THE UNIVERSITY OF CALGARY

Genetic Based Auto-Design of Fuzzy Controllers

by

Farhad Ashrafzadeh

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

JUNE, 1996

0-612-20723-4

Abstract

This dissertation represents a departure from the conventional design of fuzzy controllers. Two different design approaches are proposed. One is a full-optimization for applications where high performance is crucial. The other involves an efficient design approach where fast development is of primary concern.

A genetic algorithm, as an optimization technique, is employed to automate and at the same time to optimize the fuzzy controller design process. This optimization requires a predefined performance index.

An overview of fuzzy controllers is first presented in which the novel concept of *characteristic points* is developed. This concept allows one to appreciate the role of each set of fuzzy controller parameters, and leads to the main motivation for automating the design process. An insight into the nature of the problem leads to the suitability of a genetic algorithm, as an appropriate search technique for this automation / optimization. A particular genetic algorithm is coded for the *concurrent* optimization of controller parameters. This is contrasted with the alternative approach, where controller parameters are optimized sequentially.

As an application example, electrical drive systems are considered. A novel perspective on the field oriented control of induction motors is first presented, followed by several possible designs of the fuzzy controller for such a drive system. In each case, the fuzzy controller is designed using one of the proposed genetic algorithms, and results are compared with those of a conventional counterpart.

Also in this dissertation, a novel perspective on the robustness of a fuzzy controller is presented which suggests designing a fuzzy controller based on sliding mode control – a well established robust control scheme. Based on this view, an efficient near-optimal design technique of a fuzzy controller is proposed. For instance, given a 7×7 decision table a search space of 84 dimensions collapses into a search space of 7 dimensions. While this is achieved at the expense of decreasing the performance index slightly, it can be employed for a large class of systems where fast tuning of the controller is the primary concern. Furthermore, this approach is not restricted to the genetic-based auto-design of a fuzzy controller.

Aknowledgement

I would like to thank Dr. T. H. Barton, my supervisor, for his guidance, corrections, and valuable suggestions. I respect his high expectation.

I would also like to express my deepest gratitude to Dr. E. P. Nowicki, my co-supervisor, for his constant support throughout my Ph.D. program. He gave me the most valuable gift – freedom. His encouragement allowed me to pursue my own idea, to make mistakes and to learn. To learn how to approach a problem, how to cope, and how to solve it. Yet, he raised some key questions which put me again in the right direction.

Dr. T. Chen was an invaluable resource always helping me find answers to my control questions. I also greatfully acknowledge Dr. O. P. Malik's help in my research work.

My special thanks and appreciation goes to Mr. R. B. Boozarjomehri, a friend of mine in the Department of Chemical Engineering. It was his great insight which helped me to quickly solve some challenging problems in my research work. I should also thank my friends, Mr. F. Sabouri and Mr. M. Mohammadian for their selfless help in my Ph.D. program.

Behind any valuable achievement, there exists some invisible family support. In this respect, I must thank my mother, sisters, and brother for their constant encouragement, help, and support. I am particularly very much indebted to my elder sister, Dr. Farah Ashrafzadeh. In fact, without her leadership and management my struggle with the challenging parts of my personal and academic life would have been far more difficult.

Finally I wish to express my deepest appreciation for my wife whose understanding and steadfast support helped me to reach another goal of my life.

This work is particularly dedicated to my mother, sisters, and wife.

DEDICATED WITH LOVE:

To my **wife** and **two daughters**

&

To all my **TEACHERS**

both in personal life: To my **mother, sisters**, and **brother**

and in academic life: To my teachers in primary school, high school, and university.

List of Tables

Approval page ii

Abstract iii

Aknowledgement v

Dedication vii

Table of contents viii

List of Tables xii

List of Figures xiii

1 Introduction **1**

1.1 Background 1

1.2 Introduction to electrical motor drives 2

1.3 Introduction to fuzzy control 5

1.4 Problems in the design of fuzzy controllers 6

1.5 Introduction to genetic algorithms 7

1.6 Thesis objectives 8

1.7 Thesis outline 10

2 Fuzzy controllers **12**

2.1 Introduction 12

2.2 A comparative view 13

2.3 Basic structure of fuzzy controllers 15

2.3.1 Normalization and denormalization 16

2.3.2 Fuzzification and defuzzification 16

2.3.3 Inference mechanism 16

2.4 Different types of fuzzy controllers 18

2.4.1 PD-like fuzzy controller 19

2.4.2 PI-like fuzzy controller 19

2.4.3 P-like fuzzy controller 20

2.5 Fuzzy controller characteristics 20

2.5.1 Fuzzy controller as a static transfer function 22

2.5.2 Fuzzy controller as a non-linear element 22

2.5.3 Fuzzy system as universal approximator 23

2.6 Realization of conventional fuzzy controllers 24

2.7 Design parameters of fuzzy controllers 25

2.7.1 Role of normalization and denormalization factors . . . 26

2.7.2 Role of membership functions and rules 28

2.8 Problem description 31

2.9 Chapter summary 33

3 Problem coding based on genetic algorithms 34

3.1 Introduction 34

3.2 Optimization requirements 35

3.3 Genetic algorithms; An overview 37

3.3.1 Coding 39

3.3.2 Evaluation 39

3.3.3 Reproduction 40

3.3.4 Recombination 41

3.4 Coding the design problem of a fuzzy controller 44

3.4.1 Normalization and denormalization factors 44

3.4.2 Membership functions 46

3.4.3 Decision table 48

3.4.4 Coding of the entire controller 48

3.4.5 Fitness function 51

3.5 Chapter summary 52

4 Genetic based auto-design of a fuzzy controller for induction motor drives 54

4.1 Introduction 54

4.2 Field oriented control 57

4.2.1 Traditional view 57

4.2.2 A novel view 59

4.3 Conventional fuzzy controller 63

4.4 Optimal fuzzy controller 69

4.4.1 Performance index 69

4.4.2 Sequential approach 71

4.4.3 Concurrent approach 73

4.5 Chapter summary 78

5 Discussion 79

5.1 Introduction 79

5.2 Input partitioning versus output partitioning 80

5.2.1 Overview 80

5.2.2 Input partitioning approach 83

5.2.3 Output partitioning approach 83

5.2.4 Input-output partitioning approach 84

5.3 Robustness of a fuzzy controller 86

5.3.1 Introduction 86

5.3.2 Variable structure control 86

5.3.3 Another view of the conventional fuzzy controller . . . 90

5.4 A novel approach for the efficient design of a fuzzy controller . 94

5.5 Chapter summary 100

6 Conclusions and future research 101

6.1 Conclusions 101

6.2 Recommendations for future research 104

References 105

Glossary 113

Appendices 120

A Proof of non-linearity of fuzzy controllers 120

B Mathematical model of an induction motor 124

C Simulation programs 127

List of Tables

4.1 Normalization factors found by trial-and-error 67

4.2 Case-1: Free parameters of a genetic algorithm 71

4.3 Case-1: Normalization factors 71

4.4 Case-2: Free parameters of a genetic algorithm 73

4.5 Case3: Free parameters of a genetic algorithm 76

4.6 Case3: Normalization factors found by a genetic algorithm. . . 76

5.1 Case 3: Free parameters of a genetic algorithm 99

5.2 Case 3: Normalization factors found by a genetic algorithm. . 99

List of Figures

2.1 A classical feedback control system. 14

2.2 A fuzzy control system. 15

2.3 Basic structure of a fuzzy controller. 15

2.4 Conventional membership functions;(a): gaussian (b) triangular. 17

2.5 Sliding mode decision table 18

2.6 PI-like fuzzy controller in a closed-loop control system. 21

2.7 Approximation of a typical static function by fuzzy domains. . 24

2.8 The process of defining membership functions for a typical non-linear function 24

2.9 Normalization of the universe of discourse. 27

2.10 A typical mapping of the controller. 28

2.11 Defining the membership functions. 28

2.12 Defining the framework by membership functions. 29

2.13 Approximated function by fuzzy model. 29

2.14 The effect of membership functions on mapping. 30

2.15 Fuzzy mapping in the case of more overlap. 30

3.1 fuzzy controller in closed loop system. 36

3.2 A typical string in a genetic algorithm. 39

3.3 Contribution to the mating pool based on fitness. 40

3.4 One-point crossover. 42
3.5 Two-point crossover. 42
3.6 Chromosome viewed as a loop. 43
3.7 Mutation. 43
3.8 Graphical illustration of single iteration. 44
3.9 Flowchart of a genetic algorithm. 45
3.10 Bit-string representation of normalization and denormalization factors. 46
3.11 Membership function parameters. 47
3.12 Bit-string representation of membership function parameters. . 47
3.13 Full overlap of membership functions. 47
3.14 Bit-string representation of decision table. 48
3.15 Bit-string representation of entire controller. 49
3.16 Illustration of internal and external loops. 50
3.17 Different types of information incorporated in the fuzzy controller design. 51

4.1 General structure of global input-output linearization. 59
4.2 Global input-output linearization based on a state observer. . 60
4.3 Feedforward input-output linearization. 61
4.4 Block diagram for IFOC of an induction motor. 62
4.5 Simplified control block diagram of IFOC for an induction motor. 63
4.6 Simplified IFOC block diagram with fuzzy controller 63
4.7 Gaussian membership functions 64
4.8 Sliding mode table. 65
4.9 Conventional design algorithm for a fuzzy controller. 65
4.10 A typical desired step response for a drive system. 66

4.11 Trajectory of the desired typical response in fuzzy- partitioned state space. 67
4.12 Modified control policy of conventional approach. 68
4.13 Step response of motor speed based on semi-algorithmic approach. 68
4.14 Case-1: Step response of motor speed for self-tuning approach. 72
4.15 Case-1: Speed of convergence. 72
4.16 Case-2: Step responses for different designs of decision table. . 74
4.17 Case-2: Sequential approach (a) Auto-design versus manual design (b) Speed of convergence for a genetic algorithm. 74
4.18 Case-3: The new membership functions found by a genetic algorithm. 76
4.19 Case-3: Step response of motor speed for concurrent approach. 77
4.20 Case-3: Speed of convergence. 77

5.1 A desired function for the controller. 81
5.2 Approximated function by uniformly discretizing the input space. 81
5.3 Approximated function by uniformly discretizing the output space. 82
5.4 Bit-string for input partitioning. 83
5.5 Bit-string for output partitioning. 84
5.6 Input-ouput partitioning. 85
5.7 Switching surface in two dimensional state space. 88
5.8 Variable structure of decision table 91
5.9 Two distinct structures in the decision table 91
5.10 Fuzzy sliding surface in a two dimensional state space. 92

5.11 Designing different fuzzy sliding surfaces based on normalization factors; (a) large N_e, small $N_{\dot{e}}$; (b) moderate N_e, moderate $N_{\dot{e}}$; small N_e, large $N_{\dot{e}}$ 93
5.12 Operating lines for (a) sliding mode controller with boundary layer (b) special type of fuzzy controller. 94
5.13 Basic structure of the fuzzy logic controller. 95
5.14 Control surface for the conventional fuzzy controller. 95
5.15 A non-linear control surface. 96
5.16 A new view to the sliding table. 96
5.17 A new string for an optimal design of a fuzzy controller. . . . 97
5.18 The proposed string for an optimal design of a fuzzy controller. 97
5.19 Step responses of the non-linear system for different approaches. 99

Chapter 1

Introduction

1.1 Background

Artificial Intelligence is machine emulation of the human thinking processes. The term began to be systematically used since the Dartmouth College conference in 1956 when *Artificial Intelligence* was defined as computer processes that attempt to emulate the human thought processes that are associated with activities that require the use of intelligence.

In 1854, George Boole first published his article entitled *Investigations on the laws of thought*, and as a result, Boolean algebra and set theory was born. Later, with the aid of vacuum tubes and the invention of the bipolar junction transistor, the modern era of von Neumann type digital computation arrived. Digital computers were defined by some to be *intelligent* since they were able to emulate the process of human-like *yes* and *no* logic. Certainly, by using binary logic, computers can solve some complex engineering, scientific, and other data processing problems. In one respect, this deserves applause. However, it was in the late 1960's and early 1970's, that the limitation of computers in handling algorithmic-type problems was felt. Consequently an entirely new paradigm for structuring software more like the natural human thinking process was born. These expert systems, also called ***knowledge based systems***, are responsible for the acquisition of knowledge from human experts in a particular domain and translating it into software.

It was in the mid 1960's that a new theory called ***fuzzy logic*** was proposed which gradually helped to supplement the expert systems as another branch of artificial intelligence. L.A. Zadeh [1], the originator of this theory, argued that most human thinking is fuzzy or imprecise in nature, and therefore, Boolean logic which involves distinct "0" and "1" cases cannot properly emulate the human thinking process. In recent years, fuzzy logic has emerged as an important artificial intelligence tool to characterize and control a system whose model is not known, or ill-defined. It has been widely applied in process control, estimation, identification, diagnostics, stock market prediction, agriculture, military science, etc.

While fuzzy logic has the capability to (partly) model human knowledge, it cannot replace the human expert. In fact, the human mind has the capability to learn new things and to modify its previous knowledge to achieve better results. This immediately leads to the question: is it feasible to add a learning feature to the existing fuzzy systems such that these can generate new knowledge, learn through experience, or modify their initial knowledge to achieve higher performance?

This dissertation, is a small, but significant effort to answer the above question in the affirmative.

1.2 Introduction to electrical motor drives

It is estimated that electrical motors absorb over 60% of the electrical energy generated in North America [2]. Apart from this fact,75% of all electrical motor drive applications require either a variable speed or variable torque [3]. As the speed of an electrical machine increases, its demand for electrical power increases as well[1]. Thus a great amount of energy can be saved if electrical motors are freed from a constant speed constraint and operate at speeds dictated by the load requirement. These aspects indicate how important variable speed drives are in modern manufacturing or industrial processes from both the standpoint of practical requirements and energy savings.

Modern electrical drives are a challenging and sophisticated technology involving major disciplines in electrical engineering such as modern control theory, electrical machine theory, power electronics, signal processing, and microcomputers. The major requirements of electrical motor drives, in general, include high dynamic performance, i.e. fast response without overshoot, zero steady state error, reliability, low maintenances, and robust performance in the presence of disturbances. While some of these requirements, mostly on the control side, can be fulfilled by dc drives, the others such as cost, reliability, and low maintenance can not be met by these types of drives.

In contrast to dc motors, induction motors have the advantages of being extremely low maintenance, low cost, robust, reliable machines which have a high power to weight ratio. The control of these motors, however, is very challenging because induction motors are high order, multi-variable, non-linear, and uncertain systems[2]. Furthermore, the desirable variables such as torque and speed of the machine are not linearly dependent on accessi-

[1]If the mechanical load torque is proportional to the square of its speed, then the power absorbed by the machine would be in proportion to the cube of motor speed. In the case of constant load torque, the input power is proportionally related to the motor speed.

[2]Since the parameters of an induction motor are temperature dependent and furthermore its mechanical load is, in general, unknown, these systems are categorized as uncertain systems.

ble variables such as voltages and currents. In fact, there is a link between flux and torque that prevents fast response of the machine to a step change in torque commands. Moreover, the speed control of induction motors requires change in the frequency and phase angle of the terminal quantities in a complex manner that calls for sophisticated control schemes and costly implementation.

This has been the case until two major technological changes took place: advances in solid state switching devices resulted in the availability and decreasing cost of variable frequency power converters, and simultaneous enhancements in digital control made feasible the implementation of sophisticated non-linear control techniques. At this juncture, it is important to note that the increased concern about the present and future cost and availability of the electrical energy has accelerated such techno-economical justification.

It was such an evolution that made possible the real implementation of field oriented control, as the most sophisticated torque control method for induction motor drives. This approach which was invented in the early 1970's [4], was ignored for a long period of time because of difficulties in the hardware implementation, due to the lack of advanced microprocessor technology. It also required instantaneous values of some state variables which were neither reliably measurable nor feasible to be estimated quickly and precisely.

Today, field oriented control has been widely accepted as by far the most popular type of torque control for induction motors [3]. The new trends in this field now involve the application of modern non-linear control techniques to further enhance the performance of such controllers as well as optimizing drive operation based on a specific requirement.

1.3 Introduction to fuzzy control

Fuzzy logic[3], as one of the principal elements of artificial intelligence, is playing a key role in dealing with uncertainty and imprecise information. As stated earlier, originally, the main motivation behind fuzzy logic was the provision of a framework to (partly) represent human knowledge in which imprecision is a common feature. To perform such a task, it should be able to model variables in classes such as large, low, high, etc, as is often done by a human. If one takes a closer look at such variables, it turns out that they have more or less a domain nature rather than a point nature. This shows that in order to mathematically represent such variables, the concept of a *set* should be used, and in one respect, the term *fuzzy set* [1] has been chosen to this end.

Apart from defining variable classes, a particular logic is required for processing such variables. Such a logic is called fuzzy logic which can be viewed as a superset of two-value (i.e. Boolean) logic and even multi-value logic. It is in this sense that fuzzy logic mimics the crucial ability of the human mind to summarize data and focus on decision relevant information. In fact, the key elements in human thinking are not numbers but some fuzzy sets, that is, classes of objects in which the transition from member to non-member is gradual rather than abrupt which is the case in crisp or Boolean sets.

If such notions, i.e. fuzzy sets and fuzzy logic, give the capability to model human knowledge, then the knowledge of an expert or engineer can also be represented in the same manner. For this reason, fuzzy logic provides a framework to incorporate any knowledge including the intuition and experience of an engineer designer. It is in this way that fuzzy logic found its

[3]There is a glossary of fuzzy logic and genetic algorithm terms following the references.

applications in control and fuzzy control[4], as a process control algorithm.

As mentioned earlier, the essence of fuzzy logic is domain-wise mapping. This implies that the exact model of a controlled system is not required. Furthermore, fuzzy logic also facilitates handling systems having non-linearity, parameter variations, and perturbations. This capability of fuzzy control, on the one hand, and its feature of low cost hardware implementation, on the other, has made fuzzy control very successful in embedded control. The application of fuzzy control in electrical drives is quite new [5,6]. Since power electronic systems often do not have an exact mathematical model, and they are often non-linear with parameter variations, the fuzzy controller has a significant potential to enhance their performance [2,5,7–16].

1.4 Problems in the design of fuzzy controllers

In general, the design of a fuzzy controller consists of five different stages. These stages are normalization, fuzzification, the execution of rules, defuzzification, and denormalization. Since the exact relation between the system dynamic performance and the controller parameters is not known, no systematic approach exists to nicely design a fuzzy controller for a specific application. For this reason, the design process of fuzzy controllers at some point becomes a trial-and-error approach [2,6,13,16–20]. This equivalently means that the development of a fuzzy controller turns out to be completely based on designer intuition and experience. Such a trial-and-error approach requires a large number of repetitions, and it is therefore, time consuming and tedious. Furthermore, as the number of input/output signals of the controller increases, it tends to be more difficult, if not impossible, to end up with an acceptable solution. Moreover, there are some cases where expert

[4]In this dissertation, the term fuzzy control is used in place of the more common *fuzzy logic control*, to emphasize the control aspects of fuzzy set theory.

knowledge is not available or the required knowledge about the system dynamics is beyond the expert experience. Obviously in these cases, even the first steps in the design of the fuzzy controller cannot be taken. Apart from all these aspects, even in the case that a fuzzy controller can be designed by such a trial-and-error approach, there is no guarantee that the resultant controller will be an optimal one. Notwithstanding the success of fuzzy control, these limitations have impeded the application of fuzzy controllers to a wider range of control problems [7,21–27].

1.5 Introduction to genetic algorithms

A genetic algorithm is a probabilistic optimization approach inspired by biological evolution in nature [28]. In common with other optimization techniques, a genetic algorithm performs a search in a multi-dimensional space in which a hyperspace is defined by an objective function. In general, genetic algorithms have proven to be effective at solving a variety of complex problems that other techniques have difficulty in solving. For instance. since genetic algorithms do not rely on computing local derivatives to guide the direction of investigation in search space, they can handle problems with discontinuous and non-differentiable hyperspace. Furthermore, genetic algorithms particularly are successful at finding the optimum where the hyperspace is non-linear, or highly convoluted with many local optima. In fact, in gradient based techniques, a point-wise search is performed by which a single point of search space is selected, tested, and used with some decision rules to conduct the search process. These methods may fail in a multi-modal situation by convergence to one local optima.

In contrast, in a genetic algorithm, a population of points is chosen simultaneously to be independently processed and this gives a better picture of the entire search space. There is, hence, more probability of finding the global optimum.

The basis of a genetic algorithm is that a population of solutions is first randomly produced. The size of the population is a free parameter, which trades off coverage of the search space against the required time to compute every iteration, a so-called generation. Each solution in a population is coded as a binary string, normally called an individual. Individuals are then evaluated based on an objective function provided by the application and a value, known as the *fitness* value, is assigned to each of them. The individuals in the current generation are next processed by performing genetic operations such as reproduction, crossover, and mutation. Reproduction involves selecting two individuals as parents based on their fitness; the higher the fitness of the individuals, the more likely they can reproduce. After selecting pairs of parents, a crossover is performed for each pair of parents, in which strings are chosen randomly and are cut at a random point to produce two *heads* and two *tails*. Then one of these segments, say tails, are swapped between two individuals and in this way a new individual is generated. Furthermore, as each bit is copied from one parent to offspring, it has a probability of being flipped. Such flipping in one or more bits is called a mutation. A mutation can be viewed as a reinjection of information that may have been lost in previous generations. It can also be seen as an investigation in other parts of search space enabling the optimizer to locate the global optimum.

1.6 Thesis objectives

The research underlying this dissertation involves the development of a novel synthesis methodology to automate and at the same time, to optimize the performance of fuzzy controllers based on a predefined objective function for any particular application. It also aims, in particular, to design an optimal fuzzy controller for induction motor drives with indirect field oriented control.

The proposed novel synthesis methodology, when encoded as a computer program, provides a convenient design approach which is directly related to the desired system requirements and avoids the difficulties involved in the conventional trial-and-error techniques. Also a novel overview of fuzzy controllers with emphasis on underlying control concepts is presented which indicates fuzzy controllers can be viewed as a non-linear static transfer function.

The technique proposed in this dissertation is based on a particular genetic algorithm. In the literature, some optimization approaches, mostly gradient based, have been used to optimize one set of fuzzy controller parameters or at best to optimize membership functions and rules in a sequential manner. What they have not taken into account, however, is the fact that there exists an interaction among different sets of parameters.

The primary, and original, contribution of this dissertation is *concurrent* optimization of fuzzy controllers by which the entire set of control parameters, i.e. normalization factors, membership functions, and rules, are processed simultaneously and therefore, the effect of their interdependencies is inherent in the optimization process. The approach proposed in this dissertation, not only is able to develop a new fuzzy controller from scratch, but it also is able to enhance the performance of an existing fuzzy controller.

Although in applying the proposed auto-design approach for the field oriented control of an induction motor, it should be noted that this approach is very general, and can be applied to a wide range of non-liner systems where fuzzy control is used.

1.7 Thesis outline

To be able to propose a good solution to a given problem, a deep understanding of its underlying concepts is first required. It is in this sense that the novel overview to fuzzy controllers in Chapter 2 starts with some fundamental concepts such as static functionality and non-linearity. By illustrating the impact of each parameter in a fuzzy controller, the main insight into the design of a fuzzy controller is gained. Optimization requirements are discussed in the first Section of Chapter 3. Next, after a brief introduction to genetic algorithms as an optimization technique, the coding procedure is presented.

An induction motor drive with field oriented control is chosen as one application for the proposed technique. In fact, a novel view to the field oriented control technique is presented in Chapter 4 followed by the design of an optimal fuzzy controller for such a drive.

Since the genetic-based auto-design of fuzzy controllers is a multi-faceted issue, different aspects of this technique are discussed in Chapter 5. Concurrent design of fuzzy controllers is first compared with a sequential approach and then extended to an input and/or output partitioning approach. The concurrent auto-design of fuzzy controllers by a genetic algorithm is the principal contribution of this dissertation.

Robustness is another important aspect of modern control theory. If a higher level of robustness can be achieved, the controller can perform better and longer without retuning. While, in the literature, it has been claimed that fuzzy controllers are highly robust, not every conventional design of such controllers should be considered as a robust controller. Special considerations at the design stage of a fuzzy controller should be taken into account to achieve a robust controller. This issue is addressed in Chapter 5, where the design of a fuzzy controller is presented from the perspective of sliding mode control, as a special case of the variable structure control technique.

Conclusions, and recommendations for future work are given in Chapter 6.

Chapter 2

Fuzzy controllers

2.1 Introduction

Fuzzy control has found many applications in the past decades. This is so, largely because fuzzy control has a capability to deal with non-linear, uncertain systems even if no mathematical model is available for the controlled system. One of the most significant features of a fuzzy system is that, in principle, any continuous non-linear function can be approximated by such a system to any degree of precision. In spite of such features, there are a few bottle-necks hindering industry from broader exploitation of fuzzy control. In the first place, a systematic design approach for fuzzy controllers is not available [7, 21–26]. This means that if a reliable expert knowledge is not available or if the controlled system is too complex to derive its appropriate control rules, development of a fuzzy controller becomes time consuming and tedious and sometimes impossible. Even in the case that expert knowledge is available, fine tuning of the controller is not a trivial task. Furthermore, a near-optimal fuzzy controller is very difficult to obtain by human trial-and-error.

Some efforts have been made to solve these problems and simplify the task of parameter tuning and rule development for a fuzzy controller [23–25, 29–35]. These approaches mainly use adaptation or learning techniques drawn from artificial intelligence or neural network theories [25, 35–46].

In this chapter a novel overview of the fuzzy controller is discussed. First, a comparative perspective of the fuzzy control approach is presented. The structure of a fuzzy controller is then outlined and this is followed by examples of different types of fuzzy controllers. The characteristics of a fuzzy controller are then explained. The realization of a conventional fuzzy controller and its design parameters are discussed in the following sections. Finally after clarifying the relative importance of controller design parameters, the main motivation for the research underlying this dissertation is addressed as a problem description. Many of the notions stated in this chapter are new and are not discussed in the current literature with the exceptions of the concepts of the fuzzy controller structure, in Section 2.3, and the concept of a universal approximator, in Section 2.5.3. The purpose of this chapter is to provide the required background on the design aspects of a fuzzy controller, and to provide some motivation leading to the optimal design of fuzzy controllers.

2.2 A comparative view

To illustrate the difference between a classical controller and a fuzzy controller (FC), consider a non-linear dynamic system

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \; , \quad \mathbf{y} = g(\mathbf{x}, \mathbf{u}) \tag{2.1}$$

where $\mathbf{x} \in R^n$ is the state vector, $\mathbf{u} \in R^r$, $\mathbf{y} \in R^m$ are the system input and output vectors respectively, and r , m , n are integers respectively. The mappings of $f(.) \in R^n$, $g(.) \in R^m$ are smooth and satisfy the conditions of

$f : R^n \times R^r \to R^n$ and $g : R^n \times R^r \to R^m$. The main objective of a control designer is to find an appropriate control algorithm, using the feedback information extracted either from state variables or system outputs, to force the system output to follow prescribed trajectories as closely as possible. The tracking error can be defined as the difference between $\mathbf{x}_d$ and $\mathbf{x}$ (if state feedback is used) or between $\mathbf{y}_d$ and $\mathbf{y}$ (if output feedback is used), where $\mathbf{x}_d$ and $\mathbf{y}_d$ stand for the desired trajectories. The differential equation 2.1 can then be equivalently expressed from the mathematical viewpoint as:

> Find an appropriate mapping from the error domain to the system input domain such that the solution of the differential error system is stable.

To this end, a classical controller performs the desired mapping in a point-wise manner such as $\mathbf{u} = h(\mathbf{e})$, as shown in Fig. 2.1.

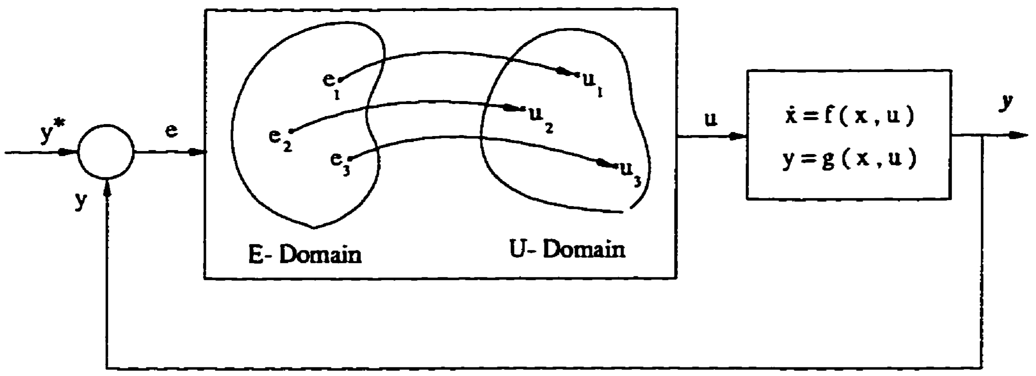


Figure 2.1: A classical feedback control system.

