Fuzzy Model Reference Learning Control

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of The Ohio State University

by

Jeffery Ray Layne, B.S.E.E.

* * * * *

The Ohio State University

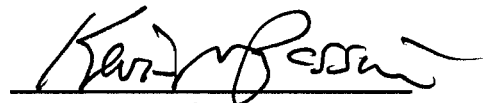
1992

Master's Examination Committee:

Professor Kevin M. Passino

Professor Stephen Yurkovich

Approved by:



Adviser
Department of Electrical
Engineering

To my wife,

Suzanne

ACKNOWLEDGMENTS

If possible, I would like to express my gratitude to those who have made this work possible.

I sincerely thank Professor Kevin Passino for his advice, support, and help in many ways throughout this work. He has helped me to mature professionally in terms of building both competence and confidence. I am sure the lessons that I have learned will greatly enhance my career and future. Also, appreciation is extended to Professor Stephen Yurkovich for his participation in my examination committee.

I have been very fortunate to work with many extremely talented fellow graduate students whose friendship and cooperation have made graduate school an enjoyable experience. I owe much gratitude to Ed Bedner and Jim Nivens for sharing their vast knowledge of automotive anti-skid braking systems. Moreover, I would like to thank John Gossett, Alfonsus Lunardhi, Greg Hanchin, Layne Lenning, and David Schoenwald for their help and assistance with computer software as well as other technical issues. Also, there have been numerous new friends along the way, too many to name. I thank all of you.

Further, I would like to extend my gratitude to David Zann at Wright Patterson Air Force Base and the administrators of the Air Force Palace Knight Program for providing me with this special opportunity to obtain an advanced education free from the financial burdens which often plague many graduate students.

On a personal level, I would like to thank my family and friends whose support

and encouragement has been a true source of inspiration. In particular, I would like to express special thanks to my parents and grandparents who taught me that through hard work and perseverance all of life's goals are within reach. Also, I would like to thank my two brothers and sister who taught me sharing and cooperation. In addition, I would like to thank my wife's family who has provided me a great deal of encouragement. Most importantly, I would like to thank my wife, Suzanne, whose loving, support, encouragement, and dedication has extended far beyond the call of duty. I would also like to thank Amit and Cindy who have been the best of friends by providing me with a shoulder to lean on when times seemed tough and someone to laugh with during the good times.

Finally, I would like to thank God who provided me the strength and courage to meet all of the challenges and obstacles faced throughout graduate school and in life.

VITA

July 8, 1966 ..... Born– Ashland, Kentucky

1985–1987 ..... Electrical Engineering Co-op,
Armco Steel, Inc.,
Middletown, Ohio

September 1987 ..... Associate of Applied Science Electronic Engineering,
Cincinnati Technical College,
Cincinnati, Ohio

March 1990 ..... Bachelor of Science Electrical Engineering
Wright State University,
Dayton, Ohio

Dec 1989–June 1990 ..... Research Assistant,
Wright State University,
Dayton, Ohio

Summer 1990 ..... U.S. Air Force Summer Fellow,
Wright Patterson A.F.B.,
Dayton, Ohio

1990–present ..... U.S. Air Force Palace Knight,
Wright Patterson A.F.B.,
Dayton, Ohio

Fields of Study

Major Field: Electrical Engineering

Major Area of Specialization: ..... Control and Systems Theory

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Introduction

Over the last few decades digital computer technology has matured in an exponential fashion. Modern digital computers are faster, smaller, cheaper, and more reliable than one would have ever imagined only a few decades ago. As a result, the world has witnessed a revolution in the way problems are solved via engineering analysis and contemporary technologies. We are now able to solve problems which were previously considered too complex to be worth the effort or even feasible for a human to solve in a lifetime.

With the increased capabilities of the digital computer, it becomes desirable to develop a computer system which emulates the complexity of the human learning and decision-making process. As a result, we have observed a dramatic increase of activity in research areas such as neural networks, expert systems, and fuzzy systems. These research areas provide algorithms for implementing "human-like" learning and decision-making capabilities. Such algorithms can be used to replace or enhance human decision-making in practical real life situations. Therefore, the applications of computer learning and decision making are almost limitless and include a large diversity of professions such as business, medicine, and engineering.

In this thesis we will focus on fuzzy system theory and its application in the

control of a variety of complex dynamical systems. In general, a "fuzzy controller" utilizes a fuzzy system to capture a human expert's knowledge about controlling a process for use in a computer algorithm. Thus, the fuzzy controller provides a means for automatic control of a process which would normally be controlled by humans. Furthermore, since high speed computers are often used for implementation, the applications of fuzzy control can be extended to include processes which require very fast control actions that are much faster than is possible for humans. In this case, the fuzzy controller is designed to control the process in a manner that a human would if he/she were capable of generating the control action themselves.

Often, fuzzy controllers are designed to emulate the human decision-making process rather than the human learning process since some knowledge about how to control the process is needed for design. In this thesis we will discuss and propose methods for obtaining the knowledge automatically. In effect, we provide the controller with a method of "learning". Hence, for control applications we often refer to such systems as "learning controllers".

Since learning control is closely related to other techniques in classical control theory, self-organizing control, and adaptive control, we will present comparisons with these methods. The objectives of this thesis may be summarized as

- Presentation of theory, design, and application of direct fuzzy controllers,
- Introduce methods and applications for "fuzzy learning control",
- Comparison of "fuzzy learning control" with conventional adaptive control techniques.

Overall, this thesis was written with the primary focus on theory and application of fuzzy learning control systems.

1.1 History and Background of Fuzzy Control

Fuzzy set theory originated in the early work of Lotfi A. Zadeh who is at the University of California, Berkeley. Zadeh proposed the possibility of using fuzzy set theory as a means of analyzing some very complex real world dynamical systems. However, it was the pioneering research by E.H. Mamdani and his colleagues at Queen Mary College in England who demonstrated how fuzzy set theory could be employed in control applications. Motivated by the success of Mamdani and his fellow researchers in applying fuzzy control in an experimental setting, other researchers followed suit.

Over recent years, fuzzy control has emerged as a practical alternative to conventional control techniques. Fuzzy control has proven its effectiveness in controlling highly non-linear, time-varying, and ill-defined systems. For instance, fuzzy control has been employed in many applications including: control of a warm water plant [39], control of a steam engine and boiler combination [51], control of a nuclear reactor [9], and control of servo-motors [50]. In all cases, fuzzy control has proven to be as, if not more, effective than conventional proportional integral derivative (PID) control and in some cases better than more sophisticated control schemes. More recently, new products such as auto-focusing cameras, washing machines, and air conditioners employ fuzzy control [65].

In fuzzy control design, it is often assumed that it is either impossible or very difficult to produce a model for the system to be controlled. In the case where it is very difficult to obtain a process model, the resulting model may be highly non-linear, time-varying and ill-defined. Therefore, traditional control techniques do not offer much in answering the question of how to design a controller. However, in

such situations the control engineer or human expert operator may have an intimate understanding of the plant dynamics and therefore a good idea of how to control the process. The design methodology for fuzzy control relies on the representation of a control engineer's or human expert operator's knowledge in a formal mathematical way so that it can be used in an algorithm to control a dynamical system.

In designing fuzzy control systems, we typically express linguistic variables such as the process inputs, outputs, functions of the inputs and outputs, etc. in terms of linguistic values such as hot, warm, cold, and cool. Note that such terms do not have exact meaning and therefore can take on a fuzzy type of characteristic. For example, some people may classify 25 degrees Celsius as warm others may classify it as hot; hence, the relationship between temperature and values such as warm and hot is fuzzy. The linguistic variables described above can be tested with a set of **IF–THEN** rules developed by the control engineer or human expert operator which produce the appropriate output depending on which of the control rules are asserted. For example, if the system were an air conditioner, one typical control rule might be "if the room is warm then increase power slightly".

Due to the way that information about controlling the process is represented many details of the underlying dynamical behavior are ignored. As a result, fuzzy controllers can often be robust to changes in the plant. This "robustness" property of fuzzy control is typically not characterized mathematically as in conventional control theory. However, empirical evidence suggests that fuzzy controllers are often robust.

1.2 Contributions of this Thesis

Despite the many successes of fuzzy control, there exist several significant drawbacks of this approach:

1. The design of fuzzy controllers is usually performed in an *ad hoc* manner; hence, it is often not clear exactly how to justify the choices for many parameters in the fuzzy controller (e.g., the membership functions).

2. The fuzzy controller constructed for the nominal plant may later perform inadequately if significant and unpredictable plant parameter variations, structural changes, and environmental disturbances occur.

In this thesis a "learning" control algorithm is presented which helps to resolve some of these fuzzy controller design issues. This algorithm employs a reference model (a model of how you would like the plant to behave) to provide closed loop performance feedback for generating and modifying a fuzzy controller's knowledge base. Consequently, this algorithm is referred to as a "fuzzy model reference learning controller" (FMRLC). The FMRLC algorithm grew from research performed on the linguistic self-organizing controller (SOC) presented in [59] by Procyk and Mamdani and ideas in conventional "model reference adaptive control" (MRAC) [3]. Since the basic architecture and functionality of the FMRLC is consistent with the prevailing definition of a "learning controller", the term "learning" is used rather than "adaptive". Additional justification for using the term "learning" is provided later in this thesis.

In the development of the FMRLC we first enhance the SOC approach in [59] by developing a new knowledge base modification algorithm. The knowledge base modification algorithm of Procyk and Mamdani [59] relies on modification of a fuzzy relation table which describes the relationship between the fuzzy controller inputs and outputs. Often, this automatically implies that all input and output universes of discourse must be quantized into discrete levels to implement the fuzzy relation in a computer. Unfortunately, this will generally result in large memory requirements and

computational demands since a fuzzy relation table often contains many entries for real world applications. In this thesis, we present a new knowledge base modification algorithm which reduces computation time and memory requirements by utilizing a rule base array table rather than a fuzzy relation table. It is shown in this thesis how this modification algorithm can be used both to enhance the SOC approach in [59] and for knowledge base modification in the FMRLC.

In addition to it knowledge base modification capabilities we will show that FMRLC has several other significant advantages over the linguistic SOC in [59], including improved performance feedback and wider applicability to challenging control problems. The performance criteria for the linguistic SOC can only characterize what is essentially a compromise between rise-time and overshoot. Hence it provides little flexibility in specifying what is to be learned. However, via a "reference model" the FMRLC has the capability to accurately quantify virtually any form of desired performance.

A final contribution of this thesis includes the application of the FMRLC for several complex real world applications. These applications includes a cart and pendulum system, a rocket velocity controller, a two degree-of-movement manipulator, an automotive braking system, and a cargo ship heading controller. The cart and pendulum system is studied to illustrate and compare the effectiveness of both the FMRLC and the linguistic SOC for controlling an unstable non-linear system. The rocket system illustrates the effectiveness of the FMRLC for controlling highly time varying processes since the mass of a rocket changes in due to fuel loss. The two degree-of-movement manipulator is studied to illustrate the design and implementation of the FMRLC for a multiple input multiple output non-linear process. The automotive braking system is studied as a practical application of the FMRLC al-

gorithm since the dynamic of braking system vary greatly for various road surfaces and slopes. Finally, the control of a cargo ship's heading was used to compare the effectiveness of the FMRLC with conventional "model reference adaptive control" (MRAC). This comparison includes both the gradient and Lyapunov approaches to MRAC.

1.3 Thesis Organization

This thesis is organized in the following way. In Chapter II we present an introduction to fuzzy system and their application in control. This chapter is provided to unify the fuzzy set theory, ideas, and notation which are used in this thesis. This chapter is necessary since there does not exist a good introduction of fuzzy sets and systems from a control-theoretic perspective. We begin this chapter with a section overviewing the mathematical preliminaries of fuzzy set and control theory. This includes an introduction to fuzzy system terminology, mathematical operations for fuzzy sets, and an explanation of fuzzy implication and inference. This section is followed by a section describing various characteristics which are often imposed on fuzzy set membership functions. The next section describes the functional architecture which is generally used for fuzzy system design. The next section provides an example fuzzy system design for a motor positioning controller. Finally the next section in this chapter provides a comparison of fuzzy control with conventional linear control methods, thus illustrating the potential advantages of fuzzy control over conventional linear control laws.

Chapter III provides an introduction to fuzzy adaptive and learning control methods. The first section of this chapter presents an overview of the literature which describes how fuzzy set theory and systems are used in adaptive and learning

control algorithms. The next section presents a generalized framework and possible for adaptation of direct fuzzy controllers. As presented in the literature overview, an innovative adaptive fuzzy control technique called a "linguistic self organizing controller" was presented by Procyk and Mamdani [59]. In the following section of this chapter, we present the architecture and design methodology of the linguistic SOC. The next section addresses the difficult issue of what distinguishes a learning system from an adaptive system. It is this section which provides the justification for using the phrase "learning" when naming the FMRLC. In the next section we present the architecture and design methodology of the FMRLC. Also, in this section we discuss the possible advantages of the FMRLC algorithm over the linguistic SOC.

Chapter IV addresses design and implementation issues of the FMRLC algorithm by employing it to control several non-linear, time-varying processes. This chapter begins with a section describing the design and simulation of both the linguistic SOC of Procyk and Mamdani and the FMRLC for a cart and pendulum system. This cart and pendulum system example clearly illustrates the advantages of the FMRLC algorithm over the linguistic SOC. Further, through this example we are able to illustrate the effect of changing various design parameters, thus providing some guidelines for FMRLC design. The next section presents the design and simulation results for a FMRLC employed to control the velocity of a rocket. Since the mass of a rocket is constantly changes due to the loss of fuel, this application illustrates the effectiveness of the FMRLC algorithm when controlling a highly time-varying process. Finally, the next section of this chapter presents the design and simulation result for a FMRLC employed to control the position of a two degree-of-freedom robot manipulator. This application is provided to illustrate the effectiveness of the FMRLC when controlling MIMO processes.

In Chapter V, we show how to use the FMRLC for a vehicle traction controller. This FMRLC was designed particularly for anti-skid stopping during an emergency braking situation. We begin this chapter with a section which describes techniques which have been investigated by others for this application. The next section discusses the mathematical modeling of the wheel, the vehicle, and the braking system dynamics for controller design and simulation. This section is followed by a section describing controller design. This includes a method for slip (the controlled parameter) estimation and the FMRLC design. Finally, the next section shows simulation results for a automotive braking system. These simulations include the effects of various road surfaces such as dry and wet asphalt, ice, split road surfaces, and road slopes.

Chapter VI provides a comparison of the FMRLC algorithm with conventional "model reference adaptive control" (MRAC) techniques by means of an autopilot application for a cargo ship. This comparison includes both the gradient and Lyapunov approaches to MRAC design. We begin this chapter with a section briefly describing a model of the ship dynamics. This section is followed by a section describing a FMRLC design for the cargo ship system. The next section presents a generalized discussion of MRAC as well as the theory and design of both the gradient and Lyapunov approaches to MRAC for the cargo ship. In the next section we show the simulation results for the FMRLC and both the gradient and Lyapunov approaches to MRAC.

The final chapter, Chapter VII summarizes this thesis. Also included is a brief conclusion section and a section detailing suggestions for possible future research directions.

An appendix is included to provide examples of Fortran programming code for

direct fuzzy control, linguistic SOC, and FMRLC. The motor positioning system and the cart and pendulum system were chosen as typical examples of these control algorithms.

CHAPTER II

An Introduction to Fuzzy Systems and Control

In this chapter we present a basic overview of the fundamental mathematical concepts related to fuzzy set theory. Also, we will discuss how these concepts are employed to develop the basic framework for a fuzzy system. In particular, we will present the standard functional architecture typically used for implementing fuzzy systems. Finally, through an example, we will illustrate how the fuzzy system may be employed as a controller in a closed loop control scheme. Also, a comparison will be made to show how the framework for fuzzy control schemes compares with the classical proportional integral derivative (PID) control framework.

This overview is intended primarily to establish a background and to develop the definitions and notations used in subsequent chapters.

2.1 Mathematical Preliminaries

As an introduction to fuzzy set theory, we have briefly summarized the fundamental concepts and definitions that are most commonly used in fuzzy controller design since that is the main topic of this thesis. For a more detailed discussion refer to the work of Lee in [46, 47], Zadeh in [89, 90], Mamdani and colleagues in [51, 52], Self in [65], Sugeno in [71], and Tong in [80].

2.1.1 Fuzzy System Terminology

A block diagram for a fuzzy system is shown in Figure 1. The primary function of this fuzzy system is to create a static nonlinear functional relationship between the input and outputs of the system. In Figure 1, the fuzzy system is assumed to have r inputs denoted by u_i such that $u_i \in \mathcal{U}_i$ where $1 \leq i \leq r$. Likewise, the system is assumed to have s outputs denoted by y_i such that $y_i \in \mathcal{Y}_i$ where $1 \leq i \leq s$. In discrete time implementation of a fuzzy system t in Figure 1 may stand for kT, where T denotes the sample period and k is the sample time index.

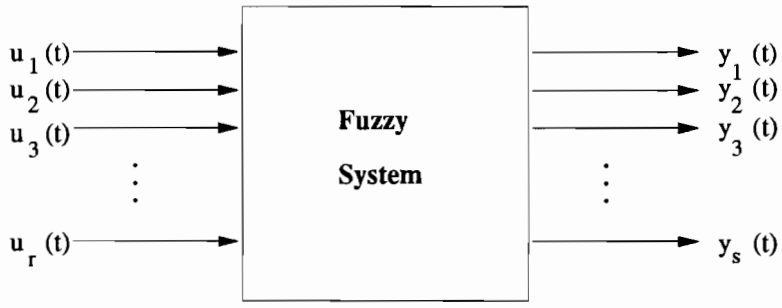


Figure 1: Block diagram of a fuzzy system.

Given the fuzzy system in Figure 1, we can define the terms "universe of discourse","linguistic variable" and "linguistic value" by example.

Definition 2.1 (Universe of Discourse) *The ordinary sets $\mathcal{U}_i$ and $\mathcal{Y}_j$ discussed above are called the "universes of discourse" for u_i and y_i in the fuzzy system shown in Figure 1.*

For example, if the temperature of liquid water in degrees Celsius was one of the inputs to the fuzzy system in Figure 1 denoted by u_i then the *universe of discourse*

for u_i could be $\mathcal{U}_i = [0, 100]$. However, it should be noted that Definition 2.1 of a universe of discourse applies only for the fuzzy system shown in Figure 1. As will be illustrated later in this section, the universe of discourse takes on a broader meaning when discussed in terms of of the generalized framework of fuzzy set theory.

A fuzzy system operates on the inputs to produce the outputs by using an abstract representation of the inputs and outputs. This abstract representation is expressed by linguistic expressions such as "temperature", "pressure", "flow rate"; hence, the name "linguistic variable".

Definition 2.2 (Linguistic Variable) *A linguistic variable is a constant symbolic description of what is in general a time varying quantity. For instance, linguistic variables denoted by $\tilde{u}_i$ are associated with the inputs u_i in Figure 1. Likewise, the linguistic variables denoted by $\tilde{y}_i$ are associated with inputs y_i in Figure 1.*

Just as u_i and y_i in Figure 1 take on values over the universe of discourse $\mathcal{U}_i$ and $\mathcal{Y}_i$, respectively, linguistic variables $\tilde{u}_i$ and $\tilde{y}_i$ will take on "linguistic values".

Definition 2.3 (Linguistic Value) *Let $\tilde{U}_i^j$ denote a linguistic value of the linguistic variable $\tilde{u}_i$ defined on the universe of discourse $\mathcal{U}_i$. If we assume that there exist many linguistic values defined over $\mathcal{U}_i$, then the linguistic variable $\tilde{u}_i$ takes on elements from the set of linguistic values denoted $\tilde{U}_i = \{\tilde{U}_i^j\}$ where $N_i^- \leq j \leq N_i^+$.*

Likewise, let $\tilde{Y}_i^m$ denote a linguistic value of the linguistic variable $\tilde{y}_i$ defined on the universe of discourse $\mathcal{Y}_i$. Moreover, assume that there exist many linguistic values defined over the universe of discourse $\mathcal{Y}_i$. Then the linguistic variable $\tilde{y}_i$ takes on elements from the set of linguistic values denoted by $\tilde{Y}_i = \{\tilde{Y}_i^m\}$ where $M_i^- \leq m \leq M_i^+$.

Linguistic values are generally expressed by terms such as: "small", "medium", and "big". For example, if we assume that $\tilde{u}_i$ denotes the linguistic variable "temperature", then we may assign $\tilde{U}_1^1$ = "cold", $\tilde{U}_1^2$ = "warm", and $\tilde{U}_1^3$ = "hot" so that $\tilde{u}_i \in \tilde{U}_1$ where $\tilde{U}_1 = \{\tilde{U}_1^1, \tilde{U}_1^2, \tilde{U}_1^3\}$.

The mapping of the inputs to the outputs for a fuzzy system is often represented by a set of *condition* $\rightarrow$ *action* rules. In a fuzzy system, these rules are generally expressed in the modus ponens form as,

If (antecedent) **Then** (consequent).

This form is most often used due to its intuitive appeal. Often, the fuzzy system inputs, expressed in terms of linguistic values and linguistic variables, are often associated with the antecedent. Likewise, the fuzzy system outputs are associated with the consequent. In general, for fuzzy systems the antecedents can be expressed in terms of both the inputs and the outputs and perhaps other internal variables. Also, in general, a sequence of rules could occur to determine what the output is; however, most often for fuzzy control the input-output relationship is expressed in terms of simple *condition* $\rightarrow$ *action* rules and not chains of such rules.

In fuzzy system design, a control engineer or human expert operator generates a set of control rules expressed by linguistic terms such as "temperature is hot". These *condition* $\rightarrow$ *action* rules are generally given in one of four forms including: 1) single input - single output (SISO); 2) multiple input - single output (MISO); 3) single input - multiple output (SIMO); and 4) multiple input - multiple output (MIMO). Each of these forms can be expressed linguistically with generic arguments as:

1. **SISO:**

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ Then } \tilde{y}_1 \text{ is } \tilde{Y}_1^m,$$

2. **MISO:**

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{U}_2^k \textbf{ and, . . . , and } \tilde{u}_r \text{ is } \tilde{U}_r^l \textbf{ Then } \tilde{y}_1 \text{ is } \tilde{Y}_1^m,$$

3. **SIMO:**

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ Then } \tilde{y}_1 \text{ is } \tilde{Y}_1^m \textbf{ and } \tilde{y}_2 \text{ is } \tilde{Y}_2^n \textbf{ and, . . . , and } \tilde{y}_s \text{ is } \tilde{Y}_s^o,$$

4. **MIMO:**

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{U}_2^k \textbf{ and, . . . , and } \tilde{u}_r \text{ is } \tilde{U}_r^l$$
$$\textbf{Then } \tilde{y}_1 \text{ is } \tilde{Y}_1^m \textbf{ and } \tilde{y}_2 \text{ is } \tilde{Y}_2^n \textbf{ and, . . . , and } \tilde{y}_s \text{ is } \tilde{Y}_s^o.$$

However, rules in the SIMO and the MIMO form can be broken up to form several fuzzy systems that share common inputs but produce independent outputs thus producing fuzzy systems with rules in the SISO and the MISO form, respectively. For example, the MIMO action rule defined above could be rewritten as s MISO rules expressed as:

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{U}_2^k \textbf{ and, . . . , and } \tilde{u}_r \text{ is } \tilde{U}_s^l \textbf{ Then } \tilde{y}_1 \text{ is } \tilde{Y}_1^m,$$
$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{U}_2^k \textbf{ and, . . . , and } \tilde{u}_r \text{ is } \tilde{U}_s^l \textbf{ Then } \tilde{y}_2 \text{ is } \tilde{Y}_1^n,$$

.

.

.

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{U}_2^k \textbf{ and, . . . , and } \tilde{u}_r \text{ is } \tilde{U}_s^l \textbf{ Then } \tilde{y}_s \text{ is } \tilde{Y}_s^o.$$

Thus, it is only necessary to consider the MISO form since the SIMO and MIMO can be reduced to it and the SISO form is a special case of the MISO form.

Everything discussed up to this point has been in terms of the linguistic statements that characterize the relationship of the inputs to the outputs. However, fuzzy set theory provides a technique to quantify this knowledge for practical use. Therefore by using fuzzy set theory, the linguistic *condition* $\rightarrow$ *action* rules are transformed into an analytical framework for controller synthesis, control system analysis, and controller implementation.

In order to introduce the concept of a fuzzy set, we first define a membership function.

Definition 2.4 (Membership Function) *Let $\mathcal{U}_i$ denote a universe of discourse and $\tilde{U}_i^j \in \tilde{U}_i$ denote a specific linguistic value for the linguistic variable $\tilde{u}_i$. The function $\mu(u_i) : \mathcal{U}_i \rightarrow [0,1]$ associated with $\tilde{U}_i^j$ is called a membership function. This membership function describes the "certainty" that an element of $\mathcal{U}_i$ denoted u_i may be classified linguistically as $\tilde{U}_i^j$.*

A membership function is used to incorporate the characteristic fuzziness involved when relating a specific value of the universe of discourse to a given linguistic value such as "big", "medium", "small", etc. This fuzziness is generally due to the fact that the assignment of a specific value to a linguistic value is a subjective process. Moreover, observed data may be disturbed by noise thus creating some additional uncertainty when assigning specific values to a given linguistic value.

For example, 25 degrees Celsius may be classified as warm 25 percent of the time and as hot 75 percent of the time. Thus we can infer that the statement "25 degrees Celsius is warm" is *25 percent true* or has 0.25 certainty. Figure 2(b) illustrates typical

membership functions that could have been defined for this example. Essentially, the membership function assigns a relative truth value or "certainty" that each numeric value of temperature may be classified by given linguistic value such as warm or hot. It should be noted that membership functions are generally determined arbitrarily from experience and should not be confused with probability density functions.

Definition 2.5 (Fuzzy Set) *Given a linguistic variable $\tilde{u}_i$ with a linguistic value $\tilde{U}_i^j$ defined on the universe of discourse $\mathcal{U}_i$, and membership function $\mu_{U_i^j}(u_i) : \mathcal{U}_i \rightarrow [0,1]$, a "fuzzy set" denoted with U_i^j is defined as*

$$U_i^j = \{(u_i, \mu_{U_i^j}(u_i)) : u_i \in \mathcal{U}_i\}. \tag{2.1}$$

Notice that this is a generalization of ordinary sets whose membership values take on only two values 0 and 1. If we had assumed ordinary sets, 25 degrees Celsius would most likely be assigned only to the linguistic description of hot with a full truth value of 1. This relationship between ordinary sets and fuzzy sets is easily seen in Figure 2 where membership functions for warm and hot are arbitrarily chosen.

For example, consider Figure 2(b) where if we assign linguistic variable $\tilde{u}_i$ = "temperature" and linguistic variable $\tilde{U}_i^j$ = "warm", then U_i^j is a fuzzy set whose membership function describes the degree of certainty that the numeric value of the temperature, $u_i \in \mathcal{U}_i$, possesses the property characterized by $\tilde{U}_i^j$. In particular, the fuzzy set consists of pairs of elements, the first a value of temperature u_i, and the second $\mu_{U_i^j}(u_i)$ which is equal to μ_{Twarm} shown in Figure 2(b).

In ordinary set theory, a set denoted A is defined to be a subset of set B if and only if all elements of set A are contained in set B. This concept of a subset for ordinary sets has been extended for fuzzy sets according to the following definition.

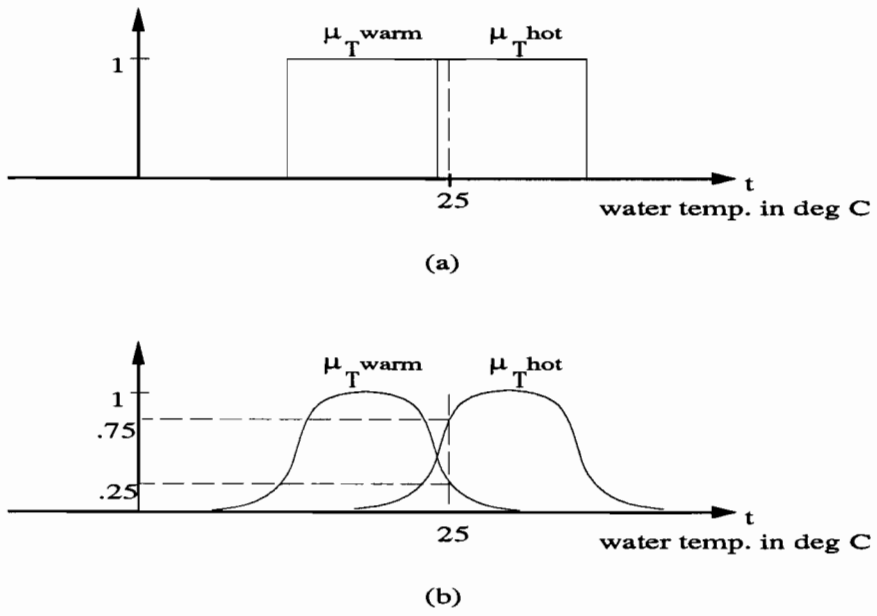


Figure 2: Ordinary set vs. fuzzy set membership functions.

Definition 2.6 (Fuzzy Subset) *Given fuzzy sets, denoted* U_i^1 *and* U_i^2*, associated with the universe of discourse* $\mathcal{U}_i$ *with a membership functions denoted* $\mu_{U_i^1}(u_i)$ *and* $\mu_{U_i^2}(u_i)$*, respectively,* U_i^1 *is defined to be a fuzzy subset of* U_i^2*, denoted as* $U_i^1 \subset U_i^2$*, if*

$$\mu_{U_i^1}(u_i) \leq \mu_{U_i^2}(u_i) \qquad \forall u_i \in \mathcal{U}_i. \tag{2.2}$$

Now that we have defined a fuzzy subset we can gain some added insight into the meaning of the term "universe of discourse". Essentially, the universe of discourse is an ordinary set for which a fuzzy set may be viewed as a subset. For instance, consider a more generalized non-numerical example in which the elements of the universe of discourse include "all cars". Therefore, we may consider the following three fuzzy sets "red cars", "yellow cars", and "blue cars" as subsets of the universe

of discourse "all cars". These are fuzzy sets since for instance a green car may be considered partially a member of "yellow cars" and partially a member of "blue cars" but never totally a member of "yellow cars" or of "blue cars". This example is illustrated in a fuzzy set version of a "Venn diagram" shown in Figure 3 below.

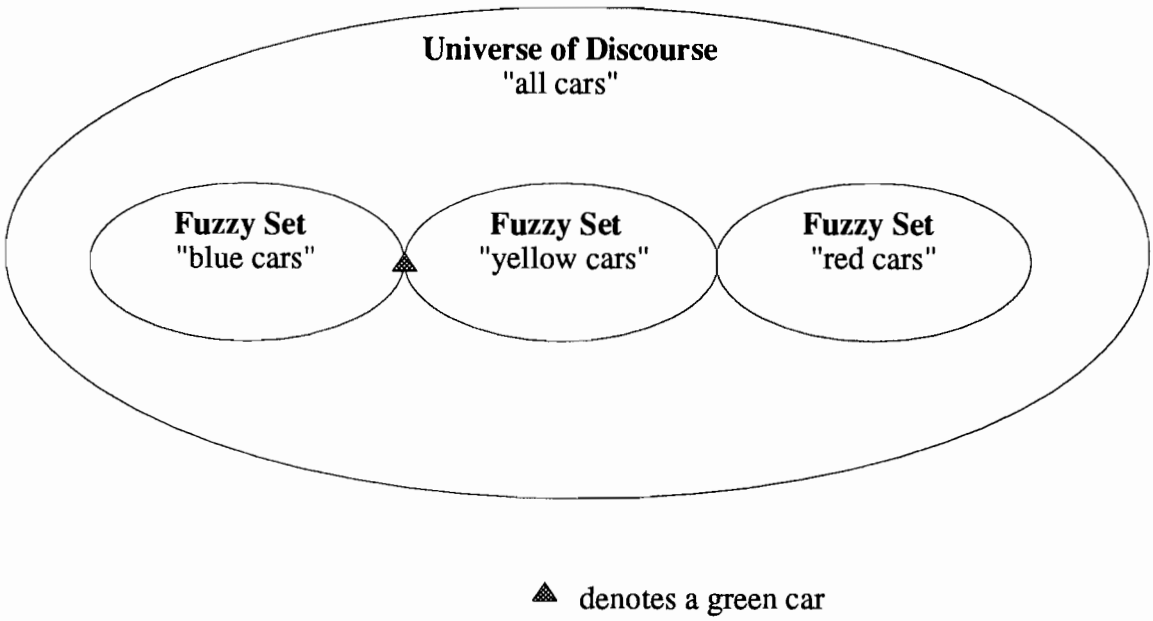


Figure 3: Fuzzy set version of a "Venn diagram".

Clearly, the universe of discourse $\mathcal{U}_i$ described as an ordinary set in Definition 2.1 may be generalized as a fuzzy set $\{(u_i, 1) : u_i \in \mathcal{U}_i\}$. Therefore all fuzzy sets defined on $\mathcal{U}_i$ are subsets of the fuzzy set associated with the ordinary set $\mathcal{U}_i$.

In classical set theory, we often refer to a set which contains no elements as a "null set". A generalized version of a "null set" for fuzzy set theory is given in the following definition.

Definition 2.7 (Null Fuzzy Set) *Given a fuzzy set, denoted $U_i^j \subset \mathcal{U}_i$ with a membership function denoted $\mu_{U_i^j}(u_i) : \mathcal{U}_i \rightarrow [0,1]$, U_i^j is equal to the null fuzzy set, denoted Φ if*

$$\mu_{U_i^j}(u_i) = 0 \qquad \forall u_i \in \mathcal{U}_i. \tag{2.3}$$

Definition 2.7 can be paraphrased as follows: if U_i^j is a fuzzy set then no element of the universe of discourse has membership in U_i^j. This concept will become very useful for future discussions.

2.1.2 Basic Fuzzy Set Operations

In ordinary set theory we typically consider fundamental operations including: union, intersection, complement, and etc.. In developing fuzzy systems, it is often necessary to consider generalized versions of these fundamental operations for fuzzy sets. Therefore, in this subsection we summarize some of the more commonly utilized operations in fuzzy systems.

Before beginning this section, we will make a few comments to clarify the notation used throughout this section and the rest of this thesis. Note that up to this point, subscripts are used to denote a particular universe of discourse that is associated with a given fuzzy set denoted by a capital letter which is shared by other fuzzy sets on different universes of discourse. However, in order to maintain the most concise notation, these subscripts may be dropped if it is assumed that only one universe of discourse is associated with all fuzzy sets denoted by a given capital letter. For example, if we assume that there is only one input to a fuzzy system, denoted u_1, then the subscript may be dropped so that u_1 could be denoted by u. Moreover, the subscript for the fuzzy sets associated with u_1, denoted U_1^j, may be

dropped so that U_1^j could be denoted by U^j. Likewise, the superscripts used to denote a particular fuzzy set associated with a given universe of discourse may be dropped if only one fuzzy set is assumed to be associated with a given universe of discourse. These modifications in notation will be used throughout the rest of this thesis. For more details on the notation scheme see the subsection in this chapter titled Notation Summary.

The following definition involving the union of two fuzzy sets may be viewed as a generalization of the union of two ordinary sets.

Definition 2.8 (Union) *The union of fuzzy sets U^1 and U^2 which are subsets of the universe of discourse $\mathcal{U}$ is a fuzzy set, denoted $U^1 \cup U^2$, with a membership function defined by*

$$\mu_{U^1 \cup U^2}(u) = max\{\mu_{U^1}(u), \mu_{U^2}(u) : u \in \mathcal{U}\}. \tag{2.4}$$

The union corresponds to the logic connective 'OR'.

Therefore, the resulting fuzzy set for the union of two fuzzy sets U^1 and U^2 contains all elements of both fuzzy sets with a membership equal to the greatest membership that the element obtains in U^1 or U^2. Hence, this definition of the union of two fuzzy sets is consistent with the union of two ordinary sets in which the resulting set contains all elements of both ordinary sets.

The union provides a means to define a fuzzy set for the situation where two or more linguistic values for a given linguistic variable are connected linguistically by 'OR'. For example, consider the linguistic statement "temperature is warm or hot". If the membership functions for warm and hot are given in Figure 2(b), then

from Definition 2.8 the fuzzy set describing "temperature is warm or hot" has a membership function shown in Figure 4 below.

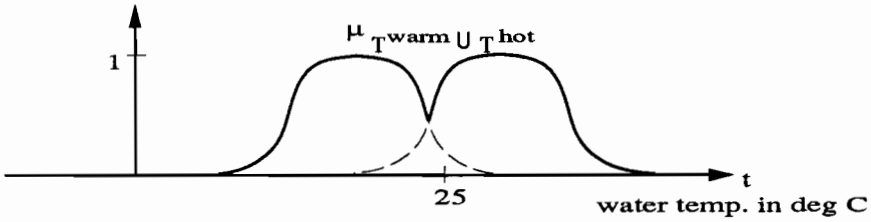


Figure 4: Membership function for the fuzzy set describing "temperature is warm or hot".

Notice in Figure 4 that the union operation in Definition 2.8 results in a loss of information where the two membership functions "overlap". Other definitions for the union of two fuzzy sets, where the union contains some information about the "overlap" between the membership functions, have been presented. For example, the following may be used to define the membership function resulting from the union fuzzy sets U^1 and U^2:

$$\mu_{U^1 \cup U^2}(u) = \min\{1, \mu_{U^1}(u) + \mu_{U^2}(u) : u \in \mathcal{U}\}. \tag{2.5}$$

Equation (2.5) may be used if one desires to quantify the fact the certainty of an element being characterized by the union of two fuzzy sets is greater than or equal to that the certainty of the element being characterized by each of the two fuzzy sets individually. In this situation, some of the information about the "membership function overlap" is retained within the resulting membership function. In this thesis Definition 2.8 will always be used since it is the most common definition of the fuzzy set union.

The following Definition 2.9 involving the intersection of two fuzzy sets may be viewed as a generalization of the intersection of two ordinary sets.

Definition 2.9 (Intersection) *The intersection of fuzzy sets U^1 and U^2 which are subsets of the universe of discourse $\mathcal{U}$ is a fuzzy set, denoted $U^1 \cap U^2$, with a membership function defined by*

$$\mu_{U^1 \cap U^2}(u) = min\{\mu_{U^1}(u), \mu_{U^2}(u) : u \in \mathcal{U}\}. \tag{2.6}$$

The intersection corresponds to the logic connective 'AND'.

Therefore, the resulting fuzzy set for the intersection of two fuzzy sets U^1 and U^2 contains all elements common to both fuzzy sets with a membership equal to the lowest membership that the element obtains in U^1 or U^2. Hence, this definition of the intersection of two fuzzy sets is consistent with the intersection of two ordinary sets in which the resulting set contains all elements common to both ordinary sets.

The intersection provides a means to define a fuzzy set for the situation where two or more linguistic values for a given linguistic variable are connected linguistically by 'AND'. For example, consider the linguistic statement "temperature is warm and hot". If the membership functions for warm and hot are given in Figure 2(b), then from Definition 2.9 the fuzzy set describing "temperature is warm and hot" has a membership function shown in Figure 5 below.

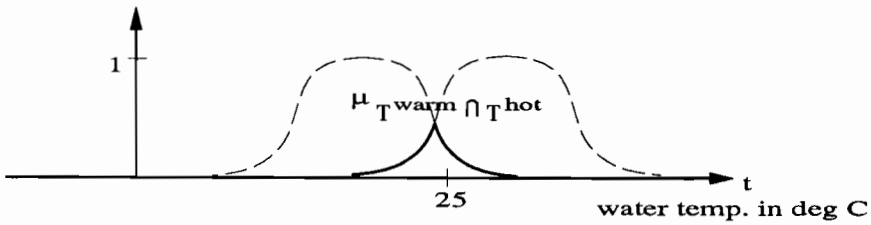


Figure 5: Membership function for the fuzzy set describing "temperature is warm and hot".

The intersection is rarely used in fuzzy system design since it is counter-intuitive to consider a linguistic variable assigned to two linguistic values on the same universe of discourse. This point is clearly illustrated by the fact that the membership function in Figure 5 describing the intersection between warm and hot temperatures never reaches full certainty or a truth value of 1.

Definition 2.10 (Complement) *The complement of the fuzzy set U which is a subset of the universe of discourse $\mathcal{U}$ is a fuzzy set, denoted $\neg U$, with a membership function defined by*

$$\mu_{\neg U}(u) = 1 - \mu_U(u) \text{ where } u \in \mathcal{U}, \tag{2.7}$$

and $\mu_U(u)$ is the membership function associated with fuzzy set U. The complement corresponds to the logic connective 'NOT'.

The complement provides a means to define a fuzzy set for the situation where a given linguistic variable is not described by a given linguistic value. For example, consider the linguistic statement "temperature is not warm". If the membership function for warm is given in Figure 2(b), then from Definition 2.10 the fuzzy set describing "temperature is not warm " has a membership function shown in Figure 6 below.

It should be noted that union, intersection, and complement are operations performed on fuzzy sets which are subsets of a single universe of discourse. However, generally in fuzzy systems design it is necessary to perform operations on fuzzy sets

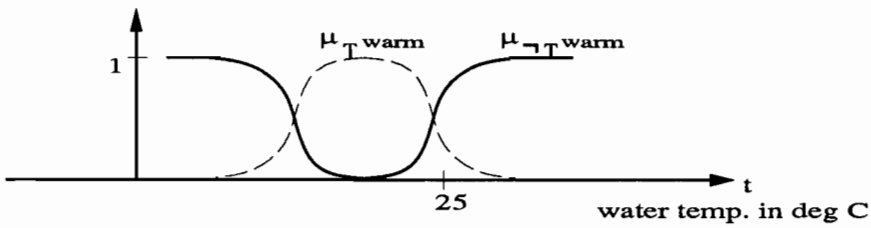


Figure 6: Membership function for the fuzzy set describing "temperature is not warm".

that are subsets of different universes of discourse. This characteristic is true in the case of a *fuzzy implication* discussed in the next subsection.

2.1.3 Fuzzy Implications, Relations, and Inference

In this subsection we will discuss how fuzzy set theory is used to quantify the linguistic *condition* $\rightarrow$ *action* rules to create fuzzy implications. This will be followed by a discussion of the operations which are typically employed to deal with such *fuzzy implications.*

Based on Definition 2.5 for a fuzzy set, we can now view the *condition* $\rightarrow$ *action* rules as *fuzzy implications*; in which the elements of the antecedent and the elements in the consequence can be viewed as fuzzy sets. For example, assume we are given the following *condition* $\rightarrow$ *action* rule in MISO form:

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{U}_1^j \textbf{ and } \tilde{u}_2 \text{ is } \tilde{U}_2^k \textbf{ and}, \ldots, \textbf{and } u_r \text{ is } \tilde{U}_r^l \textbf{ Then } y \text{ is } \tilde{Y}^m$$

As previously mentioned, these linguistic control rules are generally created by a control engineer or expert human operator. However fuzzy set theory provides a means with which to quantify this knowledge. In this case, we can define the fuzzy sets:

$$U_1^j = \{(u_1, \mu_{U_1^j}(u_1)) : u_1 \in \mathcal{U}_1\},$$
$$U_2^k = \{(u_2, \mu_{U_2^k}(u_2)) : u_2 \in \mathcal{U}_2\},$$
$$\vdots$$
$$U_r^l = \{(u_r, \mu_{U_r^l}(u_m)) : u_r \in \mathcal{U}_r\},$$
$$Y^m = \{(y, \mu_{Y^m}(y)) : y \in \mathcal{Y}\}.$$

Therefore, given the MISO *condition* $\rightarrow$ *action* rules and the fuzzy sets U_1^j, U_2^k, . . . , U_r^l, and Y^m above which quantify the antecedent and consequence of the *condition* $\rightarrow$ *action*, we may formally define a "fuzzy implication".

Definition 2.11 (Fuzzy Implication) *The condition* $\rightarrow$ *action rule in MISO form can be expressed as the fuzzy implication:*

$$\textbf{If } U_1^j \textbf{ and } U_2^k \textbf{ and, . . . ,and } U_r^l \textbf{ Then } Y^m,$$

where the fuzzy sets U_1^j, U_2^k, . . . , U_r^l, *and* Y^m *are defined above.*

Therefore, fuzzy set U_1^j is associated with the linguistic statement "$\tilde{u}_1$ is $\tilde{U}_1^j$". In practical situations, the fuzzy set U_1^j is often referred to by the linguistic statement with which it is associated. For example, if fuzzy set T^{warm} is associated with the linguistic statement "temperature is warm" then one often says that T^{warm} is the fuzzy set "temperature is warm". Likewise, the universe of discourse is often referred to by the linguistic variable. For instance, in the previous example, T^{warm} may be considered a subset of the universe of discourse $\mathcal{T}$ in which $\mathcal{T}$ may be referred to as the universe of discourse "temperature".

In order to properly characterize the fuzzy implication in a mathematical framework we must first define the "cartesian product" of fuzzy sets that are subsets of different universes of discourse.

Definition 2.12 (Cartesian Product) *If $U_1, U_2, ..., U_r$ are fuzzy sets defined as subsets of the universes of discourse $\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_r$, respectively, the cartesian product is a fuzzy set, denoted $U_1 \times U_2 \times ... \times U_r$, with a membership function defined by*

$$\mu_{U_1 \times U_2 \times ... \times U_r}(u_1, u_2, ..., u_r) = min\{\mu_{U_1}(u_1), \mu_{U_2}(u_2), ..., \mu_{U_r}(u_r)\}, \qquad (2.8)$$

where $u_1, u_2, ..., u_r$ are elements of $\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_r$ respectively.

Note that the membership function resulting from a fuzzy cartesian product is sometimes expressed as

$$\mu_{U_1 \times U_2 \times ... \times U_r}(u_1, u_2, ..., u_r) = \mu_{U_1}(u_1) \cdot \mu_{U_2}(u_2) \cdot ... \cdot \mu_{U_1}(u_r). \qquad (2.9)$$

The choice between the use of Equation (2.8) or (2.9) is generally based on the problem being considered. However, in fuzzy system applications Equation (2.8) is nearly always used for ease of computation and that is what we will assume here unless otherwise stated. In fuzzy system design, the cartesian product is most commonly encountered when developing a fuzzy relation as defined in Definition 2.13 below.

Now that we have defined the cartesian product, we can now transform the fuzzy implication defined in Definition 2.11 to a fuzzy set that is commonly called a "fuzzy relation". Definition 2.13 below describes how this fuzzy relation is obtained.

Definition 2.13 (Fuzzy Relation) *Assume we are given a fuzzy implication in MISO form expressed with generic elements as*

$$\textbf{If } U_1^j \textbf{ and } U_2^k \textbf{ and, . . . ,and } U_r^l \textbf{ Then } Y^m.$$

A fuzzy relation denoted by $R^{j,k,...,l}$ for the given fuzzy implication is defined to be the fuzzy set that is a subset of the universe of discourse $\mathcal{U}_1 \times \mathcal{U}_2 \times ... \times \mathcal{U}_r \times \mathcal{Y}^m$ and is expressed by

$$R^{j,k,...,l} = U_1^j \times U_2^k \times ... \times U_r^l \times Y^m, \tag{2.10}$$

where $\times$ denotes the cartesian product as defined in Definition 2.12.

