GENETIC OPTIMIZATION OF FUZZY CONTROLLER

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARDOF THE DEGREE OF

MASTER OF ENGINEERING

IN

CONTROL & INSTRUMENTATION

SUBMITTED BY

ZELALEM GIRMA

(ROLL NO. 3348)

UNDER THE ESTEEMED GUIDANCE

OF

DR. Parmod Kumar

(PROFESSOR & HEAD)

DEPARTMENT OF ELECTRICAL ENGINEERING

DELHI COLLEGE OF ENGINEERING

UNIVERSITY OF DELHI

2004-2005

CERTIFICATE

This is to certify that **Mr. Zelalem Girma,** student of Delhi College of Engineering, Delhi worked under my guidance on Major project entitled **"DESIGN OF FUZZY-GENETIC CONTROLLER FOR LIQUID LEVEL CONTROL"** being submitted in partial fulfilment of the requirement for the award of the degree of "**MASTER OF ENGINEERING"** in C**ONTROL AND INSTRUMENTATION** to the Department of Electrical Engineering, Delhi College of Engineering**,** Bawana Road, New Delhi. Further, it is also certified that this project has not been submitted for any other Degree or Diploma in any college to the best of my knowledge and belief.

Dr. PARMOD KUMAR

Professor and Head of,

Department of Electrical Engineering,

Delhi College of Engineering,

New Delhi

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 Knowledge Base of a Fuzzy Logic Controller (FLC) encapsulates expert knowledge and consists of the Data Base (membership functions) and Rule-Base of the controller. Optimization of both of these Knowledge Base components is critical to the performance of the controller and has traditionally been achieved through a process of trial and error. Such an approach is convenient for FLCs having low numbers of input variables. However, for large numbers of inputs, more formal methods of Knowledge Base optimization are required. The construction of these controllers can be quick and effective in the presence of expert knowledge; conversely, in the absence of such knowledge, their design can be slow and based on trial-and-error rather than a guided approach.

Genetic Algorithms provide a way of surmounting this shortcoming. These algorithms use some of the concepts of evolutionary theory, and provide an effective way of searching a large and complex solution space to give close to optimal solutions in much faster times than random trial-and-error. They are also generally more effective at avoiding local minima than differentiation-based approaches.

In this report the application of Genetic Algorithms to the design and optimization of Fuzzy Logic Controllers is demonstrated. These controllers are characterized by a set of parameters.

The efficacy of this approach had been tested by comparison of the GA-FLC's performance in controlling a liquid level control system, to that of heuristically-tuned FLC.

Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER-I
INTRODUCTION

INTRODUCTION

Traditionally, control systems modeling has been based upon the use of mathematical techniques to model the input/output relationship of the system in question [1] [2]. Such an approach relies upon a mathematical description of the plant in order to model the behavior of the system, subject to certain conditions and assumptions. Many real-world systems however, may not be as readily described mathematically due to the complexity of the components of the plant and the interaction between them, and consequently, the model may be subject to certain assumptions or conditions [3].

In such models, the degree of mathematical precision required to completely describe every aspect of the process, is either prohibitive or non-trivial. In addition, for actual implementation of such systems, heuristics, gained through human experience, are often employed in the tuning of the final controller [1][4].

The use of Fuzzy Logic [1][2] has found application in the area of control system design where human expert knowledge, rather than precise mathematical modeling, of a process or plant is used to model/implement the required controller [3] [4]. Human expert knowledge is based upon heuristic information gained in relation to the operation of the plant or process, and its inherent vagueness (“fuzziness”) offers a powerful tool for the modeling of complex systems.

Uncertainty and ambiguity are evident in many engineering problems. For example, system stability can be considered fuzzy in the sense that a system can be lightly-damped, under-damped, over damped, etc. Fuzzy Logic Control (FLC) therefore provides a formal method of translating subjective and imprecise human knowledge into control strategies, thus facilitating better system performance through the exploitation and application of that knowledge.

Optimal design of the FLC Knowledge Base (membership functions and rule -base) is central to the performance of the FLC. The rule -base reflects the human expert knowledge, expressed as linguistic variables, while the membership functions represent expert interpretation of those same variables. In the absence of such knowledge, a

common approach is to optimize these FLC parameters through a process of trial and error, with respect to the performance of the system for each Knowledge Base formulated [5]. This approach becomes impractical for systems having significant numbers of inputs, since the rule-base size grows exponentially, and consequently the number of rule combinations becomes significantly large [7]. Although a priori knowledge can eliminate the requirement to test every possible combination, a more formal method is nonetheless required. The use of Genetic Algorithms (GA) in this regard can provide such a solution [10].

Genetic Algorithms (GAs) [11][12] are robust, numerical search methods that mimic the process of natural selection. Although not guaranteed to absolutely find the true global optima in a defined search space, GAs are renowned for finding relatively suitable solutions within acceptable time frames and for applicability to many problem domains. Using primary concepts of genetic selection, crossover and mutation, GAs are a stochastic, but directed method of identifying global optima within a problem domain-space. This is achieved through successive performance testing of potential solutions (i.e. members) which collectively form a population, with respect to a problem objective function. Information exchange between "better performing" candidates is facilitated by the genetic operators and occurs across generations of populations. In this way, convergence to optimal solutions is achieved.

In this study, simple GA (selection, crossover and mutation only) was used to evaluate potential solutions for FLC membership function parameter optimization, with a view to optimizing the performance of the FLC, when applied to liquid level control system, and to identify and discuss issues arising from the evaluation.

1.2 STATEMENT OF THE PROBLEM

Recent studies on optimization showed that fuzzy controllers led proportional-integral-derivative (PID) controllers with insignificant magnitude of ISE. Thus, the advantage of fuzzy controllers over PID controllers is inconspicuous. It is, therefore, necessary to try another optimization technique to further enhance the performance of fuzzy controllers.

The main problem of this study is how to optimize a fuzzy controller using GA. Specifically; it is the concern of this project on how to genetically vary the width of membership functions to obtain minimum ISE.

1.3 OBJECTIVES OF THE STUDY

The main objective of this study is to establish a methodology on how to apply GA in tuning fuzzy controller. This tuning process determines the optimum fuzzy controller parameters that would yield minimum ISE.

It specifically intends to seek solutions on how to apply GA to automatically determine the width of the output membership functions that would result to minimum ISE. Further, it investigates the convergence rate, the lowest ISE ever attained, and the consistency of the optimization process.

1.4 SIGNIFICANCE OF THE STUDY

The result of this study will establish a sound methodology in optimizing a fuzzy controller using GA. It will also serve as a benchmark of any effort to further verify the validity of the work on an actual set-up. Furthermore, it will serve as a basis of any study regarding GA search and optimization inside and outside fuzzy controller.

1.5 SCOPE AND LIMITATIONS

The study focuses on the application of GA in the optimization of a fuzzy controller used in liquid level control system. In order to minimize the parameter set and reduce simulation's processing time, only the output membership function of the controller is varied. The input membership function and fuzzy rules are fixed. The optimization processes will then be evaluated using the ISE performance criterion. The available function in Genetic Algorithm Optimization Toolbox and Fuzzy Logic Toolbox (FLT) for MATLAB are being used. The work is limited to computer simulation only. Physical experimentation is beyond the scope of this study.

1.6 LITERATURE REVIEW

Conventional control system design is predominantly based upon the development of mathematical models, which attempt to describe the dynamic behavior of the system in question using typically, differential equations. Many such methods of system design have been developed over time, including 3-term, Proportional, Integral, Derivative control (PID-control) [3][12], root-locus design [13], state-space [14] etc. offering a variety of ways in which good controller design may be achieved. In practice, PID (P, PI) control systems predominate, with approximately 90% of all controllers in operation, being of this form [3] [5]. The dominance of PID control may be explained by the fact that it is a long-standing, successful design technique, well understood by industrial users [8].

PID control however, has some disadvantages associated with it, one of which is that PID was originally designed for single -input, single -output (SISO) operation [4]. However, in practice, many real-world systems are multiple -input, multiple-output (MIMO), and control of such systems can be complicated involving several independent control loops [4]. For such systems, complex mathematical models are required in order to exploit the interdependencies of physical variables and this may be impractical for several reasons, including lengthy model development time [4], and application of extensive simplifications or assumptions to the model, which at implementation stage will require heuristic tuning [3][5].

In addition, modeling the system is usually performed using the linear region of operation of the plant. This can lead to poor system response to disturbances or nonlinear behavior, as a result of component wear and tear over time, for example. A second disadvantage associated with mathematical-based models is that heuristic information/knowledge which does not ‘fit easily’ into the mathematical framework of the model, tends not to be included in the design process, but rather is left to the tuning stage of model implementation thus rendering the model less robust [3].

Fuzzy Control [3] is the application of Fuzzy Logic [3] to control systems design. Fuzzy Logic was pioneered by Lotfi Zadeh [15] in 1965 and is a super-set of Boolean logic theory used to represent imprecision and uncertainty. Imprecision and uncertainty

(i.e."fuzziness") are qualities inherent within human language when used to describe perceptions of observed events and objects.
Uncertainty and imprecision, caused by non-linearity such as noise and disturbances, are also evident in the operation of real-world systems. The use of mathematical models to develop such systems may result in poor system performance if the dynamics of actual events do not form part of the model. This is because a full and complete mathematical model describing every possible cause and- effect event is not feasible [5]. The required degree of mathematical precision becomes even greater for non-linear system design.

Increased mathematical complexity required for conventional modeling techniques thus led to the application of Fuzzy Logic to control systems design in the form of Fuzzy Logic Controllers and their use has steadily grown among industrial users [5].
The case for FLC is that it translates human abstractions/perceptions in the form of expert knowledge, into the numerical domain, thus facilitating suitable system controller design while avoiding the need for stringent mathematical, system descriptions [3]. Application of FLC in the domain of PID control has been, in general, used to augment existing PID controller operation or replacement of PID by FLC [16][17].

The use of FLCs however requires optimized knowledge bases. Commonly, heuristic tuning of rule - bases and/or membership functions has been through a combination of trial and error and expert knowledge [5]. For increasingly larger numbers of inputs and outputs (rule s), as well as membership functions, even with *a priori*, expert knowledge, this method becomes more difficult as the number of rules rises exponentially [6]. Artificial intelligence techniques such as Artificial Neural Networks (ANN) and Genetic Algorithms have been advocated as solutions to this problem [2][9]. Neural Nets can be used to learn FLC rule-bases using input-output data, or to tune given membership functions. Such an approach combines the learning capability of the ANN with the reasoning power of fuzzy logic [2]. Alternatively, the search for suitable rules or membership function parameters of a FLC knowledge base can be viewed as an optimization problem to which a GA can be applied [9] [18].
Genetic Algorithms (GAs) are numerical optimization techniques based upon the mechanics of natural selection [11] [12]. Pioneered in the 1970's by John Holland [18] and colleagues at the University of Michigan [19], GAs use the concept of "survival of the fittest" to determine, through successive generations of randomized, but directed,

information exchange, the optimal sample value or point in a search space [11][12]. GAs offer a robust search mechanism as *a priori* knowledge is not generally required for successful application of the algorithm, and they have been successfully applied to a wide range of optimization problems [11].

The key operators of a GA are selection, crossover, mutation and population size [10][11][23], with secondary parameters including variable encoding [10][11][24] and decoding [10][24], and population-update [10][24], also having a bearing upon the effectiveness of the algorithm. Due to the stochastic nature of operation and the wide range of problems to which GAs may be applied, a formal, analytical description of the interactions and dependencies of the various parameters of a GA is complex, and consequently selection of GA parameters and associated settings, has to date been largely based upon empirical evaluation [25].

GAs can be used to optimize a FLC by optimizing the rule -base [26], membership functions [18], or knowledge base (i.e. both rule -base and membership functions) [18] [6]. For the GA-optimization of a rule-base, three different population strategies, known as Pittsburgh, Michigan and Iterative Rule are possible [18][26]. For the Michigan strategy, each GA chromosome in the population represents a single rule and so the population as a whole represents the entire rule -base, which is modified over time through interaction with the environment. The Pittsburgh strategy is to encode entire rule bases into each chromosome so that the entire population represents many competing rule -bases. The Iterative Rule strategy is similar to the Michigan approach, in the sense that single rules are found, but only the best individual member is considered to form part of the solution at each stage with the remaining population members discarded.

Although the Pittsburgh approach involves more computational overhead [18], the *competition* among complete rule -bases, as opposed to single -rules, induces better *cooperation* among the rules of any proposed solution for the FLC [18].

In addition to choice of population strategy and all other GA parameters previously outlined, the form of objective function used in evaluating GA-population members is an essential factor of the optimization task [6]. In the realm of FLC optimization, numerous objective functions have been adopted based on minimizing particular attributes of the system response such as the Mean Square Error (MSE) [6], cost equations based on response rise and settling times [8] and integral of absolute error (IAE). Irrespective of definition, the suitability of the objective function as a system performance indicator is

critical to the successful application of a Genetic Algorithm and thus requires careful selection. Once constructed around a suitable objective function, the GA proceeds to evaluate candidate solutions using the objective function in order to gauge the performance of each potential solution when applied to the problem domain [26] [28].

1.7 REPORT LAYOUT

In relation to work performed in this study, the remainder of the report consists of the following chapters;

Chapter 2 presents a discussion on the fuzzy logic and fuzzy set theory. In this chapter we will discuss what fuzzy logic is? How it differ from classical logic and set theory and how it handles the problem of imprecision in the real world problems.

Chapter 3 presents about fuzzy modeling. Here we will explain what we mean fuzzy modeling? , types of fuzzy modeling i.e. mamdani and sugeno modeling, the advantage and disadvantage of one type of modeling over the other.

Chapter 4 discusses the development and simulation of the heuristic FLC (he-FLC)

While Chapter 5 and 6 presents the motivation for, and analysis of, a parametric evaluation of the Simple Genetic Algorithm (SGA) that was performed in order to gain insight into appropriate parameter methods and settings that should be used for application of the SGA to the FLC optimization problem. To ascertain if the GA-optimization approach of fuzzy controllers used is valid, the performance of the GA first checked by using simple multimodal function.

Chapter7 briefly discusses genetic fuzzy systems (GFS) and details the design of the SGA used to optimize the FLC and also outlines how MATLAB fuzzy inference systems (FIS) were used in conjunction with the GA to encode complete FLC rule -bases and membership function specifications.

The final chapter offers the design of fuzzy genetic system namely liquid level control system. The chapter concludes with an evaluation of the results of the optimization attempt and issues arising from it, through comparison with the conventional and heuristic FLC models.

CHAPTER-II
FUZZY LOGIC CONCEPTS

INTRODUCTION

Fuzzy Logic is a multi level logic system that uses linguistic variables such as "long", 'fast", "cool", "heavy", "middle-aged" and so on. Objects can have varying degrees of membership of such fuzzy sets, ranging from a crisp "definitely not a member" (denoted by 0) to a crisp "definitely a member" (denoted by 1). The crucial distinction is that between these crisp extremes, objects can have less certain degrees of membership such as "not really a member" (perhaps denoted by a 0.1) and "pretty much a member" (0.9 possibly).

2.2 FUZZY LOGIC

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. This can allow for the handling of borderline cases and other hard-to-categorizes situations in a more intuitively satisfying way.

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. In this thesis, fuzzy logic is not considered as a logic, but as a computing method. Fuzzy systems are considered from two points of view. First the system has linguistic representation with linguistic variables and fuzzy if-then rules. On the other hand, fuzzy logic is used as a numerical method to create a nonlinear mapping between inputs and outputs.

2.3 HISTORICAL BACKGROUND OF FUZZY LOGIC

The idea of fuzzy logic (grade of membership) was born in July 19 65 by Lofti Zadeh well respected professor of Electrical and Computer Engineering at university of California, Berkeley [1]. Proff. Zadeh believed that real world problem could be solved with efficient, analytical method. In his 1961 paper he has mentioned that traditional system analysis technique were too precise for many complex real world problems. In one of his seminal paper ,Zadeh explained the principle of incompatibility ,which state

"*…. as the complexity of a system increases, our ability to make precise yet significant descriptions about its behavior diminishes until a threshold is reached beyond which precision become almost mutually exclusive characteristics"*

When we see from practical point of view the idea of Zadeh was acceptable. One has to pay a cost for high precision but the cost may be too high to be practical. The following figure clearly indicates the relation ship between cost, utility and precision. The vertical axis serves dual purpose of representing both the cost and degree of utility and the horizontal axis represent the degree of precision.

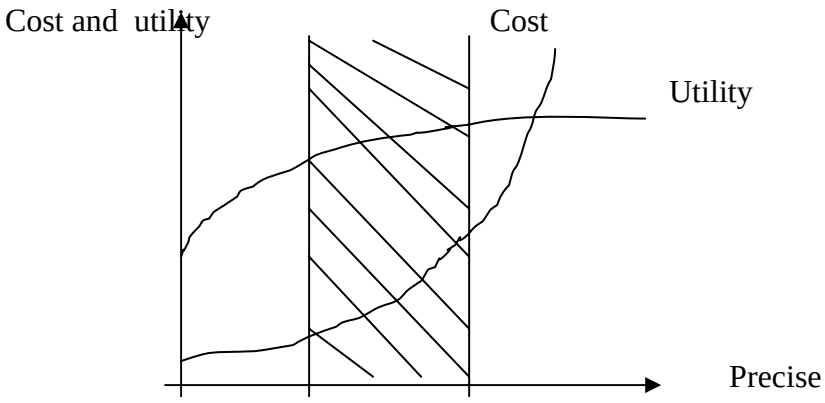


Fig2-1 The cost –precision trade-off

The above figure indicates that as precision of the system increases, the cost for developing the system also increases, in exponential manner. On the other hand usefulness (utility) of the system does not increase proportionally as its precision increases, it usually saturate after a certain point. Therefore, the gray area is both *cost effective* and *highly useful.* This leads for the development of fuzzy logic and its fundamental principle that is to develop cost effective approximate solution to complex

problem by exploiting the tolerance of imprecision. As result, fuzzy logic brings a complementary viewpoint –a view in which cost effectiveness rather than precision [1].

Zadeh has encountered sharp criticism due to fuzzy set concept from academic community. Some reject it because of the name, without knowing the content in detail. Others rejected it because of theory's emphasis on imprecision,- a major departure from the western scientific discipline's focus on precision. However, starting from 1965 many researchers around the world became Zadeh's follower.

In 1974 E.Mamdani in United Kingdom developed the first fuzzy logic controller, which was controlling a steam generator. Panasonic company in Japan was the first to apply fuzzy logic to consumer product, a shower head that control water temperature in 19 87. The first fuzzy logic VLSI chip for performing fuzzy logic inference was developed by M.Togie and H.watanabe in 1986 [1]. Later they formed a company that sold hardware and software package for developing fuzzy logic application.. Later on several companies were formed in late 19 80s but did not survive through mid 1990 due to the fact that vendors of conventional control design software such as *mathworks* started introducing add on tool boxes for designing fuzzy system in 1994.

Since the introduction of the theory of fuzzy sets by L. A. Zadeh, and the industrial application of the first fuzzy controller by E. H. Mamdani, fuzzy systems have obtained a major role in engineering systems and consumer products in the 1980s and 1990s.

2.4FUZZY BASICS

The primary objective of fuzzy logic is to map an input space to an output space. The way of controlling this mapping is to use if-then statements known as rules [4]. The order these rules are carried out in is insignificant since all rules run concurrently. The following sections will present and develop ideas such as sets, membership functions, logical operators, linguistic variables and rule bases.

2.5 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 fifteen 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 2-2 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, 0.5.

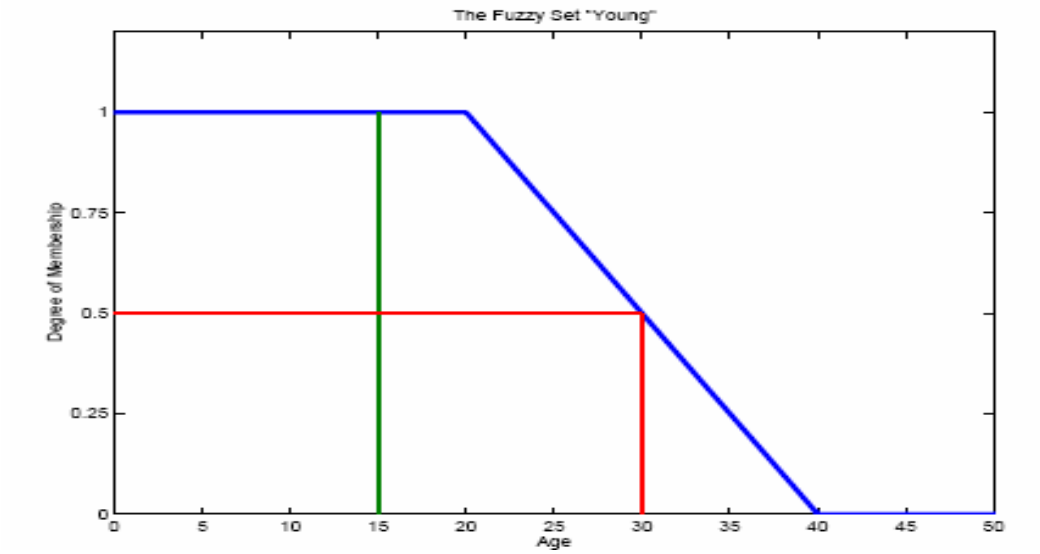


Fig 2-2 The Fuzzy Set "Young"

The Universe of Discourse of a variable is the range of values that 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 2-3 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.

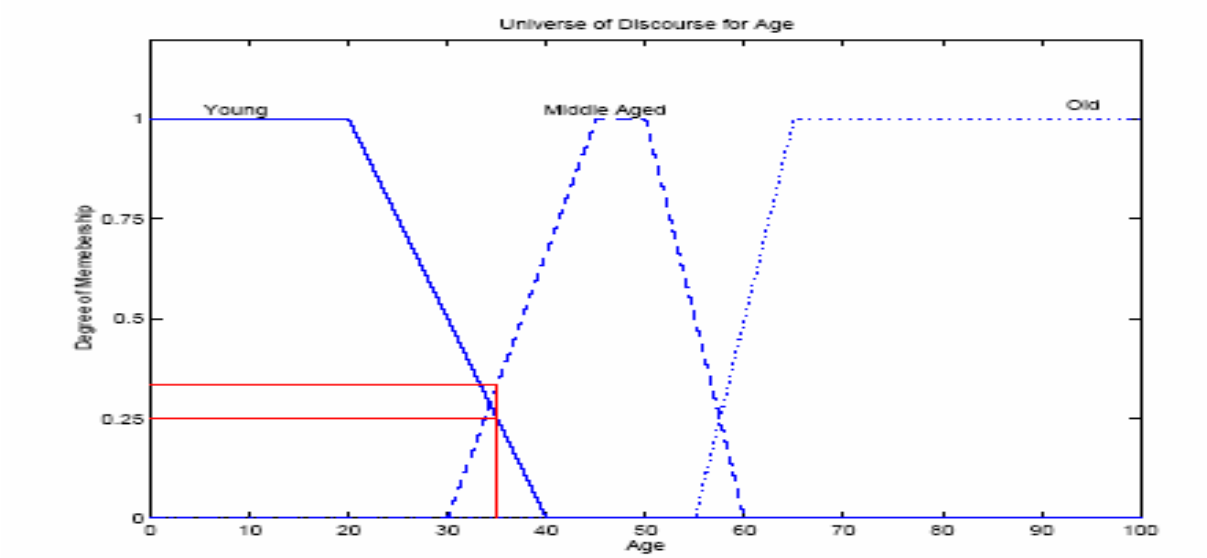


Fig2-3 "Age" Universe of Discourse

Fuzziness primarily describes uncertainty or partial truth. The concept of partial truth characterized by fuzziness has led to a new theory of fuzzy sets, which has yielded a more accurate representation of perception of truth. In Boolean logic, truth values are 0 or 1. In fuzzy logic, truth is a matter of degrees. Fuzzy logic provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy or missing input information. Fuzzy Logic incorporates a simple, rule-based IF X AND Y THEN Z approach to solving a control problem rather than attempting to model a system mathematically. Fuzzy logic uses the concept of linguistic or fuzzy variables, e.g., "large positive" error, "small positive" error, "zero" error, "small negative" error, and "large negative" error. Based on these linguistic variables, a rule matrix is constructed which relates the input to output in linguistic term.

The rule matrix is implemented with the help of membership function, which is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, defines functional overlap between inputs, and ultimately determines an output response. The inputs are combined logically using the AND operator to produce output response values for all expected inputs. The active conclusions are then combined into a logical sum for each membership function. A firing strength for each output membership function is computed. All that remains is to combine these logical sums in a defuzzification process to produce the crisp output that is accomplished by combining the results of the inference process and then computing the "fuzzy centroid" of the area. The weighted strengths of each output member function are

multiplied by their respective output membership function center points and summed. Then, this area is divided by the sum of the weighted member function strengths and summed. Finally, this area is divided by the sum of the weighted member function strengths and the result is taken as the crisp output.

2.6UNIQUE FEATURES OF FUZZY LOGIC

Fuzzy Logic offers several unique features that make it a particularly appealing choice for many control problems.

1. It is inherently robust since it does not require precise, noise-free inputs and can be programmed to fail safely if a feedback sensor quits or is destroyed. The output control is a smooth control function despite a wide range of input variations.
2. Since the Fuzzy Logic controller processes user-defined rules governing the target control system, it can be modified and tweaked easily to improve or drastically alter system performance. New sensors can be conveniently incorporated into the system by generating appropriate governing rules.
3. Fuzzy Logic is not limited to a few feedback inputs and one or two control outputs, nor is it necessary to measure or compute rate-of-change parameters in order for it to be implemented. Any sensor data that provides some indication of a system's actions and reactions is sufficient. This allows the sensors to be inexpensive and imprecise thus keeping the overall system’s cost and complexity low.
4. Because of the rule-based operation, any reasonable number of inputs can be processed (1-8 or more) and numerous outputs (1-4 or more) generated, although defining the rule base quickly becomes complex if too many inputs and outputs are chosen for a single implementation since rules defining their interrelations must also be defined. It would be better to divide the control system into smaller chunks and use several smaller FL controllers distributed on the system, each with limited individual responsibilities.

Fuzzy Logic can control nonlinear systems that would be difficult or impossible to model mathematically. This opens the door for controlling systems that would normally be deemed unfeasible for automation.

2.7 FUZZY SETS, MEMBERSHIP FUNCTIONS AND LOGICAL OPERATORS

2.7.1 FUZZY SET THEORY

Fuzzy logic starts with the concept of a fuzzy set. A *fuzzy set* is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership. To understand what a fuzzy set is, first consider what is meant by what we might call a *classical set*. A classical set is a container that wholly includes or wholly excludes any given element. Most human data is not binary (YES,NO)type.

For example let us take the concept of warm room temperature in the range [60^{o}_{c} $80\,^{o}_{c}$]

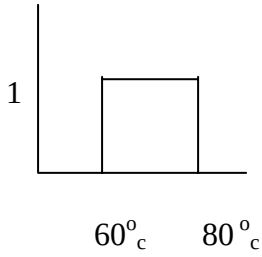


Fig.2.4 classical set

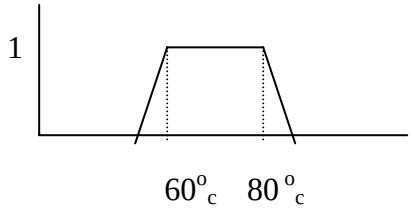


Fig.2.5 fuzzy set

In classical set as shown in the above figure the transition from non-membership to membership is abrupt whereas in fuzzy set the transition is gradual.

Fuzzy set has the following properties;

- It has smooth boundary
- Membership in a set is a matter of degree i.e. in classical set theory an object either belong to the set or not (temperature is cold/hot, glass full/empty, person is good/bad) but in fuzzy set theory belong ness is a matter of degree(depend up on grade of membership) eg. A person can be 80% good and 20% bad
- In classical set theory every individual object is assigned a membership value either 1 or 0 that discriminate between membership and non membership of the crisp set whereas in fuzzy set an object can take a grade of membership between 0 and 1 i.e. [0 1].
- In classical set theory crisp set are based on a two value logic(yes/no) whereas fuzzy set theory is based on multi-value logic

Fuzzy set is characterized by a mapping from universe of discourse in to interval [0 1]. This mapping is the membership function of the set.

- Fuzzy set violet two law of classical set theory due to permission of partial membership

    1) *Law of Contradiction*: $A \cup A' = U$ (i.e., a set and its complement must comprise the universe of discourse, any object must belong to a set or to its complement); in fuzzy set $A \cup A' \neq U$
    2) *Law of Excluded Middle*: $A \cap A' = \varnothing$ (i.e., a set and its complement are disjoint, any object can only be in one of either a set or its complement, it cannot simultaneously be in both). In fuzzy set $A \cap A' \neq \varnothing$'

Indeed $\forall x \in A$ such that $\mu_A(x) = \propto$, $0 < \propto < 1$: $\mu_{A \cup A'}(x) = \max\{\propto, 1-\propto\} \neq 1$ and $\mu_{A \cap A'}(x) = \min\{\propto, 1-\propto\} \neq 0$. Note that all other classical set operations are valid in fuzzy set theory.

2.7.2 REPRESENTATION OF FUZZY SET

A fuzzy set can be defined in two ways;

a) By enumerating membership value of those elements in the set completely or partially. This is possible if the set is discrete, because a continuous fuzzy set has infinite number of fuzzy set. A fuzzy set A can be defined by enumeration using expression

$$A = \sum_i \mu_i(x_i) \Big/ x_i$$

where the summation and addition operator refers to the union operation and the notation $\mu_i(x_i)/x_i$ refers to fuzzy set containing exactly one (partial) element x with membership degree $\mu_i(x_i)$

b) by defining the membership function mathematically

eg. The term *high* in terms of temperature may be defined us

$$\mu_{High}(T) = \begin{cases} 0 & \text{if } 0^{o}_{c} < T < 40 \\ (T-20)/40 & \text{if } 40 \leq T \leq 50 \\ 1 & \text{if } T > 50 \end{cases}$$

Fuzzy sets are sets without clear or crisp boundaries. The elements they contain may only have a partial degree of membership. They are therefore not the same as classical sets in the sense that the sets are not closed. Some examples of vague fuzzy sets and their respective units include the following.

- Loud noises (sound intensity)
- High speeds (velocity)
- Desirable actions (decision of control space)

Fuzzy sets can be combined through fuzzy rules to represent specific actions/behavior and it is this property of fuzzy logic that will be utilized when implementing a fuzzy logic controller in subsequent chapters.

A classical set may be for example written as:

$A = \{x \mid x > 3\}$

Now if X is the universe of discourse with elements x then a fuzzy set A in X is defined as a set of ordered pairs:

$A = \{x, \mu A(x) \mid x \in X\}$

Note that in the above expression $\mu A(x)$ may be called the membership function of x in A and that each element of X is mapped to a membership value between 0 and 1. Typical membership function shapes include triangular, trapezoidal and Gaussian functions. The shape is chosen on the basis of how well it describes the set it represents. Below in Figure 2.3 some example fuzzy sets can be observed.

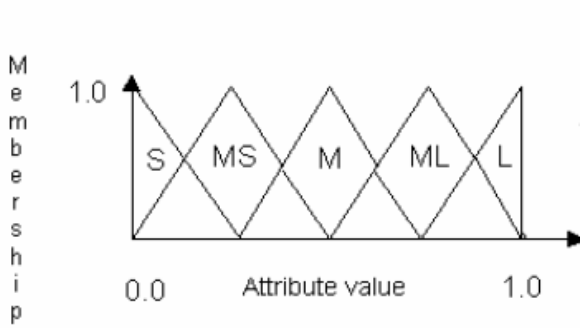


Fig 2.6 Example fuzzy sets. S,small; MS,medium small; M,medium;ML,medium Large;L,Large.

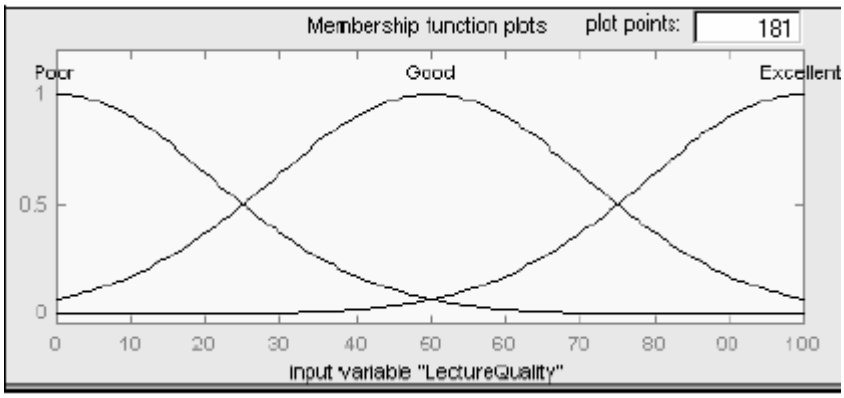


Fig 2.7 Sample three-part Gaussian shaped MF

2.7.3 MEMBERSHIP FUNCTIONS

Membership functions form the basis for fuzzy computing. The shape, the overlapping, peak values, and their continuity properties determine how the fuzzy system can be designed and how it behaves. Thus it is worthwhile to consider the membership functions in more detail.