In contrast, in a fuzzy controller, the same mapping is performed in a domain-wise manner as shown in Fig. 2.2. This domain-wise mapping is called inference. To employ such a mapping, two interfaces are required: first transferring the *crisp* values into some domain values (encoding) and second, transferring the domain values into *crisp* values (decoding). The former is called *fuzzification* and the latter is known as *defuzzification*. This suggests three different stages within a fuzzy controller, as opposed to one

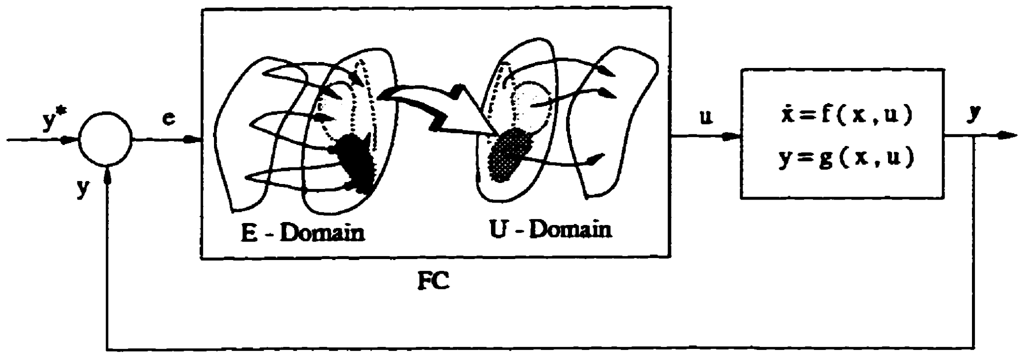


Figure 2.2: A fuzzy control system.

stage $h(e)$, in the classical controller. Further, to simplify the design process, the input-output signals may be normalized and denormalized. Then, a fuzzy controller can be viewed as a five-step structure which is discussed in greater detail in the following section.

2.3 Basic structure of fuzzy controllers

The principal structure of a fuzzy controller, as illustrated in Fig. 2.3, consists of normalization factors, fuzzification of inputs, inference or rule firing, defuzzification of outputs, and denormalization.

2.3.1 Normalization and denormalization

To design a fuzzy controller independent of the variables' physical domains, the membership functions are defined within $[-1, +1]$. This requires normalization of physical variables. Similarly, in the denormalization stage the normalized output value is mapped into the physical domain. Although these mappings are linear, they become very crucial to the performance of the controller regardless of the manner in which they are implemented, i.e. explicitly or implicitly.

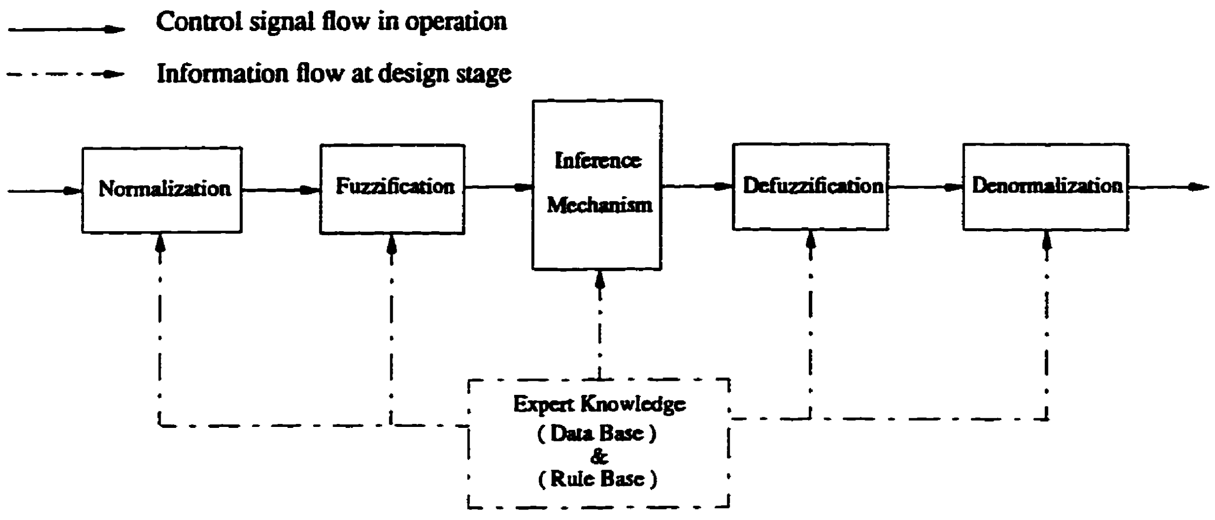


Figure 2.3: Basic structure of a fuzzy controller.

2.3.2 Fuzzification and defuzzification

Since fuzzy inferencing is performed on fuzzy values, the point-wise input values (crisp values) must be converted into fuzzy values (fuzzy sets). This is the purpose of fuzzification. In effect, in the fuzzification process, the input space is partitioned into sub-domains. Proper partitioning of this space requires some information about the system output state variables which is a part of the data base (or expert knowledge) required to design a fuzzy controller. Fig. 2.4 demonstrates two conventional types of membership functions, where the input space is partitioned into seven different fuzzy subsets in this illustration. Also, since the actuator input needs a crisp value as a control action, the output of the fuzzy inference which is again a fuzzy set, is translated into a point-wise value. This process is called defuzzification.

2.3.3 Inference mechanism

If X_t and $\dot{X}_t$ are the fuzzified controller inputs (e.g. error and error derivative) and U_t is the fuzzy value for the controller output, and R_f is the fuzzy function responsible for the mapping from the input space into the output

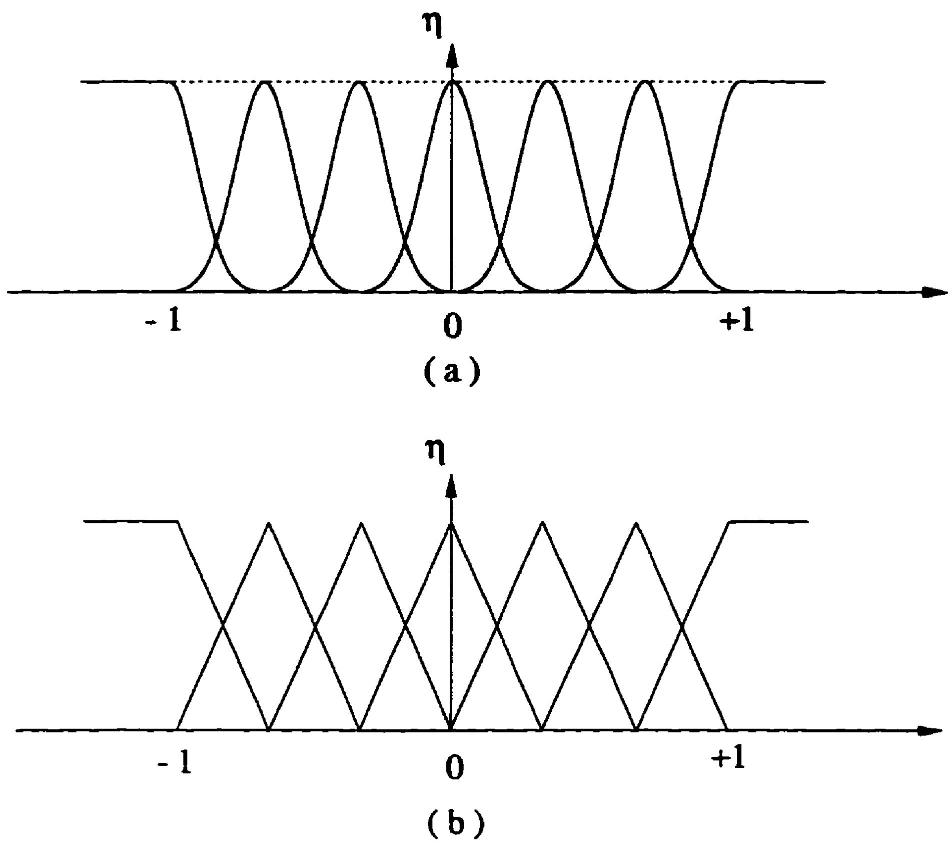


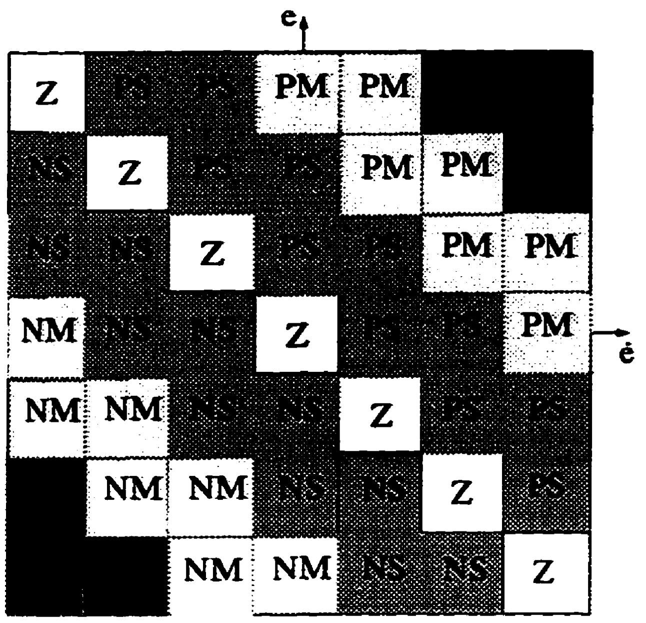
Figure 2.4: Conventional membership functions;(a): gaussian (b) triangular.

space, then the fuzzy controller can be represented [47] by

$$U_t = (X_t \times \dot{X}_t) \circ R_f \tag{2.2}$$

where $\times$ is the Cartesian product operator and $\circ$ denotes the inference mechanism (for this dissertation, the max-min operation). In the case of the rule base, if the input and output spaces are partitioned into an odd number[1] of fuzzy sets for every variable (i.e. for error, e, error derivative, $\dot{e}$, and fuzzy controller output, $\dot{u}$), the control policy can be expressed in the form of a look-up table, which is also called a ***decision table.*** Figure 2.5 shows an ex-

[1]Typically five or seven fuzzy sets are used for such partitioning.



Z	PS	PS	PM	PM	■	■
NS	Z	PS	PS	PM	PM	■
NS	NS	Z	PS	PS	PM	PM
NM	NS	NS	Z	PS	PS	PM
NM	NM	NS	NS	Z	PS	PS
■	NM	NM	NS	NS	Z	PS
■	■	NM	NM	NS	NS	Z

Figure 2.5: Sliding mode decision table

ample of a decision table for a controller of two inputs, i.e. error and error derivative denoted by e, and $\dot{e}$, and one output, $\dot{u}$. In this figure, different fuzzy sets of the controller's output, $\dot{u}$ are denoted by NB, NM, NS, Z, PS, PM, PB, which stand for negative big, negative medium, negative small, zero, positive small, positive medium, and positive big, respectively. Such a table, in the author's view, can be called a sliding mode decision table or, in short, a *sliding mode table*. Further details about this notion are presented in Chapter 5.

2.4 Different types of fuzzy controllers

In one respect, in connection with the classical control theory, four different types of fuzzy controllers can be distinguished. From this viewpoint, fuzzy controllers can be classified into PD-like, PI-like, P-like, and PID-like controllers as follows:

2.4.1 PD-like fuzzy controller

A conventional PD[2] controller can be described by

$$u = K_P \,.\, e + K_D \,.\, \dot{e}, \tag{2.3}$$

where e and $\dot{e}$ are error and error derivative and K_P and K_D are the proportional and the differential gain coefficients. The same equation in the context of fuzzy logic can be represented in a symbolic fashion as

$$\textbf{If } e(k) \text{ is } LV_e \textbf{ and } \Delta e(k) \text{ is } LV_{\dot{e}}\,, \textbf{ then } u(k) \text{ is } LV_u$$

where LV refers to a linguistic variable (e.g. positive medium), and LV_e, $LV_{\dot{e}}$, and LV_u are specified membership functions for e, $\dot{e}$, and u, respectively, and k is a sampling instant.

This symbolic representation of an equation is called a *fuzzy rule* and a set of rules can emulate the complete dynamics of a differential equation.

2.4.2 PI-like fuzzy controller

A conventional PI[3] controller can be described by

$$u = K_P \,.\, e + K_i \,.\, \int e\,dt \tag{2.4}$$

where K_P and K_i are the proportional and the integral gain coefficients. If the above integral equation is converted into a differential equation by taking the derivative with respect to time, the equivalent equation will be:

$$\dot{u} = K_P \,.\, \dot{e} + K_i \,.\, e \tag{2.5}$$

[2]Proportional plus differential
[3]Proportional plus integral

The PI-like fuzzy controller can then be modeled by following rule:

If $e(k)$ is LV_e **and** $\Delta e(k)$ is $LV_{\dot{e}}$, **then** $\Delta u(k)$ is $LV_{\dot{u}}$

In this case, the controller gives the incremental value of the output and an integrator, therefore, is required outside of the fuzzy controller to generate the final value of the control action, u.

2.4.3 P-like fuzzy controller

The fuzzy rule representing the proportional controller equation

$$u = K_P \cdot e \tag{2.6}$$

can be expressed in symbolic form as

If $e(k)$ is LV_e, **then** $u(k)$ is LV_u

It should be mentioned here that due to the requirement of a four dimensional decision table for a PID-like controller[4], it is hardly used, if at all. Furthermore, Since the PI-like fuzzy controller is easier to develop and has the property of zero steady state error, this type of controller has been chosen for this dissertation. Therefore, henceforth, whenever a fuzzy controller is referred to, the PI-like controller is meant unless stated otherwise. Fig. 2.6 demonstrates such a controller for which a typical decision table has already been shown in Fig. 2.5.

2.5 Fuzzy controller characteristics

To appreciate the essence of fuzzy controllers, these systems can be viewed from different perspectives. Following this line, one might examine this con-

[4]Proportional plus Integral plus Differential controller

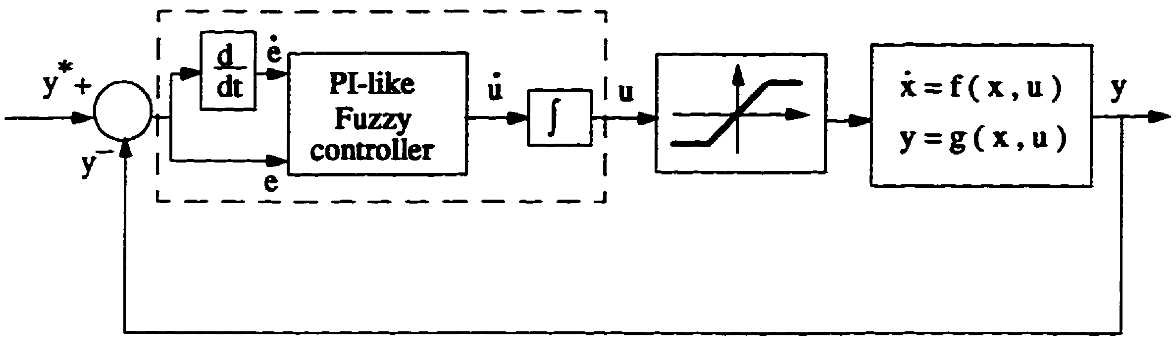


Figure 2.6: PI-like fuzzy controller in a closed-loop control system.

troller from the "functional view". If this view is employed, the fuzzy controller with the structure shown in Fig. 2.6, will appear to have a static transfer function. This automatically calls for the requirement of adding some dynamic elements in the front-end and output of the fuzzy controller. In this way, the "overall controller" would appear to have a dynamic transfer function while the fuzzy controller part is still static in nature.

It is also worthwhile to look at the controller from both the linear and the non-linear viewpoints. As shown below, fuzzy controllers perform a non-linear mapping from input to output. Furthermore, fuzzy systems, in general, and fuzzy controllers, in particular, are universal approximators. This, in turn, means the fuzzy controller of Fig. 2.6 can approximate any non-linear static transfer function to any degree of precision that is desired[5]. This makes fuzzy controllers potentially quite attractive in the control of a large class of non-linear systems. These are the issues that are discussed in greater detail in the following section. Some other features of fuzzy controllers, such as its variable structure nature, sliding mode nature, and so on, will be discussed in Chapter 5 , in which the design problems of fuzzy controllers are investigated in different respects.

[5]Provided that there is no restriction on the number of membership functions.

2.5.1 Fuzzy controller as a static transfer function

A fuzzy controller can be seen as an input-output mapping operator. One may ask whether the fuzzy controller shown in Fig. 2.6 has a static transfer function or a dynamic one (We have clamied on the previous page that it is in fact static). To answer such a question, the structure of a typical rule for a PI-like controller, is reconsidered here:

$$\textbf{If } e(k) \text{ is } LV_e \textbf{ and } \Delta e(k) \text{ is } LV_{\dot{e}}\,, \quad \textbf{then } \Delta u(k) \text{ is } LV_{\dot{u}}$$

As this rule implies, since the value of the controller output at the instant k does not depend on its previous value, no dynamics is involved inside the fuzzy controller. This makes the fuzzy controller of Fig. 2.6 with the rule structure stated in Section 2.4.2 appear as a controller with a static transfer function. This notion can also be derived from the fact that all operators with respect to time, like derivative and integration are performed outside the fuzzy controller. The dynamic behavior of the overall controller, therefore, comes about by prefiltering, i.e. derivation, and postfiltering, i.e. integration, of the input/output signals. This makes the entire controller from $e \rightarrow u$, a controller with a dynamic transfer function while the mapping $(e, \dot{e}) \rightarrow \dot{u}$ remains static.

2.5.2 Fuzzy controller as a non-linear element

A fuzzy logic controller, in general, has a non-linear transfer function. In fact, this is one of the features that has made this controller very attractive for non-linear control applications. For instance, the rule stated in Section 2.4.1 is a non-linear PD-operation and a collection of such rules can be used, and, in fact, results in the modelling of a non-linear differential equation. Such a rule-based modelling of a non-linear differential equation we would call a *qualitative differential equation.* While the source of non-linearity, on the one hand, potentially can come from rules, on the other hand, the fuzzy operators involved in fuzzification, inference, defuzzification, are non-linear in nature (refer to Appendix A for a proof). However, the point that should be clarified is that while the membership functions can introduce non-linearity, in the author's view, the main non-linearity of the system's behavior must be defined by rules (i.e. the interference mechanism). This point becomes more clear when the roles of membership functions and rules are examined in Section 2.7.2.

2.5.3 Fuzzy system as universal approximator

It has been proved that fuzzy systems are *universal approximators* [40]. As mentioned in the preceding sections, a fuzzy controller can be viewed as a non-linear mapping of

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{2.7}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output vectors respectively. Keeping this in mind, the universal approximation theorem can be stated as

Universal approximation theorem:

For any given real continuous function $F(x)$ on a set of $U \in R^n$, there exists a fuzzy system f such that

$$\sup_{\mathbf{x}\in U} |F(\mathbf{x}) - \mathbf{f}(\mathbf{x})| < \epsilon$$

where $F(x)$ is the function to be approximated and ϵ is a positive number which can be set to any arbitrary small value [42].

It is quite clear that by increasing the number of membership functions (and consequently rules), a better approximation of the original function can be attained. This is similar to the approximation of a continuous function by a number of points and some interpolation methods; the greater the number of points, the better the approximation of the function.

Figure 2.7 demonstrates a pictorial view of how a non-linear function can be approximated by a fuzzy system. By specifying such domains, the

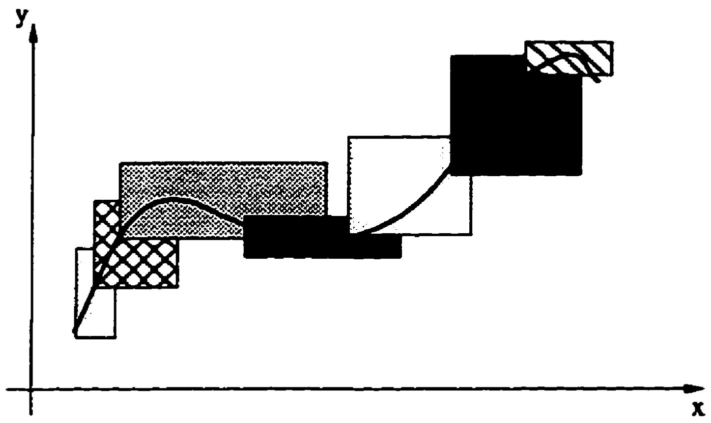


Figure 2.7: Approximation of a typical static function by fuzzy domains.

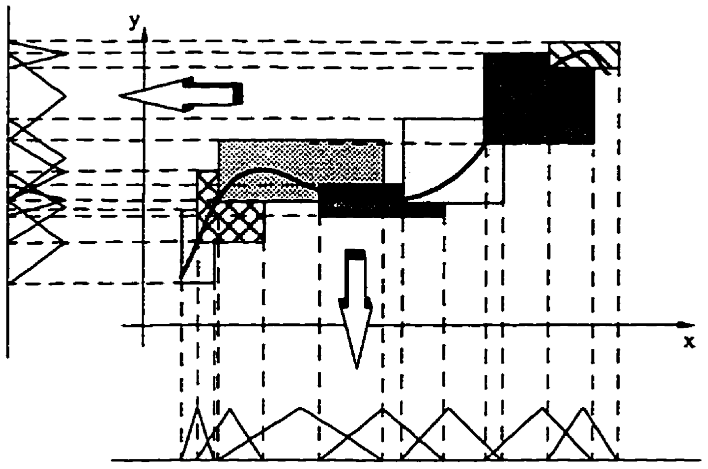


Figure 2.8: The process of defining membership functions for a typical non-linear function

membership functions of input/output spaces, in effect, can be defined as shown in Fig. 2.8 from which, in the author's view, the design process of a fuzzy system can be started and followed by the definition of control rules. This procedure will be discussed in greater detail in section 2.7.2.

2.6 Realization of conventional fuzzy controllers

In the design of a fuzzy controller, one must first decide about the number of inputs and outputs for the controller. Then the input and output spaces must be partitioned by a proper number of fuzzy sets with suitable shapes and overlaps. In the next stage, by convention, a prototype decision table should be constructed based on experience, expert knowledge, or intuition. At this point, the controller is incorporated into the feedback loop of the system and tuning is performed based on trial-and-error, by changing normalization, denormalization, membership function parameters, and consequent parts of fuzzy rules which are, in fact, the cells of the decision table.

To have an idea of the number of design parameters, consider a system with n inputs and one output, and m membership functions for each variable. Such a system requires $3mn+m^n$ design parameters where $3mn$ is the number of membership function parameters and m^n is the number of possible rules. If the number of normalization and denormalization factors are also taken into account, then the overall number of design variables would be equal to $m^n+3mn+n+1 = m^n+n(3m+1)+1$. For instance, for the controller shown in Fig. 2.6, which has two inputs and one output, if seven membership functions are chosen for each variable, the number of design parameters becomes $m^n + n(3m+1)+1 = 7^2+2(3 \times 7+1)+1 = 94$. It is now quite obvious how difficult, if not impossible, it would be to tune such a number of parameters by a trial-and-error approach if no systematic method exists. To clarify the effect of different design variables on controller performance, the role of each design parameter is first presented.

2.7 Design parameters of fuzzy controllers

As stated in Section 2.3, there are five different computational steps in the operation of fuzzy controllers. The parameters involved in these steps, can be viewed as *design parameters* at the design stage. In what follows, the role of each design parameter is explained.

2.7.1 Role of normalization and denormalization factors

As mentioned in the Section 2.3.1, the design of a fuzzy controller on a fixed domain, i.e. independent from the physical domain, requires proper normalization and denormalization. If $\mathbf{e} = \mathbf{x_d} - \mathbf{x}$ represents the actual value of the error vector, where $\mathbf{x}$ is the state vector and $\mathbf{x_d}$ is the desired value of the state vector, then the normalized error vector, $\mathbf{e_N}$, is derived by

$$\mathbf{e_N} = \mathbf{N_e}.\mathbf{e} \tag{2.8}$$

where

$$\mathbf{N_e} = \begin{pmatrix} N_{e_1} & 0 & \cdots & 0 \\ 0 & N_{e_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & N_{e_k} \end{pmatrix} \tag{2.9}$$

N_{e_i} are real numbers and the normalized domain for $\mathbf{e}$ is, say $[-a, a]$, in our case $[-1, 1]$. In particular, where $\mathbf{e} = (e_1, e_2) = (e, \dot{e})$, the mappings from e to e_N and from $\dot{e}$ to $\dot{e}_N$ are performed by

$$e_N = N_e.e \qquad \dot{e}_N = N_{\dot{e}}.\dot{e}. \tag{2.10}$$

The normalization and denormalization operations, as their names imply, are linear operations. Any linear adjustment of the controller function, therefore, can be performed by changing these factors. They have one resemblance with classical linear control, in which they emulate the proportional and the integral coefficients, i.e. K_p and K_i respectively. Thus while simple in operation, they have a crucial effect on the stability of the overall system. Moreover, this similarity between K_p, K_i and N_e, $N_{\dot{e}}$, shows that the other parts of a fuzzy controller such as fuzzification, inference, defuzzification introduce extra degrees of freedom in tuning and adding non-linearity to the controller. This also implies that the conventional PI controller is, in effect, one special case of a fuzzy controller [21]. Fig. 2.9 demonstrates how

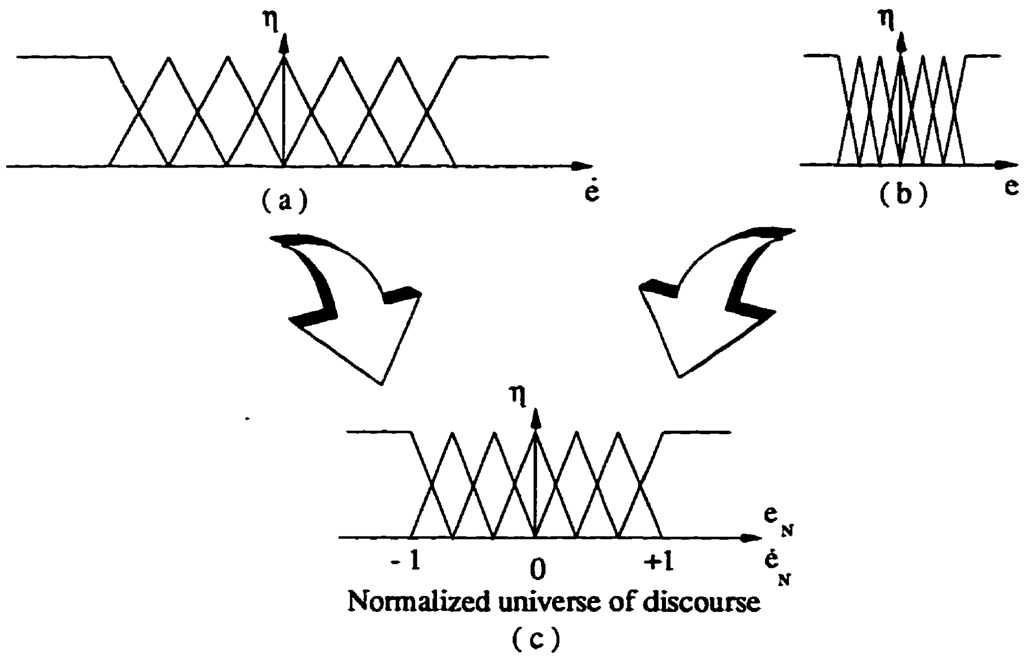


Figure 2.9: Normalization of the universe of discourse.

the universe of discourses (domain of membership functions) are changed by the normalization factors. In effect, rather than defining membership functions on different universe of discourses, one can define such functions on a fixed domain, say $[-1, 1]$, and convert the physical values to the normalized values by proper normalization factors. This makes the design of the fuzzy controller independent from the physical domains. Note that the membership functions have only a local effect on the input domains, whereas the normalization factors have a global effect, as implied by equation 2.10.

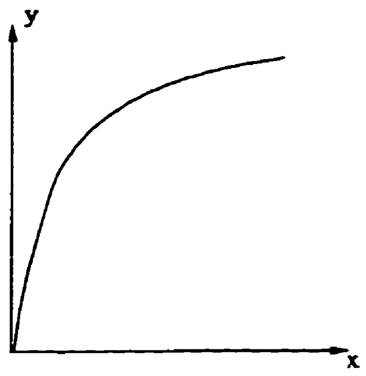


Figure 2.10: A typical mapping of the controller.

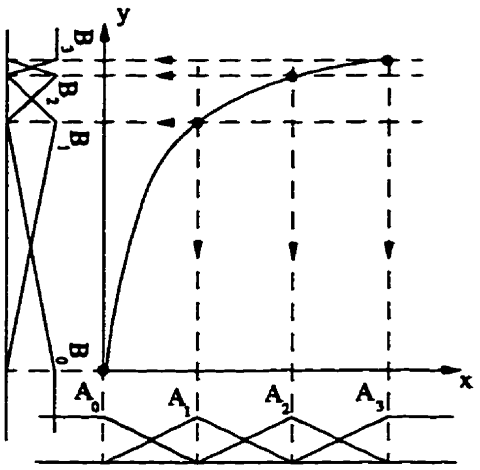


Figure 2.11: Defining the membership functions.