Thus if we assume Equation (2.8) is used to implement the cartesian product, the membership function of $R^{j,k,...,l}$ is expressed as

$$\mu_{R^{j,k,...,l}}(u_1, u_2, ..., u_r, y) = \\ min\{\mu_{U_1^j}(u1), \mu_{U_2^k}(u_2), ..., \mu_{U_r^l}(u_r), \mu_{Y^m}(y)\}, \tag{2.11}$$

where $(u_1, u_2, ..., u_r, y) \in \mathcal{U}_1 \times \mathcal{U}_2 \times ... \times \mathcal{U}_r \times \mathcal{Y}^m$.

For example, consider the following *condition* $\rightarrow$ *action* rule relating the linguistic variables temperature and pressure:

If temperature is warm **Then** pressure is high.

We can define fuzzy sets T^{warm} associated with the linguistic statement "temperature is warm" and P^{high} associated with the linguistic statement "pressure is high". Then we can create the fuzzy implication

$$\textbf{If } T^{warm} \textbf{ Then } P^{high},$$

which can be represented by the fuzzy relation R expressed as

$$R = T^{warm} \times P^{high}. \tag{2.12}$$

If we assume that the membership functions for T^{warm} and P^{high} are shown in Figure 7(a) and (b), respectively, and Equation (2.8) is used, to implement the cartesian product, then the membership function for R is given by

$$\mu_R(t,p) = \min\{\mu_{Twarm}(t), \mu_{Phigh}(p)\} \tag{2.13}$$

which is shown in Figure 7(c). Notice that since the "min" operator is used given a temperature t we can be no more certain about the value of the pressure then we are about the the value of the temperature.

Consider again the MISO fuzzy implication in modus ponens form expressed as

$$\textbf{If } U_1^j \textbf{ and } U_2^k \textbf{ and, . . . ,and } U_r^l \textbf{ Then } Y^m$$

where $U_1^j, U_2^k, ..., U_r^l$ denote the fuzzy sets associated with the inputs and Y^m denotes the fuzzy set associated with the output. Note that the fuzzy relation $R^{j,k,...,l}$ which describes this fuzzy implication has a membership function described by Equation (2.11). However in real applications, the variables denoted by $u_1, u_2, ..., u_r$ are generally known values since they are inputs to the fuzzy system. Using the intuitive meaning of the fuzzy implications, **If** U_1^j **and** U_2^k **and**, . . . , U_r^l **then** Y^m, it would be contrary to nature if Y^m obtained a higher degree of truth than U_1^j or U_2^k or, . . . , U_r^l, which as mentioned is expressed in the fuzzy relation of these fuzzy sets. Therefore, given a fuzzy relation and the specific values of the inputs $u_1, u_2, ..., u_r$ we may determine the *implied fuzzy set* defined below.

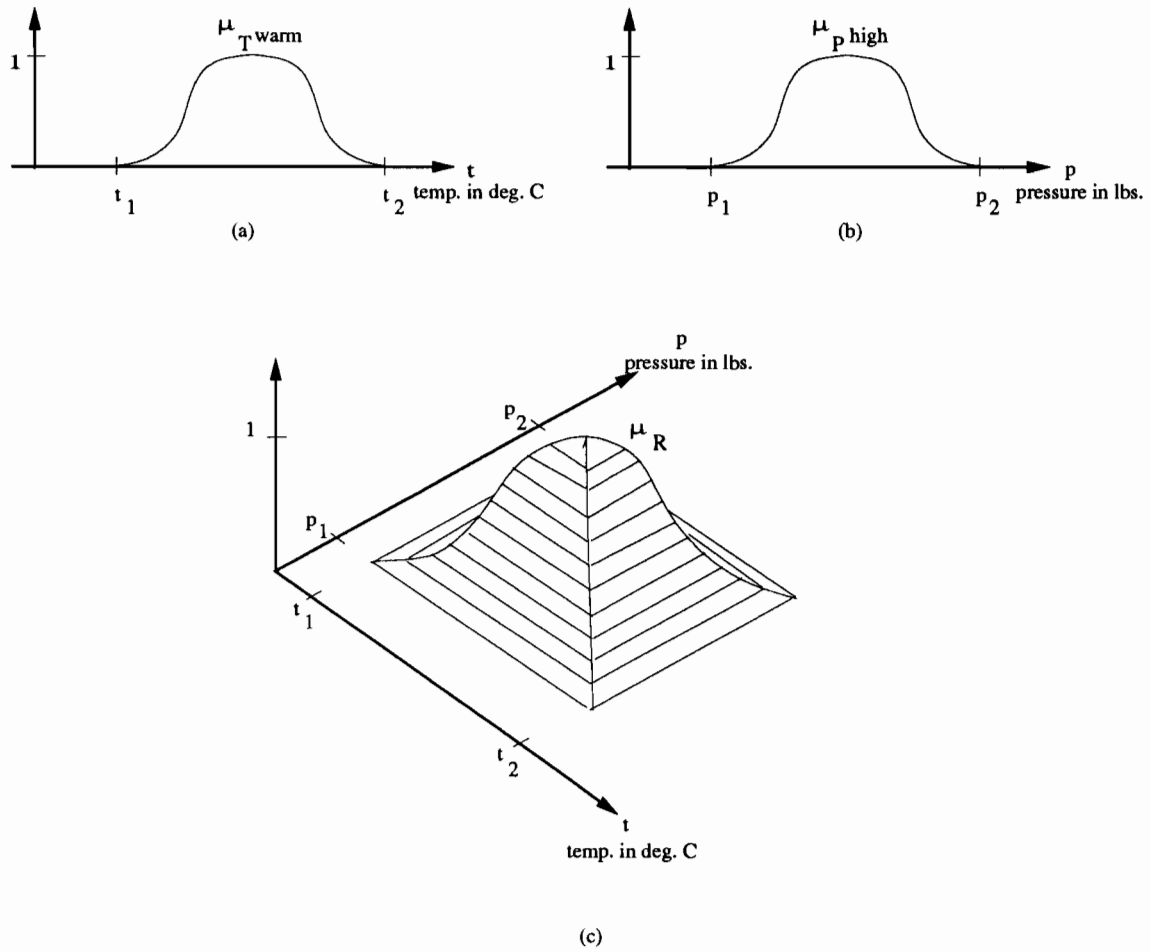


Figure 7: (a) Membership function for "temperature is warm"; (b) Membership function for "pressure is high"; and (c) Membership function for the fuzzy relation $R = T^{warm} \times P^{high}$.

Definition 2.14 (Implied Fuzzy Set) *If $u_1(t), u_2(t), ..., u_r(t)$ denote the values of the inputs at time t, then the implied fuzzy set at time t denoted $\hat{Y}^{j,k,...,l}(t)$ defined on $\mathcal{Y}$ for the output y has a membership function given as*

$$\mu_{\hat{Y}^{j,k,...,l}(t)}(y) = \mu_{R^{j,k,...,l}}(u_1(t), u_2(t), ..., u_r(t), y) = \\ min\{\mu_{U_1^j}(u_1(t)), \mu_{U_2^k}(u_2(t)), ..., \mu_{U_r^l}(u_r(t)), \mu_{Y^m}(y)\}. \quad (2.14)$$

The implied fuzzy set $\hat{Y}^{j,k,...,l}(t)$ specifies the certainty that the crisp fuzzy system output at time t, denoted $y(t)$, should take on specific values $y \in \mathcal{Y}$ given a linguistic MISO rule with associated fuzzy sets and crisp input values $u_i(t)$. An example of this process is illustrated following Definition 2.17 for the "sup-star compositional rule of inference". Likewise as will be shown later in this section, the membership function $\mu_{\hat{Y}^{j,k,...,l}(t)}(y)$ defined in Equation (2.14) is simply a reduced form of Zadeh's compositional rule of inference. Thus the same intuitive result in Equation (2.14) will be obtained by using Zadeh's compositional rule of inference. See Definition 2.17 for details of Zadeh's compositional rule of inference.

In the previous discussion, by using an intuitive procedure, we have shown how to find the implied fuzzy set for a generalized modus ponens fuzzy implication, assuming of course that the inputs to the system are crisp values and not fuzzy sets. However, the following fuzzy set operations referred to as the "triangular norm" and the "sup-star composition"are intended as a prelude to a more generalized method of determining this implied fuzzy set.

Definition 2.15 (Triangular Norm) *The triangular norm, denoted $*$, is an operation defined for two membership functions which maps $[0,1] \times [0,1]$ to $[0,1]$, i.e., $* : [0,1] \times [0,1] \rightarrow [0,1]$. In fuzzy system design, the most commonly used triangular norms include the minimum and algebraic product defined below as*

$$\textit{minimum:} \qquad x * y = \min(x, y) \tag{2.15}$$

$$\textit{algebraic product:} \qquad x * y = x \cdot y \tag{2.16}$$

Note that the name "triangular norm" is used since all properties of a norm are applicable over the domain $[0,1]$.

Definition 2.16 (Sup-Star Composition) *Assume that U and R are fuzzy relations defined on $\mathcal{U}_1 \times \mathcal{U}_2 \times, ..., \times \mathcal{U}_r$ and $\mathcal{U}_1 \times \mathcal{U}_2 \times, ..., \times \mathcal{U}_r \times \mathcal{Y}$, respectively, the "sup-star composition" of U and R is a fuzzy relation denoted by $U \circ R$ and is defined as*

$$U \circ R =$$

$$\{((u_1, u_2, ..., u_r, y), \sup_{u_1, u_2, ..., u_r} \mu_U(u_1, u_2, ..., u_r) * \mu_R(u_1, u_2, ..., u_r, y)) : (u_1, u_2, ..., u_r) \in \mathcal{U}_1 \times \mathcal{U}_2 \times, ..., \times \mathcal{U}_r \text{ and } y \in \mathcal{Y}\} \tag{2.17}$$

The $*$ symbol is used to denote any triangular norm.

Let us consider a more generalized version of the MISO fuzzy implication. This version commonly referred to as the generalized modus ponens form is expressed with generic arguments as

Fuzzy Implication: If U_1^j **and** U_2^k **and, . . . ,and** U_r^l **Then** Y^m

Premise: $\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_r(t)$

Consequence: $\hat{Y}^{j,k,...,l}(t)$,

where $\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_r(t)$ are fuzzy sets describing the actual input at time t and $\hat{Y}^{j,k,...,l}(t)$ is the implied fuzzy set associated with output y at time t given the fuzzy implication and premise above. Note that typically the input to a fuzzy system is a precise crisp value. Therefore, these crisp inputs must be "fuzzified" to obtain $\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_r(t)$. In order to fuzzify the input, we may assume that the membership function $\mu_{\hat{U}_1(t)}(u_1)$ for fuzzy set $\hat{U}_1(t)$ is equal to zero for all elements of the universe of discourse $\mathcal{U}_1$ except at the specific value $u_1(t)$ of the input at time t in which the membership function takes on a truth value of 1. Likewise, membership functions for all other fuzzified inputs denoted, $\hat{U}_2(t), ..., \hat{U}_r(t)$, are defined in a similar fashion.

Given the definitions of the triangular norm and the sup-star composition above, we define the "sup-star compositional rule of inference" in Definition 2.17. The sup-star compositional rule of inference provides a formal mathematical framework for quantifying fuzzy implications expressed in the generalized modus ponens form.

Definition 2.17 (Sup-Star Compositional Rule of Inference) *Given a MISO fuzzy implication in the generalized modus ponens form expressed with generic arguments as*

Fuzzy Implication: If U_1^j and U_2^k and, . . . ,and U_r^l Then Y^m

Premise: $\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_r(t)$

Consequence: $\hat{Y}^{j,k,...,l}(t)$,

where $\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_r(t)$ *are fuzzy sets describing the actual input at time t and* $\hat{Y}^{j,k,...,l}(t)$ *is the implied fuzzy set associated with output y at time t given the fuzzy implication and premise above. We may define fuzzy relation* $\hat{U}(t) = \hat{U}_1(t) \times \hat{U}_2(t) \times ... \times \hat{U}_r(t)$ *such that* $\hat{U}(t)$ *is defined on universe of discourse* $\mathcal{U}_1 \times \mathcal{U}_2 \times, ..., \times \mathcal{U}_r$ *and fuzzy relation* $R^{j,k,...,l} = U_1^j \times U_2^k \times ... \times U_r^l \times Y^m$ *such that* $R^{j,k,...,l}$ *is defined on the universe of discourse* $\mathcal{U}_1 \times \mathcal{U}_2 \times, ..., \times \mathcal{U}_r \times \mathcal{Y}$. *The "sup-star compositional rule of inference" asserts that the fuzzy set* $\hat{Y}^{j,k,...,l}(t)$ *defined on universe of discourse* $\mathcal{Y}$ *implied by* $\hat{U}(t)$ *is given by*

$$\hat{Y}^{j,k,...,l}(t) = \hat{U}(t) \circ R^{j,k,...,l} \tag{2.18}$$

where $\circ$ *denotes the sup-star composition of* U *and* $R^{j,k,...,l}$.

If the triangular norm denoted by $*$ used to employ the sup-star composition in Equation (2.18) is the minimum operator, then the "sup-star compositional rule of inference" reduces to what is commonly referred to as "Zadeh's compositional rule of inference". See [89, 90] for more details.

Consider the case of Zadeh's compositional rule of inference. Note that if Equation (2.8) is employed to implement the cartesian product, the membership function for fuzzy set $\hat{U}(t)$ defined in Definition 2.17 is expressed as

$$\mu_{\hat{U}(t)}(u_1, u_2, ..., u_r) = \min\{\mu_{\hat{U}_1(t)}(u_1), \mu_{\hat{U}_2(t)}(u_2), ..., \mu_{\hat{U}_r(t)}(u_r)\}. \tag{2.19}$$

Likewise, the membership function for the fuzzy relation $R^{j,k,...,l}$ defined in Definition 2.17 is expressed as

$$\mu_{R^{j,k,...,l}}(u_1, u_2, ..., u_r) = \min\{\mu_{U_1^j}(u_1), \mu_{U_2^k}(u_2), ..., \mu_{U_r^l}(u_r), \mu_{Y^m}(y)\}. \quad (2.20)$$

By employing "Zadeh's compositional rule of inference", we have the implied fuzzy set $\hat{Y}^{j,k,...l}$ expressed as

$$\hat{Y}^{j,k,...l}(t) = \hat{U}(t) \circ R^{j,k,...,l} \quad (2.21)$$

where the membership function is given by

$$\mu_{\hat{Y}^{j,k,...,l}(t)}(y) = \sup_{u_1, u_1, ..., u_r} \min\{\mu_{\hat{U}(t)}(u_1, u_1, ..., u_r), \mu_{R^{j,k,...,l}}(u_1, u_1, ..., u_r, y)\}. \quad (2.22)$$

Substituting the membership functions for $\hat{U}(t)$ and $R^{j,k,...,l}$, we obtain

$$\mu_{\hat{Y}^{j,k,...,l}(t)}(y) = \sup_{u_1, u_1, ..., u_r} \min\{\min\{\mu_{\hat{U}_1(t)}(u_1), \mu_{\hat{U}_2(t)}(u_2), ..., \mu_{\hat{U}_r(t)}(u_r)\}, \min\{\mu_{U_1^j}(u_1), \mu_{U_2^k}(u_2), ..., \mu_{U_r^l}(u_r), \mu_{Y^m}(y)\}\}. \quad (2.23)$$

As previously mentioned, typically the input to a fuzzy system is a precise crisp value and must be "fuzzified" to obtain $\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_r(t)$. Under this assumption, the fuzzy set denoted $\hat{U}_i(t)$ has a membership function defined such that $\mu_{\hat{U}_i(t)}(u_i) = 0 \ \forall \ u_i \in \mathcal{U}_i$, except at the specific value $u_i(t)$ of the input at time t in which the membership function takes on a truth value of 1. Therefore, if the inputs to the fuzzy system are crisp values, Equation (2.23) reduces to

$$\mu_{\hat{Y}^{j,k,...l}(t)}(y) = \min\{\mu_{U_1^j}(u_1(t)), \mu_{U_2^k}(u_2(t)), ..., \mu_{U_r^l}(u_r(t)), \mu_{Y^m}(y)\}, \quad (2.24)$$

where $u_1(t), u_2(t), ..., u_r(t)$ are assumed to be the precise values of the inputs at time t. Thus we obtain the same result as Equation (2.14).

Consider an example fuzzy system which contains a two input - single output *condition* $\rightarrow$ *action* rule expressed as

If "the pressure is low" **and** "the relative humidity is high"

Then "the percent chance of rain is high".

We may define fuzzy sets P^{Low}, H^{high}, and R^{high} associated with linguistic statements "the pressure is low", "the relative humidity is high", and "the percent chance of rain is high", respectively, whose membership functions are shown in Figure 8 below. Therefore, the linguistic *condition* $\rightarrow$ *action* rule can be converted to the fuzzy implication expressed as

If P^{Low} **and** H^{high} **Then** R^{high}.

Assume that the numeric value of the pressure at time t is denoted by $p(t)$ such that $\mu_{Plow}(p(t)) = 0.5$. Likewise, assume that the numeric value of the relative humidity at time t is denoted by $h(t)$ such that $\mu_{Hhigh}(h(t)) = 0.8$. By employing Equation (2.24) for the reduced form of Zadeh's compositional rule, we can find the membership function of the implied fuzzy set $\hat{R}^{low,high}(t)$ which characterizes the output of the fuzzy system. This membership function is expressed as

$$\mu_{\hat{R}^{low,high}(t)}(r) = \min\{\mu_{Plow}(p(t)), \mu_{Hhigh}(h(t)), \mu_{Rhigh}(r)\}, \tag{2.25}$$

which by substituting the values of $\mu_{Plow}(p(t))$ and $\mu_{Hlow}(h(t))$ reduces to

$$\mu_{\hat{R}^{low,high}(t)}(r) = \min\{0.5, 0.8, \mu_{Rhigh}(r)\}. \tag{2.26}$$

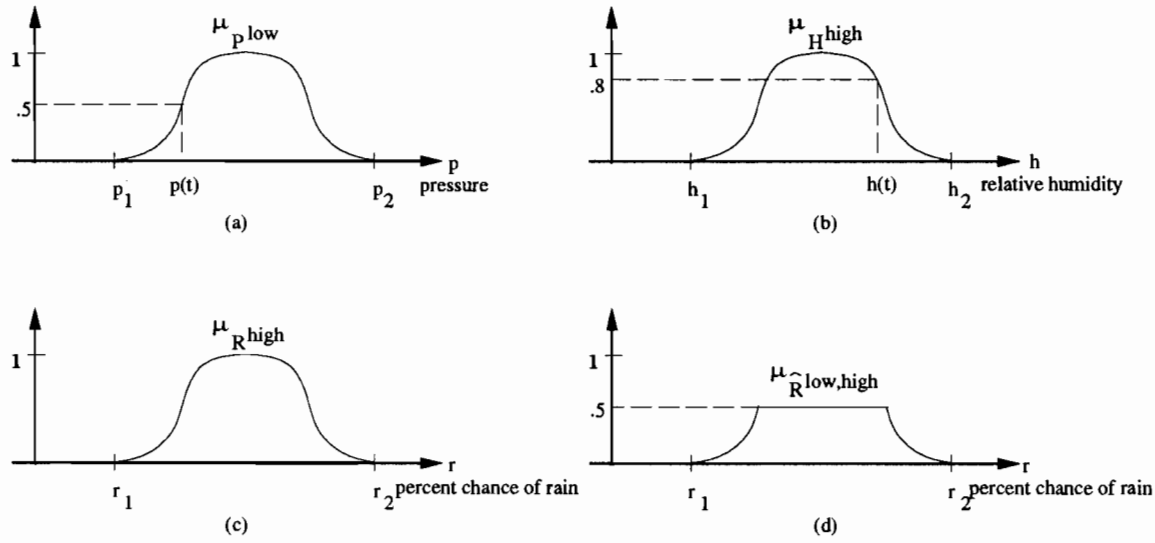


Figure 8: (a) Membership function for "the pressure is low"; (b) Membership function for "the relative humidity is high"; (c) Membership function for "the percent chance of rain is high"; and (d) membership function for the implied output if the input to the system are crisp values $p(t)$ and $h(t)$.

Figure 8(d) clearly illustrates the resulting membership function for the implied fuzzy set $\hat{R}^{low,high}(t)$ obtained from Equation (2.26).

2.1.4 Fuzzy Systems Composed of Many Fuzzy Implications

In order to generate an action for every possible set of conditions, many fuzzy implications often exist in a fuzzy system. Typically in fuzzy system design, a fuzzy implication exists for every possible combination of fuzzy sets describing the inputs to the fuzzy system. This property of designing rules such that an action must exist for every possible combination of fuzzy sets describing input conditions is often referred to as *completeness.* For example, assume we are given a set of MISO fuzzy implications,

$$\textbf{If } U_1^j \textbf{ and } U_2^k \textbf{ and, . . . , and } U_r^l \textbf{ Then } Y^m,$$

such that the total number of fuzzy sets indexed by j is N_1, by k is N_2, . . . , by l is N_r. In order to insure completeness $N_1 \cdot N_2 \cdot ... \cdot N_r$ rules are required. If we further assume that $N_1^- \leq j \leq N_1^+$, $N_2^- \leq k \leq N_2^+$, . . . , and $N_r^- \leq l \leq N_r^+$, then we may let the overall fuzzy relation denoted by R be expressed by the union of all the individual fuzzy implications such that

$$R = \bigcup_{j=N_1^-,k=N_2^-,...,l=N_r^-}^{N_1^+,N_2^+,...,N_r^+} R^{j,k,...,l}, \tag{2.27}$$

where the membership function of R may be expressed as

$$\mu_R(u_1, u_2, ..., u_r, y) = \max_{j,k,...,l} \{\min\{\mu_{U_1^j}(u_1), \mu_{U_2^k}(u_2), ..., \mu_{U_r^l}(u_r), \mu_{Y^m}(y)\}\}. \tag{2.28}$$

Note that the union of the fuzzy implication is due to the fact that *condition* $\rightarrow$ *action* rules could be expressed as

If condition 1 **Then** action 1

Or

If condition 2 **Then** action 2

Or . . .

where the 'OR' connective is implied. If we let $\hat{U}(t)$ denote the cartesian product of fuzzified inputs at time t denoted $\hat{U}_1(t), \hat{U}_2(t), ..., \hat{U}_r(t)$, then by employing Zadeh's compositional rule of inference, the overall implied fuzzy set $\hat{Y}(t)$ obtained from fuzzified inputs is given by

$$\hat{Y}(t) = \hat{U}(t) \circ R. \tag{2.29}$$

By performing similar steps used to obtain Equation (2.24), it is easily determined that the membership function for the implied fuzzy set $\hat{Y}(t)$ is

$$\mu_{\hat{Y}(t)}(y) = \mu_R(u_1(t), u_2(t), ..., u_r(t), y) = \max_{j,k,...,l} \min\{\mu_{U_1^j}(u_1(t)), \mu_{U_2^k}(u_2(t)), ..., \mu_{U_r^l}(u_r(t)), \mu_{Y^m}(y)\}, \quad (2.30)$$

where $u_1(t), u_2(t), ..., u_r(t)$ denote the precise numeric values of the inputs at time t. In effect, the fuzzy set $\hat{Y}(t)$ and the corresponding membership function $\mu_{\hat{Y}(t)}(y)$ defines the output of the fuzzy system. However, this output is defined in terms of a fuzzy set, which is not of practical use for most real world applications. Therefore, it becomes necessary to "defuzzify" the fuzzy set $\hat{Y}(t)$ to obtain a crisp value denoted $y(t)$ suitable for application. Several techniques for defuzzifying a fuzzy set will be discussed in the subsection in this chapter titled Defuzzification Interface.

2.1.5 Summary of Notation

In preparing this thesis, we have tried to maintain a consistent notation scheme. To clarify this notation scheme, in Table 1, we have compiled a summary of the notation that will be used throughout the the rest of this thesis.

2.2 Characteristics of Membership Functions

In general, no restrictions are placed on the membership function defined for a given fuzzy set. However, in fuzzy system design, several characteristic assumptions are generally made about the shape of the membership function and the way the membership functions are distributed along the universe of discourse. Therefore, the purpose of this subsection is to define some of the basic terminology which is used to characterize membership functions.

Table 1: Summary of the notation used throughout this thesis.

Example Notation[1]	Denotes
u_i	A possible numeric value of the i^{th} input to a fuzzy system.
$\mathcal{U}_i$	Universe of discourse for u_i.
$\tilde{u}_i$	Linguistic variable associated with u_i.
$\tilde{U}_i^j$	The j^{th} linguistic values associated with $\tilde{u}_i$.
$\tilde{U}_i$	The set of linguistic values associated with $\tilde{u}_i$ such that $\tilde{U}_i = \{\tilde{U}_i^j\}$.
U_i^j	The fuzzy set associated with the linguistic statement "$\tilde{u}_i$ is $\tilde{U}_i^j$".
U_i	The set of fuzzy sets associated with the linguistic statement "$\tilde{u}_i$ is $\tilde{U}_i^j$" such that $U_i = \{U_i^j\}$.
$\mu_{U_i^j}$	The membership function associated with the linguistic statement "$\tilde{u}_i$ is $\tilde{U}_i^j$".
μ_{U_i}	The set of membership functions associated with the linguistic statement "$\tilde{u}_i$ is $\tilde{U}_i^j$" such that $\mu_{U_i} = \{\mu_{U_i^j}\}$.
$\hat{U}_i(t)$	Fuzzy set describing input to the fuzzy system at time t.
$\hat{Y}^k(t)$	An implied fuzzy set resulting from the k^{th} fuzzy implication and fuzzified inputs to the fuzzy system.
Φ	A null fuzzy set.

[1] Subscripts can be removed if only one input or output is associated with a given lower case letter. Likewise, superscripts can be removed if only one linguistic value is assigned to a given linguistic variable.

Definition 2.18 (Normal) *Given a fuzzy set U_i^j that is a subset of the universe of discourse $\mathcal{U}_i$, the membership function $\mu_{U_i^j}$ associated with the fuzzy set U_i^j is called normal if*

$$\max_{u_i} \mu_{U_i^j}(u_i) = 1, \tag{2.31}$$

for some $u_i \in \mathcal{U}_i$.

Definition 2.18 can be paraphrased as follows: A membership function is said to be *normal* if at least one element in the universe of discourse has a truth value of 1. Figure 9 below illustrates the difference between normal and non-normal membership functions. Membership functions used to describe linguistic terms are nearly always normal since intuitively some value along the universe of discourse must obtain full membership; otherwise, the linguistic description would probably not be used.

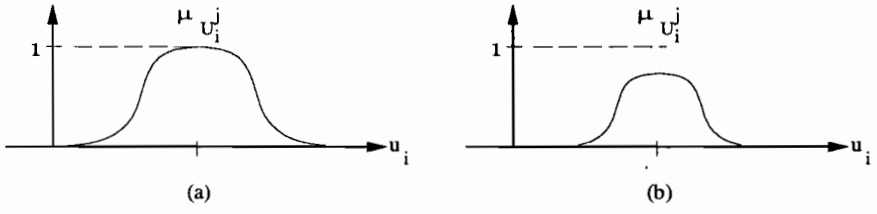


Figure 9: (a) Normal membership function and (b) non-normal membership function.

Definition 2.19 (Convex) *Given a fuzzy set U_i^j that is a subset of the universe of discourse $\mathcal{U}_i$, the membership function $\mu_{U_i^j}$ associated with the fuzzy set U_i^j is said to be convex if for any two elements $u_{i,1}, u_{i,2} \in \mathcal{U}_i$, and $0 \leq \lambda \leq 1$ the following is true.*

$$\mu_{U_i^j}(\lambda u_{i,1} + (1-\lambda)u_{i,2}) \geq \min\{\mu_{U_i^j}(u_{i,1}), \mu_{U_i^j}(u_{i,2})\} \tag{2.32}$$

where $u_i \in \mathcal{U}_i$.

Instances of convex and non-convex membership functions are shown in Figure 10 below. Membership functions used to describe linguistic values are nearly always convex since the linguistic description generally applies to a given region of the universe of discourse and becomes less applicable, or has a lower truth value, the further away from this region.

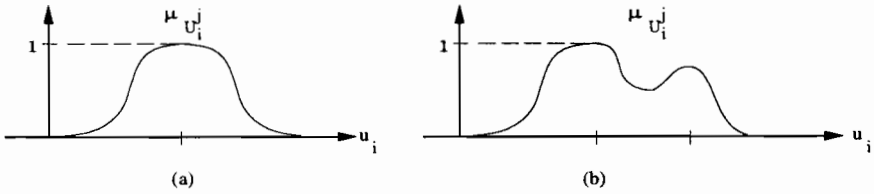


Figure 10: (a) Convex membership function and (b) non-convex membership function.

Definition 2.20 (Symmetric) *Given a fuzzy set denoted* U_i^j *that is a subset of the universe of discourse* $\mathcal{U}_i$ *with a membership function denoted* $\mu_{U_i^j}$, *membership function* $\mu_{U_i^j}$ *is said to be symmetric if it is symmetric about a "center value"*

Instances of symmetric and non-symmetric membership functions are shown in Figure 11 below and in Figure 10 above. In fuzzy systems, membership functions used to describe linguistic values are often symmetric. However as will be shown later in Section 2.4, this constraint may be too restrictive since in some cases, the symmetric constraint could reduce benefits that may be obtained by using a fuzzy system as a controller.

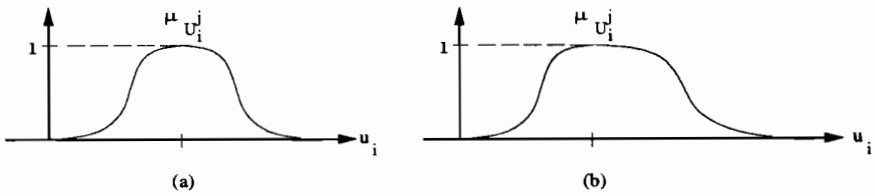


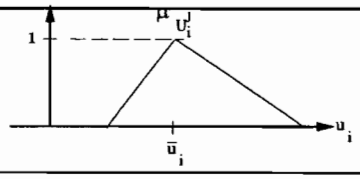
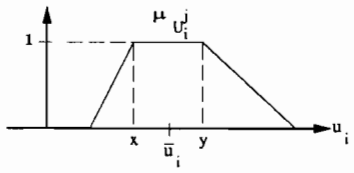
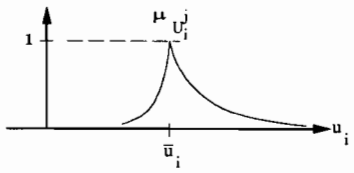
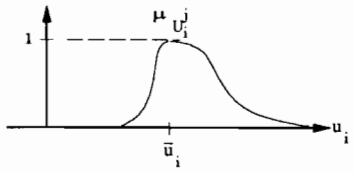
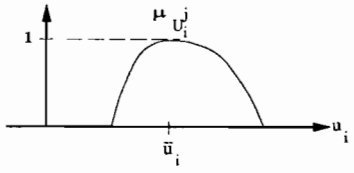
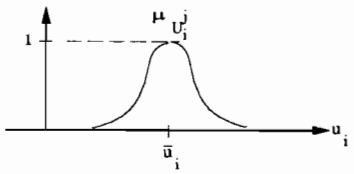
Figure 11: (a) Symmetric membership function and (b) non-symmetric membership function.

In fuzzy system design, several types of membership functions are commonly used. Therefore, in Table 2, we have compiled a set of membership functions that are typically encountered in the literature on fuzzy systems.

Coincidentally, the membership functions shown in Table 2 are normal and convex. The variables a, b, and σ given in the mathematical description column for each membership function in Table 2 are used to define the "precision" associated with the linguistic statement that it represents. For example, reducing a, b, and σ results in a more narrow membership function or a more precise linguistic statement. Likewise, increasing a, b, and σ results in a wider membership function of a weaker linguistic statement. It should be noted that the membership functions in Table 2 can be made symmetric if variables a and b are set equal; however, the Gaussian membership function is always symmetric.

Up to this point we have only considered characteristics for membership functions related to the shape of the function. However, in typical fuzzy applications, more than one fuzzy set is associated with a given universe of discourse. Therefore, in some situations, we may want to impose some restrictions on the way that the membership functions are distributed along the universe of discourse. This concept of distributing fuzzy sets along the universe of discourse is often referred to

Table 2: Commonly used membership functions

Funtion Name	Example	Mathematical Description
Triangle		$\mu_{U_i^j} = \begin{cases} \max\{0, (1-(\lvert u_i - \bar{u}_i^j \rvert)/a)\} & ; u_i \geq \bar{u}_i \\ \max\{0, (1-(\lvert u_i - \bar{u}_i^j \rvert)/b)\} & ; u_i < \bar{u}_i \end{cases}$
Trapazoidal		$\mu_{U_i^j} = \begin{cases} 1 & ; x \leq u_i \leq y \\ \max\{0, (1-(\lvert u_i - y \rvert)/a)\} & ; u_i > y \\ \max\{0, (1-(\lvert u_i - x \rvert)/b)\} & ; u_i < x \end{cases}$
Exponential		$\mu_{U_i^j} = \begin{cases} e^{-(\lvert u_i - \bar{u}_i^j \rvert)/a} & ; u_i \geq \bar{u}_i \\ e^{-(\lvert u_i - \bar{u}_i^j \rvert)/b} & ; u_i < \bar{u}_i \end{cases}$
Ratio		$\mu_{U_i^j} = \begin{cases} 1/(1+(\lvert u_i - \bar{u}_i^j \rvert)/a) & ; u_i \geq \bar{u}_i \\ 1/(1+(\lvert u_i - \bar{u}_i^j \rvert)/b) & ; u_i < \bar{u}_i \end{cases}$
Cosine		$\mu_{U_i^j} = \begin{cases} \cos((\lvert u_i - \bar{u}_i^j \rvert)/a) & ; \bar{u}_i < u_i < \bar{u}_i + a\pi/2 \\ \cos((\lvert u_i - \bar{u}_i^j \rvert)/b) & ; \bar{u}_i - b\pi/2 < u_i < \bar{u}_i \\ 0 & ; u_i > \bar{u}_i + a\pi/2 \\ 0 & ; u_i < \bar{u}_i - b\pi/2 \end{cases}$
Gaussian		$\mu_{U_i^j} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[u_i - \bar{u}_i^j]^2}{2\sigma^2}}$

as "fuzzy partitioning". For example, one restriction that is commonly imposed on the membership functions when partitioning the universe of discourse is expressed in Assumption 2.1 below.

Assumption 2.1 *Given a set of fuzzy sets* $U_i = \{U_i^j\}$ *associated with the set of membership functions* $\mu_{U_i} = \{\mu_{U_i^j}\}$, *respectively, such that* $U_i^j \subset \mathcal{U}_i$, *assume that*

$$\sum_j \mu_{U_i^j}(u_i) \leq 1 \qquad \forall\, u_i \in \mathcal{U}_i. \tag{2.33}$$

Assumption 2.1 can be interpreted to mean that no element of the universe of discourse belong to more than, as it were, one set. Thus, in order to maintain compatibility with classical crisp sets, where an element is classified by one and only one set, the following more stringent constraint is commonly applied when distributing membership functions along the universe of discourse.

Assumption 2.2 *Given a set of fuzzy sets* $U_i = \{U_i^j\}$ *associated with the set of membership functions* $\mu_{U_i} = \{\mu_{U_i^j}\}$, *respectively, such that* $U_i^j \subset \mathcal{U}_i$, *assume that*

$$\sum_j \mu_{U_i^j}(u_i) = 1 \qquad \forall\, u_i \in \mathcal{U}_i. \tag{2.34}$$

2.3 Fuzzy System Functional Architecture

Based on the previous discussion of mathematical fundamentals for fuzzy set theory, we can fill in some more details of the basic fuzzy system shown in Figure 1 to help show how fuzzy systems are implemented. Therefore, Figure 12 shows the basic configuration of a MISO fuzzy system, which comprises four principle components: a fuzzification interface, a knowledge base, an inference mechanism and a defuzzification interface.

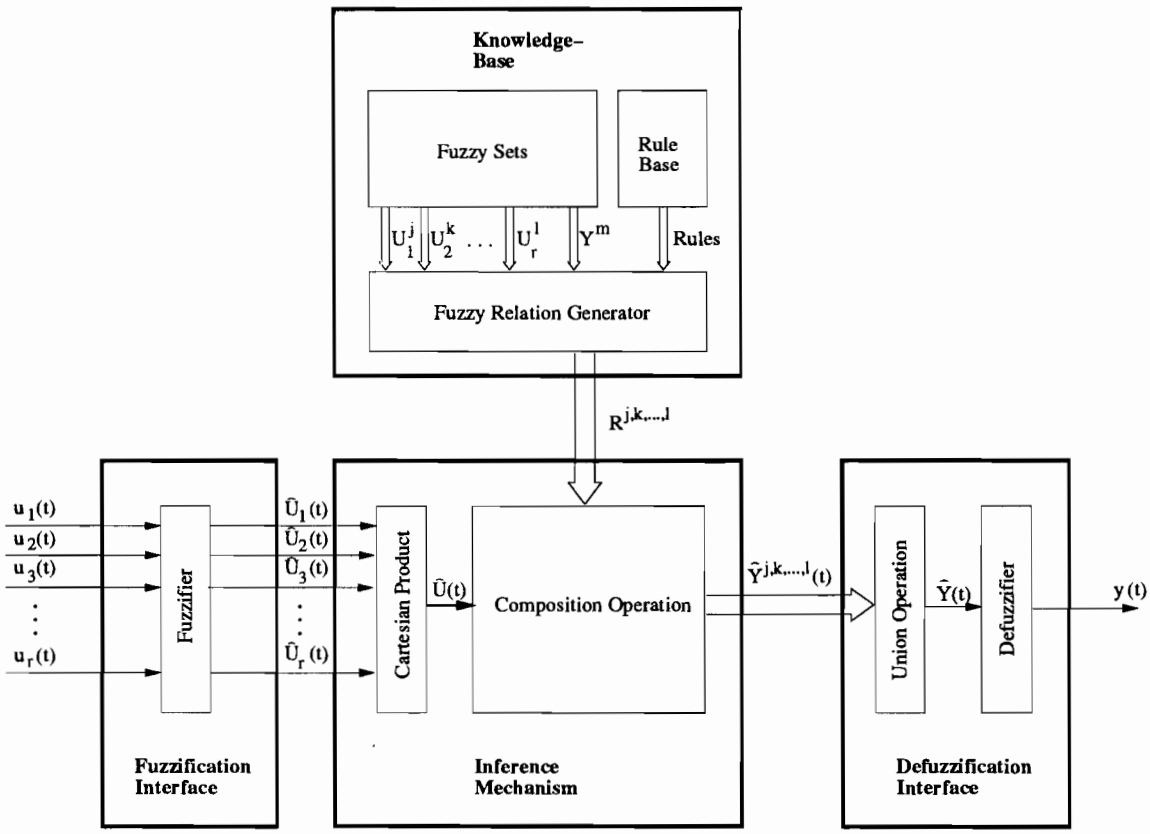


Figure 12: The basic configuration of a fuzzy system.

2.3.1 Fuzzification Interface

The primary function of the fuzzification interface involves measuring the crisp fuzzy system inputs, denoted u_i, and converting them to a fuzzy set denoted $\hat{U}_i$ defined over the universe of discourse $\mathcal{U}_i$. This process of fuzzification is necessary since fuzzy inference will take place on the inputs and the mathematical fuzzy set operations employed for fuzzy inference are defined only for fuzzy sets and not a combination of numeric values and fuzzy sets.

The fuzzification operation has the effect of transforming crisp data into fuzzy sets. This transformation may be represented symbolically as,

$$\hat{U}_i(t) = fuzzifier(u_i(t)), \tag{2.35}$$

where $u_i(t)$ is the crisp numeric value of the input at time t. The crisp input $u_i(t)$ is converted to a fuzzy set $\hat{U}_i(t)$ by assigning the membership function $\mu_{\hat{U}_i(t)}(u_i)$ equal to zero for all $u_i \in \mathcal{U}_i$ except at the current crisp input value $u_i(t)$ where the membership function obtains a value of 1. Therefore, the fuzzy set $\hat{U}_i$ can be interpreted to mean the input can assume any value in $\mathcal{U}_i$; however, we are totally certain that the input is the current crisp input $u_i(t)$ and totally certain that the input is not any other element of the universe of discourse.

For example, if we assume that the input u_i takes on a value $u_i(t)$ at time t, then the fuzzy set $\hat{U}_i(t) \subset \mathcal{U}_i$ has a membership function shown in Figure 13 below.

Although in terms of basic mathematical operations the fuzzification interface seems to be a necessary component of a fuzzy system, some experts may argue whether the fuzzification interface is needed. As shown in developing the reduced form of Zadeh's compositional rule of inference in Equation (2.14) and (2.24) where the inputs are crisp values, the input fuzzy sets denoted by $\hat{U}_i$ disappear and all

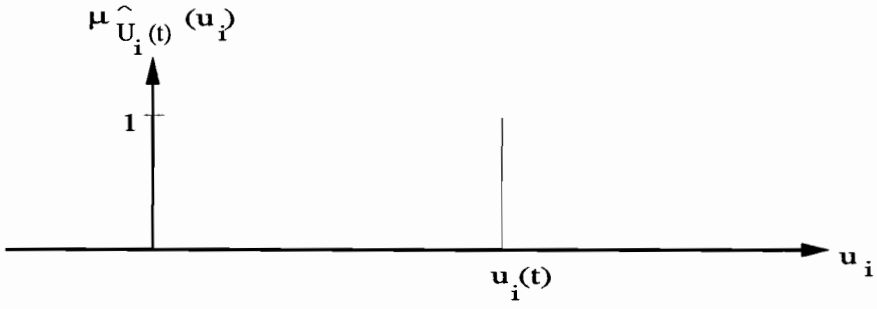


Figure 13: Membership function for a fuzzified crisp input to a fuzzy system.

that is really required to infer the appropriate value at the output is the numerical crisp value of the inputs. Typically when implementing a fuzzy system, similar reduced forms of the sup-star compositional rule of inference are used. Therefore, this fuzzification process is sometimes skipped.

2.3.2 Knowledge-Base

The function of the knowledge base is to model a human expert's qualitative knowledge about the desired relationship between the inputs and outputs of a fuzzy system. As previously mentioned, this qualitative knowledge is quantified by fuzzy set theory. Therefore, the knowledge base is comprised of three basic components including: a fuzzy set data base, a rule base, and a fuzzy relation generator.

The fuzzy set data base contains all fuzzy sets that quantify linguistic statements for the inputs such as "$\tilde{u}_i$ is $\tilde{U}_i^j$" and output such as "$\tilde{y}$ is $\tilde{Y}^m$". Therefore, the fuzzy set data base contains information about the universes of discourse and the membership functions associated with all fuzzy sets that are associated with the inputs and outputs of the system.

The rule base consists of all fuzzy implications that are used to describe the

relationship between the outputs and inputs of a fuzzy system. In order to ensure completeness and to establish a concise representation of all fuzzy implications, the rule base is often expressed by an r dimensional array, where r is the number of inputs to the fuzzy system. For example, consider a two input - single output fuzzy implication of the form

$$\textbf{If } U_1^j \textbf{ and } U_2^k \textbf{ Then } Y^m.$$

A typical rule base array for such a fuzzy implication is shown in Table 3 below. The row of numbers along the top of the fuzzy implication rule base array indicates

Table 3: Typical rule base array for a two input - single output system

Y^m		U_1^j						
		−3	−2	−1	0	+1	+2	+3
U_2^k	−3	−3	−2	−1	−1	−1	+1	+1
	−2	−3	−3	0	−1	0	+2	+2
	−1	−2	−2	−1	0	+1	+2	+1
	0	−1	−1	−1	0	+1	+2	+2
	+1	0	0	0	0	+1	+2	+3
	+2	+1	0	+1	+1	+1	+3	+3
	+3	0	0	+1	+2	+1	+2	+3

the index j for the fuzzy sets denoted U_1^j defined on the universe of discourse $\mathcal{U}_1$. Likewise, the left hand most column of numbers represent the values of index k for the fuzzy sets denoted by U_2^k. Moreover, the numbers in the table represent the values of index m for the output fuzzy sets denoted Y^m. Therefore, a typical fuzzy implication that could be extracted from the Table 3 would be expressed as

$$\textbf{If } U_1^{-3} \textbf{ and } U_2^{-3} \textbf{ Then } Y^{-3}.$$

Table 3 provides a concise representation for the entire rule base array for a two input - single output fuzzy system. Tables of higher dimension can be used to represent more general sets of rules.

The primary function of the fuzzy relation generator is to access the fuzzy sets data base and the fuzzy implication rule base to generate a fuzzy relation for each fuzzy implication defined in the rule base. The resulting fuzzy relations provide a representation of the knowledge base that is compatible with the inference process. The procedure of generating fuzzy relations is quite straight forward and simply involves forming the cartesian product of each fuzzy set defined in a given rule. Refer back to Definition 2.13.

2.3.3 Inference Mechanism

The inference mechanism is the heart of the fuzzy system. Essentially it has the capability to simulate the human decision making process by inferring an output from the system inputs and the fuzzy relations based on basic fuzzy set operations.

The inference mechanism determines the implied fuzzy set by first performing a cartesian product on the fuzzified inputs. Then as shown in Figure 12, the inference mechanism performs a sup-star composition on the resulting cartesian product and each of the fuzzy relations generated within the knowledge base. This procedure is quite straight forward and is based on the previous discussion of fuzzy sets operation.

However, in order to illustrate the operation of the defuzzification interface in the next section, an example of the implied fuzzy relations generated by the inference mechanism is given by the following example. Assume we are given a fuzzy system whose rule base is given in Table 3. Further, assume that membership functions for fuzzy sets U_1^j, U_2^k, and Y^m are normal, convex, and symmetric with exception of

the "end membership functions" for fuzzy set U_1^j and U_2^k, and distributed along the universe of discourse according to Assumption 2.2 as illustrated in Figure 14. Often fuzzy system designs include non-symmetric "end membership functions" to ensure that any input may be classified under at least one fuzzy set.

