FUZZY CONTROL USING GENETIC ALGORITHM

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering & Operations Research)

By

SRIVEENA NAMANI

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2006

MIT-334/2006-29/SN-SK

CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation entitled "**FUZZY CONTROL USING GENETIC ALGORITHM**" towards partial fulfillment of the requirements for the award of the degree of **Master of Technology** in **Electrical Engineering** with specialization in **System Engineering and Operations Research**, submitted to Electrical Engineering Department, Indian Institute of Technology Roorkee, Roorkee, is an authentic record of my own work carried out from July 2005 to June 2006, under the guidance of **Dr. Surendra Kumar**, Assistant Professor, Department of Electrical Engineering, IIT Roorkee.

The matter embodied in this dissertation has not been submitted for the award of any other degree or diploma.

Date: 15/6/06

Place: Roorkee

[illegible]

(SRIVEENA NAMANI)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

[illegible]

(Dr. Surendra Kumar) 15/6/2

Assistant Professor

Electrical Engg Department,

I.I.T. Roorkee

Roorkee- 247667

INDIA

ACKNOWLEDGEMENT

Though the deepest gratitude can only be felt inside my heart, but in words with my deepest esteem I wish to express my deep sense of gratitude and sincere thanks to my beloved guide **Dr.Surendra Kumar** Department of Electrical Engineering, IIT Roorkee, for being helpful and a great source of inspiration. His keen interest and constant encouragement gave me the confidence to complete my thesis work successfully. This work is simply the reflection of their thoughts, ideas, and concepts. I am highly indebted to them for their kind and valuable suggestions and of course their valuable time during the period of this work. The huge quantum of knowledge I had gained during their inspiring guidance would be immensely beneficial for my future endeavors.

I am very thankful to Prof. S.P.Gupta, Head of the Electrical Engineering Department, for supporting my effort.

I thank all the teaching and non teaching staff members of the department who have contributed directly or indirectly in successful completion of my dissertation work.

I also avail this opportunity to thank all my friends for their continuous support and enthusiastic help and encouragement.

Finally, I would like to say that I am indebted to my parents for everything that they have given to me. I thank them for the sacrifices they made so that I could grow up in a learning environment. They have always stood by me in everything I have done, providing constant support, encouragement and love.

(Sriveena Namani)

ABSTRACT

Many non-linear, inherently unstable systems exist whose control using conventional methods is both difficult to design and unsatisfactory in implementation. Fuzzy Logic Controllers are a class of non-linear controllers that make use of human expert knowledge and an implicit imprecision to apply control to such systems. The performance of Fuzzy Logic controller depends on its control rules and membership functions. Hence it is very important to adjust these parameters. The incorporation of genetic algorithm into a fuzzy design process adds an 'intelligent' dimension to the fuzzy controller enabling it to create and modify its rules. Genetic algorithms give the possibility of adjusting membership functions down to the level of individual rules.

In this work, the idea of model generation and optimization is explored. Fuzzy process models will be generated and parameters of fuzzy logic controller such as centre of membership function and weight of the rules are optimized using the power of genetic algorithms. The Inverted pendulum system is used as a test system for this approach and studied performances obtained from Fuzzy controller with the help of SIMULINK, Fuzzy logic Toolbox of the MATLAB 7.01 software and Genetic algorithms.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Fuzzy logic control has been developing rapidly in recent years [5], and is being used successfully and widely in an increasing number of application areas, especially in control of complex processes, such as the control of the pH of a laboratory acid-base system ,the DC motor's speed control etc. At the same time, many scholars have been attracted by the problems of combining the fuzzy logic control with other new intelligent methods, and there are many successful examples such as: achieving fuzzy controller with neural networks, optimizing neural network configurations with fuzzy logic and optimizing fuzzy controller with new optimization methods [5,8]. Conventional control theory is well suited for applications where the process can be reasonably well characterized in advance and where the number of parameters that must be considered is small. There are many important processes, however, those are not well characterized or is subject to a large number of uncontrolled, changeable or unmeasurable parameters.

Several studies have shown fuzzy logic control to be an appropriate method for the control of complex continuous unidentified or partially identified processes [5], many of which cannot easily be modeled in a mathematical way. This is because, unlike a conventional process controller such as a PID controller, no rigorous mathematical model is required to design a good fuzzy controller and in many cases they can also be implemented more easily. However, this simplicity with which they can be implemented also presents a bottleneck in their design. Fuzzy controllers rely on heuristic knowledge that is subject to the designer's interpretation and choice. However in the absence of such expert knowledge the design of FLC is trial and error approach rather than a guided approach [19].

Genetic algorithms provide a way of surmounting this shortcoming. This approach is a search and optimization technique [3]. These algorithms use some of the concepts of evolutionary theory [3] and provide an effective way of searching a large and complex solution space to give close to optimal solution.

In this report the application of GA to the design and optimization of fuzzy logic controllers is discussed. These controllers are characterized by a set of parameters. The Inverted Pendulum system, being commonly used example of a nonlinear, unstable system is used as test system for this approach. Control was successfully achieved in simulations and results from these simulations are presented.

1.2 LITERATURE REVIEW

D.A Linkens, H.O. Nyongesa [5], proposes an offline method for optimization of the membership functions of a fuzzy control rule base and they extend the same method to the complete process of design of a multivariable controller of a non linear process.

F.Herrera,M. Lozano & J.L.Verdegay [6] apply GAs to optimize a rule-base of a FLC for which the membership functions have already been created. The tuning method using GA fits the membership function of the fuzzy rules dealing with the parameters of the membership functions.

Li RenHou, Zhang Yi [8], proposed a method of designing fuzzy logic controller for complex processes. The selection of genetic algorithm control parameters is discussed for design of multi input multi output fuzzy control system.

Varsek, et al.[9],proposed a genetic algorithm based technique to design and tune the parameters of fuzzy controller and take the control of inverted pendulum to indicate the effect of GA.

C.L.Karr [10] ,applied GA's to find the position and shape of the membership functions. These parameters are coded in a chromosome, and the genetic search finds the functions that best control the system, given some evaluation function. This approach has the advantage that the controller can adapt to changes in the process during the interactions, turning the FLC into an adaptive controller.

M.A.Lee, H.Takagi [11], devised another interesting approach to the fusion of fuzzy logic and genetic algorithms. They have presented a method to control the parameters of the genetic algorithm that is mutation rate, crossover rate by fuzzy logic.

L.A.Zadeh [12], who first introduced the fuzzy logic theory in 1965, combined the multi valued logic, probability theory, artificial intelligence and neural networks to develop this digital control methodology that simulates human thinking by incorporating the imprecision inherent physical systems and uncertainty.

Mamdani [13], and his coworkers applied the fuzzy control concepts to several systems such as steam engines, warm level systems etc. In all case the fuzzy controller is located at the error channel and is composed by a fuzzy algorithm that relates (and converts) significant observed variable to control actions. The fuzzy rules employed depend on the type of system under control as well as on the heuristic functions used.

R.Palm [14] proposed to achieve the optimal adjustment in the input SF with the help of input output cross correlation function , though he assigned a higher priority to the tuning of output SF over that of input SF's. Here the input data are assumed to follow a Gaussian distribution whose parameters are unknown. An optimal input SF is obtained by maximizing the cross correlation function which is a measure of the statistical dependence input and output.

Rajini K. Mudi and Nikhil R.Pal [15], proposed a simple and robust model independent self tuning scheme for fuzzy logic controller. Here the output scaling factor is adjusted on line by fuzzy rules according to the current trend of the controlled process. The rule base for tuning the output SF is defined on error and change in error of the controlled variable using the most natural and unbiased membership functions.

K.Belarbi, F. Titel ,et al. [16], proposed how genetic algorithm is used to find stabilizing controllers that minimize the number of rules. They modeled rules with binary weights on which constraints are imposed in order to ensure stability.

Bandyopadhayay, R.Chakraborty, U.K.Patranabis [17], proposed a method for tuning the parameters of the PID controller using fuzzy genetic approach. The technique adopted is based on the format of dead beat control. Here the fuzzy inference mechanism has been used for predicting the future values of the controller output

P.Wang, D.P.Kwok [18], applied genetic algorithms to build up an optimization mechanism to refine rule base of fuzzy PID controller. This scheme automatically optimizes the whole base of Fuzzy PID control rules under various performance indices.

Gregory V.TAN and Xiheng HU [19], proposes a method for the design of fuzzy logic controller using genetic algorithms. The parameters of the fuzzy logic controller such as membership functions of the input and output variables are coded in real numbers.

O.Cordon,F. Gomide ,F.Herrera, et al. [20], provides an account of genetic fuzzy systems with special attention to genetic fuzzy rule based systems. Here they concerned with the critical evaluation of the contribution of the genetic algorithms to fuzzy knowledge extractions.

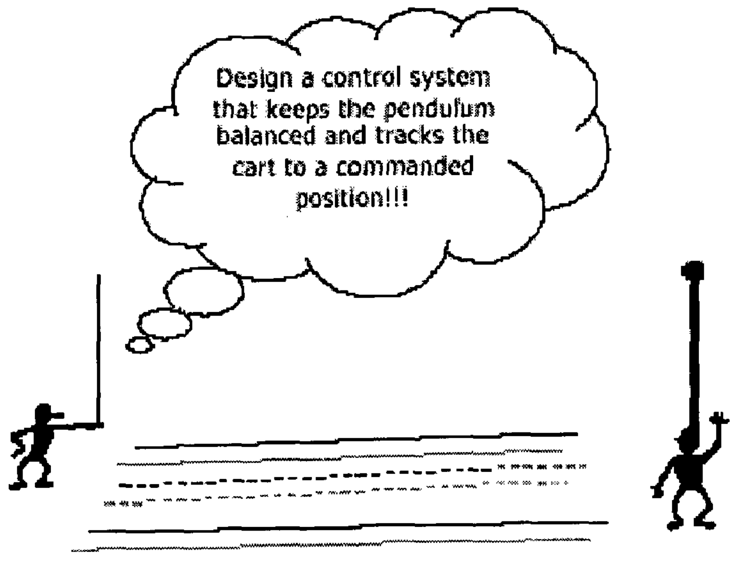
CHAPTER 2

INVERTED PENDULUM PROBLEM AND STATE-SPACE DESIGN

In this chapter the Inverted Pendulum System is examined and analyzed with a view to obtaining its equations of motion and then to linearise these equations in order to find a State-Space-based model of the system. This model is used to design a controller, which is applied to a non-linear model of the system.

2.1 INTRODUCTION:

When we tried to balance a broomstick on our index finger on the palm of our hand, we had to constantly adjust the position of our hand to keep the object upright. An Inverted Pendulum does basically the same thing.


Design a control system that keeps the pendulum balanced and tracks the cart to a commanded position!!!

Just like the broomstick, an Inverted Pendulum is an inherently unstable system. Force must be properly applied to keep the system intact. To achieve this, proper control theory is required. The Inverted Pendulum is essential in the evaluating and comparing the various control theories.

2.2 MATHEMATICAL ANALYSIS [22]

The Inverted Pendulum is made up of a cart on top of which a pole is provided as shown in Figure-2.1. The Cart is constrained to move only in the horizontal x direction and while the Pendulum can only rotate in the x-y plane.

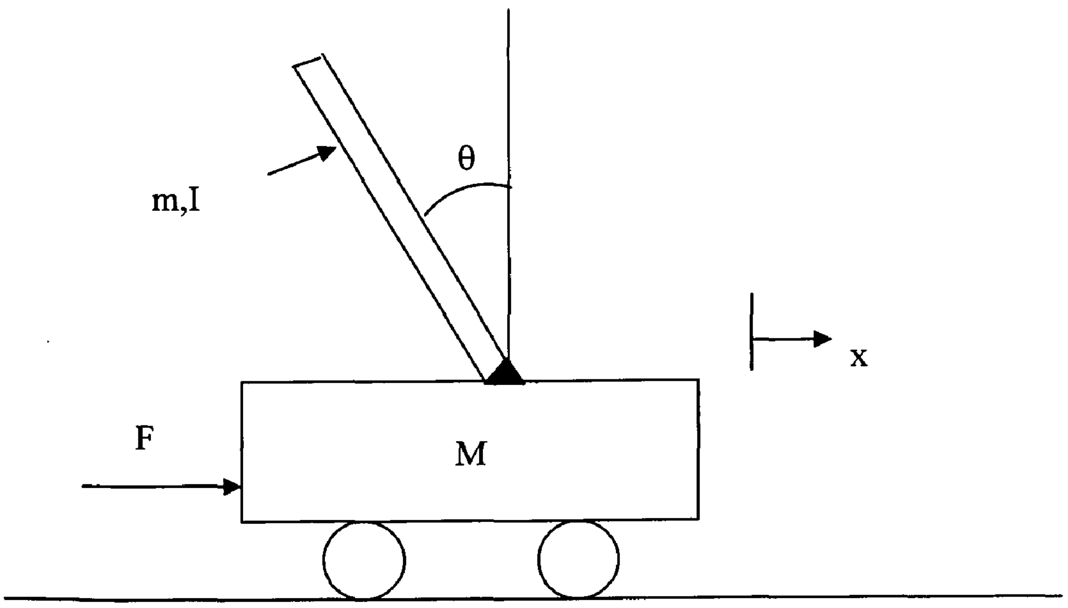


Figure 2.1 Inverted Pendulum

In order to obtain the Inverted Pendulum's model, the system's dynamics is analyzed using the cart and pendulum's free body diagram. The modeling parameters are shown below,

M	=	Mass of the cart
m	=	mass of the pendulum
b	=	friction of the cart
l	=	length to pendulum center of mass
I	=	inertia of the pendulum
F	=	force applied to the cart
x	=	cart position coordinate
θ	=	pendulum angle from vertical

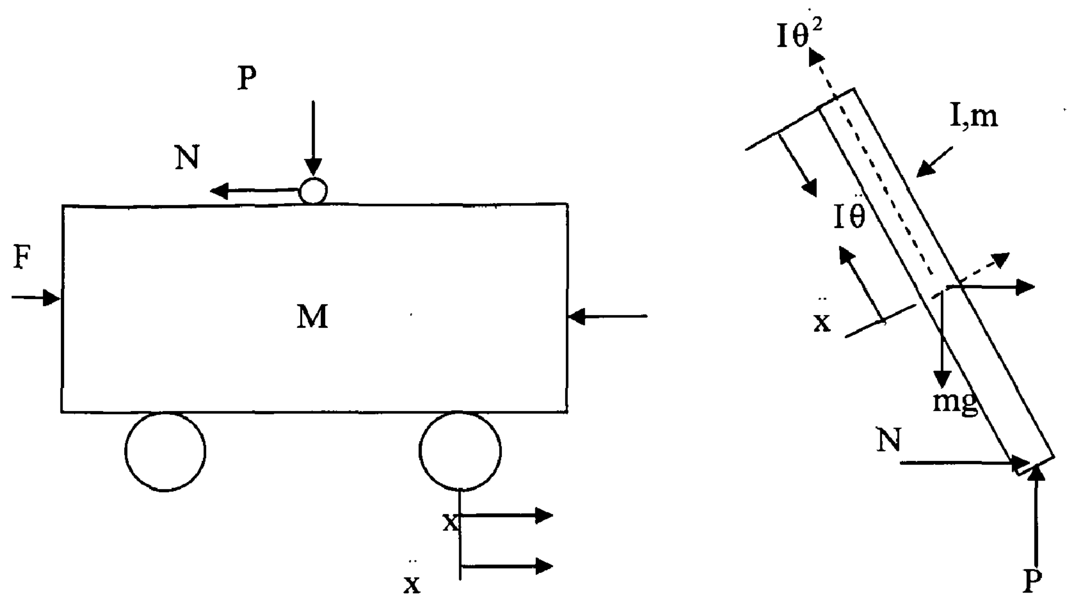


Figure 2.2 Free Body Diagrams of Pendulum

Summing the forces in the Free Body Diagram of the cart in the horizontal direction, you get the following equation of motion,

$$M\ddot{x}+b\dot{x}+N=F \quad \text{........(1)}$$

Note that you could also sum the forces in the vertical direction, but no useful information would be gained. Summing the forces in the Free Body Diagram of the pendulum in the horizontal direction, you can get an equation for N,

$$N=m\ddot{x}+ml\ddot{\theta}\cos\theta-ml\dot{\theta}^{2}\sin\theta \quad \text{........ (2)}$$

If you substitute equation (2) into the equation (1), you get the first equation of motion for this system,

$$(M+m)\ddot{x}+b\dot{x}+ml\ddot{\theta}\cos\theta-ml\dot{\theta}^{2}\sin\theta=F \quad \text{.... (3)}$$

To get the second equation of motion, sum the forces perpendicular to the pendulum. Solving the system along this axis ends up saving you a lot of algebra. You should get the following equation,

$$P\sin\theta+N\cos\theta-mg\sin\theta=ml\ddot{\theta}+m\ddot{x}\cos\theta \quad \text{.... (4)}$$

To get rid of the P and N terms in the equation above, sum the moments around the centroid of the pendulum to get the following equation,

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \quad \ldots\ldots (5)$$

Combining these last two equations, you get the second dynamic equation,

$$\left(I + ml^2\right)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \quad \ldots\ldots(6)$$

2.3 LINEARISED EQUATIONS

Linearize the above equations (3) and (6) so that cosθ = -1, sinθ = -θ, and (d(θ)/dt)^2 = 0. After linearization the two equations of motion become (where u represents the input),

$$\begin{aligned}&\left(I + ml^2\right)\ddot{\theta} - mgl\theta = ml\ddot{x}\\ &\left(M + m\right)\ddot{x} + b\dot{x} - ml\ddot{\theta} = u\end{aligned} \quad \ldots\ldots\ldots(7)$$

2.4 STATE SPACE DESIGN

After a little algebra, the linearized system equations can also be represented in state-space form:

$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & \dfrac{-(I+ml^2)b}{I(M+m)+Mml^2} & \dfrac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \dfrac{-mlb}{I(M+m)+Mml^2} & \dfrac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ \dfrac{I+ml^2}{I(M+m)+Mml^2}\\ 0\\ \dfrac{ml}{I(M+m)+Mml^2}\end{bmatrix}u$$

$$y = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \theta\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ 0\end{bmatrix}u$$

The C matrix is 2 by 4, because both the cart's position and the pendulum's position are part of the output. For the state-space design problem we will be controlling a multi-output system so we will be observing the cart's position from the first row of output and the pendulum's with the second row.

Parameters used for design are:

M	Mass of the cart	0.5kg
m	mass of the pendulum	0.2kg
b	friction of the cart	0.1N/m/sec
l	length to pendulum center of mass	0.3m
I	inertia of the pendulum	0.006kg*m^2

2.4.1 OPEN LOOP RESPONSE

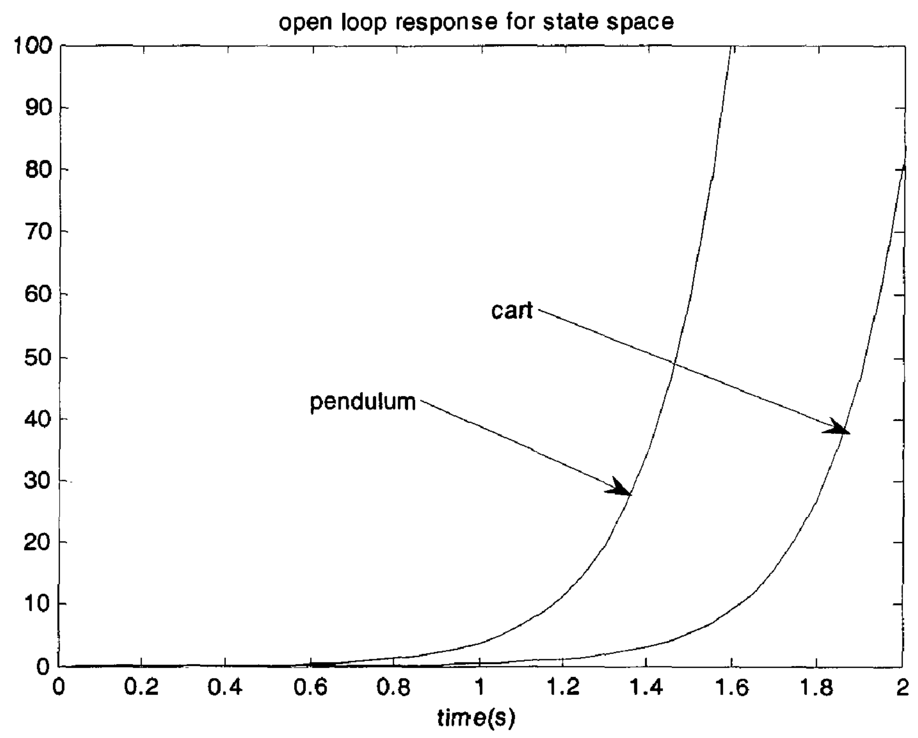


Figure 2.3 Open loop Response of system

As you can see, this should confirm your intuition that the system is unstable in open loop.