2.7.2 Role of membership functions and rules

To gain an insight into the relative importance of membership functions and rules, an example is presented. Assume that the functional mapping of the system is already known as depicted in Fig. 2.10, and our objective is to approximate this function by a fuzzy model. This assumption, in practice, is generally not valid as the rough behavior of the system may only be known, and the exact behavior of the system, more often, is not known. Nonetheless, at present, we make this assumption, in order to appreciate the role of membership functions and rules.

To design a fuzzy controller to perform the required mapping, the first step after normalization is define the membership functions for the input-output spaces, i.e. x, y. Assume the input-output membership functions are derived as based on a selection of four arbitrary points on a curve as shown in Fig. 2.11. The arrows in this figure, are an indication of the design order in the sense that one can define the proper membership functions of such a mapping by initially selecting four arbitral points on the function. This immediately leads to the partitioning of the $x-y$ space. The resultant parti-

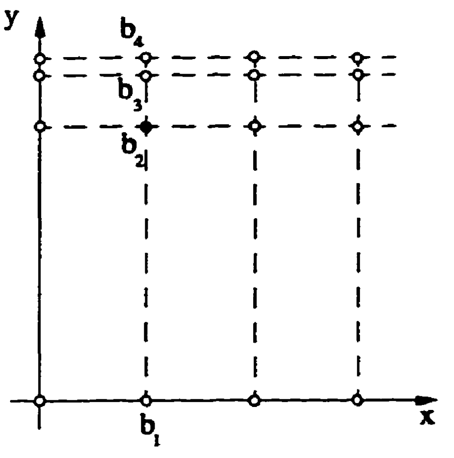


Figure 2.12: Defining the framework by membership functions.

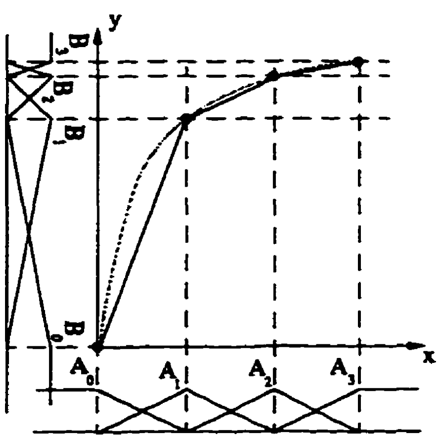


Figure 2.13: Approximated function by fuzzy model.

tioning can be viewed as defining a framework on this plane which introduces some intersection points such as b_1 and b_2 and so on as illustrated in Fig. 2.12.

The next step is to define the required rules for this mapping in the partitioned space. It is apparent that for such an ascending function, the rules can be readily stated as

If x belongs to domain A_0, **then** y belongs to domain B_0
If x belongs to domain A_1, **then** y belongs to domain B_1
If x belongs to domain A_2, **then** y belongs to domain B_2
If x belongs to domain A_3, **then** y belongs to domain B_3

As demonstrated in Fig. 2.12, the task of rules is then to select only one point, say b_1, out of the possible points, i.e. b_1, b_2, b_3, b_4. These selected points, in our view, can be called *characteristic points* because they characterize the approximated function of the controller. It is now the defuzzification operator which is responsible for interpolating between adjacent characteristic points[6]

[6]It should be clarified that the interpolation is not due to defuzzification, it is, in effect,

to construct a new function which is, in effect, an approximation of the original one (see Fig. 2.13).

Viewed from this perspective, the role of the membership functions and the rules can now be induced as follows: While the rules are responsible for general shaping of the function (i.e. locally ascending, or descending), the membership functions appear to specify the slope of each ascending or descending part. Although the influence of the membership functions can be seen more locally, with the same of set of rules, they can dramatically change the functionality of the system. For instance, consider the same membership

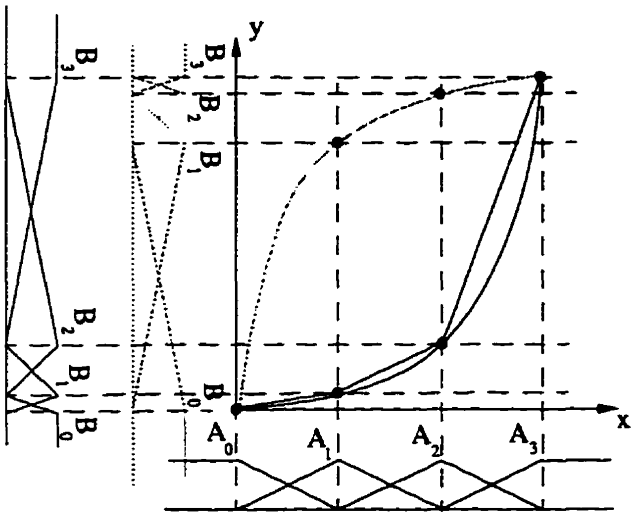


Figure 2.14: The effect of membership functions on mapping.

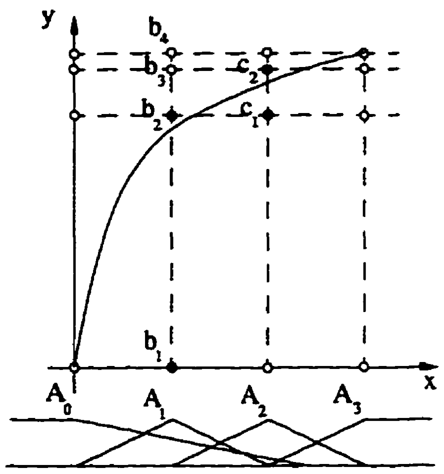


Figure 2.15: Fuzzy mapping in the case of more overlap.

functions as above for x, but a different set for the output y. With the same mapping (i.e. identical rules), this time, a new function will result which is quite different from the original one (see Fig. 2.14, where previous output membership functions are indicated by dotted lines).

This illustrates the role of membership functions in fuzzy modelling.

as a result of the overlapped membership functions together with fuzzy inference and defuzzification; but the point is that the final interpolation is performed at the defuzzification stage.

While it seems that it is the membership functions which confine the possible location for the characteristic points in each domain, the entire vertical axis can be scanned if more than two membership functions are overlapped by one membership function. This point is illustrated in Fig. 2.15 where any point on the vertical line b_1b_4 can be selected, rather than being restricted to the prescribed junctions such as b_1, b_2, b_3, b_4. In effect, more characteristic points are incorporated in the interpolation between, say c_1 and c_2, when other membership functions have greater overlap with corresponding membership functions. We will later take advantage of this fact in the solution that we will propose for the design of fuzzy controllers.

2.8 Problem description

As stated earlier, one of the most important concepts in fuzzy systems is the universal approximation theorem. This theorem also provides an explanation for the practical success of fuzzy systems in control engineering. Nonetheless, the theorem has a remarkable drawback. It is just an *existence theorem* which implies that there exists a fuzzy logic system for this non-linear function but it does not indicate how to find it. In practice, the design process of fuzzy controllers has evolved as a trial-and-error approach. Briefly, possible problems with the human trial-and-error approach can be categorized as:

- The controlled system is too complex such that its proper decision rules cannot easily be derived by human expertise.
- Designing and tuning a multi-input multi-output fuzzy controller is so tedious as to be unfeasible.
- Reliable expert knowledge is not available.
- Even with expert knowledge, fine tuning is not a trivial task.

- Some significant operating changes (in disturbances or parameters) might be outside the expert's experience.
- There is no guarantee of achieving the optimal fuzzy controller just by relying on intuition, experience, or expert knowledge.
- Since the resulting fuzzy controller is not optimal, a performance comparison of the fuzzy controller with other controllers is not valid.

To solve all the foregoing problems, some insight into the problem is required. Current fuzzy controllers perform the role of human-like controllers in the closed loop system. If, somehow, the process of knowledge development of a human can be modeled and simulated, the foregoing problems will be resolved to the extent that expert knowledge will no longer be required (i.e. we wish to automate the design process).

The approach proposed in this dissertation is not only to develop the required fuzzy system, more importantly, it is also to find and shape the best (near-optimum) non-linear function based on a prespecified performance index. While the technique is algorithmic and not based on any intuition, it is also able to incorporate heuristic knowledge in the design process.

To implement this auto-designed, auto-tuned controller, the optimization technique must be able to address the optimization problems evolved in a fuzzy controller. A fuzzy controller, in structure, is a non-linear, multi-parameter element which does not have, in general, a mathematical model in the conventional sense. Its model stems from fuzzy sets, and fuzzy rules which makes it difficult, if not impossible, to optimize by conventional techniques. On the other hand, having a large number of design variables, different in nature but interdependent, makes a very complex and unpredictable *search hyper surface* for such optimization. The degree of complexity increases when it is found that the search hyper surface has a multi-modal nature. Therefore, a calculus-based optimization technique fails in finding the global optima if it starts from any other hill rather than the highest one on the search surface. Moreover, since the search space is not well-known, existing non-differentiality or discontinuity, make it difficult to handle this problem by gradient methods. Furthermore, as this study shows, the optimization problem of a fuzzy controller is not a sequential one but rather a concurrent optimization problem. This implies that the optimization technique would preferably have a parallel nature to handle such a problem.

The genetic algorithm, as an optimization technique, has the capability to deal with a non-linear, multi-parameter, and multi-model objective function. These are the features that make this optimization approach well suited with the design process of fuzzy controllers and this is where our discussion will next turn.

2.9 Chapter summary

In this chapter, a novel overview for fuzzy control has been presented. The design structure of fuzzy controllers as well as different types of these controllers have been explained. The essential features of a fuzzy controller such as the static transfer function and non-linear approximation have been described in the way that, in the author's view, is not available in the current literature. The conventional realization of fuzzy controllers which is based on the human trial-and-error approach has briefly been discussed. The problems associated with the conventional design approach of fuzzy controllers have been addressed. The role of controller parameters has then been presented providing the background for this dissertation toward the goal of fuzzy controller auto-design.

Chapter 3

Problem coding based on genetic algorithms

3.1 Introduction

In Chapter 2, the problems associated with the design of fuzzy controllers are discussed. This chapter starts with the optimization requirements for fuzzy controller design. As will be seen, the objective function for such an optimization problem, not only is lacking a conventional analytical expression, but also, is highly multi-parameter, non-linear, and multi-modal in nature. These are the features that make the genetic algorithm a good candidate for such an optimization problem.

Genetic algorithms are search techniques based on biological evolution in nature. They were first introduced, by John Holland, his colleagues, and his students at the university of Michigan in 1975 [28]. Since then, the approach has led to some significant discoveries in both natural and artificial system science. The application of the genetic algorithm, however, for optimization in engineering is quite new [48–51]. To implement such a technique, however, a proper representation of possible solutions must first be developed. Then, by starting with an initial random population of possible solutions, employing a type of *survival of the fittest*, and exploiting old knowledge in the mating pool, the ability of each new generation to solve the problem should improve. This is achieved through the three-step processes involving evaluation, reproduction, and recombination [52]. Every individual refers to a special point in search space. This feature, which is lacking in gradient-based approaches, enhances the ability of a genetic algorithm to find the global optimum in the case where the search space has a multi-modal nature.

To design a fuzzy controller, any feasible structure for the controller as a set of parameters, should be translated into a bit-string which can easily be processed by a genetic algorithm. At the same time, a fitness function should also be defined to let the genetic algorithm evaluate possible solutions and to direct them to evolve to near-optimal ones. These are the points that will be discussed in greater detail in the last part of this chapter.

3.2 Optimization requirements

To select a particular optimization technique for the design. i.e. parameter selection, of a fuzzy controller, some insight into the problem is required. In the first place, a performance index for the control system should be defined which can be employed as the objective function in the optimization. This performance measure can be some explicit or implicit function of error for a particular transient response. For instance, for the system shown in Fig. 3.1, the performance index may be defined as

$$J = \int_{t_0}^{t_f} \psi(e)\,dt = \int_{t_0}^{t_f} \psi(f(\mathbf{x}\,,u\,,t))\,dt = \int_{t_0}^{t_f} \psi(f(\mathbf{x}\,,h(e\,,\dot{e})\,,t))\,dt \quad (3.1)$$

where ψ is some function of error, f is the system function, h is the controller function, and $\mathbf{x}$ is the system state vector. The goal of the required

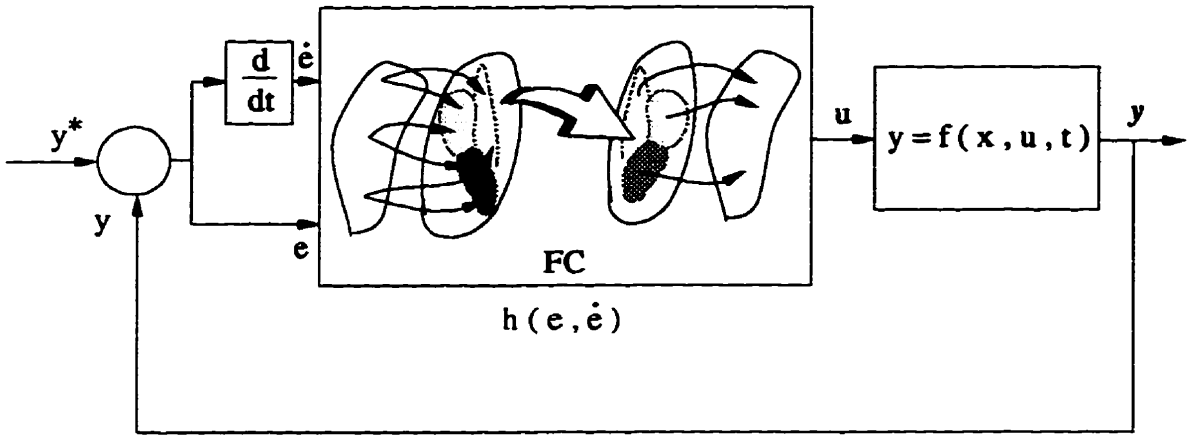


Figure 3.1: fuzzy controller in closed loop system.

optimization is to find a set of unknown parameters of $h(e, \dot{e})$ such that the objective function is minimized. In gradient based optimization techniques, in general, the partial derivative of the objective function with respect to different design parameters of the function $h(e, \dot{e})$ should be calculated and then a set of coupled non-linear differential equations is solved for the unknown parameters. Since in our case, the analytical expression of fuzzy controller in the conventional sense, i.e. $h(e, \dot{e})$, is not known, the gradient techniques cannot be employed for a problem with this nature. Furthermore, there are some other aspects that should be taken into account. First, the mapping function of fuzzy controllers, i.e. $h(e, \dot{e})$, is a non-linear function (cf. section 2.5.2). This makes the objective function non-linear even if ψ and especially the system, i.e. $f(\mathbf{x}, u, t)$ is linear. As a result, the optimization technique should be able to handle non-linearity. Second, as stated earlier, the number of parameters in the design of fuzzy controllers is very large. The objective function, therefore, appears to be a multi-parameter function by which a multi-dimensional space is introduced and more importantly, it can be shown that such a hyper surface has a multi-modal[1] nature. Viewed in this perspective, the search for a global optimum in such a multi-modal hyper surface is not a trivial task.

Genetic algorithms, in contrast with gradient based techniques, do not require an analytical expression for the controller function, i.e. $h(e, \dot{e})$, and furthermore, they are also able to handle non-linearity and the multi-modal nature of the objective function. They work with bit-strings not derivatives, and this makes them, in general, more efficient[2] particularly in the cases where the objective function is highly non-linear and multi-parameter. Moreover, genetic algorithms look at a population of points rather than to one point which is the case in the gradient based techniques. This provides a better picture of the entire search space which consequently leads to a higher probability of finding the global optimum as opposed to a local one. Gradient based techniques are point-wise search approaches and therefore, there always exists a chance of being trapped in a local optima if the hyper surface has a multi-modal nature and the starting point is not sufficiently close to the global optimum.

3.3 Genetic algorithms; An overview

Genetic algorithms are search algorithms which are based on the genetic processes of biological evolution. They are adaptive methods which may be used to solve search and optimization problems. They work with a popula-

[1] i.e., in addition to a global optimum, there are some local optima as well. This can also be visualized as a multi-hill hyper surface in search space.

[2] An optimization technique is efficient if it has the following two properties; i) a faster rate of convergence to the optimal point, and ii) a small number of calculations within one design iteration.

tion of ***individuals***, each representing a possible solution to a given problem. Each individual is assigned a fitness score according to how well it solves the given problem. For instance, the fitness score might be the strength/weight ratio for a given design or a performance index for a closed loop control system. In nature, this is equivalent to assessing how effective an organism is at competing for resources and also attracting mates. The highly adapted individuals, also called ***fit individuals***, will have relatively large numbers of offspring. Poorly performing individuals will produce few or even no offspring at all. The combination of good characteristics from different ancestors can sometimes produce ***superfit*** offspring, whose fitness is greater than that of an earlier parent. In this way, species evolve to become more and more well suited to their environment.

A new population of possible solutions is thus produced by selecting the best individuals from the current generation and mating them to produce a new set of individuals. In essence, by mixing and exchanging components of better individuals over many generations, good characteristics are spread out through the population. With the view of an optimization perspective, by randomly generating the initial population, a broad area of search space is investigated, and then by mating the more fit individuals, the most promising area of this space is explored and it is in this sense that genetic algorithms may more likely come up with the global optimum. If a genetic algorithm is designed well, the population will then converge to a near-optimal solution.

By its probabilistic nature, genetic algorithms are not guaranteed to always find the specific global optimal solution, but generally, they can find a very near-optimal solution very effectively.

Before a genetic algorithm can be implemented as a software program, a suitable coding (representation) must be made. Also required is a fitness function which is used to assign a figure of merit to each coded solution.

On the other hand, during program execution, parents must be selected for reproduction, and recombined to generate offspring. In what follows, these aspects are explained in greater detail.

3.3.1 Coding

To translate a problem into a suitable form for a genetic algorithm, a potential solution should be represented as a set of parameters (for instance the dimensions of beams in a bridge design, or the different control rules in a decision table in a fuzzy controller). These parameters are then linked in a string, most often in a bit-string. Such parameters are referred to as *genes* and the resultant string is called a *chromosome*. For instance, if our problem is to minimize a function of three variables such as $F(x, y, z)$, each variable should then be represented with a string of, say 4 bits[3]. Clearly, the resulting chromosome would consist of 12 binary digits as shown in Fig. 3.2.

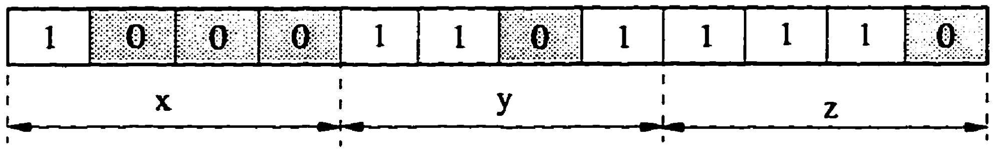


Figure 3.2: A typical string in a genetic algorithm.

3.3.2 Evaluation

The first step in every iteration of a genetic algorithm is to determine how well each chromosome can solve the problem. This step which is called evaluation is the only one in which the interpretation of the chromosome is used. The result of this evaluation, which is called a *fitness* value, is used in the next

[3]A more practical resolution for each variable might be 10 bits, giving an accuracy of one part per thousand.

step to specify how many offspring should be generated by any particular chromosome.

3.3.3 Reproduction

In this step, a new population is created based on the evaluation of the current one. For every chromosome in the current population, a number of exact copies are generated with the best chromosomes producing the most copies. As a result, good individuals might be selected several times while poor ones may not be chosen at all. Figure 3.3 illustrates this point. In

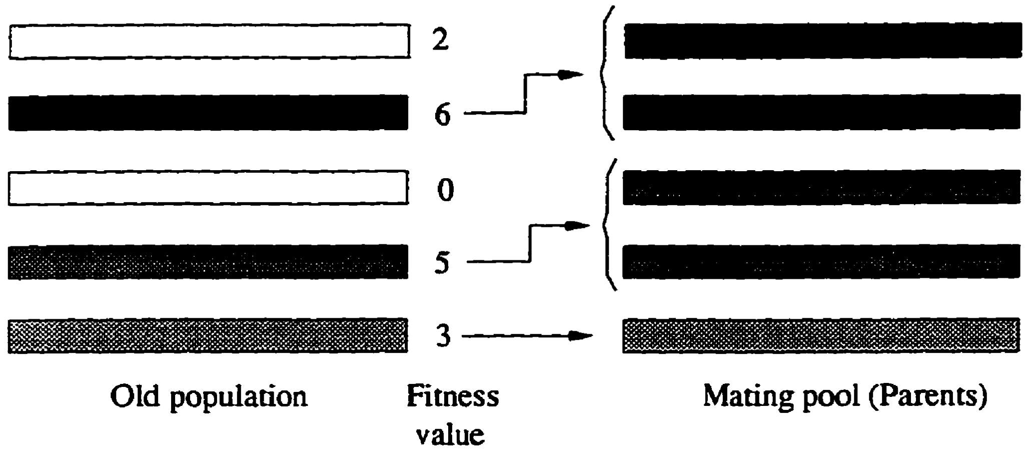


Figure 3.3: Contribution to the mating pool based on fitness.

this way, a genetic algorithm takes from biological evolution the strategy of *survival of the fittest.* There are several ways to calculate the number of offspring that each chromosome can be allocated. The two most popular methods are referred to as the *ratio technique* and the *rank scheme* [53]. In the ratio technique, each individual is reproduced in proportion to its fitness. For instance, an individual whose fitness is ten times better than another will produce ten times the number of offspring. The nice point about this method is that if a good individual emerges soon, it can guide the population quickly. The shortcoming of this approach is that if a good individual, but not the best one, arises then the population may converge prematurely on a possibly suboptimal solution.

In the ranking technique, the population is ranked and the number of offspring that each chromosome generates will depend upon its rank. For instance, the top 20% of the offspring generate two offspring, the bottom 20% offspring generate no offspring, and the rest generate only one offspring. By using this technique, no chromosome can dominate the population in only one generation. In fact, no matter how close the actual fitness values are, there is always constant pressure to improve. On the other hand, this leads to a slow convergence once a superfit individual is present and is not able to guide the population to the solution as quickly as is possible in the ratio technique.

3.3.4 Recombination

The previous step, reproduction, creats a population whose members are currently best fitted to solve the problem. However, many of the chromosomes are identical and none are different from the previous generation. Therefore, it is now necessary to generate new individuals such that they have a higher performance index. This process is referred to as recombination[4]. To do that, some *genetic operations* should be used. Among the most common are:

One-point crossover: In this case, two individuals are selected and their strings are cut at some randomly chosen position[5]. This provides two *head* segments and two *tail* segments. The heads (or tails) of these two chromosomes are then swapped to produce two new offspring. In this way, each

[4]It should be noted that in some literature, the reproduction stage is referred as the selection stage and the recombination stage is known as the reproduction stage. In this work, by recombination, we mean how to choose parents from the existing population and the essence of recombination is how to generate offspring.

[5]The cut must not take place within a parameter bit field.

offspring inherits some information from each parent. Since there is just one break point, the procedure is called ***one-point crossover***. Figure 3.4 illustrates the foregoing process by which information between two individuals is exchanged. Crossover is not usually applied to all individuals from the mating pool. A random choice is made for choosing two mates where the likelihood of crossover is more than 60% [53].

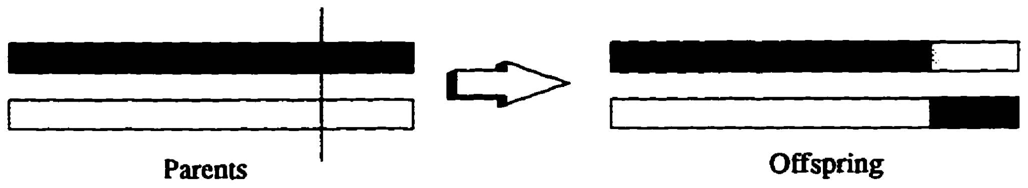


Figure 3.4: One-point crossover.

Two-point crossover: Based on this operation, two strings are split into three parts by two cut-points and the middle part is then swapped (cf. Fig. 3.5). The two-point crossover can be thought of as a one-point

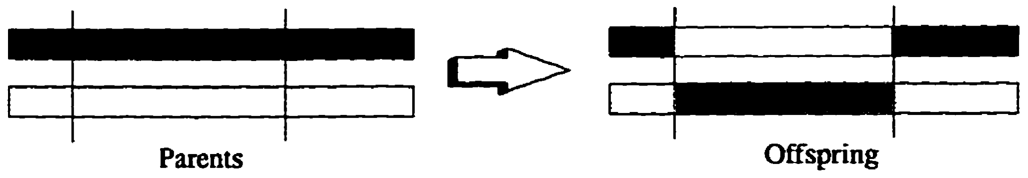


Figure 3.5: Two-point crossover.

crossover if the chromosome is viewed as a loop by joining its ends together and one cut point is assumed as the start of the string (see Fig. 3.6).

Mutation: Mutation is applied to each individual independently after crossover. It randomly changes one gene with a small probability. Figure 3.7 illustrates this point where mutation occurs in bit number four of the binary string shown in Fig. 3.2 on page 39. Although the probability of mutation is very small, it is very crucial to the success of the genetic algorithm. In effect, it explores the undiscovered part of the search space for finding the

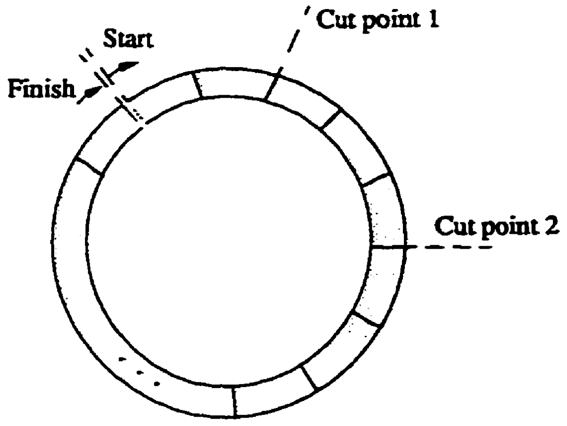


Figure 3.6: Chromosome viewed as a loop.

global optimum.

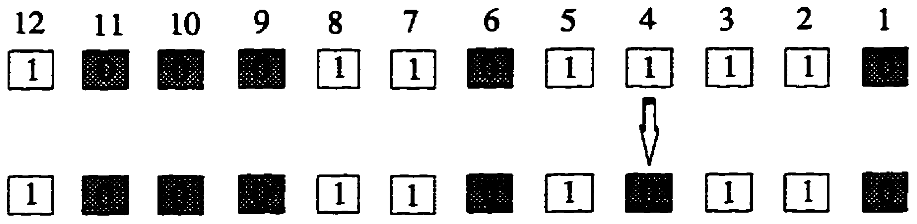


Figure 3.7: Mutation.

To visualize a single iteration, the so called generation, of the genetic algorithm technique, a population of nine elements is shown in Fig. 3.8. Every element represents an individual where the intensity is proportional to level of adaptation. First, reproduction is applied in favor of highly adapted individuals. This leads to a higher average fitness for the entire population from which the *mating pool* is formed. Following this stage, recombination operators are applied to the members of the mating pool to generate new individuals. As can be seen in Fig. 3.8 the number of *fit individuals* has increased. The flowchart of the entire process of a genetic algorithm is depicted in Fig. 3.9.

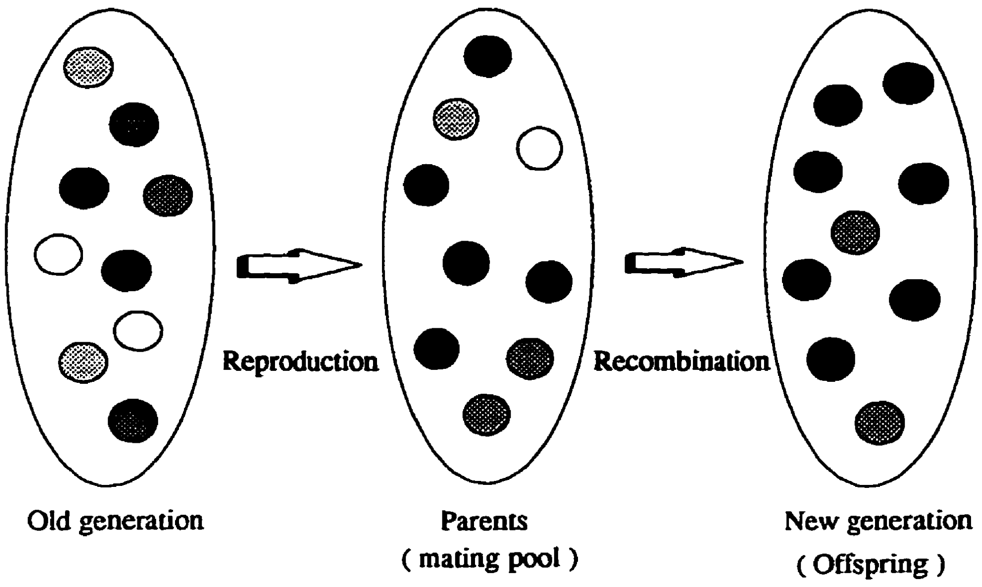


Figure 3.8: Graphical illustration of single iteration.

3.4 Coding the design problem of a fuzzy controller

To translate the fuzzy controller design problem into a genetic algorithm, different parameters of the controller should be distinguished and encoded based on the desired resolution. In what follows, three different parameters are discussed.