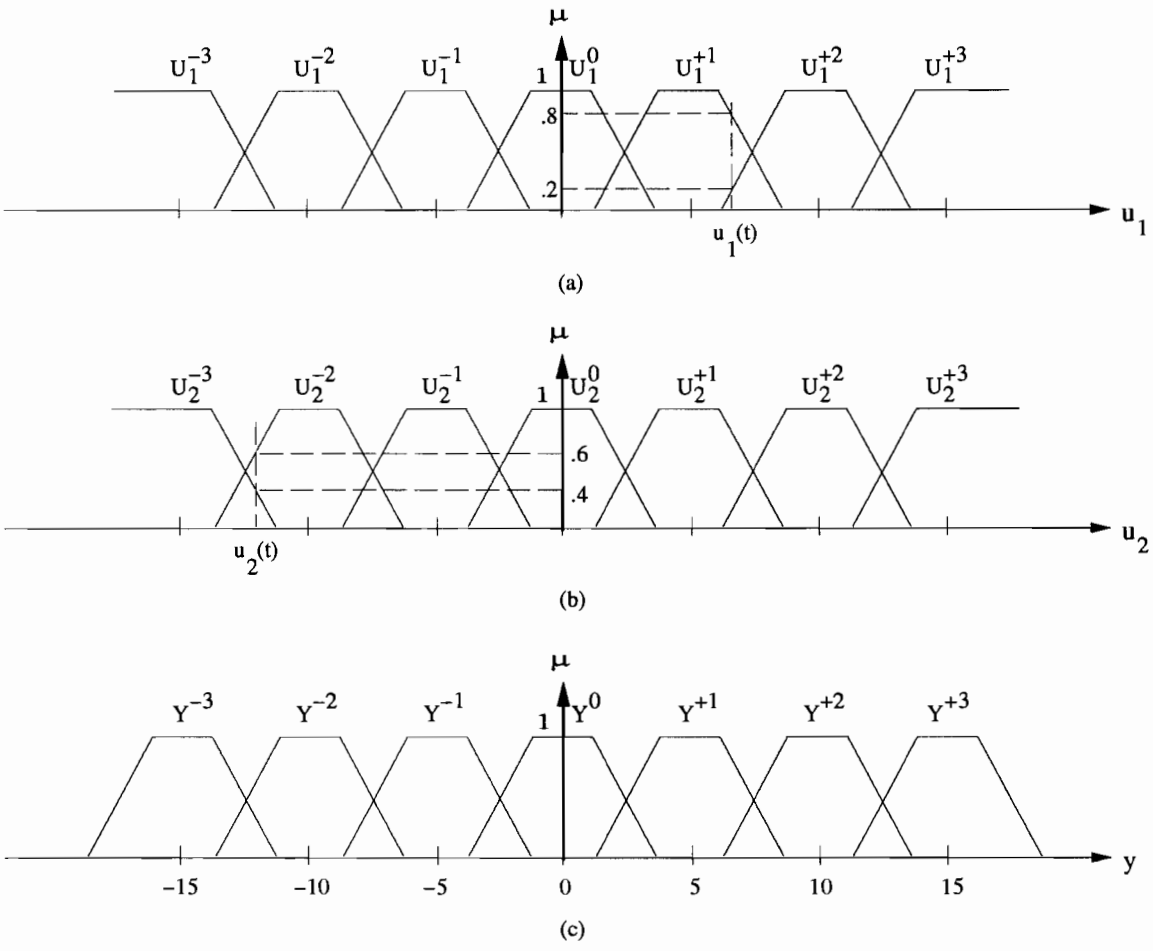


Figure 14: Membership functions associated with a) u_1, b) u_1 and c) y.

The fuzzy system inputs at time t, denoted by $u_1(t)$ and $u_2(t)$, are shown in Figure 14(a) and (b), respectively, where the actual numerical values of the inputs

are not really needed for this illustration. If Zadeh's Compositional rule of inference is assumed then the membership functions for the implied set fuzzy sets $\hat{Y}^{+1,-3}(t)$, $\hat{Y}^{+2,-3}(t)$, $\hat{Y}^{+1,-2}(t)$, and $\hat{Y}^{+2,-2}(t)$ are shown in Figure 15. By inspection, it is easy to see that all other implied fuzzy sets generated by Zadeh's compositional rule of inference are equal to the null fuzzy set Φ.

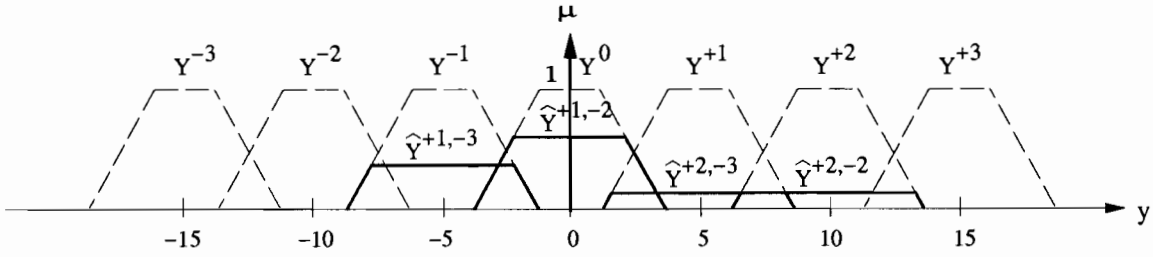


Figure 15: Membership functions for the implied fuzzy sets.– "not drawn to scale"!

2.3.4 Defuzzification Interface

Basically, the defuzzification interface generates a mapping from the implied fuzzy sets generated by the inference mechanism to a crisp output suitable for practical application. Therefore, the aim of the defuzzification strategies is to produce a crisp output that represents the combined effect of all implied fuzzy sets generated by the inference mechanism. At present, the most commonly used defuzzification strategies may be described as the max criteria (MC), the mean of the maximum (MOM), the center of area (COA), and the center of gravity (COG).

In order to employ the MC, MOM, and COA defuzzification strategies we must consider the union of all individual implied fuzzy sets as shown in the block diagram of a fuzzy system in Figure 12. If we assume that the implied fuzzy sets generated by the inference mechanism have membership functions shown in Figure 15, then

Figure 16 illustrates the membership function, denoted $\mu_{\hat{Y}}(t)(y)$ resulting from the union of these fuzzy sets (keep in mind that this is in general a membership function whose shape is time varying).

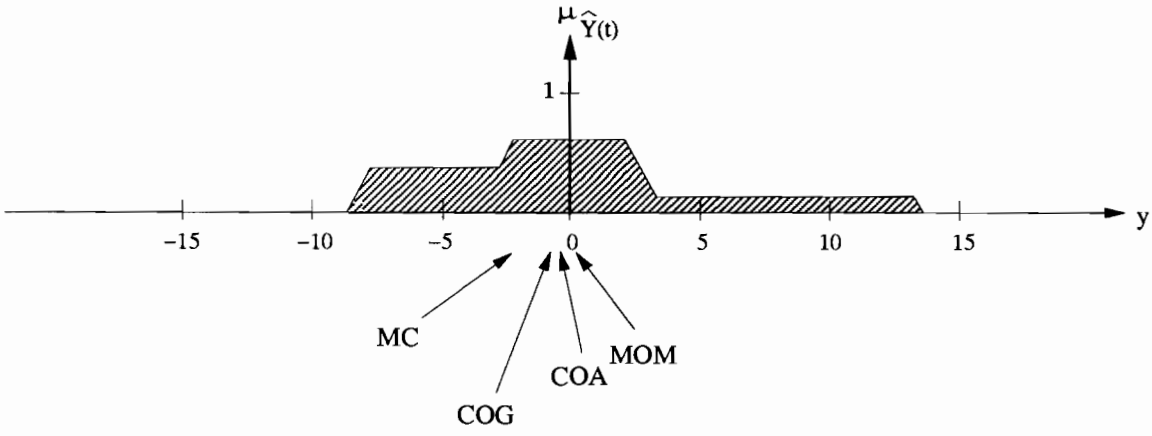


Figure 16: Membership function for union of implied fuzzy sets.

As previously mentioned, this union operation is a reasonable procedure at this point since the *condition* $\rightarrow$ *action* rules may be interpreted as

If condition 1 **Then** action 1

Or

If condition 2 **Then** action 2

Or . . .

where 'OR' is used to connect the rules. However, this union operation is not performed when the COG defuzzification strategy is employed. In this case, each implied fuzzy set is considered individually to produce the overall output of the fuzzy system. Therefore, this COG strategy moves slightly away from normal fuzzy set operations and the union operation shown in the defuzzification interface of Figure 12 must be

removed if this strategy is employed. However, as will be shown, this strategy is commonly used due to ease of computation and its intuitive appeal.

In the following discussion, We will explain each of the defuzzification strategies individually. Refer to Figure 16 for a graphical interpretation of each strategy.

Max Criteria

As previously mentioned, the defuzzification strategy should be somewhat representative of the "mean" of the implied membership function $\hat{Y}$. The MC approximates this "mean" by producing a point where $\hat{Y}(t)$ reaches a max. Notice, in Figure 16 that $\hat{Y}(t)$ maintains a maximum on the interval [-2.5,2.5]. Under such a condition that the implied fuzzy set $\hat{Y}(t)$ is a maximum for many elements of the universe of discourse, we typically choose the end element of the interval that is nearest other elements with high membership. In Figure 16 this corresponds to $y(t) = -2.5$ since the left hand side of the max interval is near another interval with high membership. In general, an algorithm for choosing the end element of the max interval which is nearest other elements with high membership has not been well established. Also regardless of the algorithm, a situation may occur in which it is not certain which end element of the max interval to choose since both end elements may be near other elements with equally high membership. Therefore, the MC defuzzification strategy is not recommended when the sup-star composition employed in the inference mechanism is likely to produce fuzzy sets whose membership functions have "flat tops". This is the case for Zadeh's compositional rule of inference since the effect is simply to remove the "tops" of the output fuzzy sets.

Mean of the Maximum

The MOM defuzzification strategy is a generalization of the MC. This defuzzification strategy is particularly useful when the implied fuzzy $\hat{Y}(t)$ set is "flat" along a single interval where the membership has reached a maximum. In this situation, the output is chosen to represent the mean value of all elements whose membership in fuzzy set $\hat{Y}(t)$ is a maximum. Therefore, for the implied fuzzy set shown in Figure 16, we would choose $y(t) = 0$. In the case of a discrete universe of discourse, the output of the fuzzy system may be computed by

$$y(t) = \sum_i \frac{y^i}{n}, \tag{2.36}$$

where y^i is the i^{th} element whose membership in fuzzy set $\hat{Y}(t)$ is a maximum, and n is the number of elements in $\hat{Y}(t)$ whose membership is a maximum. It should be noted that when the MOM is used with the Zadeh's compositional rule of inference the overall input-output relationship of the fuzzy system often behaves similar to a multilevel relay. Therefore, if the MOM defuzzification strategy is employed, we may not utilize the total flexibility of the fuzzy controller since it is possible to generate the same behavior using easier to implement classical set theory.

Center of Area

The widely used COA defuzzification strategy overcomes some of the problems associated with the MC and MOM strategies by generating the center of area of the membership function of the implied fuzzy set $\hat{Y}(t)$. For a continuous output universe of discourse the center of area of a membership function associated with an implied fuzzy set, denoted by $\hat{Y}(t)$, is expressed by

$$y(t) = \frac{\int_y y \cdot \mu_{\hat{Y}(t)}(y)dy}{\int_y \mu_{\hat{Y}(t)}(y)dy}. \tag{2.37}$$

In the case of implied fuzzy sets shown in Figure 16, the center of area of the shaded region approximately produces $y(t) = 1.2$. For continuous output universes of discourse computing the center of area is often computationally complex since $\hat{Y}(t)$ tends to have a complicated shape. However, the computation for the discrete universe is much easier for a computer since the fuzzy system output can be expressed by

$$y(t) = \frac{\sum_{j=1}^{m} \mu_{\hat{Y}(t)}(y^i) \cdot y^i}{\sum_{j=1}^{m} \mu_{\hat{Y}(t)}(y^i)} \tag{2.38}$$

where y^i is the i^{th} element along the output universe of discourse and m is the number of quantization levels of the output universe of discourse. Notice that the COA defuzzification strategy often produces a smooth input-output relationship for the fuzzy system.

The MC, MOM, and COA defuzzification strategies all employ fuzzy set operations on the individual implied fuzzy set before defuzzification. However, after employing the union operation to compute $\hat{Y}(t)$, all information included in the overlap of the membership functions for two implied fuzzy sets is lost. In fact, it is possible that a given membership function for an implied fuzzy set may completely overlap another thus eliminating all information generated due to the overlapped fuzzy set. Therefore, a defuzzification strategy which considers each implied fuzzy set individually could more closely emulate the human decision making process. This is the basic motivation for the COG method to be described next.

Center of Gravity Method

The COG method provides an alternative framework for defuzzification so that each implied fuzzy set is considered individually. In this strategy the overlap of two fuzzy sets is assumed to carry twice the weight than the non-overlapping portions. The center of gravity is most easily calculated by considering the area under each fuzzy implication individually. Therefore, a general equation for calculating the center of gravity for a set of implied fuzzy sets, denoted by $\hat{Y}^{j,k,\ldots,l}$, is expressed by

$$y(t) = \frac{\sum_{j=N_1^-,k=N_2^-,\ldots,N_r^-}^{N_1^+,N_2^+,\ldots,N_r^+} c^{j,k,\ldots,l} \int_y \mu_{\hat{Y}^{j,k,\ldots,l}(t)}(y)dy}{\sum_{j=N_1^-,k=N_2^-,\ldots,N_r^-}^{N_1^+,N_2^+,\ldots,N_r^+} \int_y \mu_{\hat{Y}^{j,k,\ldots,l}(t)}(y)dy} \tag{2.39}$$

where $c^{j,k,\ldots,l}$ is the center of area of the membership function associated with the implied fuzzy set $\hat{Y}^{j,k,\ldots,l}$. For example, consider the membership functions for the implied fuzzy set $\hat{Y}^{+1,-3}(t)$, $\hat{Y}^{+2,-3}(t)$, $\hat{Y}^{+1,-2}(t)$, and $\hat{Y}^{+2,-2}(t)$ shown in Figure 15. Notice that these fuzzy sets have trapezoidal shaped membership functions whose "tops" have been removed so that they are no longer normal membership functions. Consider a generalized, non-normal and symmetric membership function of x called "trap"as shown in Figure 17. This function is characterized by (h) the height of the function at the maximum, (b) the width of the function at the base, (w) the width of the function if extended to a height of 1 unit, and (c) the symmetric center of the function. It is easily determined that the area under the "trap" function is given as

$$Area = \int_x Trap(x : c, b, w, h)dx = b \cdot \left[h - \frac{h^2}{2}\right] + w \cdot \frac{h^2}{2}. \tag{2.40}$$

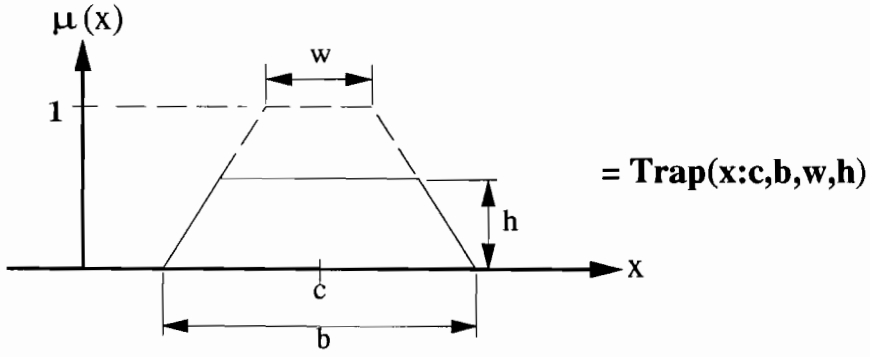


Figure 17: Generalized, non-normal, symmetric trapezoidal shaped membership function.

Substituting the result from Equation (2.40) into Equation (2.39), we obtain

$$y(t) = \frac{\sum_{j=N_1^-,k=N_2^-,\ldots,N_r^-}^{N_1^+,N_2^+,\ldots,N_r^+} c^{j,k,\ldots,l}\left\{b^{j,k,\ldots,l}\left[h^{j,k,\ldots,l} - \frac{h^{j,k,\ldots,l^2}}{2}\right] + w^{j,k,\ldots,l}\frac{h^{j,k,\ldots,l^2}}{2}\right\}}{\sum_{j=N_1^-,k=N_2^-,\ldots,N_r^-}^{N_1^+,N_2^+,\ldots,N_r^+}\left\{b^{j,k,\ldots,l}\left[h^{j,k,\ldots,l} - \frac{h^{j,k,\ldots,l^2}}{2}\right] + w^{j,k,\ldots,l}\frac{h^{j,k,\ldots,l^2}}{2}\right\}}, \quad (2.41)$$

where the superscripts are used on variables b,c,w, and h to denote which implied fuzzy set that these variables are associated. In Table 4 below, we summarize these values for the example shown in Figure 15. Substituting these values into Equation (2.41) we obtain a fuzzy system output at time t such that $y(t) = .9149$. As shown in Figure 16, the COG method produced a "more negative" result in this example than did the COA due to the relatively large overlap between $\hat{Y}^{+1,-3}(t)$ and $\hat{Y}^{+1,-2}(t)$.

Sometimes the COG method is approximated by assuming that the area under the membership function of each implied fuzzy set is proportional to the maximum height of the membership function. Under this assumption Equation (2.39) reduces to

Table 4: Summary of parameters for the implied fuzzy sets.

Fuzzy Set	c	b	w	h
$\hat{Y}^{+1,-3}(t)$	-5	8	2	0.4
$\hat{Y}^{+1,-2}(t)$	0	8	2	0.6
$\hat{Y}^{+2,-3}(t)$	5	8	2	0.2
$\hat{Y}^{+2,-2}(t)$	10	8	2	0.2

$$y(t) = \frac{\sum_{j=N_1^-,k=N_2^-,\ldots,N_r^-}^{N_1^+,N_2^+,\ldots,N_r^+} c^{j,k,\ldots,l} \sup_y \mu_{\hat{Y}^{j,k,\ldots,l}(t)}(y)}{\sum_{j=N_1^-,k=N_2^-,\ldots,N_r^-}^{N_1^+,N_2^+,\ldots,N_r^+} \sup_y \mu_{\hat{Y}^{j,k,\ldots,l}(t)}(y)}. \tag{2.42}$$

This assumption often reduces the computational complexity for some of the more complex shaped membership functions. In this thesis, we will always compute the true area under the membership function for each of the implied fuzzy sets.

2.4 The Fuzzy System Employed as a Feedback Controller

In this section we show how the fuzzy system described in the previous sections may be employed in a feedback control scheme. To illustrate this idea, we will employ a feedback fuzzy system to control the rotational position of the Photocircuits Corp. U12M4T DC motor. Since good control of the U12M4T DC motor is possible using classical control techniques, this example is intended solely for illustration.

2.4.1 Architecture for Feedback Fuzzy Control

A typical architecture for implementing a feedback fuzzy controller is shown in Figure 18 below where $y(t)$ is the output of the process, $y_d(t)$ is the desired output of the process, $e(t)$ is the error, and $u(t)$ is the process input at time t. If $\tilde{e}$, $\tilde{f}_i$, and $\tilde{u}$ are assumed to be generic arguments for linguistic variables of the error, the i^{th} function of error, and the process input, respectively, then the *condition* $\rightarrow$ *action* rules for this fuzzy system may be expressed linguistically as

If $\tilde{e}$ is $\tilde{E}^j$ **and** $\tilde{f}_1$ is $\tilde{F}_1^k$ **and** $\tilde{f}_2$ is $\tilde{F}_2^l$ **and**, . . . , **and** $\tilde{f}_n$ is $\tilde{F}_n^m$ **Then** $\tilde{u}$ is $\tilde{U}^o$,

where $\tilde{E}^j$, $\tilde{F}_1^k$, $\tilde{F}_2^l$, . . . , $\tilde{F}_n^m$, and $\tilde{U}^o$ are generic for linguistic terms such as "small", "medium", "big", and etc. Moreover, the functions of error are typically simple linear operations on the error such as $\frac{d\,e(t)}{dt}$ and $\int e(t)dt$ whose values are computed in the traditional manner and supplied to the fuzzy controller. Hence, the fuzzy controller is often "hybrid" in the sense that it contains both a conventional linear part and a fuzzy system. Other inputs to the fuzzy system which are not included in Figure 18 may include the process output, the states of the process, and functions of these variables.

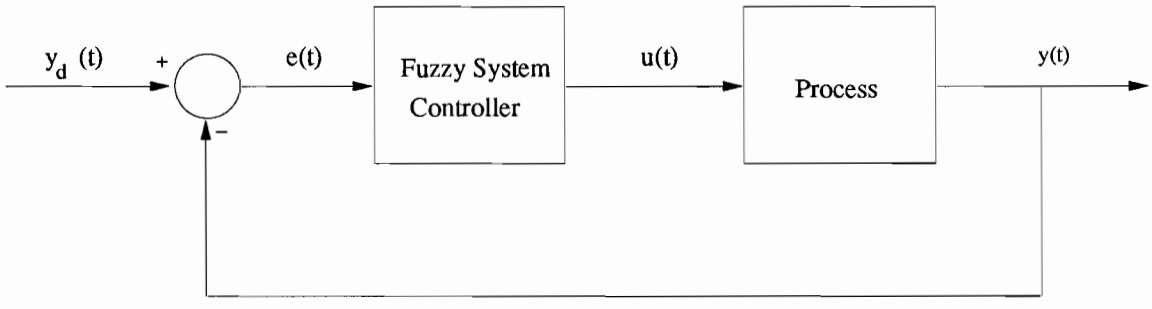


Figure 18: Architecture of a feedback fuzzy control system.

2.4.2 Motor Position Control - Problem Statement

As previously alluded to, we desire to design a fuzzy system to control the position of a permanent magnet d.c. motor. A non-linear dynamical model for permanent magnet d.c. motor may be given as

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-R\cdot B - K_b \cdot K_\tau}{R\cdot J} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_\tau}{n\cdot R\cdot J} \end{bmatrix} V_{in} - \begin{bmatrix} 0 \\ \frac{\tau_f}{n\cdot J} sgn(\dot{\theta}) \end{bmatrix} \tag{2.43}$$

where θ is the angular position in radians, V_{in} is the input voltage, and for the U12M4T DC motor the motor parameters are given below.

1. $K_\tau = 0.101 \frac{N\cdot m}{amp}$ - torque constant
2. $K_b = 0.101 \frac{V\cdot sec}{rad}$ - back EMF constant
3. $B = 3.03 \times 10^{-4} \frac{N\cdot m\cdot sec}{rad}$ - internal rotational damping
4. $J = 2.33 \times 10^{-4} \frac{N\cdot m\cdot sec^2}{rad}$ - internal rotational inertia
5. $n = 102$ - internal gear ratio
6. $R = 0.75\Omega$ - internal resistance
7. $\tau_f = 4.24 \times 10^{-2} N \cdot m$ - internal rotational friction

Fuzzy Controller Design

For this example, we will employ only the error and the derivative of the error as inputs to the fuzzy system as shown in Figure 19 below. Notice that the fuzzy system in Figure 19 is referred to as the "normalized" fuzzy system. This terminology is used since the input and output universes of discourse within the fuzzy system are normalized on the range [-1,1]. Therefore, the gains g_e and g_c are employed simply

to map the actual inputs of the fuzzy system to the internal, normalized universes of discourse. Likewise, the gain g_y maps the internal, normalized output universe of discourse to the desired output universe of discourse. Fuzzy systems are often implemented with normalized universes of discourse so that a single subroutine may be used for a number of fuzzy controllers. Also, the normalizing gain provides an easy method to modify the controller in the event that the initial controller does not work well. More discussion about the normalizing gains is given later in this section.

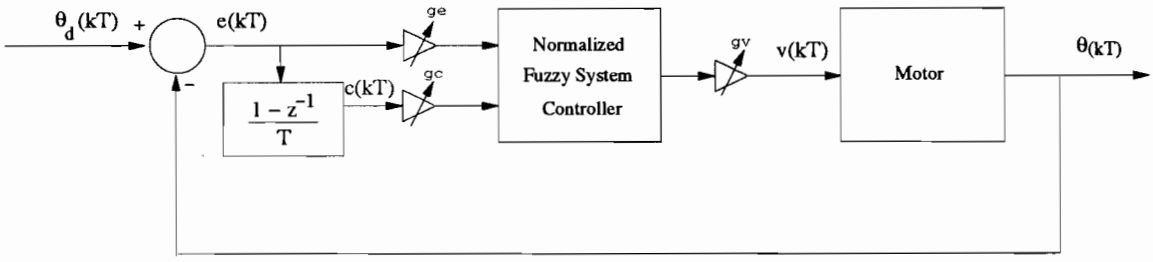


Figure 19: Fuzzy control system configuration for position control of the U12M4T permanent magnet D.C. motor.

Assuming that the fuzzy system is implemented in discrete time, $e(kT)$ denotes the numeric value of the error at time $t = kT$ such that

$$e(kT) = \theta_d(kT) - \theta(kT), \tag{2.44}$$

where $\theta(kT)$ is the position of the motor and $\theta_d(kT)$ is the desired position of the motor at time $t = kT$. We may synthesize the derivative of the error by employing the following discrete time equation

$$c(kT) = \frac{e(kT) - e((k-1)T)}{T}, \tag{2.45}$$

where $c(kT)$ is the synthesized error derivative which we will refer to as the change in error.

Since this fuzzy control system has two inputs, namely the error and change in error, and one output involving the motor input voltage, the linguistic control rules may be expressed as

$$\textbf{If } \tilde{e} \text{ is } \tilde{E}^j \textbf{ and } \tilde{c} \text{ is } \tilde{C}^k \textbf{ then } \tilde{v} \text{ is } \tilde{V}^l,$$

where $\tilde{e}$, $\tilde{c}$, and $\tilde{v}$ denote the linguistic variables for error, change in error, and motor input voltage, respectively, and $\tilde{E}^j$, $\tilde{C}^k$, and $\tilde{V}^l$ denote associated linguistic values. Furthermore, these qualitative control rules may be quantized by fuzzy implications of the form

$$\textbf{If } E^j \textbf{ and } C^k \textbf{ then } V^l,$$

where E^j, C^k, and V^l are fuzzy sets quantifying linguistic statements "$\tilde{e}$ is $\tilde{E}^j$", $\tilde{c}$ is $\tilde{C}^k$, and $\tilde{v}$ is $\tilde{V}^l$, respectively. Also, we assume that there exist 11 fuzzy sets defined on each of the input and output universes of discourse whose membership functions are symmetrical, triangular shaped, and uniformly distributed along the universes of discourse such that Assumption 2.2 holds. Moreover, we assume that the universes of discourse for the error, error derivative, and input voltage, are defined such that $\mathcal{E} = [-360, 360]$, $\mathcal{C} = [-50, 50]$, and $\mathcal{V} = [-220, 220]$, respectively. The "non-normalized" membership functions associated with fuzzy sets E^j, C^k, and V^l are shown in Figure 20. However, these membership functions are normalized in the actual fuzzy system implementation where the normalizing fuzzy controller gains were assigned such that $g_e = 1/360$, $g_c = 1/50$, and $g_v = 220$.

Notice in Figure 20 (a) and (b) that the end membership functions for both the error and error derivative are extended beyond the actual universes of discourse. This assumption is commonly made in fuzzy control design to provide a "safety factor"

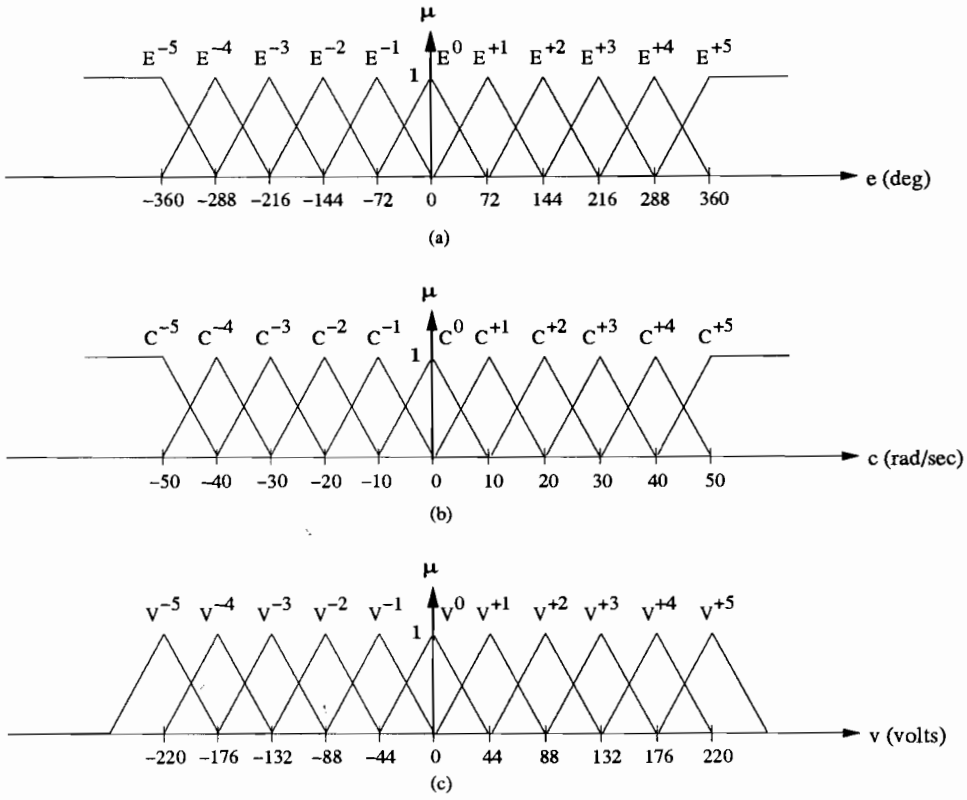


Figure 20: Membership functions for fuzzy control of the U12M4T DC motor.

in the event that the input should exceed the actual defined universe of discourse. Furthermore, as shown in Figure 20 (c), the end membership functions for the output universe of discourse are sometimes "mirrored" beyond the limits of the actual output universe of discourse. This ensures that the crisp output of the fuzzy system may take on all values of the output universe of discourse since it is impossible for the COA or COG defuzzification strategies to generate an output equal to the maximum or minimum of the output universe of discourse if no elements beyond the universe

of discourse have membership greater than 0.

Given the fuzzy sets defined above in Figure 20, we may generate the rule base array shown in Figure 5 below. This rule base array table was generated based on

Table 5: Rule base array for fuzzy control of the U12M4T DC motor.

V^l		C^k										
		−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5
E^j	−5	−5	−5	−5	−5	−5	−5	−4	−3	−2	−1	0
	−4	−5	−5	−5	−5	−5	−4	−3	−2	−1	0	+1
	−3	−5	−5	−5	−5	−4	−3	−2	−1	0	+1	+2
	−2	−5	−5	−5	−4	−3	−2	−1	0	+1	+2	+3
	−1	−5	−5	−4	−3	−2	−1	0	+1	+2	+3	+4
	0	−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5
	+1	−4	−3	−2	−1	0	+1	+2	+3	+4	+5	+5
	+2	−3	−2	−1	0	+1	+2	+3	+4	+5	+5	+5
	+3	−2	−1	0	+1	+2	+3	+4	+5	+5	+5	+5
	+4	−1	0	+1	+2	+3	+4	+5	+5	+5	+5	+5
	+5	0	+1	+2	+3	+4	+5	+5	+5	+5	+5	+5

the following five linguistic control rules:

1. **if** "error is positive" **and** "change in error is positive" **then** "motor input voltage is positive"

   Note - the motor is moving in the wrong direction,

2. **if** "error is positive" **and** "change in error is negative" **then** "motor input voltage is zero"

   Note - the motor is moving the right direction,

3. **if** "error is negative" **and** "change in error is positive" **then** "motor input voltage is zero"

   Note - the motor is moving the right direction,

4. **if** "error is negative" **and** "change in error is negative" **then** "motor input voltage is negative"

   Note - the motor is moving in the wrong direction,

5. **if** "error is zero" **and** "change in error is zero" **then** "motor input voltage is zero"

   Note - the motor is correctly positioned.

However, the rule base array in Table 5 contains 121 rules which effectively quantify various degrees of rules 1–5 above. Given some careful thought, the reader should easily become convinced that the rule base array table in Figure 5 is consistent with these five control rules.

Simulation Results

The fuzzy control system discussed above was implemented and simulated using the FORTRAN programming language. See Appendix A.1 for the actual program code. The results of this simulation are shown in Figure 21 where the COG defuzzification strategy was employed to generate the crisp output of the fuzzy system. Notice in Figure 21 that "good" control was obtained using the fuzzy controller. As was previously mentioned, good results have been obtained for the control of permanent magnet D.C. motor using classical techniques such as proportional-integral-derivative (PID) control. However, in the next section of this chapter we will illustrate why fuzzy control in some instances provides better control than the classical methods.

2.5 Comparison of Fuzzy Controller with Classical Control Laws

In this section we present a discussion of how fuzzy logic control systems compare with the classical control laws. In particular, we will show how fuzzy logic control

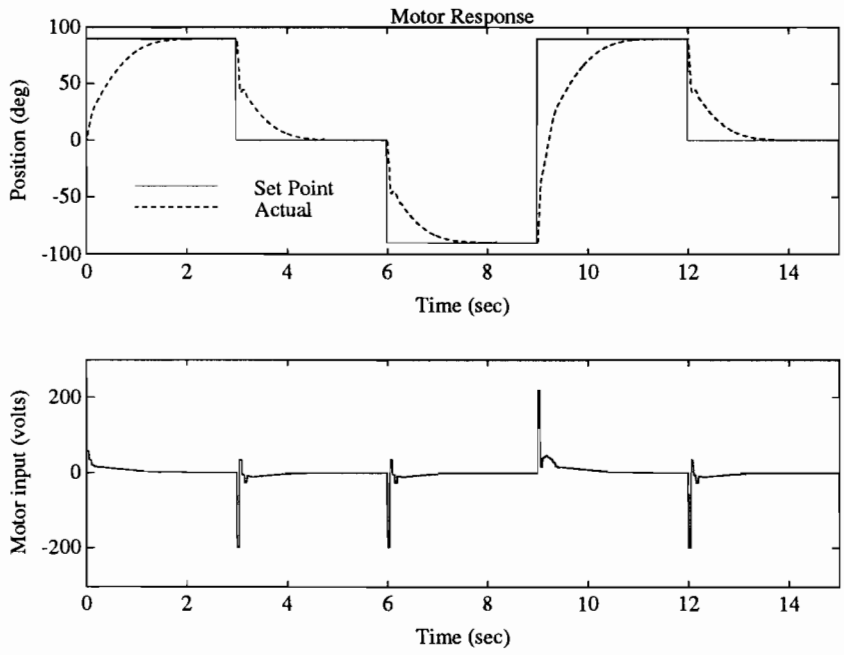


Figure 21: Simulation results for fuzzy control of the rotation position of the U12M4T DC motor.

described in the previous section compares to the proportional-integral-derivative (PID) control law. It is hoped that from this discussion that some of the advantages of fuzzy control may become more apparent to the reader. Moreover, we intend to show some aspects of fuzzy controller design which allow the designer to most effectively employ these advantages. Also, in this section, we show how linguistic control rules are often expressed such that the absolute value of the open loop steady state gain is increased to infinity. For linear systems, this change in the open loop steady state gain is often referred to as an increase in the type of the system. For instance, a type 0 linear system has a constant steady state gain. This generally results in a steady state error for the closed loop system when only proportional error and/or derivative feedback is employed. Therefore, an integrator is often employed in the feed forward

path of the open loop system to increase the gain to infinity. Moreover, non-linear, time-varying processes often produce the same steady state error if the steady state gain of the system is less than infinity.

2.5.1 Fuzzy Control Vs. PID Control

Suppose that the fuzzy system controller shown in Figure 18 is replaced by the PID control law expressed as

$$u(t) = k_p \cdot e(t) + k_d \cdot \frac{d\ e(t)}{dt} + k_i \cdot \int e(t)dt, \tag{2.46}$$

which operates on the error $e(t)$ to produce the process input $u(t)$. As shown in Figure 22 below, the PID control law could be viewed as two parts: a dynamic portion which generates the dynamic functions of the error $e(t)$, including $\frac{d\ e(t)}{dt}$ and $\int e(t)dt$, and a static portion which statically maps the error and the dynamic functions of the error to the process input $u(t)$. Notice that the static portion of

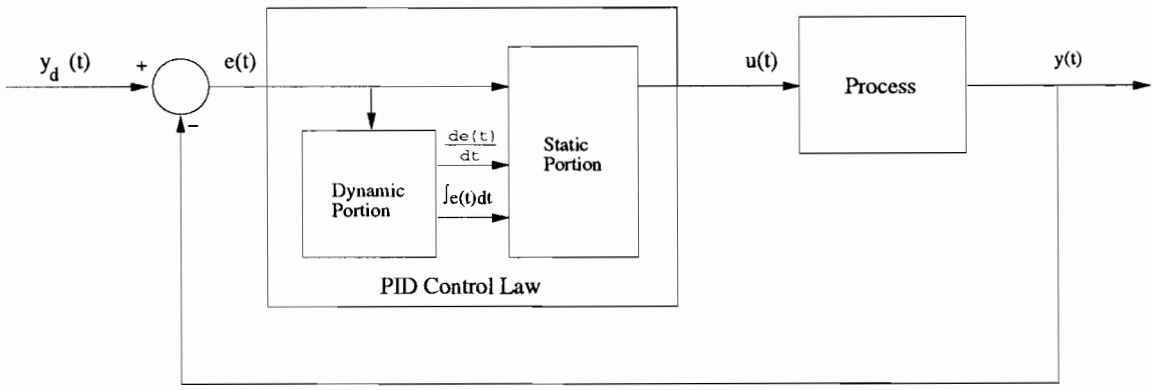


Figure 22: Configuration of feedback PID control.

the PID control law generates a simple linear "control surface" between the inputs and output of the controller since the output $u(t)$ is simply a weighted sum of the

inputs $e(t)$, $\frac{d\ e(t)}{dt}$, and $\int e(t)dt$. If we assume only proportional-derivative feedback or $k_i = 0$ in Equation (2.46), then the control surface which describes the relationship between the error, error derivative and output is illustrated in Figure 23 below.

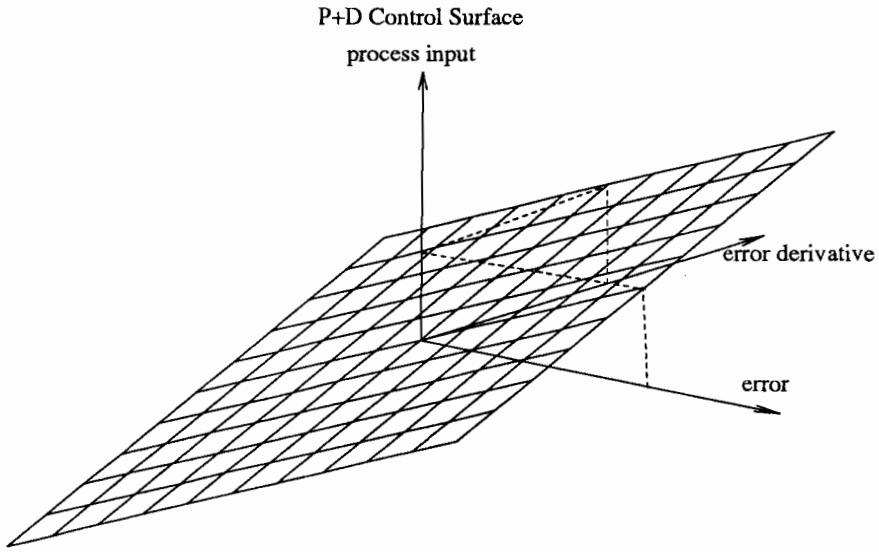


Figure 23: Control surface for the proportional-derivative control law.

Compare the proportional-derivative (PD) control surface shown in Figure 22 with the fuzzy control surface shown in Figure 24 which was generated by the fuzzy controller designed for the U12M4T DC motor discussed in the previous section such that the static portion of the PD controller was replaced by the fuzzy controller. Notice that fuzzy control surface is nearly linear except for two "flat" regions, one which occurs where e and c are "large and positive" and the other occurs when e and c are "negatively large ". Therefore, this fuzzy controller design is very unlikely

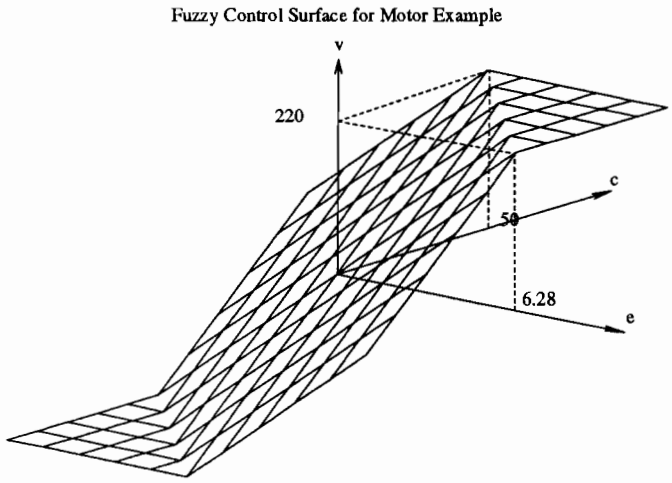


Figure 24: Control surface employed on the U12M4T DC motor in the previous section.

to produce a response better than that of the PD control law. However, the fuzzy controller was purposely implemented such that this nearly linear feature would exist to illustrate this possibility. This linear behavior occurs due to the fact that the fuzzy sets defined for both the inputs and the output of the fuzzy system have membership functions that are symmetrical and are uniformly distributed along the universes of discourse. Furthermore, the rule base was specified so that the implied fuzzy sets were symmetrical about a diagonal. Since one major advantage fuzzy control design is that it provides a structured framework for developing a non-linear control scheme, this nearly linear behavior may be undesirable. From this example, we learn that to more effectively utilize the advantages of fuzzy control, complete symmetry should be avoided in the fuzzy controller design. Therefore, often in fuzzy control design, the membership functions along the output universe of discourse are selected in a

non-uniform fashion. Yet, uniformity is often maintained on the input universes of discourse. Under this assumption nearly any desired characteristic of the fuzzy control surface is possible.

2.5.2 Linguistic Control Rules With Integration

For systems that exhibit behavior similar to that of a type 0 linear system, it is often necessary to add an integrator in the feed forward path of the open loop system to reduce the steady state error. Therefore, it is not surprising that for systems with type 0 behavior, the linguistic control rules are often expressed such that the output of the fuzzy system is effectively integrated. For instance, consider a single input fuzzy system with linguistic control rules of the form

$$\textbf{If } \tilde{e} \text{ is } \tilde{E}^j \textbf{ Then } \text{the (increase/decrease) in } \tilde{u} \text{ is } \tilde{U}^m,$$

where $\tilde{e}$ denotes the process error and $\tilde{u}$ denotes the process input. It is easily shown that control rules of this form roughly provide the function of integration at the output of the controller. For example, linguistic control rules in this form are often implemented so that the output of the fuzzy controller is added to the previous input. This operation is expressed by the following discrete time equation

$$y(kT) = y((k-1)T) + FS(kT), \tag{2.47}$$

where $y(kT)$ is the current input to the process, $y((k-1)T)$ is the previous input to the process, and $FS(kT)$ is the output of the fuzzy controller. This discrete time equation can be transformed to the z domain to obtain

$$Y(z) = z^{-1}Y(z) + FS(z). \tag{2.48}$$

Rearranging Equation (2.48) results in the following z domain transfer function

$$\frac{Y(z)}{FS(z)} = \frac{1}{1 - z^{-1}}, \tag{2.49}$$

which is a discrete time equivalent of an integrator.

The following heuristic statement provides an easy test to determine whether a given process has behavior similar to a type 0 linear system an therefore is likely to requires an integrator at the output of the fuzzy controller. "For any constant input to the process except the zero input, if the system does not continue to move in a given direction until the output reaches a maximum, then the system most likely requires an integrator".

Fortunately, the example involving fuzzy control of the U12M4T DC motor did not require the integrator. This was due to the fact that except for a small deadband for low input voltage resulting from the friction torque the system exhibits an infinite steady state gain. Under some circumstances, the integrator at the output of the fuzzy controller may not be necessary if the integral of the error or the output of the process itself is an input to the fuzzy controller since these are not necessarily zero or some other quantity when the system reaches steady state.

If the reader desires to learn more about the relationship between fuzzy control and classical linear control, some analytical studies are is presented by J. Buckley and H. Ying in [10] and H. Ying and W. Siler and J. Buckley in [87]. By choosing appropriate control rules and defuzzification strategy, this research focuses primarily on the development of a fuzzy controller that is equivalent to a PID or PI control law.

2.6 Chapter Summary

In this chapter we have presented an overview of fuzzy set and system theory and its role in control. This overview includes an introduction to the terminology and mathematical operations normally defined for fuzzy sets. Also, we have shown how fuzzy set theory and operations are employed in a typical fuzzy system architecture. We illustrate how fuzzy systems are used in control by a practical application involving the position control of a D.C. motor. The objective of this D.C. motor application was to illustrate the design and implementation of a fuzzy controller. Finally we have given a brief comparison between fuzzy control and conventional linear control. This comparison has shown that fuzzy controller can provide greater flexibility than some conventional control methods.

CHAPTER III

Linguistic SOC and FMRLC

In this chapter we provide an overview of the linguistic self-organizing controller (SOC) presented by Procyk and Mamdani in [59] as well as the refinements of this algorithm made by others. Moreover, we introduce a new "rule modification" technique to improve the SOC approach. From the basic SOC, our extensions of this algorithm, and ideas from conventional adaptive control [3, 55], we will introduce a new fuzzy learning control approach. This new control algorithm offers improved performance feedback by utilizing a reference system model whose response characteristics meets the desired design characteristics for the process. Due to the introduction of this reference model, we refer to this new technique as a fuzzy model reference learning controller (FMRLC). In the development of this chapter we highlight some of the differences between learning control systems and adaptive control systems.

3.1 Applications and Techniques in Adaptive Fuzzy Control

Over recent years, several articles have appeared which describe a variety of adaptive/learning fuzzy control schemes. Therefore, in this section we present an

overview of this literature.

One of the earliest examples of an adaptive fuzzy control algorithm was performed by Procyk and Mamdani [59] of Queen Mary College. Here, Procyk and Mamdani develop an adaptive fuzzy control scheme which they refer to as a "linguistic SOC". The SOC is capable of automatically improving its control policy by modifying a fuzzy controllers knowledge-base so that a given performance measure is minimized. More discussion of this performance measure is presented later in this chapter. Notice that since the knowledge-base is being modified, it could be argued that the fuzzy controller is "learning". However, due to the fact that at the time there was no consensus of opinion as to the meaning of "learning", Procyk and Mamdani refer to their algorithm as a linguistic self-organizing controller (SOC). Procyk illustrates the versatility of this technique by employing it on a variety of single-input single-output (SISO), multiple-input multiple-output (MIMO), and non-linear systems.

The linguistic SOC framework of Procyk and Mamdani was studied further in a paper authored by Shao in [66]. One important result of this work was the introduction of a rule base modification algorithm designed to reduce computation time and memory. Shao demonstrated the practical application of the linguistic SOC by employing it in two real time applications which contained non-linearities and large time lags. These applications included both a motor speed controller and a heater temperature controller. In each case, the linguistic SOC algorithm effectively re-organized the fuzzy controller to obtain "good" control.

One of the more interesting applications of Procyk and Mamdani's linguistic SOC framework was implemented by Scharf and Mandic [64] and again by Tanscheit and Scharf [79] on a multiple degree-of-freedom robot arm. Their design employs an

improved performance measure decision table which was proposed by Yamazaki of London University [86]. Due to the position varying load torques that are applied to the motor actuators, the process is quite non-linear and time-varying. Scharf and his colleagues investigated the controller's ability to re–organize when the robot was maneuvered through various paths in space. The results of this work have shown that the framework for SOC compares favorably with more traditional control techniques such as PID controllers. This work also demonstrates that current computer systems can be used to implement the linguistic SOC for controlling real time systems with sample times of only a few milliseconds.

Isaka, Sebald, Karimi, Smith, and Quinn in [36] demonstrate a practical application of SOC by employing it as a blood pressure controller. In this application the controller monitors a human patient's blood pressure and adjusts the rate in which a particular drug is administered such that the blood pressure is maintained at a desired level. This is an ideal application of the linguistic SOC algorithm since the dynamic responses of the human body varies greatly between various persons.

Daley and Gill in [18, 19, 20] describe the application of the linguistic SOC algorithm proposed by Procyk and Mamdani for a complex multi-variable process involving the altitude control of a flexible satellite. In this research, it was found that SOC provided "reasonably good" control for this process, thus demonstrating the potential of utilizing the linguistic SOC for very complex processes. Moreover, in [18] Daley and Gill present an empirical comparisons between the linguistic SOC and the proportional-derivative (PD) control when employed for the satellite system for which the SOC compared vary favorably.