2.4.2 CLOSE LOOP RESPONSE

Using state-space methods it is relatively simple to work with a multi-output system, so in this example we will design a controller with both the pendulum angle and the cart position in mind.

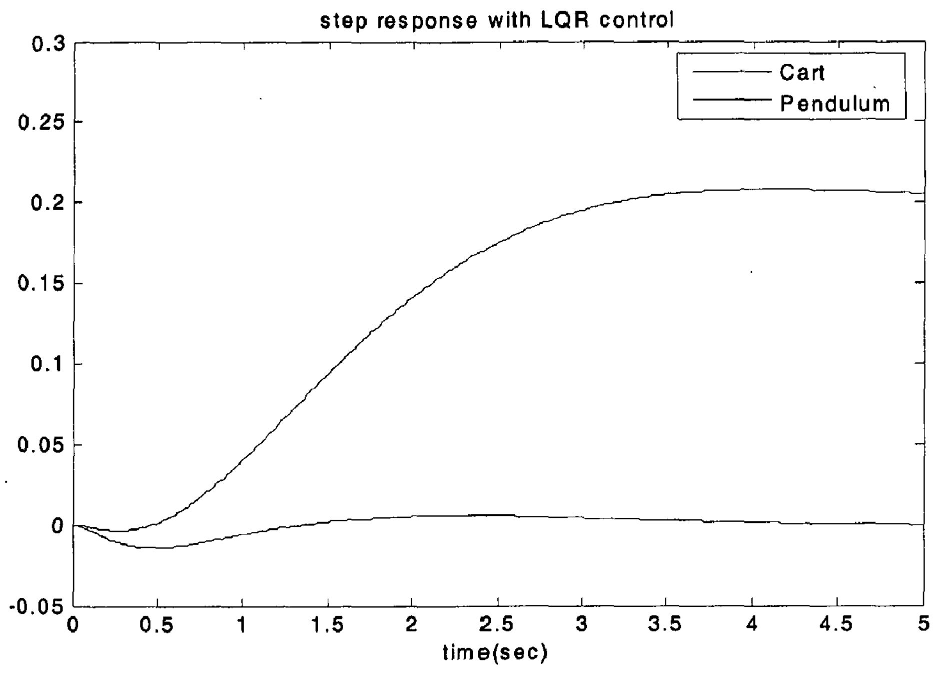


Figure 2.4 Closed loop Response of the system

This controller successfully controls the system as long as the pole is not allowed to stray too much from the equilibrium position. When the pole angle gets too large it is unable to restore it to the upright position. A controller that can successfully achieve this needs to be able to handle the non-linearities. For this reason, Fuzzy Logic Controllers are used to solve the problem.

FUZZY LOGIC CONTROLLER

3.1 BACK GROUND

Fuzzy Control is based on the principles of Fuzzy Logic developed by Zadeh [12] in 1965 and the industrial application of the first fuzzy controller by E.H. Mamdani in 1974 [13], fuzzy systems have obtained a major role in engineering systems and consumer's products in 1980s and 1990s. New applications are presented continuously. It is a non-linear control method, which attempts to apply the expert knowledge of an experienced user to the design of a controller.

3.2 INTRODUCTION

Conventional control system design depends upon the development of a mathematical description of the system's behaviour. This usually involves assumptions being made in relation to the system dynamics and any non-linear behaviour that may occur. In cases where assumptions in respect of non-linear behaviour cannot be made, the need to describe mathematically, ever increasing complexity becomes difficult and perhaps infeasible.

Fuzzy logic [12,13] is the application of logic to imprecision and has found application in control system design in the form of Fuzzy Logic Controllers (FLCs) [13]. Fuzzy logic controllers facilitate the application of human expert knowledge, gained through experience, intuition or Experimentation, to a control problem. Such expert knowledge of a system's behaviour and the necessary intervention required to adequately control that behaviour is described using imprecise terms known as "linguistic variables". The imprecision of linguistic variables reflects the nature of human observation and judgment of objects and events within our environment, and their use in FLCs thus allows the mapping of heuristic, system-related information to actions observed to provide adequate system control. In this way, FLCs obviate the need for complex mathematical descriptions of non-linear behaviour to the nth degree and thus offer an alternative method of system control [1,2].

3.3 FUZZY LOGIC

What Is Fuzzy Logic?

Fuzzy logic is based on natural language. The basis for fuzzy logic is the basis for human communication. This observation underpins many of the other statements about fuzzy logic [4]:-

So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality. — Albert Einstein

As complexity rises, precise statements lose meaning and meaningful statements lose precision. — Lotif Zadeh

Fuzzy logic is based on the theory of fuzzy sets where variables can have differing degrees of membership of sets. This is unlike the more familiar crisp set theory where a variable is either a full member of a set or it is not a member of that set at all. The degree to which a variable belongs to a set can vary between 0 and 1.

A universe of discourse is defined as the whole range of fuzzy sets to which a variable can belong. Each set on this universe of discourse is referred to as a membership function and is often described using a 'linguistic variable'. On a universe of discourse, a variable has a degree of membership of each membership function that varies between 0 and 1.

Fuzzy Logic uses rules with antecedents and consequents to produce outputs from inputs. The antecedents are the inputs that are used in the decision-making process or the "IF" parts of the rules. The consequents are the implications of the rules or the "THEN" parts.

3.3.1 FUZZY LOGIC EXAMPLE

An example of a fuzzy set is the set of humans who could be described as young. Most people would agree that anyone aged between zero and twenty could be described as being definitely young, whereas anyone over forty would be described as not at all young. The ages between twenty and forty are more of a grey area, however. The closer a person's age is to twenty the more readily they could be described as young. This fuzzy set is displayed in Figure 3.1 below. According to this fuzzy set a

Person aged fifteen is definitely young, i.e. Their degree of membership of the fuzzy set *Young* is one. However, it is less clear-cut whether a person aged thirty could be described as young, i.e. Their degree of membership is less than one, in this example,

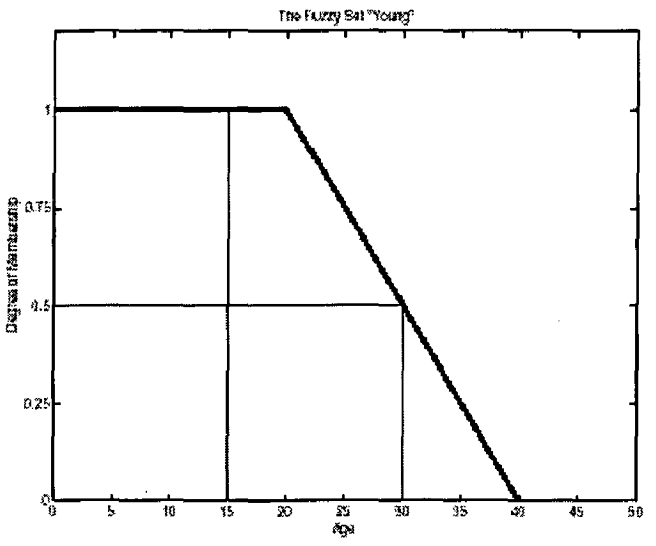


Figure 3.1 The Fuzzy Set "Young"

The Universe of Discourse of a variable is the range of values that variable can take. Many fuzzy sets can be defined on one Universe of Discourse and a single variable can have membership of more than one fuzzy set. For example, in Figure 3.2 below we return to our "Age" example and examine the Universe of Discourse on which, in addition to *Young*, the fuzzy sets *Middle Aged* and *Old* are added. Here we can see that a person aged 35 has membership of two sets, *Young* and *Middle Aged* with degrees of membership of 0.25 and 0.33 respectively.

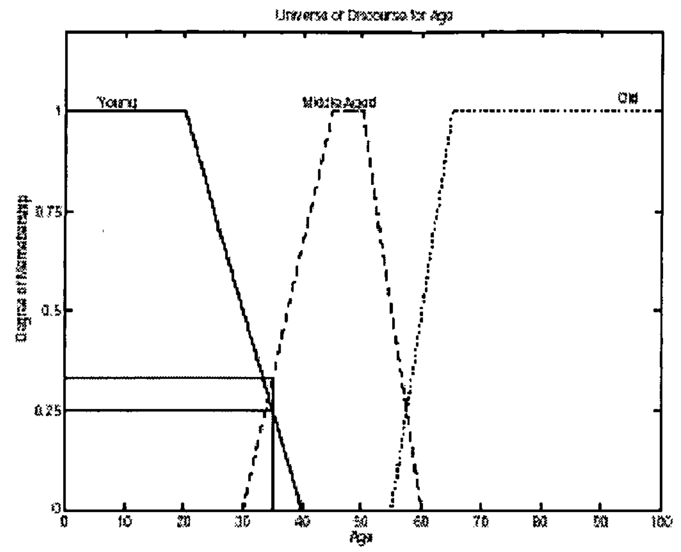


Figure 3.2 "Age" Universe of Discourse

3.4 FUZZY CONTROL

Fuzzy Control [1,2] applies fuzzy logic to the control of processes by utilizing different categories, There are specific components characteristic of a fuzzy controller to support a design procedure. In the block diagram in Fig.3.3, the controller is between a preprocessing block and a post-processing block. The following explains the diagram block by block.

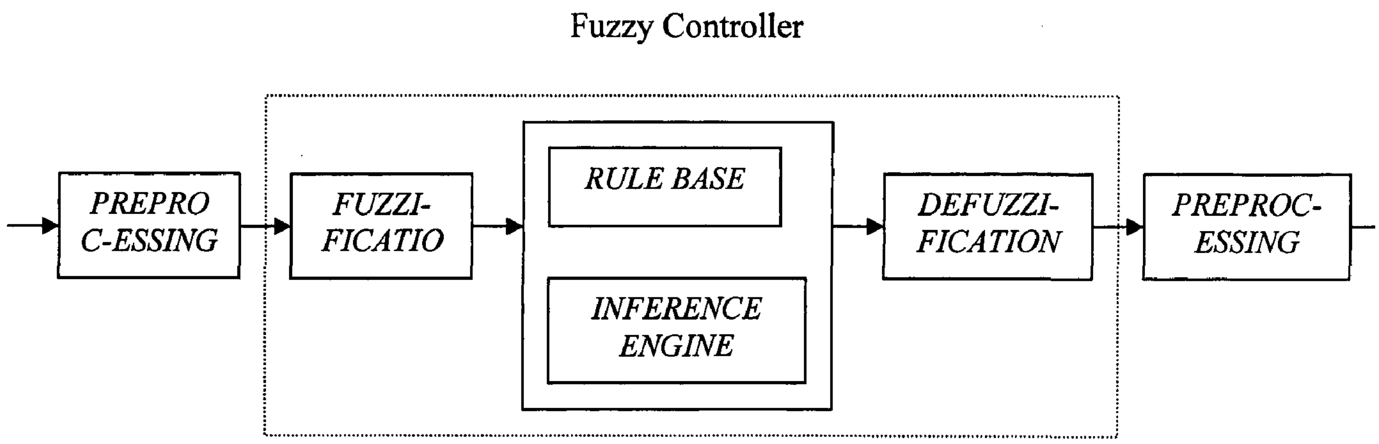


Fig 3.3 Block diagram of Fuzzy Logic Controller

3.4.1 PREPROCESSING

The inputs are most often hard or crisp measurements from some measuring equipment, rather than linguistic. A preprocessor, the first block in Fig.3.2, conditions the measurements before they enter the controller.

Examples of preprocessing are:

- Quantization in connection with sampling or rounding to integers;
- normalization or scaling onto a particular, standard range;
- filtering in order to remove noise;
- Differentiation and integration or their discrete equivalences.

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

3.4.2 FUZZIFICATION

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

3.4.3 RULE BASE

It contains the knowledge of the expert in a form of rules that could be utilized by the Fuzzy Inference Engine (FIE). The rules may use several variables both in the condition and the conclusion of the rules.

The controllers can therefore be applied to both multi-input-multi-output (MIMO) problems and single-input-single-output (SISO) problems.

The following are example of rules:

- If error is Zero and change in error is Zero then the control action is Zero
- If error is Zero and change in error is Negative, then the control action is Negative
- If error is Positive and change of error is Zero then the control action is Positive
- If error is Positive and change of error is Negative then the control action is Zero

3.4.4 FUZZY INFERENCE ENGINE

The task of fuzzy inference is to make decision. However, it can only provide a fuzzy output. This is main part of fuzzy logic controller. It is the heart of controller.
The rules reflect the strategy that the control signal should be a combination of the reference error and the change in error, for each rule, the inference engine looks up the membership values in the condition of the rule.

3.4.5 DEFUZZIFICATION

The fuzzy output provided by the FIE should be decoded into non-fuzzy output to make it a useful to control manipulator or plant's input. This is illustrated in figure 3.4 below:

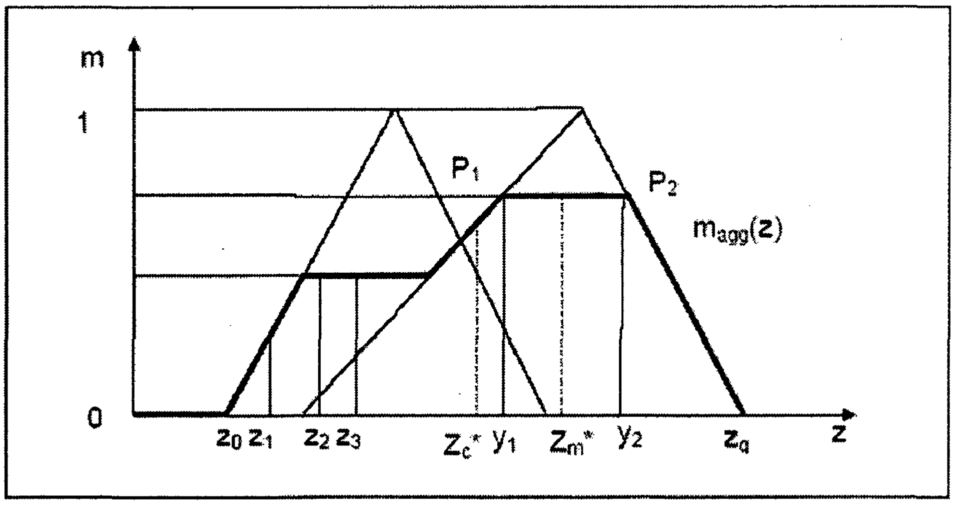


Fig 3.4. An example of an aggregated fuzzy output.

There are several defuzzification methods.

Centre of gravity (COG)

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

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

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

Mean of maximum (MOM)

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

Let the projection of the flat segment P_1P_2 with maximum height on z axis be the interval [y1, y2] (See Fig. 3.4.). Then Z_m is determined by the formula:-

$$Z_m = (y_1 + y_2) / 2$$

Left maximum (LM), and Right maximum (RM)

Another possibility is to choose the leftmost maximum (LM), or the rightmost maximum (RM). In the case of a robot, for instance, it must choose between left and right to avoid an obstacle in front of it. The defuzzifier must then choose one or the other, not something in between.

These methods are indifferent to the shape of the fuzzy set, but the computational complexity is relatively small.

Bisector of area (BOA):

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

$$u=\{x \mid \int_{Min}^{x} \mu(x)\,dx = \int_{x}^{Max} \mu(x)\,dx\}$$

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

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

3.4.6 POST PROCESSING

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

The post processing block often contains an output gain that can be tuned, and sometimes also an integrator.

CHAPTER 4

GENETIC ALGORITHM

4.1 BACKGROUND

The American John Holland devised the first genetic Algorithms in the 1970's. Genetic Algorithms were invented to mimic some of the processes observed in natural evolution. Many people, biologists included, are astonished that life at the level of complexity that we observe could have evolved in the relatively short time suggested by fossil records. There is constant debate regarding the truth of Darwinian evolutionary theory as one scientist stated[3]:

"The chance that a functioning cell could evolve in that time can be likened to the probability that a tornado sweeping through a junkyard might assemble a Boeing 747."

--- Sir Fredrick Hoyle

Regardless of its validity, the idea with GA is to use this power of evolution to follow the principles first laid down by Charles Darwin in his "survival of the fittest" theory. In nature, competition among individuals for scarce resources results in the fittest individuals dominating over the weaker ones. GAs attempt to find the optimal solution from the search space. Genetic Algorithms (GA) are search algorithms based on the mechanics of natural selection and natural genetics. [5, 3] GAs are adaptive search techniques which simulate both natural inheritance by genetics and a Darwinian struggle for survival.

A GA starts with a population of candidate solutions that evolve through generations of competition and reproduction until convergence to one solution. As such they represent an intelligent exploitation of a random search used to solve problems. Although randomized, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space.

4.2 INTRODUCTION [3, 4]

The Genetic algorithm (GA) is a randomized search and optimization technique guided by the principle of natural genetic systems. They maintain population of knowledge structures that represent candidate solution and let those populations evolve time through competition and controlled variation

The main advantage of the GA formulation is that fairly accurate results may be obtained using a very simple algorithm. It is a method of finding a good answer to a problem, based on the feedback received from its repeated attempts at a solution. The objective or fitness function is a judge of the GA's attempts. Gas do not know how to derive a problem's solution, but they do know, from the objective function, how closely they are to a better solution.

The GA maintains a set of possible solutions (population) represented as a string of, typically, binary numbers (0/1).New strings are produced in each and every generation by the repetition of a two-step cycle. This involves first decoding each individual string and assessing its ability to solve the problem. Each string is assigned fitness values, depending on how well it is performed in an environment. In the second stage, the fittest string is preferentially choosen for recombination, which involves the selection of two strings, and the switching of the segments to the right of the meeting point of the two strings. This is called crossover. Another genetic operator is mutation. It is used to maintain genetic diversity within a small population of strings. There is a small probability that any bit in a string will be flipped from its present value to its opposite(e.g., 0 to 1), this prevents certain bits from becoming fixed at a specific value due to every string in the population having the same value, often causing premature convergence to a non-optimal solution. An additional common feature of the GA is the automatic inclusion of the best procedure prevents a good string from being lost by the probabilistic nature of reproduction and speeds convergence to a good solution.

The GA goes through the following cycle: Evaluate, Select and Mate, and Mutate until some kind of stopping criteria are reached.

Genetic algorithms differ from other optimization and search procedures in the following ways [3]

- Gas work with a coding of the parameter set, not the parameters themselves. Therefore, they can easily handle integral or discrete variables.
- GAs use probabilistic transition rules ,not deterministic rules
- Gas search from a population of points, not a single point. Peaks of the search space can be climbed concurrently since works with a large number of points simultaneously. The problem of only finding local extrema is resolved.
- Gas use only objective function information, not derivatives or other auxiliary knowledge. Therefore, they can deal with the non-smooth, non-continuous, and non-differentiable function.
- Sometimes near optimal solution that can be generated quickly, using Gas, are desirable than optimal solutions which require a large amount of time.

4.3 PROCEDURE OF GENETIC ALGORITHM [4]

A genetic algorithm for a particular problem must have the following five components.

1. A genetic representation for the potential solution to the problem
2. A way to an initial population of potential solutions
3. An evaluation function that plays the role of the environment, rating solution in terms of their 'fitness'
4. Genetic operators that alter the composition of the offspring
5. Values for the various parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators. etc).