Membership functions have a number of properties according to which they can be classified to different categories. Some of them are useful also from a practical point of view.

Those properties are related to the membership function but also to the set of membership functions. Here the set means membership functions that are defined for a certain input variable and the membership functions together have some properties.

Some of the properties are characteristics for fuzzy sets but there are also connections to function approximation theory and to other methodologies. There is an increasing interest to unify the theory of basis functions and membership functions of fuzzy sets [2]. A *membership function* (MF) is a curve that defines how each point in the input space is mapped to the set of all real numbers from 0 to 1. This is really the only stringent condition brought to bear on a MF.

2.7.3.1TYPES OF MEMBERSHIP FUNCTIONS

Whereas there exist numerous types of membership functions; triangular, trapezoids, bell curves, Gaussian, and sigmoidal function, s-function and etc. However, most often used one is triangular and trapezoidal and bell-shaped membership functions [4]. These three choices can be explained by;

- the ease with which a parametric, functional description of the membership function can be obtained ,
- stored with minimum use of memory and
- Mathematically efficient, in terms of real time requirements by the inference engine.

2.7.3.2 PARAMETER CHARACTERIZING MEMBERSHIP FUNCTION

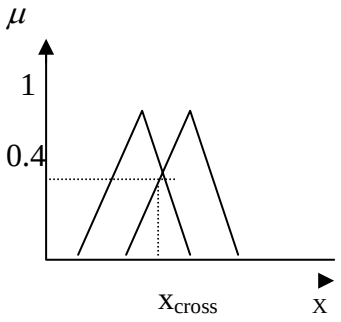
The following parameters characterize membership function [2] ;

Peak value:- Peak value is the value that attains highest membership value. In case of triangular membership function, peak value corresponds to one and only one value from domain of discourse whose membership value is 1, whereas in trapezoidal and bell shape the peak value is an interval. This affects the rate of confidence on a given value.

Left and right width;-the left width of *membership function($\mu(x)$)* is the length of interval from the peak value to the value in *X* which is located to the left of peak value and whose degree of membership is zero. The same is true for right width but to right of peak value. If the right and left width are equal the membership function is *symmetrical* otherwise it is *asymmetrical*. In case of symmetric membership function we have equal magnitude of the rate of gaining or losing our confidence on a given value from peak value. The reverse is true for asymmetrical membership function. [3]

Cross point and influence of cross point level;-let μ^1_{lx} and μ^2_{lx}be two membership function representing the meaning of two linguistic values of X which are elements of the term set lX.a cross point between μ^1_{lx} and μ^2_{lx} is that value in domain of discourse where $\mu^1_{lx}(x_{cross})=\mu^2_{lx}(x_{cross})>0$. Furthermore, a *cross point level* is defined by the degree of membership of x_{cross} i.e. $\mu^1_{lx}(x_{cross})$ which, by definition the cross point, is the same as $\mu^2_{lx}(x_{cross})$. Two membership functions defining the meaning of to different linguistic

value may have more than one cross point. The number of cross point between two membership function is called c*ross point ratio.*



cross point level=0.4 and cross point ratio=1

Fig 2.8 cross point and cross point level of triangular membership function

The mapping from a term set to membership function defined on the domain of discourse has to be such that the cross point level for every two membership function greater than zero. This means the crisp value of linguistic variable belongs to at least one membership function with degree of membership strictly greater than zero. *If it is not the case , then there will be a crisp input value of a linguistic variable which can not be matched during the fuzzification phase ,to a rule antecedent. Thus non of the rules will be fire and consequently, no value for control output will be computed. This in turn leads to discontinuity in the control output.* Further, more if the cross point ratio between two membership function is zero then only one rule at a time can fire.

For linear system up to third order and for symmetrical membership function there are some *optimal* value of the cross point level and ratio. That is if each two adjacent membership function has a *cross point level 0.5* and *cross point ratio 1,* then this provides for significantly *less overshoot, fast rise time and less undershoot* [2].

2.7.4 BASIC OPERATIONS ON FUZZY SETS

The three major operations on sets are union, intersection and complement. Since the notion of set membership in classical set theory has been generalized in to a matter of degree in fuzzy set , these operation should be extended accordingly.

Fuzzy logic reasoning is a superset of standard Boolean logic yet it still needs to use *logical operators* such as AND, OR and NOT. Firstly note that fuzzy logic differs from

Boolean yes/no logic in that although TRUE is given a numerical value '1' and FALSE numerical values '0', other intermediate values are also allowed. For example the values 0.2 and 0.8 can represent both not-quite-false and not-quite-true respectively. It will be necessary to do logical operations on these values that lie in the [0,1] set, but two-valued logic operations like AND, OR and NOT are incapable of doing this. For this functionality, the functions *min*, *max* and *additive complement* will have to be used.

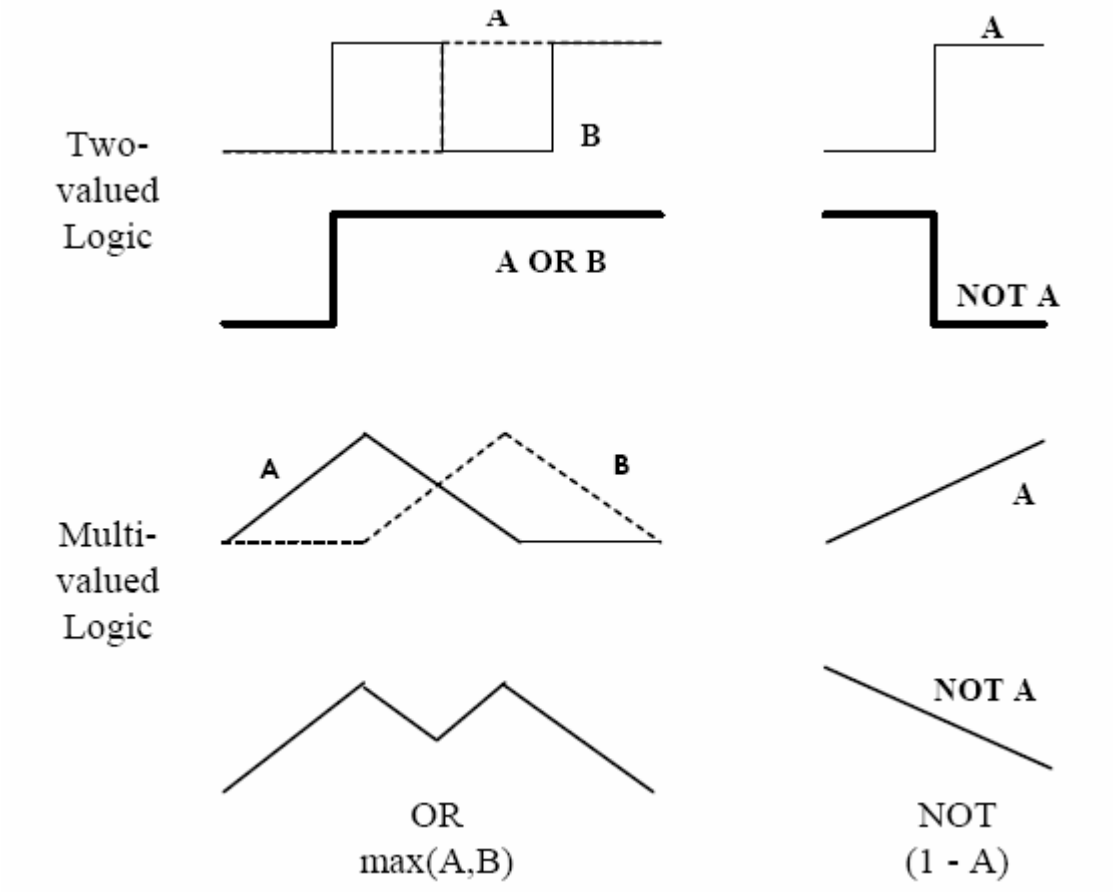


Figure 2.9 Examples of two-valued and multi-valued logical operations

2.8 LINGUISTIC VARIABLES, VALUES, HEDGES AND RULE BASES

Just like an algebraic variable takes numbers as values, a *linguistic variable takes* words or sentences as values. A linguistic variable enables its value to be described both qualititatively by *linguistic term or linguistic values* and quantitatively by corresponding *membership function* which express the meaning of the fuzzy set. The set of values that it can take is called its *term set (linguistic variable).* Linguistic term is used to express the concept and knowledge in human communication; whereas membership function is useful

for processing numeric input data. Each value in the term set is a *fuzzy variable* defined over a *base variable* or universe of discourse. The base variable defines the universe of discourse for all the fuzzy variables in the term set.

Linguistic variables are values defined by fuzzy sets. A linguistic variable such as 'High-Speed' for example could consist of numbers that are equal to or between 50km/hr and 80km/hr. The conditional statements that make up the rules that govern fuzzy logic behavior use these linguistic variables and have an *if-then* syntax. These if-then rules are what make up fuzzy rule bases. A sample if-then rule where A and B represent linguistic variables could be:

if x is A then y is B

The statement is understood to have both a premise, if 'x is A', and a conclusion, then 'y is B'. The premise also known as the *antecedent* returns a single number between 0 and 1 whereas the conclusion also known as the *consequent* assigns the fuzzy set B to the output variable y. Another way of writing this rule using the symbols of assignment '=' and equivalence '==' is:

if x == A then y = B

An if-then rule can contain multiple premises or antecedents. For example,

if velocity is high and road is wet and brakes are poor then…

Similarly, the consequent of a rule may contain multiple parts.

if temperature is very hot then fan is on and throughput is reduced

Among all the technique developed using fuzzy sets , fuzzy if then-rules are by far most visible due to their wide range of successful application. A fuzzy if-then rule associate a condition described using linguistic variables and fuzzy sets to a conclusion. From a knowledge representation view point, a fuzzy if then rule is a scheme of capturing knowledge that involves imprecision. The main feature of reasoning using fuzzy if then rule is its partial matching capability, which enables an inference to make from a fuzzy rule even when the rule's condition is only partially satisfied.

In classical logic if we know a rule is true and we also know antecedent of the rule is true, it can be inferred that the consequent of the rule is true. This is refer to as *modus ponens*.

Eg. If we know R_1 is true

R_1: *if* the annual income of a person is greater than $120,000 *Then* the person is rich.

We also know that the following statement is true

John's annual income is $121,000

Based on modus ponens, classical logic we can deduce that

John is rich

One of the limitations of modus ponens is that

- It can not deal with partial membership i.e. antecedent part does not represent smooth transition
- It does not deal with a situation where the antecedent of a rule (if part) is partially satisfied.

Eg. If John's annual income is $119,999, it can not infer any conclusion but in reality (as per human knowledge) John is still rich

Viewing such limitation, fuzzy rule based inference handle it by generalizing modus ponens to allow its inferred conclusion to be modified by the degree to which the antecedent is satisfied. For instance it handles the limitation on the above example by putting in the following way;

If *the annual income of a person is* ***high*** then *the person is rich.*

where **high** is a fuzzy set defined by the membership function.

Interpreting these rules involves a number of distinct steps.

1. **Firstly, the inputs must be *fuzzified*.** To do this all fuzzy statements in the premise are resolved to a degree of membership between 0 and 1. This can be thought of as the degree of support for the rule. At a working level this means that if the antecedent is true to some degree of membership, then the consequent is also true to that same degree.
2. **Secondly, fuzzy operators are applied** for antecedents with multiple parts to yield a single number between 0 and 1. Again this is the degree of support for the rule.
3. **Thirdly, the result is applied to the consequent.** This step is also known as *implication*. The degree of support for the entire rule is used to shape the output fuzzy set. The outputs of fuzzy sets from each rule are aggregated into a single output fuzzy set. This final set is evaluated (or *defuzzified*) to yield a single number.

The following example in Figure 2.10 shows how these steps are applied in practice by using MATLAB Fuzzy Logic Tool box..

Consider the two rules for a fuzzy model that evaluates lecture attendance:

1. If (LectureQuality) is Good) and (InterestOfMaterial is Interesting) then (Attendance is High)

2. If (LectureQuality is Poor) or (InterestOfMaterial is Boring) then (Attendance is Low)

These rules can then be seen visually in Figure2.10.

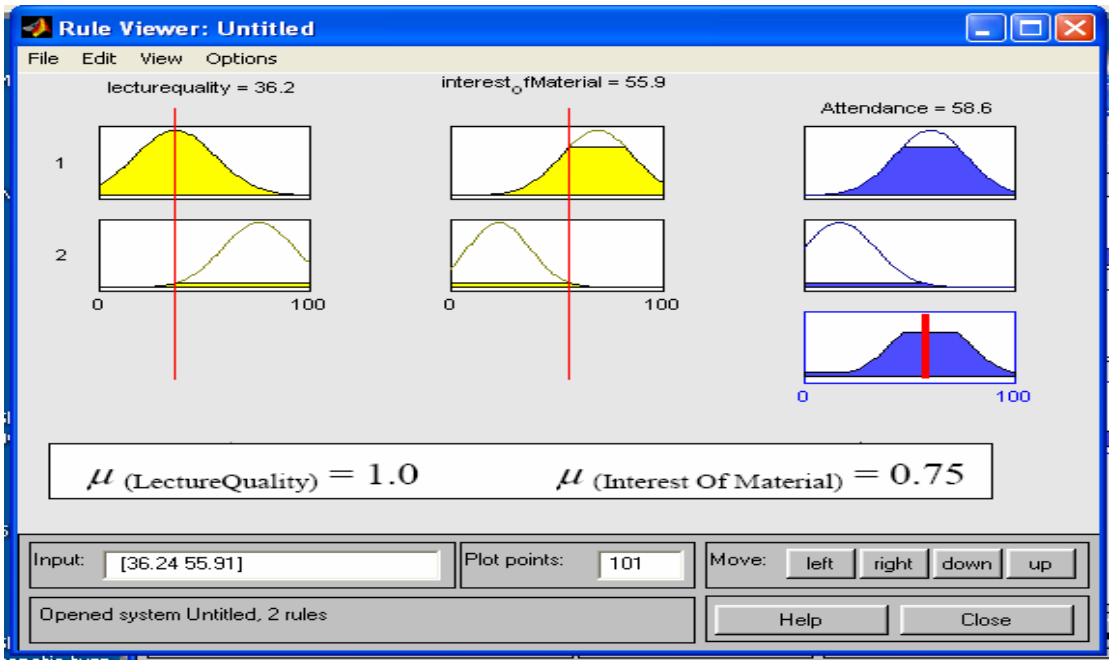


Fig 2.10 Application of rules for lecture attendance example

The yellow color Gaussian membership function indicates those rule fired and blue color indicates the implied clipped fuzzy set from two rules. The red color line indicates the difuzzified crisp value.

The fuzzy AND operator is applied in rule one and since the premise of the rule is true to a high degree then the consequent is also going to be true to a high degree. In this example both the fuzzy AND operator and the implication operator use the *min* function, hence for an input of 55.9% for LectureQuality and 36.2% for InterestOfMaterial, the defuzzified attendance percentage works out to be58.6%.

This comes from min (μLecture Quality, μInterest Of Material)

= min (1.0, 0.75)

= 0.75

Immediate advantages of this approach become apparent. Fuzzy sets can be combined using fuzzy rules to define system behavior and thus complex non-linear systems can be expressed linguistically. In fact, as will be shown later, rule tables can represent fuzzy controllers. The process of fuzzifying a single crisp input, applying fuzzy operators and then defuzzifying to produce a single crisp output is known as *fuzzy inference*. This progression of modeling is discussed in detail in the next section.

2.9 FUZZY RELATION AND COMPOSITION

A relation can be considered as a set of ordered pairs (tuples). In the same way a fuzzy relation is a fuzzy set of tuples i.e. each tuple has a membership function between 0 and 1. fuzzy relation generalize the notion of a classical black and white relation in to one that allows a partial membership. Since a relation can be viewed as a set we can easily generalize the classical notion of using fuzzy set .Classical binary relation can be represent relation R on x,y on domain X,Y as a function that maps an order pair (x,y) in X×Y to 0 if the relation does not hold between x and y or 1 if relation holds i.e. R= X×Y$\rightarrow${0 ,1}. Fuzzy relation generalizes the classical idea of relation in to a matter of degree. Therefore, fuzzy relation R between variables x and y whose universe of discourse on X,Y is defined by a function that maps ordered pair in X×Y to their degree in the relation, which is numbered between 0 and 1 i.e. R= X×Y$\rightarrow$[0 ,1].

For continuous universes it is denoted by

$$R = \int_{X\times Y} \mu_R(x,y) \Big/ (x,y)$$

For discrete universe it is denoted by

$$R = \sum_{X\times Y} \mu_R(x,y) \Big/ (x,y)$$

A fuzzy relation on X×Y can also be denoted as

$$R = \{(x,y), \mu_R(x,y)\ \}\ (x,y) \in X \times Y\ , \mu_R(x,y) \in [01]$$

$\mu_R(x,y)$ gives the degree of membership of the order pair (x,y) in R associating with each pair (x,y) in X×Y in interval [0 1]. The degree of membership indicates the degree to which X is in relation with Y.

If A and B are fuzzy sets in the universe of X×Y the fuzzy relation them has membership degree;

$$\mu_R(x,y) = \mu_{A\times B}(x,y) = \min[\mu_A(x), \mu_B(y)] \quad \forall x \in X \text{ and } \forall y \in Y$$

Two very important operations on fuzzy sets and fuzzy relation are *projection* and *cylindrical-extension*.

2.9.1 PROJECTION (PROJ)

This operation brings a ternary relation back to a binary relation, or a binary relation to a fuzzy set, or a fuzzy set to a single crisp value. In binary case it is simple. Let R be defined on X×Y then

$$proj\ \ Ron\ Y = \int \sup_x \mu_R(x,y) \Big/ y$$

This simply done by taking maximum from each column from X×Y, the same is true for proj R on X but from the row. The projection operation is almost always used in combination with cylindrical extension.

2.9.2 CYLINDRICAL EXTENSION

The cylindrical extension is more or less opposite of projection**.** It extends fuzzy set to fuzzy binary relation, fuzzy binary relation to fuzzy ternary relation, etc. it mainly serves the following goal: *let A be fuzzy set defined on X and R be a fuzzy relation defined on X×Y, then it is ,of course , not possible to take the intersection of A and R (since X is sub space of X×Y) but when A is extended to X×Y this is possible. This extension is done by cylindrical extension operator.*

Cylindrical extension of S into U is denoted by

$$ce(S) = \int_U \mu_S \ (x_i, ...x_{ik}) \Big/ (x_n, ...n)$$

CHAPTER -III
FUZZY MODELING

INTRODUCTION

Standard control techniques use numerical data to relate input and output signals. Fuzzy logic systems can use both numerical information and linguistic information to perform a mapping of an input to an output space. Consider the following diagram in Figure 1.6.

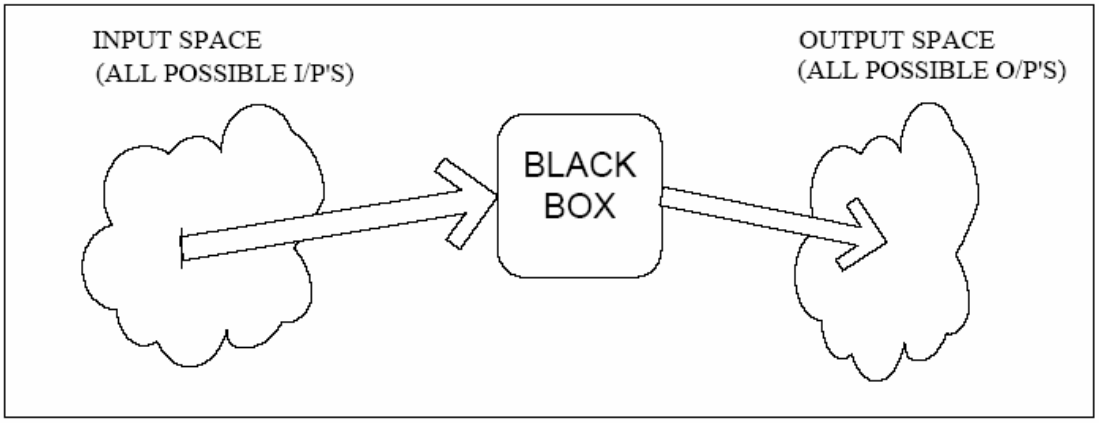


Fig 3.1 Mapping of input space to output space

Many different control mechanisms could reside within the black box but in this case the mechanism will be confined to a fuzzy logic system. Since the objective is to map inputs to outputs it becomes possible to model non-linear systems, even complex ones. This is one of fuzzy logics' greatest advantages. Put differently, fuzzy logic systems are tolerant of imprecise data. When considered this suits many real-world applications well because as real-world systems become increasingly complex often the need for highly precise data decreases. The rules that govern this mapping can be acquired through two methods. The first is a method called the *direct approach* and the second is by using *system identification*. The direct approach involves the manual formulation of linguistic rules by a human expert. These rules are then converted into a formal fuzzy system model. The problem with this approach is that unless the human expert knows the system well it is very difficult to design a fuzzy rule base and inference system that is workable, let alone efficient. For complex systems (non-linear for example) tuning these membership

functions would require the adjustment of many parameters simultaneously. Understandably no human expert could accomplish this

All of the previous elements of fuzzy logic that have been discussed up to this point are put together to form a fuzzy inference system (FIS). Two main types of fuzzy inference system exists the Mamdani and Sugeno type. They are both introduced in the following sections.

3.2 Mamdani Modeling

Owing its name to Ebrahim Mamdani the Mamdani model was the first efficient fuzzy logic controller designed and was introduced in 1975 [2]. The controller consists of a fuzzifier, fuzzy rule base, an inference engine and a defuzzifier. It is shown in the Figure 3.2 below.

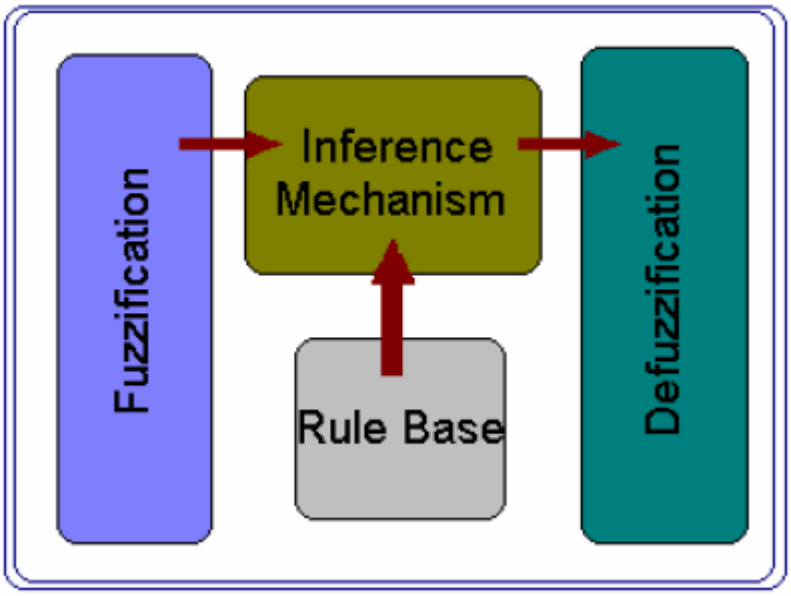


Fig 3.2 Mamdani fuzzy modeling

Conventional control systems require crisp outputs to result from crisp inputs. The above representation shows how a crisp input in R can be operated on by a fuzzy logic system to yield a crisp output. This Mamdani controller is realized using the following steps.

A. Fuzzification of Inputs

B. Application of Fuzzy Operators

C. Application of Implication Method

D. Aggregation of all Outputs

E. Defuzzification of Aggregated Output

A. Fuzzification of Inputs

The fuzzifier maps crisp input numbers into fuzzy sets. The value between 0 and 1 each input is given represents the degree of membership that input has within these output fuzzy sets. Fuzzification can be implemented using lookup tables or as in this report, using membership functions.

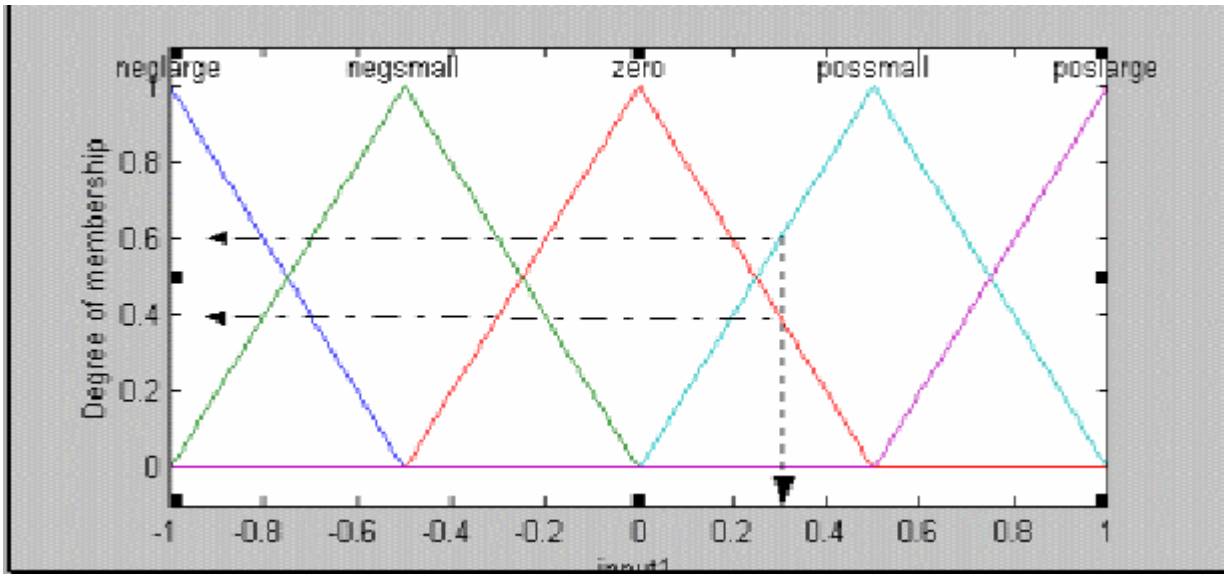


Fig 3.3. An example of an input membership function with an illustration of fuzzification of non-fuzzy input signal 0.3.

A numerical value 0.3, for example, is converted into its relevant fuzzy points.

B. Application of Fuzzy Operators

In the case where multiple statements are used in the antecedent of a rule, it is necessary to apply the correct fuzzy operators as explained in 2.5.4. This allows the antecedent to be resolved to a single number that represents the strength of that rule.

C. Application of Implication Method

This part of the Mamdani system involves defining the consequence as an output fuzzy set. This can only be achieved after each rule has been evaluated and is allowed contribute its 'weight' in determining the output fuzzy set.

D. Aggregation of all Outputs

The fuzzy outputs of each rule need to be combined in a meaningful way to be of any use. Aggregation is the method used to perform this by combining each output set into a single output fuzzy set. The order of rules in the aggregation operation is unimportant as all rules are considered. The three methods of aggregation available for use include *sum* (sum of each

rules output set), *max* (maximum value of each rule output set) and the *probabilistic OR* method (the algebraic sum of each rules output set). An example of the aggregation process using the max operator can be seen in Figure 3.4 below.

E. Defuzzification of Aggregated Output

The aggregated fuzzy set found in the previous step is the input to the defuzzifier.

As indicated in the model shown in Figure 1.8 this aggregated fuzzy set mapped to a crisp output point. This crisp output is a single number that can usefully be applied in controlling the system. A number of methods of defuzzification are possible and these include the mean of maximum, largest of maximum, smallest of maximum and centroid (centre of area) methods. The centroid method is the most widely used and can be seen in Figure 1.8 below.

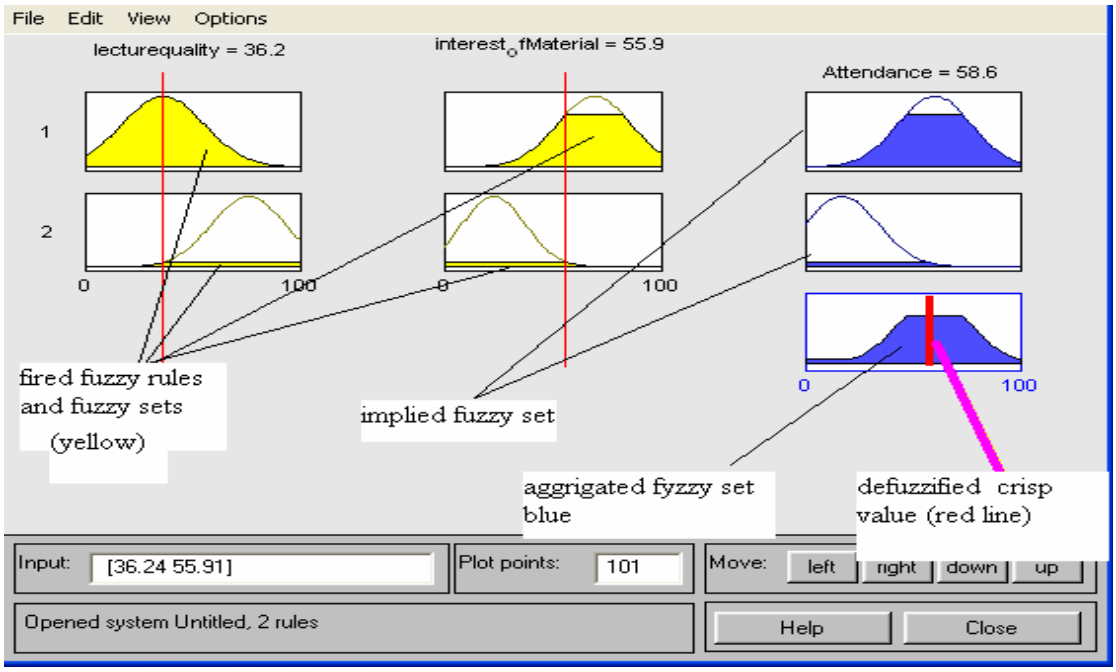


Fig 3.4 Diagram showing aggregation and defuzzification

Advantages of the Mamdani Method

•It's intuitive.

•It has widespread acceptance.

•It's well-suited to human input.

3.3 Sugeno Modeling

The Sugeno fuzzy model or more fully the Takagi-Sugeno method of fuzzy inference was first introduced in 1985 [4]. In many respects it is identical to the Mamdani method except that the output membership functions for the Sugeno method are always linear or constant. The output membership functions can be thought of as singleton spikes that undergo a simple

aggregation instead of other aggregation methods such as *max*, *sum*. In this section, we discuss the so-called Sugeno, or Takagi-Sugeno, method of fuzzy inference. Introduced in 1985 it is similar to the Mamdani method in many respects. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant. A typical rule in a Sugeno fuzzy model has the form

If Input 1 = x and Input 2 = y, then Output is z = ax + by + c

For a zero-order Sugeno model [33], the output level z is a constant (a=b =0).The output level zi of each rule is weighted by the firing strength wi of the rule. For example, for an AND rule with Input 1 = x and Input 2 = y, the firing strength is wi = AndMethod (F1(x),F2(y))where F1,2 (.) are the membership functions for Inputs 1 and 2. The final output of the system is the weighted average of all rule outputs, computed as

$$\text{Final Output} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}$$

A Sugeno rule operates as shown in the following diagram.