Up to this point in the literature review we have introduced and discussed applications of the linguistic SOC presented by Procyk and Mamdani. However other

adaptive fuzzy control schemes exist which fit into a similar class of adaptive methods. In the following we discuss articles which describe control algorithms involving the adaptation of a direct fuzzy controller such that certain performance criteria are met.

One such technique was presented by Barolini *et al.* in [5]. This algorithm involves the modification of the membership functions used in defining the fuzzy sets which were employed in the knowledge base of a direct fuzzy controller. In this research, the author considers only two types of membership functions. These membership functions are expressed here with generic arguments in Equations (3.1) and (3.2) below:

$$\mu(x) = \left[\frac{1}{1 + a(x-c)^b}\right], \tag{3.1}$$

$$\mu(x) = e^{\frac{-(x-m)^2}{2\sigma^2}}. \tag{3.2}$$

These membership functions are modified by changing the values of a, b, c, and m such that given performance criteria are met. The overall effectiveness of this algorithm is illustrated empirically by means of application on a continuous casting plant.

Another algorithm which modifies the membership functions of the fuzzy sets implemented in a direct fuzzy controller is presented by Takahashi in [76]. In this article, a direct fuzzy controller is employed as an automotive speed control device. Also, included in this control scheme is another fuzzy system which predicts individual driving preferences and characteristics. The information obtained from the "characteristics evaluator" fuzzy system is used to modify the fuzzy controller mem-

bership functions so that the vehicle acceleration during a speed resume operation is consistent with the individual operators preferences.

All of the adaptive fuzzy control techniques discussed above fit into the class of adaptive systems often referred to as "direct adaptive control". This means that some limited amount of knowledge is known about the process, and the controller updates are made based both on this knowledge and some performance measure. Alternatively, information about the process may be obtained indirectly by utilizing system identification. Therefore, if controller updates/adaptations are made based on an identified process, then the control scheme is often referred to as an "indirect adaptive controller".

Several examples of fuzzy identification have been presented in the literature. These algorithms take two approaches including: 1) fuzzy modeling of static systems and 2) fuzzy modeling of dynamic systems.

Identification methods of the first type typically involve fuzzy modeling of numerical data. Here we assume some static correlation exist between sets of numerical data. Therefore the modeling process involves the determination of fuzzy sets and fuzzy implications which describe the relationship between the data sets. Examples of process modeling of numerical data is presented by Takagi and Sugeno in [74] and Wang and Mendel in [83].

Fuzzy modeling of dynamical systems involves modeling the process by means of a discrete time fuzzy relational equation of the form:

$$\hat{X}((k+1)T) = (\hat{U}(kT) \times \hat{X}(kT)) \circ R \tag{3.3}$$

where the time $t = kT$, $\hat{U}(kT)$ denotes the cross product of the fuzzified inputs, $\hat{X}(kT)$ denotes the cross product of the fuzzified process state, and R denotes the

fuzzy relation which describes the relationship between process inputs, states, and next states. Articles which discuss discrete time fuzzy process modeling include: Tong in [81] and Cumani in [14]. However for fuzzy identification of dynamical systems, the objective is to develop an algorithm which automatically generates the fuzzy relation R. Articles which describe how this may be accomplished are presented by Czogała and Pedrycz in [15, 16] and Batur *et al.* in [6].

Over recent years several examples of indirect adaptive fuzzy control have appeared in the literature, some of which employ the fuzzy identification algorithms discussed above. Therefore we will briefly overview some of the algorithms and applications of these techniques.

Rhee *et al.* [84] present a knowledge-based fuzzy control system. In this algorithm a knowledge base is constructed which contains a number of time responses for the processes which is to be controlled. In general this knowledge-base is constructed off-line by stimulating the system with a variety of inputs. These inputs are generally chosen such that some measure of *persistent excitation* is met [3]. A fuzzy system then accesses the information contained within the knowledge-base to make control decisions.

Another example of indirect adaptive fuzzy control was presented by Graham and Newell in [29, 30]. By utilizing the previously discussed fuzzy identification algorithm developed by Czogala and Pedrycz, this algorithm performs simultaneous fuzzy identification and control. Performance is determined by a set of fuzzy sets, one for each performance attribute such as error from the desired output. The performance with respect to a particular performance attribute is given by the degree of its current value. A control action is determined directly from an identified "fuzzy process model" and the membership values of the performance attributes. Graham

and Newell refer to this control algorithm as a "fuzzy model based adaptive controller". A practical application of this algorithm is illustrated when it is employed to controlling the liquid level of a mass storage tank.

Here we have presented an overview of the literature for adaptive fuzzy control systems which involve adaptation of a "direct fuzzy controller". However, recently several articles have appeared for adaptive control algorithms which utilize a fuzzy system to modify other types of controllers other than direct fuzzy controllers. In this case the fuzzy system may be thought of as a higher level "supervisor" for a lower level classical controller. Consequently, this method of adaptation is often referred to as fuzzy supervision of classical controllers.

For example, a fuzzy system may be employed to modify the gains of proportional integral derivative (PID) control law such that the controller meets a given performance criteria. In general, this design approach may be desired when it is difficult or impossible to obtain an accurate model of the process to be controlled thus making it very difficult to select suitable controller gains. Furthermore, for time varying processes it may be necessary to vary the controller gains to maintain good control. However, given experience in tuning a PID controlled process a control engineer will often know which way to "turn the knobs" on the controller gains to obtain a more desirable response. Therefore, fuzzy set theory provides a the mathematical framework to capture the control engineers experience for use in a self-tuning algorithm.

An early example of fuzzy supervision of PID controllers was presented by van Nauta Lemke and De-zhao in [56]. In this research a method of fuzzy system performance evaluation is presented. The performance of the controller is evaluated based on several performance criteria including: rise-time, overshoot, integral squared er-

ror (ISE), and/or integral over time absolute error (ITAE). Given these performance criteria above, the performance evaluation fuzzy system computes a numeric performance index for each of the PID controller gains. If $p_p(kT)$, $p_i(kT)$, and $p_d(kT)$ denote the performance index for the proportional, integral, and derivative, respectively, at time $t = kT$, then the controller gains at this time may be computed as

$$k_p(kT) = k_p(kT - T)(1 + p_p(kT)) \tag{3.4}$$

$$k_i(kT) = k_i(kT - T)(1 + p_i(kT)) \tag{3.5}$$

$$k_p(kT) = k_d(kT - T)(1 + p_d(kT)) \tag{3.6}$$

where $k_p(kT)$, $k_i(kT)$, and $k_d(kT)$ denote the proportional, integral, and derivative controller gains,respectively.

Another example of fuzzy supervision of PID controller is presented by de Silva [21, 22, 23]. In this work de Silva presents the general framework for which fuzzy supervision of PID controllers may be implemented. The performance criteria which are considered include: rise-time, steady state error, and overshoot/oscillations which are defined in the traditional manner. In addition, de Silva includes a performance criteria referred to as "steady divergence" which provides a measure of stability for the overall system. Therefore, the fuzzy system employed in this framework consist of a knowledge base whose rules may be expressed in a single-input single-output form. For example, one such rule may be expressed linguistically as

If steady state error is positive big **Then** the change in the integral gain k_i is positive big.

In general, the controller may be modified only after step input change due to the

fact that rise time and overshoot are defined only for step inputs. Further results of this research involve an intuitive approach for analyzing some of the stability aspects of the system. However, the stability of the overall system is yet to be shown. Finally, de Silva demonstrates the practical implementation of this algorithm when controlling a non-minimum phase process with large time delays.

Oliveira, Lima, and Sentieiro in [57] present a control scheme that is similar to de Silva's algorithm and involves fuzzy supervision of a PID controller. However, the performance criteria which are considered include only the rise-time and overshoot. As a result, the controller can be modified only after a step input changes. The effectiveness of this control algorithm is studied by simulation of two arbitrary systems and implementation on a scaled down storage tank for which the liquid level was the controlled process parameter.

Tzafestas and Papanikolopoulos in [82] present a fuzzy supervision algorithm which utilizes a performance evaluation fuzzy system similar to that of Procyk and Mamdani described above. In this algorithm, a performance evaluator monitors the error and change in error of the process to obtain a measure of the deviation of the output from a desired response. For instance, a positive error with a negative change in error may be viewed as more desirable than a positive error with a positive change in error due to the fact that for the later case the process is moving in the wrong direction. Therefore, such information may be quantified in a fuzzy system. The PID controller gains may be modified by the following equations:

$$k_p(kT) = k_p(kT - T) + c_p * p_f(kT) \tag{3.7}$$

$$k_i(kT) = k_i(kT - T) + c_i * p_f(kT) \tag{3.8}$$

$$k_d(kT) = k_d(kT - T) + c_d * p_f(kT) \tag{3.9}$$

where c_p, c_i, and c_d denote constants which indicate the magnitude and direction of the necessary gain changes and $p_f(kT)$ denotes the performance factor as computed by the performance evaluation fuzzy system. This algorithm was implemented and tested for a third order linear system. Experiments were made for the cases where the PID controller was initially loaded with the nominal controller gains determined analytically by Ziegler-Nichols method and by Kalman's method. For both cases the adaptation mechanism made significant improvements over the initial PID controller.

In [58] Ollero and Garcia-Cerezo present a fuzzy supervision algorithm which is very similar to gain scheduling. As in gain scheduling, we assume the process dynamics are dependent upon the current operation conditions. Therefore, generally we generate an analytical relationship between "optimal" controller gains and the operating conditions. Therefore, given this analytical model which relates the "optimal" operating conditions and the controller gains, we may continually update the controller gains by monitoring the operating conditions. However, in some situations an analytical relationship between the operating conditions and controller gains may be difficult to obtain. Although, given some experience a control engineer may be able to determine qualitatively which controller gains provide good control under various operation conditions. This knowledge may then be quantified by a fuzzy system to modify the PID controller gains. The resulting algorithm was implemented an test on a heater temperature control device for which the lower level direct controller was the proportional integral (PI) control law.

An alternative method of using fuzzy set theory to modify PID controllers is presented by Anderson, Blankenship, and Lebow in [2]. In this research, fuzzy sets are employed to quantify the desired performance of the overall system. However, fuzzy implication and composition is not utilized. The desired performance is quantified

by assigning a fuzzy set whose membership function is centered about the desired level of performance. For example, if a rise-time of t_{r_d} is desired, then a fuzzy set may be defined around t_{r_d} such that its membership function take on the value of 1, or full membership, at a rise time of t_{r_d} and falls off to zero in a convex and possibly symmetric fashion the further away from t_{r_d}. Thus, the change in a PID controller gains may be effected by the following pseudo- *condition* $\rightarrow$ *action* rule:

If rise time needs to be increased **Then**

$$k_p(kT) = k_p(kT + c_p * (1 - \mu(t_r) \tag{3.10}$$

$$k_i(kT) = k_i(kT + c_i * (1 - \mu(t_r)) \tag{3.11}$$

$$k_d(kT) = k_d(kT + c_d * (1 - \mu(t_r)) \tag{3.12}$$

where c_p, c_i, and c_d denote constants which indicate the magnitude and direction of the necessary gain changes and $\mu(t_r)$ denotes the membership of the current rise time.

3.2 A Framework For Adaptive Fuzzy Control Systems

For systems which are controlled manually by humans, the operator relies on past experience to continually modify control actions to achieve ever increasing system performance. This human process is usually referred to as "learning". However, in many instances an experienced human expert may not exist. Therefore as presented in the literature overview the process of learning has been implemented in some fuzzy system designs. In such systems it is possible to monitor various combinations of controller inputs, plant inputs, and plant outputs to modify the existing controller to improve its performance over time.

Figure 25 below illustrates the basic framework for an adaptive fuzzy control system. In Figure 25, $\underline{y_r}(t)$ denotes the reference input signals, $\underline{y}(t)$ denotes the

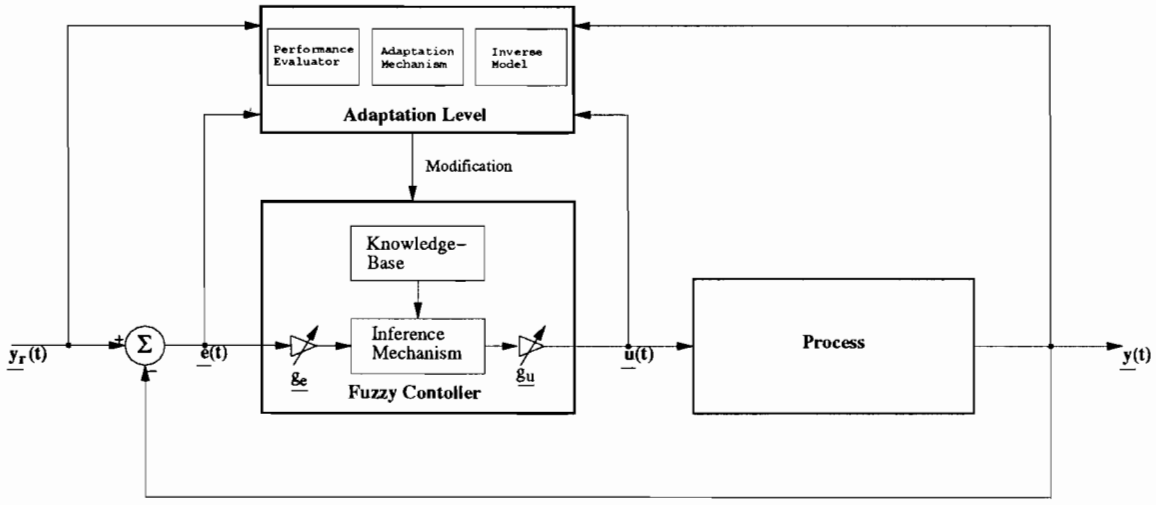


Figure 25: Basic framework for an adaptive fuzzy system.

output signals, $\underline{e}(t)$ denotes the error signals, $\underline{u}(t)$ denotes the process inputs, and $\underline{g_e}(t)$ and $\underline{g_u}(t)$ denote the normalizing controller gains. The underlining of the above variable quantities indicates the possibility that these may be vectors quantities. In order to provide a concise illustration, in Figure 25, we show only the error as a controller input. However, for the more generalized case we could consider other inputs such as the process outputs or states, functions of the process outputs or states, and functions of the process error.

Also as shown in Figure 25, for the adaptation level we may monitor the process outputs, inputs, reference inputs, and error to make decisions about controller modifications or adaptations. In general for adaptive fuzzy control systems, the adaptation level is composed of three basic subcomponents including:

- Performance Evaluator,
- Inverse Model,
- Adaptation Mechanism.

The function of each subcomponent is discussed below.

The primary function of the performance evaluator is to generate some measure of system performance with respect to the desired system performance. Typically the performance is specified by measures such as steady state error, rise-time, and overshoot. However, as will be shown later in this chapter there exist other more complex methods of specifying system performance.

The inverse model simply characterizes the inverse relationship between process inputs and outputs. For instance, given a desired change in the output of the system, the inverse model should be capable of providing some information as to the changes in the inputs necessary to obtain the desired system response. In conventional adaptive control frameworks, process models are often very analytical; however, for adaptive fuzzy systems the inverse model may indicate only the direction of the input changes, i.e. an increase or decrease of the inputs. The inverse model may be obtained in two ways. The first method involves simply generating the inverse model directly from *a priori* knowledge about the process. Alternatively, this knowledge may be obtained indirectly by an on-line and/or an off-line identification algorithm.

Finally, the adaptation mechanism utilizes the information provide by the performance evaluator and the inverse model to modify the controller. For adaptive fuzzy systems of the framework shown in Figure 25 several types of modifications are possible including:

- Modification of rules - number and structure,
- Modification of the fuzzy sets - membership functions,
- Modification of the inference mechanism,
- Modification of the defuzzification strategy,
- Modification of the normalizing gains,
- Any combination of the above.

In this chapter we will focus primarily on the modification of the fuzzy set membership functions. In particular, we will consider only the modification of the fuzzy sets associated with the consequence of the fuzzy implications which were implemented in the fuzzy controller.

3.3 Linguistic Self Organizing Controller (SOC)

Since the development of the FMRLC is based on the linguistic SOC of Procyk and Mamdani in [59], this section illustrates the basic architecture and design for the linguistic SOC. Based on this discussion, possible design problems and issues will be identified, thus providing the foundation for improvements which will be incorporated in the FMRLC algorithm.

The basic architecture of the linguistic SOC is presented below in Figure 26. This architecture consists of a dynamic process, a direct fuzzy controller, and an adaptation level. In the linguistic SOC algorithm, the performance evaluator is a fuzzy system. Given vectors $\underline{e}(kt)$ and $\underline{c}(kt)$ the performance fuzzy system generates a vector $\underline{p_o}(kT)$ of desired output changes. The inverse model is a matrix which operates on the vector $\underline{p_o}(kT)$ to produce a vector $\underline{p_i}(kT)$ consisting of necessary input

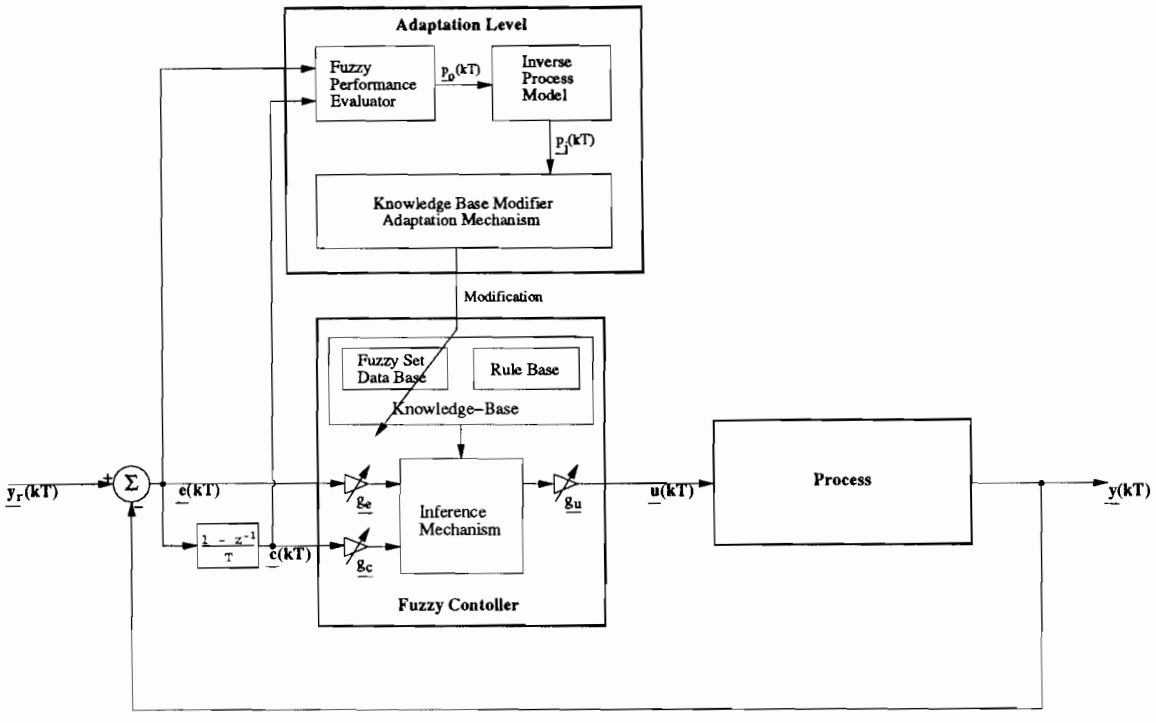


Figure 26: Functional architecture for the linguistic self-organizing controller.

changes. Finally, the adaptation mechanism performs a modification of the fuzzy set membership functions of the fuzzy controller implications which most contributed to the current output. More details of this process will be presented next where each controller component is discussed individually.

3.3.1 Fuzzy Controller Design

The process in Figure 26 is assumed to have r inputs denoted by $\underline{u}(kT)$ and s outputs denoted by $\underline{y}(kT)$ (T is the sample period). The underline in used to denote that these are, in general, vector quantities. The inputs to the controller are the

process error $\underline{e}(kT)$ and change in error $\underline{c}(kT)$ defined as

$$\underline{e}(kT) = \underline{y}_r(kT) - \underline{y}(kT), \tag{3.13}$$

$$\underline{c}(kT) = \frac{\underline{e}(kT) - \underline{e}(kT-T)}{T}, \tag{3.14}$$

respectively, where $\underline{y}_r(kT)$ denotes the desired process output. Throughout this thesis we will assume that these are the only inputs to the fuzzy controller. However, in general, many different combinations of controller inputs are possible such as functions of the error, the process states, and/or the process operating conditions.

For greater flexibility in fuzzy controller implementation, the universes of discourse for each process input are "normalized" to the interval $[-1, +1]$ by means of constant scaling factors. This will provide a capability for easy controller modifications. Also, a single fuzzy controller subroutine may be used to implement more than one fuzzy controller. For our fuzzy controller design, the controller gains $\underline{g}_e$, $\underline{g}_c$, and $\underline{g}_u$ were employed to normalize the universe of discourse for the process error $\underline{e}(kT)$, change in error $\underline{c}(kT)$, and controller output $u(kT)$, respectively.

The "normalized" fuzzy controller decision making process may be expressed by

$$\begin{aligned}\hat{U}_i(kT) = ((\hat{E}_1(kT) \times \hat{E}_2(kT) \times ... \times \hat{E}_s(kT)) \times \\ (\hat{C}_1(kT) \times \hat{C}_2(kT) \times ... \times \hat{C}_s(kT))) \circ R_i\end{aligned} \tag{3.15}$$

where $\hat{E}_j(kT)$ and $\hat{C}_j(kT)$ denote the fuzzified error and change in error, respectively associated with the j^{th} process output, $\hat{U}_i(kT)$ denotes the implied fuzzy set for the i^{th} process input, and R_i denotes the overall fuzzy relation which relates all errors and change in errors with the i^{th} input. This implies that a fuzzy controller is needed for

each process input; however, for sake of brevity only a single fuzzy controller is shown in Figure 26. See Chapter II for a mathematical explanation of Equation (3.15).

3.3.2 The Performance Evaluation Fuzzy System

In order to improve its control strategy an adaptive or learning controller must be capable of assessing it own performance. As previously stated, performance evaluation for the linguistic SOC algorithm is performed by a fuzzy system.

From a practical perspective, a control engineer generally has in mind the system response characteristics which can be tolerated. Often, this desired behavior may be qualitatively expressed by a set of linguistic rules. Thus, this performance is expressed linguistically by a set of *condition* $\rightarrow$ *action* rules where a typical linguistic rule may be expressed as:

IF error is positive big **AND** change in error is negative big

THEN the process is well behaved.

This performance rule may be interpreted more specifically as; "the process is very far from the set point but is moving toward the set point at great speed, therefore the performance is good". Procyk and Mamdani propose that this linguistically expressed performance be quantified by fuzzy set theory and implemented in a fuzzy system.

An "optimized" version of this "fuzzy performance evaluator" was discussed and implemented by Scharf and Mandic [64] and again by Tanscheit and Scharf [79] for a multi-degree of freedom robot arm. Below in Table 6 we present a typical rule base array table for this "optimized" fuzzy performance evaluator.

Recall from Chapter II, the numeric elements contained within this rule base array denote the generic labels for linguistic values whose range may be linguistically

Table 6: Rule base array for the fuzzy performance evaluator.

P_o^l		C^k										
		-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
E^j	-5	-5	-5	-5	-5	-5	-5	-4	-3	-2	-1	$+0$
	-4	-5	-5	-5	-5	-5	-4	-3	-2	-1	$+0$	$+1$
	-3	-5	-5	-5	-5	-4	-3	-2	-1	$+0$	$+1$	$+2$
	-2	-5	-5	-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$
	-1	-5	-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$
	0	-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
	$+1$	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+5$
	$+2$	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+5$	$+5$
	$+3$	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+5$	$+5$	$+5$
	$+4$	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+5$	$+5$	$+5$	$+5$
	$+5$	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+5$	$+5$	$+5$	$+5$	$+5$

expressed as "negative big" to "positive big". For example, -5 denotes the most "negative" linguistic values and $+5$ denotes the most "positive" linguistic values. Further, in Table 6, E^j and C^k denote fuzzy sets quantifying conditions for the error and change in error, respectively. Likewise, P_o^l denotes the output fuzzy sets which quantify the performance of the process. The output of the fuzzy performance evaluator expresses a rough indication of the desired magnitude of output changes necessary for good control. These output changes may then be used by an inverse process model to determine corresponding changes in the input such that the desired output changes are met. Therefore, often the performance of the fuzzy controller is evaluated for each input with respect to each of the process outputs. This implies that for a fuzzy performance evaluator with a rule base array shown above, a fuzzy performance evaluator is needed for each process input and output. For simplicity only one fuzzy performance evaluator is shown in Figure 26 above.

The zero entries in Table 6 correspond to conditions where no correction in the output is necessary. Thus, the zero entries represent a desired "trajectory" within the performance rule base array. Likewise, the non-zero values of the rule base array indicate the relative amount of correction necessary for the system to approach the desired trajectory. Therefore, Table 6 not only takes into account the distance from the set point but also the rate of approach to it. This quantifies the fact that a small error with a small change in error is more desirable than a small error and a large change in error due to the overshoot that is likely to occur. As a result, Table 6 is designed to provide a fast rise-time with good damping around the set point to reduce overshoot.

Often for practical implementation of the fuzzy performance evaluator, the input and output universes of discourse are "normalized" by means of normalizing gains which are similar to those employed in the direct fuzzy controller. In the original work by Procyk and Mamdani these gains were chosen to be the same as those used for the fuzzy controller itself. In our research, we found that allowing these gains to be chosen independent of the fuzzy controller gains provided greater flexibility and often better control.

As a result of using a fuzzy system for performance evaluation, the performance of the fuzzy controller is simply a static function of the error and change in error. If $\underline{e}(kT)$ and $\underline{c}(kT)$ denote vectors of error and change in error, respectively, for each process output at time $t = kT$, then the vector $\underline{p}_o(kT)$ corresponding to deviation of the outputs from the desired trajectory may be computed by the fuzzy performance evaluator. This mapping of $\underline{e}(kT)$ and $\underline{c}(kT)$ to $\underline{p}_o(kT)$ may be conveniently

expressed as

$$\underline{p}_o(kT) = f_{fpe}(\underline{e}(kT), \underline{c}(kT)) \tag{3.16}$$

where f_{fpe} denotes the operation provided by the fuzzy performance evaluator.

3.3.3 Inverse Process Model

As previously mentioned the inverse model consists of a matrix which is used to operate on the vector of desired output changes $\underline{p}_o(kT)$ to produce a vector $\underline{p}_i(kT)$ consisting of necessary input changes.

Consider a multi-variable process characterized by the state space expressions:

$$\dot{\underline{x}}(t) = f(\underline{x}(t), \underline{u}(t)) \tag{3.17}$$

$$\underline{y}(t) = g(\underline{x}(t), \underline{u}(t)) \tag{3.18}$$

where $\underline{x}(t)$ denotes the vector of process states, $\underline{u}(t)$ denotes the vector of process inputs, and $\underline{y}(t)$ denotes the vector of measured outputs (the appropriate continuity assumptions are made to ensure existence and uniqueness of solutions). Under appropriate assumptions (e.g. that f and g are differentiable in $\underline{x}$ and $\underline{u}$) for a small change in the input vector $\underline{u}(t)$ and subsequent change in the state vector $\underline{x}(t)$, the change in the state derivative and process output is given by

$$\delta\dot{\underline{x}}(t) = \frac{\partial f}{\partial \underline{x}^T(t)}\delta\underline{x}(t) + \frac{\partial f}{\partial \underline{u}^T(t)}\delta\underline{u}(t) \tag{3.19}$$

$$\delta\underline{y}(t) = \frac{\partial g}{\partial \underline{x}^T(t)}\delta\underline{x}(t) + \frac{\partial g}{\partial \underline{u}^T(t)}\delta\underline{u}(t), \tag{3.20}$$

respectively, where

$$\frac{\partial f}{\partial \underline{x}^T(t)} \triangleq \begin{bmatrix} \frac{\partial f_1}{\partial \underline{x}_1(t)} & \frac{\partial f_1}{\partial \underline{x}_2(t)} & \cdots & \frac{\partial f_1}{\partial \underline{x}_m(t)} \\ \frac{\partial f_2}{\partial \underline{x}_1(t)} & \frac{\partial f_2}{\partial \underline{x}_2(t)} & \cdots & \frac{\partial f_2}{\partial \underline{x}_m(t)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \underline{x}_1(t)} & \frac{\partial f_m}{\partial \underline{x}_2(t)} & \cdots & \frac{\partial f_m}{\partial \underline{x}_m(t)} \end{bmatrix} \tag{3.21}$$

$$\frac{\partial f}{\partial \underline{u}^T(t)} \triangleq \begin{bmatrix} \frac{\partial f_1}{\partial \underline{u}_1(t)} & \frac{\partial f_1}{\partial \underline{u}_2(t)} & \cdots & \frac{\partial f_1}{\partial \underline{u}_r(t)} \\ \frac{\partial f_2}{\partial \underline{u}_1(t)} & \frac{\partial f_2}{\partial \underline{u}_2(t)} & \cdots & \frac{\partial f_2}{\partial \underline{u}_r(t)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \underline{u}_1(t)} & \frac{\partial f_m}{\partial \underline{u}_2(t)} & \cdots & \frac{\partial f_m}{\partial \underline{u}_r(t)} \end{bmatrix} \tag{3.22}$$

$$\frac{\partial g}{\partial \underline{x}^T(t)} \triangleq \begin{bmatrix} \frac{\partial g_1}{\partial \underline{x}_1(t)} & \frac{\partial g_1}{\partial \underline{x}_2(t)} & \cdots & \frac{\partial g_1}{\partial \underline{x}_m(t)} \\ \frac{\partial g_2}{\partial \underline{x}_1(t)} & \frac{\partial g_2}{\partial \underline{x}_2(t)} & \cdots & \frac{\partial g_2}{\partial \underline{x}_m(t)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_s}{\partial \underline{x}_1(t)} & \frac{\partial g_s}{\partial \underline{x}_2(t)} & \cdots & \frac{\partial g_s}{\partial \underline{x}_m(t)} \end{bmatrix} \tag{3.23}$$

$$\frac{\partial g}{\partial \underline{u}^T(t)} \triangleq \begin{bmatrix} \frac{\partial g_1}{\partial \underline{u}_1(t)} & \frac{\partial g_1}{\partial \underline{u}_2(t)} & \cdots & \frac{\partial g_1}{\partial \underline{u}_r(t)} \\ \frac{\partial g_2}{\partial \underline{u}_1(t)} & \frac{\partial g_2}{\partial \underline{u}_2(t)} & \cdots & \frac{\partial g_2}{\partial \underline{u}_r(t)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_s}{\partial \underline{u}_1(t)} & \frac{\partial g_s}{\partial \underline{u}_2(t)} & \cdots & \frac{\partial g_s}{\partial \underline{u}_r(t)} \end{bmatrix} \tag{3.24}$$

and where m is the number of states, r is the number of inputs, and s is the number of measured outputs.

For small time samples T, input change $\Delta\underline{u}(kT)$ will result in a state vector change $\Delta\underline{x}(kT)$ and output vector change $\Delta\underline{y}(kT)$ as approximately expressed by

$$\Delta\underline{x}(kT) \approx T\Delta\dot{\underline{x}}(kT) = T\frac{\partial f}{\partial \underline{x}^T(kT)}\Delta\underline{x}(kT) + T\frac{\partial f}{\partial \underline{u}^T(kT)}\Delta\underline{u}(kT) \tag{3.25}$$

$$\Delta\underline{y}(kT) \approx \frac{\partial g}{\partial \underline{x}^T(kT)}\Delta\underline{x}(kT) + \frac{\partial g}{\partial \underline{u}^T(kT)}\Delta\underline{u}(kT), \tag{3.26}$$

respectively. However, assuming the appropriate inverse exist, Equation (3.25) may be rearranged to obtain

$$\Delta\underline{x}(kT) \approx \left[\left(I_m - T\frac{\partial f}{\partial \underline{x}^T(kT)}\right)^{-1} T\frac{\partial f}{\partial \underline{u}^T(kT)}\right]\Delta\underline{u}(kT) \tag{3.27}$$

where I_m is an $m \times m$ dimensional identity matrix.

Substituting Equation (3.27) into Equation (3.26), we find that the overall output vector change $\Delta\underline{y}(kT)$ resulting from an input vector change $\Delta\underline{u}(kT)$ may be approximated by

$$\Delta\underline{y}(kT) \approx M(kT)\Delta\underline{u}(kT) \tag{3.28}$$

where

$$\mathrm{M(kT)} = \left\{\frac{\partial \mathrm{g}}{\partial \underline{\mathrm{x}}^{\mathrm{T}}(\mathrm{kT})}\left[\left(\mathrm{I_m} - \mathrm{T}\frac{\partial \mathrm{f}}{\partial \underline{\mathrm{x}}^{\mathrm{T}}(\mathrm{kT})}\right)^{-1}\mathrm{T}\frac{\partial \mathrm{f}}{\partial \underline{\mathrm{u}}^{\mathrm{T}}(\mathrm{kT})}\right] + \frac{\partial \mathrm{g}}{\partial \underline{\mathrm{u}}^{\mathrm{T}}(\mathrm{kT})}\right\}. \tag{3.29}$$

Given that $\underline{p}_o(kT)$ denotes a vector of required output changes the vector of input changes $\underline{p}_i(kT)$ can be computed from

$$\underline{p}_i(kT) = M^{-1}(kT)\underline{p}_o(kT). \tag{3.30}$$

Therefore the matrix $M^{-1}(kT)$ may be viewed as a an inverse model of the process. Of course this assumes that the matrix $M(kT)$ is invertible so that we may obtain $M^{-1}(kT)$.

As mentioned in Chapter II of this thesis, one fundamental advantage of fuzzy control results from the fact that a detailed mathematical description of the process to be controlled is not necessary. Therefore, generating the inverse model based on

an exact mathematical model seems to eliminate this advantage of fuzzy control. Furthermore, the partial derivatives needed to generate the matrix $M^{-1}(kT)$ may be difficult to compute even when a process model is available.

It is easily shown that the matrix $M^{-1}(kT)$ will be a constant only for linear processes. However, due to the fact that the inverse model matrix $M^{-1}(kT)$ may be difficult or impossible to obtain, in practical implementation we will always constrain the matrix $M^{-1}(kT)$ to be a constant as others have and hope that the adaptation process will be able to compensate for any inaccuracies. In other words, due to inaccuracies in $M^{-1}(kT)$ the input corrections may not be exact, requiring repeated corrections. This sometimes slows the adaptation process.

In applications it is not necessary that $M^{-1}(kT)$ be an exact model. In fact, generally all that is needed is an idea about how each output is affected by each input. The designer must answer the following important questions:

1. How does each input affect each output (i.e. by increasing a given input does this result in a given output being increased/decreased monotonically)?

2. What is the relative magnitude of the output changes given an input change?

Once we have answered these questions we may build a normalized version of the matrix $M^{-1}(kT)$ such that the elements of the matrix have values only on the interval $[-1.0, +1.0]$. The normalized version of the inverse model can be incorporated due to the scaling factors already introduced to normalize the universes of discourse of the fuzzy performance evaluator.

Notice that question (1) above implies that the relationship between inputs and output should be monotonic. Procyk and Mamdani argue that this monotonic relationship exists for a very large class of systems.

3.3.4 Knowledge Base Modifier

The knowledge base modifier performs the function of modifying the fuzzy controller so that better performance is achieved. Given the information about the necessary changes in the input as expressed by the vector $\underline{p}_i(kT)$, the knowledge base modifier changes the knowledge base of the fuzzy controller so that the previously applied control rule will be modified for better control. In this subsection, we present a detailed description of how this may be accomplished. This includes the method of knowledge base modification as presented by Procyk and Mamdani and a new alternative method which has an overall effect that is similar to Procyk and Mamdani's method but is computationally simpler in some respects.

Consider the previously computed control action, which contributed to the present good/bad system performance. Note that $\underline{e}(kT-T)$ and $\underline{c}(kT-T)$ would have been the process error and change in error, respectively, at that time. Likewise, $\underline{u}(kT-T)$ would have been the controller output at that time. The controller output which would have been desired is expressed by

$$\underline{u}''(kT-T) = \underline{u}(kT-T) + \underline{p}_i(kT) \tag{3.31}$$

where $\underline{u}''(kT-T)$ denotes the desired input at time $t = kT - T$. By modifying the fuzzy controller's knowledge base we may force the fuzzy controller to produce this desired output given similar controller inputs.

Procyk and Mamdani's Knowledge Modification Algorithm

In order to develop the knowledge modification algorithm of Procyk and Mamdani, we consider the fuzzy controller associated with the i^{th} process input. We construct fuzzy sets on the appropriate universes of discourse around the previous

process error $e_j(kT-T)$ and change in error $c_j(kT-T)$ which are associated with the j^{th} process output and the previous input $u_i(kT-T)$ and desired previous input $u_i''(kT-T)$ which are associated with the i^{th} input. This process may be expressed by

$$\check{E}_j(kT-T) = fuzzifier(e_j(kT-T)) \tag{3.32}$$

$$\check{C}_j(kT-T) = fuzzifier(c_j(kT-T)) \tag{3.33}$$

$$\check{U}_i(kT-T) = fuzzifier(u_i(kT-T)) \tag{3.34}$$

$$\check{U}_i''(kT-T) = fuzzifier(u_i''(kT-T)) \tag{3.35}$$

where $fuzzifier$ represents the process of fuzzification. However, the fuzzification process here is different than that performed by the fuzzification interface which is typically implemented in a fuzzy system. Recall, when given a generic element $x_i \in \mathcal{X}$ the fuzzification interface employed in the fuzzy controller creates a fuzzy set X whose "crisp" membership function is defined as

$$\mu_X(x) = \begin{cases} 1, & x = x_i \\ 0, & otherwise. \end{cases} \tag{3.36}$$

In contrast, the fuzzification operation performed here generates a fuzzy set which has a membership function that is convex, symmetric, normal, and centered at the value x_i. Almost any such membership function may be chosen; here we consider only the triangular shaped membership function defined as

$$\mu_X(x) = \max\{0, 1 - \frac{|x_i - x|}{a}\}, \tag{3.37}$$

where a denotes the width of the membership function. The problem of controller modification may be expressed by saying that a fuzzy implication expressed as

$$\textbf{If } \check{E}_j(kT-T) \textbf{ and } \check{C}_j(kT-T) \textbf{ Then } \check{U}_i(kT-T).$$

should be removed and the fuzzy implication

$$\textbf{If } \check{E}_j(kT-T) \textbf{ and } \check{C}_j(kT-T) \textbf{ Then } \check{U}_i''(kT-T).$$

inserted. In the above fuzzy implications in order to maintain a concise presentation we have expressed only the fuzzy sets that are associated with the j^{th} process output. In general, the fuzzy sets associated with all process inputs are considered.

We may express the two fuzzy implications above as fuzzy relations $R_i'(kT-T)$ and $R_i''(kT-T)$ by forming cartesian products

$$R_i'(kT-T) = \check{E}_j(kT-T) \times \check{C}_j(kT-T) \times \check{U}_i(kT-T) \tag{3.38}$$

$$R_i''(kT-T) = \check{E}_j(kT-T) \times \check{C}_j(kT-T) \times \check{U}_i''(kT-T). \tag{3.39}$$

Let $R_i(kT-T)$ denote the actual fuzzy relation of the fuzzy controller which relates error and change in error associated with all outputs to the i^{th} input. There exist several possible methods of modifying $R_i(kT-T)$ such that we replace $R_i'(kT-T)$ with $R_i''(kT-T)$. For example, we may introduce a linguistic statement of the form

$$R_i(kT) = \{R_i(kT-T) \textit{ but not } R_i'(kT-T)\} \textit{ else } R_i''(kT-T) \tag{3.40}$$

Where the linguistic portion by be replaced by corresponding fuzzy set operations:

$$R_i(kT) = \{R_i(kT-T) \cap \neg R_i'(kT-T)\} \cup R_i''(kT-T). \tag{3.41}$$

Equation (3.41) forms the general method for modifying the controller.

An Alternative Knowledge Base Modification Algorithm

Equation (3.41) provides a useful technique for modifying the knowledge base when the universe of discourses are quantized into discrete elements such that a finite number of elements exist on each universe of discourse. Computing Equation (3.41)

for continuous universes of discourse becomes more difficult. This is due to the fact that the max and min operations used in implementing union or intersection operations, respectively, are simple for computers when the universes of discourse are discrete. Furthermore, since the number of entries in a fuzzy relation matrix is the product of the number of quantized levels for each controller input, a large amount of memory is required to store this fuzzy relation. This will often result in a large computation time due to the fact that a large number entries in the fuzzy relation may be changed for a given knowledge base modification.

In this section, we present a new knowledge base modification algorithm for continuous universes of discourse. This new algorithm takes a more intuitive approach. It is based on the modification of individual fuzzy implications used to implement the fuzzy controller rather than the modification of the overall fuzzy relation resulting from these rules. Since a fuzzy relation matrix is always much larger than a rule base table, this algorithm requires less computation time and memory.

The new modification algorithm modifies the process input fuzzy sets for the previous fuzzy implications which resulted in the previous input. The fuzzy implications, whose previous implied fuzzy set was not a null fuzzy set, are modified. All other fuzzy implications are left unchanged. As a result, the knowledge base modification may be performed by simply moving the membership functions of the process input (controller output) fuzzy sets associated with those fuzzy implications which most contributed to the previously applied inputs. Moving the membership functions for these fuzzy sets upward/downward along the appropriate universe of discourse by the amount specified by the vector $\underline{p}_i(kT)$, we obtain the desired controller modification. If the consequent fuzzy sets have membership functions defined such that they are symmetric and convex, the knowledge base modification reduces

to modifying only the center value of the membership functions.

By performing the above knowledge base modification algorithm, all implied fuzzy sets which resulted from the previous applied fuzzy implications will also be moved up/down the appropriate universe of discourse by an amount specified by $\underline{p}_i(kT)$. Consequently, the result of the max criteria (MC), mean of maximum (MOM), center of area (COA), and center of gravity (COG) defuzzification strategies [46, 47] is changed by an amount $\underline{p}_i(kT)$. Therefore, given a process error and change in error equal to the previous values, the above knowledge base modification algorithm should result in the desired process input regardless of the defuzzification strategy which was employed in the fuzzy controller.

Now that we have presented the basic concept as to how this modification takes place, we will consider some practical aspects of implementation as well as possible means of implementation.

First, consider a fuzzy system associated with the i^{th} process input such that the fuzzy sets for the process error and change in error have membership functions distributed along the appropriate universes of discourse such that

$$\sum_{k} \mu_{E_j^k}(e_j) = 1 \qquad \forall\, e_j \in \mathcal{E}_j \tag{3.42}$$

$$\sum_{k} \mu_{C_j^k}(c_j) = 1 \qquad \forall\, c_j \in \mathcal{C}_j, \tag{3.43}$$

where e_j and c_j denote the process error and change in error, respectively, for the j^{th} process output. Likewise, E_j^k and C_j^k denote the k^{th} fuzzy set associated with the error and change in error, respectively, for the j^{th} output. Given this constraint and that the system has s outputs, it is easily shown that at most 2^{2s} fuzzy implications contributed to the previously applied input. The knowledge base modification

algorithm should modify at most this number of fuzzy implications. The 2 in the exponent results from the fact that both error and change in error for each output are employed as inputs to the fuzzy controller.

Further, assume that all fuzzy sets assigned for a given fuzzy controller's output universe of discourse are symmetric, normal, convex, and shaped the same (e.g. all fuzzy set on the output universe of discourse could be assumed triangular shaped and of the same width). Given this assumption, any fuzzy set is completely characterized by its center value. Information about the fuzzy set defined for the output universe of discourse may be incorporated into a knowledge base array table. For example, in Table 7 below, we present a knowledge base array table where the entries in the knowledge base array table represent the center values of the membership functions associated with the implied fuzzy sets. It should be noted that Table 7 assumes a normalized universe of discourse. Generally, we assume the magnitude of all entries to be less than 1.0.

Given that the fuzzy controller employs a knowledge base array table similar to Table 7, the process of knowledge base modification reduces to a simple two step algorithm which is expressed below.

1. Determine which fuzzy implication in the knowledge base array table contributed most to the previously applied input. In other words, determine the fuzzy implications whose previous implied fuzzy set had at least one element with a membership value greater than 0.0.

2. Modify the entries in the knowledge base array for those fuzzy implications.

For example, assume that the previous process error $e(kT-T)$ was best characterized by the fuzzy set E^{+3} and E^{+4} shown in Table 7. Assume that the previous change

Table 7: Typical knowledge base array table.

$U^{j,k}$		C^k										
		−5	−4	−3	−2	−1	+0	+1	+2	+3	+4	+5
E^j	−5	+1.0	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0
	−4	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2
	−3	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4
	−2	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6
	−1	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8
	0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0
	+1	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0
	+2	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0
	+3	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0
	+4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0
	+5	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0

in the process error $c(kT-T)$ was best characterized by the fuzzy set C^{-4} and C^{-5}. The fuzzy implications which most contributed to the previously applied process are illustrated by the boxed entries in Table 7. In order to correctly modify the "normalized" knowledge base array, the desired change in the process input $p_i(kT)$ must be normalized by the same scaling factor which was introduced by the fuzzy controller for the process input. Hence, this normalized desired change in the process input, denoted $p_i'(kT)$, is expressed as

$$p_i'(kT) = \frac{p_i(kT)}{g_u}, \tag{3.44}$$

where g_u is the normalizing gain employed within the fuzzy controller. Given that the normalized desired change in the previous process input is $p_i'(kT) = 0.1$, after knowledge modification for the boxed values 0.4, 0.2, 0.0, and 0.2 in Table 7 we get 0.5, 0.3, 0.1 and 0.3, respectively, as shown in Table 8 below.

Table 8: Typical knowledge base array table after knowledge modification.

$P_i^{j,k}$		C^k										
		−5	−4	−3	−2	−1	+0	+1	+2	+3	+4	+5
E^j	−5	+1.0	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0
	−4	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2
	−3	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4
	−2	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6
	−1	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8
	0	+1.0	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0
	+1	+0.8	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0
	+2	+0.6	+0.4	+0.2	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0
	+3	+0.5	+0.3	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0
	+4	+0.3	+0.1	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0
	+5	+0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0

3.3.5 Discussion

In this section, we have presented the design details of the linguistic SOC controller which was presented by Procyk and Mamdani as well as a few new improvements to this algorithm. From this discussion, some of the major advantages of this control algorithm become apparent. For instance, this framework provides an *autonomous* capability since it has the ability to re-organize itself to find some "optimal" compromise between rise time and overshoot. Furthermore, the linguistic SOC algorithm does not require a detailed analytical description of the process for implementation. However, enough information is needed to specify the inverse model matrix M^{-1}. The transition between two adjacent fuzzy implications is "smoother" than for an expert system where ordinary sets are employed, resulting in fewer control rules. Since a direct fuzzy controller is employed, it implies that, in general,

fewer *condition* $\rightarrow$ *action* rules need to be determined than what would be required for a similar expert system adaptation [54].

Despite the many advantages and successes of linguistic SOC algorithm, several drawbacks do exist. For example, the performance measure employed in the linguistic SOC only characterizes some compromise between rise time and overshoot. For the more general control problem, the fastest possible rise time with some overshoot may not be a desirable process characteristic. For instance, if we wish to control the position of a passenger elevator, then for very fast process rise times the passengers may experience uncomfortable acceleration forces. Also, large overshoot between stops at floors could be very undesirable. It is not obvious how the fuzzy performance measure of Procyk and Mamdani should be modified to achieve a more desirable compromise between rise time and overshoot. This implies that some other method of performance evaluation should be used. In the FMRLC framework, we adopt ideas from conventional "model reference adaptive control" (MRAC) [3] by introducing a reference model for improved performance feedback.