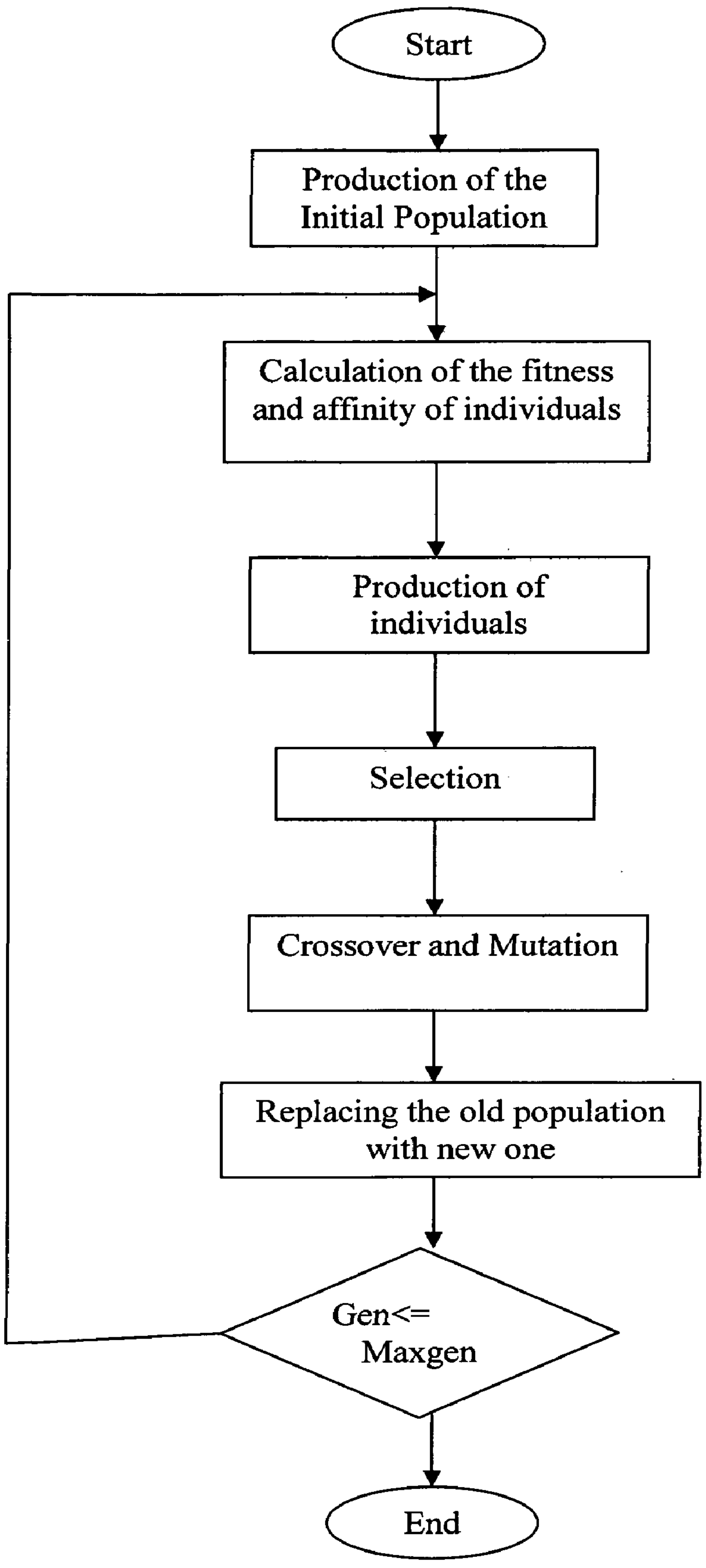

Start
Production of the Initial Population
Calculation of the fitness and affinity of individuals
Production of individuals
Selection
Crossover and Mutation
Replacing the old population with new one
Gen<= Maxgen
End

Fig 4.1 Flowchart for the GA [4]

4.4 GENETIC REPRESENTATIONS [3, 4]

Representing or encoding the problem in hand when applying GA is a vital task. Encoding can be defined as the chromosomal representation of the problem.
There are a few ways of encoding these chromosomes such as integer, real-valued and ternary but one of the most popular ways is binary encoding (bit string), because it is a simpler string to operate on.
For binary encoding each chromosome is constructed by stringing binary representations of vector components end to end (see Figure 4.2 below). The length of each chromosome depends on the vector dimension and the desired accuracy.

1011010110101110

Figure 4.2 A sample binary encoded chromosome consisting of a string of 16 bits

With real-valued encoding, the parameters are kept in their real number format. Both forms of encodings are used in practice.
The advantages of using a binary format is that it
... maximizes the number of hyper plane partitions directly available in the encoding for schema processing.
In other words, binary alphabets allow for greater sampling of the solution space and for the processing of more combinations of alleles.
However encoding using higher cardinalities can be more efficient. For example, if a certain parameter could take on five possible values then it would need to be encoded using three bits in a binary scheme. However, this leads to eight possible alleles, three of which are superfluous. Using a five-letter alphabet in this case would lead to more efficient coding.
It is also possible to represent genes using arrays, trees, lists etc .It is necessary to ensure that a suitable representation of solutions with meaningful and problem specific genetic operators is considered.

4.5 INITIALIZATION AND SELECTION [6]

4.5.1 Initialization

To initialize a population, we can simply set some 'pop_size' number of chromosomes randomly in a bitwise fashion. However, if we do have some knowledge about the distribution of potential optima, we may use such information to arrange the set of initial potential solutions. The rest of the algorithm is straightforward: in each generation, we evaluate each chromosome (using the function f on the decoded sequences of variables), select a new population with respect to the probability distribution based on fitness values, and alter the chromosomes in the new population using mutation and crossover operators. After some generations, when no further improvement is observed, the best chromosome represents an (possibly a global) optimal solution. Often, we stop the algorithm after a fixed number of iterations, depending on the speed and resource criteria

4.5.2 Selection

➢ A popular selection algorithm is stochastic sampling with replacement, more commonly known as the "Roulette Wheel" algorithm, so called because this method works in a way that is analogous to a roulette wheel. Each individual in a population is allocated a share of a wheel, the size of the share being in proportion to the individual's fitness. A pointer is spun (a random number generated) and the individual to which it points is selected. This continues until the requisite number of individuals has been selected. An individual's probability of selection is thus related to its fitness ensuring that fitter individuals are more likely to leave offspring.

The fitness values are then calculated using the function eval (v_i) for each chromosome vi(i=1,....pop_size). The total fitness of the population is given by

$$F = \sum_{i=1}^{pop\_size} eval(v_i)$$

The probability of selection for each chromosome v_i (i=1,.....,pop_size) is

P_i=eval (v_i)/F

And the cumulative probability is

$$q_i = \sum_{j=1}^{i} P_i$$

The selection process is based on spinning the roulette wheel pop_size times, each time selecting a single chromosome for a mew population in the following way.

- Generate a random (float) number r from the range [0,z].
- If the $r<q_1$,select the first chromosome (v_1); otherwise, select the ith chromosome vi(2<=i<=pop_size) such that $q_{i-1}<r<q_i$

A problem with this approach is that the number of times an individual is actually selected has a high variance so there is no guarantee that fitter individuals will be represented in the next generation.

➤ Stochastic Remainder Selection is another popular algorithm. In this method the expectation of the number of times selected is calculated for each individual. The integer portions of this expectation for each individual are assigned deterministically and the fractional remainders are assigned in the same way as in roulette wheel selection. For example, an individual whose expectation of number of times selected was 2.4 would be certain of being selected twice and the probability of being selected a third time would be 0.4. This approach reduces the variance associated with the roulette wheel algorithm and ensures that all individuals with above-average fitness will be represented in the next generation.

4.6 CROSSOVER ALGORITHMS [3, 4]

Once selection is finished crossover is performed. Individuals are paired for Mating and by mixing their strings new individuals are created. The most basic crossover algorithm is known as Single Point Crossover. A single point along the string is chosen and the strings are swapped over at this point.

Parent 1	101|11100
Parent 2	110|10010
Child 1	101 10010
Child 2	110 11100

Figure 4.3 Single Point Crossover

Multipoint crossover algorithms extend simple crossover by selecting multiple crossover points and alternately assigning to the first or second offspring the portions of string between these points.

Parent 1	10|111|100
Parent 2	11|010|010
Child 1	10 010 100
Child 2	11 111 010

Figure 4.4 Multi-point Crossover

Uniform crossover is another crossover algorithm. This time a random mask of 1s and 0s of the same length as the parent strings is generated. If a bit in the mask is 1 then the corresponding bit in the first child will come from the first parent and the second parent will contribute that bit to the second offspring. If the mask bit is 0 the first parent contributes to the second child and the second parent to the first child.

Parent 1	10111100
Parent 2	11010010
Mask	01001101
Child 1	10011110
Child 2	11110000

Figure 4.5 Uniform Crossover

Uniform crossover is the most disruptive of the crossover algorithms, i.e. it is the most likely to cause neighbouring bits that contribute in a positive way to the fitness of the individual to be split up. However at the same time, uniform crossover allows for more extensive searching of the solution space as there are significantly more potential offspring using this method.

4.7 MUTATION ALGORITHMS

For binary coding, there is really only one way to mutate. For each bit generate a random number and if it is less than the specified mutation probability, flip the bit, i.e., if it is a 1 change it to 0 or vice versa.

Original string	10110101	before mutation
Mutated string	10100101	After Mutation

Mutation Point

Fig 4.6 Example of mutation

This mutation probability is generally kept quite low and is constant throughout the lifetime of the GA. However, a variation on this basic algorithm changes the mutation probability throughout the lifetime of the algorithm, starting with a relatively high rate and steadily decreasing it as the GA progresses [6]. This allows the GA to search more for potential solutions at the outset and to settle down more as it approaches convergence.

When real-valued codings are used, the mutation algorithm can be more complex. Many different algorithms are used some of which are as follows:

1 .Uniform Mutation: A random value within the constraints of the variable is choosen.

2. Boundary Mutation: The variable is set to either its lower or upper bound.

3. Non-Uniform Mutation: The variable is assigned a value based on a bell curve that becomes progressively narrower as the GA progresses. This ensures that as convergence is approached, the range within which a variable can be mutated also narrows.

4.8 ELITISM [3]

With crossover and mutation taking place, there is a high risk that optimum solutions may be lost as there is no guarantee that these operations will preserve fitness. To combat this elitist models are often used. In these models, the best individual from a population is saved before any of the operations take place. After the new population is formed and evaluated, it is examined to see if this best structure has been preserved. If not, the saved copy is reinserted back into the population, usually at the expense of the weakest member. The GA then proceeds to perform the operations on this population.

In determining how many cycles of the GA should take place, any of the following stopping criteria can be employed. [4]

- When a particular point in the search space has been found.
  For example, the greatest extrema found up to this point in time.
- When a certain upper limit of generations has been reached.
  For example, thirty or one thousand generations have been cycled through.
- When a mean deviation in the population is obtained.
  For example, when any new solutions are very close to the previous generations solutions, using the mean as a reference.

If after meeting one of these criteria no individuals in the population yield acceptable solutions to the problem at hand, the GA may simply run again, or be continued with extended design criteria. The GA allows the user to pick from a number of potential solutions – it doesn't just yield one solution. It is good at identifying simultaneously these other solutions for a problem that by nature has multiple solutions.

CHAPTER 5

DESIGN FUZZY LOGIC CONTROLLER IN MATLAB

5.1 INTRODUCTION

It is well-known that human control mechanisms define exact mathematical modeling to perform a particular task. This is in sharp contrast with conventional approaches in the design of an automatic control system that often involve the construction of a mathematical model describing the dynamic behaviour of the plant to be controlled, and the application of analytical techniques to the model to derive an appropriate control law. Usually, such a mathematical model consists of a set of linear or nonlinear differential or difference equations, most of which are derived using some form of approximation and simplification. On the other hand, a human control mechanism uses imprecise and qualitative understanding of the processes to be controlled. Knowledge-based control or intelligent control is the name introduced to describe control systems in which the control strategies use behavioral (and not mathematical) description of the process, based on experience gathered by operators and process engineers[16]. Actions are performed either as a result of evaluating rules (reasoning) or as unconscious actions based on presented process behaviour after a learning phase. Intelligence comes in as the capability to reason about facts and rules and to learn about presented behaviour. While simulation results are typically used to 'verify' the approach and some successful implementations have been achieved.

Techniques from intelligent control and this are important, especially to the practitioner seeking a reliable implementation for a control system. The notion that since a control system is 'intelligent', it must automatically be better than other conventional approaches is hype. At the same time, it is bad for control engineers to simply ignore the field of intelligent control as being 'sloppy'. Perhaps it is not as 'tidy' as conventional control, but this is due to the fact that the field of intelligent control is relatively new and unexplored. Intelligent control has certain techniques and concepts to offer; the challenge is to find out what it is good for, and perhaps more importantly, what it is not good for. From a control engineer's perspective, the best way to assess the contributions of intelligent control is to perform careful theoretical and experimental engineering analysis as has been done in the past for conventional control systems. There is a need to build a bridge between conventional and intelligent control. Artificial intelligence field will drive the development of control theory and control technology by providing alternative strategies for the functionality and implementation of controllers for dynamic systems [4]. From the control theory point of view, the ability of intelligent control techniques to deal with uncertain and nonlinear systems is perhaps most significant. The great diversity of nonlinear systems is the primary reason why no systematic and generally applicable theory for nonlinear control design has yet been evolved. A range of traditional methods for analysis and synthesis of nonlinear controllers for specific classes of nonlinear systems exist: phase-plane methods, linearization techniques, and describing function analysis are three examples. The ability of neutral networks/fuzzy systems to represent nonlinear mappings is the feature to be exploited in the synthesis of nonlinear controllers, and many fundamental theoretical questions will need to be addressed.

5.2 FUZZY LOGIC TOOLBOX [21]

The Fuzzy Logic Toolbox is contained in a directory called fuzzy. Type help fuzzy for a listing of help topics. The Fuzzy Logic Toolbox provides tools to create and edit fuzzy inference system (FIS) within the framework of MATLAB. We can integrate our fuzzy systems into simulations with SIMULINK. The Fuzzy Logic Toolbox provides three categories of tools:

* **Command line functions**
* **Graphical interactive tools**
* **Simulink blocks**

To build a system entirely from the command line, the commands newfis, addvar, addmf and addrule would be used. The function newfis creates new FIS structures. It has up to seven input arguments, and the output argument is a FIS structure. The seven input arguments are as follows:

* fisName is the string name of FIS structure; **vee.fis** (I used).
* fisType is the type of FIS; Mamdani type is default.

* andMethod,
* orMethod,
* impMethod,
* aggMethod, and
* defuzzMethod,

Respectively, provide the methods for AND, OR, implication, aggregation, and defuzzification. The defaults are, respectively, min, max, min, max and centroid (centre of area). The function addvar adds a variable to a FIS. It has four arguments in this order:

* the name of the FIS structure in the MATLAB workspace.
* The string representing the type of the variable we want to add ('input' or 'output').
* The string representing the name of the variable we want to add.
* The vector describing the limiting range values (universe of discourse) for the variable we want to add.

```
a = newfis (vee);
a = addvar (a, 'varType', 'varName', 'varBounds');
```

Indices are applied to variables in the order in which they are added; so the first input variable added to a system will always be known as input variable number one for that system. Input and output variables are numbered independently. The Fuzzy Logic Toolbox includes many membership function types. The simplest membership functions are formed using straight lines. Of these, the simplest is the triangular membership function and it has the function name 'trimf'. The triangular curve is a function of three scalar parameters a, b, and c; the parameters a and c locate the 'feet' of the triangle and the parameter b locates the peak. The trapezoidal membership function, trapmf, has a flat top. The trapezoidal curve depends on four scalar parameters a, b, c, and d; the parameters a and d locate the 'feet' of the trapezoid and the parameters b and c locate the 'shoulders'. Each input/output variable existing in MATLAB workspace FIS (variables added by the function addvar) is resolved into a number of different fuzzy linguistic sets. The input/output variables must be fuzzified according to each of these linguistic sets. A membership function can only be added to a variable in an existing MATLAB workspace FIS. Indices are assigned to membership functions in the order in which they are added;

so the first membership function added to a variable will always be known as membership function number one for that variable. The function addmf adds a membership function to FIS. The function requires six input arguments in this order: · A MATLAB variable name of a FIS structure in the workspace. · A string representing the type of variable we want to add the membership function to ('input' or 'output').· The index of the variable you want to add the membership function to (We cannot add a membership function to input variable number two of a system if only one input has been defined). · A string representing the name of the new membership function. · A string representing the type of the new membership function. · The vector of parameters that specify the membership function.

```
a = newfis ('vee') ;
a = addvar (a, 'varType', 'varName', 'varBounds') ;
a = addmf(a, 'varType', 'varIndex', 'mfName', 'mfType', 'mfParams');
```

Probably the trickiest part of the process of building a Fuzzy System is learning the short hand that the fuzzy inference systems use for building rules. This is accomplished using the command line function addrule. Each variable, input, or, output, has an index number, and each membership function has an index number. The rules are built from statements like this: IF input 1 is MF1 and input 2 is MF3 THEN Output is MF2 this rule is turned into a structure according to the following logic. If there are p inputs to a system and q outputs, then the first p vector entries of the rule structure correspond to inputs 1 through p. The entry in column 1 is the index number for the membership function associated with input 1. The entry in column 2 is the index number for the membership function associated with input 2, and so on. The next q columns work the same way for the outputs. Column p + q + 1 is the weight associated with the rule (typically 1; all the rules have equal weightage) and column p + q + 2 specifies the connective used (where and =1 and or =2). The structure associated with the rule shown above is 1 3 2 1 1 The function addrule has two arguments. The first argument is the MATLAB workspace variable FIS name. The second argument is a matrix of one or more rows, each of which represents a given rule. The rule-list matrix takes the very specific format defined above.

```
ruleList = [
            1 1 1 1 1
            1 2 2 1 1 ];
a = addrule (a, ruleList);
```

If the above system has two inputs and one output, the first rule can be interpreted As 'IF input 1 is MF1 and input 2 is MF1 THEN output 1 is MF1'. To evaluate the output of a fuzzy system for a given input, we use the function evalfis. It have the following arguments:

* A number or a matrix specifying the input values. If input is a P × p matrix, where p is the number of input variables, then evalfis takes each of the P rows of input as an input vector and returns the P × q matrix to the variable, output, where each row is an output vector and q is the number of output variables.
* The name of the FIS structure to be evaluated.

It is possible to use the Fuzzy Logic Toolbox by working strictly from the command line. However, in general, it is much easier to build a system graphically. These are five primary GUI tools for building, editing and observing fuzzy inference systems in the Fuzzy Logic Toolbox:

- **The Fuzzy Inference System or FIS Editor.**
- **The Membership Function Editor.**
- **The Rule Editor.**
- **The Rule Viewer and**
- **The Surface Viewer.**

The FIS Editor handles the high level issues for the system: How many input and output variables? What are their names? The Membership Function Editor is used to define the shapes of all the membership functions associated with each variable. The Rule Editor is for editing the list of rules that defines the behavior of the system. The Rule Viewer is a display of the fuzzy inference diagram. The Surface Viewer is used to display the dependency of one of the outputs on any one or two of the inputs. The five primary GUIs can all interact and exchange information. For any fuzzy inference system, any or all of these five GUIs may be open. If more than one of these editors is open for a single system, the various GUI windows are aware of the existence of others and will, if necessary, update related windows; these changes are reflected in the rules shown in the Rule Editor. To start building a fuzzy inference system, type fuzzy at the MATLAB prompt. The generic untitled FIS Editor opens. At the top is a diagram of the system with input and output clearly labeled. By double-clicking on the input or output boxes, you can bring up the Membership Function Editor. Double clicking on the fuzzy rule box in the centre of the diagram will bring up the Rule Editor. Just below the diagram is a text field that displays the name of the current FIS. Lower left of window has a series of pop up menus, and the lower right has fields that provide information about the current variable.

GUI Editors:

Fuzzy	- Basic FIS Editor
mfedit	- Membership Function Editor
ruleedit	- Rule Editor
ruleview	- Rule Viewer
surfview	- Output Surface Viewer

When you save your fuzzy system to the MATLAB workspace, you are creating a variable (whose name you choose) that will act as a MATLAB structure for the FIS system. Once you have created your fuzzy system entirely from the command line or using the GUI tools, you are ready to embed your system directly into Simulink and test it out in a simulation environment. The Fuzzy Logic Toolbox in the Simulink library contains the Fuzzy Logic Controller, and the Fuzzy Logic Controller with Rule Viewer blocks. It also includes a Membership Functions sub-library that contains Simulink blocks for the built-in membership functions. The Fuzzy Logic Controller with Rule Viewer block is an extension of the Fuzzy Logic Controller block. It allows you to visualize how rules are fired during simulation. To start building a Simulink Fuzzy Model, drag the Fuzzy Logic Controller block (with or without the Rule Viewer) from the Simulink library to the simulation window (This can also be done by typing fuzblock at the MATLAB prompt). To initiate the Fuzzy Logic Controller block, double-click on the block and enter the name of the structure variable describing your FIS. This variable must be located in the MATLAB workspace. In most cases, the Fuzzy Logic Controller block automatically generates a hierarchical block diagram representation of your FIS. The block diagram representation only uses built-in Simulink blocks. This automatic, model-generation ability in called the Fuzzy Wizard. In cases where Fuzzy Wizard cannot handle FIS, the Fuzzy Logic Controller block uses the S-function sffis to simulate the FIS.