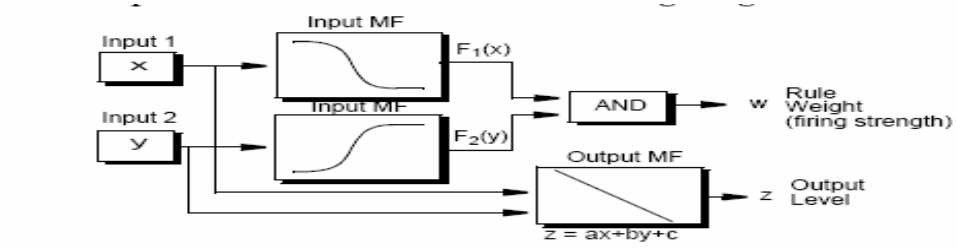


Fig.3.5 rule operation of sugeno inference

The rules for tipping problem [6] is shown below

1.If (service is poor) or (food is rancid) then (tip is cheap) (1)

2. If (service is good) then (tip is average) (1)

3. If (service is excellent) or (food is delicious) then (tip is generous) (1)

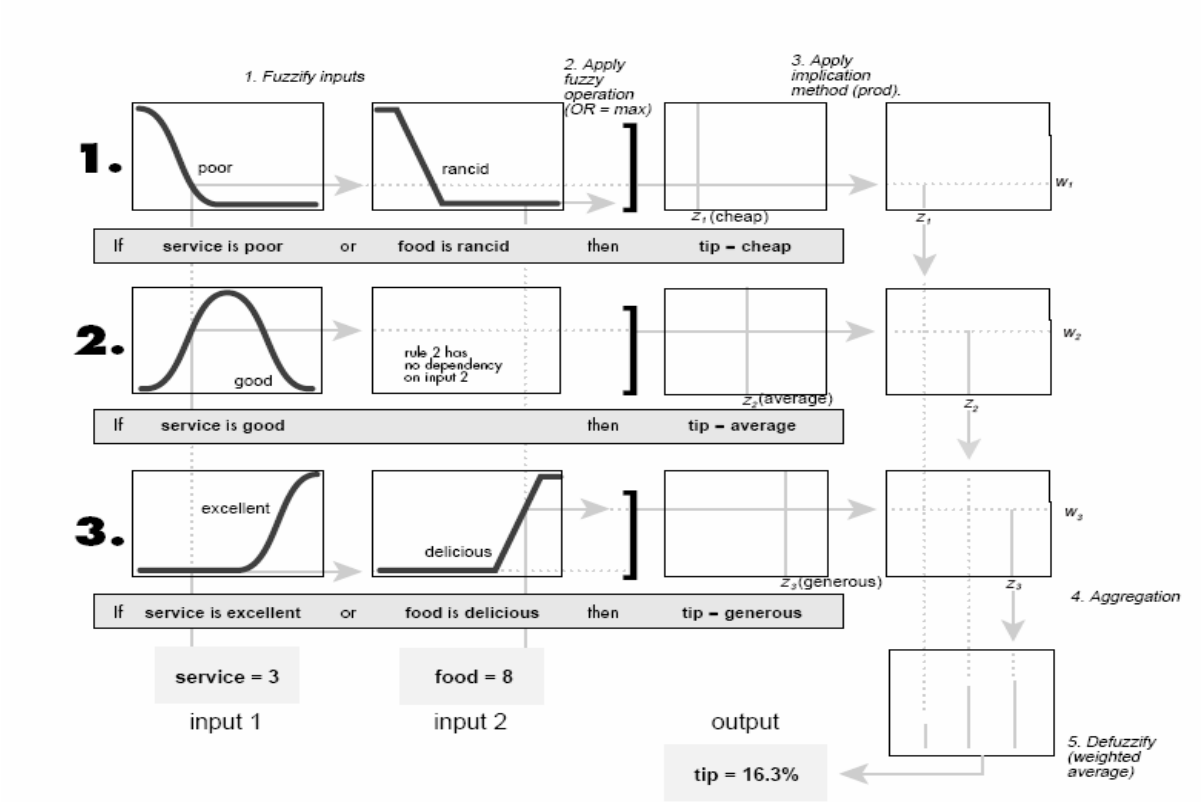


Fig 3.6 sugeno inference

The Sugeno system is computationally efficient and its ability to interpolate multiple linear models makes it particularly suited to modeling non-linear systems.

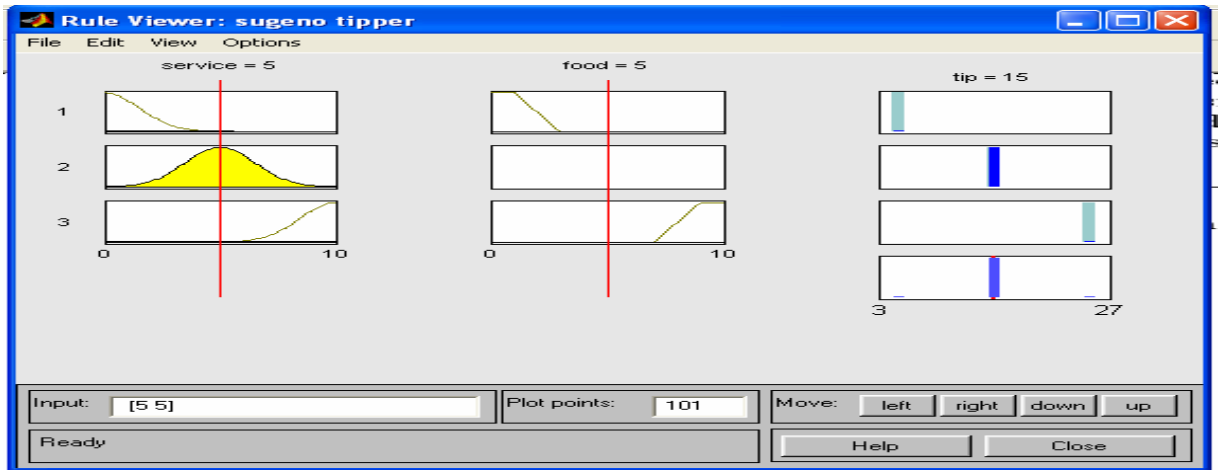


Fig 3.7 Implementation of a Sugeno model in matlab

Advantages of the Sugeno Method

•It's computationally efficient.

•It works well with linear techniques (e.g., PID control).

•It works well with optimization and adaptive techniques.

CHAPTER-IV

FUZZY CONTROLLER DESIGN

INTRODUCTION

A fuzzy controller is a fuzzy system, which is used to control a target system or it is used for supervisory control. The fuzzy controller has a linguistic interpretation, which can be expressed with help of fuzzy sets, membership functions, and fuzzy rules. However, it processes exact input data and produces exact output data in a deterministic way.

Fuzzy controllers can be used when nonlinear control action is needed, or when the controller is to be tuned manually. In many cases where fuzzy control is applied the control has been performed before manually.

Fuzzy controller is a static mapping $F:\Re^{nx} \rightarrow \Re^{nz}$ between the inputs x and the outputs z.

In this thesis we will show how the fuzzy control design methodology can be used to construct fuzzy controllers for challenging real-world applications. As opposed to "conventional" control approaches (e.g., proportional-integral-derivative (PID), lead-lag, and state feedback control) where the focus is on modeling and the use of this model to construct a controller that is described by differential equations, in fuzzy control we focus on gaining an intuitive understanding of how to best control the process, then we load this information directly into the fuzzy controller.

Fuzzy Logic can be applied to control, and when it is, is known as Fuzzy Control. Fuzzy Control is made up of control rules which mimic those used by humans when they control or operate machinery: "If you need to go a little bit faster, push the accelerator pedal slightly". Fuzzy Control can be an especially effective way of controlling non-linear systems when expert human knowledge of the system is available.

For instance, in the cruise control example we may gather rules about how to regulate the vehicle's speed from a human driver. One simple rule that a human driver may provide is "If speed is lower than the set-point, then press down further on the accelerator pedal." Other rules may depend on the rate of the speed error increase or decrease, or may provide ways to adapt the rules when there are significant plant parameter variations (e.g.,

if there is a significant increase in the mass of the vehicle, tune the rules to press harder on the accelerator pedal). For more challenging applications, control engineers typically have to gain a very good understanding of the plant to specify complex rules that dictate how the controller should react to the plant outputs and reference inputs.

Basically, while differential equations are the language of conventional control, heuristics and "rules" about how to control the plant are the language of fuzzy control. This is not to say that differential equations are not needed in the fuzzy control methodology.

4.2 FUZZY CONTROL CONSIDERATIONS

Fuzzy logic control considerations can be categories in to theoretical and practical reasons [3]

a. Theoretical Reason for fuzzy control

- As general rule a good engineering approach should be capable of making effective use of all the available information. If the mathematical model of a system is too difficult to obtain (this is true for many practical system) , then the most important information come from two sources: 1) sensors which provide numerical measurements of key variable sand 2) human experts who provide description about the system and control instructions. Fuzzy controllers, by design, provide a systematic and efficient frame work for incorporating linguistic fuzzy information from human experts. Conventional controller, however, can not incorporate the linguistic fuzzy information into their design. If in some situation the most important information come from human experts, fuzzy control is the best choice.
- Fuzzy control is model free approach i.e. it does not require mathematical of the system under control. Control engineer now facing more and more complex systems, and the mathematical model of these system are increasingly difficult to obtain. Thus, model free approaches have taken on added importance. Conventional control has also some model free approaches, e.g. nonlinear adaptive control and PID control [3] fuzzy control provides yet another model free approach.
- Fuzzy control provides nonlinear controllers .e. these fuzzy logic controllers are general enough to perform any nonlinear fuzzy controllers, it is always possible to design a fuzzy controller that is suitable for nonlinear system under control.

b. Practical reasons for fuzzy control;

- It is easy to understand. Because fuzzy controller emulates control strategy, the underlying principle can be easily understood by those who are not control specialists. There fore, practical engineers who are in the front line of designing consumer products tend to use approaches that are simple and easy to understand. Fuzzy control is just such an approach.
- It is simple to implement. Fuzzy logic systems, which are at the heart of fuzzy control, admit a high degree of parallel implementation. Many fuzzy VLSI chips have been developed which make the implementation of fuzzy controllers simple and fast [1].
- It is inexpensive to develop. From a practical point of view, the development cost is one of the most important criteria for a successful product. Because fuzzy control is to understand, the time necessary to learn the approach is short; i.e. the software cost is low. Also, because fuzzy control is simple to implement, the hardware cost is also low. What is more, there are software tools available for designing fuzzy controllers. Thus, fuzzy control is an approach that has a high performance/cost ratio.

4.3 **FUZZY CONTROL SYSTEM DESIGN**

Design of the fuzzy controller means selection of fuzzy rule base structure, including the number of fuzzy sets for each input and output. After that places and shapes of the membership functions are tuned to obtain behavior of the controller as wanted.
Often the tuning must be done on a trial-and-error basis, which is time-consuming and needs patience.
With fuzzy logic, very versatile control strategies can be implemented. Thus the design of the fuzzy controller includes many choices, and therefore each selection is discussed in the following subsections.

What, then, is the motivation for turning to fuzzy control? Basically, the difficult task of modeling and simulating complex real-world systems for control systems development, especially when implementation issues are considered, is well documented.
Even if a relatively accurate model of a dynamic system can be developed, it is often too complex to use in controller development, especially for many conventional control

design procedures that require restrictive assumptions for the plant (e.g., linearity). It is for this reason that in practice conventional controllers are often developed via simple models of the plant behavior that satisfy the necessary assumptions, and via the ad hoc tuning of relatively simple linear or nonlinear controllers. Regardless, it is well understood (although sometimes forgotten) that heuristics enter the conventional control design process as long as you are concerned with the actual implementation of the control system. It must be acknowledged, moreover, that conventional control engineering approaches that use appropriate heuristics to tune the design have been relatively successful.

4.4 STRUCTURE OF FUZZY CONTROLLER

The fuzzy controller block diagram is given in Figure 3.1, where we show a fuzzy controller embedded in a closed-loop control system. The plant outputs are denoted by $y(t)$, its inputs are denoted by $u(t)$, and the reference input to the fuzzy controller is denoted by $r(t)$.

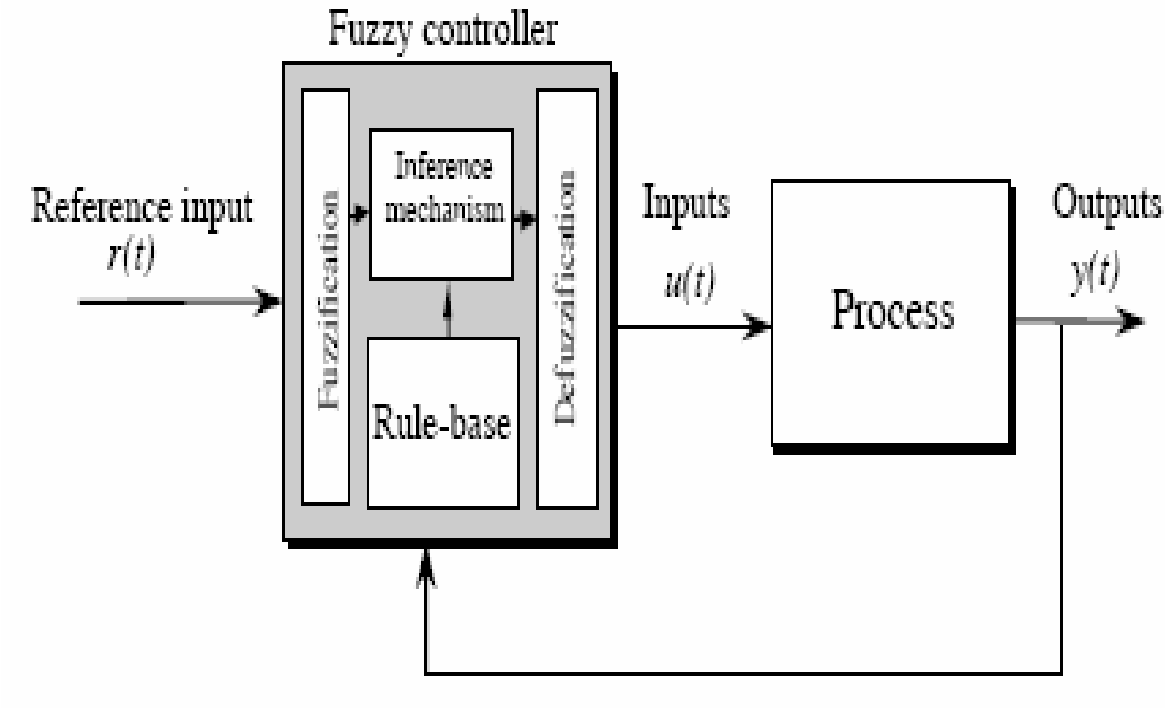


Fig. 4.1 Fuzzy controller architecture

The fuzzy controller has four main components:

4.4.1 A FUZZIFICATION INTERFACE,

Fuzzification interface converts the controller input in to information the inference mechanism can easily use to activate and apply the rules. It establish a mapping between crisp input value in the input domain and fuzzy set defined in the same universe of discourse, convert numeric input into a form that the fuzzy inference mechanism can use to determine which knowledge in the rule is most relevant at current time. It simply modifies the inputs so that they can be interpreted and compared to the rule in the rule base. The fuzzification interface allows the fuzzy rule based system to deal with real inputs and out puts. Its aim is to define a mapping that establishes a correspondence between each value in the crisp input space and a fuzzy set define in the universe of discourse of this input, obtaining the membership function associated with each one of the system input. Symbolically this component work as

$A=F(x_o)$ with $\mathbf{x_o}$ being a crisp input value for the fuzzy knowledge base system defined in the universe of discourse.**A** being a fuzzy set define in the same domain and **F** being fuzzification operator.

There are two main possibilities for choosing the operator **F;**

i. Punctual fuzzification:- **A** is built as a punctual fuzzy set (a singleton) with support $\mathbf{x_o}$ **.** that is it represents the membership function

$$\mathbf{A}(x)=\begin{cases}1 & \text{if } x=\mathbf{x_o}\\ 0 & \text{otherwise}\end{cases}$$

This method is most used due to simplicity

ii. non punctual or approximate fuzzification :- in this case $\mathbf{A}(x)=1$ and the membership function of the remaining values decrease when moving away from $\mathbf{x_o}$. This second kind of operator allows us to deal with different type of membership functions. For example triangular shape fuzzy set can be obtained by defining

$$\mathbf{A}(x)=\begin{cases}1-\dfrac{|x-\mathbf{x}_o|}{\delta} & \text{if } |x-\mathbf{x_o}|\le\delta\\ 0 & \text{otherwise}\end{cases}$$

To summarize fuzzification module perform the following functions:

- measure the value of input variable
- perform a scale transformation (normalization) that maps the physical measure value in to normalized domain
- Using membership function, converts the current value of a process state variable in to a fuzzy set to make it compatible with the fuzzy set representation of the process state variable in the rule antecedent.

4.4.2 RULE BASE

The rule base holds the knowledge, in the form of a set of rules ,of how best to control a system. It is a set of IF…THEN rules which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve a good control. Rule base is used to model human problem solving activity behavior by the use of if-then rule i.e. satisfaction of the rule antecedent gives rise to the execution of the consequent; that is one action is performed.

Property of rule base

- *Completeness* : A set of *if-then* rules are complete if any combination of input value results in output value, i.e. whether there are conclusion for every fuzzy control input. It has to be noticed that almost no rule base in practical application of fuzzy knowledge base controller is complete. This has to do with the fact that certain region of the input domain to controller are not interest (due to lack of firing strength). For instance, the well known inverted pendulum problem performs optimally only ten or twelve from twenty-five rule [2].
- *Consistency:* a set of if then rule is consistent if it does contain contradiction, i.e. whether the conclusion that rule make conflict with other rule. A set of *if then* rule is inconsistent if there are two rules with the same rule antecedent but different rule consequent.

  Reasoning using fuzzy rule has three major features;

  a. It enables a rule that partially matches the input data to make an inference

b. It typically infers the possibility distribution of an out put variable from the possibility distribution from input variable
c. The system combine the inferred conclusions from all rules to form an over all conclusion.

The data base in the knowledge base provide the necessary information for the proper function of fuzzification module, inference engine and defuzzification

This information includes fuzzy set(membership function) that represent the meaning of the linguistic values of the process state and the control output variable. Physical domains and their normalized counterparts together with the normalization (scaling) factor.

To design fuzzy controller the control engineer must gather information on how the artificial decision maker (operator of the plant) should act in closed loop system.

This information can come from:

- Expert experience and control engineering knowledge:- The most common approach to establishing such a collection of rules of thumb, is to question experts

  Or may come from operators using a carefully organized questionnaire i.e. from human decision maker who perform the control task (plant operator)

- The control engineer can come to understand the plant dynamics and write down a set of rules about how to control the system with outside help.
- Self learning: The self-organizing controller is an example of a controller that finds the rules itself. Neural networks are another possibility.

These rules basically say" *If the plant out put and reference input are behaving in a certain manner, then the plant input should be some value"*. A whole set of such IF-THEN rules is loaded into rule base , and an inference strategy is chosen ,then the system is ready to be tested to see if the closed loop specification are met.

4.4.3 INFERENCE MECHANISMS

It is also called inference engine, which emulate the experts' decision making in interpreting and applying knowledge about how best to control the plant. The inference mechanism evaluates which control rules in rule base are relevant at the current time and then decides what the input to the plant should be i.e. it apply the action indicated by this

rule. In this component the membership value obtained in the fuzzification steps are combined through T-norms, usually multiplication or minimization, to obtain the firing strength of each rule. Each rule characterizes the control goal and control policy of the domain experts by means of a set of linguistic control rules. Then depending on the firing strength, the consequent part of each qualified rule is generated. The inference engine using the fuzzified inputs and the rules stored in the rule base processes the incoming data and produces an (fuzzy) output

There are two basic types of inference mechanism;

a. *Composition based inference*:- in this case , the fuzzy relation representing the meaning of each individual rule are aggregated into one fuzzy relation describing the meaning of the over all set of rules. Then inference or firing with this fuzzy relation is performed via the operation composition between the fuzzified crisp input and the fuzzy relation representing the meaning of over all set of the rules. As a result of composition one obtain the fuzzy set describing the fuzzy value of the overall control output.

b. *Individual rule based inference:-* In this case , first each single rule is fired. This firing can be simply described by;

1 .Computing the degree of match between the crisp input and the fuzzy set describing the meaning of rule antecedent

2 .”Clipping “the fuzzy set describing the meaning of rule consequent to the degree to which the rule *antecedent* matched by the crisp input. Finally the “clipped” value of the control output of each rule are aggregated, thus forming the value of the over all control output.

4.4.4 DEFUZZIFICATION INTERFACE:

For use in the fuzzy control environment defuzzification step is needed. We need a crisp single value to be the input to the controller system. The output fuzzy set inferred by the rule base and inference engine can not be used directly as input to the controlled deterministic system. In order to obtain a crisp value from the output of the FLC we are faced the problem of selecting one element y* from universe Y to represent the value to implement. This process of selecting one representative crisp element based up on the knowledge that the fuzzy value of the output variable is called *defuzzification.*

Defuzzification interface combines the conclusion reached by the fuzzy interface mechanism and provide a numeric value as an output. Because the system must give a crisp output, the defuzzification interface has to develop the task of aggregating the information provided by each one of the fuzzy set and transform it into a single crisp value. It defines a mapping between fuzzy set defined in the output domain and crisp value defined in the same universe of discourse

4.5 DESIGN PROCEDURE OF FUZZY CONTROLLER.

The main feature of fuzzy logic control is that a control engineering base typically in terms of a set of rules created using experts knowledge of process behavior, is available within the controller and the control action are generated by applying existing process condition to the knowledge base making use of inference mechanism. The design of fuzzy controller needs through understanding of fuzzy logic. Careful attention must be given in choosing design parameters and their effect in the controller performance.

The following questions should be addressed during design process

- What are the best membership functions?
- How many linguistic values and rules should be used?
- Should the minimum or product operator be used to represent the" and" in premises and which should be used to represent implication?
- What defuzzification method should be chosen?

Things that one should take into consideration while designing fuzzy control system

- All rules are evaluated in parallel, so the order of the rules is unimportant
- Define all the terms you plan on using the adjectives that describe them
- Define the range over which the variable can be expected to vary as well as the meaning of linguistic value.
- In classical binary logic if the premise is true then the conclusion is true but in fuzzy logic if the antecedent is true to some extent (degree) then the consequent is also true to that same degree. This is because we can not be surer about the outcome than premises (input).

4.5.1 DESIGN STEPS OF FLC

Step 1: choice of fuzzy controller input and out put

This step needs identification of the input and output of fuzzy controller using linguistic variable, universe of discourse of input and output variables Fuzzy control is assumed to be the most suitable for non-trivial control tasks. Thus, the selection of the input and the output variables of the fuzzy system may be nontrivial also.

Usually the design problem is well-defined with respect to the output variables of the fuzzy system, i.e., the signals u(k) which affect the process output y(k) are known.

If that is not the case, careful process analysis is needed before any controller, even fuzzy controller, can be considered. Another part is the selection of the input variables for the fuzzy controller. In practice, there are several signals, which should be taken into account when the control signal is calculated.

In feedback control, the error signal between the set-point and the measurement

$$e(k) = yr(k) \_ y(k)$$

is observed. The control objective is to keep the error signal small. Usually the changing rate of the error signal in the form of the change in the error

$$\Delta e(k)=e(k)-e(k-1)$$

is also considered. The signs of the change and the error indicate, if the process output is going towards the set-point or not. With those two inputs, the fuzzy system can perform PI or PD type control depending on whether the output is the change in the control signal $\Delta u(k)$ or the pure control signal u(k).

In the selection of input variables, it is useful to restrict the number of variables because the more inputs the rule base has, the larger it is. The minimum number of rules grows exponentially 2^n in relation to the number of inputs n. In real applications the total number of rules is much more than 2^n because nonlinearities are not easy to generate with only two fuzzy sets for each input. Thus a practical limit for the number of inputs is three or four. If the input variables affect the system independently of each other, the rule base can be divided into two or more rule bases with less input in order to decrease the total number of rules.

Step 2: selection or choice of the input normalizing factor and output de-normalizing factor.

The use of normalized domains requires a scale transformation that maps the physical values of the process state variable in to a normalized domain. Choice of scaling factors here plays a role similar to that of gain coefficients in conventional control and may affect the performance and stability issues.

Step 3: design of each of the four component of fuzzy controller

These four components are;

1. ***Fuzzification interfac***e: convert point-wise (crisp) value into fuzzy set. Here we have considered choice of membership function i.e.
   -Choice of its shape
   -Influence of peak value, left and right width, cross point and symmetry.

The design parameter of fuzzification module is choice of fuzzification strategy which is depend up on inference engine or rule firing employed and are only two choices are available:

a. fuzzification on the case of composition based
b. individual rule firing

2. ***Design of rule base*****:** holds the knowledge in the form of if-then rule. These needs writing of rules that controller to follow in order to meet design specification.

In constructing the rule base, the numerical completeness must be kept in mind in order to prevent dividing zero by zero in the center-of-gravity defuzzification. It is easily caught by including all combinations of the input fuzzy sets into the rule base by means of fuzzy AND connectives. The number of rules demanded can be decreased by dropping some fuzzy conditions away from the antecedent.

The rules may come from;

-by interrogation of process operator/control engineer using carefully organization or

-by introspective verbalization of experience based knowledge (this used in the design of water heating system)

The design parameters involved in the construction of the rule base includes:

- Choice of contents of rule antecedent and rule consequent
- Choice of term set (range linguistic value)for the process state and control variable
- Derivation of control rule
- Choice of process state and control

3. ***Design of inference engine***: used to evaluate which control rules are relevant at he current time and decides what the plant should be

Two basic type of approaches are employed in the design

-composition based inference (firing)

-individual rule based inference (firing)

The design parameters for the inference engine design are

- Choice of representing the meaning of a single production rule
- Choice of representing the meaning of the set of rule
- Choice of inference engine
- Testing the set of rules for consistency and completeness

4 ***Design of deffuziffication module***: used to convert the conclusion reached by the inference mechanism in to crisp input to the plant

Design parameter of deffuzification module is

- Choice of defuzzification operator

The most commonly used defuzzification methods are

-center of gravity (COG)

-center average

step4: Off-line debugging, testing and verification. Test for completeness and non-ambiguity of the system. If a software simulation or sample data of the process exist, it is used in this step.

Step 5. On-Line Debugging. Connect the fuzzy logic system to the process under control and analyze its performance in operation. Because fuzzy logic lets you modify the system in a straightforward way from the performance you observe, this step can rapidly expedite system design.

Advantage of fuzzy logic based controller

It takes into account the transition from membership to non membership area is gradual rather than abrupt results smooth control

- Concerned in the main with imprecision and uncertainty of data
- Important tool for modeling complex system in which due to complexity or imprecision where classical tools are unsuccessful.
- Inference method become more robust and flexible with approximate reasoning method of fuzzy logic
- Use of heuristic technique to construct non linear controller.
- Reduction of development time;-fuzzy control system, which work at two levels of abstraction, offer languages at both level of expertise. The symbolic level is appropriate describing the application engineer's strategies, whereas the compiled level is well understood by the control engineers. Because there is a well defined formal translation between this two levels, a fuzzy bases approach can help in reduction of communication problem.
- It is easily tuned by changing rules or using scaling factor or membership function online.

Limitations of fuzzy control system

- At present, there is no systematic procedure for the design of fuzzy control system. The most straight forward approach is to define membership function and decision rule subjectively by studying an operating system or existing controller
- In case of very complex system, the proper decision rule can not easily be derived by human expertise.
- Designing and tuning a multi input –multi output (MIMO) fuzzy control system is so tedious as to be unfeasible.

- In some situation, reliable expert knowledge may not be available. Even relying on expert knowledge, fine tuning or achieving the optimal fuzzy control system not trivial task.

Some significant operating changes i.e. disturbance or parameter changes might be outside expert's experience.

Obtaining rule from an expert (knowledge elicitation) is one of the major bottlenecks in the development of fuzzy control system. Frequently, the fuzzy algorithms provided by experts are incorrect, irrelevant and incomplete. This problem can be overcame by adaptive fuzzy control system which automatically find appropriate set of rule and membership function (self learning (training) fuzzy control system)

4.6 CONCLUSION

Fuzzy controller is considered as a fuzzy computing system, which is designed for a control application. Selections of input and output variables are discussed. It is advisable to restrict the number of inputs to below four. In practice, numerical completeness of the rule base must be provided. Product-sum composition and normalized triangular shaped membership functions are usually sufficient. If each rule can have an individual consequence, the fuzzy controller can be parameterized with the cores of the membership functions. Thus the number of parameters is small but the fuzzy controller can approximate any nonlinear control action with sufficient accuracy.

Rules of the thumb are given for the selection of the fuzzy controller parameters.

The input membership functions should cover the whole varying interval of the inputs and the inner membership functions should be placed according the shape of the nonlinearity to be approximated. If a conventional controller exists, the fuzzy controller can be initially tuned with respect to the conventional controller.

Instead of being more robust or performing better, or being easier to design than a conventional controller, for fuzzy controller's systematic design and tuning methods are lacking. The success of fuzzy control is based on the easy user-interface in the form of linguistic rule base. Less skilled personnel are needed for the design and the maintenance and rough working applications can be developed within a short developing period. Fuzzy computing does not require any special computing power; therefore it is also applicable in cheap consumer products.

CHAPTER-V
GENETIC ALGORITHM (GA)

INTRODUCTION

Searching or optimizing algorithms inspired by biological evolution are called *evolutionary computations*. The features of the evolutionary computation are that its search or optimization is conducted;

- based on multiple searching points or solution candidates (population based search),
- using operations inspired by biological evolution, such as crossover and mutation,
- based on probabilistic search and probabilistic operations, and
- using little information of searching space, such as differential information.

Genetic algorithms are computerized search and optimization algorithms based on the mechanics of natural genetics and natural selection. Essentially, GA is an optimization technique that performs parallel, stochastic, but direct search method to evolve the fittest population. GA mimics the concept of natural genetics and natural selection to constitute search and optimization procedure. GA drives their name from the genetic process of natural evolution. Genetic algorithms have been developed by Professor John Holland, his colleagues, and his students at the University of Michigan in mid 1960s. it is applied and implemented successfully in a broad range of control applications for example the design of neural and fuzzy controller for tuning of industrial controllers and for the creation of hybrid fuzzy –genetic and neural-genetic controller. The techniques of GA have in common the emulation of the natural evolution of individual structure through process inspired from natural selection and reproduction. These process depends on the

fitness of the individual to survive in a hostile environment according to Darwinian principle of" *survival of the fittes*t". The idea is to have a pool of candidate solutions evaluated in parallel, from which the "fittest" solutions are chosen to mate and breed new candidate solutions using stochastic operators. This procedure is iterated until the population converges or a preset condition is met.

The terminology in the field of Genetic algorithm is derived from Biology and genetics. Thus, the candidate solutions of an optimization problem are termed *individuals*. The *population* of solution evolves in accordance with the laws of natural evolution. After initialization, the population undergoes *selection, recombination and mutation* repeatedly until some termination condition is satisfied. Each iteration termed a Generation while the individual that undergoes recombination and mutation are named parents and yields offspring.

The corner stone of Genetic algorithm is the iterative procedure in exploring the search space while simultaneously exploiting the information that being accumulated during the search. This is in fact, where their function lies. Through exploration, a systematic sampling of the search space is achieved, while through exploitation the information that has been accumulated during exploration is used to search the new area of interest. The central theme of research on genetic algorithms has been *robustness*, the balance between efficiency and efficacy for survival in many different environments. The implications of robustness for artificial systems are main fold. If artificial systems can be made more robust, costly redesigns can reduced or eliminated. If higher levels of adaptation can be achieved, existing systems can perform their functions longer and better. Genetic algorithms are theoretically and empirically proven to provide robust search in complex search. Genetic algorithms are computationally simple yet powerful in their search for improvement. Further more, they are not fundamentally limited by restrictive assumptions about the search space (assumptions concerning continuity, existence of derivatives, unimodality, and other matters)

5.2 BIOLOGICAL BACKGROUND

5.2.1. Chromosome

All living organisms consist of cells. In each cell there is the same set of **chromosomes**. Chromosomes are strings of DNA and serve as a model for the whole organism. A chromosome consists of **genes**, blocks of DNA. Each gene encodes a particular protein. Basically, it can be said that each gene encodes a **trait**, for example color of eyes. Possible settings for a trait (e.g. blue, brown) are called **alleles**. Each gene has its own position in the chromosome. This position is called **locus**.

Complete set of genetic material (all chromosomes) is called **genome**. Particular set of genes in genome is called **genotype**. The genotype is with later development after birth base for the organism's **phenotype**, its physical and mental characteristics, such as eye color, intelligence etc.

5.2.2. Reproduction

During reproduction, **recombination** (or **crossover**) first occurs. Genes from parents combine to form a whole new chromosome. The newly created offspring can then be mutated. **Mutation** means that the elements of DNA are a bit changed. These changes are mainly caused by errors in copying genes from parents.

The **fitness** of an organism is measured by success of the organism in its life (survival). In principal, a GA can be applied to any problem where the variables to be optimised (``genes") can be encoded to form a string (``chromosome") - as sheown below. Each string represents a trial solution of the problem. By analogy with biology, the values of the individual variables are known as ``alleles".