3.4.1 Normalization and denormalization factors

Normalization and denormalization factors, i.e. N_e, $N_{\dot{e}}$, $N_{\dot{u}}$, are crucial to fuzzy controllers since they determine what portion of the decision table can be used. In fact, they change the membership functions uniformly over the input/output domains. In this way, the controller's gain over the entire input domain can be adjusted. This immediately leads to the fact that the normalization and denormalization factors are also essential to the stability

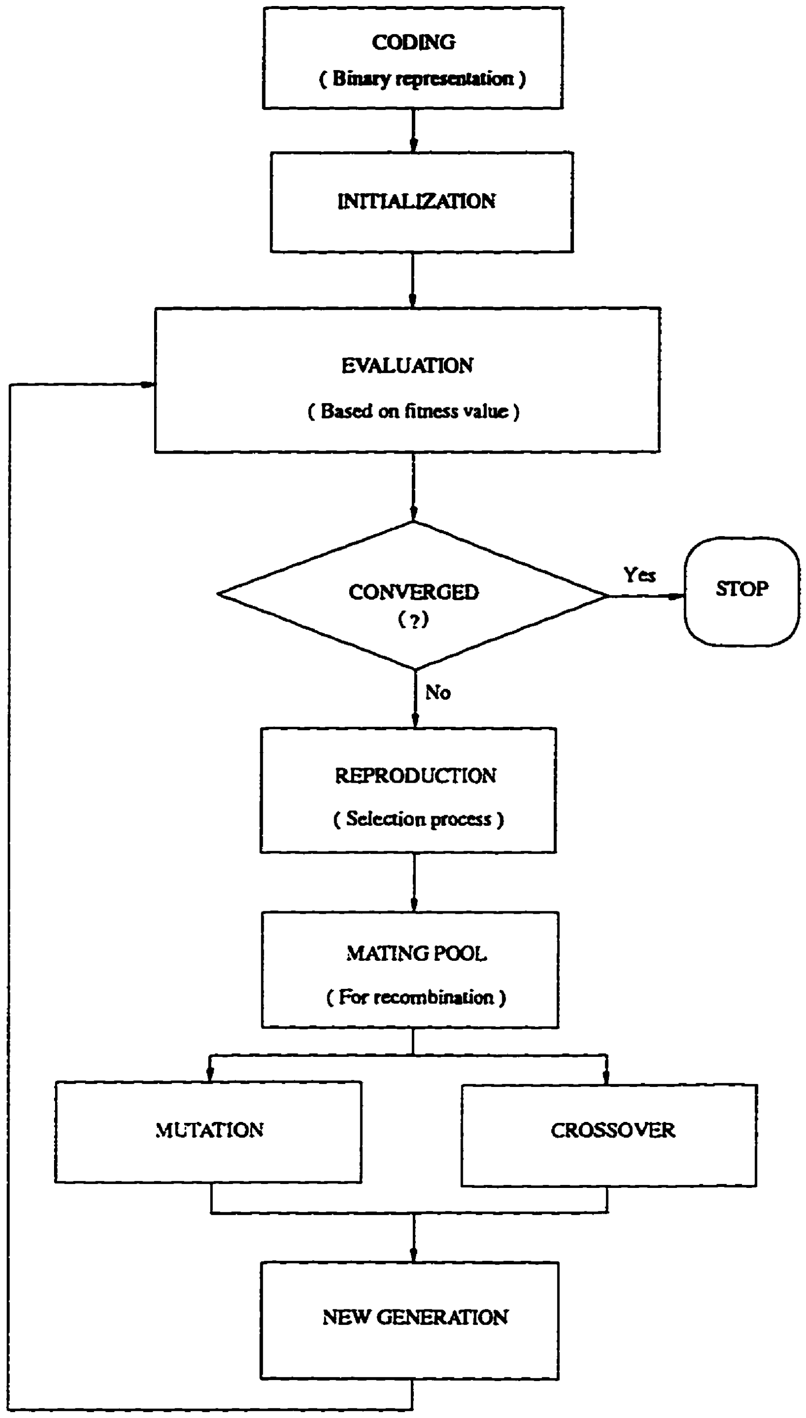


Figure 3.9: Flowchart of a genetic algorithm.

of fuzzy controllers. In reference to classical control theory, they have the same role as K_P and K_I have in the proportional plus integral controller. Figure 3.10 illustrates the genetic representation of such factors as a 30-bit

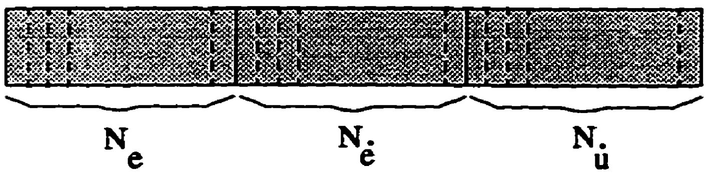


Figure 3.10: Bit-string representation of normalization and denormalization factors.

string where for every parameter a resolution of 10-bits has been assumed.

3.4.2 Membership functions

In contrast with normalization and denormalization factors, the membership functions are responsible for local adjustment of the controller's gain. Viewed in a state space perspective, by dividing this space, they construct a framework which defines the characteristic points. Therefore, they are as important as the characteristic points. As demonstrated in Section 2.7.2 of Chapter 2, two different sets of membership functions for input and output variables, results in a completely different mapping even though the decision rules are identical. To have complete freedom in partitioning the state space, asymmetrical membership functions should be chosen that consequently suggest three different design parameters, i.e. A_1, A_2, and, A_3 for each membership functions, (see Fig. 3.11). Let us consider seven membership functions for each variable for a controller with two inputs, 42 parameters are required to define the entire set of membership functions. Figure 3.12 illustrates such a string where every parameter is encoded with 10-bit resolution[6]. Note that as

[6]In the normalized domain of $[-1, 1]$, 10 bit resolution ends up with a precision of $(1-(-1)/2^{10} \simeq 0.002)$.

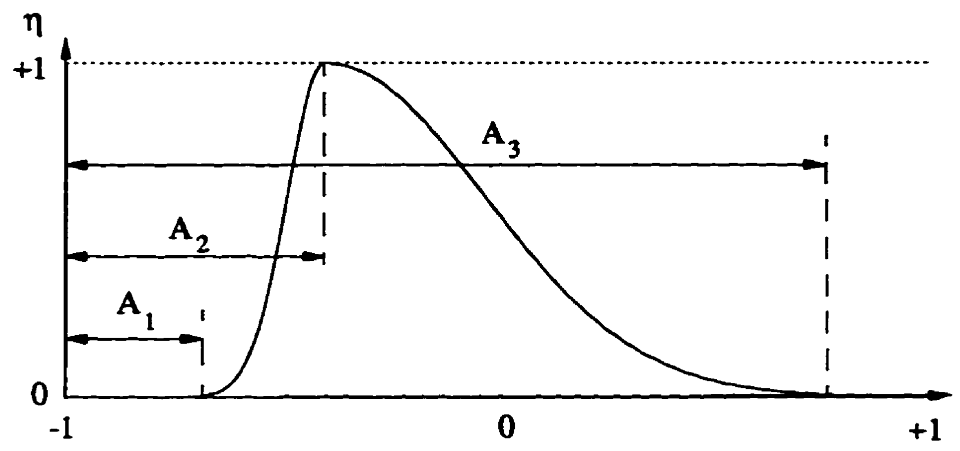


Figure 3.11: Membership function parameters.

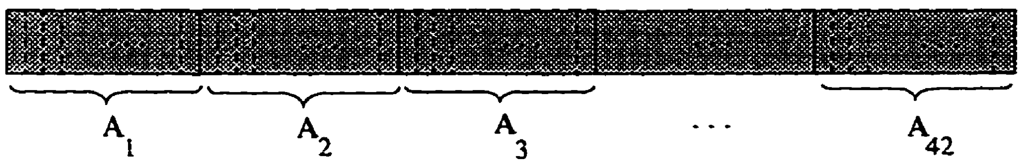


Figure 3.12: Bit-string representation of membership function parameters.

was shown in Section 2.7.2, in the case of 50% overlap for membership functions, the mapping functions are restricted only to the characteristic points.

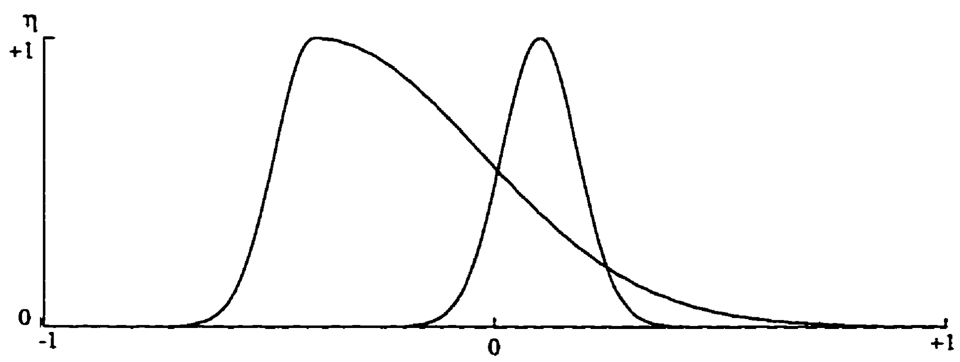


Figure 3.13: Full overlap of membership functions.

To have complete freedom, full overlap for membership functions is possible as shown in Fig. 3.13. In this way, the controller is able to scan any other mapping by which the performance index is reduced to a more desirable value.

3.4.3 Decision table

The rules of a fuzzy controller are responsible for the general shape of the fuzzy mapping function. In state space, they, in fact, control the state trajectory into equilibrium. To translate these rules into string format, every consequent part of a rule should be encoded in a binary form. Since every

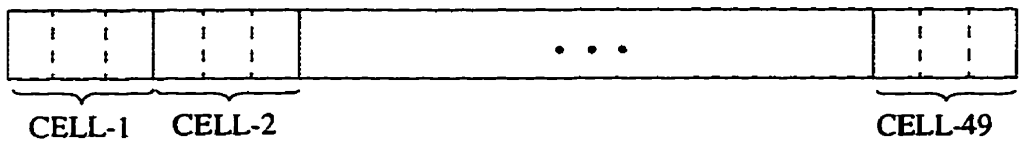


Figure 3.14: Bit-string representation of decision table.

consequent can take on only one of seven different values based on Table 2.5 in Chapter 2, every consequent can be represented by only three bits. In this way, a string of 49 parameters or equivalently $3 \times 49 = 147$ bits will represent the entire decision table, (see Fig. 3.14). Such a string can also be seen as the decision table of Fig. 2.5, once different rows are put beside each other.

3.4.4 Coding of the entire controller

The existing design techniques, to the author's knowledge, employ one part exclusively or at most three parts but sequentially tuned for the design of a fuzzy controller [22, 24, 31, 32, 34, 40, 54]. In the sequential approach, first normalization factors are found by an optimization technique where the membership functions and decision table are assumed to be constant. Once the optimization technique finds the best values of normalization factors, these parameters are fixed and the membership functions parameters are processed by the optimization algorithm. Finally, with the optimal values of normalization factors and membership functions a new decision table is found by a

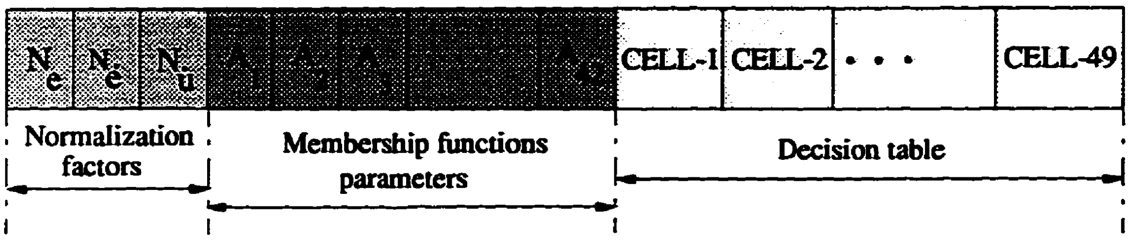


Figure 3.15: Bit-string representation of entire controller.

search technique, usually gradient based.

The point that the author intends to clarify is that the different parts of a fuzzy controller are not independent from each other and therefore, such approaches may lead, in general, to sub-optimal performance. In fact, by changing one parameter of a fuzzy controller some other parameters are changed. This implies that there is a interaction between different sets of controller parameters. For instance, by changing the normalization factors, the domain of every membership function is changed. Thus, for the same input value, some other membership functions are observed which fires some other rules. This, of course, leads to a new value for the controller output. It also indicates that the interaction of all design parameters of a fuzzy controller is, in effect, important to its success and hence the tuning and the organizing of the controller parameters should be, in principle, best done *concurrently* (see Fig. 3.15). To do this, a new bit string is required consisting of *all* the design parameters of the fuzzy controller. Such a string can be constructed by cascading the different strings together. It is evident that the resultant string as shown in Fig. 3.15, in effect, constitutes the complete information for the design of a fuzzy controller. Therefore, if such a string is incorporated in the control system loop, a performance index by which the fitness value of the string is evaluated can be obtained. In fact, the control loop is responsible for the evaluation of different individuals, i.e. solutions. This notion is illustrated in Fig. 3.16 in which two loops can be distinguished.

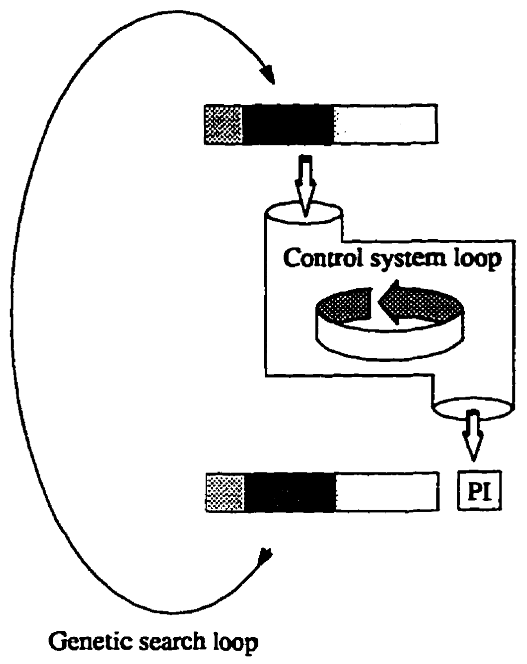


Figure 3.16: Illustration of internal and external loops.

The internal one, the closed loop system including fuzzy controller, is responsible for system operation corresponding to a given individual and assigns the performance index which in turn determines the fitness value, required by the external loop. This string is then processed by the genetic algorithm based on its fitness value. The process is continued until if converges to a near-optimal solution.

Looking back to Fig. 2.3 on page 15, and its modified version in Fig. 3.17, it is now the optimization technique which is responsible for development of each block in a fuzzy controller's structure. More importantly, it would be better if the proposed approach could incorporate any existing knowledge about the controller. The task of such a design algorithm is the modification of the existing knowledge and, at the same time, the investigation of new feasible structures. The approach proposed in this dissertation includes such a feature by the incorporation of any tentative values for controller parameters into the initial population. Figure 3.17 illustrates such a hierarchal approach

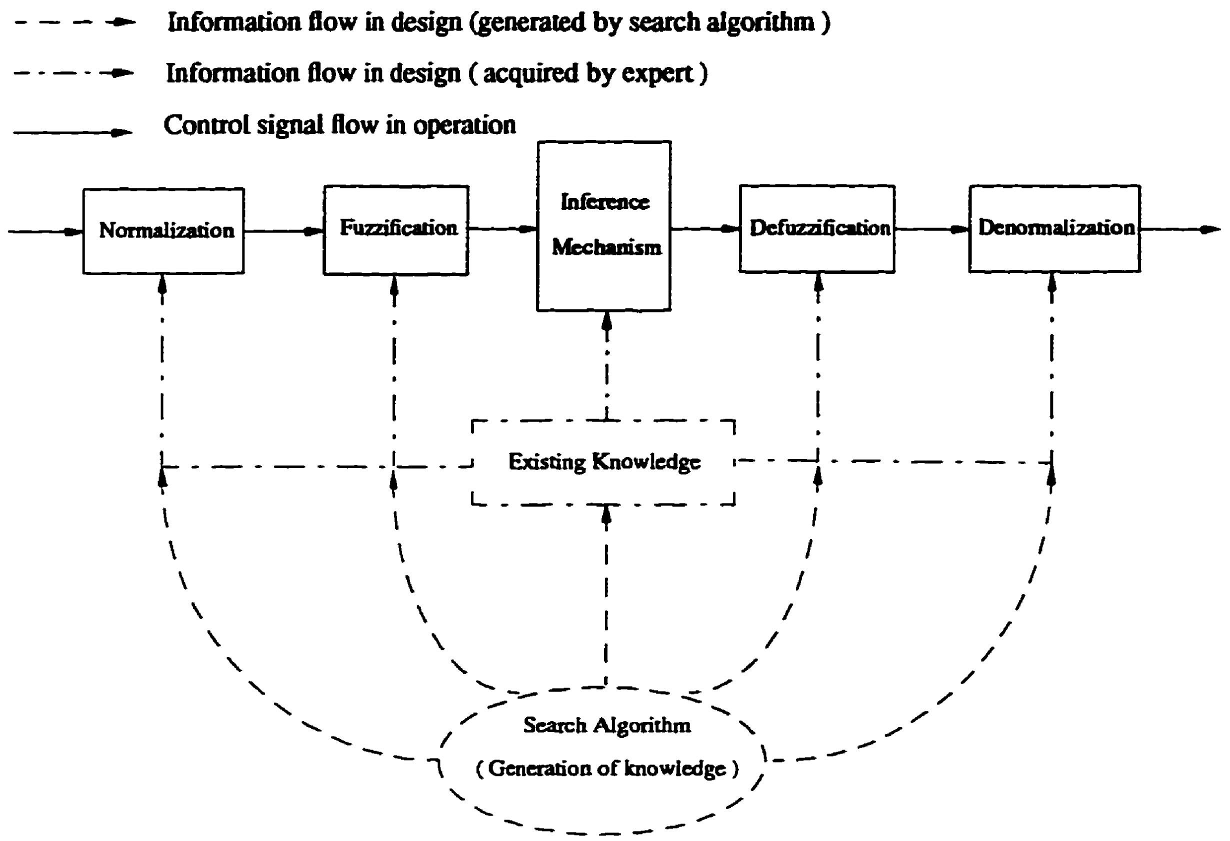

Information flow in design (generated by search algorithm)
Information flow in design (acquired by expert)
Control signal flow in operation
Normalization
Fuzzification
Inference Mechanism
Defuzzification
Denormalization
Existing Knowledge
Search Algorithm
(Generation of knowledge)

Figure 3.17: Different types of information incorporated in the fuzzy controller design.

to the design of fuzzy controllers.

3.4.5 Fitness function

Since in a genetic algorithm, each individual represents a possible solution to the problem, a particular *fitness function* is required for the evaluation of the individuals. In this way, for every particular chromosome (i.e. each solution), the fitness function returns a single numerical value, which indicates the quality of that solution. In the context of optimization it is the performance index of the closed loop system that becomes the fitness function.

While the examples presented in this dissertation are based upon a particular performance index, this does not affect the generality of the proposed technique. As shown in Section 4.4, a multiple performance index can also be used. Furthermore, any constraint on the controller parameters or system models, can be incorporated into the optimization technique to convert the unconstrained optimization problem into a constrained optimization problem.

3.5 Chapter summary

This chapter attempts to provide some insight for the task of fuzzy controller auto-design. It turns out that the objective function, which is in fact responsible for the construction of the search space hyper surface, seems to have a multi-modal nature in a space with, say 94 dimensions (cf. Section 2.6). Also, the question whether or not such a hyper surface is continuous or differentiable, cannot precisely be answered ahead of time, and this is another fact that does not allow us to employ gradient-based techniques for this optimization. It is such insight, therefore, that directed us to the genetic algorithm as an appropriate choice for our optimizer. As an alternative view, if fuzzy controllers are thought of as one type of artificial intelligence, it seems more natural to choose an optimization approach, again from artificial intelligence. For instance, both techniques, i.e. fuzzy control and genetic algorithm, share the feature that they do not require derivatives for their information processing.

With this fact in mind, a brief overview of the genetic algorithm has been presented. Our novel solution, which is concurrent auto-design of a fuzzy controller and simultaneous optimizing of performance, is proposed in the last part of the chapter. The point, worth emphasizing here, is that even though each sub-string (e.g. devoted only say to membership functions) can be optimized alone for the auto-design of a fuzzy controller, it does not yield an over-all optimal solution. The interactions among different parts of a fuzzy controller, i.e. normalization factors, membership functions, and table cells must be taken into account, as is made clear in Chapter 5.

Chapter 4

Genetic based auto-design of a fuzzy controller for induction motor drives

4.1 Introduction

Adjustable speed drive technology has evolved enormously over the past 30 years. This evolution has been made possible because of technological advances in a number of related fields, such as power semiconductor devices, converter topologies and control techniques. Moreover, the advent of microcontrollers and digital signal processors (DSP) has greatly assisted the practical implementation of the newer control techniques. In particular, the technique of field oriented control has advanced the control characteristics of ac machines to such a degree that it is now the most attractive technique for torque control of ac machines and is becoming an industry standard. Under field oriented control, an induction motor drive dynamics imitate that of a separately excited dc motor drive, with all the advantages of using an induction machine. More recently, with the application of intelligent control technologies, such as expert systems, fuzzy logic and neural networks, the frontier of adjustable speed drive technology [5,15] is advancing still further. Among the artificial intelligence (AI) techniques, fuzzy logic is perhaps the most successful one, if judged from the standpoint of the number of practical applications [27]. Fuzzy logic demonstrates significant potential for advancing power electronics technology [2,5,7–16]. Its ability to incorporate qualitative knowledge and to handle imprecise information makes it very attractive to power electronic systems where non-linearity is a common feature and a precise model, in general, is difficult to obtain.

In many cases drive systems are subject to load disturbances and parameter variations which make these systems highly uncertain. This leads to a quest for highly robust controller schemes in such applications. Fuzzy logic controllers are quite well known for their robustness if designed properly. Furthermore, if the performance benchmark of the drive system can only be expressed qualitatively, fuzzy logic control is more convenient than conventional counterparts. Despite these benefits, the application of fuzzy control has been impeded to some extend due to the lack of a systematic design approach [7,10,15,26,27,55]. Furthermore, a great deal of time and effort can be spent on fine tuning and yet, there is no guarantee of achieving an optimal performance.

The application of fuzzy logic control for the field oriented control of induction motor drives is quite new [6]. Although, some studies have been carried out, most, if not all, are based on conventional trial-and-error techniques [2,6,13,16–20]. In [6], for instance, C. Y. Won and B. K. Bose have employed fuzzy logic for induction motor position control with field oriented control. They have proposed two different decision tables, one for coarse tuning and the other for fine tuning. While this improves the system performance to some extent, the trial-and-error technique has been employed to modify a standard decision table where the membership functions are assumed to be fixed. In another attempt, H. Liang and H. Y. Chen, in [20], have used the same approach but this time, for the speed control of induction motor.

At this point, it is worthwhile to mention that the application of fuzzy control is not limited to finding the best dynamic for electrical drives. It can also be applied for other purposes such as minimizing the input power, maximizing the power factor or efficiency, and so on. In these cases, new decision tables and new membership functions are required in which the foregoing design and tuning problems arise again. The EPA[1], for instance, has conducted research [2], to enhance motor efficiency by employing fuzzy logic for variable speed drives. Here, not the closed loop, but the traditional open loop control approach is taken, in which the volts / Hz is held constant. The point is that even though the number of rules is small, e.g. 13 rules, to overcome the tuning problem of the fuzzy controller, the designers have used a development software tool, TILShell[2], by which the trial-and-error approach can be accomplished in a less tedious and less time consuming manner.

In what follows, a general description of indirect field oriented control, as used in an induction motor drive, is first presented from two standpoints; one traditional, another novel. Then a conventional fuzzy controller is employed for speed control of an induction motor with field oriented control. The proposed automatic design technique is then employed to design an optimal fuzzy controller for the same system. Three cases are considered followed by simulation results to demonstrate the efficiency and superiority of the proposed approach compared to the other conventional counterparts.

[1]Environmental Protection Agency of U.S.A.

[2]TILShell is a software development tool by which the design and tuning of a fuzzy controller can be performed in a menu driven environment.

For the proposed technique the tuning processes of input/output normalization factors and the developing and partitioning of a control surface are automated. While there has been a great deal of effort in the development of field oriented control, and the invention of the idea stems from a deep understanding of physical phenomena of electrical machines, the novel view presented here has its roots in a pure mathematical framework. able to derive the same non-linear transformation As mentioned earlier, the performance measure can be changed. This, in essence, is another indication that the proposed technique is very general, being applicable to most fuzzy control applications. It is in this sense that the application of the proposed technique is extended to highly non-linear systems in Chapter 5 where some other facets of the proposed approach are investigated.

4.2 Field oriented control

4.2.1 Traditional view

An induction motor is a high order, multi-variable, non-linear, uncertain system which seems to be very difficult to control. The well known dq model of an induction motor is presented in Appendix B. In effect, the terminal voltages and currents of the machine, which are readily accessible, are non-linearly related to the electromagnetic torque and flux. Any change in input currents not only leads to a change in electromagnetic torque, but also changes the motor flux. This indicates that there is an inherent coupling between torque and flux. Since the flux has a slow transient, this coupling leads to a sluggish change if any incremental torque is demanded. On the other hand, it is quite well known that a linear relation between the control variable (currents or voltages) and the controlled variable (torque or speed), is desirable in any control system. If this happens, not only a high performance drive for fast torque response can be achieved, but also the heritage of linear control theory is effectively used for the development of powerful control methods for non-linear systems.

In 1969, Blaschke, a german engineer, established a new decoupling control technique by which a linear relationship between torque and stator current component is attainable [4]. Based on this technique, if an observer is situated on a rotor flux line[3] and rotates with the same speed as the flux line, it will be reported that the flux is constant in time and space (of his own frame). This can be equivalently interpreted as having a constant flux component of the stator current. Viewed in this perspective, any change in the torque component, which is perpendicular to the flux axis, does not affect the flux component of stator current. This equivalently means that a decoupling between torque and flux has been achieved. In this way, the torque can be controlled in proportion to the torque component of the stator current while the level of magnetization is constant. Thus, a fast torque response is attainable and the complexity of the dynamic model is greatly simplified. This also facilitates the application of modern control techniques to enhance drive performance.

To attain the decoupling discussed above, the flux component of stator current should be aligned with the rotor flux. This immediately requires two different transformations, one for three phase to two phase mappings and the other for vector rotation of stator current. While the former is a simple linear transformation, the latter requires identification of the rotating flux at every instance. This shows how important it is to know the position of rotor flux in field oriented control.

Although the concept of field oriented control, in theory, was very im-

[3]The most suitable location for such an observer would be along the flux line of greatest flux density.

pressive, it was not readily applicable for more than one decade. However, the development of microprocessor digital circuitry and the advent of high frequency power semiconductor devices, provided the practical means to implement field oriented control. In today's drive system, field oriented control has widely been accepted as the most attractive torque control scheme for ac machines.