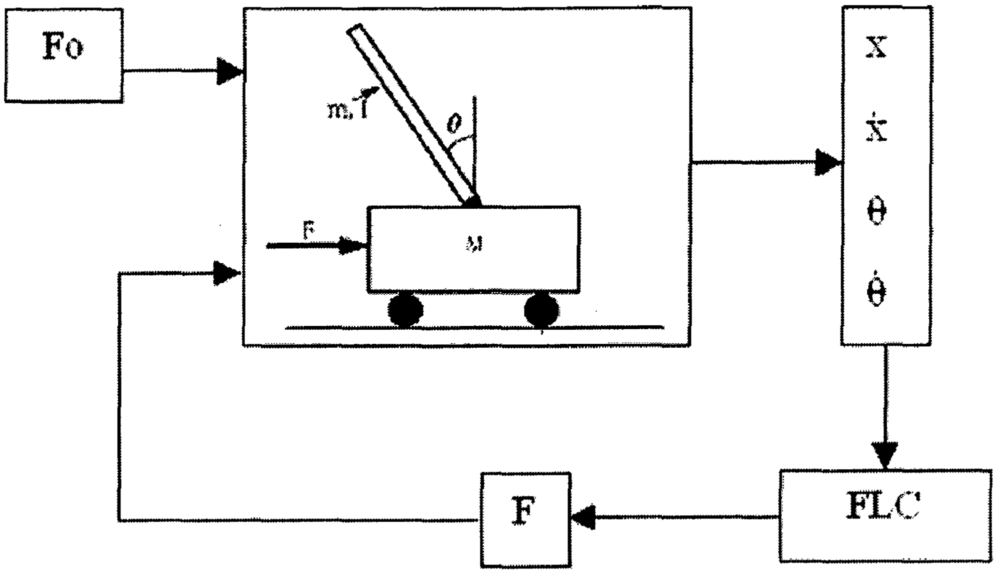


Fig 5.1 BLOCK DIAGRAM OF FLC FOR INVERTED PENDULUM

5.3 SIMULATION IN MATLAB

The block diagram of inverted pendulum is shown in figure 5.1 and the design of inverted pendulum in simulink is shown in figure 5.2.

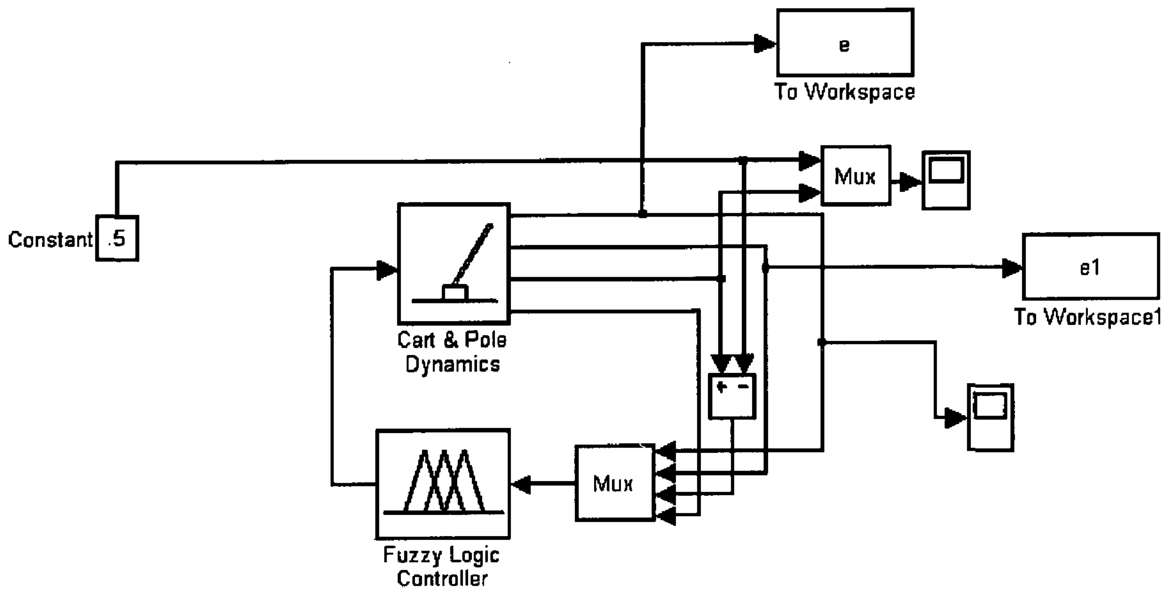


Figure 5.2 the Inverted Pendulum Model in Simulink

5.4 BUILDING UP THE MAMDANI FUZZY CONTROLLER

Triangular membership functions are used for both input and outputs [7]. All membership functions of input variables are (1) Pendulum Angle (2) Cart Position (3) Cart Velocity (4) Angular Velocity of Pendulum shown in Figure 5.3 to Figure 5.6 and output membership function is Control Force shown in Figure 5.7.

5.4.1 PENDULUM ANGLE

The controller angle is to maintain the zero. It means that the controller will keep the pendulum vertically standing on the cart. When the angle is negative, the pendulum is on the left side of the centre line. When the angle is positive, the pendulum is on the right side of the centre line.

Two linguistic variable 'negative' and 'positive' are used for inputs. The membership function for the Pendulum Angle is shown in the Figure 5.3. Consider Range of the angle is [-0.3 0.3].

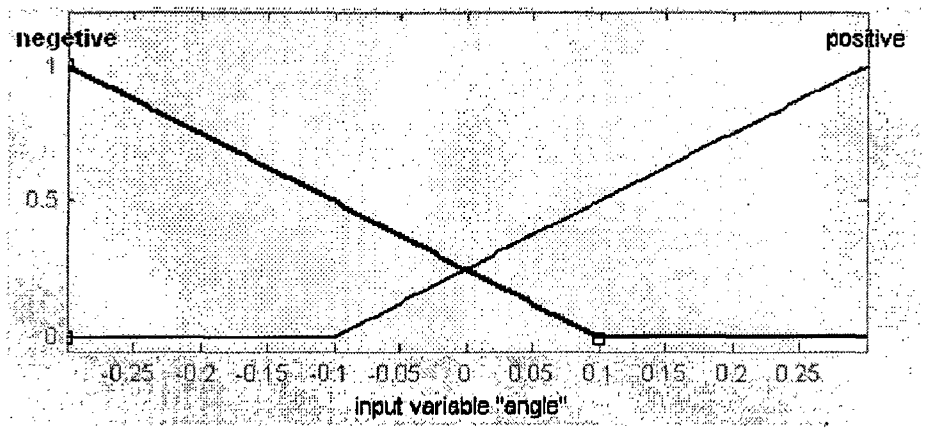


Figure 5.3 The Membership Function of angle

5.4.2 CART POSITION:

The cart is moving on the horizontal axis. When the cart position is negative, the cart is on the left side of the centre on the track. When the cart position is positive, the cart is on the right side of the centre on the track.

Two linguistic variable 'negative' and 'positive' are used for inputs. The membership function for the Cart Position is shown in the Figure 5.4. Consider Range of the cart position is [-1 1].

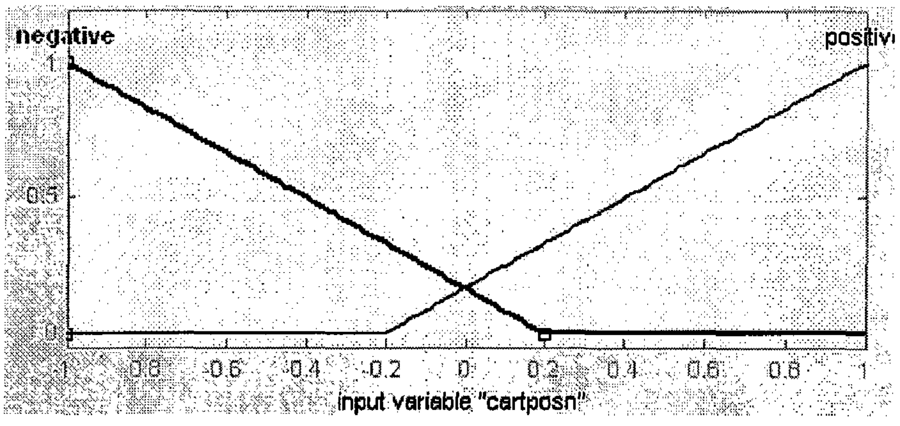


Figure 5.4 The Membership Function of Cart Position

5.4.3 CART VELOCITY

The cart velocity of the inverted pendulum, when the cart velocity is negative, the cart is moving toward the left end of the track. When the cart velocity is positive the cart is moving toward the right end of the track.

Two linguistic variable 'negative' and 'positive' are used for inputs. The membership function for the Cart Velocity is shown in the Figure 5.5. Consider Range of the cart velocity is [-3 3].

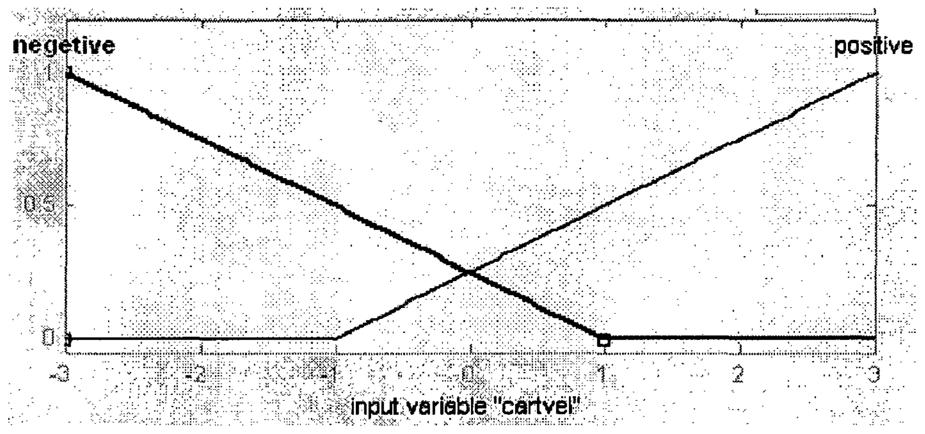


Figure 5.5 The Membership Function of Cart Velocity

5.4.4 PENDULUM ANGULAR VELOCITY

The angle velocity for the Inverted Pendulum, when the angle velocity is negative, the pendulum is moving toward the left side. When the angle velocity is positive the pendulum is moving toward the right side.

Two linguistic variable 'negative' and 'positive' are used for inputs. The membership function for the Pendulum Angle Velocity is shown in the Figure 5.6. Consider Range of the angular velocity is [3, 3]

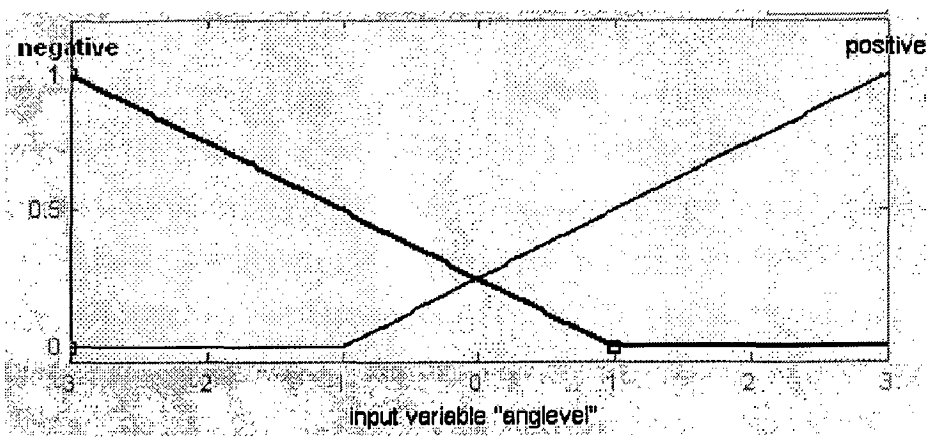


Figure 5.6 The Membership Function of Angular Velocity

5.4.5 CONTROL FORCE

The Control Force is only one output for the controller. I have consider 16 (Sixteen) linguistic variables name mfl to mfl6 , Triangular membership function is used and range for the output is [-55 55]. Later we will see the Rule base for this FLC , there are sixteen rules are used

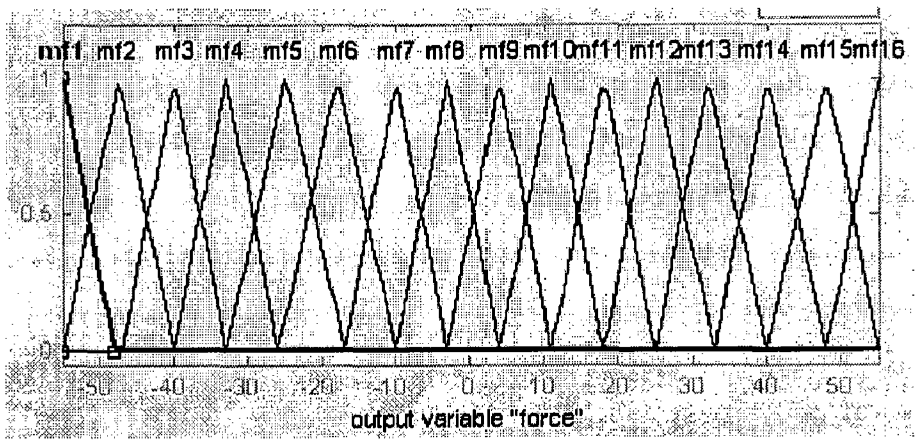


Figure 5.7 The Membership Function of Output 'Force'

5.5 RULE BASE

Rule base for the fuzzy logic controller. Here 16(Sixteen) rules are used The rule-list matrix where as follow.

IF the angle is negative and cartposn is negative and cartvel is negative and anglevel is negative, THEN output is mf1

IF the angle is negative and cartposn is negative and cartvel is negative and anglevel is positive, THEN output is mf2.

IF the angle is negative and cartposn is negative and cartvel is positive and anglevel is negative, THEN output is mf3.

IF the angle is negative and cartposn is negative and cartvel is positive and anglevel is positive, THEN output is mf4.

IF the angle is negative and cartposn is positive and cartvel is negative and anglevel is negative, THEN output is mf5.

IF the angle is negative and cartposn is positive and cartvel is negative and anglevel is positive, THEN output is mf6.

IF the angle is negative and cartposn is positive and cartvel is positive and anglevel is negative, THEN output is mf7.

IF the angle is negative and cartposn is positive and cartvel is positive and anglevel is positive, THEN output is mf8.

IF the angle is positive and cartposn is negative and cartvel is negative and anglevel is negative, THEN output is mf9.

IF the angle is positive and cartposn is negative and cartvel is negative and anglevel is positive, THEN output is mf10.

IF the angle is positive and cartposn is negative and cartvel is positive and anglevel is negative, THEN output is mf11.

IF the angle is positive and cartposn is negative and cartvel is positive and anglevel is positive, THEN output is mf12.

IF the angle is positive and cartposn is positive and cartvel is negative and anglevel is negative, THEN output is mf13.

IF the angle is positive and cartposn is positive and cartvel is negative and anglevel is positive, THEN output is mf14.

IF the angle is positive and cartposn is positive and cartvel is positive and anglevel is negative, THEN output is mf15.

IF the angle is positive and cartposn is positive and cartvel is positive and anglevel is positive, THEN output is mf16.

[Rules]

1 1 1 1, 1 (1): 1

1 1 1 2, 2 (1) : 1

```
1 1 2 1, 3 (1) : 1
1 1 2 2, 4 (1) : 1
1 2 1 1, 5 (1) : 1
1 2 1 2, 6 (1) : 1
1 2 2 1, 7 (1) : 1
1 2 2 2, 8 (1) : 1
2 1 1 1, 9 (1) : 1
2 1 1 2, 10 (1) : 1
2 1 2 1, 11 (1) : 1
2 1 2 2, 12 (1) : 1
2 2 1 1, 13 (1) : 1
2 2 1 2, 14 (1) : 1
2 2 2 1, 15 (1) : 1
2 2 2 2, 16 (1) : 1
```

5.6 RESPONSE OF THE SYSTEM

Up to this we build the mamdani type Fuzzy Inference System for the inverted pendulum in the FIS editor and save as the vee.fis with considering four inputs and one output. Simulation is run for the 10 second and taken response of angle of the inverted pendulum shown as following figure 5.8.

Next chapter we will see the how genetic algorithm help to optimize the fuzzy logic controller parameters.

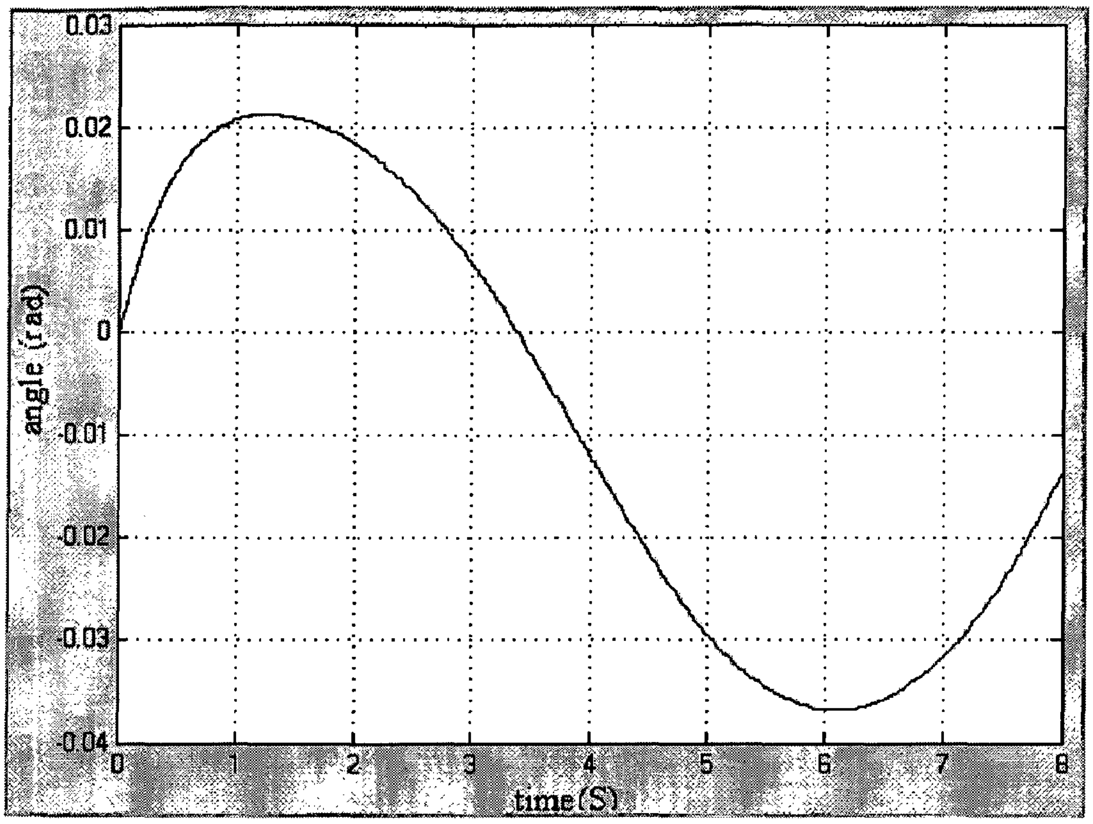


Figure 5.8 Response of System (Angle)