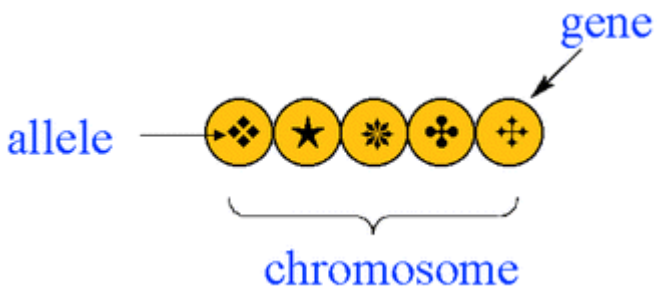


Fig.5.1 chromosome, gene and allele

The basic relation between concepts in natural evolution and genetic algorithms are given in the following table

Table 5.1 Relation between concepts in natural evolution and genetic algorithms

Concept in natural evolution	concept in genetic algorithm
Chromosome	**string**
Gene	**features in string**
Locus	**position in the string**
Allele	**position value (usually 1 and 0 in binary)**
Genotype	**string structure**
Phenotype	**set of characteristics (features)**

5.3. DIFFERENCE BETWEEN TRADITIONAL OPTIMIZATION METHODS AND GENETIC ALGORITHMS

Before answering the above questions it had better to discuss some common types of traditional optimization methods and what we mean optimization? What we are trying to accomplish when we optimize? *Optimization theory* studies how to describe and attain what is *best*, and encompasses the quantitative study of optima and method of finding them. Thus optimization seeks to improve performance toward some optimum point or points. Note that this definition has two parts; we *seek improvement* to approach some *optimal point*. This is clear distinction between the process of improvement and the destination or optimal it self. Yet, in judging optimization procedures we commonly focus solely up on convergence (i.e. does the method reach the optimum?) and forget entirely about interim performance. The most important goal of optimization is improvement (i.e. can we get some good satisfying level of performance quickly?)

There are different types of search method when we come to optimization. Most literatures identify three main types of search methods;

a) Calculus based
b) Enumerative
c) Random search

a. calculus based method

Calculus based method further subdivided into two main classes; direct and indirect methods.

i) Direct method:- direct method seeks local optima by hopping on the function and moving in direction related to local gradient. This is simply the notion of *hill climbing* to find the *local best*, climb the function in the steepest permissible direction.

ii) Indirect method::- this method seeks local extrema by solving the usual nonlinear set of equations resulting from setting the gradient of the objective function equal to zero. This is multidimensional generalization of the elementary calculus notion of extremal point. Given smooth, unconstrained function, finding possible peak starts by restricting searching to those points with slops of zero in all direction.

Drawback of calculus method

- Both methods are local in scope; the optima they seek are the best in neighborhood of the current point. Clearly, starting the search or zero finding procedure in the neighborhood of the lower peak will cause us to miss the higher peak. Further more, once the lower peak is reached, further improvement must be sought through random restart or other trickery.
- Calculus based method depend up on the existence of derivatives (well defined slope value). Even if we allow numerical approximation this is severe shortcoming and insufficiently robust.

b. enumerative method:- these schemes have been considered in many shapes and sizes. The idea is fairly straight forward with in finite search space or a discretized infinite search space, the search algorithm starts looking *objective function value at every point in space one at a time.* Although the simplicity of this type of algorithms is attractive and enumeration is very human kind of search when the number of possibilities is small, s*uch scheme must ultimately be discounted in the robustness race for simple reason lack of efficiency. It is inefficient because many practical spaces are simply too large to search one at a time.*

c. Random method:- random search algorithms have, achieved increasing popularities as researchers recognized the short comings of calculus based and enumerative schemes.

Yet, random walks and random schemes that search and save the best must also be discounted because of efficiency requirement. Random searches, in long run can be expected to do no better than enumerative schemes.

In general conventional search methods are not robust. This does not imply that they are not useful. The schemes mentioned and countless hybrid combination has been used successfully in many applications. However, as more complex problems are attacked other method will be necessary. This leads to the innovation of non conventional search method, Genetic algorithm to attack complex problems.

GAs are different from more traditional methods in the following ways

- GAs work with a coding of the parameter set, not the parameter themselves.
- GAs Seek the optimum solution by searching a population of points of the search (solution) space in *parallel* and not in isolated space (a single point) so that it searches always g*lobal optimum*.
- Uses only the values of objective function and do not require derivative information or any other information in search procedure. The direction of search is influenced by the evolutions of the objective function and of the respective fitness function only.
- GAs uses stochastic (probabilistic) transition not deterministic rules in the optimization procedure.

5.4 CHARACTERISTICS OF GENETIC ALGORITHMS

Aside from being free from dependence on functional derivatives, GAs are popular due to the following characteristics [12]:

- A population of points (trial design vectors) is used for starting the procedure instead of a single design point. Since several points are used as candidate solutions, GAs are less likely to get trapped at a local optimum.
- GAs are parallel-search procedures that can be implemented on parallel-processing machines for massively speeding up their operations.
- GAs are applicable to both continuous and discrete (combinatorial) optimization problems.
- GAs are stochastic and less likely to get trapped in local minima, which inevitably are present in any practical optimization application.

- GAs' flexibility facilitates both structure and parameter identification in complex models such as neural networks and fuzzy inference systems.
- The objective function value corresponding to a design vector plays the role of fitness in natural genetics.
- Can yield a population of optimum feasible solutions in a problem and not a unique one. The choice of the best solution is then left to the user. This is very useful in practical problems where multiple solutions exist as well as in multi objective optimization problems.

5.5 WORKING PRINCIPLE OF GENETIC ALGORITHM

Genetic algorithms are used for minimization or maximization, search for global extremes, especially when your search space is too large to check all possible solutions. Use of exact methods is mostly not possible in such cases and it is exactly the time for approximate methods like genetic algorithms.

At the very beginning we have to introduce GA dictionary of special GA terminology:

Table 5.2 Technical terms used in GA literatures

chromosome	vector which represents solutions of application task
gene	each solution which consists of a chromosome
selection	choosing parents' or offsprings' chromosomes for the next generation
individual	each solution vector which is each chromosome
population	total individuals
population size	the number of chromosome
fitness function	a function which evaluates how each solution suitable to the given task
phenotype	expression type of solution values in task world, for example, 'red,' "13 cm", "45.2 kg"
genotype	bit expression type of solution values used in GA search space, for example, "011," "01101."
parent	individual selected for reproduction
offspring	new solution, individual created from parents
crossover mutation	operations, defined on chromosomes (code), producing offsprings

Genetic algorithms begin with a population of string structures created randomly. Thereafter, the fitness measure of each string in the population is evaluated. The population is then reformed by three main operators, namely, reproduction, crossover, and mutation. Depending on the fitness measure for each solution candidate, the operation is continued until the termination criteria are met. One cycle of the application of above

mentioned operators and the evaluation procedure is known as a *generation* in the terminology of GAs. In this section, the operators in the terminology are briefly reviewed.

5.6 COMPONENTS OF GAS

The major components of GAs are namely: encoding schemes, fitness evaluation, selection, crossover, and mutation [10].

5.6.1. CODING

In order to use GAs to solve a problem, variables are first coded in some string structure. Most of the time GA encode each point in a parameter or solution space into a binary bit (binary –coded strings having 1's and 0's) string called a *chromosome*. Each point or binary string represents a potential solution to the problem that is to be solved. The length of the string is usually determined according to the desired solution accuracy i.e. the larger the length of chromosome the higher the accuracy is. Knowledge of biological terminology, though no necessary, but may help better appreciation of genetic algorithm..

In genetic algorithm , the decision variables of an optimization problem are coded by structure of one of one or more strings which are analogous to *chromosomes* in natural genetic system. The coding strings are composed of features that are analogues to *genes* (gene- eg. Animal eye color). Features are located in different position in the string where each features has it own position i.e. *locus* and definite *allele* value which compiles the proposed coding method. The string structures in the chromosome go through different operation similar to the natural evolutionary process to produce better alternative solution. The quality of new chromosome is estimated based on the '**fitness**' value which can be considered as objective function for optimization problem.

If each variable X_i, with real values, is coded as a binary string of length l , then the relation between the initial value and coding information is;

$$X_i = x_i^{(L)} + \frac{x_i^{(U)} - x_i^{(L)}}{2^{l_i} - 1} \times decodedvalue(s_i)$$

where the variable X_i can take the value from a domain $D_i=[x_i^{(L)},x_i^{(u)}]$ and is coded in substring S_i of length l. the decoded value of a binary substrings S_i is calculated as

$$\text{Si} = \sum_{i=0}^{l-1} 2^i s_i$$

, where $S_i \in (0,1)$.

Generalizing this concept, we may say that with an l_i–bit .coding for a variables , the obtainable accuracy in that variable is approximately $\frac{x_i^{(U)} - x_i^{(L)}}{2^{l_i}}$. once the coding of the variables has been done, the corresponding point $x=(x_1,x_2,.....x_N)^T$ can be found using equation (1.1). Thereafter, the function value at the point 'x' can also be calculated by substituting x in the given objective function f(x).

Encoding of chromosomes is the first question to ask when starting to solve a problem with GA. Encoding depends on the problem heavily. Here we will discuss some encodings that have been already used with some success

Binary encoding is the most common one, mainly because the first research of GA used this type of encoding and because of its relative simplicity.

In binary encoding, every chromosome is a string of **bits - 0** or **1**.

Chromosome A	10110010110010101110010 1
Chromosome B	11111110000011000001111 1

Fig.5.2.Example of chromosomes with binary encoding

Binary encoding gives many possible chromosomes even with a small number of alleles. On the other hand, this encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation.

5.6.2 POPULATION

The solution to the optimization problem requires a set of candidates, which are structurally represented as chromosomes. The content of the chromosomes is composed of zeros and ones. It is therefore evident that the length of the binary coded representation will determine the maximum number of individuals in the population as well as the accuracy of the solution.

Typically, the initial population is generated randomly.

5.6.3. OBJECTIVE FUNCTION/ FITNESS FUNCTION

The fitness/objective function is chosen depending on the problem in hand such that the individuals having high fitness values are the good solution candidates for the

optimization. Therefore, reproduction of the next generation will strictly be dependent on the fitness measure.

In general fitness function is $F(x)$ is first derived from objective function and used in successive genetic generations. Certain genetic operator requires that the fitness function be non negative, although certain operators do not have this requirement. For maximization problems the fitness function can be considered to be the same as objective function or $F(x)$=f(x). For minimization problems, the fitness function is an equivalent maximization problem chosen in such a way that the optimum point remains unchanged. A number of such transformations are possible. The following fitness function is often used;

$$F(x) = \frac{1}{1+f(x)} \quad \text{or} \quad F(x) = -f(x)$$ where f(x) is the objective function

This transformation does not alter the location of optimum point but converts maximization problem in to equivalent minimization problem. The fitness function value is called string fitness for survival.

Fitness is an important concept for the operation of the GA. The fitness of a string is a measure of the quality of the trial solution represented by the string with respect to the function being optimized. Thus, high fitness corresponds to a high value (in a maximization problem) or a low value (in a minimization problem) of the function. If the upper and lower limits of the function being optimized are known, then *absolute fitness* may be used -- where fitness values may be compared from generation to generation. Otherwise (as in most GA applications), *dynamic fitness scaling* can be adopted, where, in each generation the fitness of all the individuals are scaled relative to the best and worst members of the current population. Fitness is important in determining the likelihood of an individual taking part in crossover and also in deciding which individuals will survive into the next generation.

5.6 .4 GA OPERATORS

The operation of GAs begins with a population of random strings representing design variables. Thereafter, each string is evaluated to find the fitness value. The population is then operated by three main operators- *reproduction, crossover,* and *mutation*- to create a new population of points. The new population further evaluated and tested for termination. If termination criteria not met, the population is iteratively operated by above

three operators and evaluated. This procedure is continued until the termination criterion is met.

An evolutionary algorithm is start with population of randomly generated individual, although it is possible to use previously saved population or population of individual encoding to solution provided by human experts or by another heuristic algorithms. In case of genetic algorithm initial population will be made up of random bit strings. Once an initial population has been created, evolutionary algorithms enter a loop. At the end of each iteration a new population will have been created by applying certain stochastic operator to the previous population. One such iteration is referred to as a *generation*. The iteration is shown in the following flow chart;

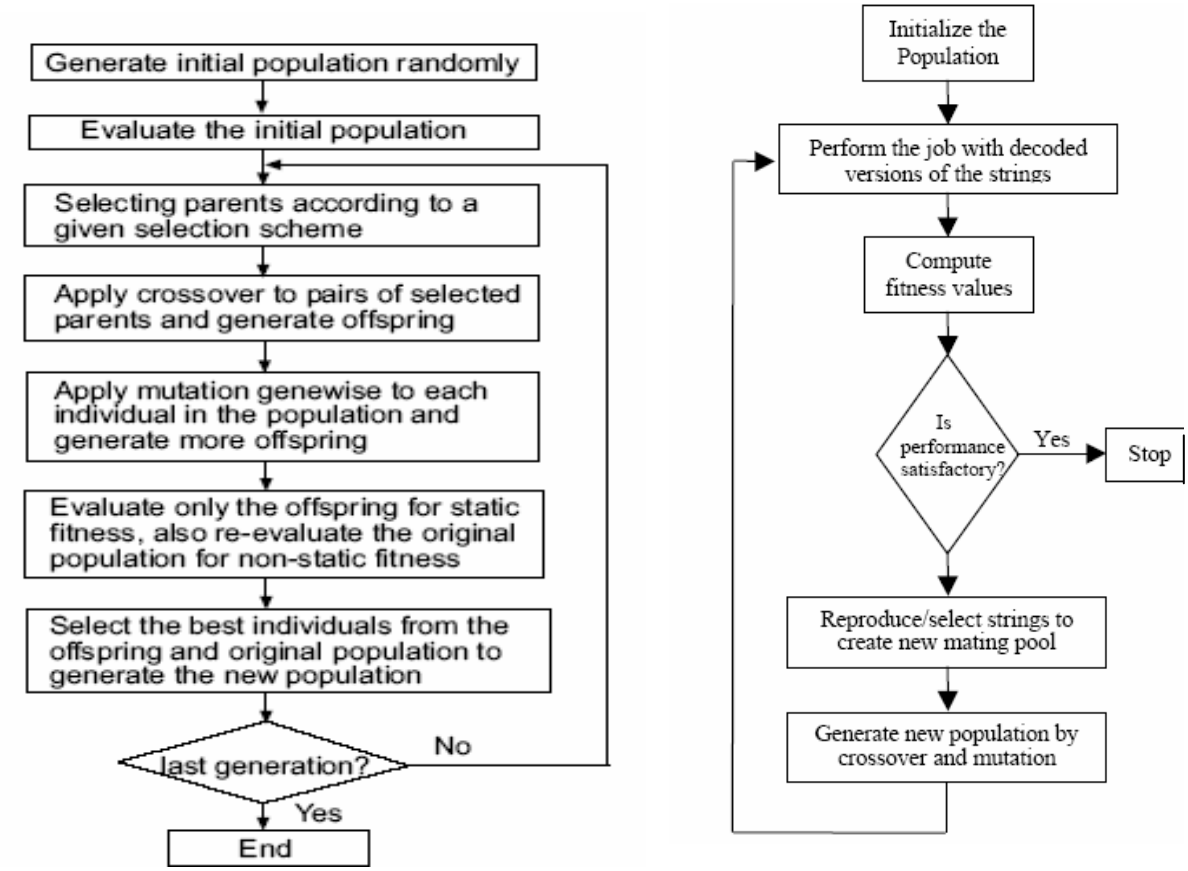


Fig5.3 Basic steps of a genetic algorithm

In principle, the above described loop is infinite, but it can be stopped when a given termination condition specified by the user is met. Examples of termination conditions are

- A pre-determined number of generation or time has elapsed;
- A satisfactory solution has been found;
- No improvement in solution quality has take place for predetermined number of generation

Evolutionary cycle can be summarized by the following pseudo code;

Generation =0;

Seed population

While not termination condition **do**

Generation=Generation+1

Calculate fitness

Selection

Crossover (P_{cross})

Mutation (P_{mut})

End while

Reproduction/ selection operator; Reproduction is a method for increasing the number of solution candidates having high fitness values. Reproduction is the first operator applied on a population. Reproduction selects good strings in a population and form a mating pool, which, at the same time, eliminates the least-fit strings from the pool.. That is why the reproduction operator sometimes called the selection operator.

According to Darwin the most qualified (fittest) creature survive to mate. Fitness is determined by creature's ability to survive predators, pestilence, and other obstacles to adulthood and subsequent reproduction. In our artificial setting, we quantify "most qualified" via a chromosome's fitness $F(x)$. The fitness function is the final arbiter of the strings-creature's life or death. Selecting strings according to their fitness value means that the string with a higher probability of contributing one or more off spring in the next generation. The commonly used reproduction operator is the proportionate reproduction operator where string is selected for the mating pool with the probability proportional to its fitness. Thus i^{th} string is in the population is selected with the probability proportional to F_i . Since the population size is usually kept fixed in a simple GA, the sum the probability of each string being selected for the mating pool must be one. Therefore, the probability of selecting the i-th string is

$$p_i = \frac{F_i}{\sum_{j=0}^{n} F_i}$$

where n is the population size. One way to implement this selection scheme Goldberg [12] uses the analogy of spinning a unit circumference *roulette wheel;* the wheel is cut like a pie in to S regions where the i^{th} region is associated with the element of p_i. one spine the wheel , and if the pointer points at region i when the wheel stops, place the corresponding string into the mating pool. Since the circumference of the wheel is marked according to a string's fitness, this roulette wheel mechanism is expected to make $\frac{F_i}{\sum_{i=1}^{n} F_i / n}$ copies of the i^{th} string in the mating pool.

Simulation procedure of roulette wheel selection scheme is as follows;

1) Using the fitness value $\mathcal{F}_i$ of all string the probability of selection of all string p_i can be calculated i.e $p_i = \frac{F_i}{\sum_{j=0}^{n} F_i}$
2) Calculate the cumulative probability of P_i of each string being copied by adding the individual probabilities from the top of the list.
3) Simulate roulette wheel concept by realizing the i-th string in the population represent the cumulative probability value from P_{i-1} to P_i. The cumulative probability of any string lies between 0 to 1. In order choose n strings , n random number between 0 and 1 are created at random. Finally copy a string that represent the chosen random number in the cumulative range , calculated from the fitness value, to the mating pool.

This way the string with a higher fitness value will represent a larger range in the cumulative probability value and there for has a higher probability of being copied into the mating pool. On the other hand , a string with a smaller fitness value represents a smaller range in cumulative probability value and has a smaller probabilities being copied in to the mating pool. It is important to note that no new strings are formed in the reproduction phase.

As we already know from the GA outline chromosomes are selected from the population to be parents for crossover. The problem is how to select these chromosomes. According

to Darwin's theory of evolution the best ones survive to create new offspring. There are many methods in selecting the best chromosomes. Examples are roulette wheel selection, tournament selection, rank selection, steady state selection and some others. In this project we will use roulette wheel selection and described as follows.

In **Roulette Wheel** Selection Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where all the chromosomes in the population are placed. The size of the section in the roulette wheel is proportional to the value of the fitness function of every chromosome - the bigger the value is, the larger the section is. See the following picture for an example.

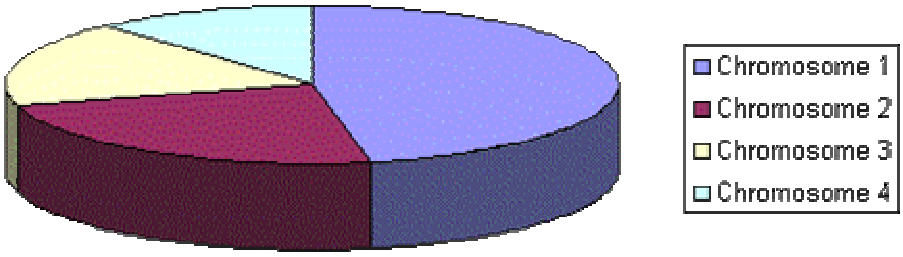


Fig 5.4 Roulette wheel selection

A marble is thrown in the roulette wheel and the chromosome where it stops is selected. Clearly, the chromosomes with bigger fitness value will be selected more times.

Crossover; As we can see from the genetic algorithm outline, the crossover and mutation are the most important parts of the genetic algorithm. The performance is influenced mainly by these two operators.

Crossover is applied to selected pairs of parents with a probability equal to a given *crossover rate*. It is hoped then that the new generated chromosomes retain the good features of the previous generation. Actually there are different types of crossover such as one point crossover, two point crossover, uniform crossover, simplex crossover etc. The two common crossover operators are the one-point and the two-point; these are illustrated in Figure 3.7.

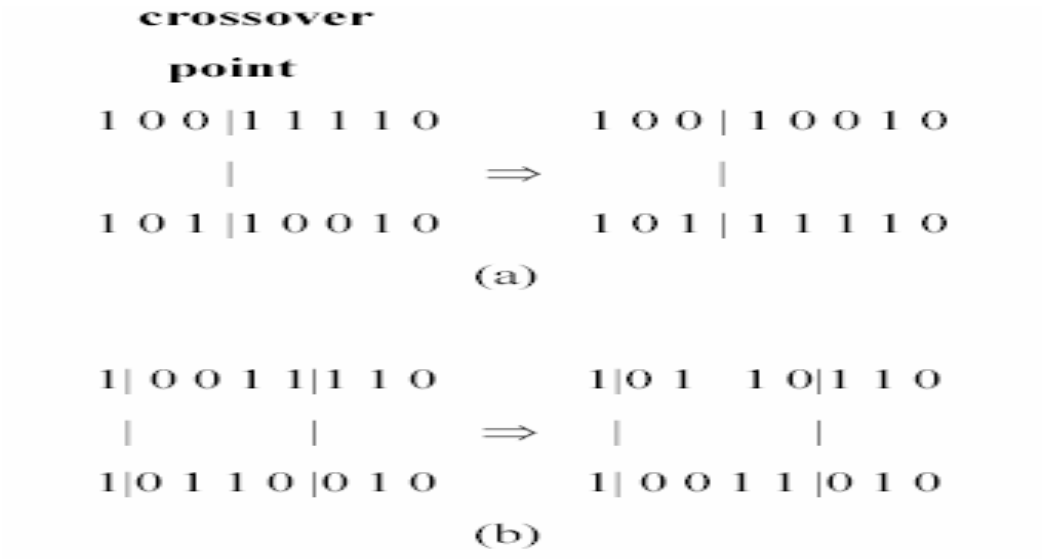


Fig 5.5 Crossover operators: (a) one-point crossover; (b)two-point crossover.

Using *selection* and *crossover* on their own will generate a large amount of different strings. However there are two main problems with this:

1. Depending on the initial population chosen, there may not be enough diversity in the initial strings to ensure the GA searches the entire problem space.
2. The GA may converge on sub-optimum strings due to a bad choice of initial population.

These problems may be overcome by the introduction of a mutation operator into the GA. Mutation is the occasional random alteration of a value of a string position. It is considered a background operator in the genetic algorithm

Mutation; Mutation operator is another degree of freedom in search procedure, which is frequently used in GA based designs. Functionally, the operator negates the value of a bit in the string. This makes it possible to reach to the inaccessible regions of search space. The need for mutation is to keep the diversity in the population and to escape from local minima. For example, if beyond a particular position along the all strings in the population have a value 0, and if a 1 is needed beyond that position to obtain the optimum, then neither reproduction nor crossover operator described above will be able to create a 1 in that position. The activation of the mutation operator is generally controlled by the excess of a certain probability threshold. Therefore the design must include such a criterion.

After a crossover is performed, mutation takes place. Mutation is intended to prevent falling of all solutions in the population into a local optimum of the solved problem. Mutation operation randomly changes the offspring resulted from crossover. In case of binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can be then illustrated as follows:

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110110

Fig.5.6 mutation

The technique of mutation (as well as crossover) depends mainly on the encoding of chromosomes. For example when we are encoding permutations, mutation could be performed as an exchange of two genes. In **static mutation**, the mutated gene is assigned a completely random value, while in **dynamic mutation** its value is changed by a small, random amount about its original value.

5.7 ELITISM

With crossover and mutation taking place, there is a high risk that the optimum solution could be lost as there is no guarantee that these operators will preserve the fittest string. To counteract this, elitist models are often used. Elitism is the name of the method that first copies the best chromosome (or few best chromosomes) to the new population. In an elitist model, the best individual from a population is saved before any of these 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 The rest of the population is constructed in ways described above. Elitism can rapidly increase the performance of GA, because it prevents a loss of the best found solution

Eg . if population size=20;

Elite count=2

Pc=0.8 then the number of each type of children in the next generation is as follows;

- There are 2 elite children
- There are 18 individual other than elite children so the algorithm rounds 0.8x 18=14 to get the number of cross over children
- The remaining 4 individual are mutation children

5.8 PARAMETERS OF GENETIC ALGORITHM

There are two basic parameters of GA - crossover probability and mutation probability.

Crossover probability: how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is **100%**, then all offspring are made by crossover. If it is **0%**, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!).

Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old populations survive to next generation.

Mutation probability: The probability of mutation is normally low because a high mutation rate would destroy fit strings and degenerate the genetic algorithm into a random search.

Mutation probability values of around 0.1% or 0.01% are common, these values represent the probability that a certain string will be selected for mutation i.e. for a probability of 0.1%; one string in one thousand will be selected for mutation.

If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is **100%**, whole chromosome is changed, if it is **0%**, nothing is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random

search**.** If the mutation rate is too high, the GA searching becomes a random search, and it becomes difficult to quickly converge to the global optimum. There are also some other parameters of GA. One another particularly important parameter is population size.

Population size: how many chromosomes are in population (in one generation). If there are too few chromosomes, GA have few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to use very large populations because it does not solve the problem faster than moderate sized populations.

5.9 ESTABLISHING THE NEXT GA-GENERATION

Any subsequent population may be overlapping or non-overlapping with respect to the current population P(t). An overlapping population is the one that includes members of the parental population P(t) and some or all of the offspring of this parental generation. Overlapping populations are based upon relative fitness among parents and offspring and are sometimes referred to as “elitist”, indicating that strongly performing individual’s progress through a number of generations.

Non-overlapping populations are those populations where P(t) is completely replaced by offspring to form the next generation of P(t+1).

In relation to optimization problems where convergence is the prime consideration, most literatures suggest that a non-overlapping population scheme performs best. Such a scheme is perhaps preferable since it obviates the need for fitness evaluations and comparisons between existing population members and offspring and therefore facilitates a simpler implementation.

5.10 TERMINATION CRITERIA OF GA

Due to the fact that GAs are stochastic search methods, it can prove difficult to prescribe a formal convergence criteria. In practice, the most common method is to allow the GA to run for a prescribed number of generations, followed by evaluation of the best population members against the target problem. If an acceptable solution has been found or there is no improvement in fitness value then the algorithm may terminate, if not, then further runs of the algorithm may be initiated.

5.11 STRENGTH OF GENETIC ALGORITHMS

One of GA's most important qualities is its ability to evaluate many possible solutions simultaneously. This ability, called implicit parallelism, is the cornerstone of GA's power. Each string may contain millions of building blocks that comprise the string, and GA assesses them all simultaneously each time it calculates the string's fitness. Besides, GAs have the quality of robustness, that is, while special case algorithms may find more optimal solutions to specific problems. GAs performs very well over a large number of problem categories. Because of this, GAs are not caught by local minima. GAS also perform well on problems whose complexity increases exponentially with the number of input parameters, since these type of problems are extremely inefficient to solve using traditional approaches.

Furthermore, GAS can produce intermediate solutions, i.e. the search can stop at any time if a suboptimal solution is acceptable. Finally, GAs easily lend themselves to parallel processing; they can be implemented on any multiprocessor architecture.

Advantage of genetic optimization

- GAs are parallel search procedure that can implemented parallel processing machine for massively speeding up their operation
- GAs is applicable to both continuous and discrete optimization problems
- GAs are stochastic and less likely to get trapped I local optima, which inevitably are present in any practical optimization applications
- GA's flexibility facilitates both structure and parameter identification in complex models.
- superior global searching capability in the space which has complex searching surface,
- Applicability to the searching space where we cannot use gradient information of the space.

5.12 WEAKNESS OF GENETIC ALGORITHMS

Genetic algorithms are adaptive methods, which may be used to solve search and optimization problems. Although GAS are easy to implement and are powerful tools to solve difficult problems featuring complex search spaces, they usually require human supervision to be exploited successfully. An important problem in the use of GAS is the premature convergence to a local optimum. This mainly arises when there is lack of diversity in the population and disproportionate relationship between exploitation and exploration.

Two general approaches towards improving genetic algorithm performance have been based on

(a) The development of adaptive mechanisms within the GA, and,

(b) The optimization of static parameters such as mutation rate or population size.

Limitation of genetic optimization

The genetic algorithm theory provides some explanation why for given problem formulation , we may obtain convergence to the sought optimal point . Unfortunately, practical applications do not always follow the theory, the main reason being;

- The coding of the problem often moves the GA to operate in a different space the problem itself
- There are practical limit on the hypothetically unlimited number of iteration or generation in genetic algorithm
- There is a limit on the hypothetically unlimited population size.
- The key disadvantage of the GAs is that its convergence speed near the global optimum becomes slow.

One of the implication of these observations is the inability of GA , under certain condition, to find optimal solution or even an approximation to the optimal solution ; such failures are usually caused by *premature convergence* to local optimum.

CHAPTER-VI
MATLAB IMPLEMENTATION OF SIMPLE GENETIC ALGORITHM OPTIMIZATION

INTRODUCTION

MATLAB is acronym of Matrix laboratory, which has been designed to speed up mathematical calculations. It has several design tool box for different subject and in the latest release it incorporates built in genetic tool box and fuzzy logic tool box. In this project MATLAB version 6.5 with external genetic tool box released by MATWORKS, company has been used.

6.2 GENETIC TOOL BOX FOR MATLAB

Genetic algorithm for optimization of simple multi-modal function was done by using matlab genetic tool box created by the MathWorks, Inc. 1993. This tool box has several m-files that can be called when required to accomplish the required task. This m-files includes the following functions;

1) **ENCODING FUNCTION**; a function which Converts from decimal value to binary representation
   the syntax is ; function [gen,lchrom,coarse,nround] = encode(x,vlb,vub,bits)
   where

[GEN,LCHROM,COARSE,nround] = ENCODE(X,VLB,VUB,BITS) Encodes non-binary variables of X to binary. The variables in the i'th column of X will be encoded by BITS(i) bits. VLB and VUB are the lower and upper bounds on X. GEN is the binary representation of these X. LCHROM=SUM(BITS) is the length of the binary chromosome. COARSE(i) is the coarseness of the i'th variable as determined by the variable ranges and BITS(i). ROUND contains the absolute indices of the X which where rounded due to finite BIT length.

2) **DECODING FUNCTION;** ------ function [x,coarse] = decode(gen,vlb,vub,bits)
   DECODE Converts from binary to variable representation.
   [X,COARSE] = DECODE(GEN,VLB,VUB,BITS) converts the binary population GEN to variable representation. Each individual of GEN should have SUM(BITS). Each individual binary string encodes length(vlb)=length(vub)=length(bits) variables.

COARSE is the coarseness of the binary mapping and is also of length-length(vub).

3) **MATE FUNCTION;** -----function [new_gen,mating] = mate(old_gen)

MATE Randomly reorders (mates) OLD_GEN.

[new_gen,mating] = mate(old_gen) performs random reordering on old_gen. new_gen is the new reordering. Individual in row 1 is to be mated with individual in row 2, etc. MATING is the reordering vector (ie: new_gen=old_gen(mating,:)).

4) **MUTATE FUNCTION**;----- function [new_gen,mutated] =mutate(old_gen,Pm)

MUTATE Changes a gene of the old_gen with probability Pm.

[new_gen,mutated] = mutate(old_gen,pm) performs random mutation on the population old_pop. Each gene of each individual of the population can mutate independently with probability Pm. Genes are assumed possess Boolean alleles. MUTATED contains the indices of the mutated genes.

5) **CROSSOVER FUNCTION**-- function [new_gen,sites] = xover(old_gen,Pc)

XOVER Creates a new_gen from old_gen using crossover.

[new_gen,sites] = xover(old_gen,pc) performs crossover procreation on pairs of old_gen with probability Pc .Crossover SITES are chosen at random .

6) **REPRODUCE FUNCTION**--function [new_gen,selected] = reproduc(old_gen,fitness)

REPRODUC selects individuals proportional to their fitness.

[new_gen,selected] = mate(old_gen,fitness) selects individuals from old_gen proportional to their fitness new_gen will have the same number of individuals as old_gen. selected contains the indices (rows) of the selected individuals (ie: NEW_GEN=OLD_GEN(SELECTED,:)).

7) **GENETIC FUNCTION**…function [xopt,stats,options,bestf,fgen,lgen] = genetic(fun, x0,options,vlb,vub,bits,P1,P2,P3,P4,P5,P6,P7P,P8,P9,P10)

GENETIC tries to maximize a function using a simple genetic algorithm.