4.2.2 A novel view

Global input-output linearization, as a part of the *differential geometry* technique, is an approach to non-linear control design which has attracted a great deal of interest in recent years. Based on this technique, if the system is input-output linearizable, which is the case with the induction machine, there exists a state feedback transformation that transforms the non-linear input-output relation into a linear one. The interesting part of this approach is global linearization as opposed to local linearization as is done quite often in control system design. Figure 4.1 depicts the general structure of global

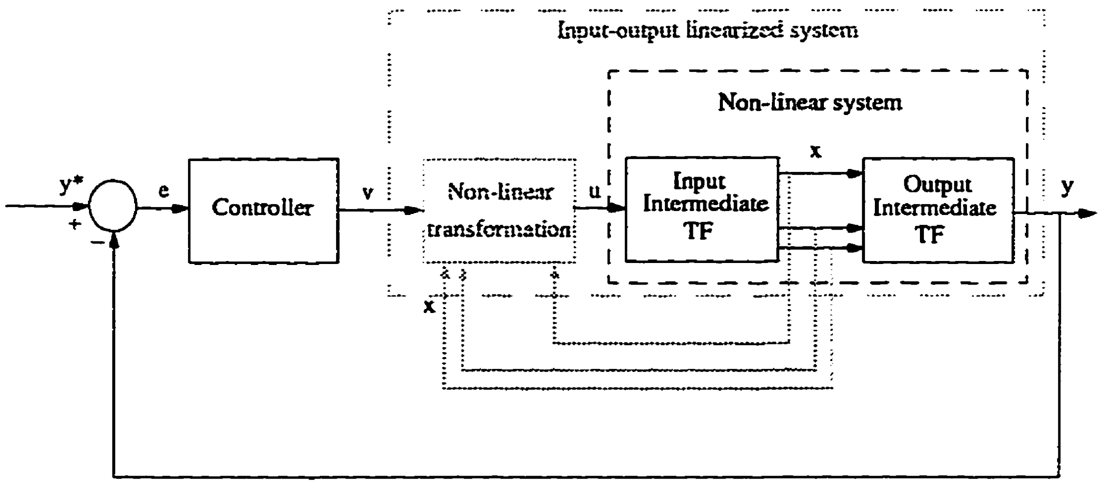


Figure 4.1: General structure of global input-output linearization.

input-output linearization where x denotes the state variable, y is the system output, u is the system input, v is the state feedback transformation input, e is error, $y*$ is the desired output, and TF denotes transfer function. In this figure, while the mapping $u \dashrightarrow y$, i.e. actual system, is non-linear, the new mapping $v \dashrightarrow y$, the overall system, becomes globally linear [56]. That is the reason that this method has been called ***input-output linearization*** as opposed to ***input-state linearization.*** It is quite clear that to achieve such a linear relationship, the state feedback transformer should be non-linear as well. If all the state variables required for the linear transformation are not available by direct measurement, as is the case with the induction motor, a state observer is required to estimate the unavailable states. This estimation would be on the basis of a dynamic model and output measurement (see Fig. 4.2). This is exactly the case for the so called direct field oriented control of an induction machine. If a feedforward input-output lineariza-

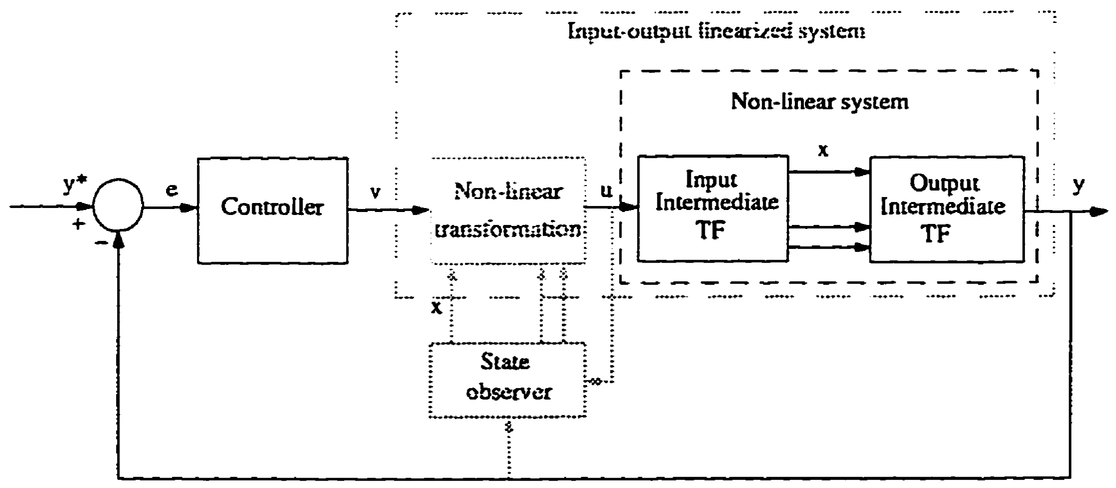


Figure 4.2: Global input-output linearization based on a state observer.

tion is applied which requires a feedforward path, as shown in Fig. 4.3, a new scheme will be achieved which, in the context of variable speed ac drives, is called an indirect field oriented control (IFOC) drive. In this case, the state transformation, which is in effect a non-linear compensator, turns out to be the field oriented equations for induction motor speed control (see Fig. 4.4).

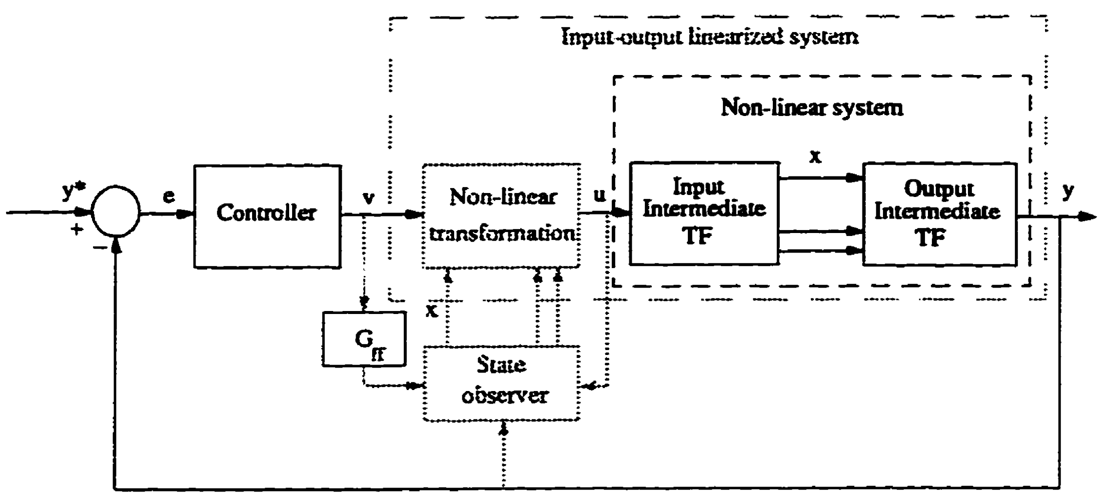


Figure 4.3: Feedforward input-output linearization.

Once this non-linear compensator is found properly, a linear relation between input, I_{qs}^*, and output electromagnetic torque, T_e, will be achieved and consequently Fig. 4.4 can be simplified as is shown in Fig. 4.5. This is valid as long as the parameter variations of the induction motor remain small, or if not, its variations are considered in the design of the non-linear compensator.

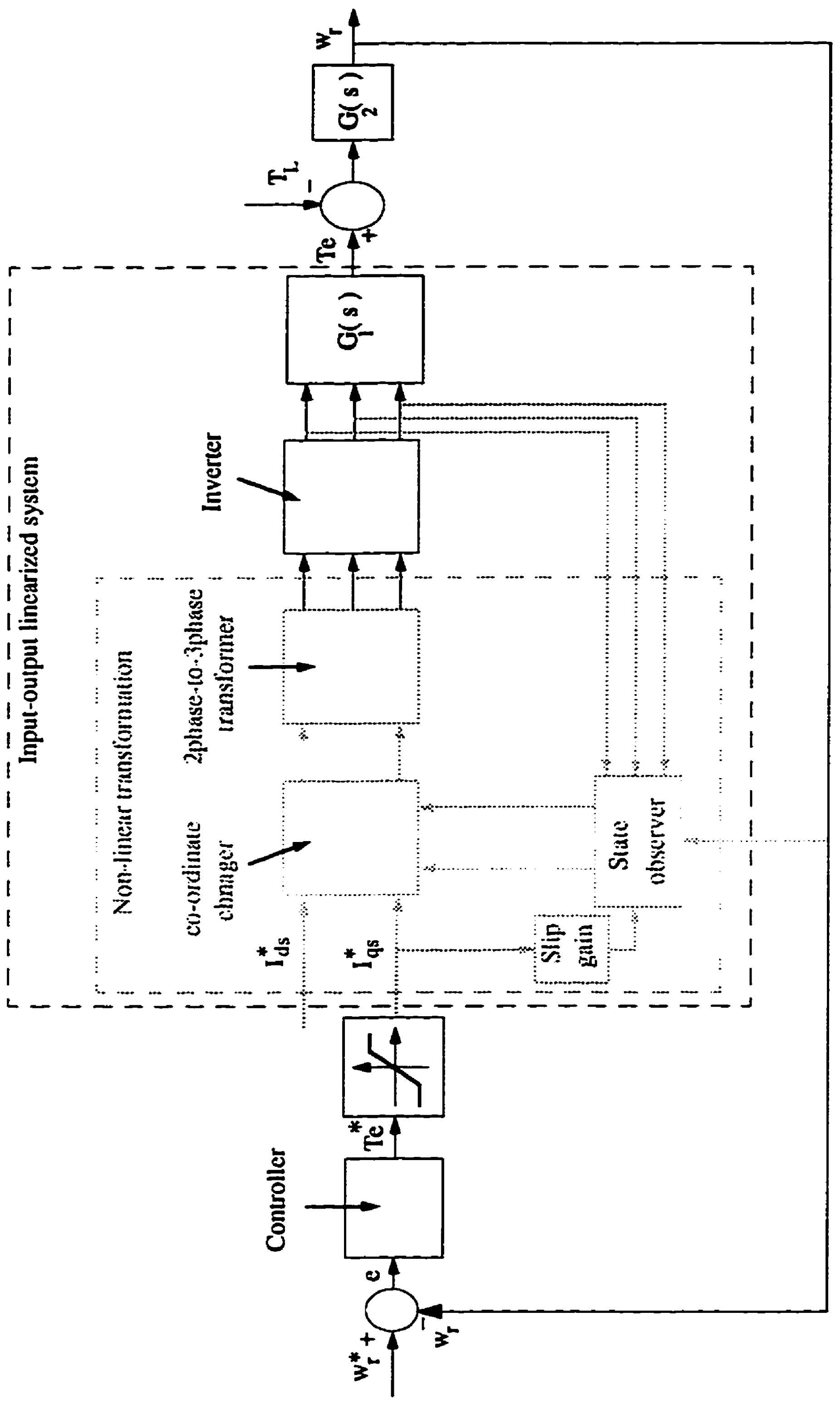


Figure 4.4: Block diagram for IFOC of an induction motor.

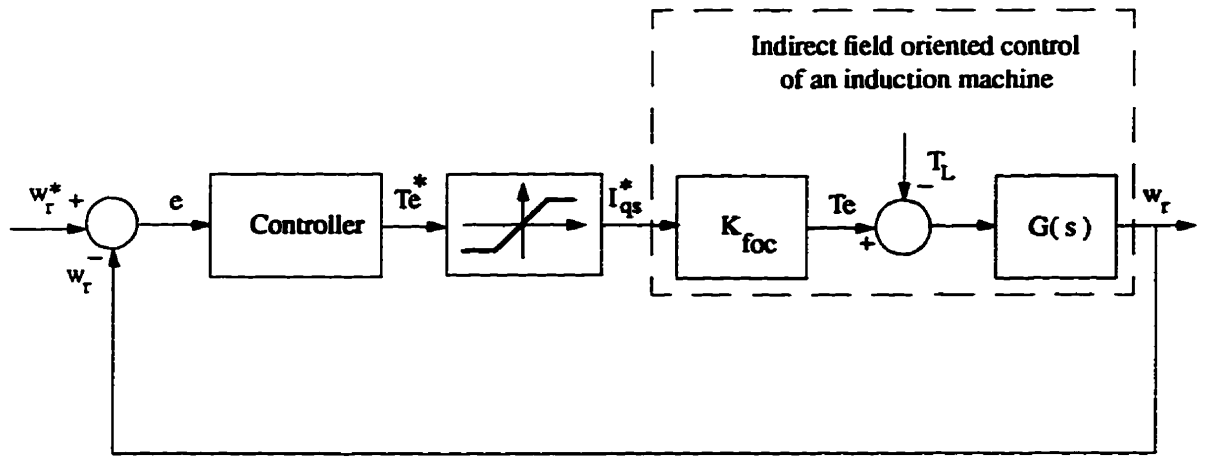


Figure 4.5: Simplified control block diagram of IFOC for an induction motor.

4.3 Conventional fuzzy controller

As described in Chapter 2, a fuzzy controller consists of a decision table, two non-linear interfaces (fuzzification and defuzzification), and two linear interfaces (normalization, and denormalization factors), (cf. Fig. 2.3 on page 15). The controller has two inputs, e and $\dot{e}$, and one output, $\dot{u}$, as shown in Fig. 4.6. Let the membership functions of input-output spaces be gaussian;

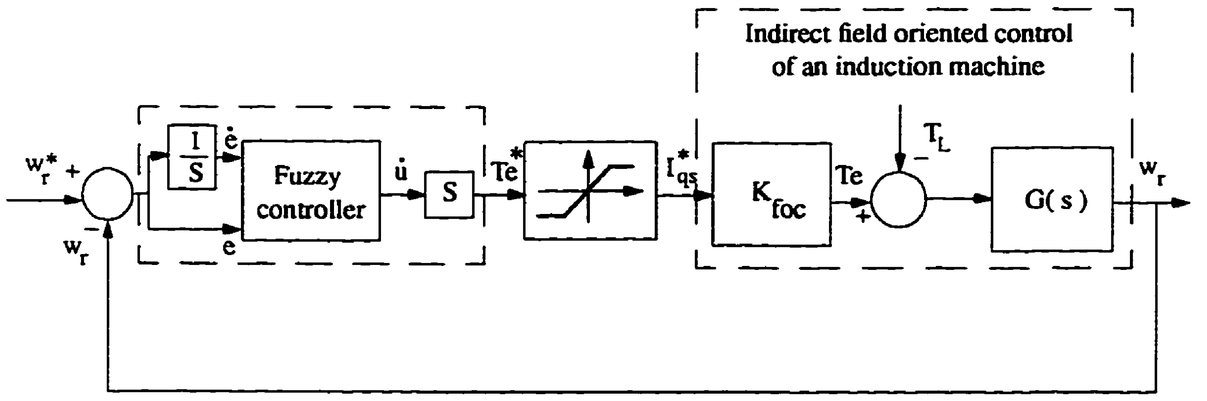


Figure 4.6: Simplified IFOC block diagram with fuzzy controller

for convenience re-drawn in Fig. 4.7. Also consider a commonly used decision table, which in Chapter 2 is referred to as the sliding mode table; for convenience re-drawn in Fig. 4.8.

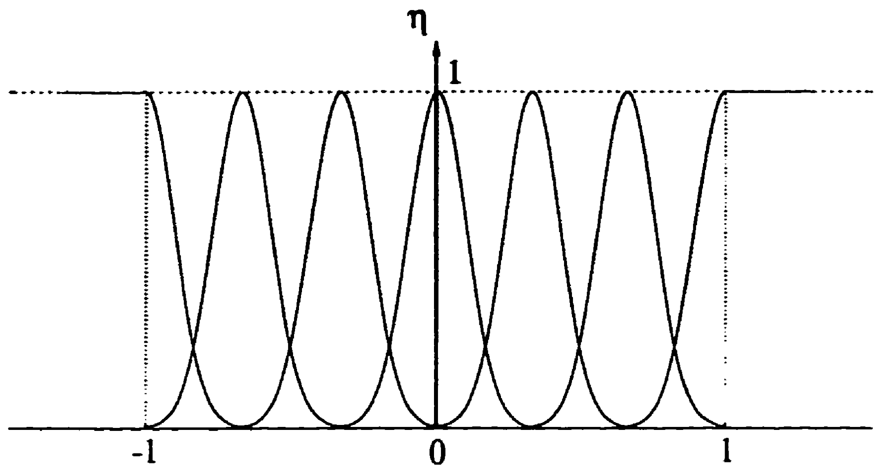
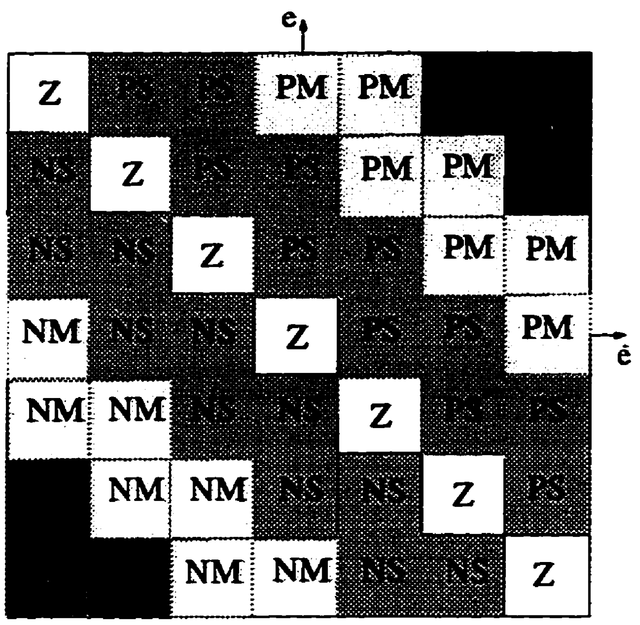


Figure 4.7: Gaussian membership functions

It is a common practice in the design of fuzzy controllers, that the designer starts with these standard forms of membership functions and decision table and then modifies the normalization factors, rules, and membership functions until a reasonable performance is achieved. Such a conventional design procedure is shown in Fig. 4.9. How many times the modification loop should be iterated depends on the designer's experience, expertise and perhaps luck! To avoid the purely blind trial-and-error technique, a new semi-algorithmic approach is described here that accelerates the modification loop. This semi-algorithmic approach is the basis for the evaluation of the proposed genetic algorithm auto-design approach, resulting in a much fairer comparison than would be otherwise possible.

Based on the semi-algorithmic approach, the system trajectory in state space is observed and then the cells of the decision table or corresponding membership functions are changed to follow the desired trajectory. The essence of this approach is based on the notion that a desired system step response has a general shape as shown in Fig. 4.10, and its corresponding



Z	PS	PS	PM	PM		
NS	Z	PS	PS	PM	PM	
NS	NS	Z	PS	PS	PM	PM
NM	NS	NS	Z	PS	PS	PM
NM	NM	NS	NS	Z	PS	PS
	NM	NM	NS	NS	Z	PS
		NM	NM	NS	NS	Z

Figure 4.8: Sliding mode table.

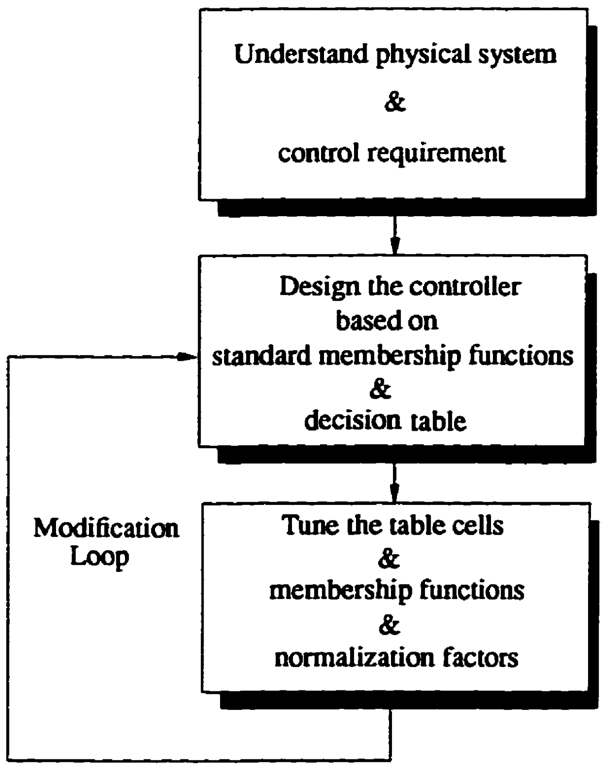


Figure 4.9: Conventional design algorithm for a fuzzy controller.

trajectory in fuzzy-partitioned state space[4] has a shape as shown in Fig. 4.11. On the other hand, the cells of the decision table are responsible for acceleration or deceleration of the system trajectory. Therefore, the link between the cells and trajectory can readily be viewed in state space, and the modification process can be directed in a more convenient fashion[5].

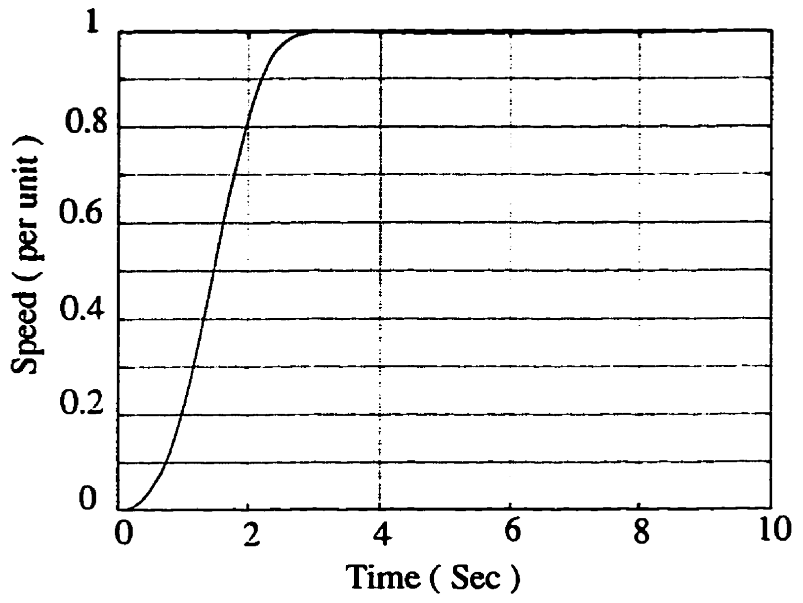


Figure 4.10: A typical desired step response for a drive system.

Based on this modification approach, a fuzzy controller is designed for indirect field oriented control of an induction motor. The induction motor that is used for this simulation is a 3-phase, 6-pole, 220 V, 10 hp, 60 Hz motor (cf. Appendix B for the mathematical model) with the following parameters expressed in per unit [57]:

$$X_s = 2.1195 \quad X_m = 2.0420 \quad R_s = 0.0453$$

$$X_r = 2.0742 \quad H = 1\,\mathrm{s} \quad R_r = 0.0222$$

[4]or as called in the fuzzy literature, *linguistic state space*.

[5]This semi-algorithmic approach is not the principal contribution of this thesis.

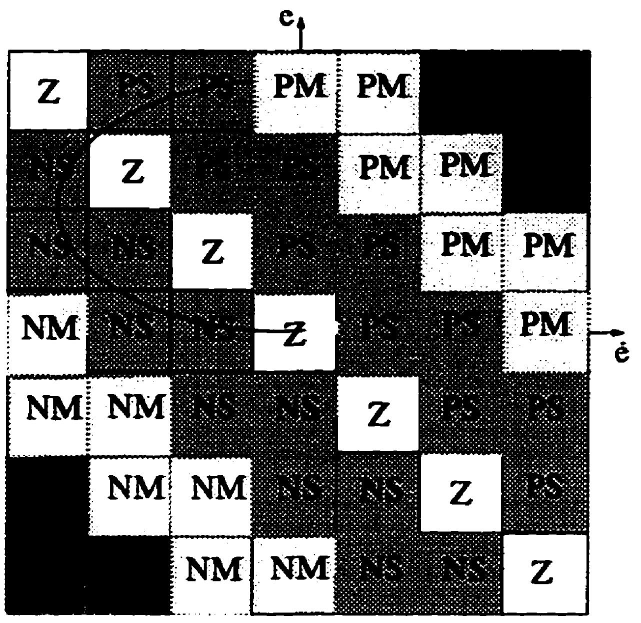


Figure 4.11: Trajectory of the desired typical response in fuzzy- partitioned state space.

The resultant control policy, i.e. normalization factors, decision table, are shown in Table 4.1 and Fig. 4.12respectively.

	Conventional fuzzy controller
N_e	10.07
$N_{\dot{e}}$	2.97
$N_{\dot{u}}$	6.60

Table 4.1: Normalization factors found by trial-and-error

It should be noted that since the modification of membership functions is more difficult, the resultant membership functions have the same shape and overlap as the standard ones (cf. Fig. 4.7 on page 64).

The speed response of this fuzzy controller is depicted in Fig. 4.13 for a step change in the reference command of the closed loop system.

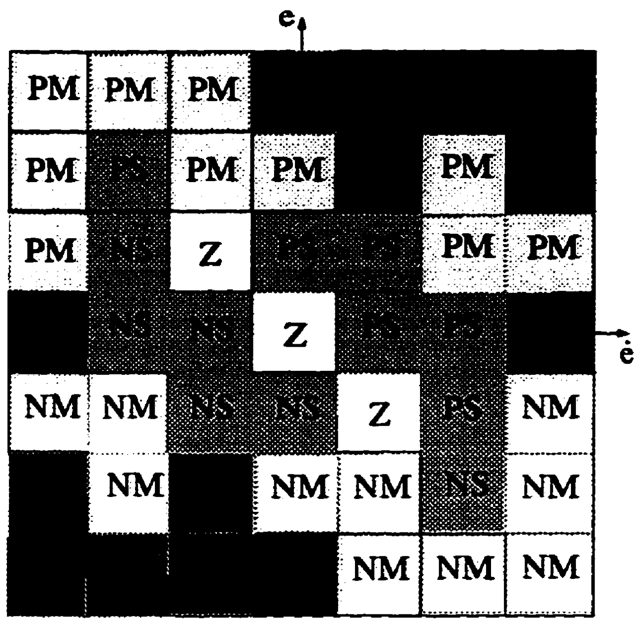


Figure 4.12: Modified control policy of conventional approach.

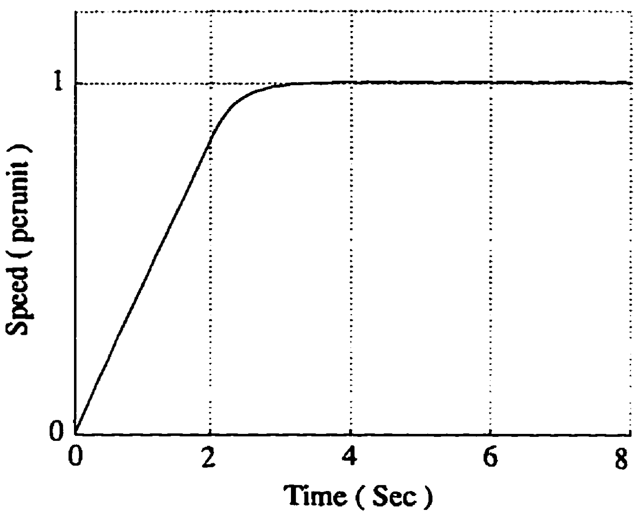


Figure 4.13: Step response of motor speed based on semi-algorithmic approach.

4.4 Optimal fuzzy controller

In this study of auto-design of a fuzzy controller, two different approaches are considered. First a sequential design of a fuzzy controller is considered, which consists of self-tuning[6] as well as self-organizing[7]. These approaches are referred to as case-1 and case-2, respectively. The second approach is a concurrent design of the fuzzy controller by which normalization factors, membership functions, and decision rules are optimized concurrently. In the following simulations, this approach is referred as case-3.

4.4.1 Performance index

Every optimization requires an objective function. In control, the objective function is, in fact, the performance index which is a quantitative value, measuring deviation from an ideal performance. In some situations, there may be more than one criteria to be satisfied. In these cases, the optimization problem is called *multi-objective optimization* in which the ultimate objective function is usually a linear combination of some different objective functions.

For drive applications, consider a step speed response where the goal is a short rise time, small overshoot, and near-zero steady state error. Since most performance indices[8] in classical control do not necessarily fulfill these requirements simultaneously, a multiple objective function is required.

In this respect, a measure of a fast dynamic response may be chosen as

$$J_1 = \omega_t \int_0^t |e|\, dt \tag{4.1}$$

[6]Some words of caution, at this point, are worth mentioning. By *self-tuning*, we do not mean an on-line tuning. The proposed technique is, in effect, an off-line approach for finding the best values of normalization factors.

[7]In a self-organizing approach, the consequent parts of the decision rules, i.e. table cells, are found based on the optimization.

[8]Such as integral of absolute error, integral of square error, and so on.

while the steady state error can be measured by

$$J_2 = \omega_{ss} \int_0^t |e|\, t\, dt \tag{4.2}$$

The penalty on the multiple overshoot of the response can be defined by

$$J_3 = \omega_p \int_0^t \delta(\frac{dy}{dt}) \,.\, |y^* - y(t)|\, dt \tag{4.3}$$

where

$$\int_{0^-}^{0^+} \delta(\frac{dy}{dt}) = \begin{cases} 1 & : \; dy/dt = 0 \\ 0 & : \; dy/dt \neq 0 \end{cases} \tag{4.4}$$

In this case, $\delta(dy/dt)$ detects the instances that overshoots (or undershoots) occur and the term $|y^* - y(t)|$ determines the response deviation from the desired value.

To achieve these objective functions simultaneously, the resultant performance index can be defined as

$$J = W.J_i \tag{4.5}$$

where $J_i = [J_1 \,,\, J_2 \,,\, J_3]$ and $W = [\omega_t,\, \omega_p,\, \omega_{ss}]^t$ is the weight vector of the objective function and is application dependant. Furthermore, even for a given system, the elements of this vector , i.e. ω_t, ω_p, ω_{ss}, are not independent from each other and should be specified based on the relative importance of each term. For the drive systems with parameters given in table 4.1, the following values were found based on trial-and-error to fulfill our requirements[9].

$$\omega_t = 1 \qquad \omega_p = 6 \qquad \omega_{ss} = 1. \tag{4.6}$$

[9] In fact, properly defining the performance index is usually a difficult task, requiring some design intuition.

Hence, the resultant performance index becomes

$$\boxed{J = \int_0^t |e|\,dt + 6\int_0^t \delta(\frac{dy}{dt}).|y^* - y(t)|\,dt + \int_0^t |e|\,t\,dt} \tag{4.7}$$

This performance index is used for all the following simulations. In every case, simulation results are compared with those of conventional counterparts designed based on the semi-algorithmic approach.

4.4.2 Sequential approach

Case-1: Self-tuning fuzzy controller In this case, the genetic algorithm is applied to find the best normalization factors using the foregoing performance index. This is done with the standard decision table and membership functions used in the controller's structure. The proposed approach is able to find near-optimum normalization factors as listed in Table 4.3. The free parameter of the genetic algorithm are shown in Table 4.2. Figure 4.14 demonstrates the simulation results. As this table indicates the optimization program has converged in only 37 generations. Since the number of design parameters are small, the mutation rate is chosen high to permit the genetic algorithm to explore other points of the search space. The speed of convergence is shown in Fig. 4.15 in which the performance index (for the most fit individual) has been depicted versus the generation number[10].

