100);
sound(300);
delay(400);
sound(1200);
delay(430);
sound(900);
delay(400);
sound(500);
delay(400);
sound(300);
delay(500);
nosound();
delay(100);
kumb3();
//getch();
}
void kumb3(void)
 {
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"c:\\BC4\\bgi");
        settextstyle(3,0,4);
        outtextxy(120,2,"F");
        snd();
        outtextxy(180,100,"L");
        snd();
                outtextxy(230,200,"C");
        snd();
        outtextxy(173,30,"UZZY");
        snd();
        outtextxy(230,128,"OGIC");
        snd();
        outtextxy(285,230,"ONTROLLER");
        snd();
        outtextxy(300,300,"GUIDED BY");
        snd();
        outtextxy(300,325,"**********");
        outtextxy(320,350,"PROF.RAMPRAKASH");
        snd();
        outtextxy(320,380,"MRS.DEVI");
        snd();
        sound(200);
        delay(200);
```

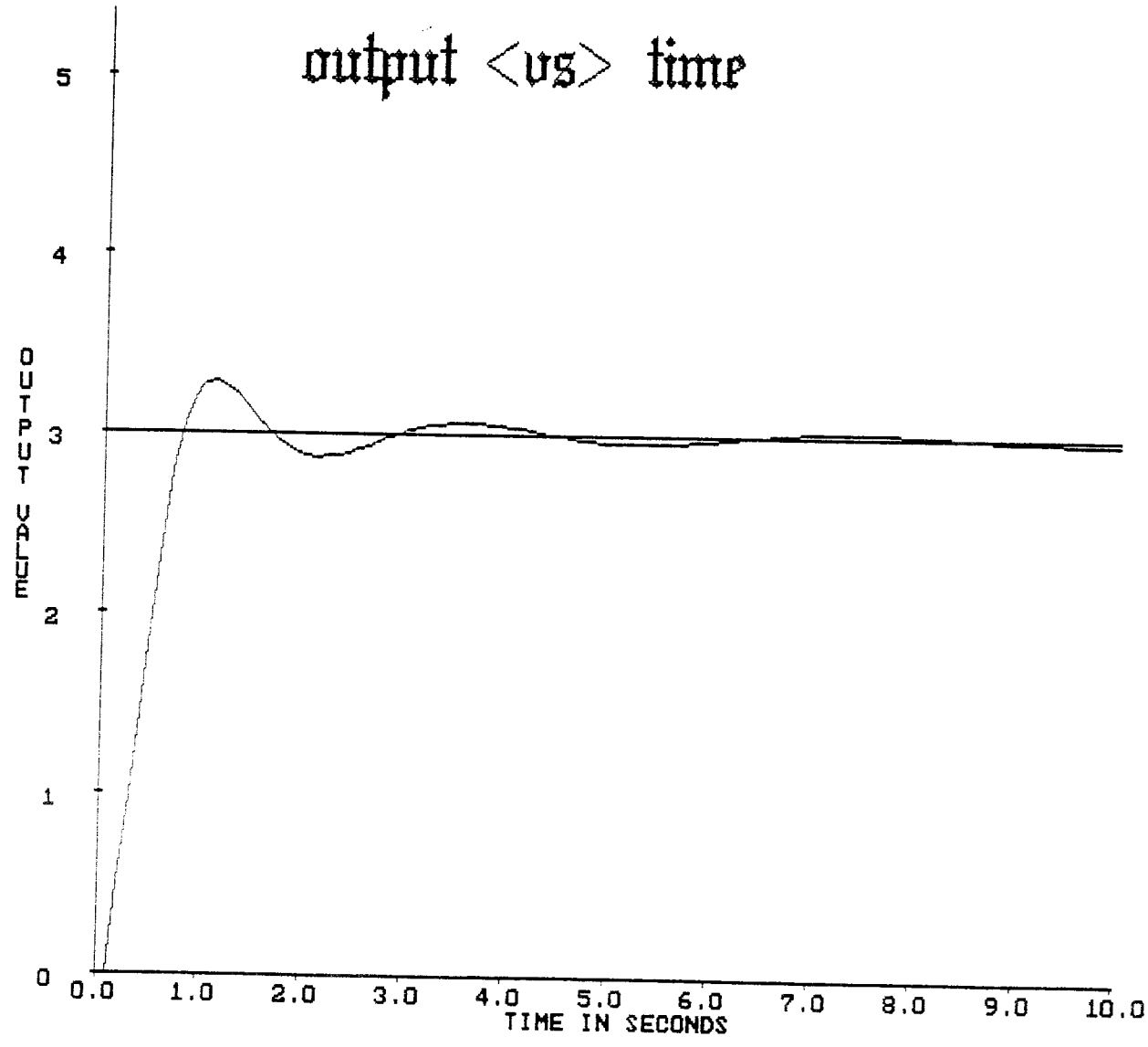
```
        sound(400);
        delay(200);
        sound(200);
        delay(200);
        sound(600);
        delay(200);
        sound(200);
        delay(200);
        sound(800);
        delay(200);
        sound(200);
        delay(200);
        sound(1000);
        delay(1000);
        sound(200);
        delay(200);
        nosound();
//      getch();
        closegraph();
        }

void snd(void)
        {
        sound(1000);
        delay(100);
        sound(6000);
        delay(100);
        nosound();
        delay(500);
        }
```

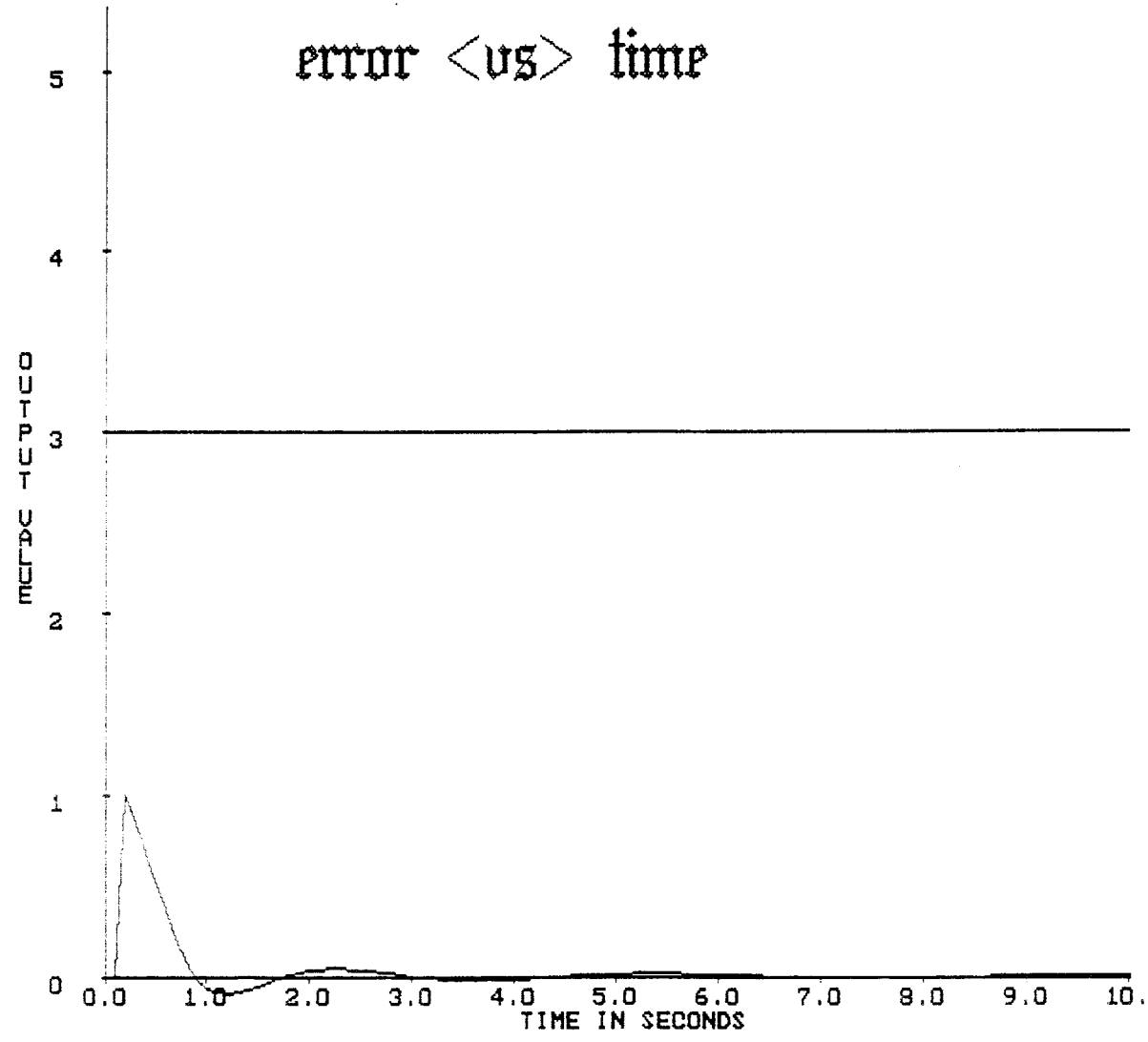
```
main2()
    {
    int gd=DETECT,gm,i,j,a=320,b=240,c=200,n=75;
    float x1,x2,x3,x4,t;
    initgraph(&gd,&gm,"c:\\bc4\\bgi");
    setcolor(15);
    for(i=0;i<n;i++)
    {
    sound(i*10);