X=GENETIC('FUN',X0,OPTIONS,VLB,VUB) uses a simple (haploid) genetic algorithm to find a maximum of the fitness function FUN (usually an M-file: FUN.M). The user may define all or part of an initial population X0 (or supply an empty argument in which case an initial population will be chosen randomly between the lower and

upper bounds VLB and VUB. Use OPTIONS to specify optional parameters such as population size and maximum number of generations produced. The default algorithm uses a fixed population size, OPTIONS(11), and no generational overlap. Three genetic operations: reproduction, crossover, and mutation are performed during procreation.
The probability that an individual of the population will reproduce is proportional to its fitness. Individuals chosen for reproduction are mated at random. Mating produces two offspring (re: constant population size.) Crossover in mating occurs with probability Pc=OPTIONS(12) and the crossover index is randomly selected. Each feature of the offspring can mutate independently with probability Pm=OPTIONS(13). Default options are OPTIONS(11:13)=[30 1 0].
The default maximum generations OPTIONS(14) is 100.
X=GENETIC('FUN',X0,OPTIONS,VLB,VUB,BITS) allows the user to define the number of BITS used to code non-binary parameters as binary strings. Note: length(BITS) must equal length(VLB).

X=GENETIC('FUN',X0,OPTIONS,VLB,VUB,BITS,P1,P2,...) allows up to ten arguments, P1, P2, ... to be passed directly to FUN.

F=FUN(X,P1,P2,...).

[X,STATS,OPTIONS,BESTF,FGEN,LGEN]=GENETIC(<ARGS>)

STATS - [max min mean std] for each generation

OPTIONS - options used

BESTF - Fitness of individual X (i.e.: best fitness)

FGEN - first generation population

LGEN - last generation population

The following two functions were added in genetic tool box to initialize the population and to calculate the fitness of our proposed function. The two functions are

8) **INIT FUNCTION**…. function phen=init(vlb,vub,siz,sea)
This function creates a random initial population

9) **SCORE FUNCTION** … function [fitness, object]=score(phen, popsize)
This function computes fitness and objective function values of population

6.3 INVESTIGATING GA PARAMETERS

There are many factors that affect how the GA performs. These include population size, mutation probability and crossover probability among others. To investigate how some of these factors affected the GA an experiment was performed using a test function with multiple maxima and minima. The main program is found in **AppendexB1**.

The test function is $f(x_1,x_2)=x_1^2+x_2^2-0.3*\cos(3*pi*x_1)-0.4\cos(4*pi*x_2)+0.7$ This function is plotted as Figure below. Genetic algorithm searches the global minimum of the function. The program searches and traces the global minimum by blue line on countor plot.

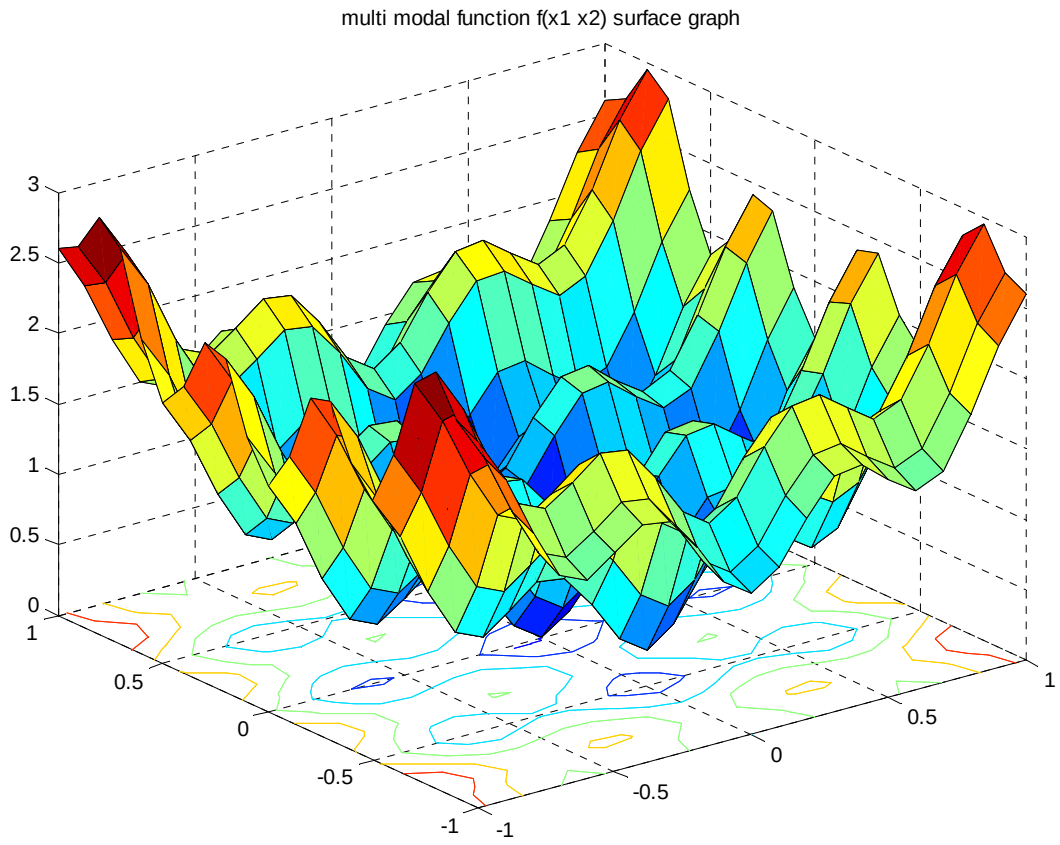


Fig.6.1 multi-modal function

As can be seen from the plot, this function has many optimum whose peak and valley values are different. It is hoped that the GA searching this solution space will find the peak value in this range which occurs at $x_1 = 0$, $x_2 = 0$ or somewhere close to it. There

are also two local minima whose peaks fall just short of the global peak. Those searches that succeed in avoiding getting stuck at these local minimum are those whose properties should be noted and emulated. To perform the GA, MATLAB was used in conjunction with the genetic toolbox, which is open-source code provided by Andrew et al. [7]. An objective function has to be provided for this toolbox that evaluates the string passed in. This code is shown below.

```
function [fitness, object]=score(phen,popsize)
for ii=1:popsize
objective(ii)=phen(ii,1)^2+phen(ii,2)^2-0.3*cos(3*pi*phen(ii,1))-
0.4*cos(4*pi*phen(ii,2)) +0.7;
   fitness(ii)=1/(objective(ii)+1);
end
```

The graph of the best average and worst value verses generation number is shown below.
Population size=10 pm=0.01 pc=0.8 bits=12 each generation =50

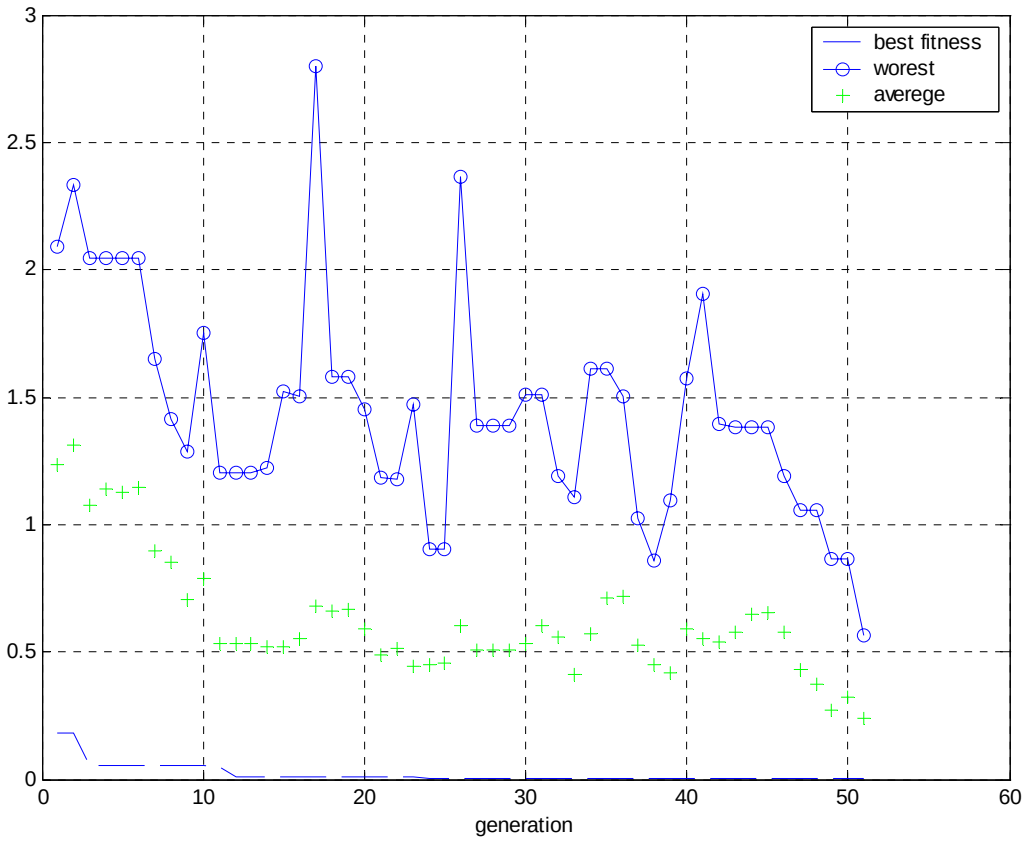


Fig.6.2 graph of the best average and worst value of multi-modal function f(x1,x2)

To apply the parameters to the GA a script file was written in which the essential parameters are set. This file is listed in Appendix B1, different runs of the GA were performed, and the parameters for these runs are given in Table below.

6.4 EFFECT OF GENETIC PARAMETERS

Table 6.1 comparison of effect of genetic parameter optimization problem

s.no	pc	pm	generation	Population size	Bit length	Best fitness value	Average fitness value	Worst fitness value
1	0.8	0.01	50	10	12	0.000932	0.777718	0.865227
2	0.4	0.9	50	10	12	0.003424	1.149183	2.007528
3	0	1	50	10	12	0.623899	1.075680	1.843257
4	1	0	50	10	12	0.008560	0.059543	0.145596
5	0.8	0.01	100	10	12	0.000039	0.328916	1.301278
6	0.4	0.9	100	10	12	0.002000	1.177807	1.832853
7	0.8	0.01	50	40	12	0.001423	1.061131	1.926886

The above table indicates that how the selection of parameter affects the out come of genetic algorithm. The following factors affect the out come of genetic algorithms;

a) Population diversity; diversity indicates that the average distance between individual. If diversity is high, average distance between individual is large and vice versa. Getting the right diversity is a matter of trial and error. If the diversity is too high or too low the GA might not perform well. Diversity is controlled by appropriately setting *initial range* of the population

b) Population size; population size determines the size of population at each generation. If population size is high GA have to search more points which leads to better result but the longer the Ga takes to compute each generation value . At least the population size should be equal to number of variable

c) fitness scaling.; converts raw fitness score that are returned by the fitness function to value in the range that is suitable for the selection function

which uses scaled fitness to select the parameter of the next generation according to its fitness value. If the scaled value too widely the individual with the highest scaled value reproduce to rapidly taking over the population generation converges to quickly ,preventing the GA from searching other area of the function space. If it vary only a little all individual have approximately the same chance of reproduction and search will progress slowly. In most GA default fitness scaling is RANK , scales the raw score based on the rank of each individual instead of its score. The rank of individual is its position in the sorted score , most fit individual rank 1……n .last fit individual

d) selection operator; choose parents for the next generation based on their scaled value from fitness scaling function
e) Reproduction operator; controls how genetic algorithm creates the next generation. Usually GA uses elite count i.e. best fitness value in the current generation directly pass to the next generation .which is called elite children. Setting elicits count high value causes the fittest individual to dominate to dominate the population, which make the search less effective.
f) Crossover probability (pc). Determines the fractions of individual in the next generation other than elite children hat are created by crossover operator.
g) Mutation probability (pm) ; Determines the fraction of individual in the next generation other than elite and crossover children by introducing random change of bit from 1 to 0 or vice versa in single parent.

To show the actual procedure or basic flow chart of genetic algorithm m-file has been created and presented in appendix A3. The program follows the basic procedure of GA and intended to give deep insight in to how GA works. The function to be optimized has been taken from reference [50] and as expected genetic algorithm searches the optimum value of the function. The graph of the best and average fitness value verses generation number is shown below for different optimization parameters.

The optimized function is

$$f(x_1,x_2)=(x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2$$

Population size=20
Pc=0.8
Pm=0.05
maxgen=100
Bits= 10 each
Number of variable=2
Range=[0 6; 0 6]

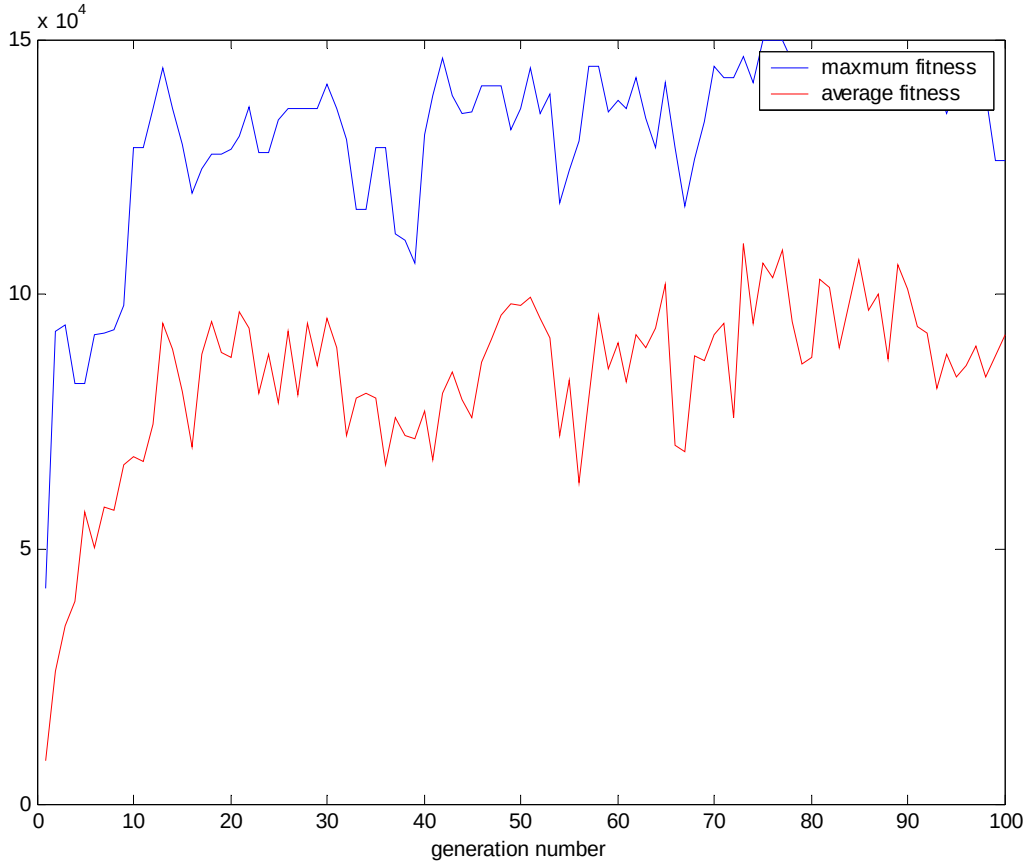


Fig.6.3 graph of Best and Average fitness of $f(x_1,x_2)$

Population size=100

Pc=0.8

pm=0.05

maxgen=100

Bits= 10 each

Number of variable=2

Range=[0 6; 0 6]

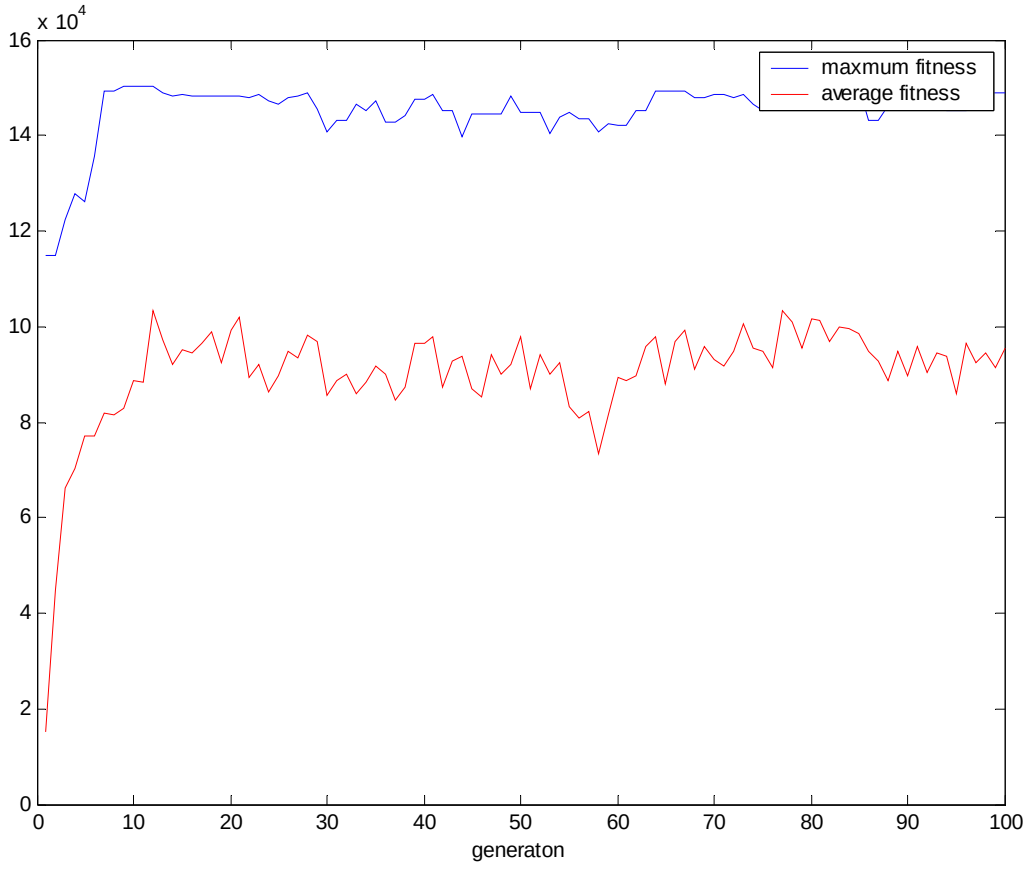


Fig.6.4 graph of Best and Average fitness of $f(x_1,x_2)$ for different parameter

CHAPTER-VII
FUZZY-GENETIC CONTROLLER

INTRODUCTION

Different approaches have been proposed to automate the design of fuzzy systems. Many of these approaches take the genetic algorithm as a base of the learning process. A GA was used to optimize the fuzzy logic input membership functions, the fuzzy rules, the output membership functions and universe of discourse. The use of an Evolutionary Algorithm to solve the above optimization problem is suggested for the investigation of randomly chosen sets of controller parameter values. Fuzzy systems (FS) which use GAs for automatic tuning/learning the FS components are known as Genetic Fuzzy Systems (GFS). Further, evolutionary algorithms provide a universal optimization technique that mimics the type of genetic adaptation that occurs in natural evolution [23], [31]. Unlike specialized methods designed for particular types of optimization tasks, they require no particular knowledge. Recently numerous publications propose evolutionary algorithms to automate the knowledge acquisition step in fuzzy system design [30], [26], [19]. These methods are described by the general term *genetic-fuzzy systems*. Genetic fuzzy systems are applicable to control design problems in which the objective is to maximize some performance index of the closed loop process itself, as the evolutionary optimization is solely based on a scalar objective function.

7.2. OBJECTIVE OF GENETIC FUZZY SYSTEM

The objective of a genetic fuzzy system is to automate the knowledge acquisition step in fuzzy system design, a task that is usually accomplished through an interview or observation of a human expert controlling the system. An evolutionary algorithm adapts either part or all of the components of the fuzzy knowledge base. At this point, it is important to notice that a fuzzy **knowledge base** is not a monolithic structure but is *composed* of the **data base** and the **rule base** which each play a specific role in the fuzzy reasoning process.

7.3 TYPES OF GENETIC FUZZY SYSTEM

According to the distinction between data base and rule base, genetic fuzzy systems are discriminated along two major approaches,

- genetic tuning processes and
- genetic learning processes.

The first method is targeted at optimizing the performance of an already existing fuzzy system. The tuning process involves the adaptation of the fuzzy database, *namely parameters of membership functions and input-output scaling factors.*

The second method is concerned with the *automatic derivation of fuzzy rules in the rule base*. A *genetic learning* process faces a much more difficult task as it has to establish the proper relationship between input and output states from scratch, rather than optimizing the performance of a fuzzy system that already operates at least approximately correct.

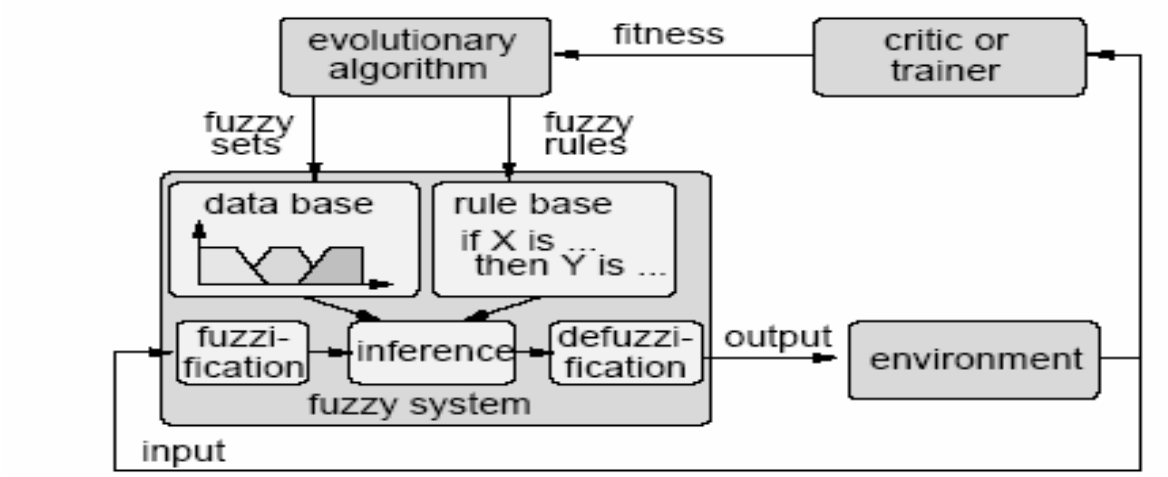


Fig. 7-1. Genetic fuzzy system

Figure 7-1 shows the major components of a genetic fuzzy system. The fuzzy system lies at the core of the hybrid structure; it fuzzifies the input state, performs the inference based on the fuzzy rules and aggregates the result of the inference process into a crisp output. Depending on the context, the environment can be a plant to be controlled, a system to be modeled or a set of data to be classified. An external critic or trainer evaluates the performance of the fuzzy system with regard to the control task, the model accuracy or the classification error. The performance is aggregated into a scalar fitness value on which basis the evolutionary algorithm selects better adapted chromosomes.

A **chromosome** either codes

- parameters of membership functions,
- scaling factors,
- Fuzzy rules or a combination thereof.

By means of crossover and mutation, the evolutionary algorithm generates new parameters for the database and/or rule base which usefulness is tested in the fuzzy system.

The optimal configuration of fuzzy sets and/or rules and in that sense can be regarded as an optimization problem. The optimization criterion is the problem to be solved at hand and the search space is the set of parameters that code the membership functions, scaling functions and fuzzy rules. The genetic learning process emerges from the hybridization of an evolutionary algorithm, which by *means of selection and genetic operators optimizes parameters of the knowledge base*, with the fuzzy system supposed to demonstrate a desired behavior.

The majority of publications are concerned with fuzzy rule based control system design. For two reasons, a fuzzy representation is particularly useful for evolutionary optimization compared to other possible parameterizations of a controller. For many real-world problems a mathematical precise and complete solution is not only unnecessary, but often also unfeasible. Fuzzy systems exploit this tolerance for imprecision by aggregation of similar states into coarse granules defined by fuzzy sets. The remaining task for the evolutionary algorithm becomes to learn the appropriate local relationships between input and output states established by fuzzy if-then rules. The locality and granularity of fuzzy rules, not only decreases the complexity of the search space but at the same time lessens the interdependency of the very genes that encode these rules. In return, limited interaction among genes expedites the evolutionary optimization process according to the building block hypothesis.

7.4 MEMBERSHIP PARAMETERS OPTIMIZATION

GAs are applied to modify the membership functions. When modifying the membership functions, these functions are parameterized with one to four coefficients (fig. below), and each of these coefficients will constitute *a gene of the chromosome for the GA.*

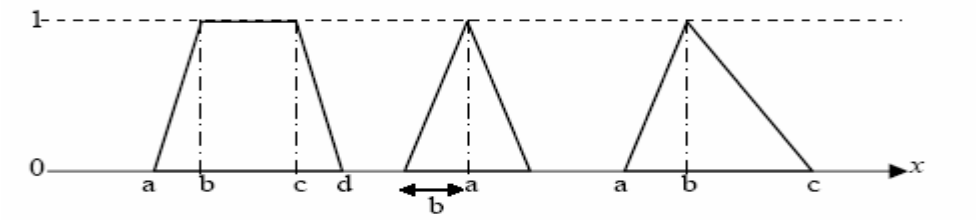


Fig.7.2 Some parameterized membership functions

Fuzzy membership functions provide the characterization of fuzzy sets by establishing a connection between linguistic terms (such as \slow", \medium", \fast" for a speed variable) and precise numerical values of variables in a physical system. A fuzzy membership function approximates the confidence with which a numerical value is described by a linguistic term [44].

A typical example of triangular fuzzy membership functions for a speed variable is given in Figure7.3.

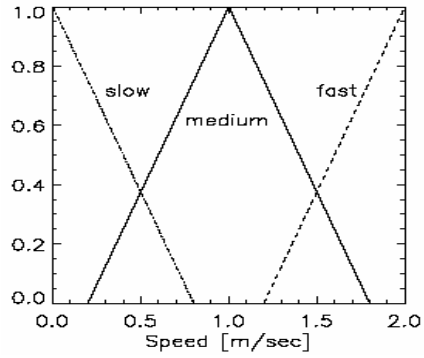


Fig 7-3: Membership functions for a physical variable \speed.

Notice that membership functions not necessarily have to be (isosceles) triangles as in this example. For instance, Gaussian membership functions

$$u(x) = \exp\left(\frac{-(x-c)^2}{2\sigma^2}\right)$$

provide an important alternative . A further possibility for the shape of membership functions is to choose trapezoidal [4].

The correct choice of the membership functions, however, is by no means trivial but plays a crucial role in the success of an application. Several example applications demonstrate

that evolutionary algorithms are capable of optimizing the membership functions of fuzzy logic controllers. The basic idea is to represent the complete set of membership functions by an individual and to evolve shape and location of the triangles (respectively the Gaussian curves).
Each triangle may be described by its anchor points on the abscissa axis, and the Gaussian membership functions are characterized by c and σ.

We conclude this section by referring to a problem which may arise from unconstrained variations of the membership function shape by the optimization algorithm: The completeness property, which requires that a fuzzy logic controller always be able to infer a control action for every state of the process might be violated if the anchor points of membership functions are shifted such that the possible range of values is no longer completely covered.
In order to solve this problem, one might consider to introduce special constraints to the evolutionary algorithms' objective function.

7.5 FUZZY RULE BASE OPTIMIZATION

Besides learning the membership functions, an even more challenging problem consists in the automatic learning of fuzzy control rules, i.e., the linguistic statements which are normally derived from expert knowledge. This idea comes close to so-called classifier systems, rule-based systems that use a genetic algorithm as a rule-generation mechanism, such that the classifier system is capable of inductive learning.
Different methods are defined to apply GA to the rule base optimization, depending on its representation. For example, GA are used to modify the decision table of an FLC, which is applied to control a system with two input (trial-and error) and one input (command action) variables. A chromosome is formed from the decision table by going row-wise and coding each output fuzzy set as an integer in 0, 1… *n*, where *n* is the number of membership functions defined for the output variable of the FLC. Value 0 indicates that there is no output, and value *k* indicates that the output fuzzy set has the *k*-th membership.

The application of the GA in the optimization of the FLC controllers can be reformulated as follows:

1. Start with an initial population of solutions that constitutes the first generation (P(0)).
2. Evaluate P(0):

a) Take each chromosome (KB) from the population and introduce it into the FLC,

b) Apply the FLC to the controlled system for an adequate evaluation period,

c) Evaluate the behavior of the controlled system by producing a performance index to the KB.

3. While the termination condition is not met, do

a) Create a new generation (P(t+1)) by applying the evolution operators(selection, crossover and mutation) to the individuals in P(t),

b) Evaluate P(t+1)

c) t = t+1.

4. End.

7.6 LEARNING WITH GA

Although GAs are not learning algorithms , they may offer a powerful and domain independent search method for a variety of learning tasks[41]. Three alternative approaches , in which GAs have been applied to learning process, have been proposed , the Michigan, the Pittsburg, and Iterative rule learning (IRL) approaches. In the first one, the chromosome corresponds to classifier rule, which are evolved as a whole, whereas in Pittsburgh approaches, each chromosome encodes a complete set of classifiers. In the IRL approach each chromosome represents only one rule learning , but contrary to the first ,only the best individual is considered as the solution , discarding the remaining chromosomes in the population. Below, we will describe them briefly.

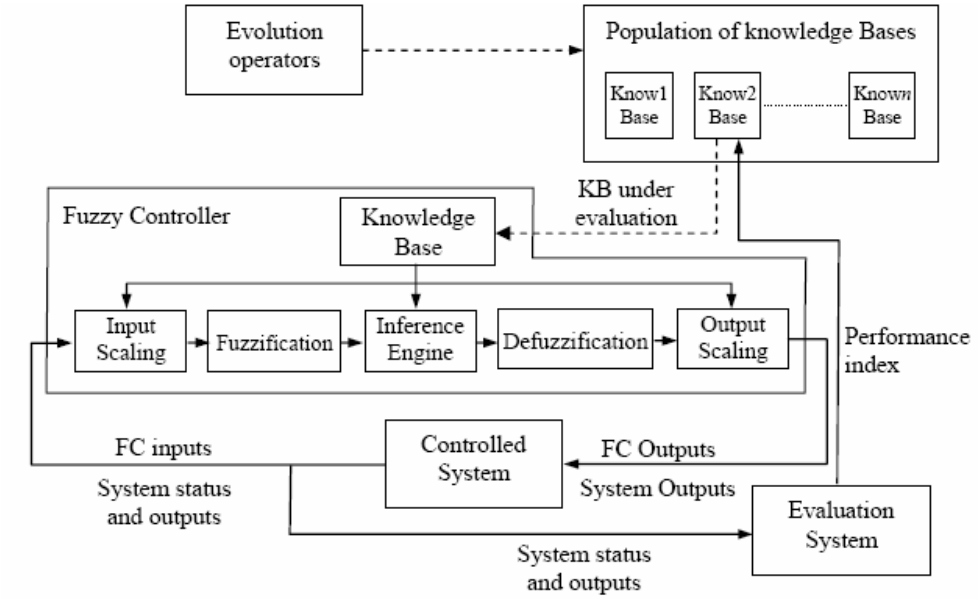


Fig.7-4: Evolutionary learning of an FLC

Most authors propose a genetic learning method for the Data Base (DB) of Mamdani fuzzy rule base system that allows us to define:

• The numbers of labels for each linguistic variable.

• The universe of discourse.

• The form of each fuzzy membership function.

In GA, we only need to select some suitable parameters, such as generations, population size, crossover rate, mutation rate, and coding length of chromosome, then the searching algorithm will search out a parameter set to satisfy the designer's specification or the system requirement.

To automate the optimization process, a meaningful performance-index algorithm is required, which assesses the performance for different operating conditions in the same way as experts. To address this requirement, a fuzzy performance-index is designed. Furthermore, an optimization strategy is required, which tunes the controller parameters stepwise, so that finally the best possible solution is obtained. An Evolutionary Algorithm is used for this task.. The determination of the values of the controller parameters by an Evolutionary Algorithm, as well as the evaluation of the resulting performance by the designed performance-index, is performed by a Personal Computer.

The performance of a fuzzy controller is improved by tuning parameterized membership functions and input-output scaling factors with respect to the desired control behavior. A population of competing chromosomes that encode the tuning parameters evolves by

means of selection, recombination and mutation. The fitness of an individual is evaluated by observing the performance of the fuzzy controller while regulating the process using the membership functions and scaling factors encoded in the chromosome. Over the course of evolution the evolutionary algorithm identifies those set of parameters, for which the fuzzy controller performs optimal with respect to the given performance index.