Another potential problem which was briefly mentioned earlier, involves the assumption that incremental relationships between process inputs and outputs are monotonic when generating the inverse process model M^{-1}. For the general control problem this assumption does not always hold true. Consequently, major improvements are needed for the implementation of the inverse process model. Since M^{-1} in the linguistic SOC framework is generally obtained from a qualitative analysis of the process, in the FMRLC framework we propose implementing the inverse process model with a fuzzy system. This fuzzy system takes qualitative information about desired change in process outputs and maps this to a necessary change in the process inputs to obtain the output changes. Furthermore, since M^{-1} is generally a function

of process states or operating conditions, they may be employed as inputs into the fuzzy system to better quantify the incremental mapping between process inputs and outputs.

3.4 Learning vs. Adaptive Control

When Procyk and Mamdani presented the original linguistic SOC algorithm in 1979, they expressed some apprehension for using the phrase "learning" or "adaptive" when naming the algorithm. This was due, in part, to the fact that at the time there was little consensus of opinion as to the meaning of the word "learning". Thus, the less controversial phrase "self-organizing" was used. Even today it is still difficult to provide a precise and completely satisfactory definition of the phrase "learning control system". Farrell and Baker in [24] present the following definition of a learning process.

> A *learning control system* is one that has the ability to improve its performance in the future based on experimental information it has gained in the past, through closed loop interaction with the plant and it's environment.

This definition provides several immediate implications about the characteristics of learning control algorithms. For instance, a learning control algorithm should have the following characteristics:

- *Autonomous capability* - since it has the ability to improve it own performance,
- *Dynamic* - since learning occurs over time for dynamical systems,
- *Memory capabilities* - since past information is used to improve future performance,

- *Performance feedback* - to optimize the control decisions with respect to some specific measure.

Some may argue that the above characteristics are also characteristics of adaptive control algorithms. Farrell and Baker try to clearly distinguish the goals of the adaptive algorithm from those of the learning algorithm by the following statement:

> A control algorithm that treats every distinct operation situation as novel is limited to *adaptive* operation, whereas a system that correlates past experiences with past situations, and that can recall and exploit those experiences, is capable of *learning.*

Implied from this statement is the fact that the learning control algorithm must fully exploit the memory capabilities. The degree of emphasis and purpose placed upon the memory capabilities is the primary ingredient which distinguishes learning control from adaptive control. Furthermore, the phrase "learning control system" implies a capability to adjust it's memory to accommodate new experience.

The primary objective of the linguistic SOC of Procyk and Mamdani is to determine the "optimal" mapping between control signals, such as error and/or change in error, and the process inputs such that some performance objective is met. This process of determining the optimal mapping between control signals and process inputs is sometimes referred to as *controller identification.*

Given the above criteria for a learning control system, it can be argued that the linguistic SOC should be classified as a learning control algorithm. For example, in this framework the process of controller identification involves *performance feedback*, which is provided by the fuzzy performance evaluator, to provide a *dynamic* and *autonomous* method to improve the existing controller. In this algorithm, *memory*

capabilities are implemented in two ways. The first method by which memory has been incorporated in the linguistic SOC is immediately recognized; the knowledge base adaptation mechanism retains information about previous control signals and the previously applied control actions. The second method involves the local nature in which the controller modification occur. For example, once an effective set of control actions is determined for a given set of control signals, control actions will likely be available the next time the same control signals occur. This is because they are not affected by modifications which occur when the process is operating under other conditions. The overall system remembers its previous experiences about the system.

Let's consider other adaptive control algorithms such as model reference adaptive control (MRAC) [3]. For an MRAC algorithm, the parameters of a classical control law are adjusted such that the overall system behaves like a reference model. For instance, assume that the MRAC algorithm was designed to modify the controller gains of a PID control law. The MRAC algorithm continually updates these PID controller gains. The effect of changing any one of the controller gains is global with respect to all possible control states. All of the information gained from previous experiences is quickly lost.

The localized nature of controller modification in the linguistic SOC closely resembles what humans do when learning about situations which occur in life. For example, in life a person will monitor the condition or state of a situation and the factors which affect these conditions. Given a particular set of conditions and previous experience with similar situations, a person may approximately know how best to respond to the existing conditions to obtain a desired response. A person generally realizes the limitations of acquired knowledge when dealing with situations

that are very different from previous experiences. To deal with unfamiliar situations often a person will try to deduce a response based on previous experience; however, in general the chosen response is nothing more than a guess. Only after sufficient experience with a given situation or similar situations is a person truly capable of choosing an appropriate response. Once the appropriate response is determined, it is modified only if new experience with similar situations indicates that modification is necessary. The entire learning process for humans tends to be local in nature. Based on this intuitive analysis of the human learning process, we observe that, in general, a learning algorithm should utilize its memory capabilities to build knowledge only for experiences which are "local" to past experiences (the reader may recognize here some similarity to notions of learning and generalization in neural networks).

3.5 Fuzzy Model Reference Learning Control

In this section, we present an entirely new learning control algorithm based on some of the concepts presented by Procyk and Mamdani in [59], our new rule-base modification algorithm introduced in Section 3.3, and ideas from conventional adaptive control [3]. This algorithm employs a dynamic reference model which characterizes how we would like the process to behave. The difference between the output of the reference model and the output of the plant serves as a measure of performance for the overall process. We refer to this new algorithm as a fuzzy model reference learning controller (FMRLC). The term "learning" is used because its control policy is one that can change with respect to changes in the controlled process and the environment in which it is operating (and it remembers these changes). Since the FMRLC has functional similarities to Procyk and Mamdani's linguistic SOC, the FMRLC which is presented next may be classified as a learning control algorithm

for the same reasons as for the linguistic SOC algorithm.

The basic architecture for the FMRLC is presented below in Figure 27. With the

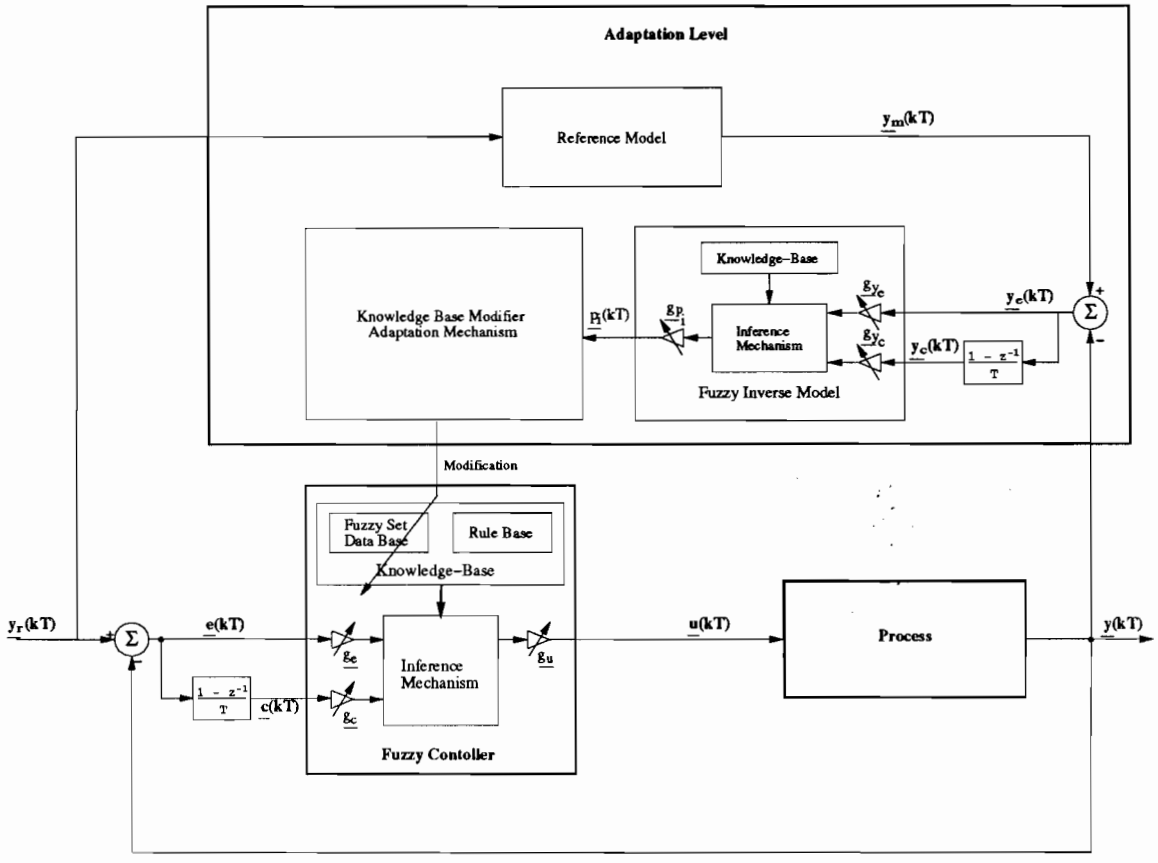


Figure 27: Basic architecture for the fuzzy model reference adaptive controller.

exception of the performance evaluation mechanism and the inverse process model, this framework is similar to that of linguistic SOC. In this framework, the performance evaluation mechanism utilizes a reference model and roughly speaking the inverse model M^{-1} is implemented by means of a fuzzy system. The knowledge base modifier is implemented by utilizing the new knowledge base modification algorithm introduced in Section 3.3 of this chapter. The fuzzy controller remains the same as

for the linguistic SOC framework. Both the performance evaluation mechanism and the inverse model will be discussed next.

3.5.1 The Reference Model

As stated earlier, the performance of the process in this framework is evaluated with respect to a reference model. This model is generally a dynamic system whose characteristics are such that they meet the desired design criteria for the controlled process. The reference model provides a capability for quantifying the desired performance of the process. In general, the reference model may be any type of dynamical system (linear or non-linear, time-invariant or time-varying, etc.). Furthermore, this reference model may be scheduled since we cannot always expect the system to obtain the same level of performance under all operating conditions. Throughout this thesis, we will consider only first or second order linear systems whose behavior is well studied and understood.

The performance of the overall system is computed with respect to the reference model by generating an error signal $\underline{y}_e(kT)$ between the reference model output $\underline{y}_m(kT)$ and the process output $\underline{y}(kT)$. As shown in Figure 27, this operation may be expressed as

$$\underline{y}_e(kT) = \underline{y}_m(kT) - \underline{y}(kT). \tag{3.45}$$

Given that the reference model meets the desired engineering design criteria such as rise time, overshoot, etc. and the input to the reference model is the controlled process reference signal $y_r(kT)$, the desired performance of the controlled process is met if the adaptation mechanism forces $\underline{y}_e(kT)$ to remain very small for all time. Just as the output of the fuzzy performance evaluator $\underline{p}_i(kT)$ represents the magnitude and direction of the desired change in the process output, $y_e(kT)$ is also a measure of

the desired change in the process output. However, this desired change in the process output is made with respect to a more specific performance objective. Therefore, $y_e(kT)$ may serve as the input to the fuzzy inverse model presented next.

3.5.2 Fuzzy Inverse Model

Recall that the inverse model for the linguistic SOC algorithm consisted of a simple constant matrix M^{-1} which approximately describes the relationship between small incremental changes in the process outputs and small changes in the process inputs. However, often for implementation of the linguistic SOC, this model is obtained from the control engineer's qualitative knowledge about the process rather than from rigorous mathematical analysis. Further, by assuming that the inverse is a constant matrix we assume that the relationship between a given process output and a given process input is monotonically increasing or decreasing which is not always true for the general control problem. In addition, the inverse model matrix quantifies the desired process input changes as a function of only the desired process output changes. In general, the desired process input changes may be a function of other parameters such as process operating conditions and process state. For many processes, given changes in the operating conditions may result in a reversal of the incremental relationship between the process input and outputs. As will be shown later, the relationship between the desired output changes and other process parameters such as the operating conditions or process states may be expressed qualitatively by linguistic *condition* $\rightarrow$ *action* rules.

Through experience or educated intuition a control engineer often has acquired some qualitative knowledge about the relationship between the desired process output changes and the necessary input changes. This knowledge may be conveniently

quantified by a fuzzy system. We propose replacing the inverse model matrix M^{-1} in the linguistic SOC algorithm with a fuzzy system to provide greater flexibility and accuracy in the overall design. This new inverse model is referred to as a "fuzzy inverse model".

The fuzzy inverse model as it is shown in Figure 27 simply maps the desired output changes $\underline{y}_e(kT)$ and possibly other parameters such as the functions of $\underline{y}_e(kT)$ and the process operating conditions to the necessary changes in the process inputs. Since there exist numerous combinations of inputs to the fuzzy inverse model, in Figure 27 only the desired output changes $\underline{y}_e(kT)$ and the change in the desired output changes $\underline{y}_c(kT)$ are shown for the sake of brevity (e.g. delayed versions and functions of the variables can be used). In this case, the change in the desired output change $\underline{y}_c(kT)$ is computed by

$$\underline{y}_c(kT) = \frac{\underline{y}_e(kT) - \underline{y}_e(kT - T)}{T}. \tag{3.46}$$

We will generally assume $\underline{y}_e(kT)$ and $\underline{y}_c(kT)$ are always employed as inputs to the fuzzy inverse model. If others are deemed necessary for a particular application, they may be incorporated as well.

For an illustrative example of a fuzzy inverse model design, consider a given single-input single-output process whose relationship between the input and output is always monotonically increasing. A typical rule base array which may be employed in a fuzzy inverse model for this process is shown in Table 9 below where Y_e^j and Y_c^k denote the fuzzy sets associated with $y_e(kT)$ and $y_c(kT)$, respectively, and $P_i^{j,k}$ denotes the fuzzy sets quantifying the desired process input change $p_i(kT)$. The reasons for using the change in the desired output change is to provide some "damping" in the adaptation mechanism. In other words, since we have information about the

Table 9: Typical rule base array table for the fuzzy inverse model.

$P_i^{j,k}$		Y_c^k										
		−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5
Y_e^j	−5	−5	−5	−5	−5	−5	−5	−4	−3	−2	−1	0
	−4	−5	−5	−5	−5	−5	−4	−3	−2	−1	0	+1
	−3	−5	−5	−5	−5	−4	−3	−2	−1	0	+1	+2
	−2	−5	−5	−5	−4	−3	−2	−1	0	+1	+2	+3
	−1	−5	−5	−4	−3	−2	−1	0	+1	+2	+3	+4
	0	−5	−4	−3	−2	−1	0	+1	+2	+3	+4	+5
	+1	−4	−3	−2	−1	0	+1	+2	+3	+4	+5	+5
	+2	−3	−2	−1	0	+1	+2	+3	+4	+5	+5	+5
	+3	−2	−1	0	+1	+2	+3	+4	+5	+5	+5	+5
	+4	−1	0	+1	+2	+3	+4	+5	+5	+5	+5	+5
	+5	0	+1	+2	+3	+4	+5	+5	+5	+5	+5	+5

rate of change of the desired output changes, we may quantify that a small value of y_e with a small change in y_e is more desirable than a small value of y_e with a large change in y_e due to the fact that overshoot is likely to occur. Thus the rate of change information provides a feature analogous to "damping" in conventional control since overshoot may be reduced.

The fuzzy inverse model rule base array shown in Table 9 was designed to take advantage of the "damping" feature described above. For example, consider the case where $y_e(kT) = 0$ and is best characterized by Y_e^j where $j = 0$. This represents the smallest possible absolute value of $y_e(kT)$. The "optimal" change in $y_e(kT)$, is $y_c(kT) = 0$ which is characterized by Y_c^k where $k = 0$. Increasing or decreasing k while maintaining j constant causes the process input $P_i^{j,k}$ to become larger. Given careful analysis of Table 9, the reader should become convinced this feature is consistent throughout the rule base array table.

Note that similar to the fuzzy controller, the fuzzy inverse model shown in Figure 27 contains normalizing scaling factors, namely $\underline{g}_{y_e}$, $\underline{g}_{y_c}$, and $\underline{g}_{p_i}$, for each universe of discourse. Selection of the normalizing gains can be very critical with respect to the overall performance of the system. However, in subsequent chapters we will present several example applications of the FMRLC which should provide a step by step design procedure in which suitable selection of the gains may be demonstrated.

Due to physical constraints for a given system, the range of values for the process inputs and outputs are generally known from a qualitative analysis of the process. As a result, we can determine the "range of values" or the universe of discourse for $\underline{y}_e(kT)$ and $\underline{p}_i(kT)$. Consequently, $\underline{g}_{y_e}$ and $\underline{g}_{p_i}$ are chosen such that the appropriate universes of discourse are mapped to $[-1, +1]$.

Due to the fact that a detailed mathematical model of the process is assumed to be unavailable, the universe of discourse for $\underline{y}_c(kT)$ generally cannot be determined in advance. Therefore, $\underline{g}_{y_c}$ is left as a tuning parameter for this algorithm. Recall, that the scaling factor $\underline{g}_{y_c}$ associated with the change in the desired output changes has the effect of providing "damping" to the controller modifications. Moreover, the "damping" effect is increased as the elements of the scaling factor $\underline{g}_{y_c}$ are increased. A suitable selection of $\underline{g}_{y_c}$ may be obtained by monitoring the response of the overall process with respect to the reference model response. If undesirable oscillations exist between a given process and the associated reference model output response, it is likely that the element of $\underline{g}_{y_c}$ associated with this output is too small and should be increased. Likewise, if given element $\underline{g}_{y_c}$ is too large, the process will be unable to keep up with the reference model due to the resulting damping.

Below a "simple" algorithm is presented for selecting the fuzzy inverse model gains.

1. Select the controller gains $\underline{g}_{y_e}$ associated with the desired output change $\underline{y}_e(kT)$ such that each universe of discourse is mapped to the interval $[-1, 1]$.

2. Choose the controller gain $\underline{g}_{p_i}$ to be the same as for the fuzzy controller output gain $\underline{g}_y$. This will allow $\underline{p}_i(kT)$ to take on values as large as the largest possible inputs.

3. Assign the numerical value 0 to the scaling factor associated with the change in the desired output changes (i.e. all elements of $\underline{g}_{y_c}$ are set equal to 0).

4. Apply a step input to the process which is of a magnitude that may be typical for the process during normal operation. Observe the process response and the reference model response.

5. Three cases:

   (a) If there exist unacceptable oscillations in a given process output response about the reference model response, then increase the associate element of $\underline{g}_{y_c}$. Go to step 4.

   (b) If a given process output response is unable to "keep up" with the reference model response, then decrease the associated element of $\underline{g}_{y_c}$. Go to step 4.

   (c) If the process response is acceptable with respect to the reference model response, then the controller design is completed.

For the applications studied in this thesis, the above gain selection methodology has proven very successful. However, given that the algorithm is a result of practical experience with the control algorithm rather than strict mathematical analysis, it is likely that it will not work for all processes. For some applications (none of the ones studied here), the algorithm may always result in an unstable process. In such situations, it may be necessary to modify other controller parameters such as the controller sampling period T or the number of fuzzy controller rules. Clearly, the stability analysis of the FMRLC is an important research direction.

3.6 Chapter Summary

The primary objectives of this chapter were to present the linguistic SOC of Procyk and Mamdani [59] and discuss possible limitations of the algorithm, introduce the new FMRLC algorithm which resolves many of the design issues in the linguistic SOC, and discriminate the characteristics of a learning control algorithm from those of an adaptive algorithm.

Throughout the presentation of the FMRLC algorithm in this chapter, we have mentioned the possible advantages as they have became apparent. However, it is important that these advantages be summarized in order to provide the philosophy behind the use of FMRLC.

Since fuzzy set theory provides the underlying mechanism for implementation, many of the typical advantages of fuzzy control can be found here also. These may be summarized as follows:

- A detailed mathematical model of the process is not needed to design the controller,
- Only qualitative information about the process is needed, as a result time and money is saved in the modeling phase of design,
- Provides a good technique for non-linear, ill-defined, and/or time-varying processes where other techniques have failed.

Further, since a fuzzy system is employed as the lower-level controller, this generally implies that few rules are required to obtain a smooth mapping between the control signals and the process inputs. As a result of fewer control rules in the FMRLC algorithm, the converges of the algorithm is often very fast.

Since learning capabilities are introduced additional features are provided. For example, the FMRLC control algorithm has an autonomous ability to identify an "optimal" mapping between control signals and process inputs such that the system behaves like the reference model. Most importantly, this mapping is continually updated even when the behavior of the process changes in time. Therefore, FMRLC provides a technique to design a controller when the *a priori* information is so limited that it is impossible/impractical to design a controller in advance which meets the desired design criteria for all operating condition and all possible changes in the process behavior. A final advantage of FMRLC results from the fact that it is a learning control algorithm. Farrell and Baker [24] claim that often a learning control algorithm will have a faster convergence than an adaptive control algorithm due to the enhanced memory capabilities of the learning control algorithm.

CHAPTER IV

Applications of Fuzzy Model Reference Learning Control

In this chapter we illustrate the practical application of the fuzzy model reference learning controller (FMRLC) algorithm, discussed in the previous chapter, by simulation for the following applications:

- Cart and inverted pendulum,
- Rocket velocity controller,
- Two-degree of freedom manipulator position control.

The purpose of these simulations is to investigate the effectiveness of FMRLC control for unstable, time-varying, and multiple input and output, non-linear processes. Further, through these examples, we hope to illustrate a "step by step" procedure for the design of the FMRLC algorithm. Also in this chapter, the linguistic SOC of Procyk and Mamdani will be design and simulated for the cart and pendulum for comparison with the results obtained from FMRLC control of the same process. This example is offered to illustrate the advantages of the FMRLC algorithm over the linguistic SOC.

Since one major objective of this section is to illustrate design methodology, many of the design issues which were presented in Section 3.3 and 3.5 will be re-

iterated here for the cart and pendulum system. However, after this application, it is hoped that all design issues will be resolved. Therefore, for all subsequent applications only the pertinent details will be included.

4.1 Cart and Pendulum System - Problem Statement

Figure 28 below illustrate the cart and pendulum system. It consist of a cart on which rigid pole (inverted pendulum) is mounted on a hinged ball bearing having no friction. The cart is free to move on wheels and the pole has one degree of freedom about the hinge point.

A mathematical model for the cart and inverted pendulum was developed by Cannon [11] and is expressed by the following non-linear differential equations:

$$\ddot{\theta} = \frac{g \sin\theta + \cos\theta \left[\frac{-f - m_p l \dot{\theta}^2 \sin\theta}{m_p + m_c}\right]}{l\left[\frac{3}{4} - \frac{m_p \cos^2\theta}{m_p + m_c}\right]}, \tag{4.1}$$

$$\ddot{x} = \frac{f + m_p l[\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta]}{m_p + m_c} \tag{4.2}$$

where θ is the angle of the pole in radians with respect to the vertical position, x is the horizontal position of the cart in meters, f is the driving force applied to the cart in Newtons, and for purpose of our simulations the cart and pendulum parameters are given below:

1. $g = 9.8 \ \frac{meters}{s^2}$ - the acceleration due to gravity,
2. $m_c = 1.0 \ kg$ - the mass of the cart,
3. $m_p = 0.5 \ kg$ - the mass of the pole,
4. $l = 0.5 \ meters$ - the half pole length.

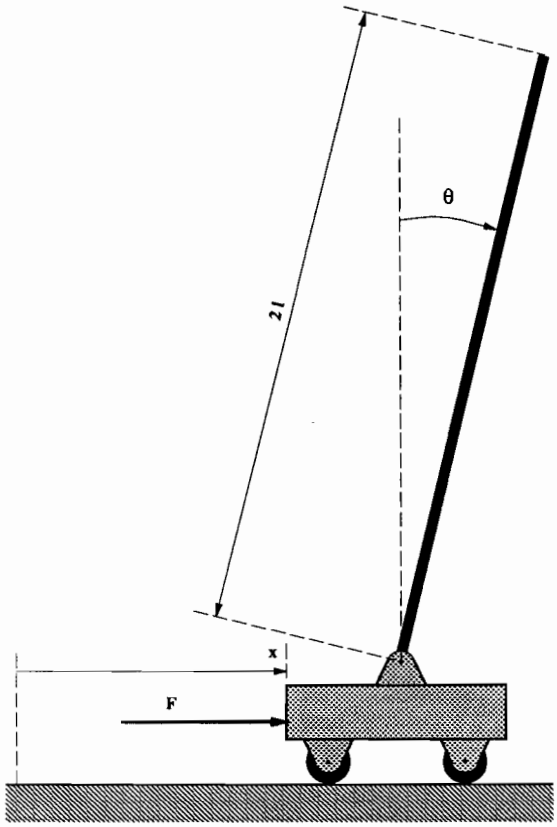


Figure 28: A simple graphical representation of the cart and pendulum.

The cart and pendulum is a very interesting and often studied classical control problem. At a first glance, this control may appear to be very academic. However, several real world analogies of this control problem do exist. For instance, a rocket or missile in vertical flight is essentially an inverted pendulum with more degrees of freedom at the hinge than considered here.

One major reason for the interest in this system is due to the fact that the system model is non-linear and has two equilibrium points. The first equilibrium point occurs when $\theta = \pi$ radians which is stable since the pendulum will always tend

to move toward this point if no forces are exerted upon the system (provided $\theta \neq 0$ initially and there is friction in the system). Due to possible physical constraints which may result from the cart blocking the pendulum movement, the pendulum should never reach $\theta = \pi$ radians. However, in this chapter, we will assume that the pendulum is free to take on all values on the interval $[-\pi, \pi]$.

The second and more interesting equilibrium point occurs when the pendulum is completely inverted, or $\theta = 0$ radians. This is an unstable equilibrium point since the system will always tend to move away from this condition if it is perturbed. Therefore, the real challenge with this system is to design a controller which will apply forces to the cart such that the pendulum is stabilized in an inverted position even when there are disturbances to the system.

4.1.1 Related Research

Over recent years, several examples of control algorithms for the cart and pendulum system have appeared in the control literature. For example, Friedland in [26] presents several classical control design involving full state feedback. For classical full state feedback design, generally a linear process model is needed. Friedland obtains a linear process model by linearizing the non-linear model in Equation (4.1) about the equilibrium point $\theta = 0$ radians. Three cases of implementation for full state feedback control of this process are considered. These include:

1. All states and outputs can be measured for feedback,
2. Only the controlled output is measured - all states are obtained by a full order observer,
3. The output and some states can be measured - all other states are obtained by a reduced order observer.

Another example of a controller design for the cart and pendulum system is presented by Berenji, Chen and Yager in [8]. In this article a direct fuzzy controller was employed to control the cart position and the pendulum position. This work clearly illustrates that a fuzzy controller is capable of stabilizing the vertical position of the pendulum. However, the primary intent of this work was to introduce a new fuzzy set operation which was referred to by the author as an "ordered-weighted-averaging (OWA) operation. This new OWA operation has the property of lying between the *and* operation and the *or* operation traditionally defined for fuzzy sets.

Michie and Chambers in [54] present a learning control algorithm for the cart and pendulum system. In this research, the inputs to the controller are the cart position x, the cart velocity $\dot{x}$, the pole position θ, and the poles angular velocity $\dot{\theta}$. which are quantized into ordinary sets, with five discrete levels or sets for x and θ and three discrete levels or sets for $\dot{x}$ and $\dot{\theta}$. As a result, the cartesian product which relates these sets may be described by $5 \times 5 \times 3 \times 3 = 225$ discrete states. Further, the control input f is quantized into two level: "+" and "-", or "left" and "right". As a result, the overall control law is one of "bang-bang" control. The entire control law may be represented by a table with 225 entries. This "control law table" is very similar to the rule base array table which is typically employed in a fuzzy controller designs. As a result, the table actually denotes control actions which may be expressed linguistically. For example, a typical control rule my be expressed as:

If cart is far right **and** cart is hardly moving **and** pendulum is hardly leaning **and** pendulum is swinging to the left **Then** apply force left.

Learning in this framework is achieved by a simple scoring method which was developed for mathematical programming of games such as checkers or chess. Scoring

is based on how long the pendulum stayed vertical and the number of times that the pendulum was in a particular discrete state. Consequently, the scoring denotes a measure of the system performance which is used as a means for performance feedback.

The learning control research of Michie and chambers is related to the FMRLC presented here. However, several key improvements are made such as improved performance feedback due to the introduction to the reference processes model. Also, since fuzzy system is employed the mapping between controller inputs and output is smoother.

4.1.2 Linguistic SOC Design

Here we present a brief description of a linguistic SOC controller which was designed for the cart and pendulum system. This design was obtained by employing the design methodology presented by Procyk and Mamdani with our improvements to their rule-base modification technique. The resulting system and controller is simulated for comparisons with the FMRLC algorithm.

For the linguistic SOC framework, the components which need to be specified by the control designer include:

- The fuzzy controller,
- The fuzzy performance evaluator, and
- The inverse process model matrix.

Once the design for these above components is determined, the knowledge base modifier is completely specified by the design procedure outlined in Section 3.3. In the following discussion, we will present the design of each of the above components

individually. For this presentation the reader may want to refer to Figure 26 in the previous chapter for more details.

Fuzzy Controller Design

As previously mentioned in Section 3.3, the input to the fuzzy controller in the linguistic SOC framework are the output error $e(kT)$ and change in the error $c(kT)$. For the cart and pendulum problem these inputs may be expressed as:

$$e(kT) = \theta_r(kT) - \theta(kT), \tag{4.3}$$

$$c(kT) = \frac{e(kT) - e(kT - T)}{T}, \tag{4.4}$$

where $\theta(kT)$ is the sampled measurement of pole position at time $t = kT$ and $\theta_r(kT)$ is the desired pole position at time $t = kT$.

In general, the sample period T should be chosen such that it is significantly faster than plant dynamics and all time-varying system disturbances or model variations. As a rule of thumb, T should be defined to be a least twenty - five times faster than the fastest system "rise-time", of course, this assumes that some approximation of rise time is available for this system in the first place. Given a step input change, the fastest rise time is loosely defined to be the fastest possible time for a system to move 90% of the distance from one steady state condition to a new steady state condition. For this design the sampling period T was somewhat arbitrarily chosen as 5 milliseconds.

For each fuzzy controller input we defined 11 fuzzy sets whose membership functions are "triangular shaped" functions defined such that they are convex, symmetric, and normal. Further, the membership functions associated with the fuzzy controller inputs are assumed to be evenly distributed along the appropriate universes of dis-

course such that the following properties hold:

$$\sum_k \mu_{E^k}(e) = 1 \qquad \forall\, e \in \mathcal{E} \tag{4.5}$$

$$\sum_k \mu_{C^k}(c) = 1 \qquad \forall\, c \in \mathcal{C}, \tag{4.6}$$

where E^k is the k^{th} fuzzy set associated with the normalized error universe of discourse and C^k is the k^{th} fuzzy set associated with the normalized change in error universe of discourse. Figure 29 below shows a typical set of membership functions for the input fuzzy sets defined on a normalized universe of discourse whose elements are denoted by the generic argument x.

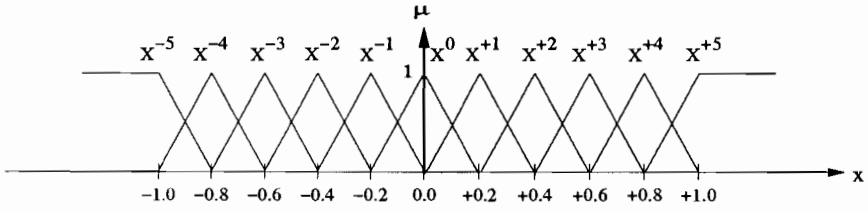


Figure 29: Example normalized input universe of discourse

Recall that the universes of discourse for each input and output are generally normalized by means of the scaling factors g_e and g_c which are associated with the error and change in error, respectively. If we assume that the numeric value of the pole position and the desired pole position is restricted to the interval $[-\pi, \pi]$, then the pole position error $e(kT)$ is restricted to the interval $[-2\pi, 2\pi]$. As a result, the fuzzy controller gain g_e was chosen to be $g_e = \frac{1}{2\pi}$.

In general, it is difficult to determine the range of the numeric values for the change in error given *a priori* information about the process. Particularly, if a detailed mathematical model of the process is unavailable. However, by exciting the

system with at variety of inputs, a suitable choice of the controller gain g_c may be obtained from these empirical results. After performing this procedure for the cart and pendulum, we found that the change in the error approximately remained within the interval $[50, -50]$. As a result, the controller gain g_c was chosen to be $g_c = \frac{1}{50}$.

For the cart and pendulum system, the fuzzy controller output is the force that is applied to the cart, denoted $f(kT)$. Above, we have clearly defined the fuzzy sets which characterize all fuzzy controller inputs. However, since this linguistic SOC design is an adaptive control algorithm, the fuzzy sets associated with the controller output are left as "learning parameters" and do not need to be specified initially.

Recall that if we assume that the fuzzy sets associated with the output have membership functions which are normal, symmetric, and convex and of the same shape, then the knowledge base may be concisely represented by a knowledge base array table. Therefore, Table 10 represents the initial knowledge base array table used for this fuzzy controller design. Further recall that the entries in the knowledge base array table simply represent the "center value" of the membership functions for the normalized fuzzy controller output fuzzy sets.

Since the linguistic SOC is a learning control algorithm, the entries in the knowledge base array table may be almost any arbitrary values on the interval $[-1.0, 1.0]$. Therefore, in Table 10, we chose all entries to be zero. This choice reflects the fact that no information is known about controlling the process so that the given performance criteria is meet. Also, as a result of entering all zeros in the initial knowledge base array table, the initial output of the fuzzy controller will always be zero.

For this fuzzy controller design, we will assume that all membership functions defined on the normalized output universe of discourse are "triangular shaped" with a base width of 0.4. Therefore, if after several knowledge base modifications, the

Table 10: Typical knowledge base array table.

$F^{j,k}$		C^k										
		-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
E^j	-5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	-4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	-3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	-2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	-1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$+1$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$+2$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$+3$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$+4$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$+5$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

entry in the knowledge base array table associated with E^j and C^k is 0.2, then the fuzzy set which is characterized by this entry is shown below in Figure 30. Moreover,

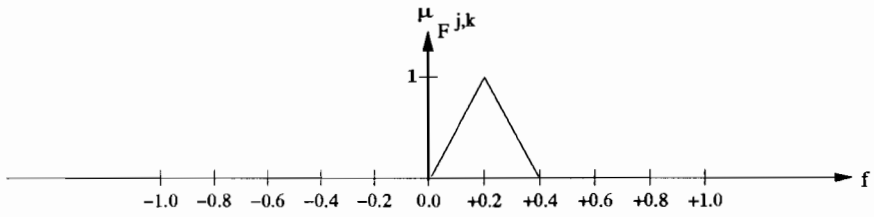


Figure 30: Example of normalized fuzzy controller output membership function.

if we assume that the force applied to cart is constrained to the interval $[-200, 200]$, then the normalizing controller output gain g_f associated with the force $f(kT)$ is selected to be $g_f = 200$.

Fuzzy Performance Evaluator Design

Recall from Section 3.3, that with the exception of the knowledge base, the fuzzy performance evaluator design is very similar to the fuzzy controller design. In particular, all fuzzy sets defined for the inputs of the fuzzy controller are also employed in the fuzzy performance evaluator design. Also, the normalizing gains g_e and g_c associated with the error and change in error, respectively are chosen to be the same as for the fuzzy controller. Furthermore, all fuzzy set membership functions which characterize these inputs are chosen to be the same as those employed for the fuzzy controller. The normalizing gain for the output $p_o(kT)$ of the fuzzy performance evaluator was chosen to the same as the normalizing gain g_y associated with the fuzzy controller output.

Since the primary function of the fuzzy performance evaluator is to monitor performance rather than performing control operations, the knowledge base is chosen differently. Recall from Section 3.3 that the rule base array table shown in Figure 6 represents an "optimal" fuzzy performance evaluator. However, for our implementation we transformed this rule base into the knowledge base array shown in Table 11 below.

Inverse process Model

In Section 3.3, we found that when the relationship between the inputs and outputs is assumed monotonic the inverse process model may be approximated by a constant $s \times r$ matrix where r is the number of process inputs and s is the number of outputs. Therefore, since the cart and pendulum is a single-input, single-output process the inverse process matrix M^{-1} is a scalar.

Table 11: Fuzzy performance evaluator knowledge base array table.

$P_i^{j,k}$		C^k										
		-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
E^j	-5	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0
	-4	-1.0	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$
	-3	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$
	-2	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$
	-1	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$
	0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$
	$+1$	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$
	$+2$	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$
	$+3$	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$	$+1.0$
	$+4$	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$	$+1.0$	$+1.0$
	$+5$	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$	$+1.0$	$+1.0$	$+1.0$

The sign of this scalar inverse model quantity is determined by qualitative analysis of the controlled process. For example, consider both the hanging pendulum position and the inverted pendulum position shown in Figure 31 below. Notice when the pendulum is inverted a "positive" force applied to the cart is likely to force the pendulum in the "negative" direction, assuming of course that the force is large enough to overcome gravitational forces. This implies that the relationship between the input and the output is "monotonically negative". As a result, M^{-1} should be chosen as a "negative" scalar. However, when the pendulum is in the hanging position, a "positive" force result in a positive change in the pendulum position which implies that M^{-1} should be a "positive" scalar. Thus, the incremental relationship between the process input and output is non-monotonic over the entire state space as is normally assumed for the linguistic SOC framework. In fact, there exist two regions of operation for this process, namely $|\theta| \leq \frac{\pi}{2}$ and $|\theta| > \frac{\pi}{2}$. However, since

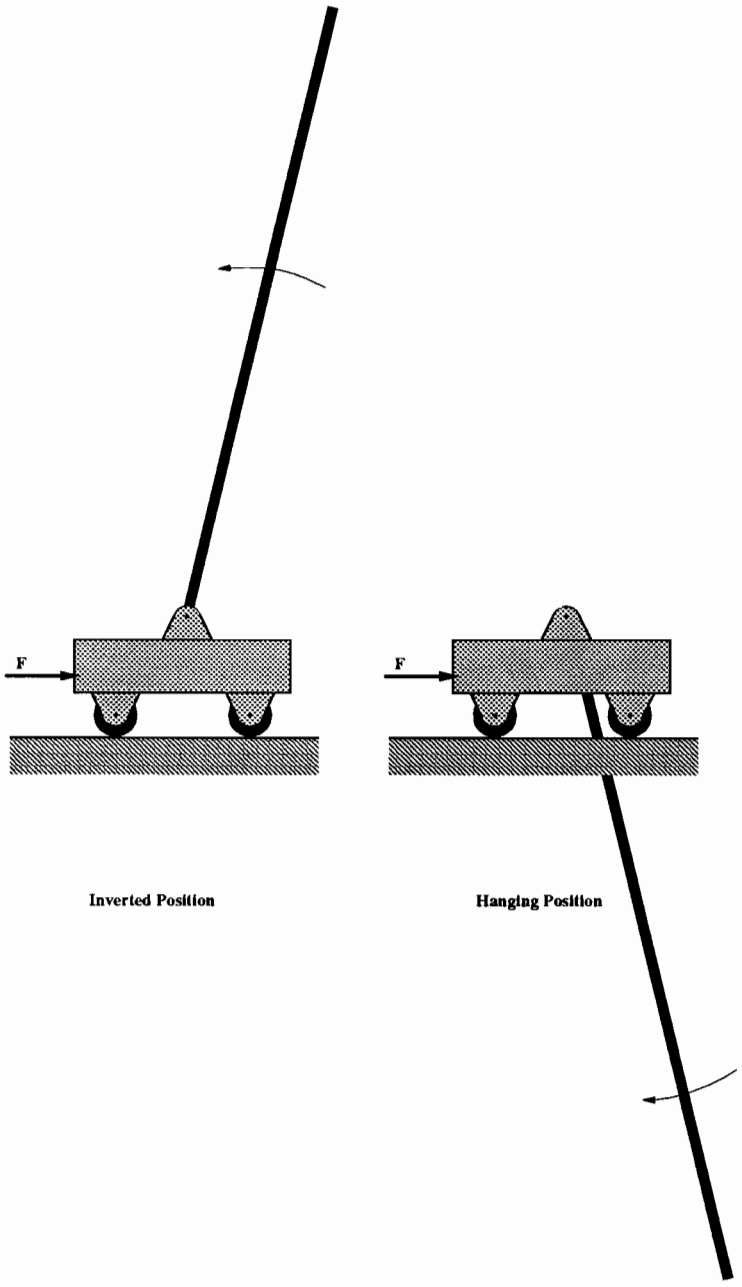


Figure 31: Inverted and hanging pendulum positions.

we are primarily concerned with the inverted positions, we will choose M^{-1} to be a negative scalar quantity and accept the fact that this inverse process model is not valid when $|\theta| > \frac{\pi}{2}$.

At this point in the design process the only remaining design parameter is the magnitude of elements of the inverse process model matrix. However, since all other components of the linguistic SOC have been designed, the magnitude of the elements of inverse process model matrix are left as a tuning parameters. However, in order to eliminate the possibility of $p_i(kT)$ becoming larger than the largest possible process input, the scalar inverse process model for SISO should be normalized to interval $[-1.0, 1.0]$. This may be explained by recognizing the fact that the output for the fuzzy performance evaluator is constrained to the same interval as the fuzzy controllers output due to the equal normalizing gains chosen for both fuzzy systems.

Often for the linguistic SOC algorithm, the control designer will arbitrary choose the initial magnitude of the elements contained in the inverse process model matrix. Then by iteratively adjusting these elements after successive experiments a desirable response may be obtained. Consequently, the entire process can be very *ad hoc*. By performing this procedure for the cart and pendulum system, -0.9 was found to be a suitable value for the inverse process model matrix.

Simulation Results

The linguistic SOC presented above was implemented and simulated using the FORTRAN programming language. See Appendix A.2 for the program listing. The results of this simulation are shown below in Figure 32. In order to make the problem more challenging, the desired pole position was allowed to take on values $\{-25, 0, 25\}$ degrees rather than controlling only for the completely vertical position. Conse-

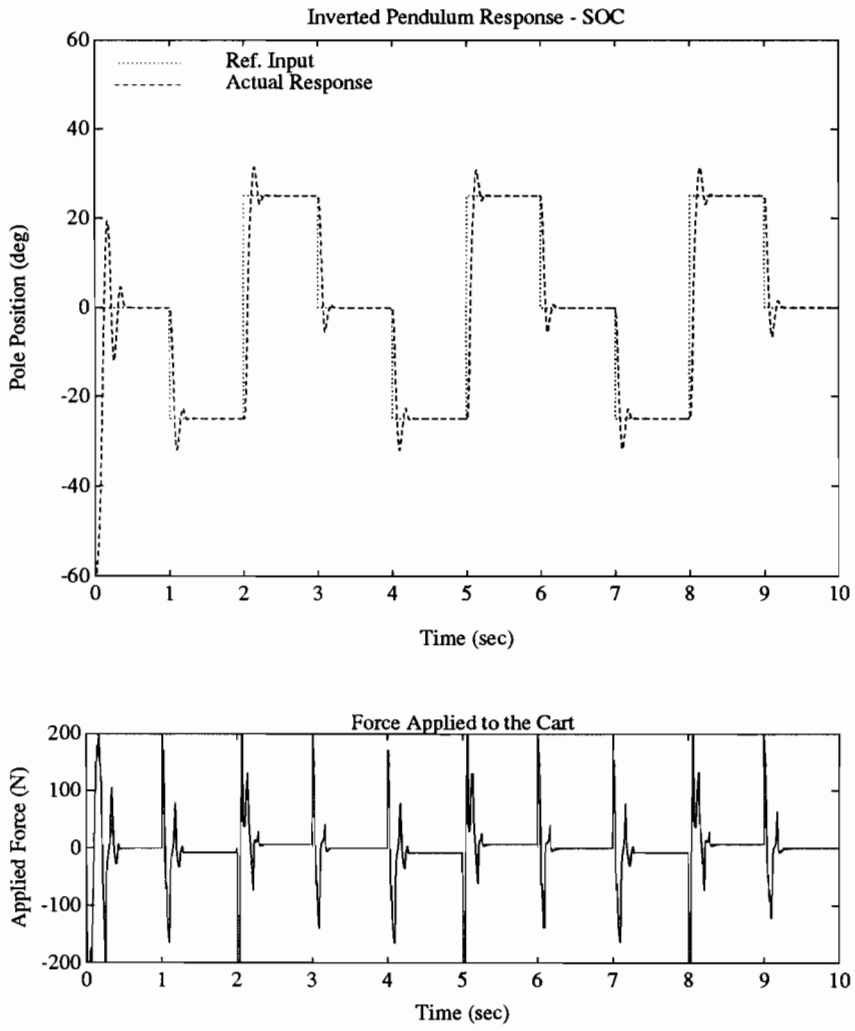


Figure 32: Simulation results for linguistic SOC control of the cart and pendulum system.

quently, for non-vertical positions the controller must keep the cart moving so that the momentum will keep the pole from falling.

Notice in Figure 32 that "good" steady state performance and a fast rise time were achieved using the linguistic SOC design. Furthermore, as we would expect from a learning control system, the performance of the system got better as time progressed. More surprisingly, the convergence of the algorithm was very fast. In fact, from this simulation it appear that good performance was obtained after only one step input change.

In general, from Figure 32 we observe that the linguistic SOC has been very successful in stabilizing the pendulum in the inverted positions. However, several negative features of the linguistic SOC design may also be immediately recognized in Figure 32. For example, there exists some overshoot and oscillations in the transient response of the system even after the algorithm appears to have "converged". Moreover, as we previously discussed, the linguistic SOC design provides very little control over the final process performance. On the other hand, the FMRLC approach presented next provides an explicit characterization of the desired performance via the reference model.

The next experiment with this system was designed to study the effects of inaccuracies in the inverse process model. For example, we previously stated that the inverse process model was only valid when the magnitude of the pole position is less than $\frac{\pi}{2}$ radians. Therefore we would not expect the learning mechanism to be capable of recovering from modeling inaccuracies when $|\theta|$ is greater than $\frac{\pi}{2}$. This fact is clearly verified in the simulation results shown in Figure 33 below. For this simulation, the system is initialized with the pendulum in the hanging position. Therefore, the object of the experiment was to bring pendulum from the hanging

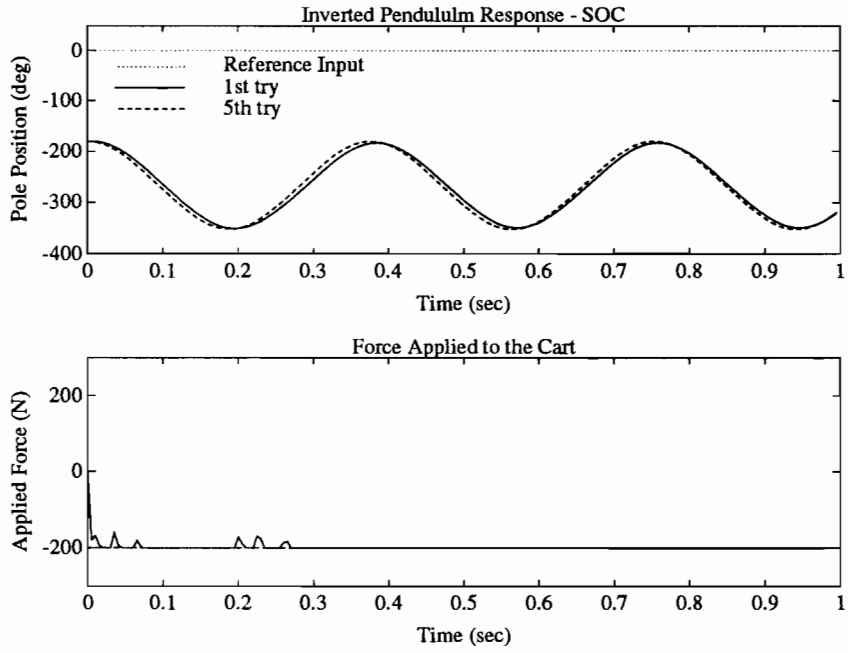


Figure 33: Simulation results for linguistic SOC control of the cart and pendulum system when initiated with pendulum hanging vertically downward.

vertical position, $\theta = -\pi$ radians, to the inverted vertical position, $\theta = 0$ radians. Five successive runs of the system were executed such that the knowledge base from each of the previous runs was preserved. As a result of inaccuracies in the inverse model matrix the pendulum was forced in the wrong direction.