    setcolor(15);
    circle(a,b,c-i);
    circle(a,b,i);
    rectangle(i,i,640-i,480-i);
    }
    nosound();
    for(i=0;i<185;i++)
    {
    sound(i*50);
    setcolor(0);
    circle(320,240,i);
    }
    setcolor(15);
    settextstyle(4,0,8);
    for(i=100;i<2000;i+=100)
    {
    sound(i);
    settextstyle(4,0,8);
    outtextxy(175,90,"fuzzy");
    outtextxy(230,160,"logic");
    outtextxy(200,220,"controller");
    settextstyle(1,0,5);
    }
    nosound();
   // getche();
    setfillstyle(0,0);
    for(i=0;i<480;i++)
    bar(0,i,640,50+i);
   // getche();
    }
```

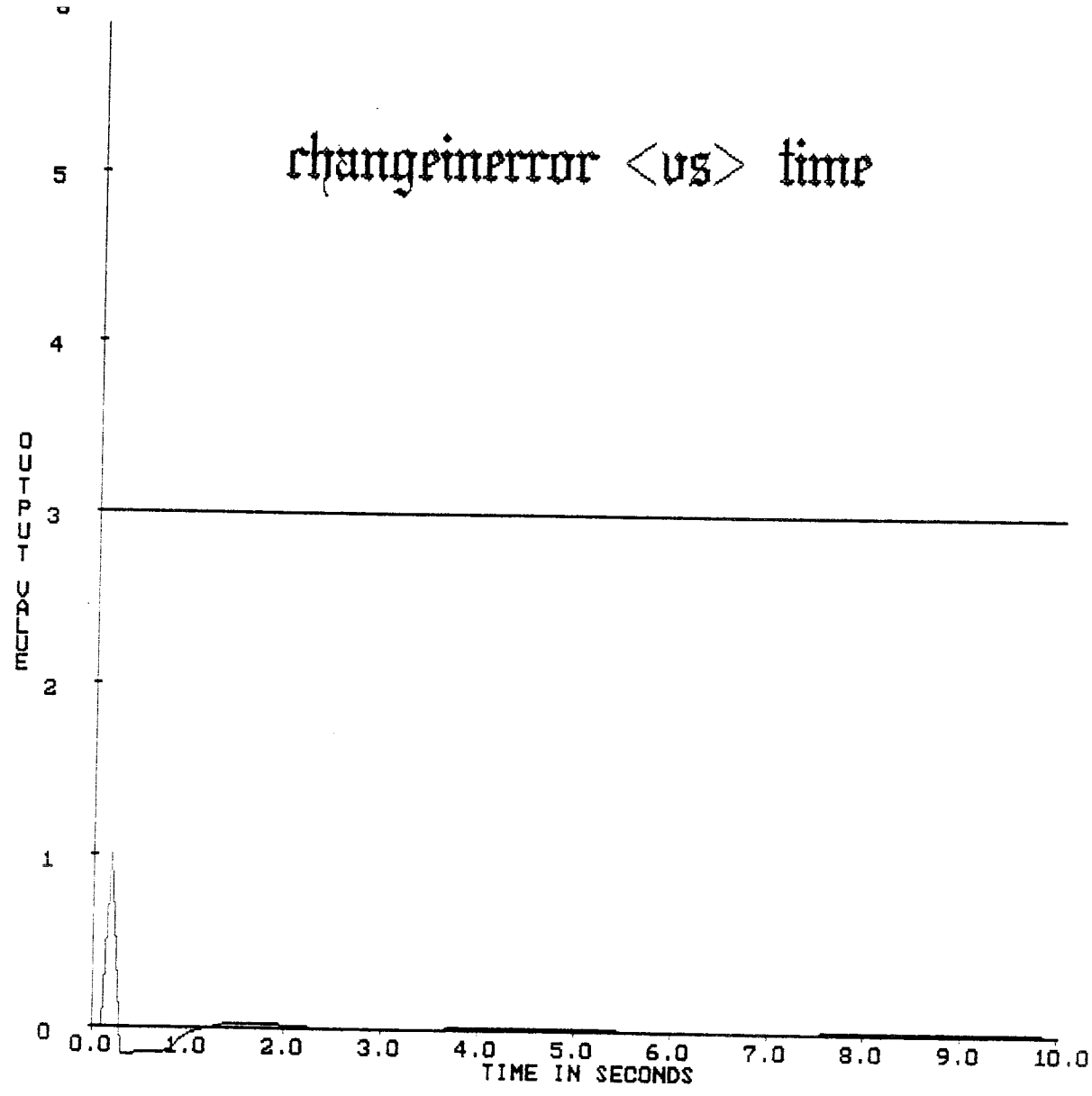
press enter to show the error graph

****



****
press enter to show the changeinerror graph
****




changeinerror <vs> time
OUTPUT VALUE
0
1
2
3
4
5
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
TIME IN SECONDS