7.6.1. MICHIGAN APPROACH

The population is composed of a set of individual rules that compete with each other to suggest the optimal control action. The method is suitable for on-line learning tasks as the evolutionary algorithm incrementally improves the performance of the fuzzy controller constituted by the population of rules. The chromosomes are individual rules and a rule set is represented by the entire population the collections of rules are modified over time via interaction with environment. . In *Michigan approach,* data base and rule base are two clearly separate entities. All rules share the same membership functions defined in a common data base. Fuzzy sets are associated with linguistic terms such as *small, medium, large*. A fuzzy if-then rule combines multiple linguistic labels in a way that facilitates an intuitive interpretation of the relationship between input and output states defined by this rule.

7.6.2. PITTSBURG LEARNING APPROACHES

The Pitt learning approaches operate on a population of rule bases. As the evolutionary algorithm evaluates the performance of the fuzzy controller as a whole, rather than that of isolated fuzzy rules, the credit assignment mechanism becomes obsolete. In this approach each chromosome encodes a whole RB or KB. Crossover serves to provide a new combination of rules and mutation provides new rule.

The genetic representation of the fuzzy knowledge base is another important issue in the conception of a genetic fuzzy system. The main distinction is made between *descriptive* and *approximate* knowledge bases. In a *Pittsburg* approach fuzzy system, each rule operates with its own fuzzy sets. Data base and rule base merge as the rule itself contains the parameters of the underlying membership functions. Such a representation possesses more degrees of freedom which permits a more accurate approximation of the desired input-output relationship, hence the term approximative.

The price of increased accuracy is that the resulting fuzzy system becomes more difficult to analyze as an individual fuzzy set no longer coincides with a linguistic concept.

7.6.3 ITERATIVE RULE LEARNING APPROACH

In this approach, as in Michigan one , each chromosome in the population represents a single rule, but contrary to the Michigan one , only the best individual is considered to form part of the solution, discarding the remaining chromosome in the population. Therefore, in the iterative approach, the GA provides a partial solution to the problem of learning. In order to obtain a set of rules, which will be a true solution to the problem, the GA has to be placed with in an iterative scheme similar to the following;

1. use GA to obtain the rule of the system
2. incorporate the rules into the final set of rules
3. penalize this rule
4. if the set of rules obtained till now is adequate to be a solution to the problem, the system ends up returning the set of rules as the solution. Otherwise return to step 1.

A very easy way to penalize the rules already obtained, and thus be able to learn new rules when performing inductive learning, consist of eliminating from the training set all those examples that covered by the set of rule obtained previously. The main difference with respect to the Michigan approach is that the fitness of each chromosome is computed individually, without taking in to account cooperation with other one. This substantially reduces the search space because in each sequence of iterations only one rule is searched.

7.7 DESIGN OF THE PERFORMANCE-INDEX

It is the task of the performance-index to assess the control performance in the same way as so far experts do. The performance-index has to consider to what extent the requirements, such as stability and satisfactory responses to different stimulus signals, or satisfactory responses to disturbances, in consideration of the physical limits of the controlled system, are met. Common performance- indices usually include only parts of these requirements. In comparison, we deal with the task of considering all of the relevant aspects and summarizing them in a single performance-index, that satisfies the following requirements. The value of the performance-index, which is obtained for the considered set of controller parameter values, is used as fitness value of the individual. Here the fuzzy performance-index serves as *fitness function*.

A performance index is a quantitative measure of the performance of a system [42]. It must be a number that is always positive or zero. A system is considered an optimum control system when the parameters are adjusted so that the index reaches an extremum value, commonly a minimum value.

The following are the known performance indices:

• ISE (Integral of the square of the error)

• IAE (Integral of the absolute magnitude of the error)

• ITAE (Integral of time multiplied by absolute error)

• ITSE (Integral of time multiplied by the squared error)

Among the four enumerated indices, ISE is the most common criterion in optimizing a control system. It is minimized by minimizing Areas 1, 2 and 3, as shown in Figure 7.5.

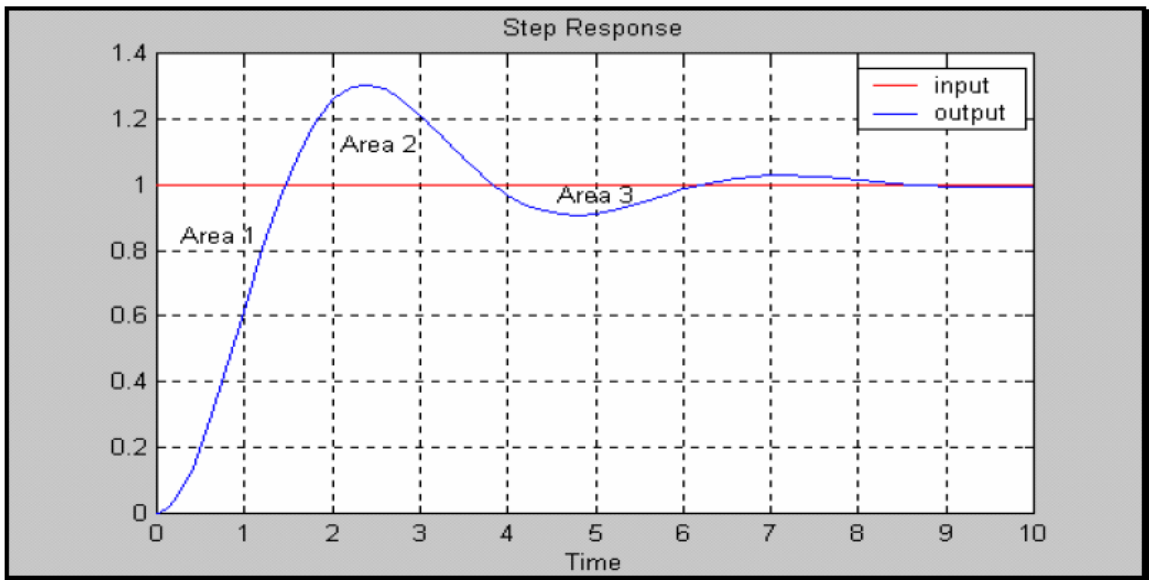


Fig 7.5. The graph of ideal step response (input) and actual step response.

7.8 OPTIMIZATION TECHNIQUES

In optimizing the above performance indices there is a correspondingly appropriate technique or techniques for each of them, which would give a fast or at least guarantee convergence. There are two categories of optimization techniques, namely; the derivative-based and derivative-free optimizations [43]. The former is capable of determining search directions according to an objective function's derivative information, while the latter is stochastic, which means that it uses random number generators in determining subsequent search directions. The most popular derivative-free optimization methods are the following: genetic algorithms (GA), simulated annealing (SA), random search method, and downhill simplex search. They share these common characteristics; derivative freeness, intuitive guidelines, flexibility, randomness and analytic capacity. The random search method and downhill simplex search concepts and implementations are simple but GAs and SA are regarded better for all problems all the time [43]. Due to the parallel-search procedures of GA, it is expected to converge faster and less likely to get trapped in local minima than SA. GAs provide a stochastic optimization method where if they get stuck at a local optimum, they try to simultaneously find other parts of the search space and jump out of the local optimum to a global one

CHAPTER-VIII
FUZZY-GENETIC OPTIMIZATION OF LIQUID LEVEL CONTROLLER

INTRODUCTION:

In order to demonstrate the utility of fuzzy-genetic algorithm, a prototype liquid flow in a closed vessel has been considered. The MATLAB software version 6.5 has been used in this project, which have inbuilt fuzzy logic and a external genetic tool boxes. This has been released by MATWORK company USA..

8.2 PROBLEM FORMATION:

- Design and development of Fuzzy logic controller for the given prototype process level control.
- Optimization of Fuzzy logic controller by using genetic algorithm. :Genetic algorithm has been used to tune the output membership function to obtain minimum integral square error (ISE) as performance index.

8.3 PROCESS DETAILS

The system includes an inflow and outflow, both of which is adjustable by using electronic valves. The controller that is developed has applied to a scaled down version of the simulated tank. The actual tank has digital sensor and pumps that accept digital control. The Objective of this project is to develop a heuristic fuzzy logic controller capable of driving the liquid level in the tank to a given set point. In this project the level of the liquid will be controlled by adjusting only the input or inlet valve. The prototype of the process is shown below

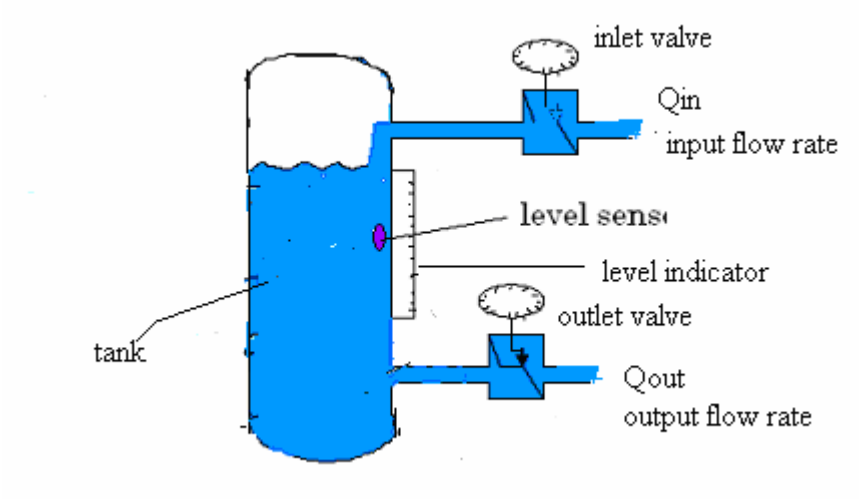


Fig8-1.A tank containing liquid

The liquid level and the rate at which this level is changing are initially set to arbitrary values. The control objective is to drive the system to a given set point in the shortest time possible by adjusting the inflow and outflow.

It is not particularly difficult for human to develop rules that when applied to the system drive the liquid level to a value that is relatively close to the set point. The rules for controlling the system are almost trivial: if the level is above the set point then remove liquid, if it is below the set point then add liquid.

There are numerous approaches to FLC development but in this project we will use straight forward approach to develop FLC for liquid level system. A step by step procedure for fuzzy control of the liquid level system is provided below. In designing special care has been taken to relate the mechanics of FLC to the method of human might use to control the liquid level system. We think of our system as models of the action taken by expert human operator and later we will use genetic algorithm to optimize the performance of the system.

8.4 DESIGN OF FUZZY LOGIC CONTROLLER

The first step in developing the liquid level Fuzzy Logic Controller is to select the variables that will be important in choosing an effective control action. The variables are known as the *condition variables.* Two condition variables are readily identified as being important in the liquid level system. Since the control rules depend not on the current value of 'h'(height) and Δh (rate of change of height), rather on the respective distance from the set point (ERROR) , the rules presented for the proposed system use the following two condition variables;

1. ERROR(E)……. $E=h_{set}- h$
2. CHANGE IN ERROR (ΔE)…… $\Delta E=dE/dt$

Where h_{set} is the desired fixed set point for the level h

$\Delta E=dE/dt$ is the rate of change of error

These variables are **input variables** to FLC controller, both of which are to be driven to zero in our system.

Once the condition variables have been chosen, the second step is to choose the *action variable*. In our liquid level system identifying the output variable (action) is straight forward task. Fore there are two things the controller can adjust to alter the state of liquid level system; either the input flow rate Qi (adjustment of input electronic valve) or the output flow rate Q_o(adjustment of output electronic valve), which can be increased or decreased by adjusting the valve. The task of our controller is to adjust the input electronic valve according to desire set point value.

Once the important conditions and actions have been identified the next step is to define the linguistic terms (values) used to describe these linguistic variables i.e. fuzzy sets for each variables. These fuzzy sets are written to describe the condition variables E and ΔE and action variable input Knob setting(kQ_{in}) . It must be understood that a greater number of linguistic terms can be expected to give precise control. However, when more linguistic terms are used describe condition variables more rules are needed by fuzzy logic controller. Since the number of rules or complexity increases exponentially with the number of variable and linguistic terms as much as possible moderate linguistic terms should be used in design, most literatures shown, linguistic terms in the range of 3-7

works fine for many control system. Choosing the number of linguistic terms used to describe each variable is not an arbitrary decision it is based on knowledge gained through working with physical system.

For our case, we will choose

i. four fuzzy sets to characterize E
   - Ngative Big(NB)
   - Ngative Small(NS)
   - Positive Small(PS)
   - Positive Big(PB)

ii. five fuzzy sets to characterize ΔE
   - Ngative Big(NB)
   - Ngative Small(NS)
   - Near Zero(NZ)
   - Positive Small(PS)
   - Positive Big(PB)

iii. five fuzzy sets to characterize kQ_{in}
   - Ngative Big(NB)
   - Ngative Small(NS)
   - No change(NC)
   - Positive Small(PS)
   - Positive Big(PB)

All the above fuzzy sets were chosen because they are similar to the descriptive terms human operator might use to control the liquid level system.

After choosing the number and name of linguistic terms the next step is to delimit the range in which the corresponding linguistic term can take a value for each linguistic variable. The following tables indicate the range and the shape of each fuzzy set for corresponding variable. The ranges are normalized so that to lie between [-1 1].

Table 8.1. ERROR – total range [-1 1]

Linguistic term	range	Shape and parameter value
NB	[-1 -0.4]	Triangular –param -[-1 -1 -0.4]
NS	[-0.8 0.2]	Triangular –param -[-0.8 -0.4 0.2]
PS	[-0.2 0.8]	Triangular –param -[-0.2 0.4 0.8]
PB	[0.4 1]	Triangular –param -[0.4 1 1]

Table8. 2. CHANGE IN ERROR (ΔE) - total range [-1 1]

Linguistic term	range	Shape and parameter value
NB	[-1 -0.5]	Triangular –param -[-1 -1 -0.5]
NS	[-0.8 -0.2]	Triangular –param -[-0.8 -0.5 -0.2]
NZ	[-0.4 0.4]	Triangular –param -[-0.4 0 0.4]
PS	[0.2 0.8]	Triangular –param -[0.2 0.5 0.8]
PB	[0.5 1]	Triangular –param -[0.5 1 1]

Table 8.3. adjustment of input electronic valve (kQ_{in}) - total range [-1 1]

Linguistic term	range	Shape and parameter value
NB	[-1 -0.5]	Triangular –param -[-1 -1 -0.5]
NS	[-0.77 -0.25]	Triangular –param -[-0.75 -0.5 -0.25]
NC	[-0.5 0.5]	Triangular –param -[-0.5 0 0.5]
PS	[0.25 0.75]	Triangular –param -[0.25 0.5 0.75]
PB	[0.5 1]	Triangular –param -[0.5 1 1]

The above number of linguistic terms for each variable allowed for an FLC of reasonable size while providing adequate size while providing adequate control for liquid level system. With thus choice of fuzzy sets 5 x 4= 20 different production rules are required to describe all of the possible conditions that in the liquid level system. With 4 fuzzy sets for

E and 5 fuzzy sets for ΔE as shown above there are 4x5=20 possible conditions in the liquid level system and a human expert provide a desired action for each condition based on prior experience.

The rules are of the form;

IF E is A and ΔE is B THEN kQ_{in} is C.

Where A ,B and C are linguistic terms represented by fuzzy sets.

An example of fuzzy production rule used in liquid level system is as shown below;

IF E is PB and ΔE is PB THEN is kQ_{in} NB.

This rule simply says that if the liquid is well above the set point and rising rapidly the net flow in to the tank should be made Negative Big.

The complete rule set for the liquid level FLC is depicted below

Table8. 4. Rule matrix

		ERROR (E)			
		NB	NS	PS	PB
Δ E	NB	PB	PB	PS	NS
	NS	PB	PS	NC	NS
	NZ	PB	PS	NS	NB
	PS	PS	NC	NS	NB
	PB	PS	NS	NB	NB

This rule matrix is used by locating the descriptive term E along he top matrix ,locating the descriptive term for Δ E along the left side of the matrix and then extracting the appropriate value kQ_{int} for the given condition.

The method for giving precise meaning to the fuzzy, linguistic variables involves the use of fuzzy membership function. The fuzzy membership function used in the liquid level control system ,which gives meaning to linguistic terms used in the rule, are shown in the figure below

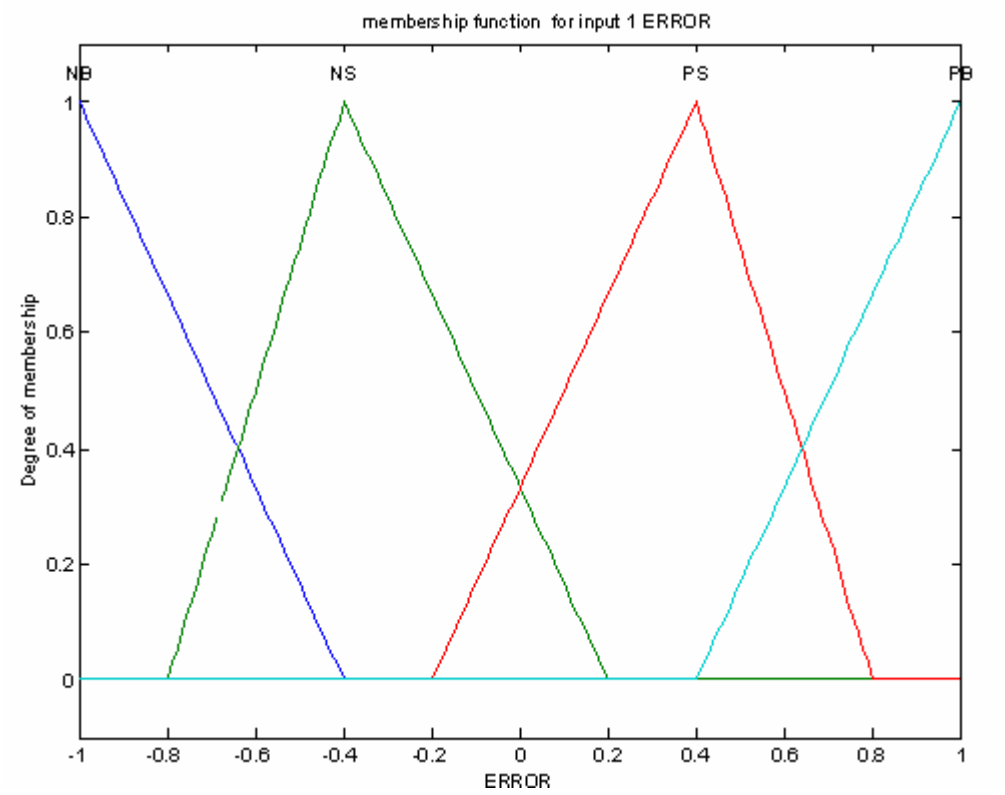


Fig 8.2 membership function of input 1 ERROR

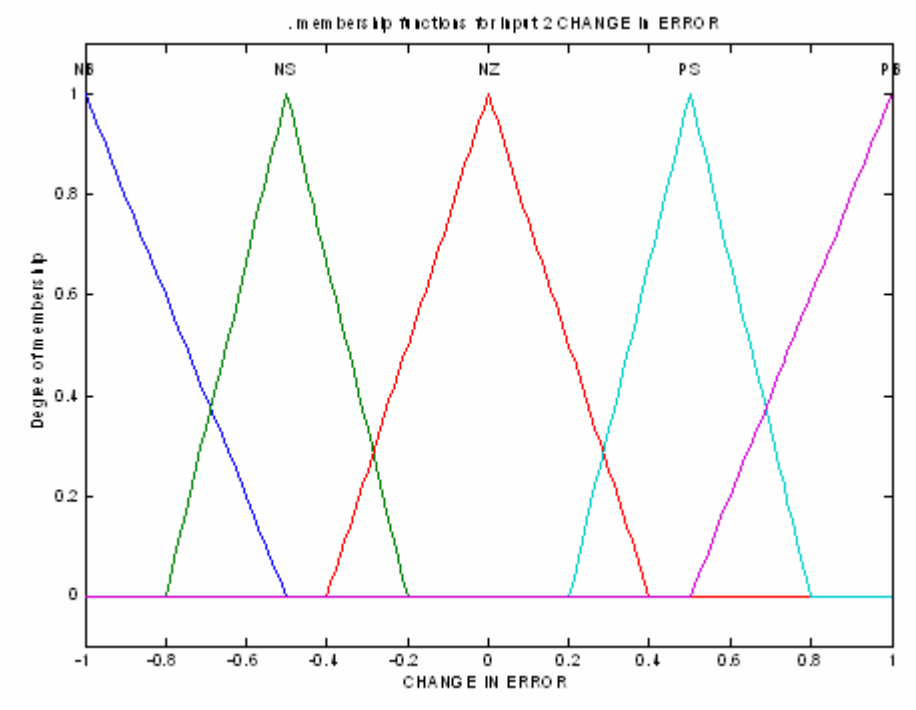


Fig 8.3 membership function of input 2 CHANGE IN ERROR

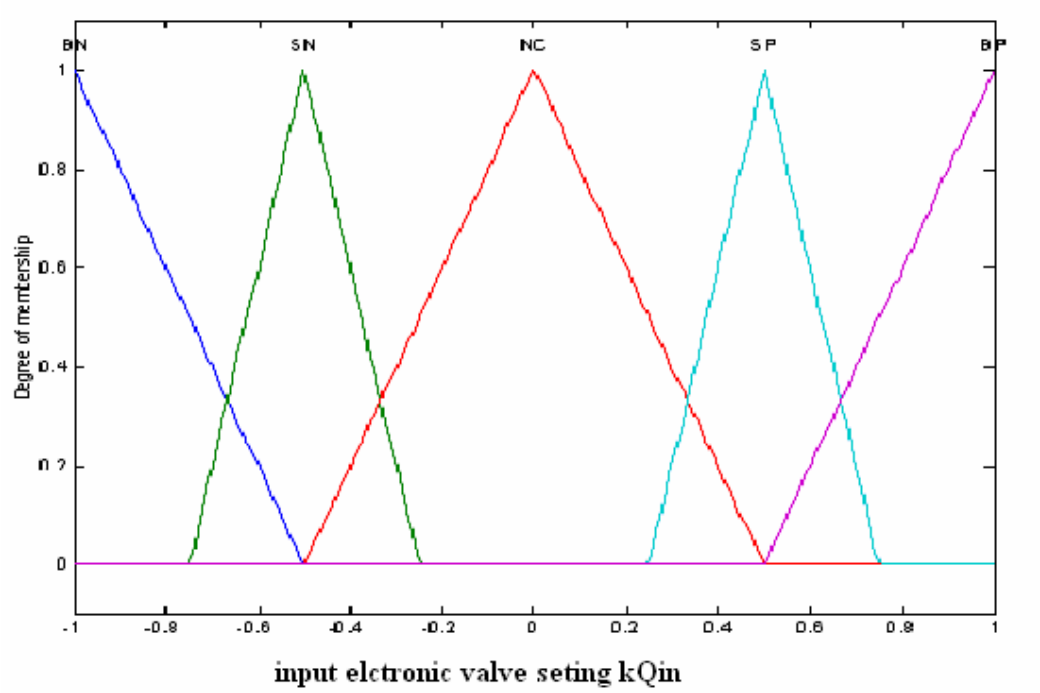


Fig 8.4 membership function of input valve setting (kQin)

The above fuzzy membership functions allow the crisp value of E and Δ E to be transformed in to a fuzzy membership value between 0 and 1. Actually the fuzzy membership can be thought of as determining the degree to which a discrete value of a variable is described by a particular fuzzy linguistic terms. The crisp conditions (definite value of E and Δ E) existing in liquid level system at any given time can be accounted for

using production rule that include fuzzy linguistic terms. Now process for determining a crisp action to take on liquid level system must be developed. In other word the crisp conditions have been fuzzified and then defuzzified to yield a single action to the system. The set of fuzzy production rules provides a fuzzy action for any condition that could possibly exist in the problem environment.

Unlike conventional expert system where only one rule is applicable for any given set of conditions, all of the rules in the FLC take effect to some degree at every time step. There fore there still remains the task of converting the 20 fuzzy actions prescribed by the fuzzy production rules in to a single crisp action to be taken on the liquid level system.

In our system center of area or gravity (COA/COG) method of defuzzification is used which is also found in fuzzy tool box of MATLAB 7.

$$u = \frac{\sum_{i=1}^{n} \mu(u).u}{\sum_{i=1}^{n} \mu(u)}$$

This value defines the single crisp value of the out put may be voltage value to adjust the input electronic valve according to the desire set value. The rules that create triangles with large area, those arising from conclusions that are most applicable, have the greatest effect on the action.

We have a means for converting a crisp set of measurement in the liquid level system to a set of fuzzy condition and a set of fuzzy production rules prescribing a fuzzy action associated with a particular set fuzzy condition has been developed. This step will be done by using ‘MATLAB ‘fuzzy control tool box which automatically calculate the required value of variable by using built in function according to written program.

The following diagram is the result of program in appendix A m-file *wlc.m* which shows the fuzzification and defuzzification phase of the controller.

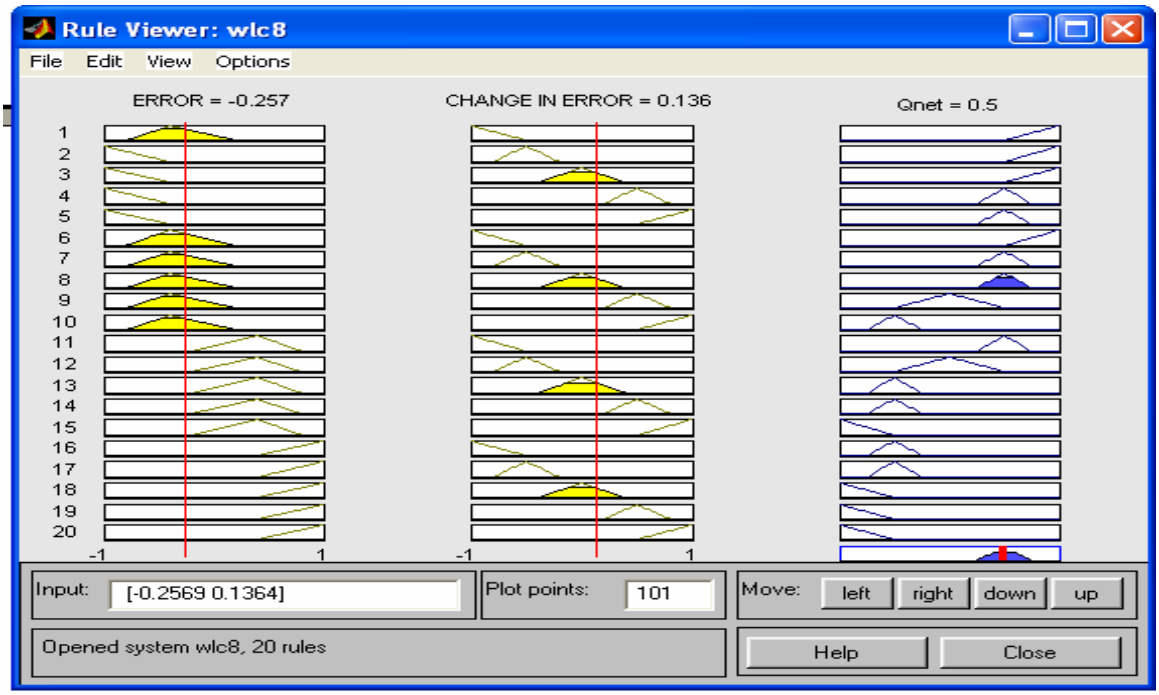


Fig8.5. fuzzification ad defuzzificaation phase of FLC for liquid level system for particular crisp inputs.

The yellow shade indicates those rules that have been fired for a given particular input and the blue shade clipped triangle indicates the fuzzified output value where as the red line with blue triangle indicates the defuzzified value.

MATLAB fuzzy tool box has also an option to visualize the interaction between inputs and outputs by displaying the surface view of the variable .For liquid level control system the surface view is shown below

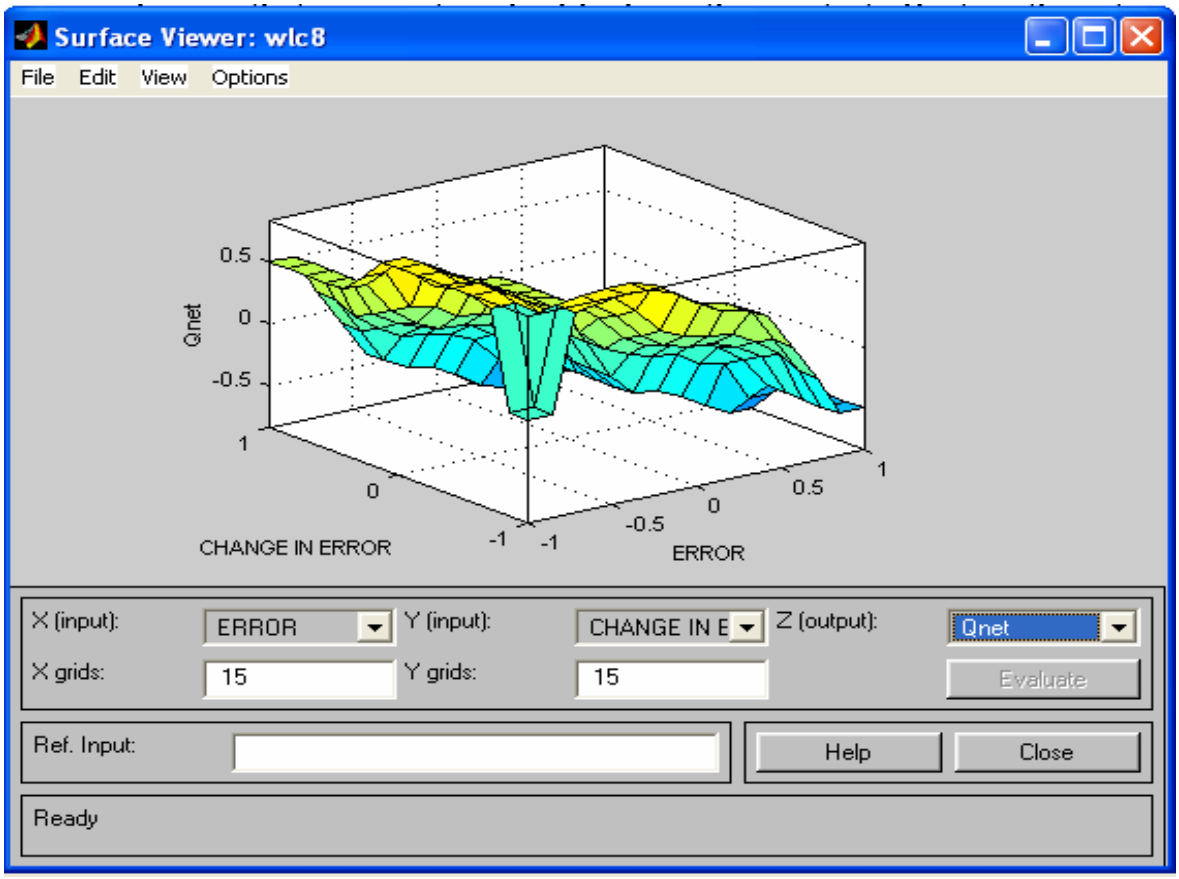


Fig8.6. surface view of E and Δ E

The step by step procedure described above for developing an FLC is summarized as

1. Determine the condition variables to be considered. These are the variables that are measured when a change to the system considered.
2. Determine the action variables. These are the variable that are manipulated to elicit a change in the system state
3. Describe the fuzzy sets for both the condition and action variables
4. Establish fuzzy production rules that cover all of the possible conditions that exist in the problem environment
5. Define the fuzzy membership function
6. determine all condition membership value for each rule
7. apply fuzzy production rule by taking the weighted average of the action prescribed by all of the rules. The fuzzy production rule are generally written in matrix form table and apply deffuzification method suitable for the problem at hand.

8.5 GENETIC OPTIMIZATION OF FLC FOR LIQUID LEVEL CONTROL

The objective of genetic fuzzy system is to automate knowledge acquisition step in fuzzy system design ask that is usually accomplishes through an interview or observation of a human expert controlling the system. It adapts either all part or the entire component of the fuzzy knowledge base. As mention in the previous chapters the knowledge base contains the data base and rule base. According to the distinction between the data base and the rule base in fuzzy system, genetic fuzzy systems are discriminated along two major approaches;

- genetic tuning process and
- genetic learning process

The first method is targeted at optimizing the existing fuzzy system. The tuning process involves the adaptation of the fuzzy data base, *namely parameter of membership function and input output scaling factor.*

The second method is concerned with the automatic derivation of *fuzzy rules* in the rule base. A genetic learning process faces a much more difficult task as it has to establish the proper relation ship between input and out put state from the scratch rather than optimizing the performance fuzzy system that already operates at least approximately correct.