Given that we had prior knowledge of the inverse model inaccuracies, the "poor" results shown in Figure 33 would be expected and therefore trivial. However, the intention of performing this experiment is to illustrate one possible advantage of using a "fuzzy inverse model" which will be discussed in the next subsection.

4.1.3 FMRLC Design

In this subsection we present a description of a FMRLC controller which was designed for the cart and pendulum system. The resulting system and controller is simulated using the FORTRAN programming language. See Appendix A.3 for the program listing.

For this design, the fuzzy controller is assumed to be the same as that designed for the linguistic SOC design. We need to specify the design for the reference model and the fuzzy inverse model which were presented in Section 3.5. Therefore in this subsection we present a detailed description of each of these component. The reader may want to refer back to Figure 27 in the previous chapter for more details.

Reference Model Design

As previously mentioned, the reference model is a dynamic system whose behavior characterizes the desired design specifications for the process. In general, this reference model may be any dynamical system of any order. However, often we will consider only first or second order linear processes. Furthermore, given the physical constraints which may exist for the given controlled processes, the dynamics of this reference model system should be reasonably chosen such that the controlled process is capable of "keeping up" with the model reference system.

Given the design factors described above, we somewhat arbitrarily chose a reference model which is described by the following differential equation:

$$\frac{d\ y_m(t)}{dt} + 10y_m(t) = 10y_r(t), \tag{4.7}$$

where $y_m(t)$ denotes the reference model output and $y_r(t)$ denotes the desired process output which is equal to $\theta_r(t)$ for the cart and pendulum system. By employing the

Laplace operator s, Equation (4.7) may be conveniently expressed by the following Laplace transfer function:

$$\frac{Y_m(s)}{Y_r(s)} = H_m(s) = \frac{10}{s+10} \tag{4.8}$$

where $Y_m(s)$ and $Y_r(s)$ denote the Laplace transform of the time domain signal $y_m(t)$ and $y_r(t)$, respectively.

Fuzzy Inverse Model Design

Recall that the inverse process model in the FMRLC is a fuzzy system whose knowledge base simply quantifies the qualitative knowledge that a controller designer may have about the process behavior. Further recall that this qualitative knowledge may have been obtained empirically or from intuitive analysis of the process. Therefore, we will present a fuzzy inverse model design based on intuitive analysis of the cart and pendulum system.

Recall from earlier analysis that the cart and pendulum system has two regions of operation, namely $|\theta| \leq \frac{\pi}{2}$ and $|\theta| > \frac{\pi}{2}$, such that the system dynamics are different in each region. Therefore, these regions of operation define two ordinary sets which characterize the magnitude of the pole position with respect to the two operation regions. We may view these two ordinary sets as fuzzy sets with membership functions shown in Figure 34 below.

In order to incorporate the two operation regions into the fuzzy inverse model, the magnitude of the pole position will serve as an additional input to the fuzzy inverse model. Consequently, the fuzzy inverse model has three inputs which may be described as:

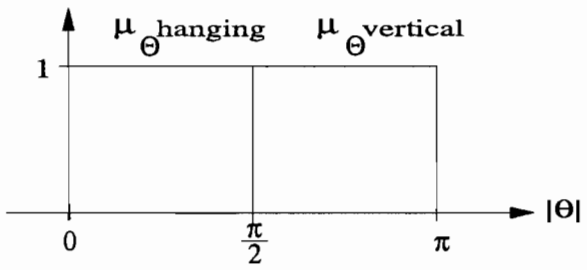


Figure 34: Membership functions for $|\theta|$ which quantify regions of operation for the cart and pendulum system.

- the desired process output change $y_e(kT)$,
- the change in the desired output change $y_c(kT)$,
- the magnitude of the pole position $|\theta(kT)|$.

where the desired process output change and the change in the desired output change is expressed by

$$y_e(kT) = y_m(kT) - \theta(kT), \tag{4.9}$$

$$y_c(kT) = \frac{y_e(kT) + y_e(kT - T)}{T}, \tag{4.10}$$

respectively.

For both $y_e(kT)$ and $y_c(kT)$, we quantize the respective universes of discourse into 11 fuzzy sets or levels. These fuzzy sets are defined using the same constraints as for the inputs to the fuzzy controller. Namely, the fuzzy set membership functions are "triangular shaped" and are defined such that they are symmetric, convex, and normal. Moreover, the membership function for the fuzzy inverse model inputs are assumed to be evenly distributed along the respective universes of discourse such

that the following properties hold:

$$\sum_k \mu_{Y_e^k}(y_e) = 1 \qquad \forall\, y_e \in \mathcal{Y}_e \tag{4.11}$$

$$\sum_k \mu_{Y_c^k}(y_c) = 1 \qquad \forall\, y_c \in \mathcal{Y}_c, \tag{4.12}$$

where Y_e^k is the k^{th} fuzzy set associated with the normalized desired process output change $y_e(kT)$ and Y_c^k is the k^{th} fuzzy set associated with the normalized change in the desired process output change $y_c(kT)$.

The relationship between the force applied to the cart and the position of the pendulum is "negative" when the pendulum is inverted. Moreover, this relationship is reversed or "positive" when the pole is in a hanging position. Therefore, using this qualitative information we may generate the knowledge base array tables shown in Table 12 for the case where $|\theta| \leq \frac{\pi}{2}$ and Table 13 for the case where $|\theta| > \frac{\pi}{2}$. The output of this fuzzy inverse model represents the necessary changes in the fuzzy controller output for the previously applied controller inputs.

The fuzzy inverse model is typically implemented with normalized universes of discourse for both inputs and outputs. Therefore, we need to determine the scaling factors g_{y_e}, g_{y_c}, and g_{p_i} associated with the desired change in the process output $y_e(kT)$, the change in the desired change in the process output $y_c(kT)$, and the desired change in the controller input $p_i(kT)$, respectively. Appropriate selection of these scaling factors is obtained by employing the gain selection algorithm presented in Section 3.5. This 5 step process is given below for the cart and pendulum system. It is suggested that the reader refer to Section 3.5 for an explanation of these steps.

Table 12: Knowledge base array table employed for the fuzzy inverse model when $|\theta| \leq \frac{\pi}{2}$ in the cart and pendulum system.

$P_i^{j,k}$		Y_c^k										
		-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
Y_e^j	-5	+1.0	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0
	-4	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2
	-3	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4
	-2	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6
	-1	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8
	0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0
	$+1$	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0
	$+2$	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0
	$+3$	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0
	$+4$	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0
	$+5$	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0

Table 13: Knowledge base array table employed for the fuzzy inverse model when $|\theta| > \frac{\pi}{2}$ in the cart and pendulum system.

$P_i^{j,k}$		Y_c^k										
		-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
Y_e^j	-5	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0
	-4	−1.0	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2
	-3	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4
	-2	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6
	-1	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8
	0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0
	$+1$	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0
	$+2$	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0
	$+3$	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0
	$+4$	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0	+1.0
	$+5$	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0	+1.0	+1.0

Step #1 For the cart and pendulum problem we assumed that the pole position $\theta(t)$ was constrained to the interval $[-\pi, \pi]$ radians. If we assume that the same constraint exists for the output of the reference model $y_m(t)$, then it is easily shown that $y_e(t)$ in constrained to the interval $[-2\pi, 2\pi]$. Therefore, a suitable selection for this controller gain is $g_{y_e} = \frac{1}{2\pi}$.

Step #2 For the cart and pendulum problem we choose $g_{p_i} = g_f = 200$.

Step #3 For the cart and pendulum problem we choose $g_{y_c} = 0$.

Step #4 - Step #5 After iteratively applying steps 4 and 5 for the cart and pendulum problem we found that $g_{y_c} = 0.1$ was a suitable selection.

Simulation Results

The FMRLC design for the cart and pendulum was simulated using the FORTRAN programming language. See Appendix A.3 for the program listing. The results of this simulation are shown below in Figure 35. As was performed for the linguistic SOC simulations, the problem was made more challenging by allowing the desired pole position to take on values $\{-25, 0, 25\}$ degrees rather than controlling only for the completely vertical position.

Notice in Figure 35 that "good" tracking with respect to the reference model was achieved from the FMRLC design. Consequently, the system exhibits "good" transient and steady state performance. Furthermore, the fast rate of convergence which was observed in the linguistic SOC algorithm results has been retained in this algorithm. Nearly perfect tracking was obtained after approximately three step input changes.

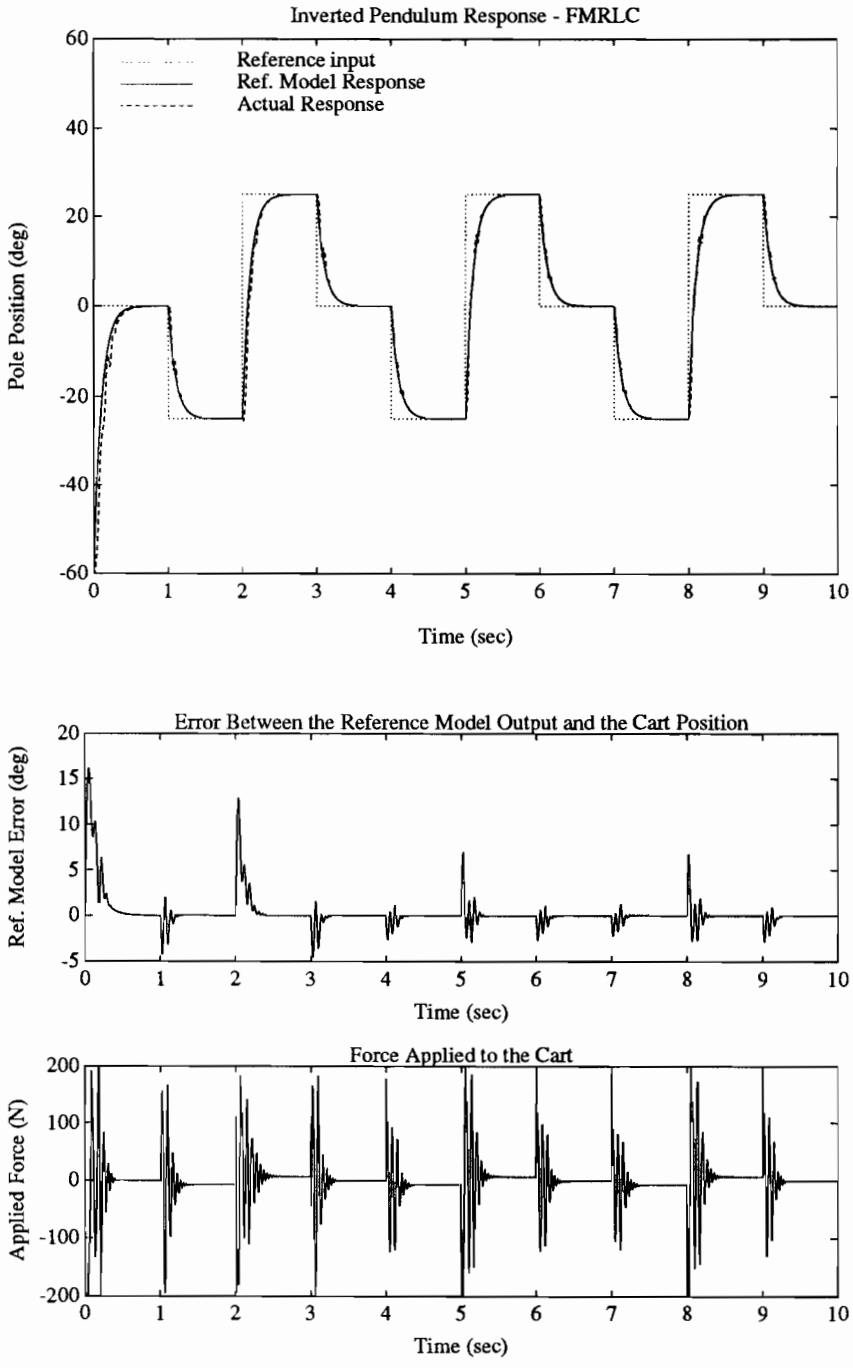


Figure 35: Simulation results for FMRLC of the cart and pendulum system.

The next experiment with this system was designed to show that the fuzzy inverse model is more accurate than the inverse process model matrix which was employed in the linguistic SOC algorithm. In Figure 36 below, we show the results of the FMRLC algorithm when the pendulum is initiated in the hanging vertical position $\theta = -\pi$. This experiment is similar to the result shown in Figure 33 for the

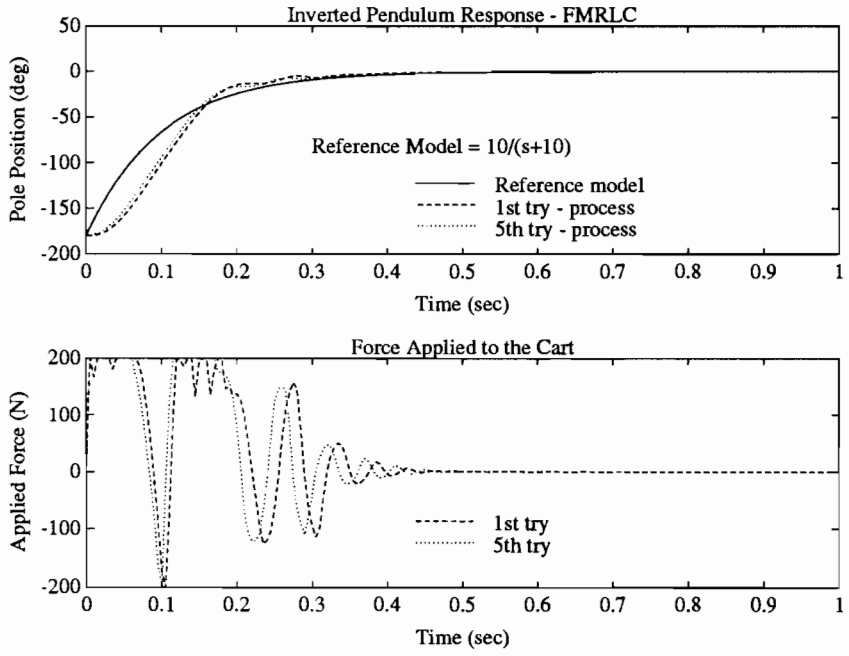


Figure 36: Simulation results for FMRLC control of the cart and pendulum system when initiated with the pendulum hanging vertically downward.

linguistic SOC algorithm. Once again, the object of this experiment was to bring pendulum from the the hanging vertical position, $\theta = -\pi$ radians, to the inverted vertical position, $\theta = 0$ radians. Five successive runs of the system were executed such that the knowledge base was preserved from each of the previous runs.

Recall from Figure 33 that the linguistic SOC design was unable to bring the pendulum from the hanging position to the vertical position due to inaccuracies in

the inverse process model. However, by using the fuzzy inverse model, we are better able to quantify the relationship between process inputs and outputs. Consequently, the simulation results for the FMRLC algorithm are much better than those obtained from the linguistic SOC design. In fact the FMRLC algorithm was completely successful in controlling the process. Also note that the tracking got marginally better with each successive try, however, perfect tracking with the reference model was never achieved. This inaccuracy may have resulted from the fact that gravitational effects cause the pendulum response to be much slower when the pendulum is in the hanging position. As a result, the pendulum is unable to "keep up" with the reference model even when the maximum force is applied to the cart. This implies that two reference models may be needed (one for each region of operation) in order to provide a reasonable performance expectations from the process for all operating conditions.

The simulation results shown in Figure 37 illustrate the effect of changing the controller gain g_{y_c}. Recall from earlier discussion that by employing $y_c(kT)$ as an input to the fuzzy inverse model we are able to provide a "damping" effect into the adaptation process. This "damping" effect is clearly illustrated in Figure 37 which shows the response of the system for g_{y_c} equal to 0.01, 0.1, and 1.0, respectively. In Figure 37(a) for the case where $g_{y_c} = 0.01$, observe that the pole position response exhibits some oscillatory behavior about the reference model response. This oscillatory behavior is particularly large early in the time response. However, after some time has passed and the learning mechanism has acquired enough knowledge about controlling the process it is able to greatly reduce these oscillations. Compare this response with that shown in Figure 37(b) for $g_{y_c} = 0.1$. Notice after "turning up" the scaling factor g_{y_c} somewhat, we find that the oscillations are nearly eliminated

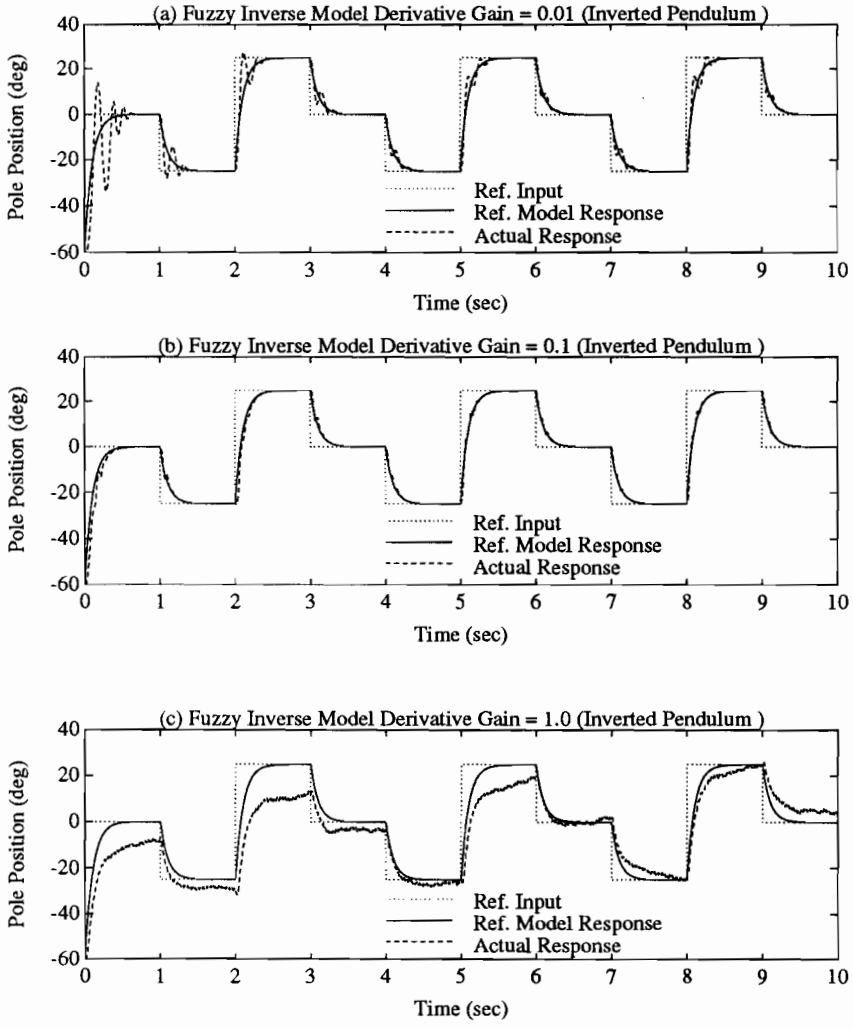


Figure 37: Simulation results for FMRLC control of the cart and pendulum system for various values of g_{y_c}.

altogether thus clearly illustrating the "damping" effect obtained from employing $y_c(kT)$ in the fuzzy inverse model design. Finally, in Figure 37(c) for the case where $g_{y_c} = 1.0$, we find that further increases in g_{y_c} adversely affects the control such that the system is no longer capable of "keeping up" with the process.

For the final set of experiments performed for the cart and pendulum and the FMRLC control algorithm, we examine the effect of changing the number of control rules or fuzzy implications employed in the fuzzy controller. For a fuzzy controller, the number of fuzzy control rules necessary for "completeness" is the product of the number of fuzzy sets defined for each universe of discourse. Observe that when smaller numbers of fuzzy sets are defined for a given fuzzy controller input, each fuzzy set for this input becomes less "precise" or has a "wider" membership function. Consequently, a smaller number of fuzzy controller implications is likely to result in a less accurate mapping between the inputs and outputs of the fuzzy controller. Therefore, the effect of a single knowledge base modification when few control rules are employed is less "localized". As a result, a given fuzzy implication may be continually updated in the effort to obtain an accurate controller mapping.

From the above analysis, it is clear that it is very important to define a sufficient number of fuzzy sets for an accurate mapping between controller inputs and outputs. In Figure 38 below, we show the simulation results for FMRLC of the cart and pendulum system for a various number of fuzzy controller control rules. In Figures 38(a) and (b) which show the response for 9 and 25 fuzzy control rules, respectively, the overall system exhibits an oscillatory time response such that the oscillations never die out nor are attenuated. However, as we increase the number of control rules to 49 and 121, as shown in Figure 38(c) and (d), respectively, the oscillations are almost completely eliminated. Therefore, we may conclude that the

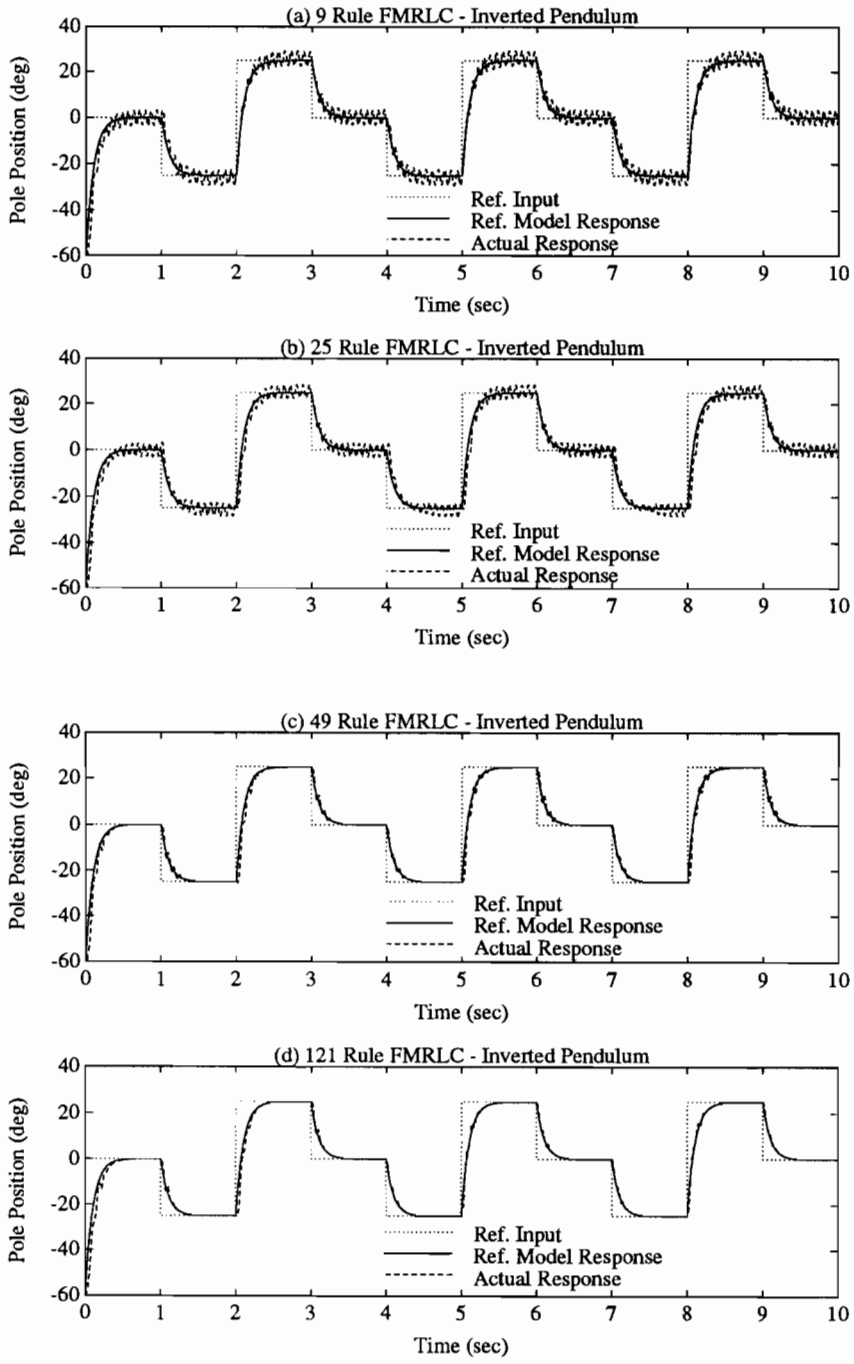


Figure 38: Simulation results for FMRLC control of the cart and pendulum system for a varying number of fuzzy control rules.

poor results shown in Figures 38(a) and (b) are most likely due to the fact that an insufficient number of control rules are defined to accurately describe the necessary mapping between controller inputs and outputs. Thus, the adaptation mechanism is continually modifying the control rules to compensate for these inaccuracies creating an oscillatory response.

4.2 Rocket Velocity Control - Problem Statement

In this section we show the design and simulation of a FMRLC design which as employed to control the velocity of a single stage rocket. A mathematical model for this process is presented by Barrére *et al.* in [4] and Mandell *et al.* in [53] and is expressed by the following differential equation:

$$\frac{d\ v(t)}{dt} = c(t)\left(\frac{m}{M - m\ t}\right) - g_o\left(\frac{R}{R + y(t)}\right) - 0.5\ v^2(t)\left(\frac{\rho_a\ A\ C_d}{M - m\ t}\right), \qquad (4.13)$$

where $v(t)$ is the rocket velocity at time t, $y(t)$ is the altitude of the rocket (above sea level), $c(t)$ is the velocity of the exhaust gases, and for our simulation the rocket parameters are given below:

1. $M = 15000.0\ kg$ - initial mass of the rocket and fuel,
2. $m = 100.0\ \frac{kg}{s}$ - exhaust gases mass flow rate (approximately constant for some solid propellant rockets),
3. $A = 1.0\ meter^2$ - maximum cross sectional area of the rocket,
4. $g_o = 9.8\ \frac{meters}{s^2}$ - the acceleration due to gravity at sea level,
5. $R = 6.37 \times 10^6\ meters$ - radius of the earth,
6. $\rho_a = 1.21\ \frac{kg}{m^3}$ - density of air,

7. $C_d = 0.3$ - drag coefficient for the rocket.

The mathematical model in Equation (4.13) was developed based on the simple dynamics of a point mass. However, in general, rockets dynamics are studied in the realm of exterior ballistics. This type of analysis often tends to be very complex and falls outside the scope of this thesis. However, even in this restricted context the modeled dynamics have characteristics which make for difficult control.

For example, due to the loss of fuel resulting from the combustion and exhaust the rocket has a time-varying mass. Furthermore, it can be determined by inspection of Equation (4.13) that the system is a non-linear process. The primary purpose for considering this control application is to investigate the capability of the FMRLC algorithm for controlling highly non-linear, time-varying processes.

As stated before, the control objective is to control the velocity of the rocket. To accomplish this task the rocket is assumed to have one input, namely the velocity of the exhaust gases $c(t)$. In general, the exhaust gas velocity is proportional to the cross-sectional area of the nozzle. Consequently, the exhaust gases may be controlled by changing this cross sectional area. However, for this controller implementation, we assume that the dynamics of the actuators which change the nozzle area and the dynamics of the exhaust gases are fast relative the rocket velocity dynamics and therefore may be eliminated from the model.

4.2.1 FMRLC Design

The fuzzy controller for this application was designed similar to the one for the inverted pendulum. The inputs to the fuzzy controller are the velocity error and change in error and the controller output is the velocity of the exhaust gases. In this fuzzy controller design, 11 fuzzy sets are defined for each controller input such

that the membership functions are "triangular shaped" and evenly distributed on appropriate universe of discourse. The "normalizing" controller gains for the error, change error, and the controller output are chosen to be $g_e = \frac{1}{4000}$, $g_c = \frac{1}{2000}$, and $g_u = 10000$, respectively. The fuzzy sets for the fuzzy controller output are also assumed to be "triangular shaped" with a width of 0.4 on the "normalized" universe of discourse. The knowledge base array was initially chosen with all zero entries. The fuzzy controller sampling period was chosen to be $T = 100$ milliseconds.

The reference model for this process was arbitrarily chosen and is expressed by the following differential equation

$$\frac{d\ y_m(t)}{dt} = -0.2\ y_m(t) + 0.2\ y_r(t), \tag{4.14}$$

where $y_m(t)$ specifies the desired system performance for the rocket velocity $v(t)$. For implementation in the FMRLC framework, the input to the reference model $y_r(t)$ is equal to the desired rocket velocity.

The input to the fuzzy inverse model include the error and change in error between the reference model and the rocket velocity expressed as

$$y_e(kT) = y_m(kT) - v(kT), \tag{4.15}$$

$$y_c(kT) = \frac{y_e(kT) - y_e(kT-T)}{T}, \tag{4.16}$$

respectively. For these inputs, 11 fuzzy sets are defined with "triangular shaped" membership functions which are evenly distributed on the appropriate universes of discourse. The "normalizing" controller gains associated with $y_e(kT)$, $y_c(kT)$, and $p_i(kT)$ are chosen to be $g_{y_e} = \frac{1}{4000}$, $g_{y_c} = \frac{1}{2000}$, and $g_{p_i} = 10000$, respectively. For the rocket process for an increase in the exhaust gas velocity we would generally expect an increase in the process output. This implies that the incremental relationship

between the controller inputs and outputs is monotonically increasing. Consequently, the knowledge base array shown below in Table 14 was employed for the fuzzy inverse model.

Table 14: Knowledge base array table employed in the fuzzy inverse model for the rocket system.

$P_i^{j,k}$		Y_c^k										
		-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
Y_e^j	-5	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0
	-4	-1.0	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$
	-3	-1.0	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$
	-2	-1.0	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$
	-1	-1.0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$
	0	-1.0	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$
	$+1$	-0.8	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$
	$+2$	-0.6	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$
	$+3$	-0.4	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$	$+1.0$
	$+4$	-0.2	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$	$+1.0$	$+1.0$
	$+5$	0.0	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.0$	$+1.0$	$+1.0$	$+1.0$	$+1.0$

4.2.2 Simulation Results

The simulation results for the FMRLC of the rocket are shown below in Figure 39. Note that the system exhibits "good" tracking with the reference model even after the mass of the rocket has been reduced significantly from the initial mass due to fuel loss. This application clearly illustrates the effectiveness of the FMRLC algorithm for controlling highly time varying processes.

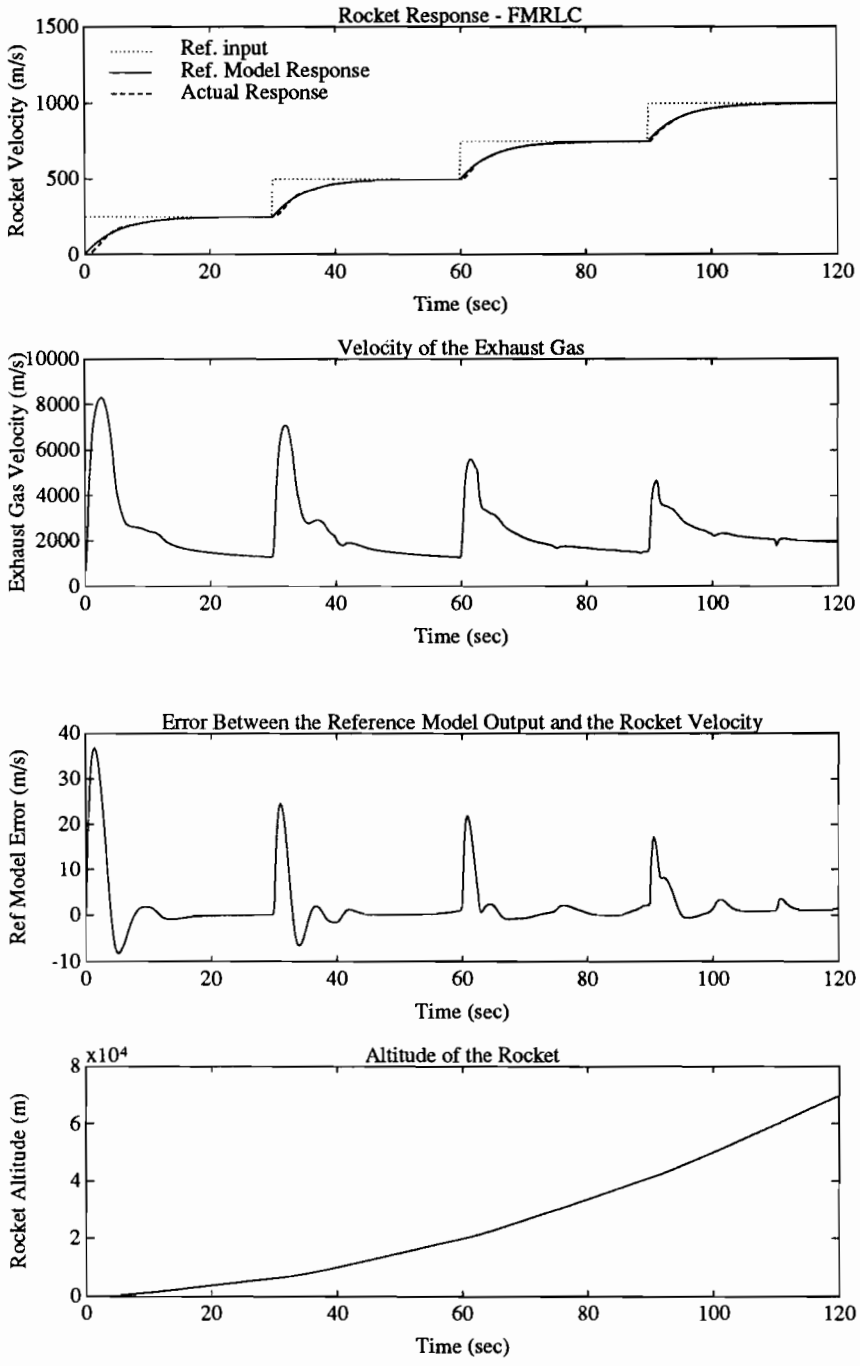


Figure 39: Simulation results for FMRLC control of the rocket system.

4.3 Two-Degree of Freedom Robot Manipulator - Problem Statement

Figure 40 illustrates the physical model of a two degree of freedom manipulator. It consists of two links where link #1 is mounted on a rigid base by means frictionless hinge and link #2 is mounted at the end of link #1 by means of a frictionless ball bearing. This control problem is provided to illustrate the application of the FMRLC

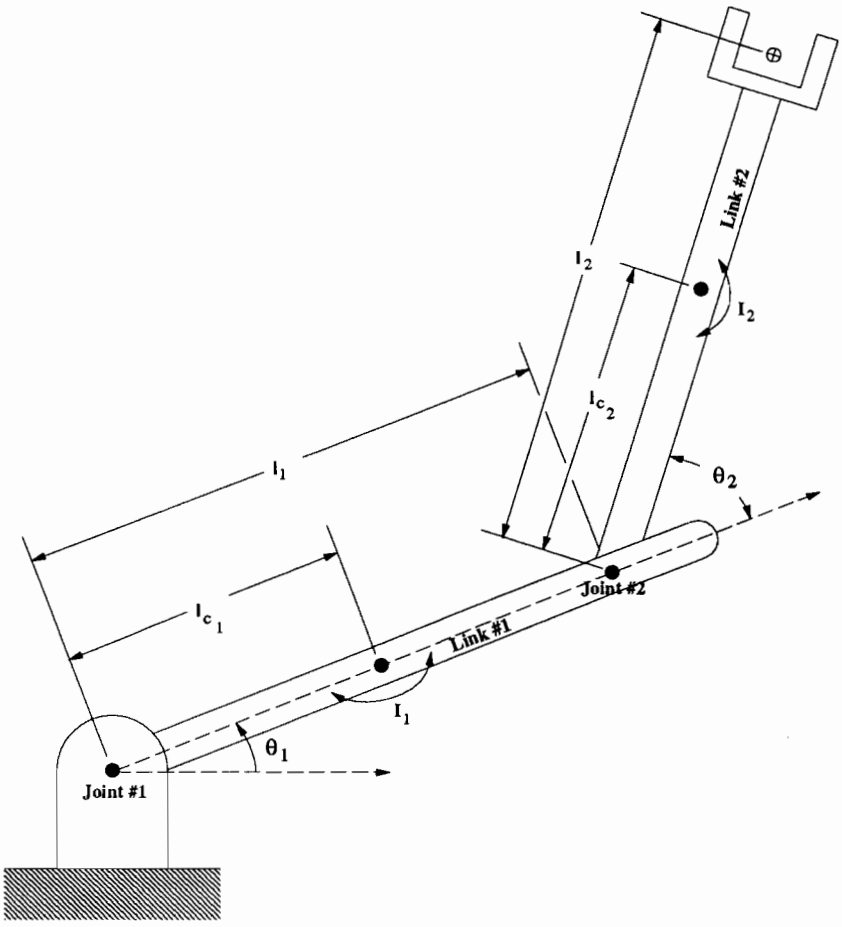


Figure 40: Graphical representation of a 2-link robot.

to a multi-input multi-output system. The inputs to the process are the torques τ_1

and τ_2 which are applied to the links at joints #1 and #2. The outputs are the joint positions θ_1 and θ_2.

A mathematical model for this process is presented by Slotine and Li in [68] and Narendra and Annaswamy in [55]. This model was developed using the well-known Lagrangian equations in classical dynamics and is expressed by the following matrix differential equation:

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} -h\dot{\theta}_2 & -h\dot{\theta}_1 - h\dot{\theta}_2 \\ h\dot{\theta}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \quad (4.17)$$

where

$$H_{11} = m_1 l_{c_1}^2 + I_1 + m_2[l_1^2 + l_{c_2}^2 + 2l_1 l_{c_2} \cos(\theta_2)] + I_2 \quad (4.18)$$

$$H_{22} = m_2 l_{c_2}^2 + I_2 \quad (4.19)$$

$$H_{12} = H_{21} = m_2 l_1 l_{c_2} \cos(\theta_2) + m_2 l_{c_2}^2 + I_2 \quad (4.20)$$

$$h = m_2 l_1 l_{c_2} \sin(\theta_2) \quad (4.21)$$

$$g_1 = m_1 l_{c_1} g \cos(\theta_1) + m_2 g[l_{c_2} \cos(\theta_1 + \theta_2) + l_1 \cos(\theta_1)] \quad (4.22)$$

$$g_2 = m_2 l_{c_2} g \cos(\theta_1 + \theta_2), \quad (4.23)$$

and where $\theta = [\theta_1 \ \theta_2]^T$ are the two joint angles, $\tau = [\tau_1 \ \tau_2]^T$ are the input joint torques. For purpose of our simulation the robot parameters are given below:

1. $m_1 = 1.0\ kg$ - mass of link #1,

2. $m_2 = 1.0\ kg$ - mass of link #2,

3. $l_1 = 1.0\ meters$ - length of link #1,

4. $l_2 = 1.0\ meters$ - length of link #2,

5. $l_{c_1} = 0.5\ meters$ - distance from joint #1 to the center of gravity of link #1,

6. $l_{c_2} = 0.5\ meters$ - distance from joint #2 to the center of gravity of link #2,

7. $I_1 = 0.2\ kg - m^2$ - lengthwise centroidal inertia of link #1,

8. $I_2 = 0.2\ kg - m^2$ - lengthwise centroidal inertia of link #2.

4.3.1 FMRLC Design

For this application, the process contains the two inputs, namely τ_1 and τ_2. Consequently, two MISO fuzzy controllers are needed for this process (one for each process input). The inputs to the fuzzy controller are the robot joint position error $\underline{e} = [e_1\ e_2]^T$ and change in error $\underline{c} = [c_1\ c_2]^T$. The fuzzy controllers have outputs τ_1 for the first controller and τ_2 for the second controller. For both fuzzy controller designs, 11 fuzzy sets are defined for each controller input such that the membership functions are "triangular shaped" and evenly distributed on appropriate universes of discourse. Also, the "normalizing" controller gains for the error, change error, and the controller output are chosen to be $\underline{g}_e = [\frac{1}{2\pi}\ \frac{1}{2\pi}]^T$, $\underline{g}_c = [\frac{1}{20}\ \frac{1}{20}]^T$, and $\underline{g}_u = [100\ 25]^T$, respectively. The fuzzy sets for the both fuzzy controllers outputs are also assumed to be "triangular shaped" with a width of 0.4 on the "normalized" universes of discourse. The knowledge base array for both fuzzy controllers was initially chosen with all zero entries. The fuzzy controller sampling period was chosen to be $T = 5$ milliseconds.

The reference model for this FMRLC design is given by the following differential equation

$$\begin{bmatrix} \dot{y}_{m_1}(t) \\ \dot{y}_{m_2}(t) \end{bmatrix} = \begin{bmatrix} -0.75 & 0.0 \\ 0.0 & -1.5 \end{bmatrix} \begin{bmatrix} y_{m_1}(t) \\ y_{m_2}(t) \end{bmatrix} + \begin{bmatrix} +0.75 & 0.0 \\ 0.0 & +1.5 \end{bmatrix} \begin{bmatrix} y_{r_1}(t) \\ y_{r_2}(t) \end{bmatrix}, \quad (4.24)$$

where y_{m_1} and y_{m_2} specify the system performance for θ_1 and θ_2, respectively. For FMRLC implementation, the inputs to the reference model y_{r_1} and y_{r_2} are equal to the desired position of joints #1 and #2, respectively,

For this FMRLC design, two fuzzy inverse models are needed, one for each fuzzy controller. In general, both process inputs will affect both process outputs. However, for this fuzzy inverse model design we will assume that the cross-coupling between the inputs is negligible (i.e., τ_1 affects only θ_1 and τ_2 affects only θ_2). As a result, the input to a given fuzzy inverse model includes the error and change in error between the associated reference model output and robot position. Therefore, for the i^{th} fuzzy inverse model, these inputs may expressed as

$$y_{e_i}(kT) = y_{m_i}(kT) - \theta_i(kT), \tag{4.25}$$

$$y_{c_i}(kT) = \frac{y_{e_i}(kT) - y_{e_i}(kT-T)}{T}, \tag{4.26}$$

respectively. For these inputs, 11 fuzzy sets are defined with "triangular shaped" membership functions which are evenly distributed on the appropriate universe of discourse. The "normalizing" fuzzy system gains associated with $\underline{y}_e(kT)$, $\underline{y}_c(kT)$, and $\underline{p}_i(kT)$ are chosen to be $\underline{g}_{y_e} = [\frac{1}{2\pi} \ \frac{1}{2\pi}]^T$, $\underline{g}_{y_c} = [1 \ \frac{1}{2}]^T$, and $\underline{g}_{p_i} = [100 \ 25]^T$, respectively. For the robot process for an increase in the torque τ_1 we would generally expect an increase in the process output θ_1. Likewise, for an increase in the torque τ_2 we would generally expect an increase in the process output θ_2. This implies that the incremental relationship between both controller inputs and outputs is monotonically increasing. Consequently, the knowledge base array shown below in Table 15 was employed for both fuzzy inverse model.

Table 15: Knowledge base array table employed in the fuzzy inverse model for the rocket system.

$P_i^{j,k}$		Y_c^k										
		−5	−4	−3	−2	−1	+0	+1	+2	+3	+4	+5
Y_e^j	−5	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0
	−4	−1.0	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2
	−3	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4
	−2	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6
	−1	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8
	0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0
	+1	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0
	+2	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0
	+3	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0
	+4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0	+1.0
	+5	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0	+1.0	+1.0

4.3.2 Simulation Results

The simulation results for the FMRLC of the two degree-of-freedom robot manipulator are shown below in Figure 41 for joint #1 and Figure 42 for joint #2. Once again the FMRLC provided good system tracking with respect to the reference model. As a result, the system exhibits good steady state and transient response. In fact, the response for joint #1 in Figure 41 was so close to the response of the reference model that the two almost perfectly overlap.

4.4 Chapter Summary

The purposes of this chapter were to present illustrate the design methodology of the linguistic SOC and the FMRLC, compare the effectiveness of the two algorithms by application for a cart and pendulum system, illustrate the effectiveness of the

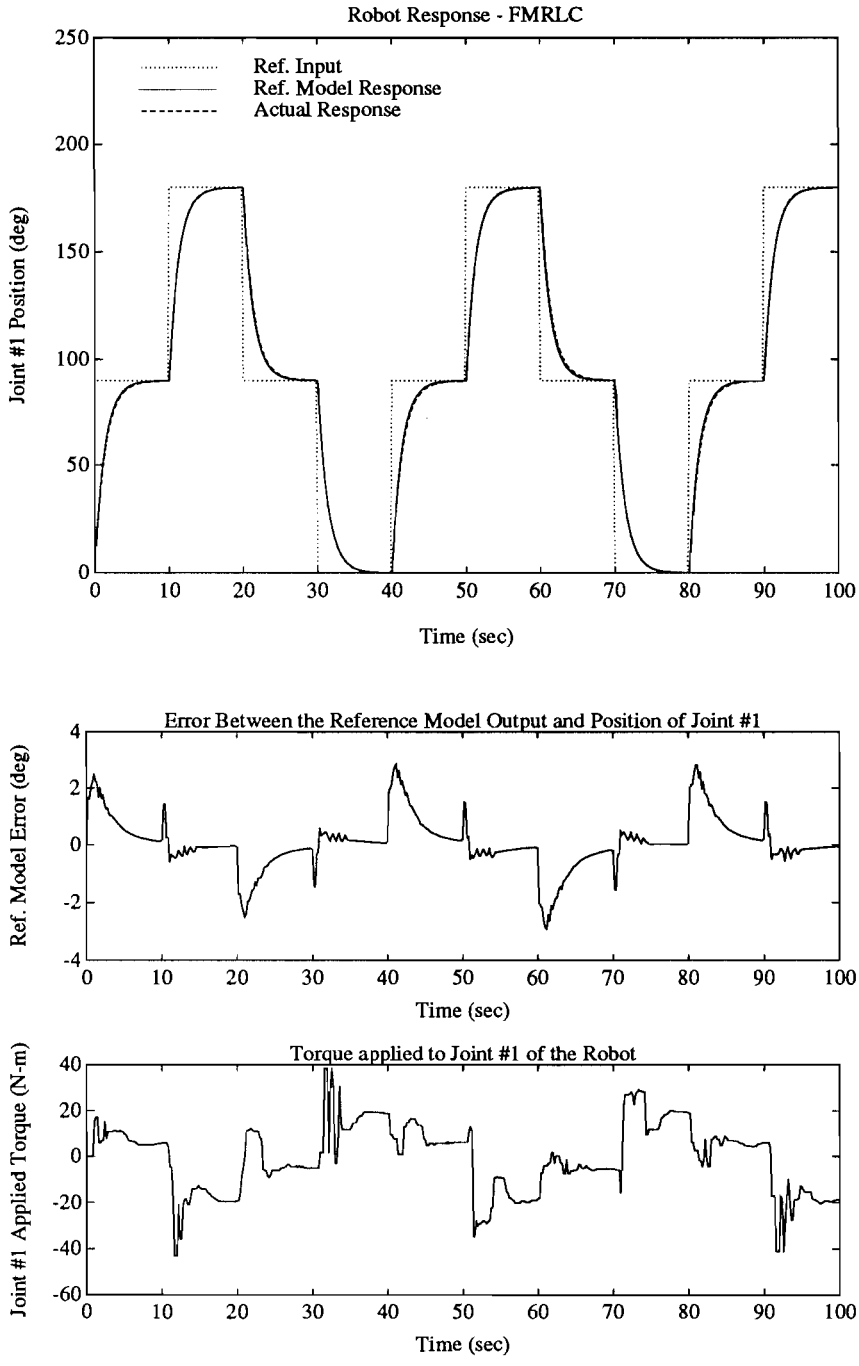


Figure 41: Simulation results for joint #1 of FMRLC controlled robot system.