Number of individuals	30
Percentage of crossover	100%
Max Percentage of mutation	8%
Number of generations	37
Number of design parameters	3
Bit-string length	30

Table 4.2: Case-1: Free parameters of a genetic algorithm

	Auto-Design FLC	Conventional FLC
N_e	4.58	10.07
$N_{\dot{e}}$	0.55	2.97
$N_{\dot{u}}$	10.05	6.60

Table 4.3: Case-1: Normalization factors

[10]For our genetic algorithm with the parameters of Table 4.2, it takes around 30 minutes

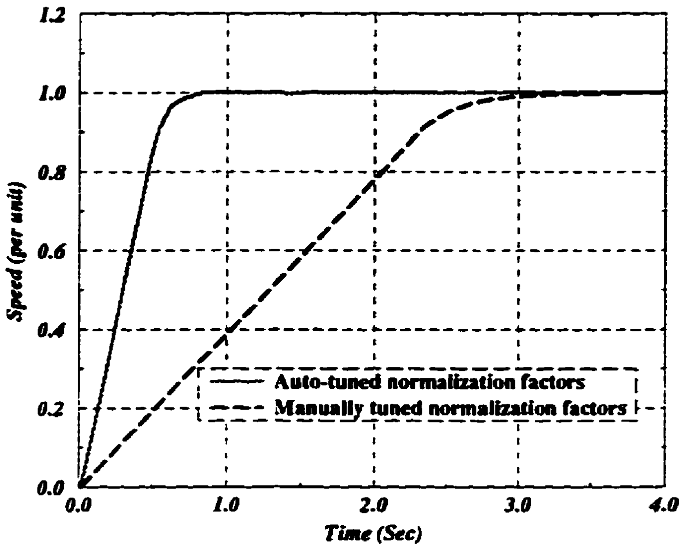


Figure 4.14: Case-1: Step response of motor speed for self-tuning approach.

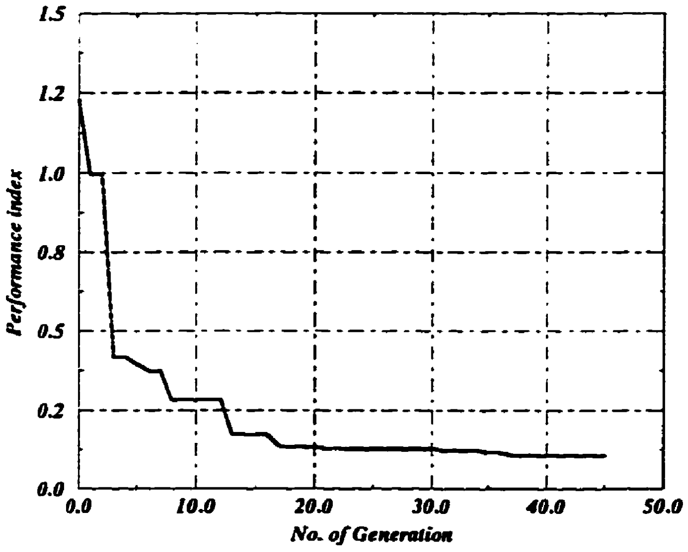


Figure 4.15: Case-1: Speed of convergence.

Case-2: Self-organizing fuzzy controller In another simulation experiment, the normalization factors, found in case-1 by the genetic algorithm (see Table 4.3), are chosen and the genetic algorithm is used to generate the decision table in a sequential manner. Also, a standard decision table is manually tuned using the semi-algorithmic approach described above. The simulation results shown in Fig. 4.16 indicates the superiority of the sequential auto-designed fuzzy controller over the sequential manually tuned counterpart.

The genetic algorithm with the free parameters shown in Table 4.4 was able to determine these rules after about 80 generations. The speed of convergence is shown in Fig. 4.17.

Number of individuals	48
Percentage of crossover	100%
Max Percentage of mutation	3%
Number of generations	80
Number of design parameters	49
Bit-string length	147

Table 4.4: Case-2: Free parameters of a genetic algorithm

4.4.3 Concurrent approach

Case-3: Self-tuning self-organizing fuzzy controller Since the different stages in a fuzzy controller are not independent, the sequential auto-design approach may not lead to the optimal solution. Therefore, concurrently generating and modifying different parameters of the fuzzy controller seems to be a logical approach. Hence, in this case, the three normalization

on SPARC 5 computer to find the optimum values.

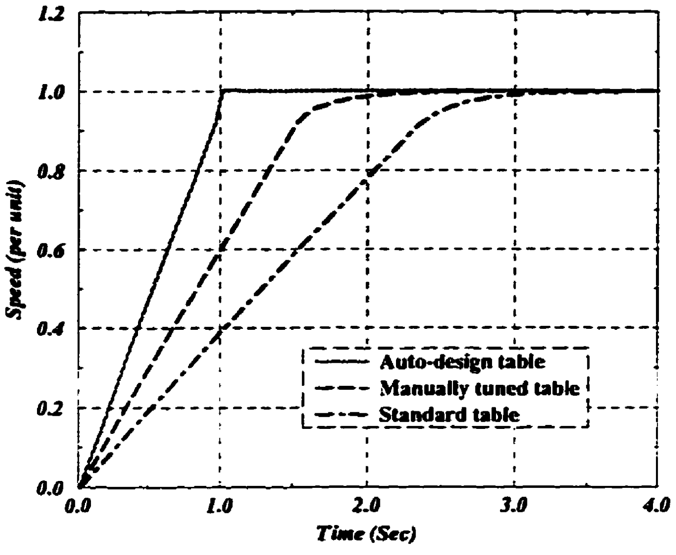


Figure 4.16: Case-2: Step responses for different designs of decision table.

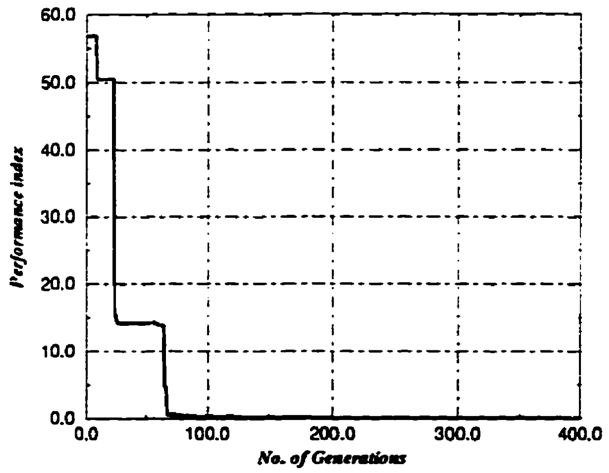


Figure 4.17: Case-2: Sequential approach (a) Auto-design versus manual design (b) Speed of convergence for a genetic algorithm.

factors, membership parameters, and the table cells, are used in the construction of the bit-string. While normalization factors determine the proper domain of the control surface, the table cells are responsible for the best coverage of the control space. At the same time, membership functions are involved in the partition of this surface in an optimal fashion. In effect, not only the antecedents and consequents of control rules are optimized, but also the proper domain on which these membership functions rely, are found as well.

To achieve complete freedom in overlap, support, and asymmetry, every membership function is represented by three parameters (cf. Fig. 3.11 on page 47). The center of gravity method is chosen as the defuzzification technique, to achieve better smoothness in the control surface. Furthermore, the membership functions have been defined as gaussian-shaped to accompany the defuzzification technique in this smoothness process.

A concurrent auto-design fuzzy controller for indirect field oriented control was designed. Based on this experience, the genetic algorithm with the free parameters shown in Table 4.5 was able to find the near-optimum solution with a population of 32 individuals, in almost 310 generations (see Fig. 4.20). This is due to the large number of design parameters involved in concurrent optimization. The normalization factors and membership functions are given in Table 5.2 and Fig. 4.18, respectively. The simulation results are shown in Fig. 4.19.

Number of individuals	32
Percentage of crossover	100%
Max Percentage of mutation	12%
Number of generations	310
Number of design parameters	94
Bit-string length	597

Table 4.5: Case3: Free parameters of a genetic algorithm

	Auto-Design FLC	Conventional FLC
N_e	10.14	10.07
$N_{\dot{e}}$	1.56	2.97
$N_{\dot{u}}$	9.35	6.60

Table 4.6: Case3: Normalization factors found by a genetic algorithm.

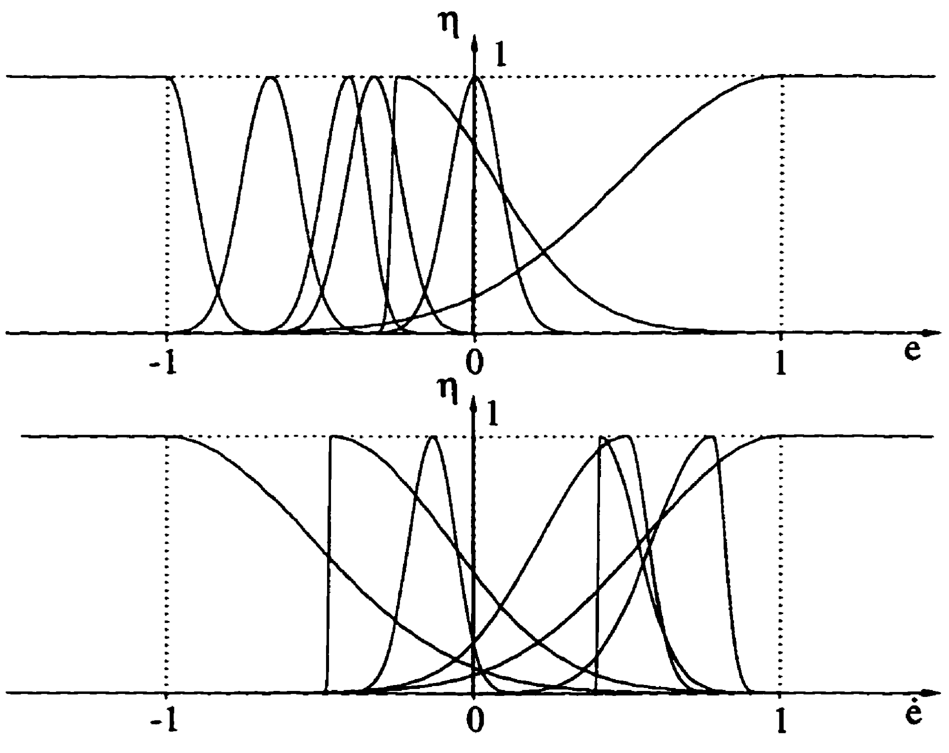


Figure 4.18: Case-3: The new membership functions found by a genetic algorithm.

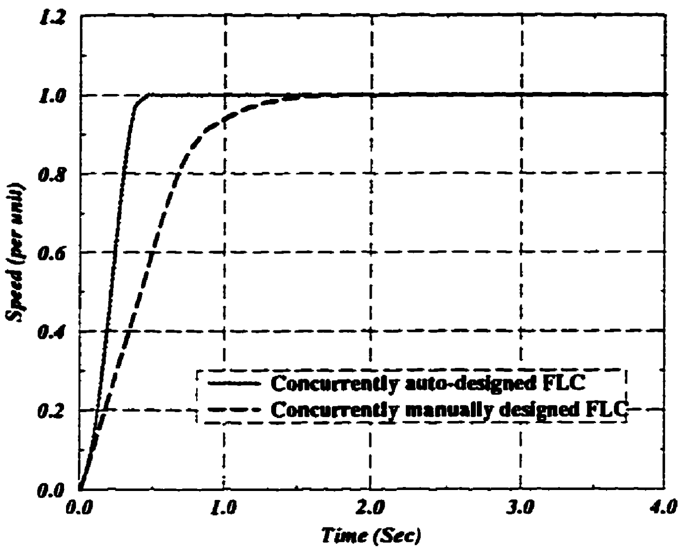


Figure 4.19: Case-3: Step response of motor speed for concurrent approach.

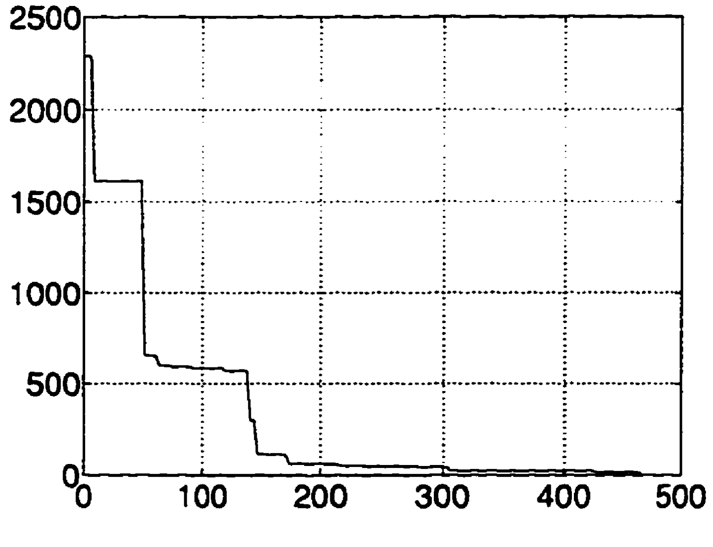


Figure 4.20: Case-3: Speed of convergence.

4.5 Chapter summary

In this chapter, a novel approach to field oriented control is presented, preceded by a brief review of the traditional approach. Based on differential geometry, a field oriented controller can be seen as a non-linear compensator that requires a state feedback observer to linearize the overall system from a new dummy input to the original output. Also in this chapter, the proposed auto-design technique is employed for the speed control of an induction motor drive with indirect field oriented control. While in this particular example, the performance index is chosen based on obtaining the best dynamic response, in other applications other criteria, such as input power, efficiency, power factor. and so on. can be employed to define the required control strategy. To illustrate the efficiency of the proposed technique, a semi-algorithmic approach is also proposed to enhance the trial-and-error approach of conventional fuzzy control design. Finally, to demonstrate the flexibility of the proposed approach, three different cases are considered based on different sets of design parameters. The simulation results demonstrate the superiority of the proposed technique in contrast with the conventional counterpart.

Chapter 5

Discussion

5.1 Introduction

While the proposed approach was originally developed for high performance induction motor drives, it should not be confined only to this type of system. The approach presented in this dissertation, in fact, can be viewed as a much more general approach for a large class of non-linear systems. It can also be applied for multi-input multi-output systems where the conventional trial-and-error approach is difficult, if not impossible. For this reason, the main motivation of this chapter is to demonstrate the generality of the proposed approach, and to point out several of its characteristics.

Aside from the full optimization for the fuzzy controller design [58, 59], a novel technique for the efficient design of a fuzzy controller is proposed. This technique is based on output partitioning as well as the sliding table. If fast tuning and development of the controller is of primary concern, the efficient approach should be employed.

In this chapter, a non-linear system is first chosen to explore some aspects of the proposed approach. An output partitioning approach as opposed to the input partitioning approach is first described. Then, a new point of view to the robust design of a fuzzy controller is discussed. Based on this view, a fuzzy controller, as a special class of variable structure controllers, can be designed in such a way as to have a sliding motion. If that happens then the robustness of a fuzzy controller can be ensured.

The non-linear system, employed in this chapter, has the following state space equations

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1^2 - 7x_1x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y = x_1 \end{cases} \tag{5.1}$$

where x_1 and x_2 are state variables and u and y denote the system input and output, respectively. Since the control objective is again to achieve the best dynamic and steady state behaviour, the same performance index as of equation 4.7 on page 71 is considered. For this system, however, the weight vector was found to be

$$W = [1 \quad 4 \quad 0.5] \tag{5.2}$$

Thus the resultant performance index becomes

$$\boxed{J = \int_0^t |e|\,dt + 4\int_0^t \delta(\frac{dy}{dt}).\,|y^* - y(t)|\,dt + 0.5\int_0^t |e|\,t\,dt} \tag{5.3}$$

5.2 Input partitioning versus output partitioning

5.2.1 Overview

While the optimal design of fuzzy controllers requires a search of the entire set of design parameters in a multi-dimensional space, the optimization problem can be simplified by reducing the number of these parameters.

To illustrate this point, consider the following example. For the time being, assume that the desired function of the controller is already known as given in Fig. 5.1, and our objective is to approximate this function to some degree of precision. One approach is to uniformly discretize the output

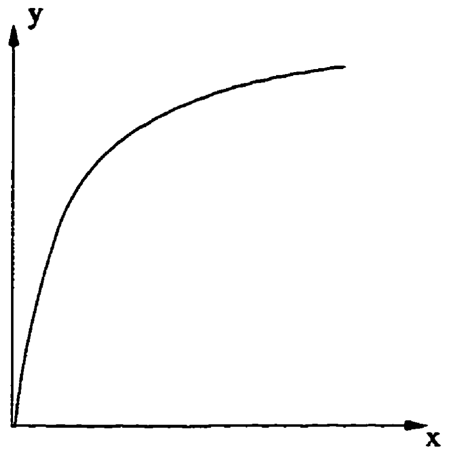


Figure 5.1: A desired function for the controller.

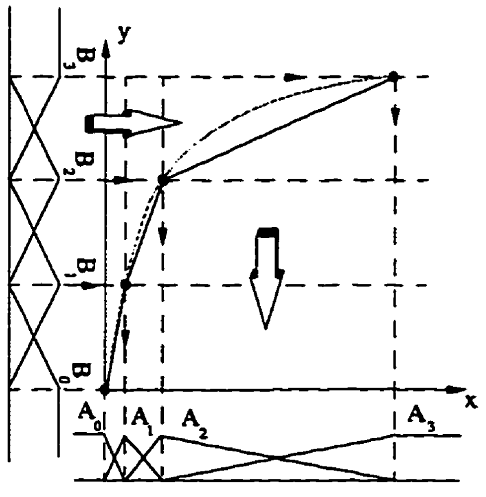


Figure 5.2: Approximated function by uniformly discretizing the input space.

space, along the y axis, and then obtain the desired discretized value for the input space, along the x axis, as shown in Fig. 5.2. The description rules are

If x is in domain A_x, Then y is in domain A_y

If x is in domain B_x, Then y is in domain B_y

With these rules and the uniform membership functions for output space the approximated curve is, in fact, the piecewise-linear function shown in Fig. 5.2. On the other hand, one might think of uniformly partitioning the input space, i.e. x axis, by standard membership functions and then specifying the desired membership functions for the output space. i.e. y axis. Figure 5.3 illustrates such an approach and again the piecewise lines are the approximated function.

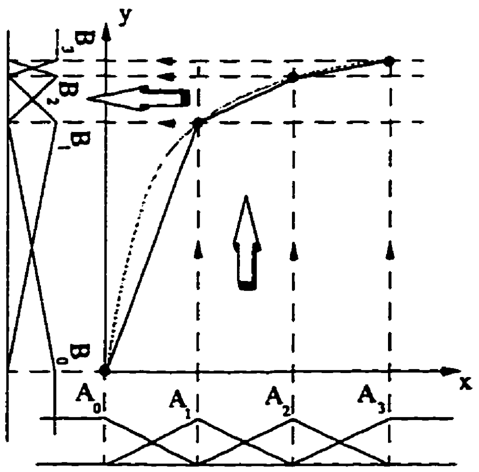


Figure 5.3: Approximated function by uniformly discretizing the output space.

The essence of this section is that the same approach, i.e. output partitioning, can be applied for our problem which involves discretization of a three dimensional space. In this case, the number of design parameters is decreased and a more efficient approach for the optimal design of fuzzy controllers can be achieved.

5.2.2 Input partitioning approach

In this approach, the parameters of the output space are assumed to be constant and uniformly distributed, while the optimal values of the input space are sought after.

In this case, the optimization problem should be able to find 45[1] parameters, and hence the optimization problem should search in a hyperspace of 45 dimensions. As shown in the following sections, while this is simpler than complete input-output optimization, it is far more complex than is the case for the output partitioning approach. Figure 5.4 demonstrates the required bit string for this approach.

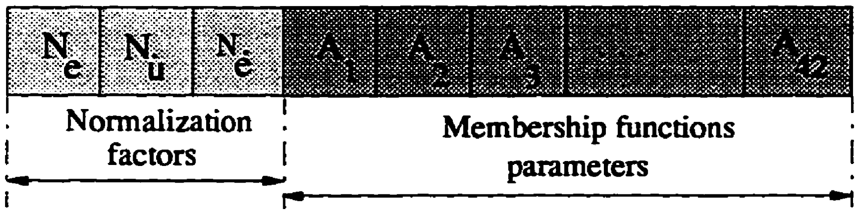


Figure 5.4: Bit-string for input partitioning.

5.2.3 Output partitioning approach

In this approach, for the sake of simplicity, the input space is uniformly partitioned and it is then only the consequent parts of rules as well as the normalization factors which are processed by the genetic algorithm. This, in effect, means the standard membership functions (cf. Fig. 2.4 on page 17) are chosen for the input space. Since the table cells are singletons, they do not require many parameters to be found. This decreases the number of design parameters of the controller. Figure 5.5 depicts the required string for this optimization approach.

[1] $3 + 42 = 45$

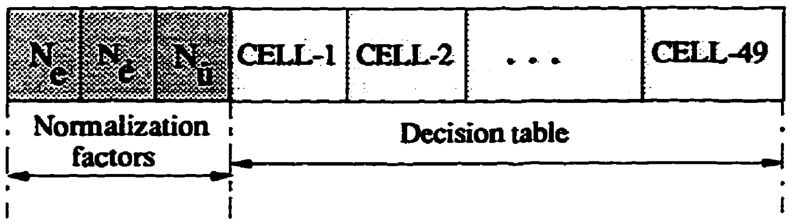


Figure 5.5: Bit-string for output partitioning.

However, as is shown later, if special considerations are taken into account, the optimization can be further simplified. Such simplification can be achieved using the sliding table approach as well as ouput partitioning which results in an efficient approach for the auto-design of a fuzzy controller. The detailed study of this technique and the corresponding simulation results are given in Section 5.4.

5.2.4 Input-output partitioning approach

While the output and input partitioning might give a satisfactory response, they are not, in general, strictly optimal. Consider again the previous example shown in Fig. 5.6. If the first and last points (points of A, D) are assumed to be constant, the function can be approximated by choosing two arbitrary points (B, C) at any arbitrary locations. This is not, however, the optimal approach. Therefore, the question which arises here is how to select the points B, C such that the best approximation of function can be achieved. In other words: which form of partitioning of the plane can lead to the optimal approach of the function? To answer this question, it is evident that the framework should be fixed neither along the horizontal axis nor along the vertical axis. It should be noted that in the case of output partitioning, vertical frames are fixed and the horizontal frames are free to scan the plane and in the case of input partitioning it is vice versa. In fact, in output partitioning the optimal algorithm should be able to find b', c' on

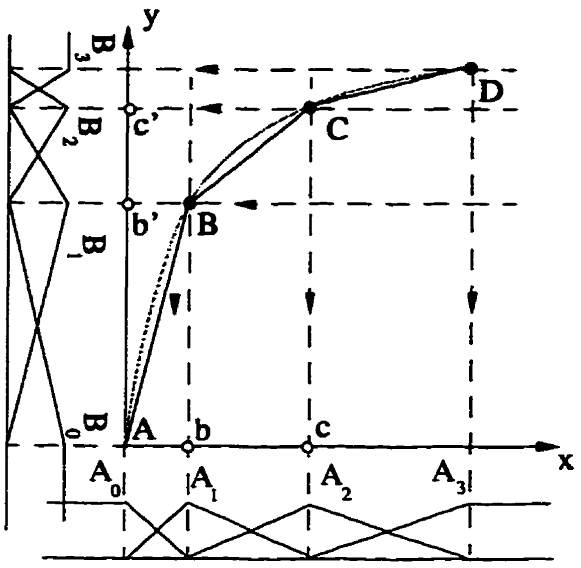


Figure 5.6: Input-ouput partitioning.

y axis and point b, c on the x axis in Fig. 5.6.

The origin of the problem comes from the fact that in the discretizing process of a fuzzy controller, first one should define the input output membership functions i.e. constructing a framework such as Fig. 5.6, and the values that are responsible for connection of different vertices. Based on this insight, it is quite evident that if the construction of such a framework is not appropriate, a good approximation cannot be achieved even if the rules are completely perfect. In fact, the rules are responsible for increasing or decreasing the input to output function. It is the width of each frame (i.e. membership functions shapes) which determines the rate of change of the input to output function. This is the idea underlying the concurrent optimization of input and output parameters of a fuzzy controller. Based on this approach the entire set of parameters of a fuzzy controller should be processed by a genetic algorithm.

5.3 Robustness of a fuzzy controller

5.3.1 Introduction

Robustness is an issue that should be addressed in the design of a controller where the system is subject to parameter variations or load disturbances. Although it is quite well known that the fuzzy controller is very robust in nature [16,60], its robustness has mostly been addressed by either empirical studies or simulation results [14,61] There are also a few cases where this issue has been tackled by a pure mathematical approach for some specific fuzzy operators [62]. While these approaches are effective to some extent, they do not sufficiently address the underlying concepts of fuzzy controller robustness. In what follows, it is shown that fuzzy controllers can be viewed as a class of variable structure controllers. This perspective, in turn, leads to a novel view that a conventional fuzzy controller may be designed such that it possesses a sliding mode. To attain this goal, some particular conditions should be satisfied.

To this end, a brief introduction to variable structure control is first presented.

5.3.2 Variable structure control

Variable structure control systems are a class of non-linear feedback control systems whose structure changes depending upon the state of the system. Hence, there are different structures in different regions of state space. This control system has its roots in relay and bang-bang control theory.

To illustrate the fundamentals of the variable structure control approach, consider a single input non-linear dynamic system which can be represented by the following state equations [63]:

$$\dot{x} = f(x,t) + g(x,t)u \tag{5.4}$$

and

$$\begin{array}{rcl} u = g_1(x,t) & \text{if} & x \in \Omega_1 \\ u = g_2(x,t) & \text{if} & x \in \Omega_2 \\ \vdots & \text{if} & \vdots \\ u = g_m(x,t) & \text{if} & x \in \Omega_m \end{array}$$

where $x \in R^n$, $u \in R$, Ω_i are mutually exclusive regions of the state space, $g(x,t)$ is the control action in region Ω_i. It is quite clear that the union of all regions should be equal to the entire state space, i.e.

$$\Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_m = R^n \tag{5.5}$$

The design of a sliding mode variable structure controller consists of the following steps.

Step-1: Design a switching surface S in state space to represent the desired dynamics for the system. S can be defined as

$$S = \{x \in R^n | s(x) = 0\} \tag{5.6}$$

The interesting point of this surface is that it has a lower order than the original plant, and in the case of $n = 2$, it becomes a switching line in two dimensions as illustrated in Fig. 5.7. As its name implies, once the state trajectory of the system is above the line, the controller has one gain and another gain (of opposite polarity) if the trajectory drops below the line.

Step-2: Design a variable structure control with two different regions as follows.

Let

$$u(t) = \begin{cases} u^+(x) & \text{if} \quad s(x) > 0 \\ u^-(x) & \text{if} \quad s(x) < 0 \end{cases} \tag{5.7}$$

such that the system state x can reach the switching surface from any initial state in state space in a finite time. The interesting point is that while neither of these structures is necessarily stable, their combined system results in a stable sliding mode. Once on the switching surface, the sliding mode takes place and the system state is pushed toward the origin (equilibrium point) following the switching surface. This makes the system, on the one hand, globally asymptotically stable [64], and on the other hand, insensitive to parameter perturbations and external disturbances which consequently leads to a highly robust control approach.

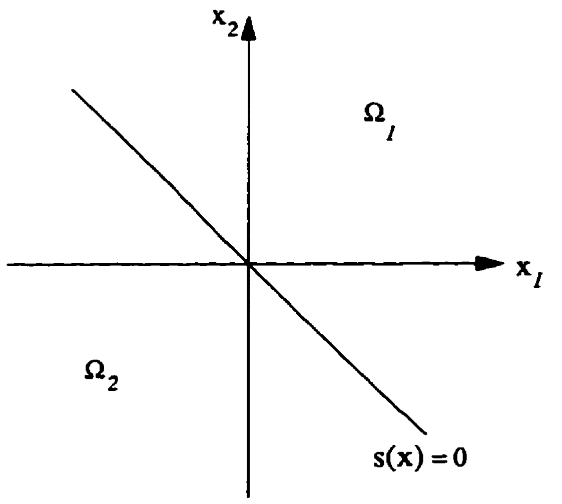


Figure 5.7: Switching surface in two dimensional state space.

There are two different modes in variable structure control; *reaching mode* and *sliding mode*. To determine the dynamics of a system in the reaching mode, the dynamics of the switching function s(x) should be defined. If the switching function is represented by

$$\dot{s}(x) = -K sign(s) \quad ; \quad K > 0 \tag{5.8}$$

then the reachability condition is already satisfied since the equation inherently has the property of $s(x)\dot{s}(x) < 0$[2]. In fact, equation 5.7 is the control law for reachability that results from the above condition.