In both cases, the optimization criteria is the problem to be solved at hand and the search space is the set of parameters that code ;

i) the membership function parameters
ii) scaling function
iii) fuzzy rules

in our design we will use the first approach i.e. tuning the data base of fuzzy controller. As pointed in different literatures, when applying GAs to FLC there are two basic decisions to made;

1) how to code the problem solutions to the problem as finite bit string
2) how to evaluate the merit of each bit string, the fitness function

the tuning method using GA fits the membership functions of the fuzzy rules dealing with the parameters of the membership functions and obtaining high performance that is making minimum an error function defined by means of input -output data set for evaluation.

The heuristic fuzzy controller designed for liquid level system in pervious section may not be work in optimal condition. The reason is that the rule that is loaded in to the rule base and the membership function for each input and out put simply came from operator of the plant or from the knowledge gained by the designer through experience. This means the efficiency of the controller to meet the given objective is depend up on the interviewed operator and the knowledge of con trolling the particular plant differs from operator to operator. This is the reason to search some means to optimize the controller efficiency in order to meet the given performance criteria.

As seen in the heuristic FLC design section the development was straight forward and produced a controller that worked well. However, there are factors that dramatically affect performance and so must be carefully adjusted by the developer for optimal control. The factors are;

- the shape of the membership function
- the location of the membership function
- the action value that characterize each linguistic rule

The shape and location of membership function determines the nature of the interpolation between the anchor points. The action value of the rules locate the anchor points.

At the beginning of this project a through literature search was conducted to discover published results involving fuzzy system tuning. However, this search failed to reveal any accepted or even an efficient algorithm for defining membership functions. At best the selection of membership function was a difficult time consuming task ; at worst , it was the task entwined with other facts of fuzzy system such as the choice of implication operator.

However, genetic algorithm was a natural tool to use to define the membership function and rule of the controller. The use of GA solves a rather difficult and time consuming aspect of fuzzy system development. The difficult is the tuning of fuzzy a fuzzy system so that it efficiently solves a particular control problem. Tuning of a fuzzy system mainly refers to the definition of the membership function and to the formulation of efficient rule set. Both the membership function and the rule set must be adjusted simultaneously to obtain a well tuned and efficient fuzzy controller.

This section focus on the details necessary for employing a GA to improve the performance of the simple liquid level FLC presented in the previous section. The key issue of selecting the coding scheme a fitness function for this problem will be addressed.

Results will be presented comparing the effectiveness of the heuristic FLC with GA optimized FLC for liquid level control system.

8.5.1 Genetic algorithm for membership function adjustment.

As mentioned in the previous section, to optimize any problem by using genetic algorithm one should address the points how to present or code possible solutions to search the problems as a string of character and how to evaluate the effectiveness of possible solution using a fitness function. Thus considering the above two points in relation to fuzzy membership function determination for liquid level fuzzy control, the coding scheme and constraints as well as fitness function will be addressed in the following section.

Coding scheme

The liquid level fuzzy controller presented in previous section employed triangular membership function. Thus, the following presentation describes the use of a GA to adjust triangular membership function. However, the same method can be applied to other type of membership functions.

The un-optimized, arbitrary selected triangular membership function used initially for (heuristic FLC) the liquid level controller were shown in the fig 8.2, 8.3 and 8.4. There are four fuzzy sets to describe error E ,five fuzzy sets used to describe the time rate of change of error (ΔE) and five fuzzy sets used to describe the valve setting (kQ_{in}). These fuzzy sets provide linguistic terms that can describe both condition (E and ΔE) and the action (kQ_{in}) variable. With this definition, there are 20 rules in the controller. The membership function determines under precisely what condition and to what extent each rule is applied for particular value of E and ΔE. Improving these memberships function improves the performance of the controller.

Constraints in optimizing membership function by GA for liquid level control (LLC)

1) As a constraint on GA's search, the membership function in the liquid level controller were forced to maintain their triangular shape and their height but the size of the base and their position relative to one another was allowed to change.
2) The triangles describing the extreme membership function; those right triangles at the upper and lower limit of the variable (NB and PB) were forced to remain fixed

to their associated extreme limits. In other word , they were forced to remain right angle triangle with their apex at either the lower or upper limit of the condition or action variable they describe. Thus the definition of each triangle requires a single point a left or right base point.

3) Membership function defining interior membership functions were forced to remain isosceles triangles thus the definition of interior membership function requires two points both left and right base point.

The number of point that must be defined under these constraints are shown below to completely define the membership function for E , ΔE and kQ_{in}.

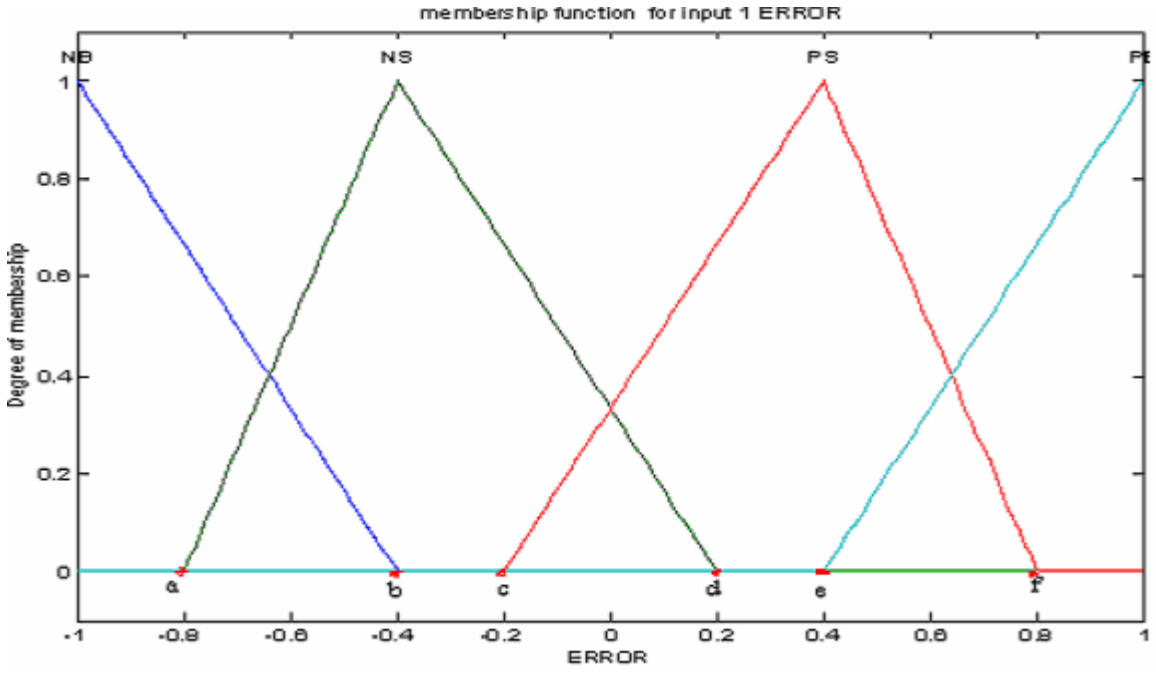


Fig 8.7 a, b, c ,d ,e, and f – parameters of error membership function for GA optimization

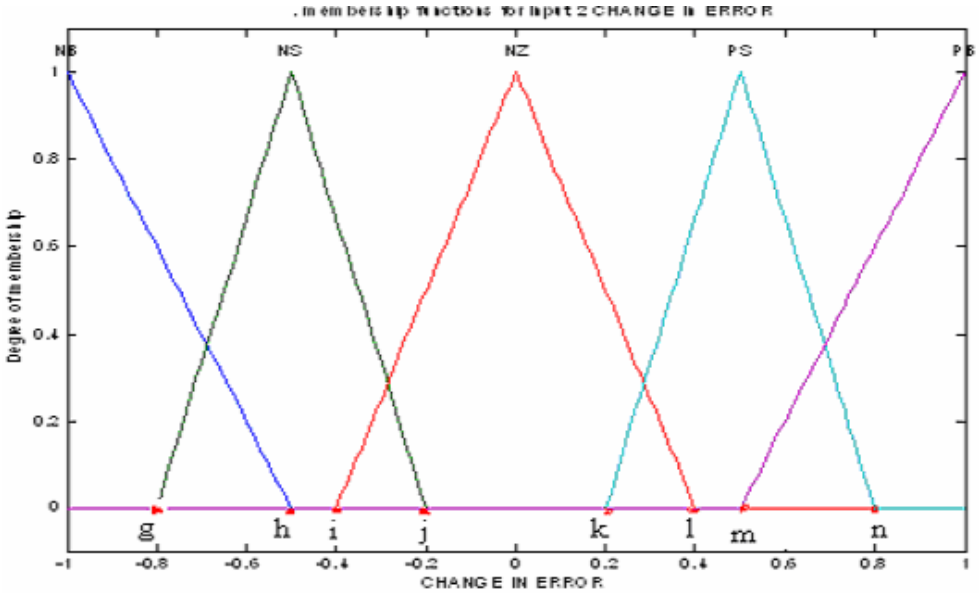


fig8.8 g,h,i,j,k,l,m and n --parameters of change in error which optimized by GA.

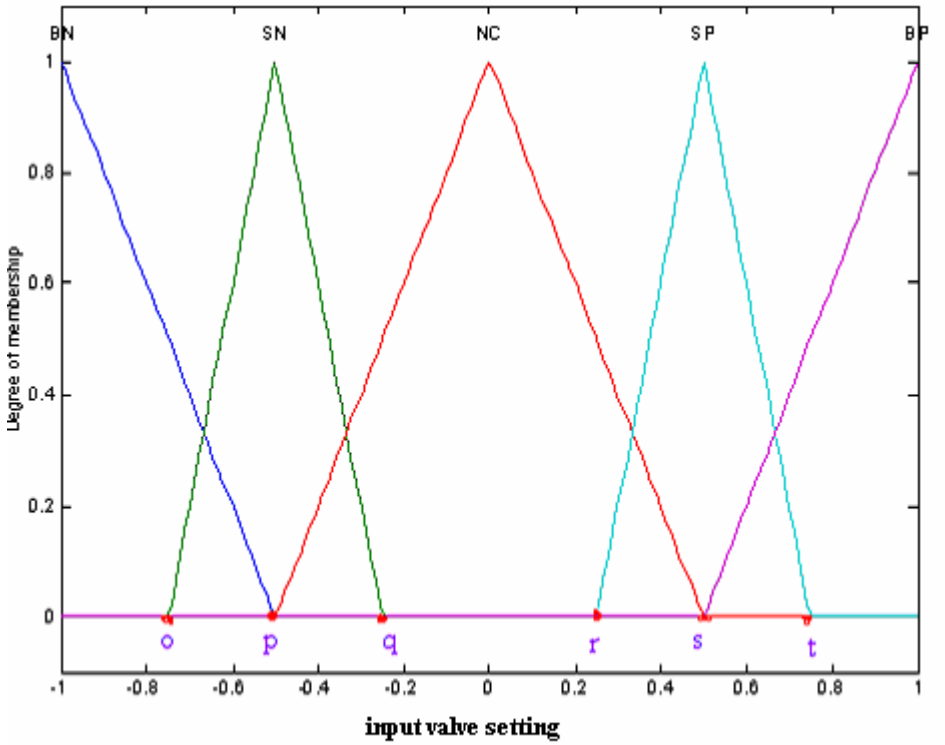


fig 8.9 o,p q r s t --parameter optimized by GA **input valve setting kQin**

When the member ship function associated with E , ΔE and kQ_{in} are considered the result is the search problem with 20 parameters i.e. 6 points (a-f) for E. 8 points (g-n) for ΔE and 6 points (o-t) for kQ_{in}. .However in this project we will only optimize the output membership function by using genetic algorithm.

Thus the search for efficient membership function for the liquid level control problem defined in the previous section for heuristic FLC involves the selection of 20 parameters for optimal control. This is a much larger search problem than the simple function optimization dealt in chapter 6. However, the application of genetic algorithm to search problems requires only the coding of the parameter and definition of fitness function; the size of the problem does not prevent the use of GA.

In section 6.1 the simple function optimization we have solved with GA using a concatenated, mapped, unsigned binary coding. In this coding parameter that varies continuously between some *minimum* and *maximum* value are represented with bit string. This same approach is applicable to the points defining triangular membership function in the current problem. A substring is allocated a specified number of bits, *L*, and mapped using the familiar formula

$$X_i = x_i^{(L)} + \frac{x_i^{(U)} - x_i^{(L)}}{2^{l_i} - 1} \times decodedvalue(s_i)$$

where the variable X_i can take the value from a domain $D_i=[x_i^{(L)},x_i^{(u)}]$ and is coded in substring S_i of length *l*. the decoded value of a binary substrings S_i is calculated as

$$\sum_{i=0}^{l-1} 2^i s_i$$

, where $S_i \in (0,1)$.

Each of the remaining parameters can be represented with similar substrings, and the entire parameter set can be represented by a string formed through concatenation of the individual substrings. Again, the GA manipulates the population of strings using the familiar of reproduction, crossover and mutation.

The remaining task includes determining the number of bits to be allocated for each of the 20 parameters and the selection of appropriate minimum and maximum value for each parameter. To keep things simple and because there is no apparent reason for using substrings of different length, each substring is represented with *4-bits* thus the resulting string are thus 20 x4bit=80-bit in length. These are chromosomes that the genetic algorithm will use fore optimization. In other problem it is possible to use substrings with more or fewer bits as well as different length of substring.

Initially, the selection of minimum and maximum value for each of the parameter seems to be an obvious decision; it seems that the minimum and maximum parameter value should simply coincide with the lower and upper limits of available range i.e. the minimum value of the parameter for the membership function describing E is -1 and the maximum value is 1 and the same for other variables. However the selection of minimum and maximum value for each membership function is at present subjected to the following three constraints.

In this project only output membership function had been optimized by genetic algorithm thus the resulting string are 6x4bit=24 bit in length. The optimization of input membership functions is left for future work.

Constraints in selecting min and max or lower and upper bound value for linguistic terms for liquid level control system

1) The first constraints require that each parameter value fall in the range selected by the user to represent each variable. For example, for E the linguistic terms range should lie in between -1 and 1..[-1 1] and the same is true for other linguistic variable's linguistic terms.
2) The second constraint is that the definition of the linguistic term should be consistent with the normal meaning of the terms. For example, the membership function associated with the term NB should be *'zero'* for positive numbers and non-zero for small negative number.
3) The third constraint is that each point in the range being described should have at least one membership value assigned to it i.e. membership function must be constructed so that all relevant points have a membership value.

According to the above constraints the following table shows the min and max (bounds) value of each linguistic variable's linguistic terms. The point also assures that all points are associated with appropriate membership values. Here, each point is allowed to slide within pre-defined window bounded by values of min and max for each parameter. The permitted ranges are selected by taking in to account the range must assure that there will be some degree of overlap of the membership functions.

Table 8.5 Min and max value of input valve setting (kQ_{in}) linguistic variable- linguistic terms

Parameter Name	Minimum value	Maximum value	Bound [min max]
o	-0.9	-0.6	[-0.9 -0.6]
p	-0.7	-0.4	[-0.7 -0.4]
q	-0.43	0	[-0.43 0.0]
r	0.1	0.3	[0.1 0.3]
s	0.3	0.6	[0.3 0.6]
t	0.7	1	[0.7 1]

8.6 FITNESS FUNCTION AND SIMULATION OF LIQUID LEVEL CONTROL SYSTEM

The fitness function selected for output fuzzy variable is defuzzification function D_f

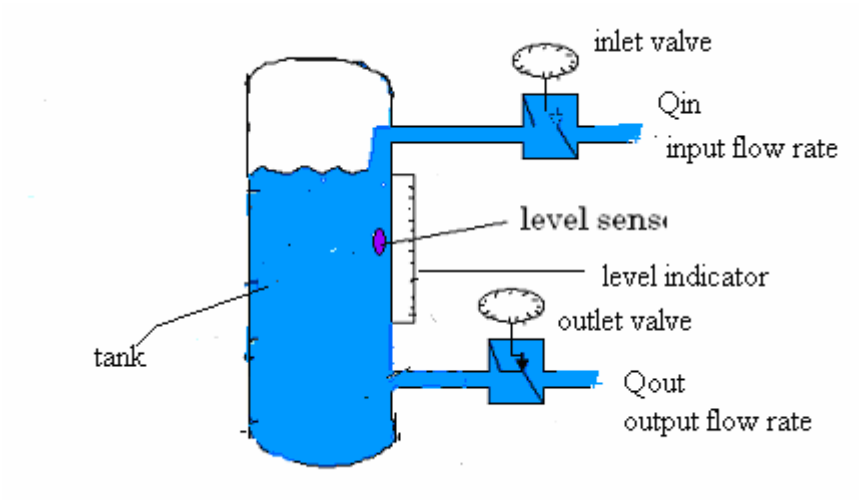
$$D_f = \frac{\sum_{i=1}^{n} \mu(u).u}{\sum_{i=1}^{n} \mu(u)}$$

to vary the parameter of membership function and ISE integral square of error as performance measure.

$$\text{ISE} = \int_0^T e^2 \, dt$$

8.6.1 DERIVATION OF TRANSFER FUNCTION OF LIQUID LEVEL CONTROL SYSTEM

The schematic diagram of liquid in the tank is shown below.



The above figure indicate that the control of the height of liquid in the tank being filed. The parameter of the system are defined as follows

- Q(t); the angular position of the valve in degree that controls the liquid in to the tank
- Q_{in} the rate of flow of liquid in to the tank at time t
- Q_{out} the rate of flow of liquid out of the tank at time t
- h(t) the height of liquid at time t

The liquid level in the tank is remains constant when the inflow rate q_{in} ia equal to the outflow rate q_{out} i.e. the error between inflow and outflow is zero. If the inflow rate is greater than outflow rate, the liquid level will rise from the set point. If the inflow rate is less than the outflow rate, the level will fall. During a certain time interval Δt, the amount of liquid in the tank will change by an amount ΔV equal to the average difference between the inflow rate and the outflow rate multiplied by Δt.

$$\Delta V=(q_{in}- q_{out})_{avg}*\Delta t \quad m^3$$

the change in the liquid level in the tank Δh is equal to the change in volume (ΔV) divided by the cross sectional area of the tank(A);

$$\Lambda h = \frac{\Lambda h}{A} = \frac{(\text{qin} - \text{qout})\text{avg} * \Lambda t}{A}$$

the average rate of change of the level in the tank is equal to the change in level Δh divided by the time interval Δt i.e

$$\textit{average rate of level} = \frac{\Lambda h}{\Lambda t} = \frac{(\text{qin} - \text{qout})\text{avg}}{A}$$

for example if the level changed 0.26m during a time interval of 100sec, the average rate of change of level would be 026/100 =0.0026m.

When the time interval diminishes to 0 sec we call it an instant of time. As time interval Δt diminishes to an instant of time ,the average rate of change become instantaneous rate of change of liquid level is called *derivative of level h with respect to time t,* designated by dh/dt. If the time interval Δt diminishes to an instant ,the average rate of change of level becomes the instantaneous rate of change dh/dt i.e.

$$as \quad \Lambda t \rightarrow 0 \quad \frac{dh}{dt} = \frac{(\text{qin} - \text{qout})}{A}$$

let us assume the out flow from the tank is linear (laminar flow)[51] the outflow rate q_{out} is given by the equation;

$$q_{out} = \frac{\zeta \text{ gh}}{\text{R}_{\text{L}}} m^3 / \text{sec}$$

where q_{out};output liquid flow rate m^3/sec

ζ ; liquid density kg/m^3

g; gravitational acceleration

h; liquid level in m

R_L; laminar flow résistance $pa.s/m^3$

There fore

$$\frac{dh}{dt} = q_{in} - \frac{\zeta \text{ gh}}{\text{R}_{\text{L}}} m^3 / \text{sec}$$

$$R_L\left[\frac{A}{\zeta g}\right]\frac{dh}{dt} = \frac{R_L}{\zeta g}q_{in} - h$$

The term $\frac{A}{\zeta g}$ is the capacitance of the C_L of the tank

The entire term $R_L\left[\frac{A}{\zeta g}\right]$=$R_L$ C_L is called the time constant τ of the liquid

The term $\left[\frac{R_L}{\zeta g}\right]$ is the steady state gain G of the system

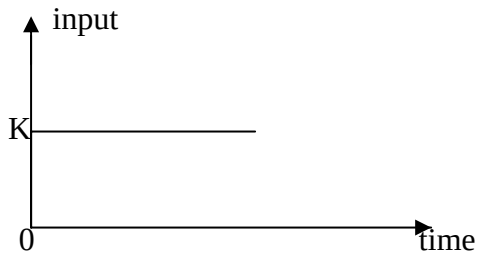
Substituting τ and G in the preceding equation gives us the final form of the differential equation of the liquid tank;

$$\tau\frac{dh}{dt} + h = Gq_{in}$$

let us take the laplace transform of the above equation for further analysis . let us assume before time t=0 sec , the thank is empty and the input flow rate is zero i.e.

$h=q_{in}=0$ for $t<0$. at time t=0 , the input valve is opened and the input flow rate changes to K cubic meter/sec i.e. $q_{in}=K$ for $t>0$. this type of change is step change in the input signal q_{in}. one may ask what is the level of the liquid in the thank after a step change in input i.e. h(t)? for t>0.

To answer this question let us take laplace transform of the above differential equation



$q_{in}=0 \quad t<0$

$=K \quad t>0$

$$\tau \frac{dh}{dt} + h = Gq_{in}$$

$Q_{in}=K/s$ …….step input

$$\tau sH(s) + H(s)=GQ_{in}(s)$$

Operational calculus

algebra

$h=GK(1-e^{-t/\tau})$ ← inverse laplace transform

$H=GK/s(\tau s+1)$

the laplace transform is of the form

$$H(s) = \frac{GK}{s(\tau s + 1)}$$

Where

$\tau = R_L\left[\frac{A}{\zeta g}\right] = R_L C_L$ is called the time constant of the liquid

$\tau = R_L C_L$

R_L; laminar flow résistance pa.s/m^3

$C_L = \frac{A}{\zeta g}$ is the capacitance of the tank

$G=\left[\frac{R_L}{\zeta g}\right]$ is the steady state gain of the system

Assuming the liquid as water the above constants will be [51](taken from design data table)

ζ =1000kg/m^3
g=9.81m/s^2
A=2m^2
R_L=5.6 x10^5 pa.s/m^3
$C_L=\frac{A}{\zeta g}$ =2/1000*9.8
=2.04e-4
$\tau=$ $R_L C_L$
=5.6 x10^5 pa.s/m^{3}x 2.04e-4
=114s
$G=\left[\frac{R_L}{\zeta g}\right]$
=5.6 x10^5 pa.s/m^3/1000*9.8
=57s/m^2

Assuming step input K=1

$$H(s)=\frac{GK}{s(\tau\ s+1)} = H(s)=\frac{57\times 1}{s(114s+1)}$$

$$H(s)=\frac{57}{114s^2+s}$$

8.6.2 SIMULATION OF LIQUID LEVEL CONTROL SYSTEM

With the selected GA operator combination, the developed simulator will be allowed to autonomously optimize the parameter of the controller. Further tuning will be done to reach a satisfactory optimum ISE. Several trials will be run for every adjustment to have conclusive results. ISE and *generations* has been tabulated for analysis. The system block diagram in Figure 9.2 was generated with the aid of *Simulink*. This represents a liquid level control system. As seen from the figure, the diagram consists the following blocks:

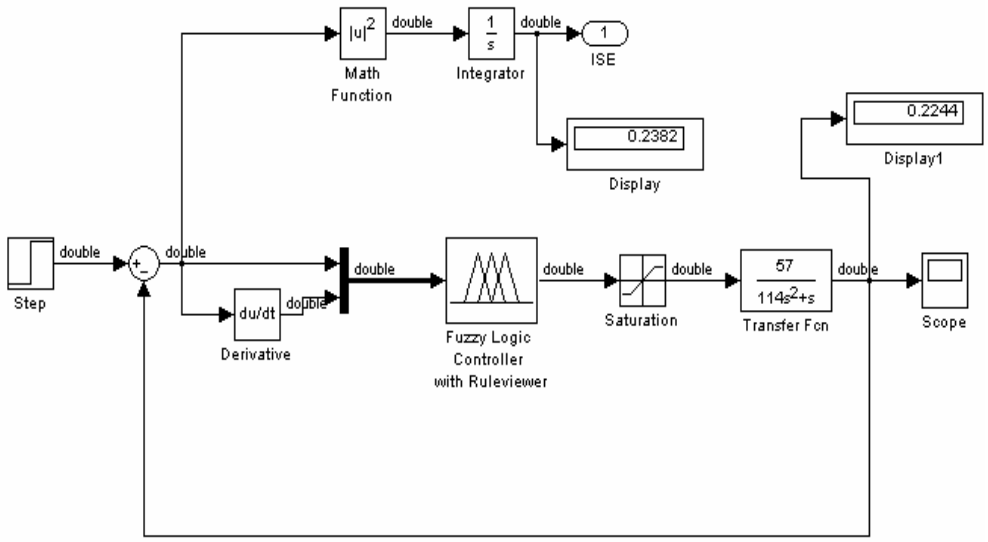


Fig 8.10. Liquid control system simulink block diagram.

- Step Source , Math Function , Integrator , Output Fuzzy Logic Controller, Saturation , transfer function of plant

The *step source block* simulated the input flow to be supplied to the plant under control which is a *thank containing liquid* and is depicted by the *zero-pole block-set*. As the chosen performance index criterion of this study, the ISE was calculated by the *math function* and *integrator block-sets.* These two block-sets implemented the ISE formula,

$$\text{ISE} = \int_0^T e^2 \, dt$$

where T is the finite time chosen arbitrarily so that the integral approaches a steady-state value. The computed value of the ISE was then captured by the *output block* labeled *ISE*. To control the height of the liquid in the tank, FLC was used in this study. It is represented by the *fuzzy logic controller block* in the diagram above. Non-linear application is the area where FLC proves to be more effective than many traditional control techniques. To be in that benchmark, the system was made non-linear by integrating a *saturation block* into the system.

8.7 OPTIMIZATION PROGRAM CODING

This work proposed a methodology on how to apply GA, derivative-free optimization technique, to tune fuzzy controller in a MATLAB environment. In this study, liquid level control system whose transfer functions are modeled in section 8.5 is used as a testing set-up. The fuzzy controller is designed to have a triangular output membership function with a variable width, a fixed 20 fuzzy rules and a fixed triangular input membership function. A genetic optimization program has been developed to automatically tune the controller. As illustrated in Figure 9.1 below, this program update the controller's parameters, which are the width of the output membership functions. It will then simulate the model developed in *Simulink*. The model will get the parameters for simulation and return ISE. This iterative process will go on until the desired minimum ISE is attained.

Since this study intends to apply GA in the optimization of a fuzzy controller, a genetic optimization program has been developed. This program has been coded using MATLAB. It utilizes functions from the Genetic Algorithm Optimization Toolbox for MATLAB. The mechanics of this program is described and illustrated in Figure 9.1. A program had been coded under a filename, *gafuz.m*. Refer to Appendix C for the source code of this routine. This program utilized the *genetic* function from the Genetic Algorithm optimization tool box for MATLAB. With the selected GA operator combination, *gafuz.m* was allowed to autonomously optimize the ISE of the system. Further tuning on the termination option was done since satisfactory optimum ISE was not reached. Five trials were run for every tuning. ISE and *generations* were then tabulated for analysis

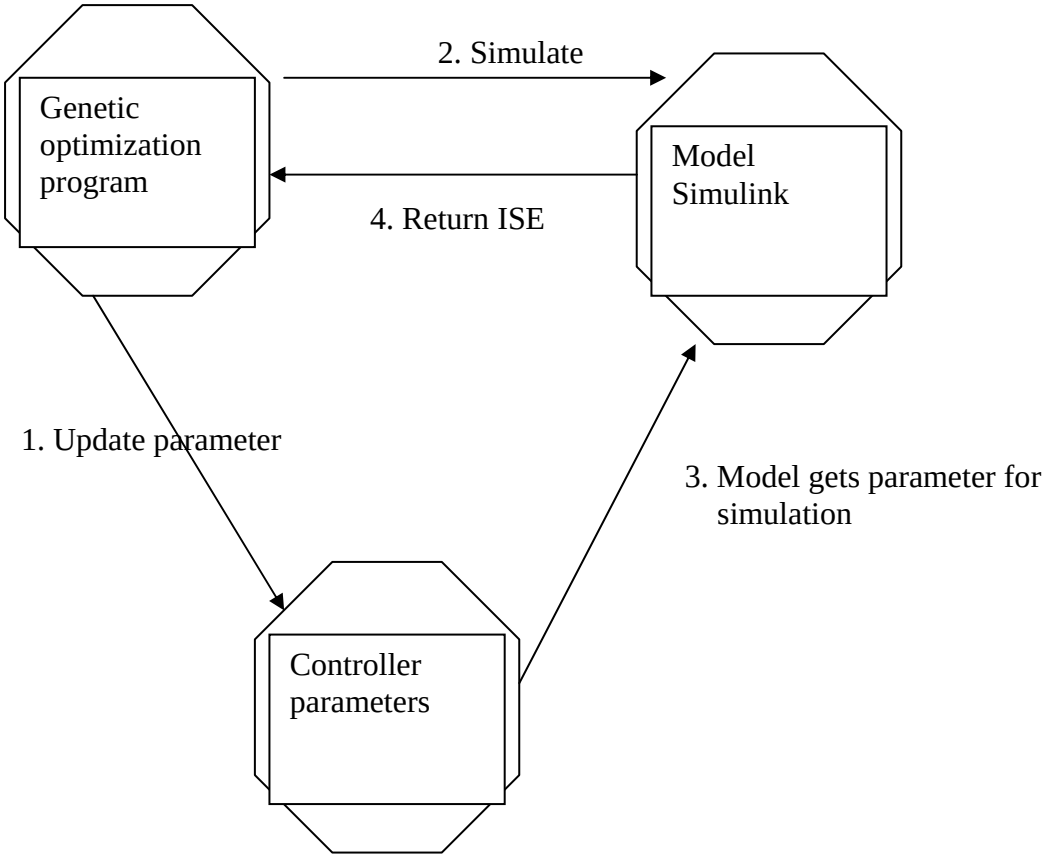
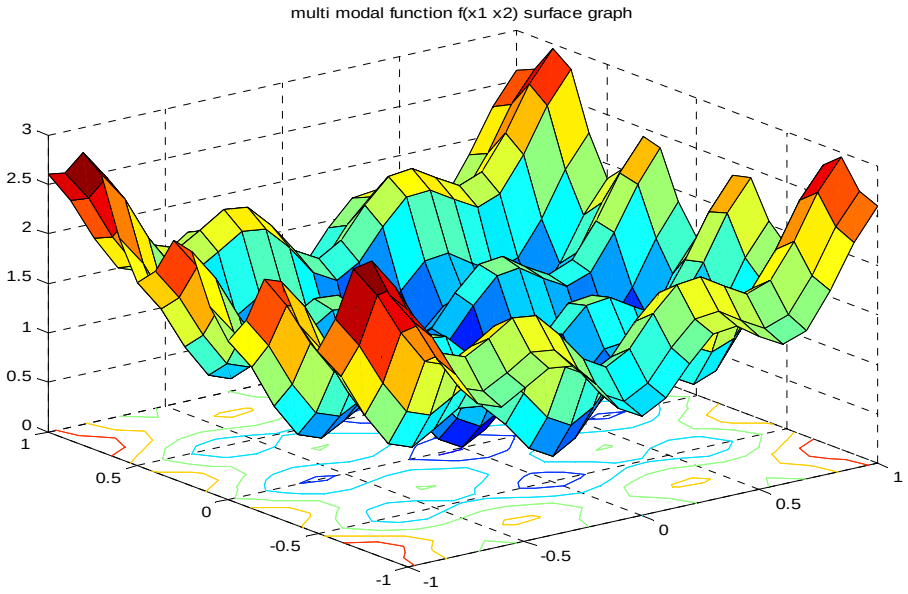


Fig 8.11 Fuzzy controller tuning diagram

8.8 RESULTS AND DISCUSSION

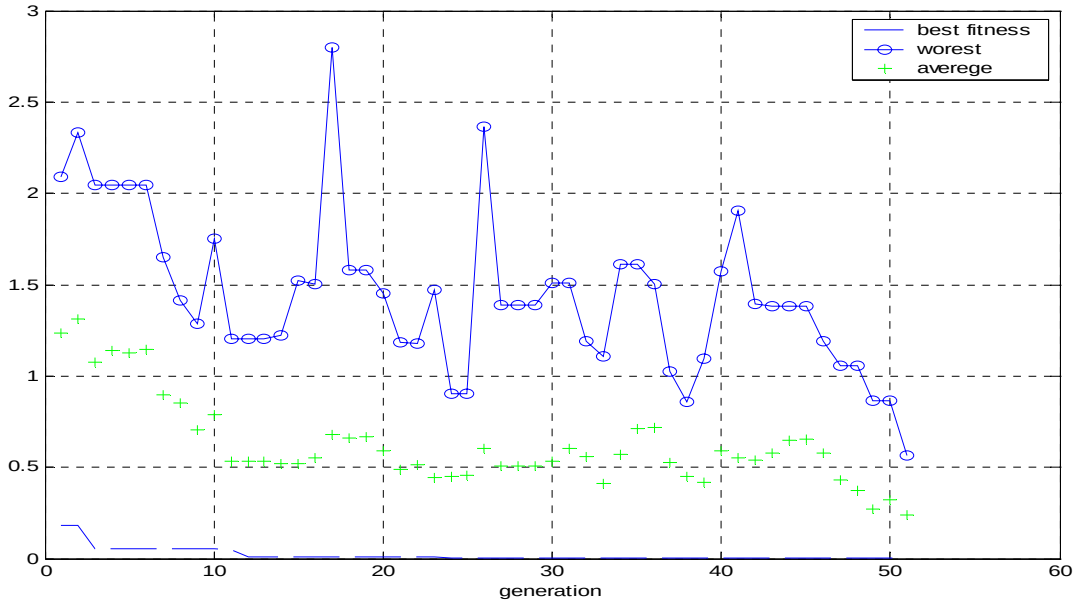
Before using GA to optimize the parameter of fuzzy controller (i.e. parameter of output membership function) for liquid level control system the efficiency of GA has been checked by using known simple function with several picks ad valleys. The program coding is found in APPENDEX B1 and as expected genetic algorithm searches the global minimum of the given multi modal function. The effect of parameters of GA algorithm also investigated so that one can select appropriate parameters while optimizing a given function at ha The result and graph of best fitness value verses generation are given below. $f(x_1,x_2)=x_1^2+x_2^2-0.3*\cos(3*pi*x_1)-0.4\cos(4*pi*x_2)+0.7$



Result of genetic optimization of the above function

s.no	pc	pm	generation	Population size	Bit length	Best fitness value	Average fitness value	Worst fitness value
1	0.8	0.01	50	10	12	0.000932	0.777718	0.865227
2	0.4	0.9	50	10	12	0.003424	1.149183	2.007528
3	0	1	50	10	12	0.623899	1.075680	1.843257
4	1	0	50	10	12	0.008560	0.059543	0.145596
5	0.8	0.01	100	10	12	0.000039	0.328916	1.301278
6	0.4	0.9	100	10	12	0.002000	1.177807	1.832853
7	0.8	0.01	50	40	12	0.001423	1.061131	1.926886

As seen from the above table the selection of GA parameter should be done carefully so that to get the desired result. For more detail refer chapter 6.