CHAPTER 6

OPTIMIZATION OF FLC USING GA

6.1 STATEMENT OF THE PROBLEM

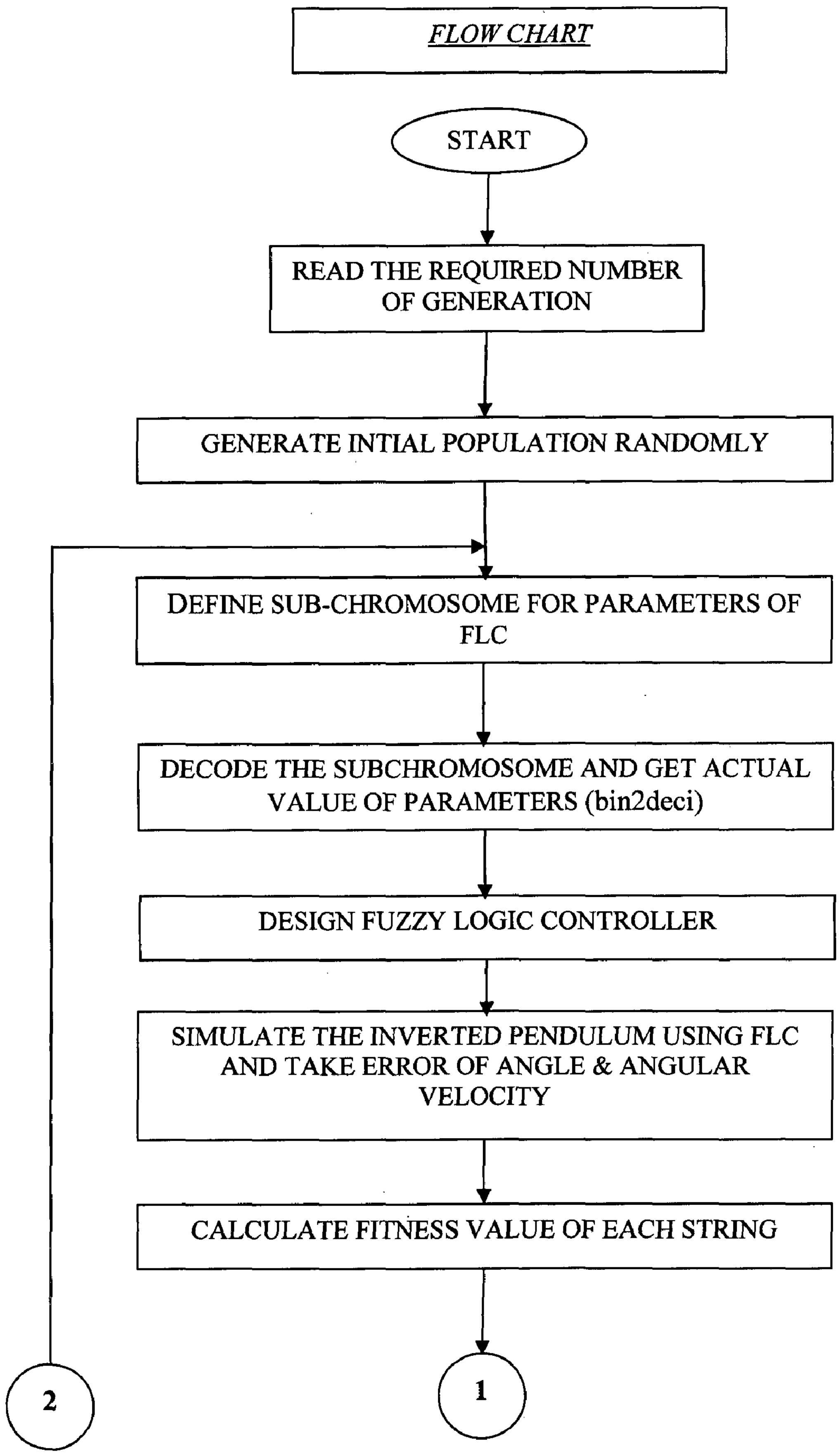
Consider that we have fuzzy logic with four input variable and one output variable. Our goal is to optimize the fuzzy logic parameters such as centre of triangular membership function [10] and weight of the rule to get optimum result.

In this work, triangular membership functions are used and two linguistic variable for each inputs. The triangular membership has three parameters a, b, & c, here we were change only one parameter. Thus for the four input each have two MFs resulting 8 parameters to change. In rule base there are 16 rules, which require 16 weight parameters.

So our main objective is to tune the parameters of fuzzy logic controller by Genetic Algorithm. The incorporation of genetic algorithm into a fuzzy design process adds an 'intelligent' dimension to the fuzzy controller enabling it to create and modify its rules. Genetic algorithms give the possibility of adjusting membership functions down to the level of individual rules [18,7].

6.2 IMPLEMENTATION

Our goal is to optimize the fuzzy logic controller by tuning the membership function [6,8] of the input variable as the output variable parameters are fixed and each of the input variables quantified into two MFs, 24 parameters of the FLC are needed to be tuned by Genetic Algorithm. Functional block diagram is shown in fig 6.2 and the whole work is divided into eight steps can be shown in flow chart.


FLOW CHART
START
READ THE REQUIRED NUMBER OF GENERATION
GENERATE INTIAL POPULATION RANDOMLY
DEFINE SUB-CHROMOSOME FOR PARAMETERS OF FLC
DECODE THE SUBCHROMOSOME AND GET ACTUAL VALUE OF PARAMETERS (bin2deci)
DESIGN FUZZY LOGIC CONTROLLER
SIMULATE THE INVERTED PENDULUM USING FLC AND TAKE ERROR OF ANGLE & ANGULAR VELOCITY
CALCULATE FITNESS VALUE OF EACH STRING
2
1

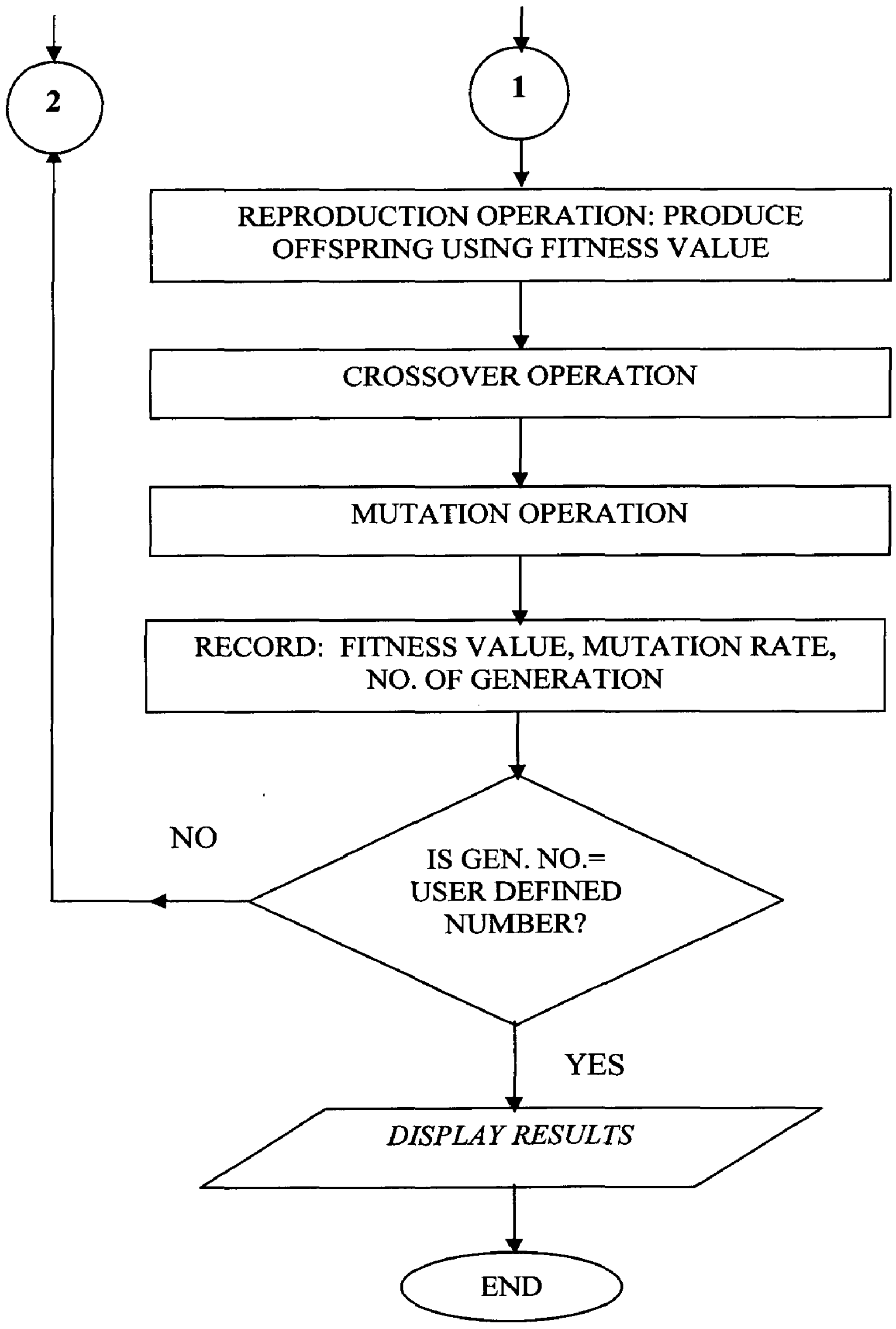
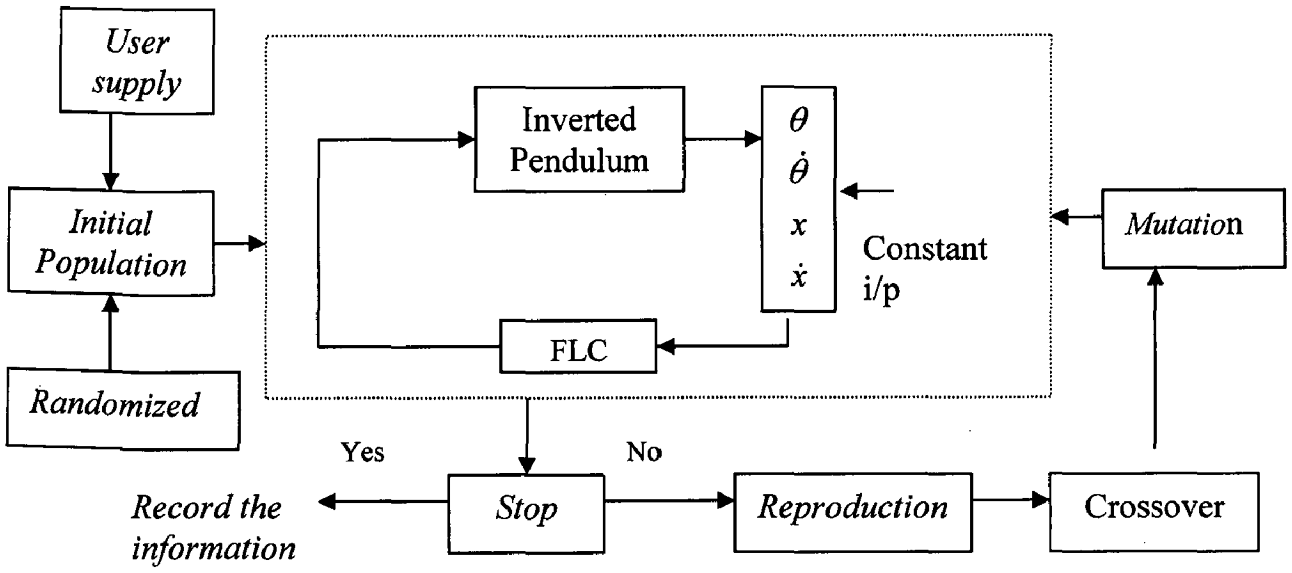

2
1
REPRODUCTION OPERATION: PRODUCE OFFSPRING USING FITNESS VALUE
CROSSOVER OPERATION
MUTATION OPERATION
RECORD: FITNESS VALUE, MUTATION RATE, NO. OF GENERATION
NO
IS GEN. NO.= USER DEFINED NUMBER?
YES
DISPLAY RESULTS
END

Fig 6.1 Flow Chart for combining both Fuzzy and GA

Major Steps in Combining both Fuzzy and Genetic Algorithms

1. Initialization of population.
2. Coding of FLC parameters.
3. FLC design.
4. Calculation of fitness value.
5. Reproduction using Roulette Wheel.
6. Crossover.
7. Mutation.
8. Record and display of the results.



6.2 FUNCTIONAL BLOCK DIAGRAM OF GA OPTIMISING PROCESS

6.2.1. INTILIZATION OF POPULATION

Initial population is the random population, here 8 MFs each have 5 bits thus for the all inputs allotted 40 bits. 16 rules, each rule have allotted 3 bits thus for rules have 48 bits. So length of the string 40 + 48 = 88 bits strings with 0 and 1 bit is required and considering initial population is 10 which is generated by the *randsrc* command with dimension 88 by 10 matrices with 0 and 1 bits with probability of 0.5.

6.2.2. CODING OF FLC PARAMETERS

The Genetic Algorithm deals with the coded parameters, 8 parameters of the FLC that need to be tuned to get the optimize results, must be encoded into a finite length of string. The linear mapping method is used for this purpose that can be expressed as follows [4].

$$Va = Vmin + (Vmax-Vmin)*Vb / (2^{N}-1)$$

Where Va is the actual value of the parameter, and Vb is the integer represented by a N-bit string gene. Vmax is Vmin are user defined upper and lower limits of the gene respectively. The encoded genes are concatenated to form a complete chromosome.

It can observed that Gene 1 to 2 are allotted to the sub chromosome of the first controller input , each way 3 to 4 , 5 to 6 and 7 to 8 are respectively for the remain three inputs, 9 to 16 gene are allocated to the 16 rules. The coded parameters of the FLC are arranged as shown in the following.

BITS	: | 1......10|	|11......20|	|21......30|	|31.......40|	|41..............88|
	Input-1	Input-2	Input-3	Input-4	16 Rules
GENE	: |1|2|	|3|4|	|5|6|	|7|8|	|9|10|........|16|

6.2.3 CALCULATION OF FITNESS VALUE

The task of defining the fitness is usually application specific. Our main objective is to keep upright the Inverted Pendulum means pole remain vertical position during the cart moving along the horizontal axis. Initial value of the pole is 0 rad, so consider it is desired value.

In this project the value(data) for two output of the system during the simulation is taken, (1) Angle(e) and (2) Angular velocity(e_1). Because considering all four inputs the problem becomes the Multi Objective Genetic Algorithm, it's required so much time.

So the fitness function is as follows.

$$f = e2 + e_1^2$$

6.2.4. REPRODUCTION

The *roulette wheel* selection is used to reproduce new strings in the new generation. Following MATLAB command is used for that purpose.

RW = *randscr* (1,1,[1 2 3 4 5 6 7 8 9 10; f1 f2 f3 f4 f5 f6 f7 f8 f9 f10]);

Where RW is roulette wheel output, f1,f2...f10 is fitness value of strings. The randscr command randomly generates one digit between 1 and 10. The fitness value is the probability of generating particular digit.

Thus the string with the higher fitness value has a higher probability of contributing offspring in the next generation.

6.2.5 CROSSOVER.

Crossover in GA occurs when the selected chromosomes partially exchange the information in the gene, i.e., part of a string is interchanged between two selected candidates. The probability of crossover is selected as 70%. If the crossover occurs than it is required to find the mate in mating pool and cross-site.

The *rand perm* command is used to select one of the strings from first five strings randomly. The selected string and sixth string does crossover after selecting cross-site randomly. For selection of cross-site *randsrc* command is used. These steps are repeated for five times. After selecting cross-site *flipud* command is used to does the crossover.

6.2.6 MUTATION

Dynamic mutation rate is used in the GA operation as it provides faster convergence. To prevent from immature convergence and high mutation rate, step wise dynamic mutation is used. That is given as follows.

(1) For generation 1 to 5 numbers of mutation = 0

(2) For generation 6 to 15 numbers of Mutation = 0.015*88*10*(1/fitness value)

(3) For generation 16 to 35 numbers of Mutation = 0.02*88*10*(1/fitness value)

(4) For generation 36 to 50 numbers of Mutation = 0.03*88*10*(1/fitness value)

6.2.7 RECORD AND DISPLAY OF THE RESULTS

All the new generation information is recorded for analysis of results. The graph showing the change the position of the MFs parameters, pendulum angle position after optimization, the variation in the fitness value and mutation rate is generated at the end of GA operation.

The following figure is shown the membership function of the variable for all inputs after applying the Genetic algorithms. Figure 6.3 to 6.6 are for the inputs and 6.7 is for the response of the system. Figure 6.8 shows the response of the system with parameter variation and figure 6.9 shows the response of the system with fuzzy and fuzzy combined with GA and finally Figure 6.10 shows the plot of fitness value & mutation rate Vs generation.

6.2.7.1 PENDULUM ANGLE

After applying the GA, the Membership function is tuned and change its position for all four inputs, here shows for Input1 (angle)

Range=[-0.3 0.3]

MF1='negetive':'trimf',[-0.3 -0.3 **0.0929**]

MF2='positive':'trimf',[**-0.08** 0.3 0.3]

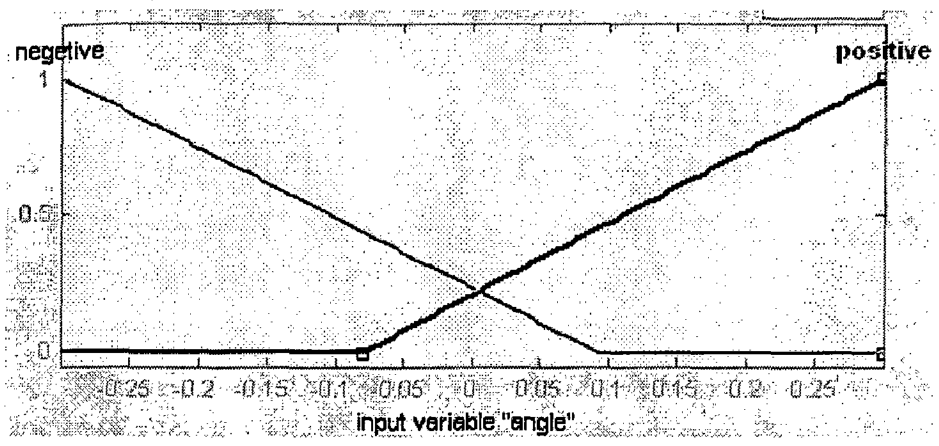


Figure 6.3 Membership Function of angle after Optimization

6.2.7.2 CART POSITION

Range=[-1 1]

MF1='negative':'trimf',[-1 -1 **0.260645161290323**]

MF2='positive':'trimf',[**-0.22** 1 1]

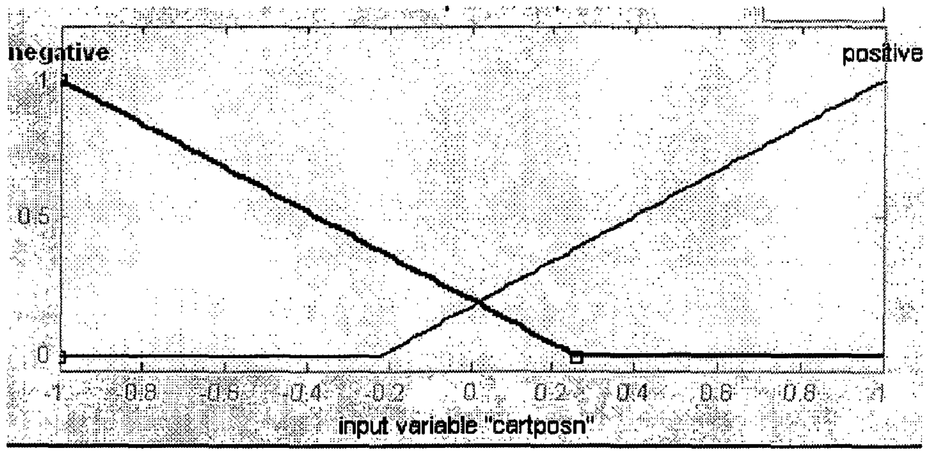


Figure 6.4 The Membership Function of Cart Position after Optimization

6.2.7.3 CART VELOCITY

Range=[-3 3]

MF1='negetive':'trimf',[-3 -3 **1.01935483870968**]

MF2='positive':'trimf',[**-0.941935483870968** 3 3]

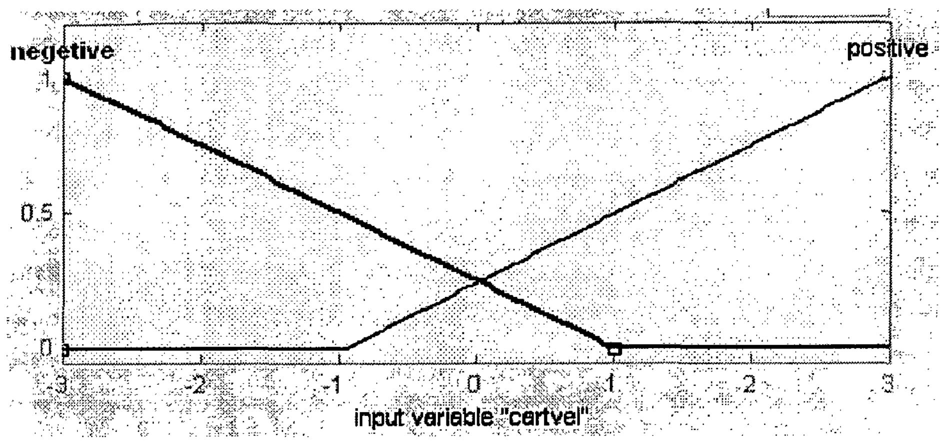


Figure 6.5 The Membership Function of Cart Velocity after Optimization

6.2.7.4 PENDULUM ANGULAR VELOCITY

Range=[-3 3]