While the sliding mode control approach has the benefits of stability, desired performance, and robustness, it has two drawbacks [64]. First, the insensitivity property of a variable structure control system is present only when the system is in the sliding mode. In other words, the state trajectory starting away from the sliding surface remains sensitive to parameter variations and external disturbances. Second, smaller values of control gain increase the reaching time while reducing the chattering and vice versa for larger gain. Thus, there is a trade off between reaching time and the chattering problem. Chattering is not acceptable for control systems since it causes significant changes in the control action. Furthermore, it may excite some unmodelled high frequency dynamics present in system.

The assumption that switching from one control action to the other be infinitely fast is not realistic. This is due to the finite time delay for control computation and to the limitations of physical actuators. In effect, the fact that the control action cannot be changed very fast, may also lead to chattering.

[2]In effect, if the Lyaponov function is considered as $0.5\, s^2(x,t)$, which is globally positive definite. For the stability condition, it is sufficient that its derivate becomes negative, i.e. $s(x)\dot{s}(x) < 0$.

5.3.3 Another view of the conventional fuzzy controller

A closer look at the structure of a fuzzy controller shows that for a large class of non-linear systems, these controllers are designed based on fuzzy partitioning of the state space. In effect, the decision table shown in Fig. 5.8 can be viewed as a partitioned state space with respect to the state variables e, $\dot{e}$. Since the controller output, in general, can change from one cell to another, one might think of the fuzzy controller as a ***state dependent controller*** whose control law is a function of the system states.

This implies that, in essence, the fuzzy controller has a variable structure nature, which in turn implies that fuzzy controllers can be considered a particular class of variable structure controllers. Furthermore, as shown in Fig. 5.9, the diagonal axis of the decision table can potentially be seen as the sliding surface of $s(x) = 0$ (cf. equation 5.6, in which 0 is a fuzzy number). In other words, the surface of $s(x) = 0$ in the fuzzy controller becomes a *fuzzy sliding surface* as illustrated in Fig. 5.10.

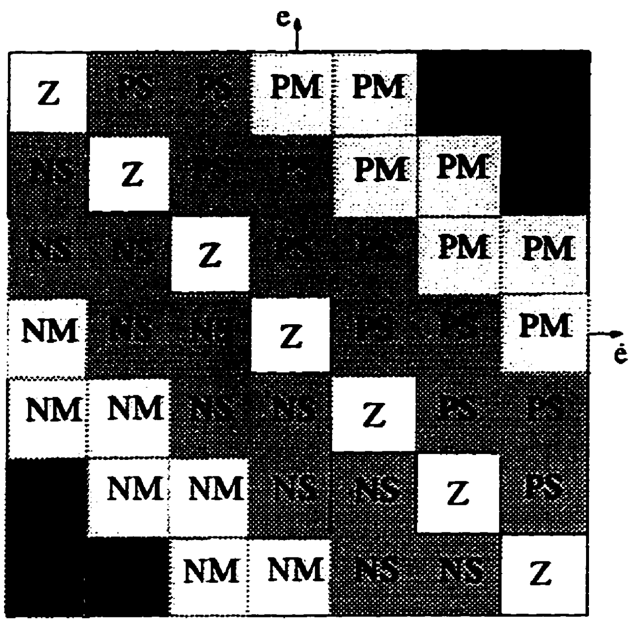
Mathematically, the existence of the sliding surface for the table of Fig. 5.8. can be verified by the existence condition

$$\lim_{s\to 0} s(x)\dot{s}(x) < 0. \tag{5.9}$$

In fact, as Fig. 5.9 demonstrates for every point in the region-1 while the value of s is positive, the value of $\dot{s}$ remains negative if a trajectory is approaching the sliding surface and vice versa for region-2. In other words,

$$\begin{array}{lll} \lim_{s\to 0} s(x)\dot{s}(x) = \lim_{s\to 0} s^{+}(x)\dot{s}^{-}(x) < 0 & \text{if} & (e,\dot{e}) \in \Omega_1 \\ \lim_{s\to 0} s(x)\dot{s}(x) = \lim_{s\to 0} s^{-}(x)\dot{s}^{+}(x) < 0 & \text{if} & (e,\dot{e}) \in \Omega_2. \end{array} \tag{5.10}$$

Equivalently, the control law of equation 5.7 can be easily seen in the decision table shown in Fig. 5.9.



Z	PS	PS	PM	PM		
NS	Z	PS	PS	PM	PM	
NS	NS	Z	PS	PS	PM	PM
NM	NS	NS	Z	PS	PS	PM
NM	NM	NS	NS	Z	PS	PS
	NM	NM	NS	NS	Z	PS
		NM	NM	NS	NS	Z

Figure 5.8: Variable structure of decision table

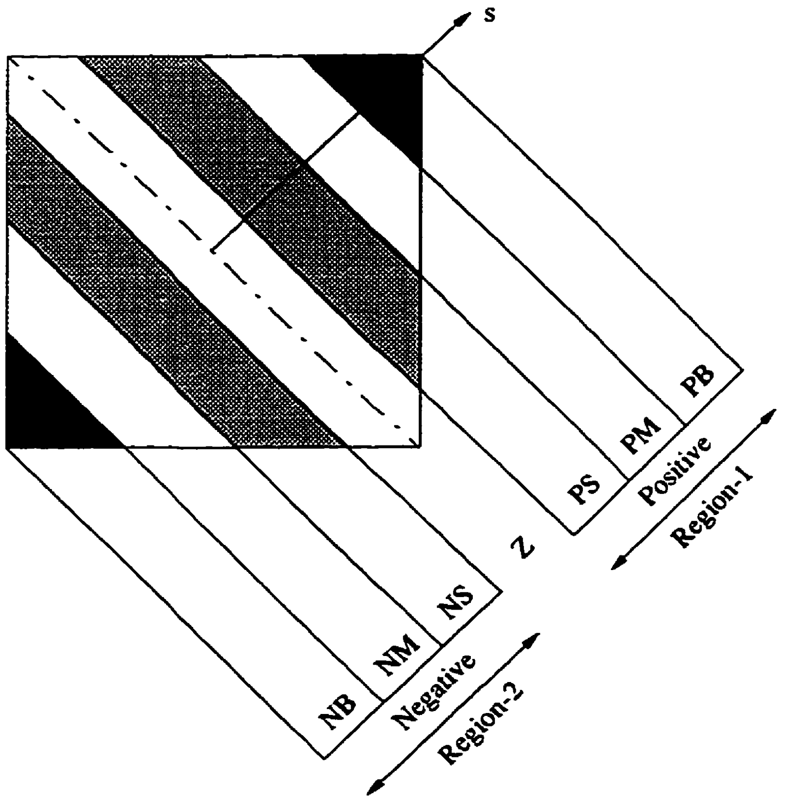


Figure 5.9: Two distinct structures in the decision table

As a result, a fuzzy controller with the table shown in Fig. 5.8 possesses a variable structure nature and if proper values for normalization factors are chosen, a sliding motion is achieved in which the switching surface is a fuzzy surface, (see Fig. 5.10).

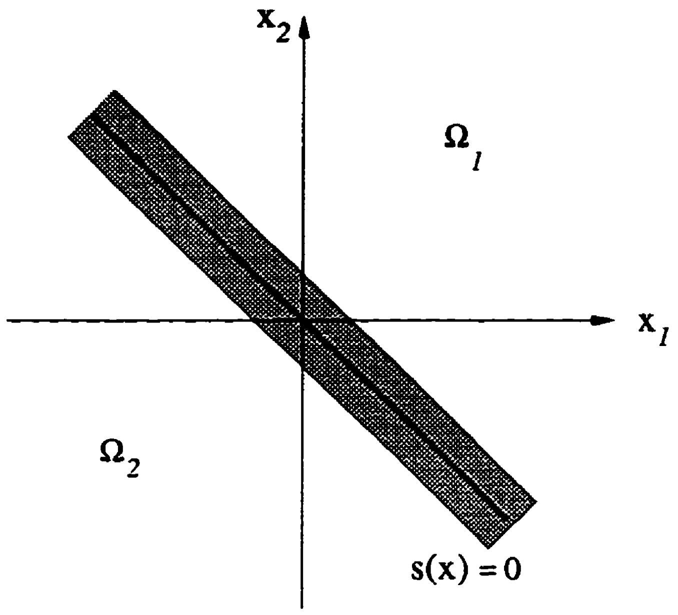


Figure 5.10: Fuzzy sliding surface in a two dimensional state space.

The slope of the sliding surface can be adjusted by changing normalization factors, $N_e, N_{\dot{e}}$. In fact, the higher N_e and smaller $N_{\dot{e}}$ is, the larger is the slope of the sliding surface.

Since the proper design of the sliding surface is crucial to the success of sliding mode control, finding the best value of the normalization factors is essential to the controller design. Figure 5.11 illustrates how the sliding surface is changed with the different values of the normalization factors.

Figure 5.12 shows a typical operating line for a sliding mode controller having upper and lower bounds, compared to the fuzzy controller counterpart. While for the sliding mode controller the operating line is linear, for the fuzzy controller it appears to be piecewise linear whose slope and shape

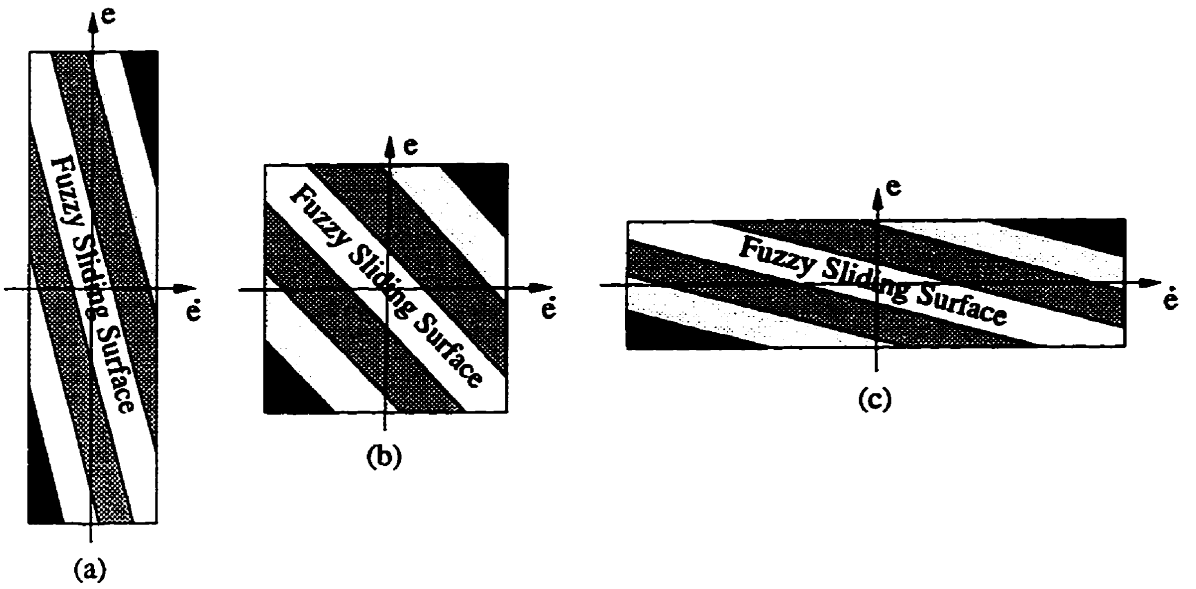


Figure 5.11: Designing different fuzzy sliding surfaces based on normalization factors; (a) large N_e, small $N_{\dot{e}}$; (b) moderate N_e, moderate $N_{\dot{e}}$; small N_e, large $N_{\dot{e}}$

strongly depend upon the number and shape of the membership functions. Therefore, specifying an optimal operating line will lead to the search for optimal membership functions.

To this end, a fuzzy controller with the decision table of Fig. 5.8, can be viewed as a sliding mode controller whose sliding surface and operating line should be designed by proper selection of normalization factors, N_e, $N_{\dot{e}}$, and membership functions. For this class of fuzzy controllers, the robustness of the controller has its basis in sliding mode control and furthermore, the proposed auto-design approach can be employed for finding the optimal sliding surface and operating line of the controller. In this case, the table which is processed by the optimization technique should sustain the feature that the sign of the controller output does not change on each side of the sliding surface. This implies that while our intention is the optimal design of the fuzzy controller, a search is performed for an optimal sliding mode controller. If this happens, a new type of controller which we call a *fuzzy controller with*

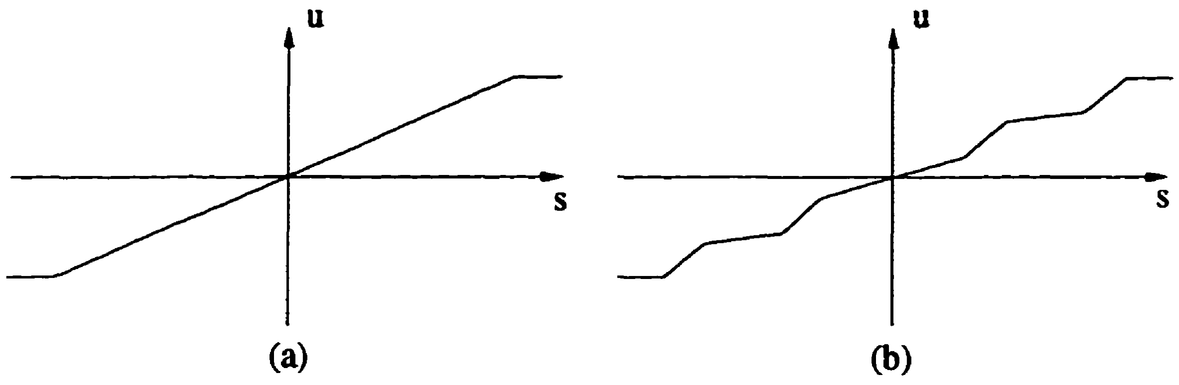


Figure 5.12: Operating lines for (a) sliding mode controller with boundary layer (b) special type of fuzzy controller.

sliding mode can be achieved.

The interesting point is that at a large distance from the sliding surface, any modelled frequency dynamics are not able to cause a change in the sign of controller output. Therefore, the decision table can be adjusted in such a way that the further the system state is from the sliding surface, the larger the controller gain. This equivalently means better dynamics in the reaching mode and at the same time, a smooth transition from one structure to the other. This resolves the trade-off problem for the reaching time and the chattering effect which is a major drawback in the conventional sliding mode control scheme.

5.4 A novel approach for the efficient design of a fuzzy controller

A closer look at the sliding table shown in Fig. 5.8 reveals that with uniformly partitioning of the output space, the fuzzy controller of Fig. 5.13 demonstrates a linear behavior. This can simply be seen in the three-dimensional space as illustrated in Fig. 5.14 where $\dot{u}$ is the controller output and e and

$\dot{e}$ are the controller inputs. However, in general, for a non-linear plant, a

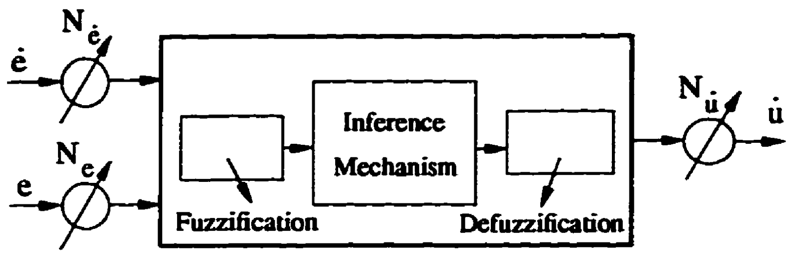


Figure 5.13: Basic structure of the fuzzy logic controller.

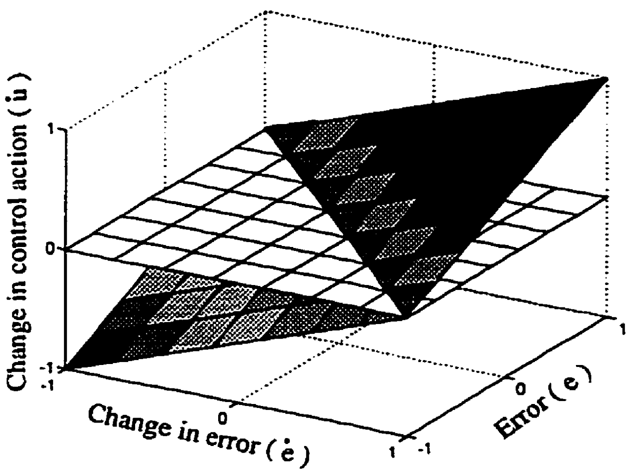


Figure 5.14: Control surface for the conventional fuzzy controller.

non-linear controller is required to obtain the desired performance. Such a non-linear controller, for instance, is shown in Fig. 5.15 in a three dimensional space. To construct a fuzzy controller with non-linear behavior, one may think of changing only the output singletons, while the input membership functions are assumed constant. If this happens, there will be no need for processing the membership function parameters and hence the number of design parameters will be reduced to a large extent. Furthermore, if the sliding structure of this table, as shown in Fig. 5.8, is sustained, then one can view the whole table as the string illustrated in Fig. 5.16.

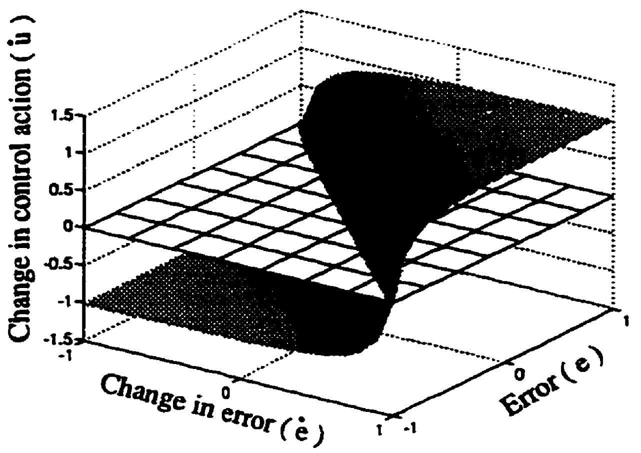


Figure 5.15: A non-linear control surface.

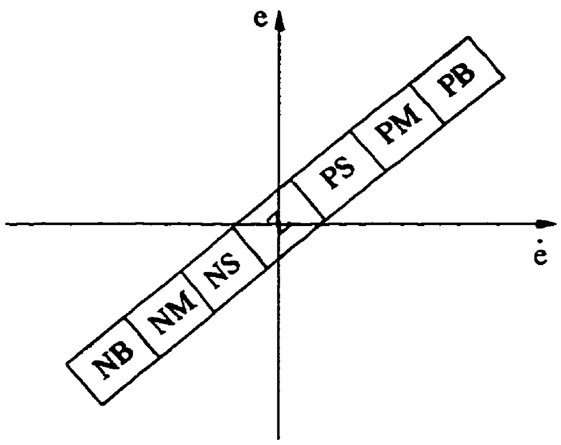


Figure 5.16: A new view to the sliding table.

Viewed in this perspective, a string of seven parameters can model the entire decision table . Therefore, together with the normalization factors, N_e, $N_{\dot{e}}$, $N_{\dot{u}}$, a string with ten parameters would be sufficient for the construction of such a non-linear surface (see Fig. 5.17). It is now the responsibility of the optimizer to select these parameters based on a particular performance index. It is interesting to note that the string can be further simplified if one can incorporate the following knowledge prior to the design. First. in steady state conditions, i.e. when error and error derivative are zero, the value of the

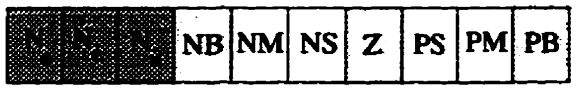


Figure 5.17: A new string for an optimal design of a fuzzy controller.

controller output, $\dot{u}$, must be zero. Second, since the input-output space has been normalized, the value of the two extremes of the output space, i.e. PB and NB, should be 1 and -1, respectively. These two facts imply that the

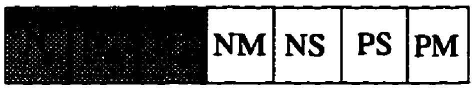


Figure 5.18: The proposed string for an optimal design of a fuzzy controller.

design of this class of fuzzy controllers can be further simplified to a string of $3+4 = 7$ parameters (see Fig. 5.18) where the singletons of NM, NS, PS, and PM are free parameters to be found based on the optimization algorithm.

It is worthwhile to notice that while the resultant decision table shares the same structure with the conventional one, it has a different interpretation in reference to the output singletons. For instance, in the proposed approach, the value of singletons PS, PM are completely free and it is the optimizer that determines their values. In contrast, in the conventional table, the above singletons have constant values, say 0.333, 0.667 respectively.

Of course, such a simplification is achieved at the expense of rough approximation of the non-linear function. The point, however, we would like to clarify here is that, while the optimization problem is simplified to a large extent, the performance index decreases only slightly.

Furthermore, the proposed technique can be viewed as an alternative approach for the full-optimization described in Section 3.4.4 and it can be applied where the fast tuning of a fuzzy controller is the primary concern.

It is also instructive to mention that, while a genetic algorithm is used for the proposed efficient tuning approach, other optimization techniques can be employed.

The proposed efficient tuning approach is applied for the non-linear system defined in 5.1. For every parameter of the string shown in Fig. 5.18, a resolution of 10 bits is assigned.

The control structure is found after 45 generations with a population of 30 individuals (see Table 5.1). The input membership functions have the standard form as shown in Fig. 2.4 in Section 2.3.2. The step response of such a controller is depicted in Fig. 5.19.

Number of individuals	30
Percentage of crossover	90%
Randomly generated individuals	10%
Max Percentage of mutation	2%
Number of generations	45
Number of design parameters	3
Bit-string length	30

Table 5.1: Case 3: Free parameters of a genetic algorithm

	Auto-Design FLC	Conventional FLC
N_e	4.58	8.07
$N_{\dot{e}}$	0.55	3.97
$N_{\dot{u}}$	10.05	6.60

Table 5.2: Case 3: Normalization factors found by a genetic algorithm.

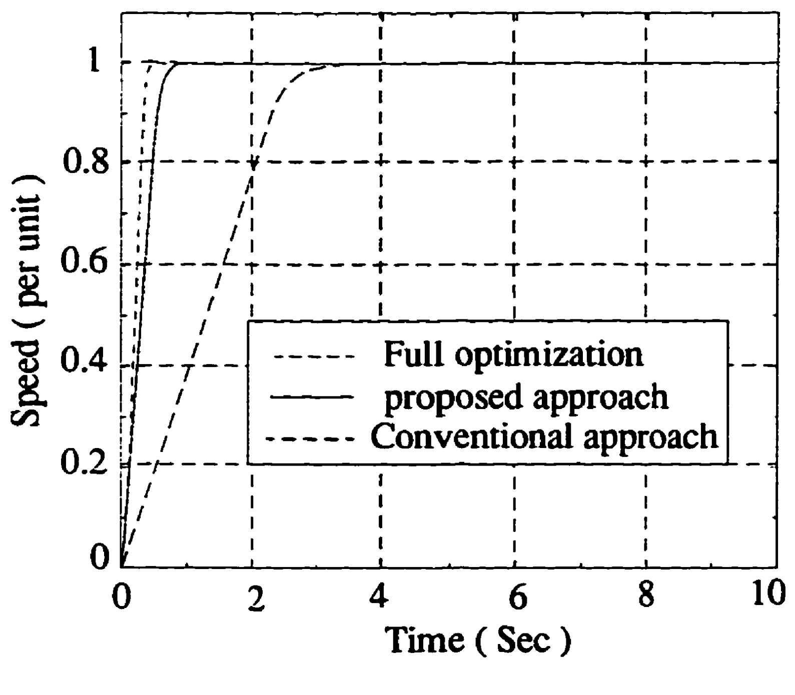


Figure 5.19: Step responses of the non-linear system for different approaches.

5.5 Chapter summary

In this chapter an efficient approach for the auto-design of a fuzzy controller is proposed. It has been shown that such an approach can efficiently lead to a near-optimal solution.

In addition to these techniques, some other aspects of fuzzy controller auto-design are discussed. First, the input / ouput partitioning approaches are proposed by which the number of controller design parameters can be decreased. While in either approach, i.e. input partitioning or output partitioning, one set of parameters is kept constant, the other set is processed by the optimizer concurrently. This indicates that even in these cases the essence of simultaneous design of the controller parameters is sustained. Next, a novel view of the conventional table has been proposed by which the reachability condition of this table is satisfied. It then follows that such a view can potentially lead to the design of a fuzzy controller based on the sliding mode concept whose performance and robustness is ensured mathematically. This view is achieved by categorizing fuzzy controllers as a particular class of variable structure controllers.

Chapter 6

Conclusions and recommendations for future research

6.1 Conclusions

Fuzzy control has been found to be much more interesting when applied to non-linear, uncertain systems. Many unique features such as non-linear capability, domain-wise mapping, and robustness make this type of controller very attractive for a wide variety of applications.

Although fuzzy control was originally introduced by L. A. Zadeh, it was Mamdani who first practically applied this control technique to a real plant in 1972. Since then, a great deal of effort has been devoted to further develop this control approach.

The wide applications of fuzzy control can be viewed as a natural consequence of the *universal approximation theorem.* Based on this theorem, any non-linear, continuous function can be approximated by a fuzzy system to any desired precision. However, while quite significant. this theorem does not indicate how to develop such a fuzzy system. As a systematic design technique is lacking, fuzzy controllers to this point have been designed by human trail-and-error. The existing trial-and-error approaches require a large number of iterations without the guarantee of an optimal solution. Furthermore, if the number of controller inputs and outputs increases, a trial-and-error approach may be so tedious as to be unfeasible. *Adaptive fuzzy systems*, in general, and *artificial neural networks*, in particular, are the systems which use a learning algorithm to train a fuzzy controller for a specific task based on available input-output data. The key point. however. is that these approaches require a well-designed reference controller. *a priori*. which may not be available. Moreover, only a small part of the fuzzy controller is designed by these approaches.

The essence of this dissertation involves the synthesis of a new design methodology for fuzzy controllers *without* the requirement for any input-output training data. This is achieved using a genetic algorithm as the optimization technique, which employs a predefined performance index to guide its search.

To verify the proposed technique, as just one example, an induction motor drive with field oriented control has been chosen where the performance index is defined for the best dynamic and steady state response. Based on the proposed approach, an optimal fuzzy controller has been designed for such a system in Chapter 4. Furthermore, a novel view of the field oriented control of ac machines has also been proposed in the same chapter.

Since the proposed approach is very general, a non-linear, uncertain system has been considered in Chapter 5. Different aspects of this approach, such as sequential versus concurrent optimization and input partitioning versus output partitioning have been investigated. In the sequential approach, first the normalization factors are optimized while the membership functions and decision table are kept constant. Once the optimization algorithm finds the best values for the normalization factors, these parameters are fixed while the membership function parameters are processed by the optimization technique. Finally, the consequent parts of the decision table are optimized while the optimal values of normalization factors and membership functions are used. As can be inferred, this sequential technique ignores the interdependency between different sets of parameters.

In Chapter 5, a novel view of the robust design of a fuzzy controller is presented which facilitates implementation of sliding mode control by a fuzzy controller. Full optimization of a fuzzy controller involves searching for a large number of parameters. A novel alternative approach for the design of fuzzy controllers is presented in Chapter 5 which facilitates optimization with an order of magnitude fewer parameters. With this approach, the system performance is only decreased slightly.

The contributions and achievements of this dissertation can be summarized as:

- A novel view of the fuzzy controller has been proposed in Chapter 2 including the definition of characteristic points, a key concept which helps to define the role of the different parameters of a fuzzy controller.

- A new coding for a fuzzy controller has been presented in Chapter 3 based on asymmetrical membership functions, with complete freedom of overlap. This coding facilitates optimal and near-optimal design of a fuzzy controller.

- A novel perspective on field oriented control has been proposed in Chapter 4 employing differential geometry. Also in this chapter, the design of an optimal fuzzy controller for an induction motor drive with field oriented control is presented.

- An in depth investigation of the different aspects of the proposed technique is carried out in Chapter 5. This includes input-output partitioning approaches and the robustness issue.
- A novel approach to the optimization of a particular, but very common, class of fuzzy controllers is presented resulting in an efficient optimization process.

The main contribution of this dissertation is the concurrent design of fuzzy controllers based on a new coding presented in Chapter 3.

In short, this research work can be viewed as a departure from the uncertainty and complexities involved in the conventional trial-and-error method of fuzzy controller design. It results in auto-design approaches for the development of fuzzy controllers in two different manners; a full-optimization and an efficient design approach.

6.2 Recommendations for future research

To extend the current work, further research can be carried out which would comprise:

- Designing optimal controllers for none minimum phase systems. In fact, a fuzzy controller which works well for a minimum phase plant, does not necessarily control a non-minimum phase plant. Special considerations should be taken into account at the design stage to enable a fuzzy controller to handle such a system.
- Hardware implementation of the optimal fuzzy controller for induction motor drives.
- An investigation of the possible on-line adaptation of a fuzzy controller based on the proposed efficient design approach.

References

[1] L. A. Zadeh, "fuzzy sets", *Information Control*, vol. 8, pp. 338–353, 1965.

[2] J. Cleland, W. Turner, and et. al. B. Bose, "Fuzzy logic control of ac induction motors", in *Proc. of IEEE Conf. on Fuzzy Systems*, 1992, pp. 843–850.