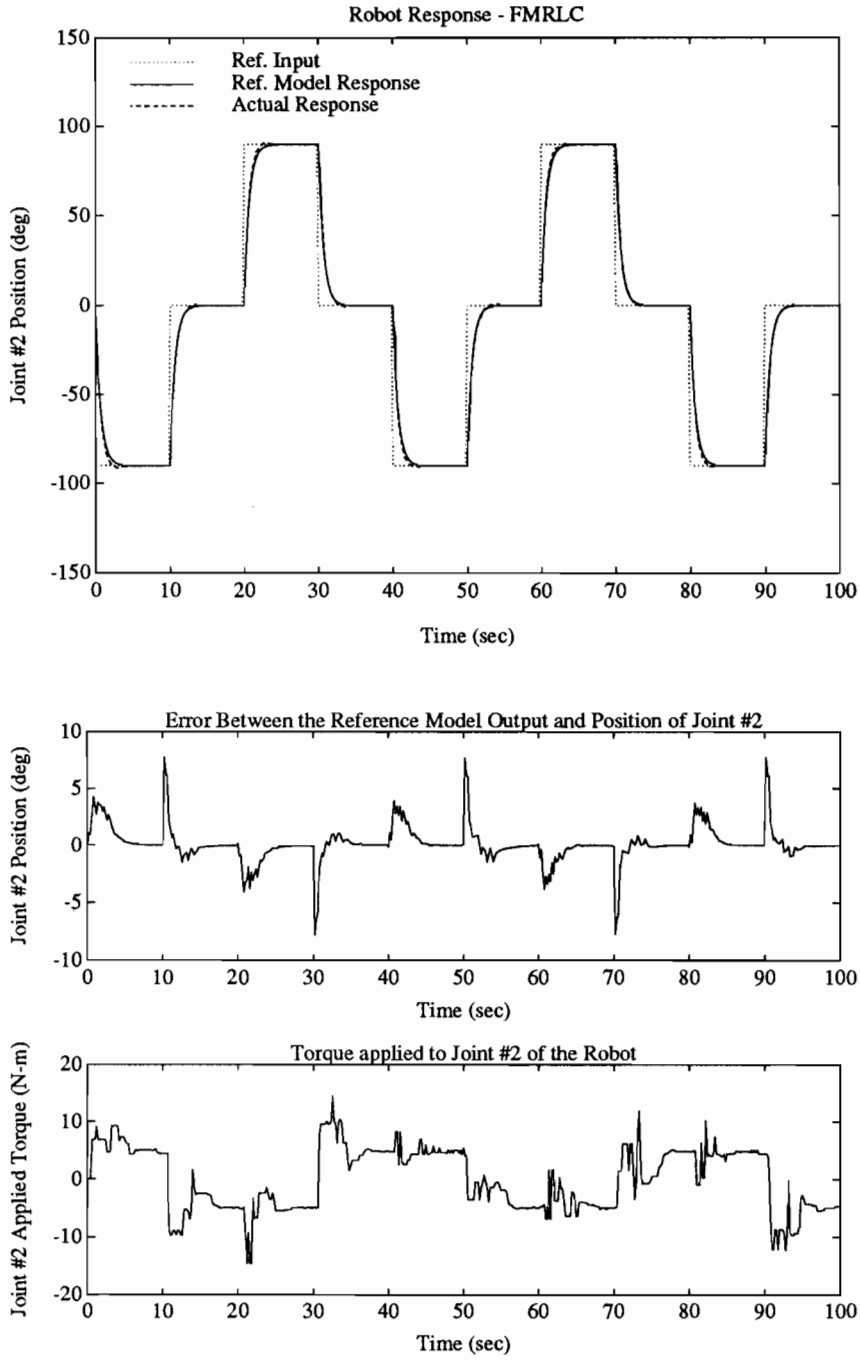


Figure 42: Simulation results for joint #2 of FMRLC controlled robot system.

FMRLC for controlling highly time-varying, non-linear, process by application for the rocket, and demonstrate the ability of the FMRLC for controlling non-linear, time-varying, MIMO processes by application on the two degree-of-freedom robot manipulator.

By application of the linguistic SOC and the FMRLC for the cart and pendulum system, we have shown that the FMRLC provides greater flexibility in defining the desired system performance. Furthermore, we have shown that by employing a fuzzy inverse model rather than a simple matrix inverse model we are able to better quantify the qualitative relationship between process inputs and outputs.

Finally, we have shown that there exist certain limitations in the FMRLC framework. For instance, the entire design process tends to be somewhat *ad hoc*. As a result, the stability and reliability of a given control algorithm is not easily determined. However, we have shown that some basic design guidelines do exist for FMRLC.

CHAPTER V

Vehicle Traction Control

In this chapter we evaluate the design and performance of FMRLC algorithm when employed to control an automotive braking system. These experiments illustrate the practical application of the FMRLC algorithm for a very complex real world process.

During an emergency braking procedure, the "optimal" braking force which can be applied to a vehicle occurs when the wheels are moving slower than the vehicle but are not completely locked-up. Generally this optimal braking will result in faster stops and improved steering capabilities. However, due to the very fast dynamics of conventional braking systems, it is difficult or even impossible for a human operator to achieve this optimal braking response, particularly given that the human operator is likely to be preoccupied with other tasks during an emergency stop such as steering. Furthermore, the braking characteristics become very different for changing road surfaces or conditions (i.e., asphalt, concrete, dry, wet, ice, etc.). In order to obtain faster/safer braking and stable steering capabilities during an emergency stop, many companies have adopted the use of Anti-skid/lock Braking Systems (ABS).

5.1 Relevant Literature

In this section we overview some of the literature which describes control of automotive traction systems including both anti-skid braking and anti-spin acceleration control.

In [77, 78], Tan and Tomizuka present a control algorithm for vehicle traction control, which includes both anti-skid braking control and anti-spin acceleration. This algorithm employs a non-linear feedback scheme that can be easily implemented in a digital computer. Since the controller presented in this article was designed so that is accounts for large system parameter variations, Tan and Tomizuka claim this algorithm provides "robust" control. The effectiveness of this control scheme was demonstrated by experiments on anti-skid braking systems which was conducted in the dynamometer test cell at General Motors Research Laboratories.

Fling and Fenton in [25] present a describing function approach for designing anti-skid braking systems. This research involves modeling of the braking system and vehicle dynamics, a proposal of a feedback compensator scheme to provide anti-skid behavior, and determination of design parameters by a describing function analysis of the system non-linearities. The resulting design was implemented on a 1969 Plymouth and evaluated in field tests. The results of these tests have shown that the describing function approach is capable of accurately predicting the system performance.

In [87], Yoneda, Naitoh, and Kigoshi present a rear brake anti-lock system which was mounted on a Mitsubishi Starion. This system employs two sensors for performance feedback. One is a speed sensor mounted at the speedometer cable take-out of the transmission, to generate a wheel velocity signal. The other is a G-sensor

mounted in the luggage compartment, to generate the vehicle deceleration signal in the car's longitudinal direction. By monitoring these sensors a knowledge based controller employs one of three type of brake line pressure control actions including: 1) *release* (REL), 2) *fast build* (FB), and 3) *slow build* (SB). The control scheme was tested on a large variety of road surfaces from a high coefficient of adhesion to a low one and also split road surface. These tests proved that a car equipped with this system provides better braking than a car without the system. This advantage was most markedly seen on a split surface where conventional braking systems often resulted in the vehicle going out of control.

Another knowledge based control scheme for anti-skid braking systems was presented by Tabo, Ohka, Kuraoka, and Ohba in [73]. In this research, the overall control algorithm may be viewed as a wheel speed servo mechanism. The system employs a single sensor which measures wheel velocity. The command input is a wheel speed command. The system actuator provides several modes of operation which include 4 brake line pressure commands: 1) *slow decrease*, 2) *quick decrease*, 3) *slow increase*, and 4) *quick increase*. As a result, the controller becomes a knowledge based system which provides a type of *sliding mode* control.

An adaptive control scheme was applied to an anti-locking braking system by Guntur and Ouwerkerk in [32]. The scheme uses prediction and reselection methods to determine wheel lock-up. Since wheel slip is difficult to measure, other measurable parameters are used to estimate the wheel slip. Using these parameters, it is possible to determine whether or not a braked wheel is on the verge of locking up. If the wheel locks-up (prediction point) then action is taken, otherwise, braking is continued (reselection point). Guntur and Ouwerkerk argue that this scheme is an adaptive algorithm since prediction and reselection points are continually updated.

Takahashi and Ishikawa in [75], present an anti-skid braking control system which is based in fuzzy set theory and control. This algorithm includes a detection scheme for sensing and estimating various parameters such as vehicle speed, wheel speed, wheel slip rate, first time derivatives of vehicle and wheel speed, and the second time derivatives of vehicle and wheel speed. Fuzzy inference is employed on the values of the above parameters to generate a desired brake line pressure.

In [48], Leiber and Czinczel study the effects of various brake line pressure waveforms during an ABS cycle. In this research, a combination of wheel acceleration and wheel slip signals are employed to determine the "optimal" braking action. The results of this research indicate that the five gradient pressure waveform results in a more accurate control of the wheel speed when subjected to changing friction coefficients. The five gradient waveform may be described by the following pattern: 1) a quick pressure increase, 2) quick pressure decrease, 3) pressure retention, 4) quick pressure increase, 5) followed by a slow stepwise pressure increase. In [49], Leiber and Czinczel provide a summary of other experiences with ABS systems including reliability, safety, and design issues.

A method for evaluating ABS systems was illustrated by Grimm, Bremer, Jain and Levijoki in [31]. This research involves generating a mathematical model of the vehicle dynamics for real-time hybrid computer simulations. The simulation was interconnected with a ABS modulator hardware and air brake system of a multi-axle truck to provide a laboratory tool for simulating vehicle braking performance. It was found that this hybrid simulation configuration provided an accurate means for predicting and evaluating the actual ABS performance.

5.2 Problem Statement

The objective of an ABS system is to regulate wheel slip to maximize the coefficient of friction between the tire and road for any given road surface. In general, the coefficient of friction can be described as a function of slip, λ, which for a braking operation is defined as

$$\lambda(t) = \frac{\frac{V_v(t)}{R_w} - \omega_w(t)}{\frac{V_v(t)}{R_w}}, \tag{5.1}$$

where $\omega_w(t)$ is the angular velocity of the wheel, $V_v(t)$ is the velocity of the vehicle, and R_w is the radius of the tire. Consequently, 0% slip represents the free rolling wheel condition while 100% corresponds to a wheel that is locked up and sliding. Note that the term $\frac{V_v(t)}{R_w}$ is the angular velocity of the vehicle with respect to the tire angular velocity; therefore, we will sometimes denote this quantity by $\omega_v(t)$.

Typical tire-to-road friction coefficients are shown as a function of slip (λ) in Figure 43[34, 45]. The coefficient of braking affects the the amount of force which is applied to the car as a result of applying the brakes. The lateral coefficient affects the force which can be generated at the sides of the tires and the road surface.

There are several characteristics of the curve shown in Figure 43 which illustrate the possible advantages of ABS system. First, the ratio of the peak value of the braking coefficient curve to the locked wheel value indicates the relative degree of potential for an ABS system with respect to that achieved from a locked wheels situation.

The second interesting characteristic involves the negative slope in the braking coefficient curve which occurs for slip values which are larger that the peak braking torque slip value. During a typical braking action, brake line pressure is increased

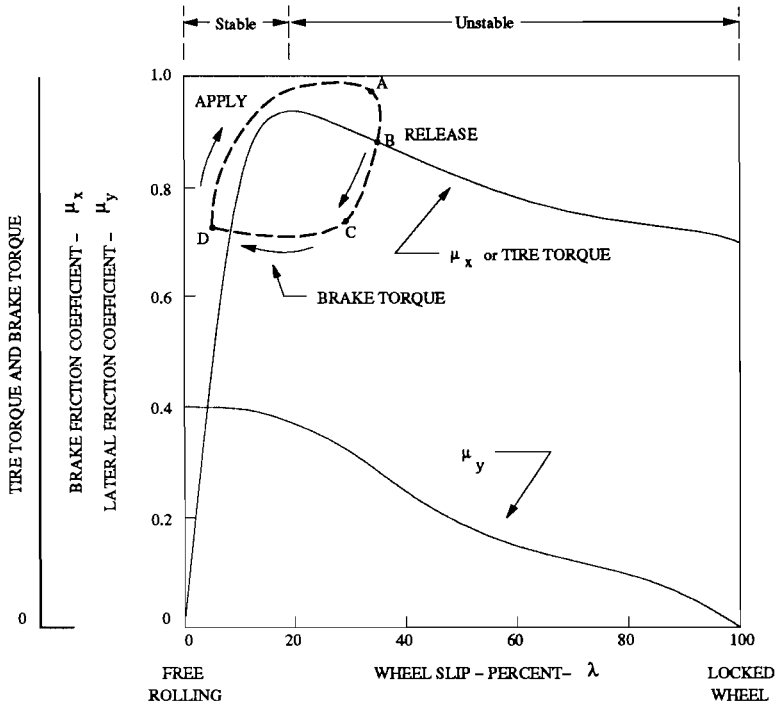


Figure 43: Road-tire friction coefficient vs. slip ratio.

to generate a torque which opposes movement of the wheel. As a result, friction is generated between the road surface and the tire which tend to oppose the direction of movement of the vehicle. However, due to the large momentum of the vehicle relative to the small inertia of the wheels, the dynamics of the vehicle braking tend to be slower than that of the braking action of the wheels. Consequently, slip is increased from a zero value. If the braking action of the wheel is sufficient to increase the slip beyond the peak value of the coefficient of braking (approx. 20%), the negative slope of the curve causes the uncompensated system to go unstable which causes the wheel to lock-up very quickly. Generally this *lock-up effect* occurs much faster than

a human operator's response thus making accurate manual compensation difficult or even impossible. As a result of the *lock-up effect*, a typical ABS system exhibits a limit cycle behavior as illustrated by the dash curve in Figure 43 which connects points A,B,C, and D.

Another interesting characteristic of the coefficient of friction curves involves the effect of wheel slip on the tires lateral force capabilities. Note in Figure 43 that an increase in slip results in a significant decrease in the lateral friction coefficient. Therefore, the maximum efficiency of the lateral force occurs when the slip is 0%. Furthermore, if the brakes are locked, slip value is 100%, then lateral coefficient of friction becomes 0%. Since the lateral friction coefficient affects the vehicle steering and directional stability, this explains the loss of steering which often occurs when the brakes are inadvertently locked during an emergency stop. Generally, there exists a sufficient lateral force at the peak braking coefficient of friction to provide a suitable compromise between braking and lateral forces.

The braking coefficient of friction vs. slip curve shown in Figure 43 is most influenced by tire and road surfaces/conditions. This fact is illustrated by Harned, Johnston, Scharpf in [34] and Rhee in [62] where the braking coefficients were measured with respect to wheel slip for various surfaces, loads, speeds, and temperatures. The results of these experiments were approximated for dry asphalt, wet asphalt, and ice as shown in Figure 44. As one would expect the braking coefficient of friction is greatest for dry asphalt, slightly reduced for wet asphalt, and greatly reduced for ice. This approximated data shown in Figure 44 will be used later for simulation of a proposed FMRLC algorithm for traction control.

During normal vehicle operation the road surface and conditions are constantly changing. Since the road surface directly affects the braking characteristic, a "ro-

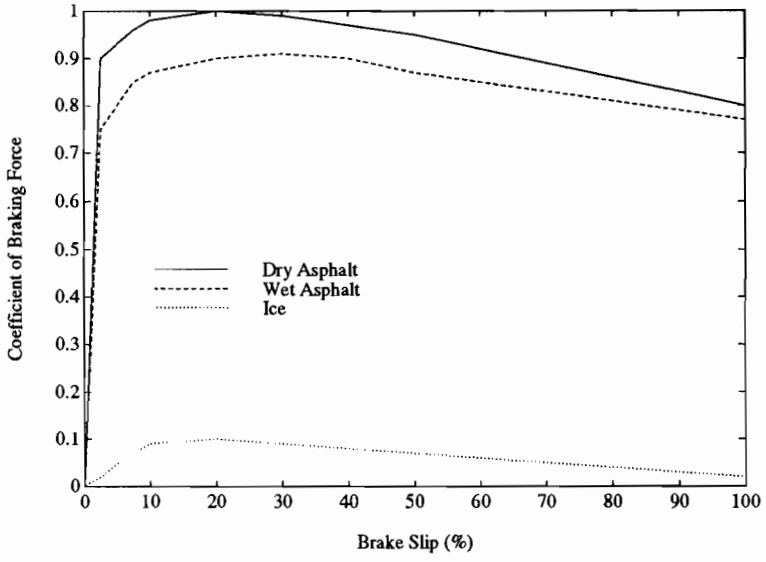


Figure 44: Road-tire friction coefficients vs. slip ratio for various road surfaces.

bust" controller design which compensates for all possible types of road conditions is difficult. Also, as will be shown later, the braking characteristics change significantly for inclined road surfaces. Therefore, in this chapter we propose implementing the FMRLC algorithm for this system to obtain and maintain a suitable controller for many road surfaces and conditions.

5.2.1 Modeling of Vehicle, Wheels, and Braking System

A simplified model for a vehicle, a single wheel, and its braking system is shown below in Figure 45. This model is appropriate for both vehicle acceleration and deceleration. The process model contains both linear vehicle dynamics and a one-wheel rotational dynamics where wind resistance effects and all the vertical dynamics associated with the suspension system are assumed negligible.

The wheel and vehicle dynamics are determined by employing Newton's Laws

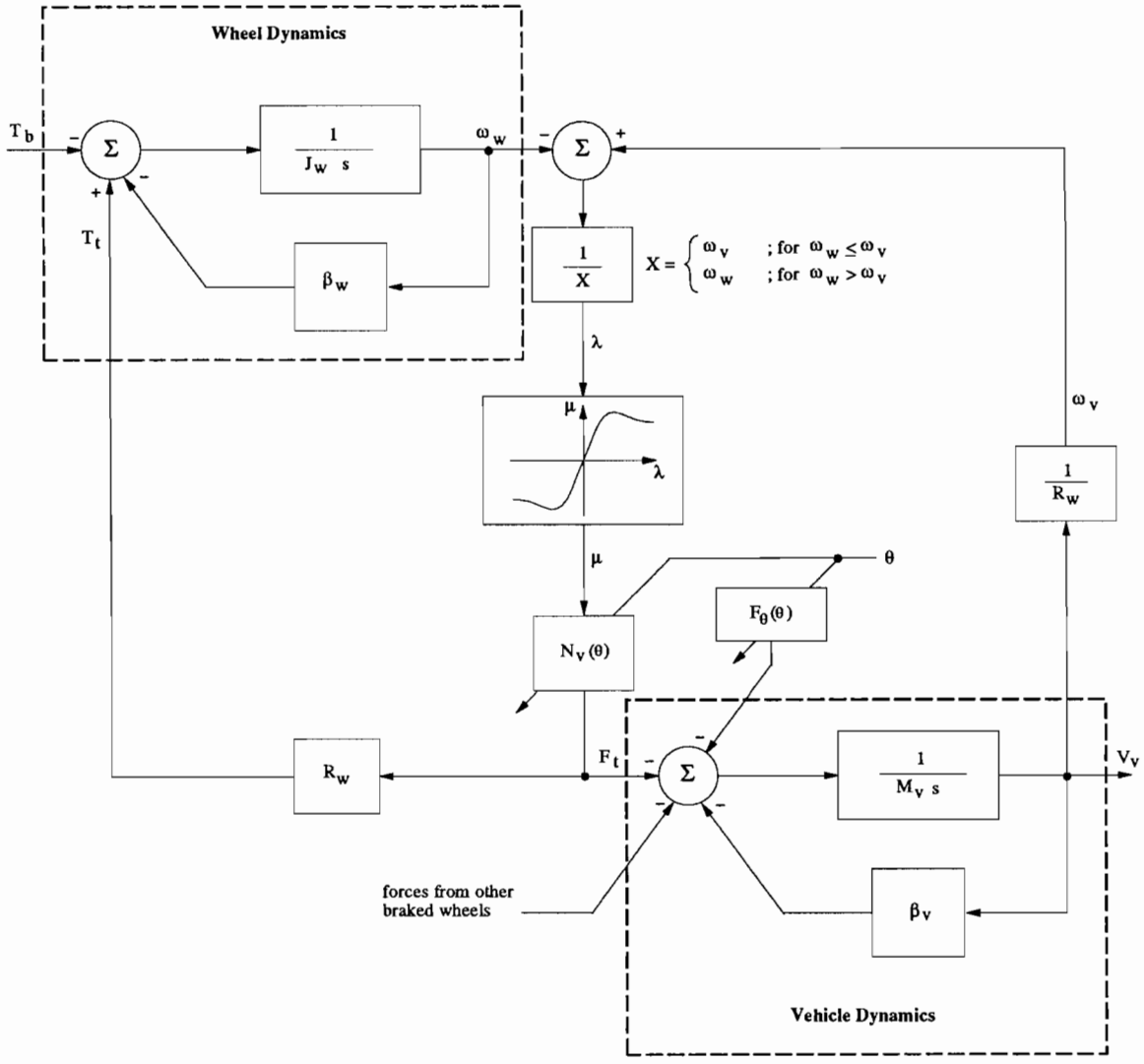


Figure 45: One wheel model of vehicle and braking system.

of motion. Consider the free body diagram of a single wheel traveling on an incline shown in Figure 46 below. The dynamic equation which describes the motion of the

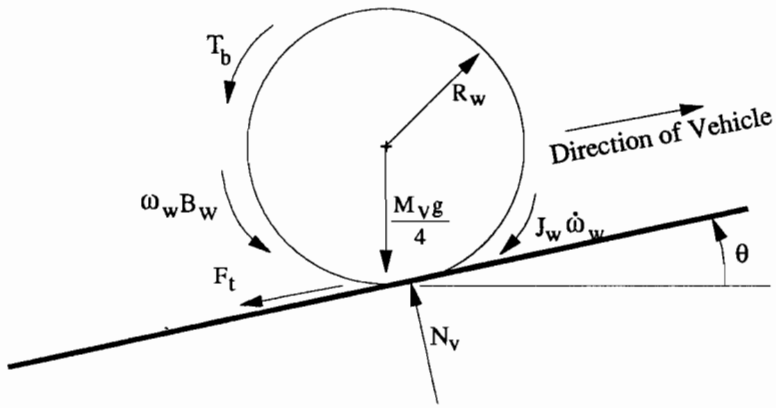


Figure 46: Free body diagram of the wheel during a braking action.

wheels can be determined by summing the rotational torques which are applied to the wheel; thus we obtain the following differential equation

$$\dot{\omega}_w(t) = \frac{1}{J_w}[-T_b(t) - \omega_w(t)B_w + T_t(t)], \tag{5.2}$$

where J_w is the rotational inertia of the wheel, B_w is the viscous friction of the wheel, $T_b(t)$ is the braking torque, and $T_t(t)$ is the torque generated due to slip between the wheel and the road surface. In general, $T_t(t)$ is a function of the force $F_t(t)$ exerted between the wheel and the road surface, or

$$T_t(t) = R_w F_t(t), \tag{5.3}$$

where R_w is the radius of the wheel.

The forces which are applied to the vehicle are shown in the free body diagram in Figure 47 below. The vehicle dynamics are determined by summing the total

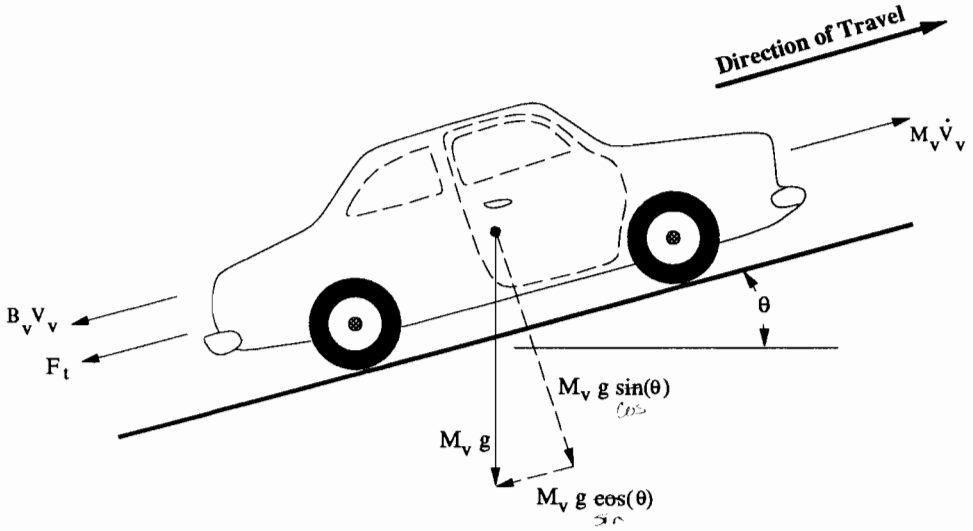


Figure 47: Free body diagram of the vehicle during a braking action.

forces applied to the vehicle during a normal braking operation. Thus we obtain the following differential equation

$$\dot{V}_v(t) = \frac{-1}{M_v}[F_t(t) + B_v V_v(t) + F_\theta(\theta)], \tag{5.4}$$

where M_v is the mass of the vehicle, B_v is the vehicle viscous friction, g is the gravitational acceleration constant, and $F_\theta(\theta)$ is the force applied to the car which result from a vertical gradient in the road. $F_\theta(\theta)$ is expressed by the following equation,

$$F_\theta(\theta) = M_v g \sin(\theta(t)). \tag{5.5}$$

The negative sign in Equation (5.4) results from the fact that the vector $M_v\dot{V}_v$ in Figure 47 is associated with a deceleration rather than an acceleration. Recall that

the vehicle angular velocity ω_v relative the to wheel angular velocity is expressed as

$$\omega_v(t) = \frac{V_v(t)}{R_w}. \tag{5.6}$$

The force $F_t(t)$ is generally expressed as a function of the coefficient of friction and the normal force on the wheel, or

$$F_t(t) = \mu(\lambda)N_v(\theta), \tag{5.7}$$

where $N_v(\theta)$ is the normal force applied to the tire and $\mu(\lambda)$ is the coefficient of friction as a function of slip. The wheel slip of a vehicle during deceleration ($\omega_w \leq \omega_v$) and acceleration ($\omega_w > \omega_v$) are defined as

$$\lambda(t) = \begin{cases} \frac{\omega_v(t)-\omega_w(t)}{\omega_v(t)} & ;\omega_w \leq \omega_v, \\ \frac{\omega_v(t)-\omega_w(t)}{\omega_w(t)} & ;\omega_w > \omega_v. \end{cases} \tag{5.8}$$

For this model we assume that the vehicle has 4 wheels and the weight of the vehicle is evenly distributed among these wheels. As a result, the normal force $N_v(\theta)$ may be expressed by

$$N_v(\theta) = \frac{M_v g}{4}\cos(\theta(t)), \tag{5.9}$$

where $\theta(t)$ is the angle of inclination in the road surface. This clearly illustrates the effect that road surface inclinations have on the braking characteristics.

Braking system parameters for a 1969 Plymouth are presented by Fling and Fenton in [25]. These values are summarized below in Table 16

5.3 Controller design

In this section we present a FMRLC algorithm for controlling slip of a conventional braking system. It is desired that a controller be capable of tracking any

Table 16: Vehicle/Wheel parameters for a 1969 Plymouth.

Symbol	Definition	Value
M_v	Vehicle Mass	$4 \times 342\ kg$
B_v	Viscous friction associated with the linear motion of the vehicle	$6\ N\ s$
J_w	Rotational inertia of the wheel	$1.13\ N\ m\ s^2$
R_w	Rolling radius of the wheel	$0.33\ m$
B_w	Viscous friction associated with the motion of the wheel	$4\ N\ s$
g	Gravity acceleration constant	$9.8\ m\ s^2$

desired wheel slip under all road conditions. However, in general, a slip of 20% will approximately result in the maximum possible coefficient of friction for most road surfaces. Therefore, for simulations presented later, the FMRLC algorithm is employed to regulate the slip about 20% for a variety of road surfaces including: dry asphalt, wet asphalt, ice, and combinations of these (split surfaces).

We begin by presenting a method for determining the slip value. This will be followed by a brief discussion of the design of the controller components.

5.3.1 Estimation of Slip

Since slip is the controlled parameter of the braking system, we desire to measure this quantity. However, currently it is difficult/impossible to measure slip directly. Given that slip is a function of $\omega_w(t)$ and $\omega_v(t)$ it seems logical to obtain it by measuring these parameters. However, measuring the angular velocity of the vehicle $\omega_v(t)$ has certain drawbacks. For instance, one possible method of measuring $\omega_v(t)$ involves leaving one wheel un-braked and measuring the velocity of that wheel. How-

ever, in general, this would result in longer stops since the potential braking force of this un-braked wheel is lost. Alternatively, we could add a small wheel solely for measuring $\omega_v(t)$; this would significantly increase the expense and maintenance for an ABS system. Therefore, it would be desirable if the slip could be accurately estimated with some sort of observer/estimator and a minimum number sensors.

As others have in [75, 88], we will assume that sensors for measuring acceleration and wheel speed are available for estimating slip. Equation (5.8) for the deceleration case may be rewritten to obtain

$$\omega_w(t) = (1 - \lambda(t))\, \omega_v(t). \tag{5.10}$$

Taking the time derivative of Equation (5.10) yields

$$\dot{\omega}_w(t) = (1 - \lambda(t))\, \dot{\omega}_v(t) - \dot{\lambda}(t)\, \omega_v(t), \tag{5.11}$$

where $\dot{\omega}_w(t)$ is related to the vehicle linear acceleration $a_v(t)$ by

$$\dot{\omega}_w(t) = \frac{\dot{V}_v(t)}{R_w} = \frac{a_v(t)}{R_w}. \tag{5.12}$$

Substituting Equation (5.12) and the fact $\omega_v = \frac{V_v}{R_w}$ into Equation (5.11), we obtain

$$\dot{\omega}_w(t) = (1 - \lambda(t))\, \frac{a_v(t)}{R_w} - \dot{\lambda}(t)\, \frac{V_v(t)}{R_w}. \tag{5.13}$$

Thus, by rearranging Equation (5.11) we can solve for the wheel slip derivative $\dot{\lambda}(t)$ which yields

$$\dot{\lambda}(t) = \left(\frac{1 - \lambda(t)}{V_v(t)}\right) a_v(t) - \left(\frac{R_w}{V_v(t)}\right) \dot{\omega}_w(t). \tag{5.14}$$

Equation (5.14) provides the general approach for estimating slip.

The next step is to discretize the system in Equation (5.14) for computer implementation. This may be achieved by a simple Euler's integration algorithm. As a result, Equation (5.14) may be approximated by the following difference equation

$$\lambda(kT+T) = \lambda(kT) + T\left[\left(\frac{1-\lambda(kT)}{\bar{V}_v(kT)}\right) a_v(kT) - \left(\frac{R_w}{\bar{V}_v(kT)}\right)\bar{\omega}_w(kT)\right], \quad (5.15)$$

where T is the sample period and where $\bar{\omega}_w(kT)$ and $\bar{V}_v(kT)$ are approximations for the wheel angular velocity and vehicle velocity and may be obtained by the following equations

$$\bar{\omega}_w(kT) = \frac{\omega_w(kT) - \omega_w(kT-T)}{T}, \quad (5.16)$$

$$\bar{V}_v(kT) = \bar{V}_v(kT-T) + T\ a_v(kT-T), \quad (5.17)$$

if the sampling period T is kept small enough. Above we have illustrated one possible method for approximating slip. However, this is not necessarily the best method and others may be used. See [45, 77, 78] for alternative methods of estimation of slip.

5.3.2 FMRLC Design

Given that we have an estimate of slip, we may design a FMRLC which uses this estimated slip value to regulate the actual wheel slip. For this design, we choose the desired slip to be 20% which is near "optimal" for most surfaces. Alternatively, a "selection knob" could be added to the vehicle's control panel which would allow the operator to select the value of slip which is most appropriate for the current driving conditions. Next we present a brief overview of the basic components of the FMRLC design which was employed for this system including: fuzzy controller design, reference model design, and the fuzzy inverse model design. It is suggested that the reader refer back to Chapter III for more details of this design.

The inputs to the fuzzy controller are the slip error and change in slip error expressed as

$$e(kT) = \lambda_r(kT) - \lambda(kT), \tag{5.18}$$

$$c(kT) = \frac{e(kT) - e(kT-T)}{T}, \tag{5.19}$$

respectively, where $\lambda_r(kT)$ is the desired slip. The controller output is the braking torque $T_b(kT)$ which is applied to the wheels. Here we assume the dynamics of the hydraulic actuator, which is generally used to create the braking torque, are much faster than the wheel and vehicle dynamics; thus they may be neglected. In this fuzzy controller design, 11 fuzzy sets are defined for each controller input such that the membership functions are "triangular shaped" and uniformly distributed on the appropriate universe of discourse. The "normalizing" controller gains for the error, change error, and the controller output are chosen to be $g_e = 1$, $g_c = \frac{1}{1000}$, and $g_u = 2200$, respectively. The fuzzy sets for the fuzzy controller output are also assumed to be "triangular shaped" with a width of 0.4 on the "normalized" universe of discourse. The knowledge base array was initially chosen with all zero entries. The fuzzy controller sampling period was chosen to be $T = 1$ millisecond.

The reference model for this process was arbitrarily chosen and is expressed by the following differential equation

$$\frac{d\ \lambda_m(t)}{dt} = -10.0\ \lambda_m(t) + 10.0\ \lambda_r(t), \tag{5.20}$$

where $\lambda_m(t)$ specifies the desired system performance for the braking slip $\lambda(t)$.

The inputs to the fuzzy inverse model include the error and change in error between the reference model and the wheel slip expressed as

$$\lambda_e(kT) = \lambda_m(kT) - \lambda(kT), \tag{5.21}$$

$$\lambda_c(kT) = \frac{\lambda_e(kT) - \lambda_e(kT-T)}{T}, \tag{5.22}$$

respectively. For these inputs, 11 fuzzy sets are defined with "triangular shaped" membership functions which are evenly distributed on the appropriate universes of discourse. The "normalizing" controller gains associated with $\lambda_e(kT)$, $\lambda_c(kT)$, and $p_i(kT)$ are chosen to be $g_{\lambda_e} = 1$, $g_{\lambda_c} = \frac{1}{1000}$, and $g_{p_i} = 2200$, respectively. In a typical braking system, an increase in the braking torque $T_b(kT)$, will generally result in an increase in the wheel slip. This implies that the incremental relationship between the process inputs and outputs is monotonically increasing. Consequently, the knowledge base array shown below in Table 17 was employed for the fuzzy inverse model. In Table 17, Λ_e^j is the j^{th} fuzzy set associated with the error signal λ_e and

Table 17: Knowledge base array table employed in the fuzzy inverse model for an ABS system.

$P_i^{j,k}$		Λ_c^k										
		−5	−4	−3	−2	−1	+0	+1	+2	+3	+4	+5
Λ_e^j	−5	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0
	−4	−1.0	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2
	−3	−1.0	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4
	−2	−1.0	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6
	−1	−1.0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8
	0	−1.0	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0
	+1	−0.8	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0
	+2	−0.6	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0
	+3	−0.4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0
	+4	−0.2	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0	+1.0
	+5	0.0	+0.2	+0.4	+0.6	+0.8	+1.0	+1.0	+1.0	+1.0	+1.0	+1.0

Λ_e^k is the k^{th} fuzzy set associated with the change in error signal λ_c.

5.4 Simulation Results

The FMRLC design described above was simulated using the FORTRAN programming language. The results of this simulation for wet asphalt, dry asphalt, and an icy surface are shown below in Figures 48, 49, and 50, respectively. For these simulation results, only one brake was applied. The braking action was initiated when the vehicle was moving 25 $\frac{meters}{sec}$ (appox. 56 $\frac{miles}{hour}$) on a level surface ($\theta = 0$).

Due to the fact that the wheel and vehicle velocity are nearly zero at low speed, the magnitude of slip tends to goes to infinity as the vehicle speeds approaches zero. This often causes the slip to become very sensitive at slow speeds and as a result slip is very difficult to control at slow speeds. Furthermore, wheel speed and vehicle acceleration measurements are often not very reliable for low vehicle speeds. Therefore as in similar ABS studies, we only performed the simulation until the vehicle is slowed to approximately 5 $\frac{meters}{sec}$.

Note for the wet and dry asphalt cases that the reference model output and the braking system slip value tracked almost perfectly. As a result, the system does not exhibit the limit cycle effect which is typical of most ABS designs. Also, the braking torque for these cases were very smooth. The controller seems to have found the appropriate level of braking torque which needs to be applied to the wheels to maintain a slip of 20%.

Although the simulation results for the icy surface shown in Figure 50 are likely to be considered acceptable by most control engineers, they are not as good as the results obtained for wet asphalt and dry asphalt road conditions. In general, it is very difficult to control slip on an icy surface due to the fact that a very small braking torque is likely to cause lock-up. This becomes even more difficult at slower

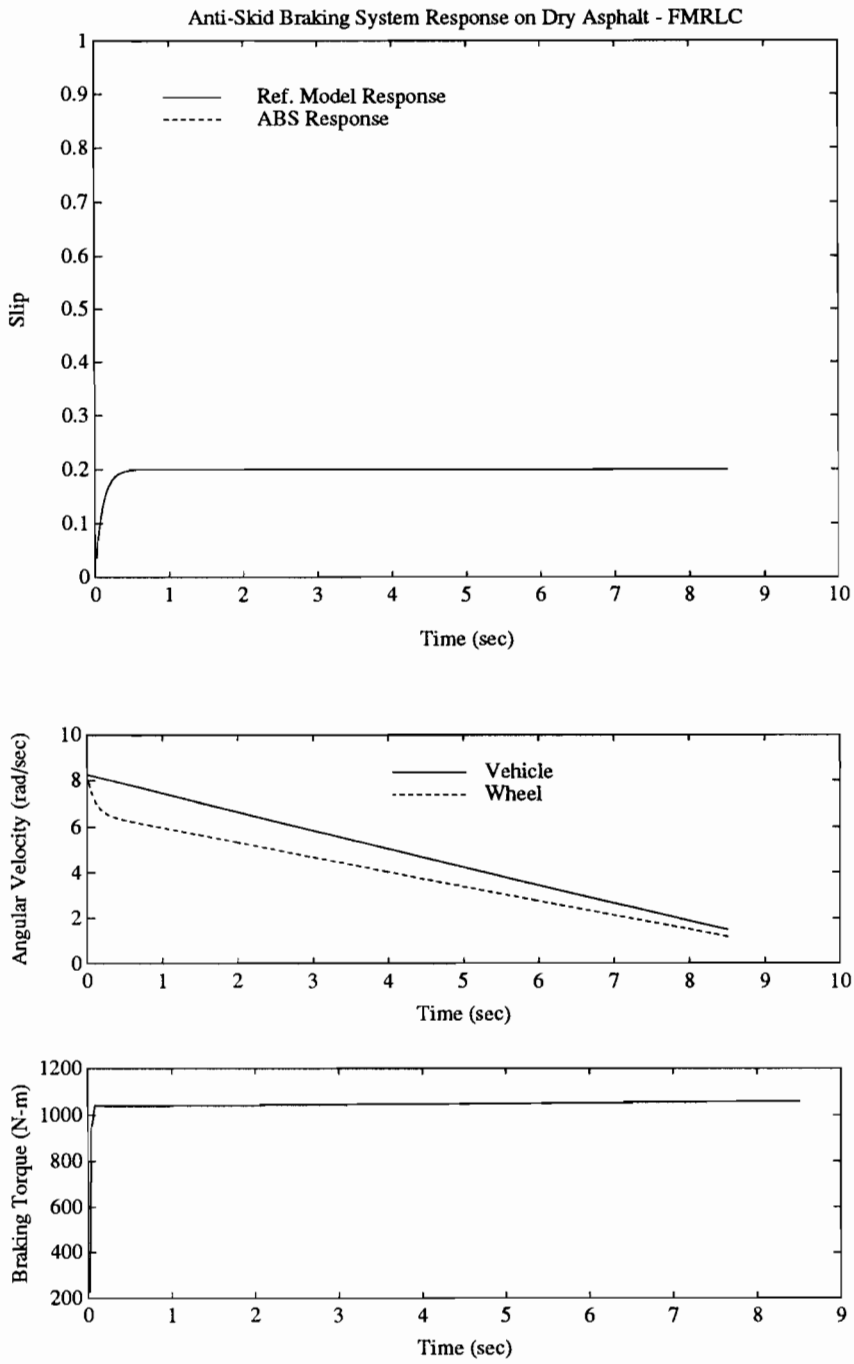


Figure 48: Simulation results for FMRLC of a vehicle braking system on a level dry asphalt surface.

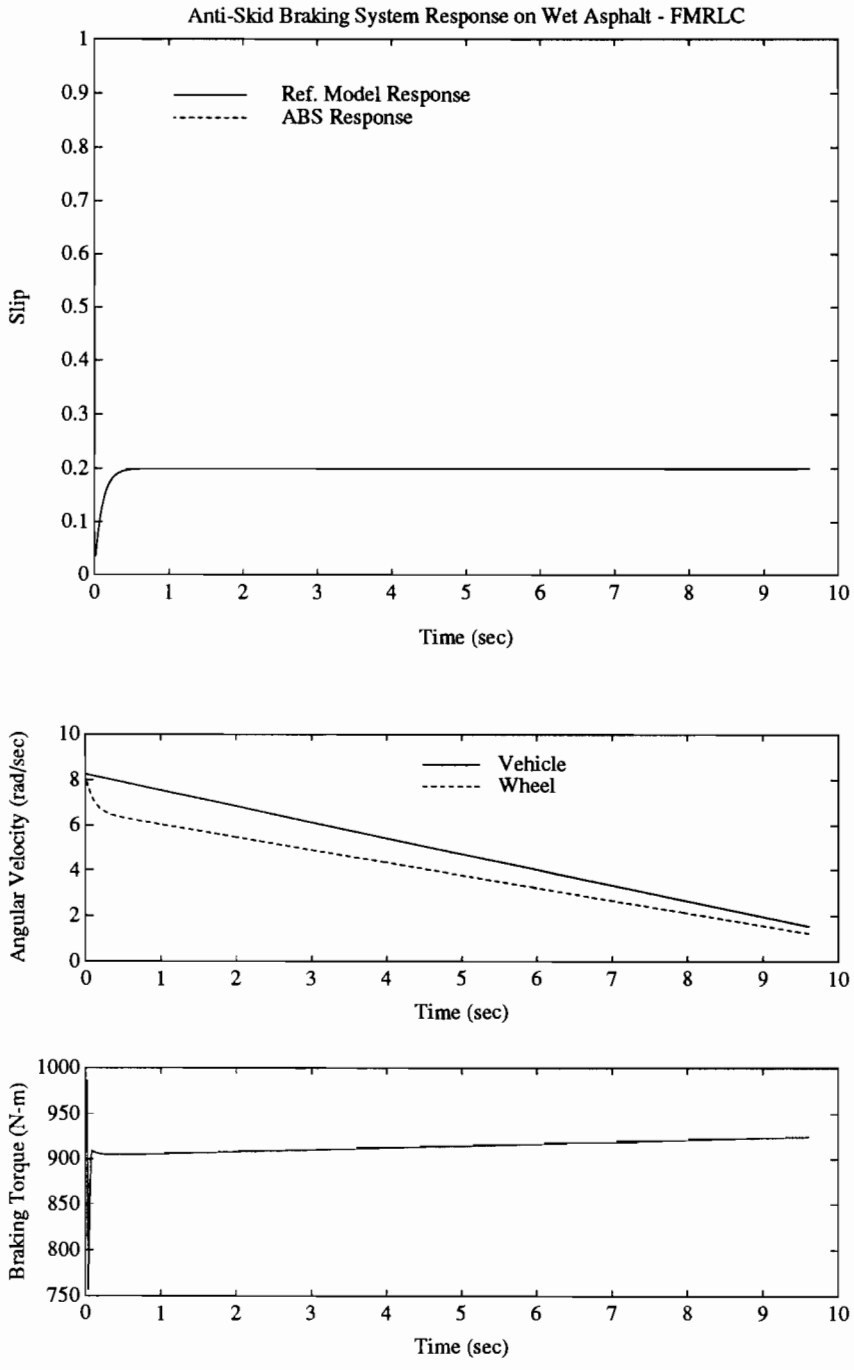


Figure 49: Simulation results for FMRLC of a vehicle braking system on a level wet asphalt surface.