MF1='negative':'trimf',[-3 -3 **1.13548387096774**]

MF2='positive':'trimf',[**-1.17419354838871** 3 3]

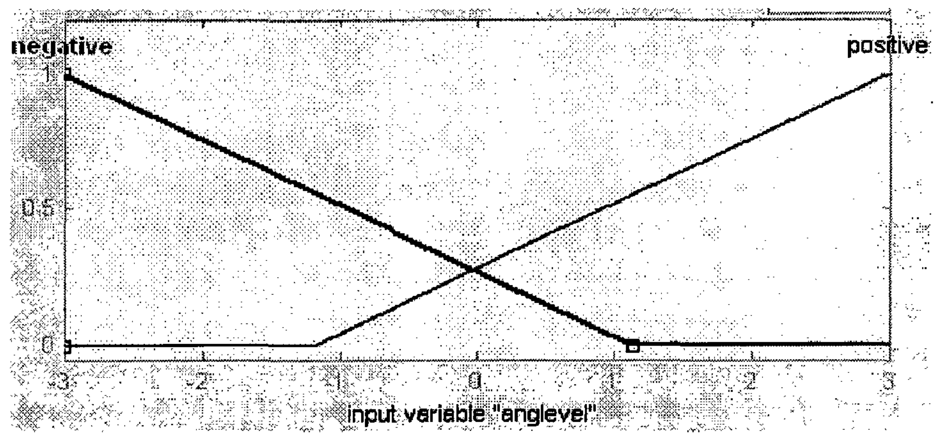


Figure 6.6 The Membership Function of Angular Velocity after Optimization

6.2.7.5 Rule List after Optimization of FLC

1 1 1 1, 1 (**0.928571428571429**) : 1

1 1 1 2, 2 (**0.857142857142857**) : 1

1 1 2 1, 3 (**0.5**) : 1

1 1 2 2, 4 (**0.642857142857143**) : 1

1 2 1 1, 5 (**1**) : 1

1 2 1 2, 6 (**0.8571428571428571**) : 1

1 2 2 1, 7 (**0.928571428571429**) : 1

1 2 2 2, 8 (**0.928571428571429**) : 1

2 1 1 1, 9 (**0.785714285714286**) : 1

2 1 1 2, 10 (**0.571428571428571**) : 1

2 1 2 1, 11 (**0.642857142857143**) : 1

2 1 2 2, 12 (**1**) : 1

2 2 1 1, 13 (**0.928571428571429**) : 1

2 2 1 2, 14 (**0.928571428571429**) : 1

2 2 2 1, 15 (**0.5**) : 1

2 2 2 2, 16 (**0.785714285714286**) : 1

6.2.7.6 RESPONSE OF THE SYSTEM

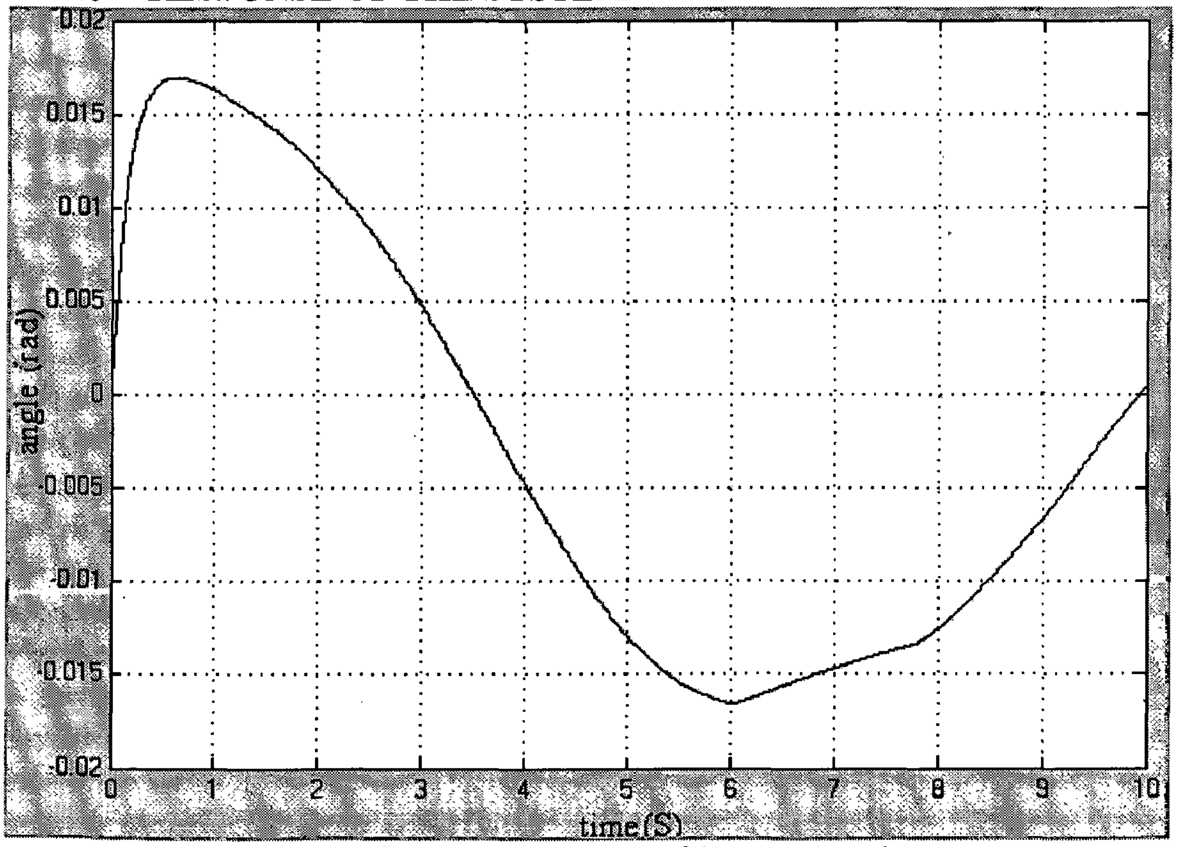


Figure 6.7 Response of System (Angle)

The response of the system with Parameter variation is shown in Fig 6.8 and Comparison of both Fuzzy and Fuzzy combined with Genetic Algorithms is shown in Fig 6.9

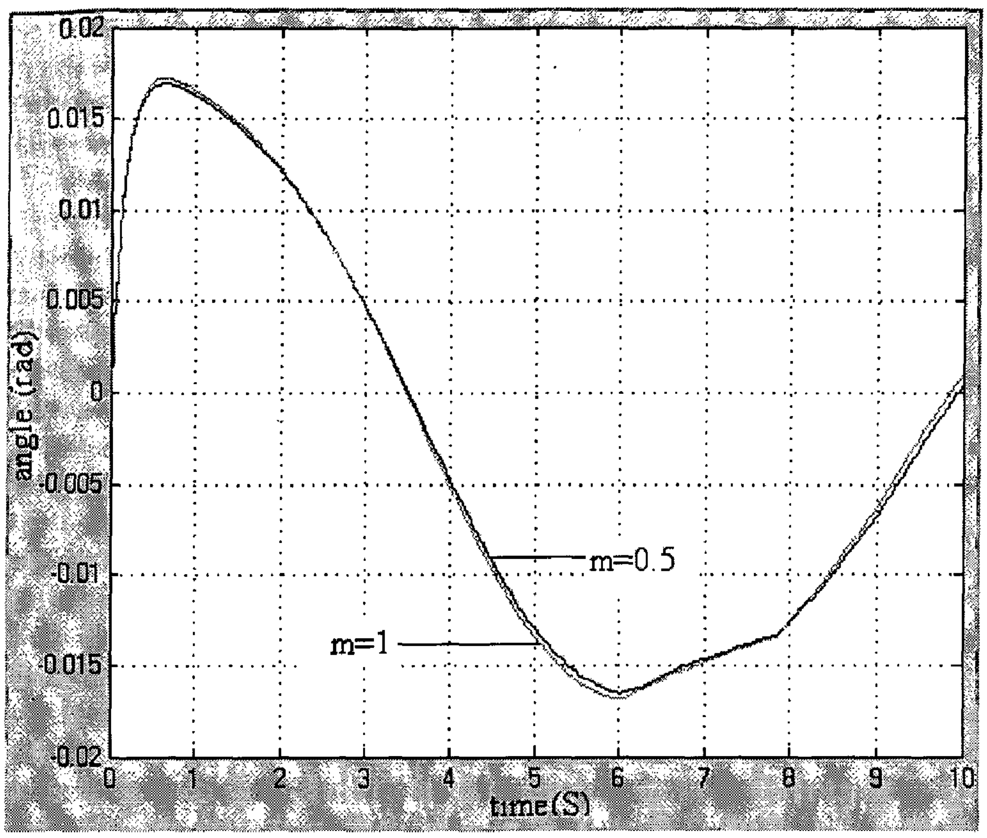


Fig 6.8 System response with parameter variation
Pole mass=.5 kg,1 kg

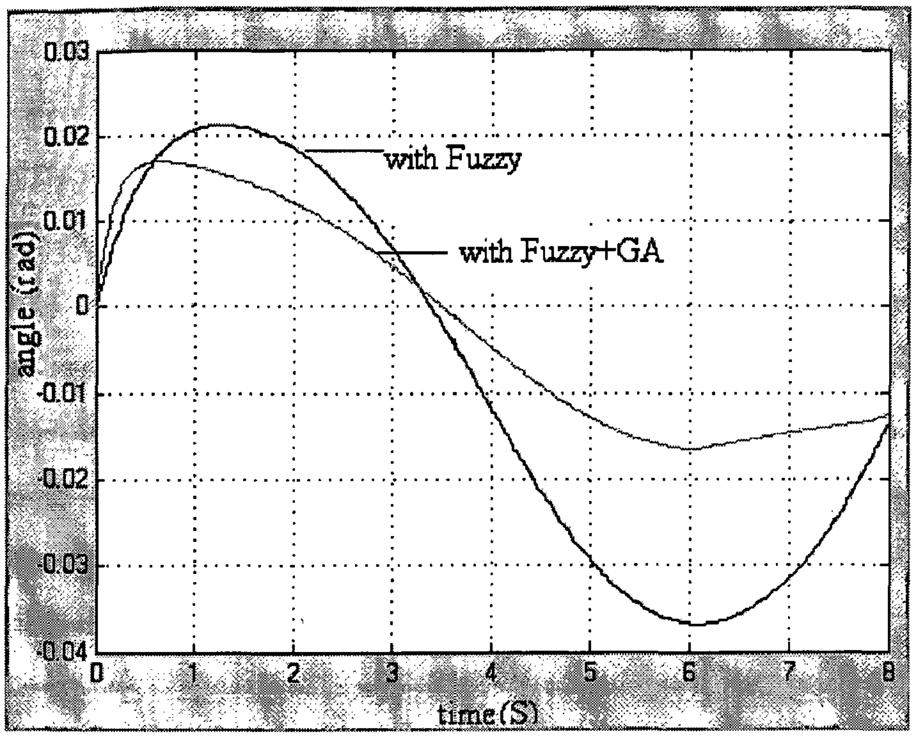


Fig 6.9 Response of system with fuzzy and fuzzy+Ga

6.2.7.7. TABLE FOR FITNESS VALUE & MUTATION RATE

SR NO.	*GENERATION*	*FITNESS VALUE*	*MUTATION RATE*
1.	0	1.6301	0
2.	1	1.3478	0
3.	2	1.5529	0
4.	3	1.6820	0
5.	4	2.8109	0
6.	5	3.0183	4.3733
7.	6	2.7634	4.7768
8.	7	2.6835	4.9189
9.	8	2.7104	4.8702
10.	9	2.7723	4.7614
11.	10	2.7676	4.7696
12.	11	2.7892	4.7325

13.	12	3.0000	4.4000
14.	13	2.8480	4.6349
15.	14	3.0190	4.3724
16.	15	2.8134	6.2558
17.	16	2.3357	7.5353
18.	17	2.2985	7.6571
19.	18	2.8468	6.1824
20.	19	3.1739	5.5452
21.	20	3.1640	5.5626
22.	21	3.2419	5.4289
23.	22	3.3246	5.2939
24.	23	1.9317	9.1112
25.	24	3.3579	5.2414
26.	25	3.4933	6.2977
27.	26	3.5797	6.1458
28.	27	3.2789	6.7095
29.	28	3.4999	6.2858
30.	29	3.5744	6.1548
31.	30	3.5949	6.1197
32.	31	3.5252	6.2407
33.	32	3.4293	6.4154
34.	33	3.0430	7.2297
35.	34	4.6315	4.7501
36.	35	5.9241	4.4564
37.	36	6.0568	4.3588
38.	37	4.8241	5.4725
39.	38	4.2622	6.1940
40.	39	4.4746	5.9000
41.	40	4.9343	5.3502

42.	**41**	**5.2650**	**5.0142**
43.	**42**	**4.8964**	**5.3917**
44.	**43**	**3.8351**	**6.8839**
45.	**44**	**4.0455**	**6.5257**
46.	**45**	**5.1748**	**5.1016**
47.	**46**	**4.8283**	**5.4678**
48.	**47**	**5.3564**	**4.9287**
49.	**48**	**5.3728**	**4.9136**
50.	**49**	**5.5836**	**4.7281**
51.	**50**	**5.5949**	**4.7612**

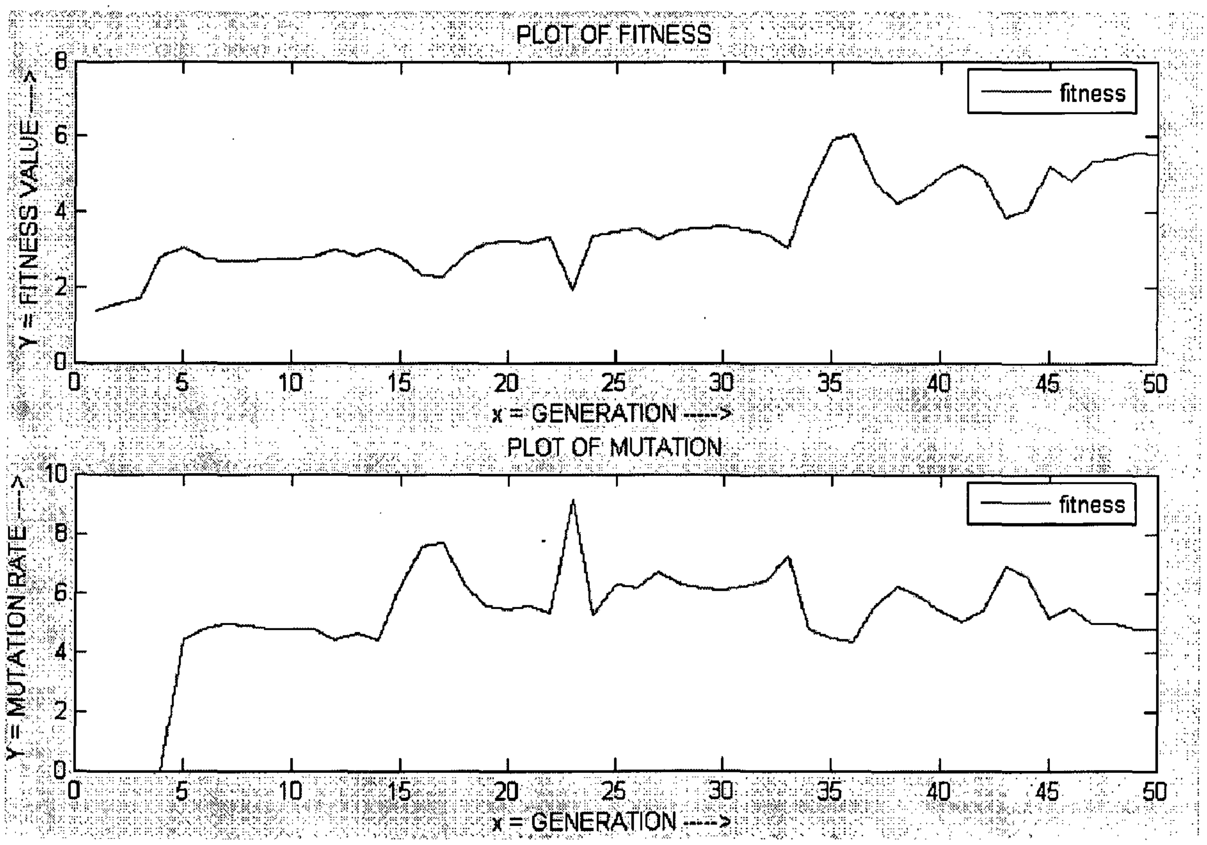


Figure 6.10 The Plot of Generation Vs Fitness & Mutation

The objective values for the maximum fitness and mutation rate per iteration are shown in above figure. Here, 50 generation is used. we can see the fitness value is increased or decreased as no. of generation increase but at last got highest fitness value.

Selection criteria are used: - Roulette Wheel selection. Plots for Fitness Vs. Generation for Roulette Wheel Selection and Number of bits Mutate in every generation.

CONCLUSIONS

In this project first design of Fuzzy logic controller for inverted pendulum is carried out and it is showed that the design of FLC is easy to work with and to adapt, rules can always be deleted, modified ,added and they can provide more effective control of non-linear systems than a linear controller as there is more flexibility in designing the mapping from the input to the output the only difficulties are in the designing the rule base, that is to decide weights of the rules and ranges of parameters. The incorporation of genetic algorithm into a fuzzy design process adds an intelligent dimension to the fuzzy controller enabling it to create and modify its rules.

It has been shown that the use of genetic algorithms offers a feasible method for the optimization of the knowledge-base of fuzzy logic controllers. The simulation results proved that the proposed method based on GA is an effective way to tune the parameters such as centre of membership functions, weight of rules of fuzzy logic controller for inverted pendulum problem.

7.1 FUTURE SCOPE OF WORK

For this project, a broad investigation into applying Genetic Algorithms to Fuzzy Logic Controllers was carried out.
Suggestions for follow-up work that may come after this project are:

1. Perform a more in-depth study of how the various parameters and encodings affect GA performance in FLC optimization. This work require enough time for more investigations of this kind.

2. Investigate more thoroughly if GAs can be applied to FLCs or other types of nonlinear controllers, like Neural network so that they may successfully control fully the inverted pendulum system.

3. Investigate whether any of the assumptions made in designing FLCs such as using only triangular membership functions and consider two MFs for each input allowing an more number of these sets have a significant impact on the performance obtainable from the controllers.

Genetic algorithms combined with other intelligent techniques, such as neural networks, expert systems and fuzzy logic control systems open a new way to design and construct intelligence control systems adapted to complex processes.