[3] R. D. Lorenz, T. A. Lipo, and D. W. Novotny, "Motion control with induction motors", *Proc. of the IEEE*, vol. 82, no. 8, pp. 1215–1240, August 1994.

[4] F. Blaschke, "The principle of field orientation as applied to the new TRANSVECTOR closed-loop control system for rotating-field machines", *Siemens Review*, , no. 5, pp. 217–223, 1972.

[5] B. Lin and R. G. Hoft, "Neural networks and fuzzy logic in power electronics", *Cont. Eng. Practice*, vol. 2, no. 1, pp. 113–121, 1994.

[6] C. Won, S., and B. K. Bose, "Robust position control for induction motor using fuzzy logic control", in *Proc. of IEEE Conf. on Fuzzy Systems*, 1992, pp. 472–481.

[7] H. Hellendoorn, "Design and development of fuzzy system at siemens R&D", in *Proc. of IEEE Conf. on Fuzzy Systems*, 1993, pp. 1365–1370.

[8] B. da Silva, G. E. April, and G. Oliver, "Real time fuzzy adaptive controller for an asymmetrical four quadrant power converter", *proc. of IEEE IAS Ann. Meeting*, pp. 872–878, 1987.

[9] K. Dong, "Control of permanent magnet ac servo motors via fuzzy reasoning", *Proc. of IEEE IAS Conf.*, vol. 40, no. 1, pp. 482–489, 1992.

[10] H. R. Azevedo and K. P. Wong, "A fuzzy logic controller for permanent magnet synchronous machine", *Proc. of APEC*, pp. 122–127, 1993.

[11] B. R. Lin, "Analysis of fuzzy control method applied to dc-dc converter control", *Proc. of APEC*, March 1993.

[12] W. Hwang and W. E. Thompson, "An improved method for designing fuzzy controller for position control systems.", *Proc. of IEEE Conf. on Fuzzy Systems*, pp. 1361–1364, 1993.

[13] F. F. Cheng and S. N. Yeh, "Application of fuzzy logic in the speed control of ac servo system and an intelligent inverter", *IEEE Trans. Energy Conversion*, vol. 8, no. 2, pp. 312–318, June 1993.

[14] T. Zhao and T. Virvalo, "Fuzzy control of a hydraulic position servo with unknown load", in *Proc. of IEEE Conf. on Fuzzy Systems*, 1993, pp. 785–788.

[15] Y. Dote, "Fuzzy and neural network controller", *IEEE IECON*, pp. 1314–1343, 1990.

[16] Y. F. Lee and C. C. Lau, "Development of fuzzy algorithms for servo systems", *IEEE Control System Mag.*, April 1989.

[17] S. A. Mir, M. E. Elbuluk, and D. S. Zinger, "Fuzzy implementation of direct self control of induction machines", *IEEE Trans. Ind. Appl.*, vol. 30, no. 3, May/June 1994.

[18] S. A. Mir, M. E. Elbuluk, and D. S. Zinger, "Fuzzy controller for inverter fed induction machines", in *Proc. of IEEE IAS Conf.*, 1992, pp. 464–471.

[19] J. Cleland, W. Turner, and B. K. Bose, "Fuzzy logic control of induction motors", in *Proc. of IEEE IAS Conf.*, 1992, pp. 843–850.

[20] H. Liang and H. Chen, "A new-variable- speed system with fuzzy identification and fuzzy control", in *Proc. of International Power Eng. conf., Singapore*, 1993, pp. 749–754.

[21] K. Liu and F. L. Lewis. "Some issues about fuzzy logic controller". *Proc. of IEEE 32rd Con. on Des. and con.*, pp. 1743–1748, 1993.

[22] D. E. Schneider, P. P. Wang, and M. Toga, "Design of a fuzzy logic controller for a target tracking system", in *Proc. of IEEE Conf. on Fuzzy Systems*, 1992, pp. 1131–1138.

[23] S. Isaka and A. V. Sebald, "An optimization approach for fuzzy control design", *IEEE Trans. Sys. Man Cybern.*, vol. 22, no. 6, pp. 1469–1473, Nov./Dec. 1992.

[24] B. S. Zhang and J. M. Edmunds, "Self-organization fuzzy logic controller", *IEE Proc.s-D*, vol. 139, no. 5, pp. 460–464, Sep. 1992.

[25] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements", *IEEE Trans. Neural Networks*, vol. 3, no. 5, Sep. 1992.

[26] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—part i", *IEEE Trans. Man, Sys., and Cybernetics*, vol. 20, no. 2, pp. 404–418, March/April 1990.

[27] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—part ii", *IEEE Trans. Man, Sys., and Cybernetics*, vol. 20, no. 2, pp. 419–435, March/April 1990.

[28] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1975.

[29] E. Czogala and W. Pedrycz, "Control problems in fuzzy systems", *Fuzzy Sets and Systems*, vol. 26, no. 7, pp. 257–273, 1982.

[30] S. Shao, "Fuzzy self-organizing controller and its application for dynamic processes", *Fuzzy Sets and Systems*, vol. 7, no. 26, pp. 151–164, 1988.

[31] T. Hojo, T. Terano, and S. Masui, "Design of quasi-optimal fuzzy controller by fuzzy dynamic programming", in *Proc. of IEEE Conf. on Fuzzy Systems*, 1993, pp. 1253–1258.

[32] P. Ramaswamy, M. Riese, R. M. Edwards, and K. Y. Lee, "Two approaches for automating the tuning process of fuzzy logic controllers", in *Proc. of the 32nd conf. on Decision and Control*, 1993, pp. 1753–1758.

[33] J. Lee, "On methods for improving performance of pi-type fuzzy logic controller", *IEEE Trans. Fuzzy Systems*, vol. 1, no. 4, pp. 298–301, November 1993.

[34] H. X. Li and H. B. Gatland, "A new methodology for designing a fuzzy logic controller", *IEEE Trans. Systems*, vol. 25, no. 3, pp. 505, March 1995.

[35] K. Hirota and Y. Yoshinari, "Identification of fuzzy control rules based on fuzzy clustering method", in *Proc. of the 5th Fuzzy System Symposium*, 1989, pp. 253–258.

[36] R. Jang, "Self-learning fuzzy controllers based on temporal back propagation", *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 714–723, 1992.

[37] H. Takagi and I. Hayashi, "Nn-driven fuzzy reasoning", *Int'l J. Approximate Reasoning (special issue of IIZUKA '88)*, vol. 5, no. 3, pp. 191–212, 1991.

[38] R. Ichikawa, K. nishimura, and M. Kunugi, "Auto-tuning method of fuzzy membership functions using neural network learning algorithm", in *Proc. of the 2nd Int. Conf. on Fuzzy Logic and Neural networks(IIZUKA '92)*, 1992, pp. 345–348.

[39] L. X. Wang and M. J. Mendel, "Generating fuzzy rules by learning from examples", *IEEE Trans. Systems, Man and Cybernetic*, vol. 22, no. 6, 1992.

[40] B. Kosko. *Neural networks and fuzzy systems - A dynamical system approach to machine intelligence*. Printice hall, Englewood cliffs. NJ. 1992.

[41] D. Nauck, F. Klawoon, and R. Kruse, "Combining neural networks and fuzzy controllers", in *Proc. of the FLAI'93, Linz, Austria*, 1993.

[42] Li-Xin Wang, *Adaptive Fuzzy Systems and Control*, Printice hall, Englewood cliffs, NJ, 1994.

[43] E. Cox, "Adaptive fuzzy systems", *IEEE Spectrum*, pp. 27–31, Feb. 1993.

[44] P. J. Werbos, "Neurocontrol and elastic fuzzy logic: Capabilities, concepts, and applications", *IEEE Trans. Industrial Electronics*, vol. 40, no. 2, pp. 170–180, April 1993.

[45] C. Hung, "Building a neuro-fuzzy learning control system", *AI Expert*, pp. 40–49, Nov. 1993.

[46] C. T. Lin and C. S. George Lee, "Neural-network-based fuzzy logic control and decision system", *IEEE Trans. Computers*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.

[47] B. S. Zhang and J. M. Edmunds, "On fuzzy logic controllers", in *Proc. of IEE International Conf. on Control, Edinburgh, UK*, 1991, pp. 961–965.

[48] D. Whitley, "A genetic algorithm tutorial", *Technical Report CS-93-103*, Nov. 1993.

[49] S. Rajeev and C. S. Krishnamoorthy, "Discrete optimization of structures using genetic algorithms", *Journal of Structural Engineering*, vol. 118, no. 5, pp. 1233–1239, May 1992.

[50] I. P. Androulakis and V. Venkatasubramanian, "A genetic algorithm framework for process design and optimization", *Computer Chem. Eng.*, vol. 15, no. 4, pp. 217–228, 1991.

[51] D. Beasly, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part ii, research topics", *University Computing*, vol. 4, no. 15, pp. 170–181, 1993.

[52] D. Beasly, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part i, fundamentals", *University Computing*, vol. 2, no. 15, pp. 58–69, 1993.

[53] J. F. Frenzel, "genetic algorithms", *IEEE Potentials*, pp. 21–24, Oct. 1993.

[54] F. Guely and P. Siarry, "Gradient descent method for optimizing various fuzzy rule bases", *IEEE Trans. Fuzzy Systems*, pp. 1241–1246, 1993.

[55] M. Sugeno, "An introductory survey of fuzzy control", *Inform Sci.*, vol. 36, pp. 59–83, Aug. 1985.

[56] C. Kravaris and c. Chung, "Nonlinear state feedback synthesis by global input / output linearization", *AICHE Journal*, vol. 33, no. 4, pp. 592–603, 1987.

[57] P. C. Krause, O. Wasynczuk, and S. D. Sudhoff, *Analysis of Electric Machinary*, IEEE Press, Piscataway, NJ, 1995.

[58] F. Ashrafzadeh. E. P. Nowicki, and J. C. Salmon. "The automatic design and optimization of fuzzy logic controllers based on the genetic algorithm", in *Proc. of ANNIE Conf.*, 1995, pp. 537–542.

[59] F. Ashrafzadeh, E. P. Nowicki, and J. C. Salmon, "A self organizing and self tuning fuzzy logic controller for field oriented control of induction motor drives". in *Proc. of IEEE IAS Conf.*, 1995, pp. 1656–1662.

[60] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*, John Wiley & Sons Inc., New York, NY10158, USA, 1993.

[61] H. T. Nguyen, "An empirical study of robustness of fuzzy systems", in *Proc. of IEEE Conf. Fuzzy Systems*, 1993, pp. 1340–1345.

[62] H. T. Nguyen, V. Kreinovich, and D. Tolbert et al, "On robustness of fuzzy logics". in *Proc. of IEEE Fuzzy Systems Conf.*, 1993, pp. 543–548.

[63] Y. Hwang and M. Tomizuka, "Fuzzy smoothing algorithms for variable structure systems", *Trans. Fuzzy SYSTEMS*, vol. 2, no. 4, pp. 277–284, Nov. 1994.

[64] M. B. Ghalia and A. T. Alouani, "Sliding mode control synthesis using fuzzy logic", in *Proc. of American control Conf.*, 1995, pp. 1528–1532.

[65] B. K. Bose, *Power Electronics and AC Drives*, Printice-Hall, Englewood Cliffs, NJ, 1986.

[66] G. T. Kim, K. S. Kim, M. H. Park, C. Y. Won, and D. S. Ahn, "Time optimal control for indution motor servo system", in *Proc. of IEEE PESC Conf.*, 1988, pp. 1053–1062.

Glossary

Fuzzy Logic:

Artificial intelligence:

"...the study of how to make computers to do things at which, at the moment, people are better"—Elaine Rich (1988)

Antecendent:

The initial (or *if*) part of a fuzzy rule.

Consequent:

The final (or *then*) part of a fuzzy rule.

Crisp value:

The point-wise, i.e. normal, value of a variable.

Defuzzification:

The process of transforming a fuzzy output of a fuzzy inference system into a crisp output.

Degree of membership:

The output of a membership function. This value is always limited to between 0 and 1. Also known as a membership value.

Fuzzification:

The process of generating membership values for a fuzzy variable using membership functions.

Fuzzy inference system

The overall name of a system that uses fuzzy reasoning to map an input space to an output space.

Fuzzy set:

A set which contain elements with full or partial degree of membership.

Linguistic variable:

A variable whose values are words or sentences in a natural or artificial language. For instance, age is a linguistic variable if its values are linguistic rather than numerical, i.e. *young, not young, very young, quite young, old, not old, and not very young*, etc., rather than 20, 21, 22, 23, ...

Mamdani-type inference:

A type of fuzzy inference in which the output fuzzy sets are not singletons and they are combined to yield a complex fuzzy set. This resultant fuzzy set should then be defuzzified to generate the crisp output of the fuzzy system.

Membership function:

A function that specifies the degree to which a given input belongs to a set.

Singleton:

A fuzzy set with a membership function that is unity at a one particular point and zero everywhere else.

Sugeno-type inference:

A type of fuzzy inference in which the consequent of each rule is a singleton, in general, a linear combination of inputs. The crisp output then becomes a weighted linear combination of the consequents.

Universe of discourse:

The domain on which the fuzzy sets are defined.

Genetic Algorithms:

Chromosome:

A data-structure which holds a string of parameters (or genes). This may be stored, for instance, as a binary bit-string, or an array of integers.

Crossover:

A reproduction operator which forms a new chromosome by combining parts of each of two parents. The simplest is a single-point crossover, in which an arbitrary point in the chromosome is chosen. All the information from one parent is copied from the starting point to the crossover point, while the all the information from the other parent is compiled from the cross point to the end point of the chromosome. In this way, the new chromosome gets the *head* of one parent's chromosome combined with the *tail* of the other. Other definitions exist which use more than one crossover point, or even combine the information from parents in other ways.

Evolution:

The process by which a set of possible solutions (or population of individuals) for a problem improves with each iteration, or so called generation).

Fitness:

A value assigned to an individual which reflects how well an individual solves a specific task. A fitness function is employed to map a chromosome to a fitness function.

Fitness landscape:

The hyper surface obtained by applying a fitness function to every point of the search space.

Function optimization:

The task of finding the set of parameters which produce the maximum or minimum value of a function.

Gene:

A subsection of a chromosome which usually encodes the value of a single parameter.

Generation:

An iteration of the measurements of fitness and the creation of a new population by means recombination operators.

Genetic algorithm:

A model of machine learning that uses a genetic metaphor from nature. It usually employs a string to represent their information, together with a population of individuals which is processed by crossover and mutation operations in order to find the interesting regions of the search space.

Genetic operator:

A search operator acting on a coding structure.

Global optimization:

The process by which a search is made for the extremum of a function. In a genetic algorithm, this extremum corresponds to the fitness function that is used to assess the performance of any individual.

Individual:

A possible solution to the task being tacked, i.e. a single point in the search space. Every individual contains a chromosome and some other information such as fitness.

Mating pool:

The whole set individuals ready for recombination, also called population.

Mutation:

A recombination operator which forms a new chromosome by making changes to the values of genes in a copy of a single parent.

Offspring:

An individual generated by the process of recombination.

Optimization:

The process of iteratively improving the solution to a problem with respect to a specific objective function.

Parent:

An individual which takes part in recombination to generate one or more other individuals, known as offspring.

Population:

A group of individuals which may interact together, for instance by mating, to produce offspring.

Recombination:

The creation of a new individual from two parents. It involves the genetic operators such as crossover and mutation.

Reproduction:

The duplication process of a current generation for the selection of parents.

Search space:

If the solution of a task can be represented by a set of n real-valued parameters, then the job of finding this solution may be thought of as a search in an n-dimensional space. This is referred as the search space.

Selection:

The process by which some individuals in a population are chosen for recombination.

Vector optimization:

An optimization problem wherein multiple objectives must be satisfied.

Appendix A

Proof of non-linearity of fuzzy controllers

To prove a system is linear, two following properties must be satisfied simultaneously:

1. **Additivity property** or superposition property, i.e. if

   $$y_1 = f(x_1) \qquad \text{and} \qquad y_2 = f(x_2) \tag{A.1}$$

   Then the additivity property requires

   $$y_1 + y_2 = f(x_1 + x_2) \tag{A.2}$$

   Hence

   $$f(x_1) + f(x_2) = f(x_1 + x_2) \tag{A.3}$$

2. **Scaling property**, i.e. if assume

   $$y = f(x) \tag{A.4}$$

   then the scaling property requires that for any real constant α

   $$\alpha . y = f(\alpha . x) \tag{A.5}$$
   $$\alpha . f(x) = f(\alpha . x) \tag{A.6}$$

Every system which does not satisfy both properties is a non-linear system. In order to check the non-linearity of a fuzzy controller, consider the following fuzzy rule:

If x is LV, then u is LU

where LV and LU are the linguistic values taken on by the process state variable x and the control output variable u, respectively. The meaning of these two linguistic values is given by the membership functions $\mu_{LX} : X \rightarrow [0,1]$ and $\mu_{LU} : U \rightarrow [0,1]$. Furthermore, let x_1 and x_2 be two crisp inputs and u_1 and u_2 be their respective crisp outputs. Then the linearity or non-linearity of different stages of a fuzzy controller can be checked as follows.

Normalizations and denormalization

These steps are linear because they simply involve multiplication by a scalar N_x

$$N_x \,.\, x_1 + N_x \,.\, x_2 = N_x \,.\, (x_1 + x_2) \tag{A.7}$$

and

$$\alpha \,.\, N_x \,.\, x_1 = N_x \,.\, (\alpha \,.\, x_1). \tag{A.8}$$

Fuzzification

Membership function μ_{LX} of the linguistic value LX is, in general, a non-linear function (cf. Fig. 2.4). The fuzzification of x_1 and x_2 results in $\mu_{LX}(x_1)$ and $\mu_{LX}(x_2)$, respectively. Linearity requires

$$\mu_{LX}(x_1) + \mu_{LX}(x_2) = \mu_{LX}(x_1 + x_2). \tag{A.9}$$

But this can not be fulfilled because of the non-linear characteristic of μ_{LX}.

Rule firing

The membership function μ_{LU} of the linguistic value LU are, in general, non-linear functions. With this in mind, the result of firing the rule for input x_1 would be:

$$\forall u \,:\, \mu'_{CLU}(u) = \mu_{LX}(x_1) \wedge \mu_{LU}(u), \tag{A.10}$$

and for input x_2

$$\forall u : \mu''_{CLU}(u) = \mu_{LX}(x_2) \wedge \mu_{LU}(u), \tag{A.11}$$

where $\wedge$ denotes the *min* fuzzy operator and $\mu_{CLU}(u)$, in general, refers to a clipped fuzzy set for linguistic variable LU. Linearity requires

$$\forall u : \mu'_{CLU}(u) + \mu''_{CLU}(u) = \mu_{LX}(x_1 + x_2) \wedge \mu_{LU}(u). \tag{A.12}$$

However, this condition can not be met since μ_{LX} and μ_{LU} are a non-linear functions. Furthermore, the $\wedge$ operator is non-linear for $\wedge$ = min. As a result, the inference process, or the rule firing, is again non-linear.

Defuzzification

Assume the defuzzification procedure is performed with the help of *center of area* approach [60]. Furthermore, let u_1 and u_2 be the defuzzification results obtained by

$$u_1 = \frac{\sum_{I=1}^{l} \mu'_{CLU}(u_i) \cdot u_i}{\sum_{I=1}^{l} \mu'_{CLU}(u_i)} \tag{A.13}$$

and

$$u_2 = \frac{\sum_{I=1}^{l} \mu''_{CLU}(u_i) \cdot u_i}{\sum_{I=1}^{l} \mu''_{CLU}(u_i)} \tag{A.14}$$

Linearity requires

$$u_1 + u_2 = \frac{\sum_{I=1}^{l}(\mu'_{CLU}(u_i) + \sum_{I=1}^{l} \mu''_{CLU}(u_i)) \cdot u_i}{\sum_{I=1}^{l}(\mu'_{CLU}(u_i) + \mu''_{CLU}(u_i))}. \tag{A.15}$$

This results in

$$u_1 + u_2 = \frac{\sum_{I=1}^{l} \mu'_{CLU}(u_i) \cdot u_i}{\sum_{I=1}^{l}(\mu'_{CLU}(u_i) + \mu''_{CLU}(u_i))} + \frac{\sum_{I=1}^{l} \mu''_{CLU}(u_i) \cdot u_i}{\sum_{I=1}^{l}(\mu'_{CLU}(u_i) + \mu''_{CLU}(u_i))} \tag{A.16}$$

which cannot be fulfilled. Instead of this equation, we have

$$u_1 + u_2 = \frac{\sum_{l=1}^{l} \mu'_{CLU}(u_i) \,.\, u_i}{\sum_{l=1}^{l} \mu'_{CLU}(u_i)} + \frac{\sum_{l=1}^{l} \mu''_{CLU}(u_i) \,.\, u_i}{\sum_{l=1}^{l} \mu''_{CLU}(u_i)}. \qquad (A.17)$$

Therefore, the defuzzification process is non-linear as well.

This proof indicates that the source of non-linearity of fuzzy controllers, in general, comes from fuzzification, rule firing, and defuzzification. As has been pointed out in Section 5.3.3, a fuzzy controller can be linearized piecewise if it is designed based on sliding mode controller.

Appendix B

Mathematical model of an induction motor with field oriented control

The dynamic behavior of an induction motor in the synchronously rotating frame can be described by the following state equations [65, 66]

$$\begin{bmatrix} \dot{i}_{ds} \\ \dot{i}_{qs} \\ \dot{\lambda}_{dr} \\ \dot{\lambda}_{qr} \end{bmatrix} = \begin{bmatrix} \frac{-R_s}{\sigma L_s} - \frac{R_r(1-\sigma)}{\sigma L_r} & \omega_e & \frac{L_m R_r}{\sigma L_s L_r^2} & \frac{P\omega_r L_m}{2\sigma L_s L_r} \\ -\omega_e & \frac{-R_s}{\sigma L_s} - \frac{R_r(1-\sigma)}{\sigma L_r} & \frac{P\omega_r L_m}{2\sigma L_s L_r} & \frac{L_m R_r}{\sigma L_s L_r^2} \\ \frac{L_m R_r}{L_r} & 0 & -\frac{R_r}{L_r} & \omega_e - \frac{P}{2}\omega_r \\ 0 & \frac{L_m R_r}{L_r} & -\left(\omega_e - \frac{P}{2}\omega_r\right) & -\frac{R_r}{L_r} \end{bmatrix} \begin{bmatrix} I_{ds} \\ I_{qs} \\ \lambda_{dr} \\ \lambda_{qr} \end{bmatrix} + \frac{1}{\sigma L_s} \begin{bmatrix} v_{ds} \\ v_{qs} \\ 0 \\ 0 \end{bmatrix} \quad (B.1)$$

$$T_e = \frac{3P}{4}\frac{L_m}{L_r}(I_{qs}\lambda_{dr} - I_{ds}\lambda_{qr}) \quad (B.2)$$

where

$$\begin{aligned}
\sigma &= 1 - L_m^2/(L_sL_r) \\
\lambda_{dr} &= L_m I_{ds} + L_r I_{dr} \\
\lambda_{qr} &= L_m I_{qs} + L_r I_{qr} \\
R_s &= \text{stator resistance per phase} \\
L_m &= \text{magnetizing inductance per phase} \\
R_r &= \text{rotor resistance per phase} \\
L_r &= \text{rotor inductance per phase referred to stator} \\
L_s &= \text{stator inductance per phase} \\
P &= \text{number of poles} \\
\omega_e &= \text{electrical angular speed} \\
V_{ds} &= \text{d axis stator voltage} \\
V_{qs} &= \text{q axis stator voltage} \\
I_{ds} &= \text{d axis stator current} \\
I_{qs} &= \text{q axis stator current} \\
I_{dr} &= \text{d axis rotor current} \\
I_{qr} &= \text{q axis rotor current}
\end{aligned} \qquad \text{(B.3)}$$

In the field-oriented control for an induction motor, the ideal decoupling between the d and axes can be achieved by letting the rotor flux linkage in the d axis, i.e.

$$\begin{cases} \lambda_{qr} = 0 \\ d\lambda_{qr}/DT = 0. \end{cases} \qquad \text{(B.4)}$$

Using B.4, the desired rotor flux linkage $\tilde{\lambda}_r = \lambda_{dr}$ in terms of I_{ds} can be found from the third row of B.1 as

$$\lambda_{dr} = (I_{ds}L_m)/(1 + L_r s/R_r). \qquad \text{(B.5)}$$

For the highest utility of the machine core, I_{ds} can be set constant for the desired rated rotor flux. In this situation if the dynamic characteristic of rotor flux in B.5 is neglected, the torque equation B.2 then becomes

$$T_e = K_{foc} I_{qs} \tag{B.6}$$

where

$$K_{foc} = (3P/4)(L_m^2/L_r)I_{ds}. \tag{B.7}$$

For the mechanical system, the torque and rotor angular speed are related by

$$\omega_r(s) = G(s)(T_e(s) - T_L(s)) \tag{B.8}$$

with

$$G(s) = \frac{1/J}{s + B/J} = \frac{b}{s+a} \tag{B.9}$$

where B and J denote the total damping ratio and inertia constant of the drive system, respectively.

Appendix C

Simulation programs

The computer program for auto-design of fuzzy controllers has been written in C. The program includes three different files; optimization file, closed loop file, and a header file. As the header file indicates, the optimal fuzzy controller can be optimized based on any combination of different sets of controller parameters. In what follows, the header file and only the *main function* of the genetic algorithm file and closed loop file are presented.

```
************************************************
*  FILE:       header.h
*  AUTHOR:     Farhad Ashrafzadeh
*  FUNCTION:   header file for ga.c & flc.c files
************************************************


/*~~~~~~~~ FREE PARAMETERS OF GENETIC ALGORITHM :
#define GENERATE_NOR          "off"
#define GENERATE_MF           "off"
#define GENERATE_TABLE        "on"
#define GENERATE_SINGLETON    "on"
```

```
#define WHICH_PART_CROSS 2   /* 0: nor, 1: mf, 2: table, 3:one point */
#define WHICH_PART_MUT  2   /* 0: nor, 1: mf, 2,3,4,.. : table */
#define NO_OF_INITIAL    0

#define EXISTING_KNOW_NOR     "yes"
#define EXISTING_KNOW_MF      "yes"
#define EXISTING_KNOW_TABLE   "yes"

#define MAX_GENERATION   3
#define POPULATION_NO    10
#define MAX_MUT_PERC     30
#define SEED  1
#define AV_SCORE_PERC    .4
#define TRANSFER         1
#define HELP_GA          0
#define STRING_LENGTH    20

/*~~~~~~~~ FUZZY CONTROLLER PARAMETERS:
#define NOR_DOMAIN   10
#define MF_PRECISION     10000
#define NO_OF_MF_E       7
#define NO_OF_MF_E_DOT   7
#define TOTAL_MF_NO  (NO_OF_MF_E + NO_OF_MF_E_DOT)
#define MF_PARAMETERES   3 * (NO_OF_MF_E + NO_OF_MF_E_DOT)
#define NO_OF_CELLS  (NO_OF_MF_E * NO_OF_MF_E_DOT)
#define min(x,y) ( x < y ? x : y )     /* MACRO  */
#define max(x,y) ( x > y ? x : y )     /* MACRO  */
```

```
#define NN_E 10 * 100

#define NN_E_DOT 1.61 * 100

#define NN_U_DOT 10 * 100


/*---------    CONTROL LOOP PARAMETERS:

#define  TIME_WINDOW     10

#define  Y_REF           1      /* 1 pu, i.e. 1800 rpm  */

#define  H               0.001

#define  U_MAX    +1.5

#define  U_MIN    -1.5


/*---------  ELECTRICAL MOTOR PARAMETERS:

#define  T_LOAD          1

#define  K_torque        4.44

#define  K_speed         1 /* K_speed = 1 / J  */


*************************************

*  FILE:      gcc.c

*  AUTHOR:    Farhad Ashrafzadeh

*  FUNCTION:  genetic algorithm file.

*************************************


main()

{

init(old_generation, new_generation, variable_length, random_vector);

while ( no_of_generation <= MAX_GENERATION )

{

evaluation(old_generation);
```

```
new_parents_ptr = select_parents(old_generation,
   best_parent_ptr, no_of_generation);
mating_pool(new_parents_ptr, variable_length, random_vector,
    no_of_generation);
cross_over_1_table(new_parents_ptr, random_vector,
    new_generation);
mutation( random_vector, new_generation, best_parent_ptr);
new_gen(old_generation, new_generation);
no_of_generation++;
}
}


************************************************************
*  FILE:      flc.c
*  AUTHOR:    Farhad Ashrafzadeh
*  FUNCTION:  Closed loop file with fuzzy logic controller.
************************************************************


double closed_loop (k, nor_mem_table)
int k;
struct nor_mf_table nor_mem_table;
{
while ( t < TIME_WINDOW )
{
e = Y_REF - y;
e_dot = (e - e_old) / H;
mp = over_shoot (y, e_dot);
pi += t * fabs(e) * H + 6 * mp;
```

```
e_old = e;
u_dot = flc (nor_mem_table, &e, &e_dot);
u += u_dot * H;
u_sat = saturation (u);
y = system_foc_IM (u_sat);
t += H;
}
return(pi);
}
```