Heuristic FLC verses GA optimized FLC

Heuristic fuzzy controller can be designed by getting rules from plant operator or intuitively designing the rule base and membership function for input and output from the knowledge gained from the dynamics of the plant. However, when this heuristically designed FLC for liquid level control system put in simulation environment the parameter of controller could not give us the required performance i.e. minimum ISE. Most often the tuning of fuzzy controller has been done by trial and error, by changing rule base or the parameter of membership function or both manually. However, this method is too difficult and time consuming when the plant complexity increases.

Therefore, another method of automatic tuning of FLC should be devised and in this project GA based tuning of out put membership function parameter has been developed. As expected genetic algorithm automatically tunes the parameter of the given FLC until the desired performance index i.e. minimum ISE reached. The result of genetic tuning of FLC for liquid level control system has been shown in the following table for different run. The program code is found in APPENDIX C .

When heuristic fuzzy controller designed in section 8.1 put under simulation environment it generates big ISE, which we want to minimize by using fuzzy- genetic simulation in the next section. The ISE result of heuristic FLC is ISE=0.59

Simulated genetic optimization results

With the selected genetic operator as seen in the program in APPENDIX C, the following results are obtained after running different number of generation. As seen from the table below the minimum value of ISE was obtained after 400 generation and the corresponding optimum parameter for output membership function are tabulated below.

Table 8.6 parameter for output membership function for different number of generation

Number of generation	ISE	Parameter of corresponding output membership function				
		BN	SN	NC	SP	BP
100	3.31	-1 -1 -0.58	-0.8,-0.5,-0.43	-0.58 ,0, 0.48	-0.1,0.5 0.8	-0.4, 1,1
200	0.59	-1,-1,-0.7	-0.8,-0.5,-0.43	-0.7 ,0, 0.46	0.126,0.5,0.7	0.46,1,1
300	0.31	-1,-1,0.64	-0.74,-0.5,-0.43	-0.64 ,0, 0.38	0.13,0.5,0.78	0.38,1,1
400	0.23	-1,-1,-0.7	-0.86,-0.5,-0.34	-0.7 ,0, 0.42	0.12,0.5,0.78	0.42,1,1
500	0.29	-1,-1,-0.62	-0.9,-0.5,-0.37	-0.62,0,0.42	0.14,0.5,0.7	0.4,1,1

As seen from the above table the minimum value of ISE is obtained at 400 generation with value ISE=0.23 and the optimum output membership function is shown below

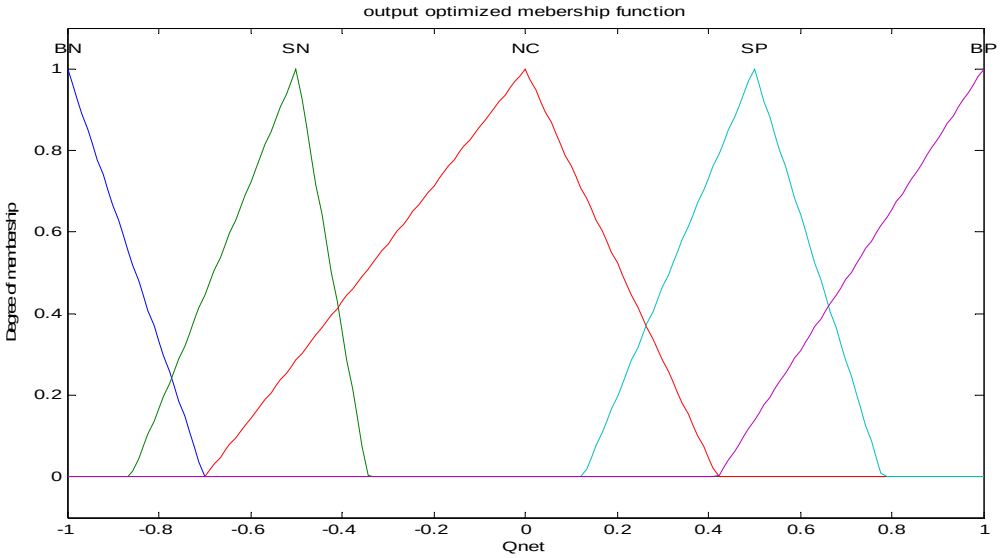


Fig 8.12optimized membership function

8.9 FURTHER WORK

In this project we have tried to show how to use genetic algorithm to tune the membership function fuzzy logic controller fro liquid level system. In order to optimally tune fuzzy controller most literatures suggest that both the rule base and membership function must be tuned. In the future one can tune the same controller rule base and membership function and compare it with the one tuned membership function parameter. In addition to this different kind of membership function such as trapezoidal, Gaussian and etc can be used together instead of single triangular type membership function. Investigate whether any of the assumptions made in designing FLCs such as using only triangular membership functions or only allowing an odd number of these sets have a significant impact on the performance obtainable from the controllers. One can also investigate more thoroughly if GAs can be applied to FLCs or other types of nonlinear controllers so that they may successfully control fully the liquid level system.

8.10 CONCLUSION

Optimization of a fuzzy logic controller can prove a lengthy process when performed heuristically. In this study it has been shown that the use of genetic algorithms offers a feasible method for the optimization of the membership function fuzzy logic controllers. Genetic Algorithms have been shown to be powerful search tools that can reduce the time and effort involved in designing systems for which no systematic design procedure exists. They can quickly find close-to-optimal solutions and if set-up well can avoid local optima. They are certainly useful tools when trying to solve analytically difficult problems. Fuzzy Logic Controllers can provide more effective control of non-linear systems than linear controllers, as there is more flexibility in designing the mapping from the input to the output space. Whereas expert knowledge is usually required to design a fuzzy controller using traditional methods, it has been shown in this report that even without using any knowledge of the system, GAs can build an effective controller relatively quickly. This technique may lead an increase in the use of FLCs as the previously time-consuming design procedure can be reduced dramatically. This study intends to establish a methodology on how to apply GA in the search of a global optimal solution of a fuzzy controller in a liquid level control system. As a result, this population-based optimization algorithm was able to converge satisfactorily to the lowest ISE of 0.23 in 400 generations. Even though the performance reported here did not arrive at a significantly minimal index, the trend of the optimization process encountered in this study showed a very promising result. To fully optimize fuzzy controller with genetic algorithm both the rule base and membership function should be optimized simultaneously. This will give better performance index. Weaknesses were also seen in GA based on the results. It sometimes terminated in an unsatisfactory step response. By mistakenly picking MATLAB functions to be used for selection, crossover, and mutation operations, this algorithm would even fail to converge.

REFERENCES

[1] John YEN and Reza Langari. Fuzzy Logic, intelligence, control and information, Pearson Edication, 1999

[2] Dinter Driankov ,Hans Hellendoorn ,Michael Reinfrank. Introduction to fuzzy control , Narosa publishing house 1996, Delhi.

[3] Passino, Kevin M. and Yurkovich, Stephen *" Fuzzy Control"* Addison-Wesley 1998

[4]I.J Nagrath and M.gopal. control system engineering 3rd edition , New age international publishing company Ltd 2003

[5] Verbruggen, H.B. and Brujin, P.M. *"Fuzzy Control and Conventional Control – What is the real contribution of Fuzzy Systems"* Fuzzy Sets and Systems 1997 (151-160)

[6] Dragan.D.K,. Kuzmanovic.S.B,. Emil.L *"Design of a PID-like compound fuzzy logic controller"* Engineering Applications of Artificial Intelligence, 14 (2001) 785-803

[7] Shi.Yuhui.,. Eberhart.Russell,. Chen.Yaobin *"Implementation of Evolutionary Fuzzy Systems"* IEEE Transactions on Fuzzy Systems, Vol. 7, No. 2, April 1999

[8] Visioli.A *"Tuning of PID controllers with fuzzy logic"* IEEE Proc.-Control Theory Appl., Vol.148, No.1, January 2001

[9] Hyun-Joon.Cho,. Kwang-Bo.Cho,. Bo-Hyeun.Wang *"Fuzzy-PID hybrid control: Automatic rule generation using genetic algorithms"* Fuzzy Sets and Systems 92 (1997) 305-316

[10] Gurocak ,H.B. *"A Genetic Algorithm method for tuning Fuzzy Logic Controllers"* Fuzzy Sets and Systems 108 (1999) 305-316

[11] Goldberg. D.E. *"Genetic Algorithms in Search, Optimisation and Machine Learning"* Addison-Wesley, 1989

[12] Whitley, Darrell *"A Genetic Algorithm Tutorial"* Statistics and Computing (4): 65-85, 1994

[13] http://www-me.mit.edu/Lectures/RLocus/ Root Locus internet-based tutorial, Massachussets Institute of Technology (M.I.T)

[14] http://www.engin.umich.edu/group/ctm/state/state.html State-Space internet-based tutorial, University of Michigan.

[15] www.cs.Berkeley.edu/~zadeh Internet home-page of Professor Lotfi Zadeh, University of California, Berkeley.

[16] Bandyopadhyay, R, Chakraborty, U.K, Patranabis,D.
"Autotuning a PID controller: A Fuzzy-Genetic Approach"
Journal of Systems Architecture 47 (2001) 663-673
[17] Karr, Charles L.
"Design of an Adaptive Fuzzy Logic Controller using a Genetic Algorithm"

[18] www.brunel.ac.uk/depts./AI/alife/ga-holla
Synopsis of John Holland's early development of Genetic Algorithms.

[19] Grefenstette ,John J. and Baker,James E.
"How Genetic Algorithms Work: A critical look at Implicit Parallelism"
Proceedings of the 3rd International Conference on Genetic Algorithms, 1989.
[20] Syswerda,Gilbe rt
"Uniform Crossover in Genetic Algorithms"
Proceedings of the 3rd International Conference on Genetic Algorithms, 1989.
[21] Hinterding, Robert, Gielewski, Harry and Peachey, T.C.
"The Nature of Mutation in Genetic Algorithms"
Proceedings of the 6th International Conference on Genetic Algorithms, 1995.
[22] Reeves, Colin R.
"Using Genetic Algorithms with Small Populations"
Proceedings of the 5th International Conference on Genetic Algorithms, 1993
[23] Chipperfield, Andrew, Fleming, Peter, Pohlheim, Harmut, Fonseca, Carlos.
"Genetic Algorithm Toolbox For use with Matlabâ v5– User Guide"
University of Sheffield, United Kingdom, 1994.
[25] Deb, Kalyanmoy and Agrawal, Samir
"Understanding Interactions Among Genetic Algorithm Parameters"
Foundations of Genetic Algorithms 5 conference, Leiden, Holland,1998.
[26] Cordon, O, Herrera, F, Hoffmann, F, Magdalena, L and Gomide, F.
"Ten Years of Genetic Fuzzy Systems: Current Framework and Trends"
[27] Odetayo, M.O. and McGregor, D.R.
"Genetic Algorithm for Inducing Control Ru les for a Dynamic System"
Proceedings of the 3rd International Conference on Genetic Algorithms, 1989.
[29] *"Fuzzy Logic Toolbox – User Guide for use with Matlabâ"*
The Mathworks Inc
[30] Goldberg.David.E
"Sizing Populations for Serial and Parallel Genetic Algorithms"
Proceedings of the 3rd International Conference on Genetic Algorithms, 1989.
[31] Schaffer.J.David., Caruana.Richard.A,. Eshelman.Larry.J,. Das.Rajarshi
"A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimisation".
Proceedings of the 3rd International Conference on Genetic Algorithms, 1989.
[33] http://www.matworks.com
[34] Jantzen.Jan
"Tuning of Fuzzy PID Controllers"
Technical University of Denmark, Department of Automation
Technical report no. 98-H 871

[35] Song Y.H.. Johns.A.T

"Applications of Fuzzy Logic in Power Systems"
Power Engineering Journal, April 1999
[36] Bramlette.M.F.
"Initialisation, Mutation and Selection Methods in Genetic Algorithms for Fucntion Optimisation"
Proceedings of the 4th International Conference on Genetic Algorithms, 1991.
[37] Cornelius T.Lenods. Volume 1,3,3 Knowledge based technique and application
[38] Robert E.king computational intelligence in control engineering
Marcel Dekker inc 1999

[39] Rnald R.Yager and Dimitar P.Filev. Essentials of fuzzy modeling and control , PHI 2001
[40] S.Rajasekaran and G.A Vijayalakshmi Pai Neural Networks, Fuzzy Logic and Genetic algorithm , PHI 2003
[41] Francisco Herrera Luis Magdalena Genetic fuzzy system tutorial
[42] Dorf, R.C. and Bishop, R.H., *Modern Control Systems*. Addison-Wesley, Menlo Park,CA,1998.
[43] Jang, J.-S. R., et. al., *Neuro-Fuzzy and Soft Computing*. Printice Hall, Upper Saddle River, NJ, 1997.

[44] Thomas BACK, Frank KURSAW evolutionary algorithms for fuzzy logic:
a brief over view
[45] Lennon, W.K. and Passino, K.M., 1997, "Strategies for Genetic Adaptive Control", *Proceedings of IEEE Conference on Decision and Control*, pp. 1908-1913.
[46] Tang, K.S, Man, K.F., Chen, G. and Kwong, S., 2001, "An Optimal Fuzzy PID Controller", *IEEE Transactions on Industrial Electronics*, **48**, No. 4, pp. 757-765
[47] Cao, Y.-D., Hu, B.-G., and Gao, D.-J., 2001, "Genetic Based Robust Optimal Design for One Input Fuzzy PID Controllers", *IEEE International Conference on Systems, Man and Cybernetics*, Volume 4, pp. 2263-2268.
[48] Bandyopadhyay, R., Chakraborty, U.K. and Patranabis, D., 2001, "Autotuning a PID Controller: A Fuzzy-Genetic Approach", *Journal of Systems Architecture*, **47**, pp. 663- 673.
[49] Mehamed Kantardzic - Data Mining concepts, Models, method and Algorithms
IEEE press, John wiley and son's publication

[50] Kalyanmoy Deb Optimization for engineering design algorithms and examples PHI,2002

[51] Robert N.Bateson introduction to control system technology 5th edition ,prentice hall

19 89

APPENDIX A

```
% simulation program for liquid level FLC
% written by Zelalem Girma
% 11/06/05
% wlc7.m
clear all;
close all;
sys=newfis('wlc7');  %  to  creat  a  new  FIS  with  file  name
"wlflc.fis" for level control

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%definition of Membership function for the input variable "error"
to the FLC controller
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sys=addvar(sys,'input','ERROR',[-1  1]);      %  range  of  input
variable "ERROR"
sys=addmf(sys,'input',1,'NB','trimf',[-1   -1   -0.4]);   %   the
left,mid, right value of mbership

%function(trianglular NB )for first input "ERROR"
sys=addmf(sys,'input',1,'NS','trimf',[-0.8 -0.4 0.2]);
sys=addmf(sys,'input',1,'PS','trimf',[-0.2 0.4 0.8]);
sys=addmf(sys,'input',1,'PB','trimf',[0.4 1 1]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%definition of Membership function for the input variable "change
in error"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
sys=addvar(sys,'input','CHANGE IN ERROR',[-1 1]);             %
range of input variable "change in error"
sys=addmf(sys,'input',2,'NB','trimf',[-1  -1  -0.5]);       %  the
left,mid, right value of mbership

%function(trianglular NB )forsecond input "change in error"
sys=addmf(sys,'input',2,'NS','trimf',[-0.8 -0.5 -0.2]);
sys=addmf(sys,'input',2,'NZ','trimf',[-0.4 0 0.4]);
sys=addmf(sys,'input',2,'PS','trimf',[0.2 0.5 0.8]);
sys=addmf(sys,'input',2,'PB','trimf',[0.5 1 1]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%definition  of  Membership  function  for  the  output  variable
"Knobsetting"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%

sys=addvar(sys,'output','Qnet',[-1 1]);
sys=addmf(sys,'output',1,'BN','trimf',[-1 -1 -0.5]);
```

```
sys=addmf(sys,'output',1,'SN','trimf',[-0.75 -0.5 -0.25]);
sys=addmf(sys,'output',1,'NC','trimf',[-0.5 0.0 0.5]);

sys=addmf(sys,'output',1,'SP','trimf',[0.25 0.5 0.75]);
sys=addmf(sys,'output',1,'BP','trimf',[0.5 1 1]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% definition of fuzzy rules
%%%%%%%%%%%%%%%%%%%%%%%%%%%
rulelist=[1 1 5 1 1;...
          1 2 5 1 1;...
          1 3 5 1 1;...
          1 4 4 1 1;...
          1 5 4 1 1;...
          2 1 5 1 1;...
          2 2 4 1 1;...
          2 3 4 1 1;...
          2 4 3 1 1;...
          2 5 2 1 1;...
          3 1 4 1 1;...
          3 2 3 1 1;...
          3 3 2 1 1;...
          3 4 2 1 1;...
          3 5 1 1 1;...
          4 1 2 1 1;...
          4 2 2 1 1;...
          4 3 1 1 1;...
          4 4 1 1 1;...
          4 5 1 1 1];


 sys=addrule(sys,rulelist);

 % plot FIS Input Output diagram
 figure(1);
 plotfis(sys);
```

APENDIX B1

```
% Note that the functions called by main program downloaded
(genetic tool box) directly from the MATWORKS web site
www.matworks.com except the last two functions
% minimize the following function by using genetic algorithm
% f(x1,x2)=x1^2+x2^2-0.3*cos(3*pi*x1)-0.4cos(4*pi*x2)+0.7
% where X1,x2element of[-1 1]
%main program
%geneticexample.m
popsize=10;
maxgen=50;
lengtha=12;
pcross=0.8;
pm=0.01;
bits=[lengtha lengtha];
vlb=[-1 -1];
vub=[1 1];
phen=init(vlb, vub,popsize,2); %find phenotype of each initial
population
                                %create random population
[gen, lchrom,coarse, nround]=encode(phen,vlb,vub,bits) %convert
                                                        %phenotype
into binery
[ fitness, object]=score(phen, popsize) % evaluation of fitness
and                                              %objective
function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%display of the contour graph of objective fuction            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x= -1:0.1:1;
y=-1:0.1:1;
[x1, y1]=meshgrid(x,y);
z=x1.^2+y1.^2-0.3*cos(3*pi*x1)-0.4*cos(4*pi*y1)+0.7;
figure(1);
contour(x,y,z);
title('\bf the contour plot of the objective function');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%store the best candidate of the initial population %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[best_obj(1),index]=min(object);
 best_gen=gen(index,:);   % returns the row of binary (gentype)
best                             %candidate
 best_phen=phen(index,   :);   %%   returns   the   row   of
decimal(phenotype)                              %best  candidate
that gives the best                                  %objective
function value(eg (x,y))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % the worst population of the initial candidate %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 [worest_obj(1), index1]=max(object);

worest_cur_gen=gen(index1);
 worest_cur_phen=phen(index1);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %calculate the average performance of the population%
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 avg_obj(1)=0;
 for k=1:popsize
    avg_obj(1)= avg_obj(1)+object(k);
 end;
 avg_obj(1)=avg_obj(1)/popsize;
 best_x(1)=best_phen(1); %store best of the first variable on
best_x(1)
 best_y(1)=best_phen(2);  %store best of the second variable on
best_y(1)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
 % display the BEST:  WOREST: AVERAGE: value of objective
     function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
 for i1=1:2
     fprintf(1,'%f' ,best_phen(i1));
 end;
 fprintf('\n');
 fprintf(1,'BEST:     %f     WORST:   %f           AVG:      %f
\n',best_obj(1),worest_obj(1),avg_obj(1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% START OF MAIN GENETIC ALGORITHM LOOP                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 for ii=1:maxgen
     newgen=reproduc(gen,fitness);  %REPRODUCTION OPERATION
     gen=mate(gen);                       %mate-Randomly reorders
(mates)
                                   %two member of  the population
                                     %
     gen=xover(gen,pcross);         %CROSSOVER operation
     gen=mutate(gen,pm)             %MUTUATION operation
     [phen  ,  coa]=decode(gen,  vlb,vub,bits);  %decode  the
                                                 %genotype(binary
string) of                          %the  new  population  into
phenotype (decimal)
     [  fitness,  object]=score(phen,  popsize)  %  evaluation  of
                                                 %fitness
```

```
[best_cur_obj,index]=min(object); %store the best candidate of
the                                  %current   population
 best_cur_gen=gen(index,:);
 best_cur_phen=phen(index, :);
 [worest_obj(ii+1), index1]=max(object);
 worest_cur_gen=gen(index1);
 worest_cur_phen=(index1);
 avg_obj(ii+1)=0;   %calculate the average performance of the
current                          population
   for k=1:popsize

avg_obj(ii+1)=avg_obj(ii+1)+object(k);
       end;
 avg_obj(ii+1)=avg_obj(ii+1)/popsize;
 if (best_cur_obj>best_obj(ii))    %apply elitist strategy
     phen(index1,:)=best_phen;
     gen(index1,:)=best_gen;
     object(index1)=best_obj(ii);
     best_obj(ii+1)=best_obj(ii);
 elseif(best_cur_obj<=best_obj(ii))
     best_phen=best_cur_phen;
     best_gen=best_cur_gen;
     best_obj(ii+1)=best_cur_obj;
 end;
best_x(ii+1)=best_phen(1);  %display  evolution  of  the  best
solution on surface countur          best_y(ii+1)=best_phen(2);
                          %graph
 surfc(x,y,z);
 hold;
line(best_x,best_y);
 for i1=1:2
     fprintf(1,'%f  ',best_phen(i1));
 end;
 fprintf(1,'---> %f\n',best_obj(ii+1));
 fprintf('\n');
 fprintf(1,'BEST:     %f      WORST:   %f            AVG:      %f
\n',best_obj(ii+1),worest_obj(ii+1), avg_obj(ii+1));
  end

 xx=1:maxgen +1 ; %display evolution of objective function for
the                   %worst , average and best solution
 figure(2);
 plot(xx,best_obj,'b--',xx,worest_obj,'o-',xx,avg_obj,'g+');
 xlabel('generation');
 legend('best fitness','worest','averege');

 grid on;
_____________________// end//_____________________________
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% init.m this function creates a random initial population
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function phen=init(vlb,vub,siz,sea) %
for ii=1:siz
    phen(ii,:)=(vub-vlb).*rand(1, sea) + vlb
end

_________________//    //_________________
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% score.m this function compute fitness and objective function values of population
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [fitness, object]=score(phen,popsize)
for ii=1:popsize
 object(ii)=phen(ii,1)^2+phen(ii,2)^2-0.3*cos(3*pi*phen(ii,1))-
0.4*cos(4*pi*phen(ii,2)) +0.7;
    fitness(ii)=1/(object(ii)+1);
end
__________________//    //_______________
```

APPENDIX B2

```
% This is the genetic algorithm program which searches global
optimum  of the given function by following flow chart of simple
genetic algorithm

% Genetic.m
function
[opt_param]=genetic(popsize,objfunc,N,minmax,nbit,pc,pm,maxg)
gc=1  % initialize generetaion counter
stop=0; %stopping flag
for ii=1:popsize
    for jj=1:N
        initialvar(ii,jj)=round(rand*(2^nbit(jj)-1));
    end;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% convert decimal values to binary bits and generate
popsizezxlchrom   %
% bitmatrix where lchrom is the length of chromosome
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

lchrom=sum(nbit)   % length of chromosome
bitmatrix=zeros(popsize,lchrom); % initialize bitmatrix zeros of
matrix popsize X lcrom
for ii=1:popsize
    k=lchrom;
    for jj=N:-1:1
        x=initialvar(ii,jj);
        for m=1:nbit(jj)
            bitmatrix(ii,k)=rem(x,2)
            x=fix(x/2);
            k=k-1;      % reduce the length of chromosome by one
        end;
    end;
end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MAIN GENETIC ALGORITHM LOOPS STARTS HERE                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    while (stop==0)
        gc

for jj=1:N
    for ii=1:popsize
    param(ii,jj)=minmax(jj,1)+(minmax(jj,2)-
minmax(jj,1))/(2^nbit(jj)-1)*initialvar(ii,jj);
    end;

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% FITNESS VALUE CALCULATION
%%%%%%%%%%%%%%%%%%%%%%%%%%

sumfitness=0;
for ii=1:popsize
    argument=param(ii,1:N);
    fitness(ii)=feval('objfun',argument);
    sumfitness=sumfitness+fitness(ii);
end;
avgfitness=sumfitness/popsize;        % calculation of average
fitness
[maxfitness,index]=max(fitness)  % parameters giving maximum
fitness given out as optimum parameter
opt_param=param(index,:) % returns the row number of optimum
parameter
fitnessratio=avgfitness/maxfitness; % fitness ratio will be used
as termination criteria

avgfit(gc)=avgfitness;  % array of average fitness in each
generation ,recorded generation-wise
maxfit(gc)=maxfitness;   %% array of maximum fitness in each
generation ,recorded generation-wise

%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
% GENERATE MATING POOL THROUGH ROULETTE WHEEL SELECTION     &
%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
sum=0;
for ii=1:popsize
    ps=fitness(ii)/sumfitness;  %computation  of  probability  of
selection of the ii-th chromosome
    sum=sum+ps;
    cumulativesum(ii)=sum;  %  cumulative  sum  of  the  selection
probabilities
end;
mplocation=randperm(popsize) % return random sequencing of the
number from 1 to population size
```

```
%*************************************************************
THE RANDOMLY SPUN ROULETTE WHEEL SELECTS JJ-TH CHROMOSOME AND
PLACES IT AT RANDOM LOCATION IN MATING POOL SIMULATED AS FOLLOWS

%*************************************************************
for ii=1:popsize
    rwspin=rand;
    for jj=1:popsize
        if(rwspin<=cumulativesum(jj))
            for k=1:lchrom
            matingpool(mplocation(ii), k)=bitmatrix(jj,k);

        end;

        break;
       else
       end;
    end;
end;


  %$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
  % CROSS OVER OPERATOR         $
  %$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  nc=round(pc*popsize/2); % number of cross over

  for ii=1:2:nc   % increment by two since crossover is pairwise
      crossite=round(rand*lchrom);
      for jj=1:crossite
          temp=matingpool(ii,jj);
          matingpool(ii,jj)=matingpool(ii+1,jj);
          mattingpool((ii+1),jj)=temp;

      end;
  end;
% for jj=1:popsize
   %for k=1:lchrom
    % matingpool(mplocation(ii), k)=bitmatrix(jj,k);
 end;
end;

   %$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
   % MUTATION OPERATOR           $
   %$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
   nm=round(pm*lchrom*popsize); % number of mutation
   while (nm>0)
       for ii=1:nm
           mutationsite=round(rand*lchrom*popsize);     %mutation
                           %being worked out not just within a
                  %particular chromosome but over the entire

            if(mutationsite==0)
                mutationsite=1; % perform at least one mutation
```

```
           end;
         %location of the bit to be mutated
         colm=rem(mutationsite,lchrom) %column number
         if(colm==0)
             clom=lchrom;
             row=fix(mutationsite/lchrom);
         else
             row=fix(mutationsite/lchrom)+1;
         end;
         if (matingpool(row,colm)==0)
            matingpool(row,colm)=1;
        else
           matingpool(row,colm)=0;
       end;
   end;
   nm=0;
end;

%DECODING
for ii=1:popsize
    b1=0;
    bn=0;
    for jj=1:N
        sum=0;
        n=nbit(jj);
        if(jj==1)
            b1=1;
        else
            n1=b1+nbit(jj-1);
        end;
        bn=bn+nbit(jj);
  % DECIMAL VALUE CALCULAION
        for k=b1:bn
            n=n-1;
            sum=sum+(matingpool(ii,k))*(2^n);
        end;
        variable(ii,jj)=sum;
    end;
end;
gc=gc+1 %increment the generation counter
initialcar=variable;
bitmatrix=matingpool;
% STOPPING CRITERIA
if( (gc>maxg) |(fitnessratio>0.9999))
    stop=1;
else
    stop=0;
end;
end;
if(gc>maxg)
    disp('GOAL NOT REACHED')
end;
generation=1:gc-1;
plot(generation,maxfit,'b',generation,avgfit,'r')
legend('maxmum fitness','average fitness')
```

APPENDIX C

```
% program which optimizes the parameter of output mebership
function by GA
%gafuz.m
echo on
options = foptions([1 1e-3]);
options(13) = 0.01;
options(14) = 400;
options(11)=20;
options(12)=1;
vlb = [-0.9 -0.7 -0.43 0.1 0.3 0.7];
vub = [-0.6 -0.4 0.0 0.3 0.6 1];
bits =4*ones(1,6);
pause % Hit any key to continue
%while(options(14)<200)
[x,stats,options,bf,fgen,igen]=genetic('defo',[],options,vlb,vub,
bits);

% A few notes:
% First notice that x is returned as a real between -1 and 1
x

pause % Hit any key to continue
% Also the initial and final generations, fgen and lgen, are
% returned as reals between 0 and 1
fgen
pause % Hit any key to continue

% The total number of times the fitness function was evaluated is
% equal to the ((number of generations)+1)*size_pop
%  re:  initial  population  also  requires  size_pop  function
evaluations
%
% In the present case, the fitness function was called
num_fit_call = (options(10)+1)*options(11)
pause % Hit any key to continue
[gen,lchrom,coarse,nround] = encode(igen,vlb,vub,bits)
[gen,lchrom,coarse,nround] = encode(fgen,vlb,vub,bits)

sys = readfis('wlc7.fis')

sys.output.mf(1).params=[-1 -1 x(2)];
sys.output.mf(2).params=[x(1) -0.5 x(3)];
sys.output.mf(3).params=[x(2) 0.0 x(5)];
sys.output.mf(4).params=[x(4) 0.5 x(6)];
sys.output.mf(5).params=[x(5) 1 1];
open_system('model')         %open simulink for simulation
pause

     plotmf(sys,'output',1)
     title('output optimized mebership function')
     sys.output.mf.params    % update parameters
```

```
%defo.m
%fitness function for genetic algorithm
function def=fitness(x)
      numerator=0.35*x(1)+x(2)*0.5 +x(3)*0.8 +x(4)*0.8
                              +x(5)*0.2+x(6)*0.75
              deno=0.35+0.5+0.8+0.8+0.2+0.75
        obj=numerator/deno;
        if deno > numerator
           def=1/(1+ obj);
       else
           def=obj;
       end
```