REFERENCES

1. D.Driankov, H.Hellendorn and M.Reinfrank, **'An Introduction to Fuzzy Control'**. NewYork: Springer-Verlag, 1993.
2. Passino, Kevin M. and Yurkovich, Stephen, **'Fuzzy Control'** Addison-Wesley 1998.
3. Goldberg, D.E. ,**'Genetic algorithms in search, optimization and machine learning'** Addison-Wesley, 1989
4. N.P.Padhy, '**Artificial Intelligence and Intelligent Systems**' (Oxford University Press, 2005).
5. Linkens, D.A., Nyongesa, H.O., **'Genetic Algorithms for Fuzzy Control. Part 1: Offline System Development and Application'** IEEE Proc.- Control Theory Appl., Vol. 142 No.3 May 1995 pp161-176
6. Herrera, F., Lozano, M., Verdegay, J. L., **'Tuning Fuzzy Logic Controllers by Genetic Algorithms'**, International Journal of Approximate Reasoning 1995; 12:299-315
7. Lee, M.A., Takagi, **' Integrating Design Stages of Fuzzy Systems Using Genetic Algorithms'**, Proc.2nd IEEE Int. Conf. Fuzzy Systems, SanFrancisco, 1993; 612-617
8. Renhou & Z. Yi, **'Fuzzy logic controller based on genetic algorithm**', Fuzzy Sets and Systems, v. 83, pp1-10, 1996.
9. Alen Varsek, Tanja Urbancic and Bodgan Filipic,**'Genetic Algorithms in Controller Design and Tuning'**, IEEE Transaction on Systems, Man and Cybernetics. Vol.23, No.5 October 1993.
10. KARR, C.L.: '**Design of an adaptive fuzzy logic controller using a genetic algorithm**', in SCHAFFER, I. (FA.): Proceedings of the third international conference on Genetic algorithms (Morgan Kauffman Publishers, Cambridge, MA, 1989), pp. 450-457

11. M.A. Lee,H.Takagi, **'Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques'**, proceedings of the 5th International Conference on Genetic Algorithms pp 76-83(1993)
12. L.A.Zadeh, **'Fuzzy sets'** information control vol.-8, pp.338-353, 1965.
13. S.Assilian and E.H.Mamdani, **'An Experiment in Linguistic Synthesis with Fuzzy Logic Controller'**, International Journal on Man Machine studies. Vol.7 pp 1-13 1975.
14. R.Palm,**'Scaling of Fuzzy Controller using the cross correlation'**, IEEE Transaction on Fuzzy System, Vol.3, pp.116-123, Feb 1995.
15. Rajini K. Mudi ,Nikhil R.Pal,**'A Robust self Tuning scheme for PI and PD type Fuzzy controller'**, IEEE Transaction on Fuzzy systems, Vol. 7 ,February 1999.
16. K.Belarbi, F.Titel, W. Bourbia,K. Ben mahammed,**'Design of Mamdani Fuzzy Logic Controllers with Rule base minimization using Genetic Algorithm'**, Engineering Application of Artificial Intelligence, March 2005.
17. Bandyopadhyay, R. Chakraborty, U.K. Patranabis.D, **'Auto Tuning a PID Controller: A Fuzzy –Genetic Approach'**, Journal of System Architecture 47(2001).pp 663-673.
18. P.Wang and D.P. Kwok.,**'Optimal Fuzzy PID Control based on genetic algorithms'** Industrial Electronics, Control Instrumentation &Automation 1992.Proceedings of the International Conf .Vol.2.pp 977-981.
19. Gregory V.TAN & Xiheng HU **'On designing Fuzzy controllers using Genetic algorithms'**- Fuzzy Systems 1996 ,Proceedings of fifth International Conf.,pp 905-911.
20. O.Cordon, F.Gomide, F.Herrera, F.Hoffmann, L.Magdalena,**'Ten years of Genetic Fuzzy Systems: Current Framework and New Trends'**, Fuzzy Sets and Systems 141 (2004), pp 5-31.
21. **'Fuzzy logic Toolbox User's Guide'**, Version 2,www.mathworks.com
22. www.engin.umich.edu/group/ctm/examples/pend/invpen.html. CTM inverted pendulum Modeling.

Appendix

```
% FUZZY  CONTROL FOR INVERTED PENDULUM using GA
% this program uses randbits.m(to produce the random matrics with 0 & 1
element)
% bin2deci.m(convert parameters value binary to decimal value)

%   INITIALIZE POPULATION:-
IP=IG1;
user_entry=input('Enter no. generations range[1 50]=');
G=user_entry;
%  1.INITIALIZATION OF POPULATION:-
%  =============================
g=1;
while g<=G+1;
p=1;
q=10;
while p<=q

%   CODING (FLC PARAMETERS)

%  STIRINGS ARE SELECT FROM THE INITIAL POPULATION

s1 = [IP(1,1:88)];
s2 = [IP(2,1:88)];
s3 = [IP(3,1:88)];
s4 = [IP(4,1:88)];
s5 = [IP(5,1:88)];
s6 = [IP(6,1:88)];
s7 = [IP(7,1:88)];
s8 = [IP(8,1:88)];
s9 = [IP(9,1:88)];
s10 =[IP(10,1:88)];

% Substrings for the MFs of Input(angle)

C11= [IP(p,1:5)];
A12= [IP(q,6:10)];

% Substring for the MFs of Input(cartposn)

C21= [IP(p,11:15)];
A22= [IP(p,16:20)];

%  Substrings for the MFs of Input(cartvel)

C31= [IP(p,21:25)];
A32= [IP(p,26:30)];

% Substring for the MFs of Input(anglevel)

C41= [IP(p,31:35)];
```

```
A42= [IP(p,36:40)];

% Substrings for the Rule Weights
W1 =[IP(p,41:43)];
W2 =[IP(p,44:46)];
W3 =[IP(p,47:49)];
W4 =[IP(p,50:52)];
W5 =[IP(p,53:55)];
W6 =[IP(p,56:58)];
W7 =[IP(p,59:61)];
W8 =[IP(p,62:64)];
W9 =[IP(p,65:67)];
W10 =[IP(p,68:70)];
W11 =[IP(p,71:73)];
W12 =[IP(p,74:76)];
W13 =[IP(p,77:79)];
W14 =[IP(p,80:82)];
W15 =[IP(p,83:85)];
W16 =[IP(p,86:88)];

% DECODING THE PARAMETERS OF MFs VALUE FROM BINARY TO DECIMAL
% FOR A

a12 =bin2deci(A12,5,-0.12,-0.08);
a22 =bin2deci(A22,5,-.28,-.22);
a32 =bin2deci(A32,5,-1.2,-0.8);
a42 =bin2deci(A42,5,-1.2,-0.8);

% FOR C
c11 =bin2deci(C11,5,0.08,0.12);
c21 =bin2deci(C21,5,0.23,0.28);
c31 =bin2deci(C31,5,0.8,1.2);
c41 =bin2deci(C41,5,0.8,1.2);

% FOR RULE WEIGHTS
w1 =bin2deci(W1,3,0.5,1);
w2 =bin2deci(W2,3,0.5,1);
w3 =bin2deci(W3,3,0.5,1);
w4 =bin2deci(W4,3,0.5,1);
w5 =bin2deci(W5,3,0.5,1);
w6 =bin2deci(W6,3,0.5,1);
w7 =bin2deci(W7,3,0.5,1);
w8 =bin2deci(W8,3,0.5,1);
w9 =bin2deci(W9,3,0.5,1);
w10 =bin2deci(W10,3,0.5,1);
w11 =bin2deci(W11,3,0.5,1);
w12 =bin2deci(W12,3,0.5,1);
w13 =bin2deci(W13,3,0.5,1);
w14 =bin2deci(W14,3,0.5,1);
w15 =bin2deci(W15,3,0.5,1);
w16 =bin2deci(W16,3,0.5,1);

% FLC DESIGNING :-

s= newfis('vee','mamdani','prod','max','prod','max','centroid');
```

```
% Input angle MFs
s =addvar(s,'input','angle',[-0.3 0.3]);
s =addmf(s,'input',1,'negetive','trimf',[-0.3 -0.3 c11]);
s =addmf(s,'input',1,'positive','trimf',[a12 0.3 0.3]);
% Input cart_position MFs
s =addvar(s,'input','cartposn',[-1 1]);
s =addmf(s,'input',2,'negative','trimf',[-1 -1 c21]);
s =addmf(s,'input',2,'positive','trimf',[a22 1 1]);

% Input cart_velocity MFs
s =addvar(s,'input','cartvel',[-3 3]);
s =addmf(s,'input',3,'negetive','trimf',[-3 -3 c31]);
s =addmf(s,'input',3,'positive','trimf',[a32 3 3]);

% Input angular_velocity MFs
s =addvar(s,'input','anglevel',[-3 3]);
s =addmf(s,'input',4,'negative','trimf',[-3 -3 c41]);
s =addmf(s,'input',4,'positive','trimf',[a42 3 3]);

% Output control_force MFs
s =addvar(s,'output','force',[-55 55]);
s =addmf(s,'output',1,'mf1','trimf',[-55 -55 -48]);
s =addmf(s,'output',1,'mf2','trimf',[-55 -47.5 -40]);
s =addmf(s,'output',1,'mf3','trimf',[-47 -40 -33]);
s =addmf(s,'output',1,'mf4','trimf',[-40 -33 -26]);
s =addmf(s,'output',1,'mf5','trimf',[-33 -25 -17]);
s =addmf(s,'output',1,'mf6','trimf',[-26 -18 -10]);
s =addmf(s,'output',1,'mf7','trimf',[-17 -10 -3]);
s =addmf(s,'output',1,'mf8','trimf',[-10 -3 4]);
s =addmf(s,'output',1,'mf9','trimf',[-3 4 11]);
s =addmf(s,'output',1,'mf10','trimf',[4 11 18]);
s =addmf(s,'output',1,'mf11','trimf',[11 18 25]);
s =addmf(s,'output',1,'mf12','trimf',[18 25 33]);
s =addmf(s,'output',1,'mf13','trimf',[25 32 40]);
s =addmf(s,'output',1,'mf14','trimf',[33 40 48]);
s =addmf(s,'output',1,'mf15','trimf',[40 48 55]);
s =addmf(s,'output',1,'mf16','trimf',[48 55 55]);

% Rule List
ruleList=[
    1 1 1 1 1 w1 1
    1 1 1 2 2 w2 1
    1 1 2 1 3 w3 1
    1 1 2 2 4 w4 1
    1 2 1 1 5 w5 1
    1 2 1 2 6 w6 1
    1 2 2 1 7 w7 1
    1 2 2 2 8 w8 1
    2 1 1 1 9 w9 1
    2 1 1 2 10 w10 1
    2 1 2 1 11 w11 1
    2 1 2 2 12 w12 1
    2 2 1 1 13 w13 1
    2 2 1 2 14 w14 1
    2 2 2 1 15 w15 1
    2 2 2 2 16 w16 1];
```

```
s = addrule(s,ruleList);
writefis(s,'vee');
fismatrix = readfis('vee');
vee = fismatrix;
invl
set_param(gcs,'SaveOutput','on')
set_param(gcs,'SaveFormat','StructureWithTime')
sim(gcs);
b =
abs(e(10,1))+abs(e(30,1))+abs(e(50,1))+abs(e(70,1))+abs(e(90,1))+abs(e(
100,1))+abs(e(120,1))...

+abs(e(140,1))+abs(e(160,1))+abs(e(180,1))+abs(e(200,1))+abs(e(220,1))+
abs(e(240,1))+abs(e(260,1))...

+abs(e(280,1))+abs(e(300,1))+abs(e(320,1))+abs(e(340,1))+abs(e(350,1))+
abs(e(370,1))+abs(e(380,1))...

+abs(e(390,1))+abs(e(400,1))+abs(e(410,1))+abs(e(420,1))+abs(e(430,1))+
abs(e(440,1))+abs(e(450,1))...

+abs(e(460,1))+abs(e(470,1))+abs(e(480,1))+abs(e(490,1))+abs(e(500,1));
j =
abs(e1(10,1))+abs(e1(30,1))+abs(e1(50,1))+abs(e1(70,1))+abs(e1(90,1))+a
bs(e1(100,1))+abs(e1(120,1))...

+abs(e1(140,1))+abs(e1(160,1))+abs(e1(180,1))+abs(e1(200,1))+abs(e1(220
,1))+abs(e1(240,1))+abs(e1(260,1))...

+abs(e1(280,1))+abs(e1(300,1))+abs(e1(320,1))+abs(e1(340,1))+abs(e1(350
,1))+abs(e1(370,1))+abs(e1(380,1))...

+abs(e1(390,1))+abs(e1(400,1))+abs(e1(410,1))+abs(e1(420,1))+abs(e1(430
,1))+abs(e1(440,1))+abs(e1(450,1))...

+abs(e1(460,1))+abs(e1(470,1))+abs(e1(480,1))+abs(e1(490,1))+abs(e1(500
,1));
switch p;
case 1
    y1=b;
    z1=j;
case 2
    y2=b;
    z2=j;
case 3
    y3=b;
    z3=j;
case 4
    y4=b;
    z4=j;
case 5
    y5=b;
    z5=j;
case 6
    y6=b;
    z6=j;
case 7
```

```
     y7=b;
     z7=j;
case 8
     y8=b;
     z8=j;
case 9
     y9=b;
     z9=j;
case 10
     y10=b;
     z10=j;
end
p=p+1;
end;
Generation = g-1
R = [y1 y2 y3 y4 y5 y6 y7 y8 y9 y10];
Q = [z1 z2 z3 z4 z5 z6 z7 z8 z9 z10];


% 3. CALCULATION OF FITNESS VALUES :-
% ==================================
p=1;q=10;
while p<=q;
V1=R(1,p);
Q2=Q(1,p);
f = V1*V1 + Q2*Q2 + 0.0000000001;
switch p;
case 1;
     o1=f;
case 2
     o2=f;
case 3
     o3=f;
case 4
     o4=f;
case 5
     o5=f;
case 6
     o6=f;
case 7
     o7=f;
case 8
     o8=f;
case 9
     o9=f;
case 10
     o10=f;
end
p=p+1;
end
T  =o1+o2+o3+o4+o5+o6+o7+o8+o9+o10;
Fitness_Value = T
f1 =o1/T;
f2 =o2/T;
f3 =o3/T;
f4 =o4/T;
```

```
f5 =o5/T;
f6 =o6/T;
f7 =o7/T;
f8 =o8/T;
f9 =o9/T;
f10=o10/T;
Fi = [o1 o2 o3 o4 o5 o6 o7 o8 o9 o10];

% 4. REPRODUCTION USING ROULETTE WHEEL :-
% ==========================================
% wheel spin 1 to 10
p=1;q=10;
while p<=q;
RW=randsrc(1,1,[1 2 3 4 5 6 7 8 9 10;f1 f2 f3 f4 f5 f6 f7 f8 f9 f10]);
switch RW
case 1
    r=s1;
case 2
    r=s2;
case 3
    r=s3;
case 4
    r=s4;
case 5
    r=s5;
case 6
    r=s6;
case 7
    r=s7;
case 8
    r=s8;
case 9
    r=s9;
case 10
    r=s10;
end;
     switch p;
     case 1
         r1=r;s1=r;
     case 2
         r2=r;s2=r;
     case 3
         r3=r;s3=r;
     case 4
         r4=r;s4=r;
     case 5
         r5=r;s5=r;
     case 6
         r6=r;s6=r;
     case 7
         r7=r;s7=r;
     case 8
         r8=r;s8=r;
     case 9
         r9=r;s9=r;
     case 10
         r10=r;s10=r;
```

```
      end;
    p=p+1;
end;
% 5. CROSSOVER :-
% ===============
Pc=randsrc(1,1,[1 0; 0.7 0.3]);     % Pc = probability of crossover
if Pc==1
p=1;q=5;
M  =randperm(5);                    % M  = Mating
while p<=q
CS =randsrc(1,1,[1:1:87]);          % CS = crossover site
      if      M(1,p)==1;
          R2=r1;
      elseif M(1,p)==2;
          R2=r2;
      elseif M(1,p)==3;
          R2=r3;
      elseif M(1,p)==4;
          R2=r4;
      elseif M(1,p)==5;
          R2=r5;
      end;
              if      p==1;
                      R1=r6;
              elseif p==2;
                      R1=r7;
              elseif p==3;
                      R1=r8;
              elseif p==4;
                      R1=r9;
              elseif p==5;
                      R1=r10;
              end;
               switch CS
                  case 1
                      a=[R1(1,2:88);R2(1,2:88)];
                      F=flipud(a);
                      R1=[R1(1,1) F(1,1:87)];
                      R2=[R2(1,1) F(2,1:87)];
                  case 2
                      a=[R1(1,3:88);R2(1,3:88)];
                      F=flipud(a);
                      R1=[R1(1,1:2) F(1,1:86)];
                      R2=[R2(1,1:2) F(2,1:86)];
                  case 3
                      a=[R1(1,4:88);R2(1,4:88)];
                      F=flipud(a);
                      R1=[R1(1,1:3) F(1,1:85)];
                      R2=[R2(1,1:3) F(2,1:85)];
                  case 4
                      a=[R1(1,5:88);R2(1,5:88)];
                      F=flipud(a);
                      R1=[R1(1,1:4) F(1,1:84)];
                      R2=[R2(1,1:4) F(2,1:84)];
                  case 5
                      a=[R1(1,6:88);R2(1,6:88)];
                      F=flipud(a);
```

```
        R1=[R1(1,1:5) F(1,1:83)];
        R2=[R2(1,1:5) F(2,1:83)];
    case 6
        a=[R1(1,7:88);R2(1,7:88)];
        F=flipud(a);
        R1=[R1(1,1:6) F(1,1:82)];
        R2=[R2(1,1:6) F(2,1:82)];
    case 7
        a=[R1(1,8:88);R2(1,8:88)];
        F=flipud(a);
        R1=[R1(1,1:7) F(1,1:81)];
        R2=[R2(1,1:7) F(2,1:81)];
    case 8
        a=[R1(1,9:88);R2(1,9:88)];
        F=flipud(a);
        R1=[R1(1,1:8) F(1,1:80)];
        R2=[R2(1,1:8) F(2,1:80)];
    case 9
        a  = [R1(1,10:88);R2(1,10:88)];
        F  = flipud(a);
        R1 = [R1(1,1:9) F(1,1:79)];
        R2 = [R2(1,1:9) F(2,1:79)];
    case 10
        a  = [R1(1,11:88);R2(1,11:88)];
        F  = flipud(a);
        R1 = [R1(1,1:10) F(1,1:78)];
        R2 = [R2(1,1:10) F(2,1:78)];
    case 11
        a  = [R1(1,12:88);R2(1,12:88)];
        F  = flipud(a);
        R1 = [R1(1,1:11) F(1,1:77)];
        R2 = [R2(1,1:11) F(2,1:77)];
    case 12
        a  = [R1(1,13:88);R2(1,13:88)];
        F  = flipud(a);
        R1 = [R1(1,1:12) F(1,1:76)];
        R2 = [R2(1,1:12) F(2,1:76)];
    case 13
        a  = [R1(1,14:88);R2(1,14:88)];
        F  = flipud(a);
        R1 = [R1(1,1:13) F(1,1:75)];
        R2 = [R2(1,1:13) F(2,1:75)];
    case 14
        a  = [R1(1,15:88);R2(1,15:88)];
        F  = flipud(a);
        R1 = [R1(1,1:14) F(1,1:74)];
        R2 = [R2(1,1:14) F(2,1:74)];
    case 15
        a  = [R1(1,16:88);R2(1,16:88)];
        F  = flipud(a);
        R1 = [R1(1,1:15) F(1,1:73)];
        R2 = [R2(1,1:15) F(2,1:73)];
    case 16
        a  = [R1(1,17:88);R2(1,17:88)];
        F  = flipud(a);
        R1 = [R1(1,1:16) F(1,1:72)];
        R2 = [R2(1,1:16) F(2,1:72)];
```

```
case 17
    a  = [R1(1,18:88);R2(1,18:88)];
    F  = flipud(a);
    R1 = [R1(1,1:17) F(1,1:71)];
    R2 = [R2(1,1:17) F(2,1:71)];
case 18
    a  = [R1(1,19:88);R2(1,19:88)];
    F  = flipud(a);
    R1 = [R1(1,1:18) F(1,1:70)];
    R2 = [R2(1,1:18) F(2,1:70)];
 case 19
    a  = [R1(1,20:88);R2(1,20:88)];
    F  = flipud(a);
    R1 = [R1(1,1:19) F(1,1:69)];
    R2 = [R2(1,1:19) F(2,1:69)];
case 20
    a  = [R1(1,21:88);R2(1,21:88)];
    F  = flipud(a);
    R1 = [R1(1,1:20) F(1,1:68)];
    R2 = [R2(1,1:20) F(2,1:68)];
 case 21
    a  = [R1(1,22:88);R2(1,22:88)];
    F  = flipud(a);
    R1 = [R1(1,1:21) F(1,1:67)];
    R2 = [R2(1,1:21) F(2,1:67)];
case 22
    a  = [R1(1,23:88);R2(1,23:88)];
    F  = flipud(a);
    R1 = [R1(1,1:22) F(1,1:66)];
    R2 = [R2(1,1:22) F(2,1:66)];
 case 23
    a  = [R1(1,24:88);R2(1,24:88)];
    F  = flipud(a);
    R1 = [R1(1,1:23) F(1,1:65)];
    R2=  [R2(1,1:23) F(2,1:65)];
case 24
    a  = [R1(1,25:88);R2(1,25:88)];
    F  = flipud(a);
    R1 = [R1(1,1:24) F(1,1:64)];
    R2=  [R2(1,1:24) F(2,1:64)];
 case 25
    a  = [R1(1,26:88);R2(1,26:88)];
    F  = flipud(a);
    R1 = [R1(1,1:25) F(1,1:63)];
    R2 = [R2(1,1:25) F(2,1:63)];
case 26
    a  = [R1(1,27:88);R2(1,27:88)];
    F  = flipud(a);
    R1 = [R1(1,1:26) F(1,1:62)];
    R2 = [R2(1,1:26) F(2,1:62)];
case 27
    a  = [R1(1,28:88);R2(1,28:88)];
    F  = flipud(a);
    R1 = [R1(1,1:27) F(1,1:61)];
    R2 = [R2(1,1:27) F(2,1:61)];
 case 28
    a  = [R1(1,29:88);R2(1,29:88)];
```

```
        F  = flipud(a);
        R1 = [R1(1,1:28) F(1,1:60)];
        R2 = [R2(1,1:28) F(2,1:60)];
    case 29
        a  = [R1(1,30:88);R2(1,30:88)];
        F  = flipud(a);
        R1 = [R1(1,1:29) F(1,1:59)];
        R2 = [R2(1,1:29) F(2,1:59)];
    case 30
        a  = [R1(1,31:88);R2(1,31:88)];
        F  = flipud(a);
        R1 = [R1(1,1:30) F(1,1:58)];
        R2 = [R2(1,1:30) F(2,1:58)];
    case 31
        a  = [R1(1,32:88);R2(1,32:88)];
        F  = flipud(a);
        R1 = [R1(1,1:31) F(1,1:57)];
        R2 = [R2(1,1:31) F(2,1:57)];
    case 32
        a  = [R1(1,33:88);R2(1,33:88)];
        F  = flipud(a);
        R1 = [R1(1,1:32) F(1,1:56)];
        R2 = [R2(1,1:32) F(2,1:56)];
    case 33
        a  = [R1(1,34:88);R2(1,34:88)];
        F  = flipud(a);
        R1 = [R1(1,1:33) F(1,1:55)];
        R2 = [R2(1,1:33) F(2,1:55)];
    case 34
        a  = [R1(1,35:88);R2(1,35:88)];
        F  = flipud(a);
        R1 = [R1(1,1:34) F(1,1:54)];
        R2 = [R2(1,1:34) F(2,1:54)];
    case 35
        a  = [R1(1,36:88);R2(1,36:88)];
        F  = flipud(a);
        R1 = [R1(1,1:35) F(1,1:53)];
        R2 = [R2(1,1:35) F(2,1:53)];
    case 36
        a  = [R1(1,37:88);R2(1,37:88)];
        F  = flipud(a);
        R1 = [R1(1,1:36) F(1,1:52)];
        R2 = [R2(1,1:36) F(2,1:52)];
    case 37
        a  = [R1(1,38:88);R2(1,38:88)];
        F  = flipud(a);
        R1 = [R1(1,1:37) F(1,1:51)];
        R2 = [R2(1,1:37) F(2,1:51)];
    case 38
        a  = [R1(1,39:88);R2(1,39:88)];
        F  = flipud(a);
        R1 = [R1(1,1:38) F(1,1:50)];
        R2 = [R2(1,1:38) F(2,1:50)];
    case 39
        a  = [R1(1,40:88);R2(1,40:88)];
        F  = flipud(a);
        R1 = [R1(1,1:39) F(1,1:49)];
        R2 = [R2(1,1:39) F(2,1:49)];
     case 40
        a  = [R1(1,41:88);R2(1,41:88)];
        F  = flipud(a);
        R1 = [R1(1,1:40) F(1,1:48)];
        R2 = [R2(1,1:40) F(2,1:48)];
    case 41
        a  = [R1(1,42:88);R2(1,42:88)];
        F  = flipud(a);
        R1 = [R1(1,1:41) F(1,1:47)];
        R2 = [R2(1,1:41) F(2,1:47)];
    case 42
        a  = [R1(1,43:88);R2(1,43:88)];
        F  = flipud(a);
        R1 = [R1(1,1:42) F(1,1:46)];
        R2 = [R2(1,1:42) F(2,1:46)];
     case 43
        a  = [R1(1,44:88);R2(1,44:88)];
        F  = flipud(a);
        R1 = [R1(1,1:43) F(1,1:45)];
        R2 = [R2(1,1:43) F(2,1:45)];
    case 44
        a  = [R1(1,45:88);R2(1,45:88)];
        F  = flipud(a);
        R1 = [R1(1,1:44) F(1,1:44)];
        R2 = [R2(1,1:44) F(2,1:44)];
     case 45
        a  = [R1(1,46:88);R2(1,46:88)];
        F  = flipud(a);
        R1 = [R1(1,1:45) F(1,1:43)];
        R2 = [R2(1,1:45) F(2,1:43)];
    case 46
        a  = [R1(1,47:88);R2(1,47:88)];
        F  = flipud(a);
        R1 = [R1(1,1:46) F(1,1:42)];
        R2 = [R2(1,1:46) F(2,1:42)];
    case 47
        a  = [R1(1,48:88);R2(1,48:88)];
        F  = flipud(a);
        R1 = [R1(1,1:47) F(1,1:41)];
        R2 = [R2(1,1:47) F(2,1:41)];
    case 48
        a  = [R1(1,49:88);R2(1,49:88)];
        F  = flipud(a);
        R1 = [R1(1,1:48) F(1,1:40)];
        R2 = [R2(1,1:48) F(2,1:40)];
    case 49
        a  = [R1(1,50:88);R2(1,50:88)];
        F  = flipud(a);
        R1 = [R1(1,1:49) F(1,1:39)];
        R2 = [R2(1,1:49) F(2,1:39)];
     case 50
        a  = [R1(1,51:88);R2(1,51:88)];
        F  = flipud(a);
        R1 = [R1(1,1:50) F(1,1:38)];
        R2 = [R2(1,1:50) F(2,1:38)];
    case 51
```

```
        a   = [R1(1,52:88);R2(1,52:88)];
        F   = flipud(a);
        R1  = [R1(1,1:51)  F(1,1:37)];
        R2  = [R2(1,1:51)  F(2,1:37)];
    case 52
        a   = [R1(1,53:88);R2(1,53:88)];
        F   = flipud(a);
        R1  = [R1(1,1:52)  F(1,1:36)];
        R2  = [R2(1,1:52)  F(2,1:36)];
    case 53
        a   = [R1(1,54:88);R2(1,54:88)];
        F   = flipud(a);
        R1  = [R1(1,1:53)  F(1,1:35)];
        R2  = [R2(1,1:53)  F(2,1:35)];
    case 54
        a   = [R1(1,55:88);R2(1,55:88)];
        F   = flipud(a);
        R1  = [R1(1,1:54)  F(1,1:34)];
        R2  = [R2(1,1:54)  F(2,1:34)];
     case 55
        a   = [R1(1,56:88);R2(1,56:88)];
        F   = flipud(a);
        R1  = [R1(1,1:55)  F(1,1:33)];
        R2  = [R2(1,1:55)  F(2,1:33)];
    case 56
        a   = [R1(1,57:88);R2(1,57:88)];
        F   = flipud(a);
        R1  = [R1(1,1:56)  F(1,1:32)];
        R2  = [R2(1,1:56)  F(2,1:32)];
    case 57
        a   = [R1(1,58:88);R2(1,58:88)];
        F   = flipud(a);
        R1  = [R1(1,1:57)  F(1,1:31)];
        R2  = [R2(1,1:57)  F(2,1:31)];
    case 58
        a   = [R1(1,59:88);R2(1,59:88)];
        F   = flipud(a);
        R1  = [R1(1,1:58)  F(1,1:30)];
        R2  = [R2(1,1:58)  F(2,1:30)];
    case 59
        a   = [R1(1,60:88);R2(1,60:88)];
        F   = flipud(a);
        R1  = [R1(1,1:59)  F(1,1:29)];
        R2  = [R2(1,1:59)  F(2,1:29)];
    case 60
        a   = [R1(1,61:88);R2(1,61:88)];
        F   = flipud(a);
        R1  = [R1(1,1:60)  F(1,1:28)];
        R2  = [R2(1,1:60)  F(2,1:28)];
     case 61
        a   = [R1(1,62:88);R2(1,62:88)];
        F   = flipud(a);
        R1  = [R1(1,1:61)  F(1,1:27)];
        R2  = [R2(1,1:61)  F(2,1:27)];
    case 62
        a   = [R1(1,63:88);R2(1,63:88)];
        F   = flipud(a);
        R1 = [R1(1,1:62) F(1,1:26)];
        R2 = [R2(1,1:62) F(2,1:26)];
    case 63
        a  = [R1(1,64:88);R2(1,64:88)];
        F  = flipud(a);
        R1 = [R1(1,1:63) F(1,1:25)];
        R2 = [R2(1,1:63) F(2,1:25)];
    case 64
        a  = [R1(1,65:88);R2(1,65:88)];
        F  = flipud(a);
        R1 = [R1(1,1:64) F(1,1:24)];
        R2 = [R2(1,1:64) F(2,1:24)];
    case 65
        a  = [R1(1,66:88);R2(1,66:88)];
        F  = flipud(a);
        R1 = [R1(1,1:65) F(1,1:23)];
        R2 = [R2(1,1:65) F(2,1:23)];
    case 66
        a  = [R1(1,67:88);R2(1,67:88)];
        F  = flipud(a);
        R1 = [R1(1,1:66) F(1,1:22)];
        R2 = [R2(1,1:66) F(2,1:22)];
     case 67
        a  = [R1(1,68:88);R2(1,68:88)];
        F  = flipud(a);
        R1 = [R1(1,1:67) F(1,1:21)];
        R2 = [R2(1,1:67) F(2,1:21)];
    case 68
        a  = [R1(1,69:88);R2(1,69:88)];
        F  = flipud(a);
        R1 = [R1(1,1:68) F(1,1:20)];
        R2 = [R2(1,1:68) F(2,1:20)];
    case 69
        a  = [R1(1,70:88);R2(1,70:88)];
        F  = flipud(a);
        R1 = [R1(1,1:69) F(1,1:19)];
        R2 = [R2(1,1:69) F(2,1:19)];
    case 70
        a  = [R1(1,71:88);R2(1,71:88)];
        F  = flipud(a);
        R1 = [R1(1,1:70) F(1,1:18)];
        R2 = [R2(1,1:70) F(2,1:18)];
    case 71
        a  = [R1(1,72:88);R2(1,72:88)];
        F  = flipud(a);
        R1 = [R1(1,1:71) F(1,1:17)];
        R2 = [R2(1,1:71) F(2,1:17)];
    case 72
        a  = [R1(1,73:88);R2(1,73:88)];
        F  = flipud(a);
        R1 = [R1(1,1:72) F(1,1:16)];
        R2 = [R2(1,1:72) F(2,1:16)];
    case 73
        a  = [R1(1,74:88);R2(1,74:88)];
        F  = flipud(a);
        R1 = [R1(1,1:73) F(1,1:15)];
        R2 = [R2(1,1:73) F(2,1:15)];
```

```
case 74
    a  = [R1(1,75:88);R2(1,75:88)];
    F  = flipud(a);
    R1 = [R1(1,1:74) F(1,1:14)];
    R2 = [R2(1,1:74) F(2,1:14)];
case 75
    a  = [R1(1,76:88);R2(1,76:88)];
    F  = flipud(a);
    R1 = [R1(1,1:75) F(1,1:13)];
    R2 = [R2(1,1:75) F(2,1:13)];
case 76
    a  = [R1(1,77:88);R2(1,77:88)];
    F  = flipud(a);
    R1 = [R1(1,1:76) F(1,1:12)];
    R2 = [R2(1,1:76) F(2,1:12)];
 case 77
    a  = [R1(1,78:88);R2(1,78:88)];
    F  = flipud(a);
    R1 = [R1(1,1:77) F(1,1:11)];
    R2 = [R2(1,1:77) F(2,1:11)];
case 78
    a  = [R1(1,79:88);R2(1,79:88)];
    F  = flipud(a);
    R1 = [R1(1,1:78) F(1,1:10)];
    R2 = [R2(1,1:78) F(2,1:10)];
case 79
    a  = [R1(1,80:88);R2(1,80:88)];
    F  = flipud(a);
    R1 = [R1(1,1:79) F(1,1:9)];
    R2 = [R2(1,1:79) F(2,1:9)];
case 80
    a  = [R1(1,81:88);R2(1,81:88)];
    F  = flipud(a);
    R1 = [R1(1,1:80) F(1,1:8)];
    R2 = [R2(1,1:80) F(2,1:8)];
case 81
    a  = [R1(1,82:88);R2(1,82:88)];
    F  = flipud(a);
    R1 = [R1(1,1:81) F(1,1:7)];
    R2 = [R2(1,1:81) F(2,1:7)];
case 82
    a  = [R1(1,83:88);R2(1,83:88)];
    F  = flipud(a);
    R1 = [R1(1,1:82) F(1,1:6)];
    R2 = [R2(1,1:82) F(2,1:6)];
case 83
    a  = [R1(1,84:88);R2(1,84:88)];
    F  = flipud(a);
    R1 = [R1(1,1:83) F(1,1:5)];
    R2 = [R2(1,1:83) F(2,1:5)];
case 84
    a  = [R1(1,85:88);R2(1,85:88)];
    F  = flipud(a);
    R1 = [R1(1,1:84) F(1,1:4)];
    R2 = [R2(1,1:84) F(2,1:4)];
case 85
    a  = [R1(1,86:88);R2(1,86:88)];
```

```
                                F  = flipud(a);
                                R1 = [R1(1,1:85) F(1,1:3)];
                                R2 = [R2(1,1:85) F(2,1:3)];
                            case 86
                                a  = [R1(1,87:88);R2(1,87:88)];
                                F  = flipud(a);
                                R1 = [R1(1,1:86) F(1,1:2)];
                                R2 = [R2(1,1:86) F(2,1:2)];
                            otherwise
                                a  = [R1(1,88);R2(1,88)];
                                F  = flipud(a);
                                R1 = [R1(1,1:87) F(1,1)];
                                R2 = [R2(1,1:87) F(2,1)];
                            end;
                              if     M(1,p)==1;
                                      s1=R2;
                              elseif M(1,p)==2;
                                      s2=R2;
                              elseif M(1,p)==3;
                                      s3=R2;
                              elseif M(1,p)==4;
                                      s4=R2;
                              elseif M(1,p)==5;
                                      s5=R2;
                              end;

                              if      p==1;
                                      s6=R1;
                              elseif p==2;
                                      s7=R1;
                              elseif p==3;
                                      s8=R1;
                              elseif p==4;
                                      s9=R1;
                              elseif p==5;
                                      s10=R1;
                              end;
p=p+1;
end;
end;
% 6.MUTATION :-
% ----------
i= 0.03*88*10*(1/T);
if g<=5
    i=0;
elseif g>=6 & g<=15
   i=0.015*88*10*(1/T);
elseif g>=16 & g<=25
   i=0.02*88*10*(1/T);
elseif g>=26 & g<=35
    i=0.025*88*10*(1/T);
else i=i;
end;
i=i;Mutation_rate=i,a=1;
RG = [s1; s2; s3; s4; s5; s6; s7; s8; s9; s10;];
while a<=i
    l=randsrc(1,1,[(1:10)]);
```

```
      m=randsrc(1,1,[1:88]);
      o=RG(1,m);
       if o==0;
           RG(1,m)=1;
       elseif  o==1;
           RG(1,m)=0;
       end;
        a=a+1;
end;
IP=RG;
% 7.GENERATION OUTPUT :-
% -----------------------
     if      g==1;
             G1=IP;P0=T;i0=i;F0=Fi;
     elseif g==2;
             G2=IP;P1=T;i1=i;F1=Fi;
     elseif g==3;
             G3=IP;P2=T;i2=i;F2=Fi;
     elseif g==4;
             G4=IP;P3=T;i3=i;F3=Fi;
     elseif g==5;
             G5=IP;P4=T;i4=i;F4=Fi;
     elseif g==6;
             G6=IP;P5=T;i5=i;F5=Fi;
     elseif g==7;
             G7=IP;P6=T;i6=i;F6=Fi;
     elseif g==8;
             G8=IP;P7=T;i7=i;F7=Fi;
     elseif g==9;
             G9=IP;P8=T;i8=i;F8=Fi;
     elseif g==10;
             G10=IP;P9=T;i9=i;F9=Fi;
     elseif g==11;
             G11=IP;P10=T;i10=i;F10=Fi;
     elseif g==12;
             G12=IP;P11=T;i11=i;F11=Fi;
     elseif g==13;
             G13=IP;P12=T;i12=i;F12=Fi;
     elseif g==14;
             G14=IP;P13=T;i13=i;F13=Fi;
     elseif g==15;
             G15=IP;P14=T;i14=i;F14=Fi;
     elseif g==16;
             G16=IP;P15=T;i15=i;F15=Fi;
     elseif g==17;
             G17=IP;P16=T;i16=i;F16=Fi;
     elseif g==18;
             G18=IP;P17=T;i17=i;F17=Fi;
     elseif g==19;
             G19=IP;P18=T;i18=i;F18=Fi;
     elseif g==20;
             G20=IP;P19=T;i19=i;F19=Fi;
     elseif g==21;
             G21=IP;P20=T;i20=i;F20=Fi;
     elseif g==22;
             G22=IP;P21=T;i21=i;F21=Fi;
     elseif g==23;
```

```
        G23=IP;P22=T;i22=i;F22=Fi;
elseif g==24;
        G24=IP;P23=T;i23=i;F23=Fi;
elseif g==25;
        G25=IP;P24=T;i24=i;F24=Fi;
elseif g==26;
        G26=IP;P25=T;i25=i;F25=Fi;
elseif g==27;
        G27=IP;P26=T;i26=i;F26=Fi;
elseif g==28;
        G28=IP;P27=T; i27=i;F27=Fi;
elseif g==29;
        G29=IP;P28=T;i28=i;F28=Fi;
elseif g==30;
        G30=IP;P29=T;i29=i;F29=Fi;
elseif g==31;
        G31=IP;P30=T;i30=i;F30=Fi;
elseif g==32;
        G32=IP;P31=T;i31=i;F31=Fi;
elseif g==33;
        G33=IP;P32=T;i32=i;F32=Fi;
elseif g==34;
        G34=IP;P33=T;i33=i;F33=Fi;
elseif g==35;
        G35=IP;P34=T;i34=i;F34=Fi;
elseif g==36;
        G36=IP;P35=T;i35=i;F35=Fi;
elseif g==37;
        G37=IP;P36=T;i36=i;F36=Fi;
elseif g==38;
        G38=IP;P37=T;i37=i;F37=Fi;
elseif g==39;
        G39=IP;P38=T; i38=i;F38=Fi;
elseif g==40;
        G40=IP;P39=T;i39=i;F39=Fi;
elseif g==41;
        G41=IP;P40=T;i40=i;F40=Fi;
elseif g==42;
        G42=IP;P41=T;i41=i;F41=Fi;
elseif g==43;
        G43=IP;P42=T;i42=i;F42=Fi;
elseif g==44;
        G44=IP;P43=T;i43=i;F43=Fi;
elseif g==45;
        G45=IP;P44=T;i44=i;F44=Fi;
elseif g==46;
        G46=IP;P45=T;i45=i;F45=Fi;
elseif g==47;
        G47=IP;P46=T;i46=i;F46=Fi;
elseif g==48;
        G48=IP;P47=T;i47=i;F47=Fi;
elseif g==49;
        G49=IP;P48=T;i48=i;F48=Fi;
elseif g==50;
        G50=IP;P49=T;i49=i;F49=Fi;
elseif g==51;
        P50=T;i50=i;F50=Fi;
```

```
     end;
     g=g+1;
 end;
% 9 DISPLAY OF RESULTS :-
if Generation ==50
y=[P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19
P20 P21 P22 P23 P24 P25 P26...
   P27 P28 P29 P30 P31 P32 P33 P34 P35 P36 P37 P38 P39 P40 P41 P42 P43
P44 P45 P46 P47 P48 P49 P50];
I=[i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 i13 i14 i15 i16 i17 i18 i19
i20 i21 i22 i23 i24 i25 i26...
   i27 i28 i29 i30 i31 i32 i33 i34 i35 i36 i37 i38 i39 i40 i41 i42 i43
i44 i45 i46 i47 i48 i49 i50];
x=1:1:50;
subplot(2,1,1);
plot(x,y,'-r')
xlabel('x = GENERATION ----->')
ylabel('Y = FITNESS VALUE ---->')
title('PLOT OF FITNESS')
legend('fitness')
subplot(2,1,2);
plot(x,I,'-r')
xlabel('x = GENERATION ----->')
ylabel('Y = MUTATION RATE ---->')
title('PLOT OF MUTATION')
legend('fitness')
end
```