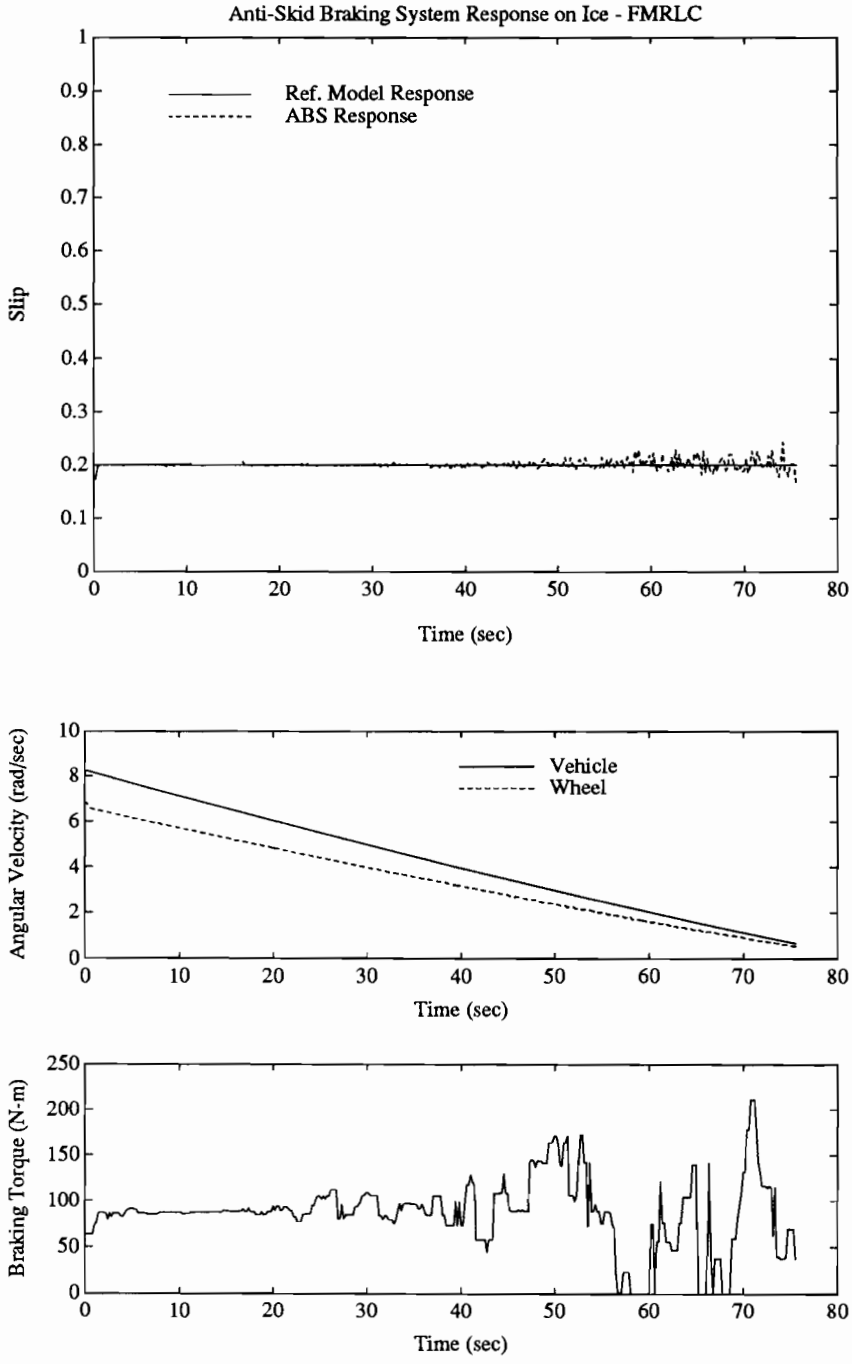


Figure 50: Simulation results for FMRLC of a vehicle braking system on a level icy surface.

speeds since the wheel slip becomes very sensitive at slow speeds. This fact is clearly illustrated by the simulation results shown in Figures 50 where "good" control of slip was achieved at the faster vehicle velocity and degrades slightly as the vehicle slows. Also, note the control signal becomes vary erratic at slower speed indicating a possible limit cycle.

Table 18 below is provided to illustrate the potential of the ABS system described above by comparing the stopping distance which resulted for the FMRLC algorithm with the case where the wheels are locked-up. This table was generated

Table 18: Stopping distance for an ABS system implemented using FMRLC Vs. a wheel lock-up situation.

	Stopping Distance (meters)	
Road Surface	**FMRLC**	**Lock-up**
Dry Asphalt	130.2611	153.4057
Wet Asphalt	147.5513	159.5036
Ice	990.0790	2714.800

based on a single braked wheel. Note that a substantial decrease in the stopping distance is obtained on all road surfaces which were considered above.

The next set of experiments was performed to illustrate the effectiveness of the FMRLC algorithm for split road conditions. Here we consider two very likely real world scenarios. The first involves the situation where the brakes are applied on wet asphalt and during the braking action the road surface suddenly becomes icy. Notice that during the initial braking action, the wet asphalt would allow for a relatively large braking torque without lock-up occurring. However, when the vehicle reaches

the icy road condition braking torque must be quickly reduced to prevent lockup. This large system variation provides very demanding controller modification on the FMRLC algorithm. However, the simulation results for this scenario shown below in Figure 51 illustrate that the FMRLC algorithm is capable of dealing with such drastic process variations.

The second case involves the reverse of the situation described above. In this case, the brakes are applied on a icy surface and during the braking action the road surface suddenly becomes wet asphalt. This situation would require the FMRLC to reconfigure itself to increase the torque when the vehicle reaches the wet asphalt. Figure 52 below illustrates the simulation result for this scenario. Once again the FMRLC was successful.

We must point out to the reader that the plots shown in Figure 51 and 52 do not represent all of the data which was obtained during numerical integration. Due to computer memory constraints it was necessary to reduce the data by considering only every n^{th} point. Unfortunately, a short-time spike in the slip plot, which occurs when the vehicle transitions between surfaces, is lost in both plots due to "aliasing" which results from data reduction.

The final experiment was designed to illustrate the effect that road inclination had on the FMRLC algorithm. Figure 53 shows the results for the FMRLC algorithm when the vehicle was braked while moving downhill on dry asphalt. For this simulation, the slope of the road, θ, was chosen to be -10 degrees. Notice that the performance of the FMRLC was comparable to those obtain for the non-inclined case shown in Figure 48. However, the stopping time and distance is increased significantly. This increase in the stopping distance is expected since the normal force N_v is reduced for inclined surfaces; thus reducing the friction between the road and

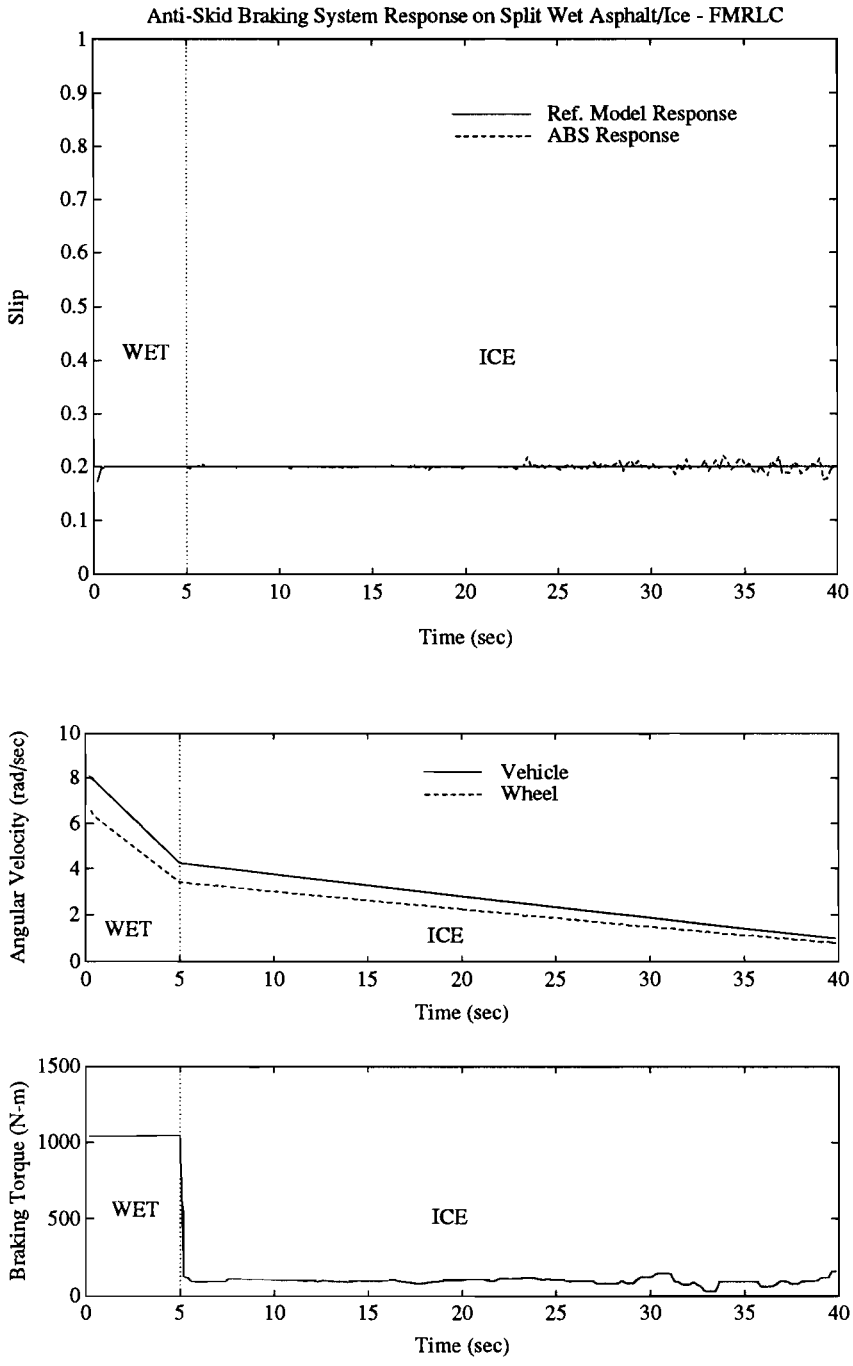


Figure 51: Simulation results for FMRLC of a vehicle braking system on a level split wet asphalt/icy surface.

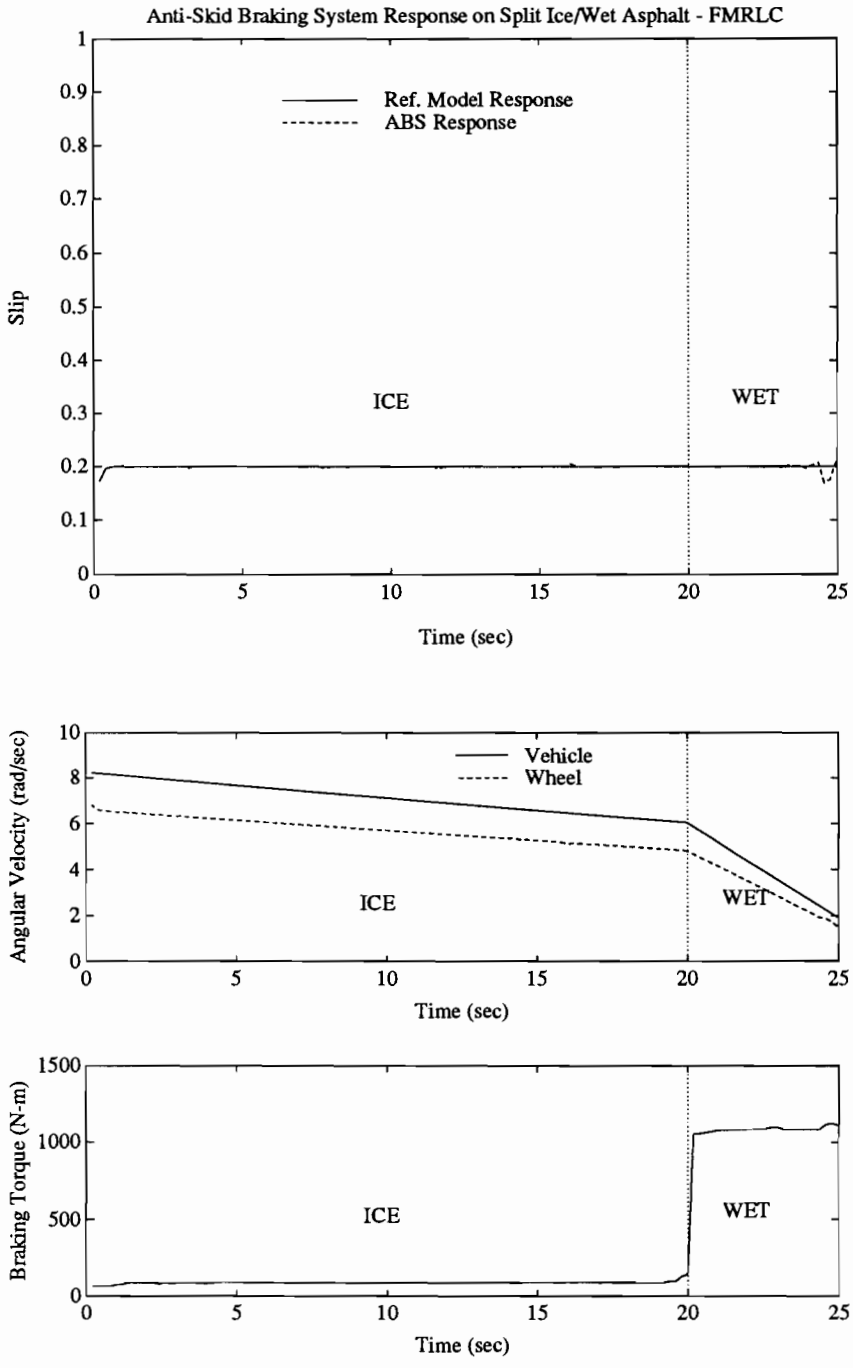


Figure 52: Simulation results for FMRLC of a vehicle braking system on a level split icy/wet asphalt surface.

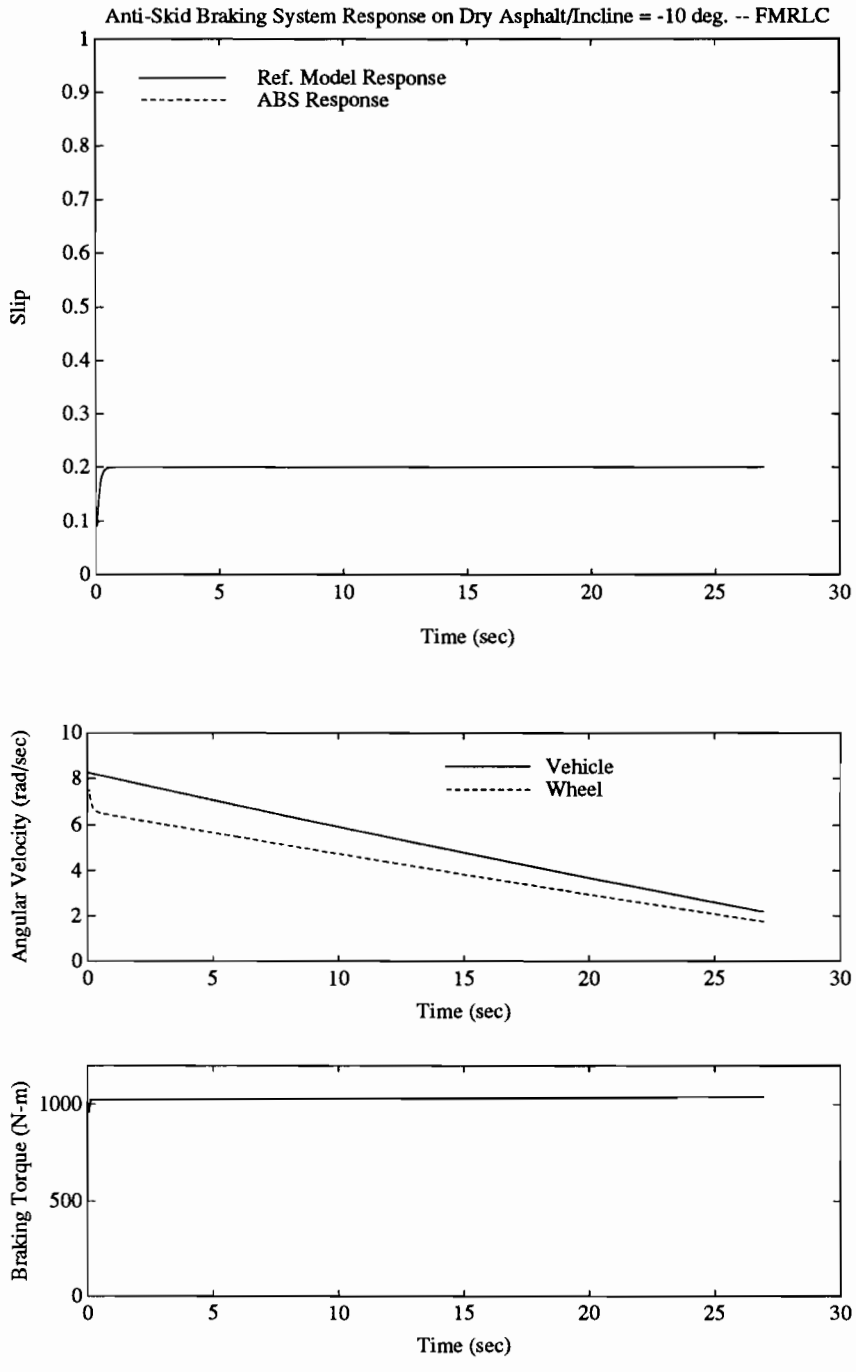


Figure 53: Simulation results for FMRLC of a vehicle braking system on a -10 deg. inclined dry asphalt surface.

the tire. Also, the force F_θ is increased thus providing an added force for which the braking system must overcome.

5.5 Chapter Summary

In this chapter we have illustrated the practical application of the FMRLC algorithm for a very complex real world applications involving slip control of a vehicle's braking system. This research involves three parts: 1) modeling of the process, 2) development of a slip estimation mechanism, and 3) FMRLC design.

Conventional ABS systems often exhibit limit cycle behavior which may be uncomfortable to the vehicle operator and sometimes creates excessive wear on the actuators. By estimating and directly controlling wheel slip it is hoped that the resulting ABS system exhibits "smoother" control characteristics and no limit cycle behavior. Furthermore, the behavior of a conventional braking system varies significantly for different road and operating conditions. Therefore, FMRLC is employed to continually update the control law to overcome these large process variations.

Simulation results for the ABS systems, which was implemented with the FMRLC algorithm, demonstrate the effectiveness of this control scheme for various road surfaces and slopes.

CHAPTER VI

Learning and Adaptive Autopilots for a Cargo Ship

In this chapter we design and evaluate the performance of the FMRLC algorithm when employed to control the directional heading of a cargo ship. Furthermore, since the FMRLC algorithm fits into the framework for direct adaptive control, the results obtained from applying the FMRLC to the cargo ship will be compared with conventional direct model reference adaptive control techniques. In particular we will employ both the gradient approach and the Lyapunov stability-theoretic approach to model reference adaptive control design such that the gains on a proportional derivative (PD) controller are adaptively adjusted to offset influences from plant variations or disturbances.

6.1 Problem Statement

Over recent years, to improve fuel efficiency and reduce wear on ship components, autopilot systems have been developed and implemented for controlling the directional heading of ships. Generally, the autopilots utilize simple control schemes such as PID control. Often, however, the capability for manual adjustments of the parameters of the controller is added to compensate for disturbances acting upon the ship such as wind and currents. Relatively good control can be achieved by manu-

ally adjusting the parameters of the controller. Once suitable controller parameters are found the controller will generally work well for small variations in the steering. However, for large variations the parameters of the autopilot must be continually modified.

Such continual adjustments are necessary because the dynamics of a ship vary with speed, trim, and loading. Also, it is useful to change the autopilot control law parameters when the ship is exposed to large disturbances which result from changes in the wind, waves, current, and water depth. Manual adjustment of the controller parameters is often a burden on the crew. Moreover, poor adjustment may result from human error. As a result, it is of great interest to have a method for automatically adjusting or modifying the controller.

6.1.1 Ship Dynamics

Generally, ship dynamics are obtained by applying Newton's laws of motion to the ship. For very large ships, the motion in the vertical plane may neglected since the "bobbing" or "bouncing" effects of the ship are small for large vessels. The motion of the ship is generally described by a coordinate system which is fixed to the ship. See Figure 54 below.

A simple model which describes the dynamical behavior of a ship is presented by Åström and Wittenmark in [3]. This model may be expressed by following differential equation:

$$\dddot{\psi}(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\ddot{\psi}(t) + \left(\frac{1}{\tau_1\,\tau_2}\right)\dot{\psi}(t) = \frac{K}{\tau_1\,\tau_2}\left(\tau_3\,\dot{\delta}(t) + \delta(t)\right), \tag{6.1}$$

where ψ is the heading of the ship and δ is the rudder angle. Assuming zero initial

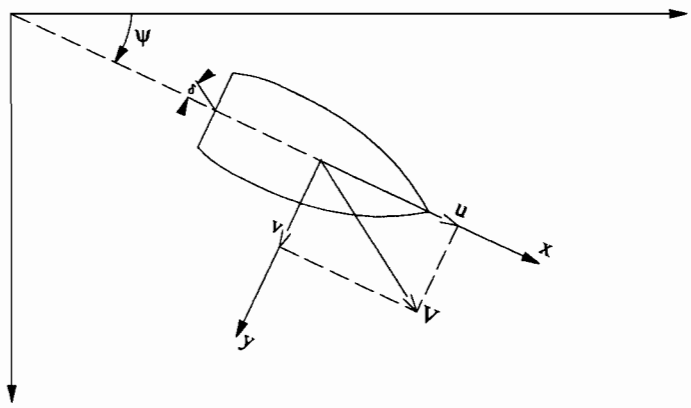


Figure 54: Coordinate system for a cargo ship.

conditions and using the Laplace transformation, Equation (6.1) can be written

$$\frac{\psi(s)}{\delta(s)} = \frac{K(s\tau_3 + 1)}{s(s\tau_1 + 1)(s\tau_2 + 1)}, \tag{6.2}$$

where K, τ_1, τ_2, and τ_3 are parameters which are function of the ship's constant forward velocity u and its length l as expressed below:

$$K = K_0 \left(\frac{u}{l}\right), \tag{6.3}$$

$$\tau_i = \tau_{i0} \left(\frac{l}{u}\right); \qquad i = 1, 2, 3. \tag{6.4}$$

In [3], Åström and Wittenmark present the values of these parameters for a mine sweeper, a cargo ship, and a tanker. For our simulations we will use the process parameters for a cargo ship which are presented below in Table 19. Also, we will assume that the ship is traveling in the x direction at a velocity of $5\frac{meters}{sec}$.

In normal steering, a ship often makes only small deviations from a straight line path. Therefore, Åström and Wittenmark obtain the model in Equation (6.1)

Table 19: Cargo ship parameters.

K_0	τ_{10}	τ_{20}	τ_{20}	l (meters)
-3.86	5.66	0.38	0.89	161

by linearizing the equations of motion around the zero rudder angle ($\delta = 0$). As a result, the rudder angle should not exceed approximately 5 degrees otherwise the model will be inaccurate. For our purposes, we need a model which is suited for rudder angles which are larger than 5 degrees; hence, we use the model proposed by Bech and Smitt in [7]. This extended model is given by

$$\dddot{\psi}\,(t) + \left(\frac{1}{\tau_1} + \frac{1}{\tau_2}\right)\ddot{\psi}(t) + \left(\frac{1}{\tau_1\,\tau_2}\right) H(\dot{\psi}(t)) = \frac{K}{\tau_1\,\tau_2}\left(\tau_3\,\dot{\delta}(t) + \delta(t)\right), \qquad (6.5)$$

where $H(\dot{\psi})$ is a nonlinear function of $\dot{\psi}(t)$. The function $H(\dot{\psi})$ can be found from the relationship between δ and $\dot{\psi}$ in steady state such that $\dddot{\psi} = \ddot{\psi} = \dot{\delta} = 0$. An experiment known as the "spiral test" has shown that $H(\dot{\psi})$ can be approximated by

$$H(\dot{\psi}) \approx a\,\dot{\psi}^3 + b\,\dot{\psi} \qquad (6.6)$$

where a and b are real valued constants such that a is always positive. For our simulations we choose the values of both a and b to be 1.

6.2 FMRLC Design

In this section we present a FMRLC algorithm for controlling the directional heading of a cargo ship. Only a very brief discussion of the FMRLC design is presented. For more details of the FMRLC algorithm, see Chapter III.

The inputs to the fuzzy controller are the heading error and change in heading error expressed as

$$e(kT) = \psi_r(kT) - \psi(kT), \tag{6.7}$$

$$c(kT) = \frac{e(kT) - e(kT-T)}{T}, \tag{6.8}$$

respectively, where $\psi_r(kT)$ is the desired ship heading. The controller output is the rudder angle, $\delta(kT)$, of the ship. Here we assume the dynamics of actuator which is used to position the rudder are much faster than the ship dynamics; thus they may be neglected. In this fuzzy controller design, 11 fuzzy sets are defined for each controller input such that the membership functions are "triangular shaped" and evenly distributed on the appropriate universes of discourse. The "normalizing" controller gains for the error, change error, and the controller output are chosen to be $g_e = \frac{1}{\pi}$, $g_c = 100$, and $g_u = \frac{8\pi}{18}$, respectively. The fuzzy sets for the fuzzy controller output are also assumed to be "triangular shaped" with a width of 0.4 on the "normalized" universe of discourse. The knowledge base array was initially chosen with all zero entries. The fuzzy controller sampling period was chosen to be $T = 50$ milliseconds.

The reference model for this process was arbitrarily chosen and is expressed by the following differential equation

$$\ddot{\psi}_m(t) + 0.1\ \dot{\psi}_m(t) + 0.0025\ \psi_m(t) = 0.0025\ \psi_r(t), \tag{6.9}$$

where $\psi_m(t)$ specifies the desired system performance for the ship heading $\psi(t)$.

The input to the fuzzy inverse model includes the error and change in error between the reference model and the ship heading expressed as

$$\psi_e(kT) = \psi_m(kT) - \psi(kT), \tag{6.10}$$

$$\psi_c(kT) = \frac{\psi_e(kT) - \psi_e(kT-T)}{T}, \tag{6.11}$$

respectively. For these inputs, 11 fuzzy sets are defined with "triangular shaped" membership functions which are evenly distributed on the appropriate universes of discourse. The "normalizing" controller gains associated with $\psi_e(kT)$, $\psi_c(kT)$, and $p_i(kT)$ are chosen to be $g_{\psi_e} = \frac{1}{\pi}$, $g_{\psi_c} = 5$, and $g_{p_i} = \frac{8\pi}{18}$, respectively. For a cargo ship, an increase in the rudder angle $\delta(kT)$, will generally result in a decrease in the ship heading angle. This implies that the incremental relationship between the process inputs and outputs is monotonically decreasing. Consequently, the knowledge base array shown below in Table 20 was employed for the fuzzy inverse model. In

Table 20: Knowledge base array table employed in the fuzzy inverse model for a cargo ship.

$P_i^{j,k}$		Ψ_c^k										
		-5	-4	-3	-2	-1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$
Ψ_e^j	-5	+1.0	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0
	-4	+1.0	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2
	-3	+1.0	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4
	-2	+1.0	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6
	-1	+1.0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8
	0	+1.0	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0
	+1	+0.8	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0
	+2	+0.6	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0
	+3	+0.4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0
	+4	+0.2	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0
	+5	0.0	−0.2	−0.4	−0.6	−0.8	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0

Table 20, Ψ_e^j denotes the j^{th} fuzzy set associated with the error signal ψ_e and Ψ_c^k denotes the k^{th} fuzzy set associated with the change in error signal ψ_c.

6.3 Model Reference Adaptive Control

In this section we present a model reference adaptive control (MRAC) design which utilizes an underlying proportional derivative (PD) control law for the direct controller. The PD control law is used to obtain a fair comparison with FMRLC algorithm where error and change in error are employed as controller inputs. We will consider both the gradient method and the Lyapunov stability method MRAC design.

As shown in Figure 55, the framework of MRAC involves four basic components:

1. Controlled process,
2. Conventional direct controller,
3. Reference model,
4. Controller parameter adjustment mechanism.

The system employs an ordinary feedback loop which is composed of the process and the controller. In general, the controller may be any mapping between the controller inputs, often the process error $\underline{e}$ and functions of the process error, and the process input $\underline{u}$ such that the mapping can be explicitly expressed mathematically. Another loop is created by the reference model and the parameter adjustment mechanism to provide performance feedback. The reference model is a dynamical system whose behavior meets the desired design characteristics for the process. The controller parameter adjustment mechanism simply performs the function of updating the controller parameters so that the system behaves like the reference model. In general, this is achieved by operating on the error $\underline{y}_e$ between the reference model output $\underline{y}_m$ and the process output $\underline{y}$ to generate the controller parameters $\underline{\theta}$.

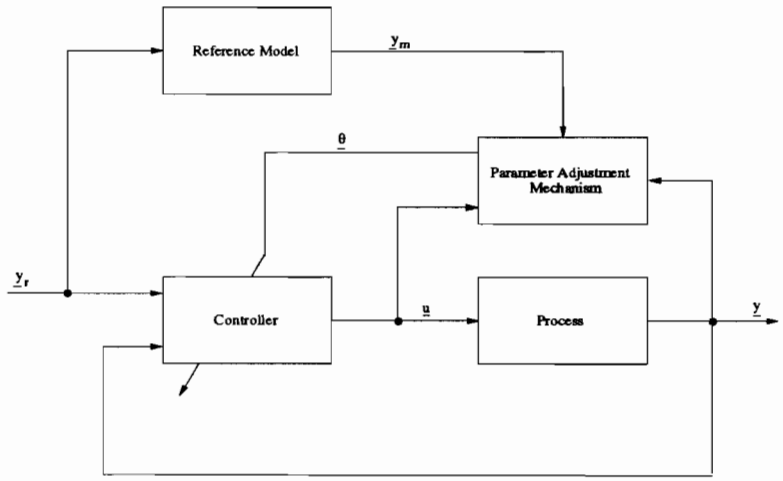


Figure 55: Block diagram for the MRAC algorithm.

6.3.1 Gradient Approach

The controller parameter adjustment mechanism for the gradient approach to MRAC can be implemented via the *MIT rule*. Generally, this approach involves defining a cost function, denote $J(\underline{\theta})$, which must be a function of the controller parameters $\underline{\theta}$. For example, often a suitable choice for a cost function for a single output system may be expressed by

$$J(\underline{\theta}) = \frac{1}{2} y_e^2(t), \tag{6.12}$$

where $y_e(t)$ is the error defined as

$$y_e(t) = y_m(t) - y(t). \tag{6.13}$$

The objective for the parameter adjustment mechanism is to minimize the cost criteria $J(\underline{\theta})$. To achieve this, it is reasonable to change the parameters in the direction

of the negative gradient of J, i.e.,

$$\frac{d\underline{\theta}}{dt} = -\gamma \frac{\partial J}{\partial \underline{\theta}}, \tag{6.14}$$

For the cost criteria given in Equation (6.12), the negative gradient expressed in Equation (6.14) becomes

$$\frac{d\underline{\theta}}{dt} = -\gamma \, y_e(t) \frac{\partial y_e(t)}{\partial \underline{\theta}}, \tag{6.15}$$

which is commonly referred to as the MIT rule. Also the partial derivative $\frac{\partial y_e(t)}{\partial \underline{\theta}}$ is sometimes referred to as the *sensitivity derivative*. Note that Equation (6.15) represents n differential equations where n is the number of controller parameters. To change the parameters in the direction of the negative gradient it is not necessary that γ be the same for all equations.

Next we show how the MIT rule may be implemented for the ship and a PD controller. The PD controller is used for fair comparison with the FMRLC algorithm where error and change in error are process inputs. A similar design for a ship is presented by Amerongen and Cate in [1].

PD Controller Design

For implementing the MIT rule on the ship we assume that the ship may be modeled by a linear second order differential equation. This model is obtained by eliminating the process pole resulting from τ_2 in Equation (6.1) since its associated dynamics are significantly faster than those resulting from τ_1. Also, for small heading variations the rudder angle derivative $\dot{\delta}$ is likely to be small and may be neglected. Therefore we obtain the following reduced model for the ship

$$\ddot{\psi}(t) + \left(\frac{1}{\tau_1}\right) \dot{\psi}(t) = \left(\frac{K}{\tau_1}\right) \delta(t) \tag{6.16}$$

The PD control law which will be employed for this process may be expressed by

$$\delta = k_p \left(\psi_r(t) - \psi(t)\right) - k_d \dot{\psi}(t) \tag{6.17}$$

where k_p and k_d are the proportional and derivative gains, respectively and $\psi_r(t)$ is the desired process output. Substituting Equation (6.17) into Equation (6.16) we obtain

$$\ddot{\psi}(t) + \left(\frac{1 + K\ k_d}{\tau_1}\right)\dot{\psi}(t) + \left(\frac{K\ k_p}{\tau_1}\right)\psi(t) = \left(\frac{K\ k_p}{\tau_1}\right)\psi_r(t). \tag{6.18}$$

It follows from Equation (6.18) that

$$\psi(t) = \frac{\frac{K\ k_p}{\tau_1}}{p^2 + \left(\frac{1+K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right)}\psi_r(t) \tag{6.19}$$

where p is the differential operator.

Reference Model Design

The reference model for this process may be expressed by the following generic differential equation for a second order linear process

$$\ddot{\psi}_m(t) + 2\zeta\omega_n\dot{\psi}_m(t) + \omega_n^2\psi_m(t) = \omega_n^2\psi_r(t) \tag{6.20}$$

where ζ is the process damping and ω_n is the natural frequency. By employing the differential operator p, it follows directly from Equation (6.20) that

$$\psi_m(t) = \frac{\omega_n}{p^2 + 2\zeta\omega_n\ p + \omega_n^2}\psi_r(t). \tag{6.21}$$

In order to be consistent with the FMRLC design we choose $\zeta = 1$ and $\omega_n = 0.05$.

To apply the MIT rule to the compensated cargo ship model, we introduce the reference model error

$$\psi_e(t) = \psi_m(t) - \psi(t) \tag{6.22}$$

where $\psi(t)$ now denotes the closed-loop output. Substituting Equations (6.21) and (6.19) into Equation (6.22) we obtain

$$\psi_e(t) = \left(\frac{\omega_n}{p^2 + 2\zeta\omega_n\ p + \omega_n^2} - \frac{\frac{K\ k_p}{\tau_1}}{p^2 + \left(\frac{1+K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right)} \right) \psi_r(t) \qquad (6.23)$$

Sensitivity Model and Parameter Update Laws

The sensitivity derivatives are obtained by taking the partial derivatives of Equation (6.23) with respect to the proportional gain k_p and the derivative gain k_d. Therefore the *sensitivity model* for k_p may be obtained by as:

$$\frac{\partial\ \psi_e}{\partial k_p} = \left(\frac{\frac{K}{\tau_1}}{p^2 + \left(\frac{1+K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right)} \right) (\psi - \psi_r). \qquad (6.24)$$

Likewise, the sensitivity model for k_d is

$$\frac{\partial\ \psi_e}{\partial k_d} = \left(\frac{\frac{K}{\tau_1}\ p}{p^2 + \left(\frac{1+K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right)} \right) \psi. \qquad (6.25)$$

In general, Equations (6.24) and (6.25) cannot be used because the controller parameters k_p and k_d are not known. This implies that some approximation is needed to obtain a realizable parameter adjustment rule. To determine this approximation we observe that for the "optimal values" of k_p and k_d we have

$$p^2 + \left(\frac{1 + K\ k_d}{\tau_1}\right)\ p + \left(\frac{K\ k_p}{\tau_1}\right) = p^2 + 2\zeta\omega_n\ p + \omega_n^2. \qquad (6.26)$$

Furthermore, the term $\frac{K}{\tau_1}$ may be absorbed into the adaptation gain γ. However, this requires that the sign of $\frac{K}{\tau_1}$ be known since, in general, γ should be positive to ensure that the controller updates are made in the direction of the negative gradient. For a forward moving cargo ship the sign of $\frac{K}{\tau_1}$ happens to be negative which implies

that the term γ with $\frac{K}{\tau_1}$ absorbed into it must be negative to achieve the appropriate negative gradient. After making the above approximations we obtain the following differential equations for updating the PD controller gains:

$$\frac{d\ k_p}{dt} = -\gamma_1 \left(\frac{1}{p^2 + 2\zeta\omega_n p + \omega_n^2} \ (\psi - \psi_r) \right) \psi_e, \tag{6.27}$$

$$\frac{d\ k_d}{dt} = -\gamma_2 \left(\frac{p}{p^2 + 2\zeta\omega_n p + \omega_n^2} \ \psi \right) \psi_e. \tag{6.28}$$

where γ_1 and γ_2 are negative real numbers.

Figure 56 below illustrates the complete gradient MRAC design for the cargo ship. For this design, suitable values for γ_1 and γ_2 were found to be -0.005 and -0.1, respectively.

6.3.2 Lyapunov Approach

Next we present a Lyapunov MRAC design for controlling the heading of a cargo ship. The controller parameter adjustment mechanism for the Lyapunov approach to MRAC is usually called such because it is based on Lyapunov's *Direct method.* This method provides a sufficient condition for global stability and is base on the following theorem.

Theorem 6.1 (Lyapunov Global Stability) *Assume that there exists a scalar function $V(\underline{x})$ of the state $\underline{x}$, with continuous first order derivatives such that*

- *$V(\underline{x})$ is positive definite*
- *$\dot{V}(\underline{x})$ is negative definite*
- *$V(\underline{x}) \to \infty$ as $||\underline{x}|| \to 0$*

then the equilibrium point at the origin of the space of $\underline{x}$ is globally asymptotically stable.

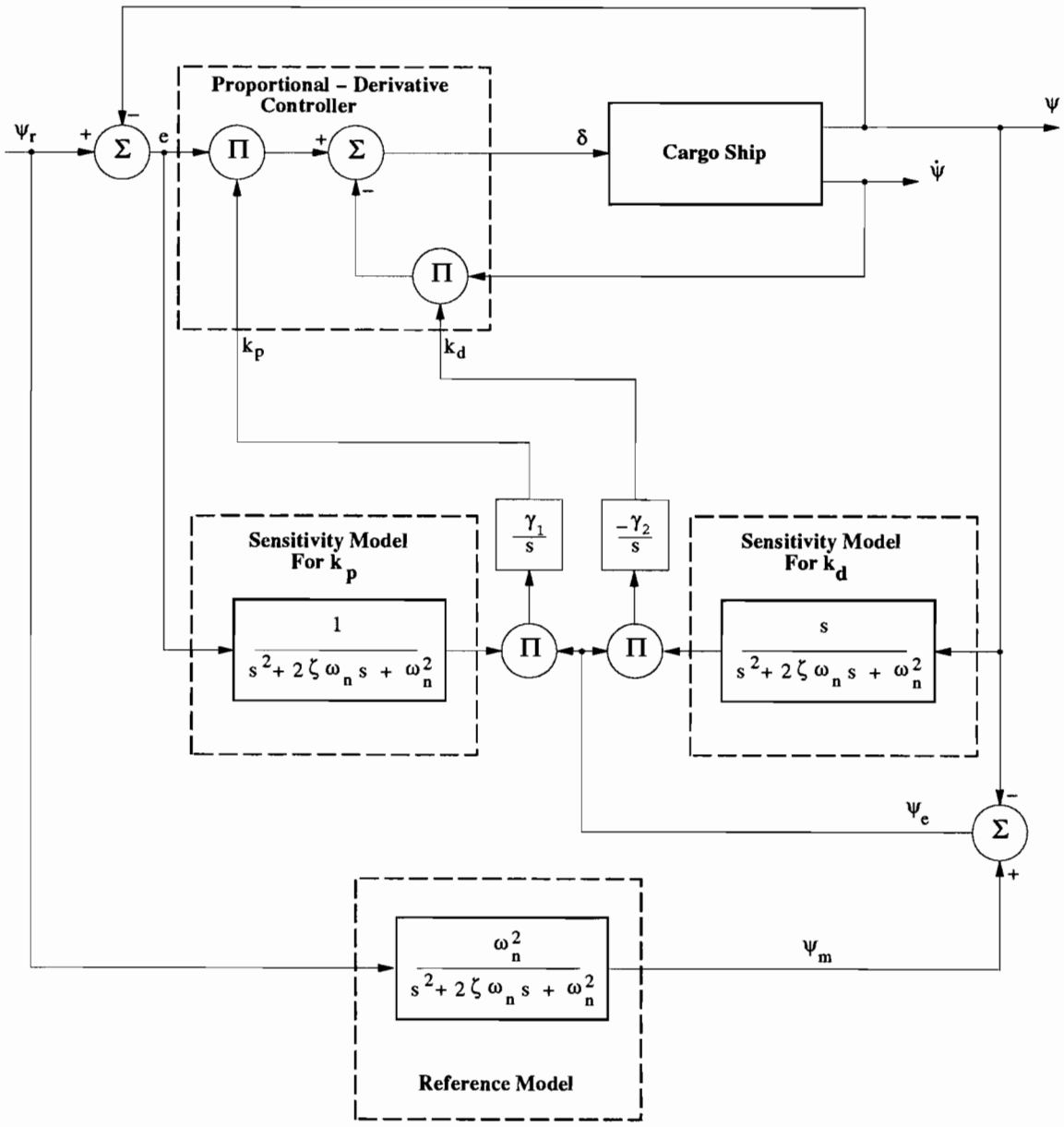


Figure 56: Block diagram for the gradient approach to MRAC for a cargo ship.

The proof for Theorem 6.1 is presented in [68]. Consider next how this theorem may be used to derive a stable adjustment mechanism in the MRAC framework.

Consider a multiple-input multiple output linear time varying system whose states are accessible. Assume the process is n^{th} order with r inputs and is described by the following vector differential equation

$$\dot{\underline{x}} = A(t)\underline{x} + B(t)\underline{u}. \tag{6.29}$$

where the elements of matrix $A \in \Re^{n\times n}$ and $B \in \Re^{n\times r}$ are generally slowly time varying and unknown at any instance in time. Furthermore, assume that a linear full state feedback control law is employed such that

$$\underline{u} = -G(t)(\underline{x} + \underline{x}_r). \tag{6.30}$$

where $\underline{x}_r$ is a vector of desired states and $G(t)$ is a matrix whose elements may be specified in time. Substituting Equation (6.30) into Equation (6.29) results in the following differential equation for the compensated process

$$\dot{\underline{x}} = A_c(t)\underline{x} + B_c(t)\underline{x}_r \tag{6.31}$$

where

$$A_c(t) = A(t) - B(t)G(t), \tag{6.32}$$

$$B_c(t) = -B(t)G(t). \tag{6.33}$$

Next we assume that the reference model is linear, time-invariant, and asymptotically stable such that it may be expressed by the following differential equation

$$\dot{\underline{x}}_m = A_m(t)\underline{x}_m + B_e(t)\underline{x}_r. \tag{6.34}$$

where $A_m \in \Re^{n\times n}$. If the state error and the parameter error are defined as

$$\underline{x}_e \triangleq \underline{x}_m - \underline{x}, \tag{6.35}$$

$$\Phi(t) \triangleq A_m(t) - A_c(t), \tag{6.36}$$

$$\Upsilon(t) \triangleq B_m(t) - B_c(t), \tag{6.37}$$

then the dynamical equation which describes the error may be given by

$$\dot{\underline{x}}_e = A_m(t)\underline{x}_e + \Phi(t)\underline{x} + \Upsilon(t)\underline{x}_r \tag{6.38}$$

If we choose the adaptation laws to be

$$\dot{A}_c(t) = -\dot{\Phi}(t) = \gamma P \underline{x}_e \underline{x}, \tag{6.39}$$

$$\dot{B}_c(t) = -\dot{\Upsilon}(t) = \gamma P \underline{x}_e \underline{x}, \tag{6.40}$$

where $P \in \Re^{n\times n}$ is a symmetric positive definite matrix and $\gamma \in \Re$ is a positive scalar adaptation gain, then the equilibrium state ($\underline{x}_e = 0, \Phi = 0, \Upsilon = 0$) in Equation (6.38) is asymptotically stable in the large and $\lim_{t\to\infty} x_e(t) = 0$.

Next we use Lyapunov's Direct Method to prove that the adaptation laws in Equations (6.39) and (6.40) result in stability for $\underline{x}_e$ in Equation (6.38). Consider the Lyapunov candidate function $V(\underline{x}_e, \Phi, \Upsilon)$ of the form

$$V(\underline{x}_e, \Phi, \Upsilon) = \gamma \underline{x}_e^T P \underline{x}_e + tr(\Phi^T \Phi) + tr(\Upsilon^T \Upsilon) \tag{6.41}$$

where P is a positive definite matrix, γ is a positive scalar, and tr(A) denotes the trace of matrix A. Taking the time derivative of Equation (6.41) yields

$$\dot{V}(\underline{x}_e, \Phi, \Upsilon) = \gamma \dot{\underline{x}}_e^T P \underline{x}_e + \gamma \underline{x}_e^T P \dot{\underline{x}}_e + tr(\dot{\Phi}^T \Phi + \Phi^T \dot{\Phi}) + tr(\dot{\Upsilon}^T \Upsilon + \Upsilon^T \dot{\Upsilon}). \tag{6.42}$$

Recall the following properties of the trace operation from basic matrix theory

$$\begin{aligned} tr(A+B) &= tr(A)+tr(B); && A,B \in \Re^{n\times n}, \\ tr(A^T) &= tr(A); && A,B \in \Re^{n\times n}, \\ tr(AB) &= tr(BA); && A \in \Re^{n\times m}, B \in \Re^{m\times n}, \\ tr(\alpha A) &= \alpha\, tr(A); && \alpha \in \Re, A \in \Re^{n\times n}. \end{aligned}$$

Further recall the following property of the transpose of matrices

$$(A^T B)^T = B^T A.$$

Using the above matrix relation and properties of the trace operation, It is easily shown that Equation (6.42) reduces to

$$\dot{V}(\underline{x}_e, \Phi, \Upsilon) = \gamma(\dot{\underline{x}}_e^T P \underline{x}_e + \underline{x}_e^T P \dot{\underline{x}}_e) + 2tr(\dot{\Phi}^T \Phi) + 2tr(\dot{\Upsilon}^T \Upsilon) \tag{6.43}$$

Solving Equation (6.43) along the trajectories of Equation (6.38) we obtain

$$\begin{aligned} \dot{V}(\underline{x}_e, \Phi, \Upsilon) = \gamma \underline{x}_e^T (A_m^T P + P A_m) \underline{x}_e + 2\gamma \underline{x}_e^T \Phi^T P \underline{x} + 2tr(\dot{\Phi}^T \Phi) + \\ 2\gamma \underline{x}_e^T \Upsilon^T P \underline{x}_r + 2tr(\dot{\Upsilon}^T \Upsilon). \end{aligned} \tag{6.44}$$

Since A_m is an asymptotically stable matrix, there exists a solution to the Lyapunov matrix equation

$$A_m^T P + P A_m = -Q \tag{6.45}$$

such that for any symmetric positive definite matrix $Q \in \Re^{n\times n}$ the resulting matrix $P \in \Re^{n\times n}$ is also symmetric positive definite. Using this fact, Equation (6.44) reduces to

$$\begin{aligned} \dot{V}(\underline{x}_e, \Phi, \Upsilon) = -\gamma \underline{x}_e^T Q \underline{x}_e + 2\gamma \underline{x}_e^T \Phi^T P \underline{x} + 2tr(\dot{\Phi}^T \Phi) + \\ 2\gamma \underline{x}_e^T \Upsilon^T P \underline{x}_r + 2tr(\dot{\Upsilon}^T \Upsilon). \end{aligned} \tag{6.46}$$

Substituting the adaptation laws in Equations (6.39) and (6.40) into Equation (6.46) yields

$$\dot{V}(\underline{x}_e, \Phi, \Upsilon) = -\gamma \dot{\underline{x}}_e^T Q \dot{\underline{x}}_e \leq 0. \tag{6.47}$$

Hence, the equilibrium state the equilibrium state ($\underline{x}_e = 0, \Phi = 0, \Upsilon = 0$) is stable and $\underline{x}_e$ is bounded which implies that

$$\lim_{t \to \infty} \underline{x}_e(t) = 0. \tag{6.48}$$

Next we show how the Lyapunov approach to MRAC may be implemented for cargo ship and a PD controller. Once again, the PD controller is used for fair comparison with the FMRLC algorithm where error and change in error are process inputs.

Lyapunov MRAC Design for a Cargo Ship

Examples of Lyapunov based MRAC designs are illustrated by Narendra and Annaswamy in [55] and by Amerongen and Cate in [1]. Rather than designing our own Lyapunov based MRAC we will present the design by Narendra and Annaswamy for comparison with the other methods described above.

Recall from Section 6.3.1 that the ship dynamics may be approximated by a second order linear time-invariant differential equation expressed as

$$\ddot{\psi}(t) + \left(\frac{1}{\tau_1}\right) \dot{\psi}(t) = \left(\frac{K}{\tau_1}\right) \delta(t) \tag{6.49}$$

Once again, the PD control law which will be employed for this process may be expressed by

$$\delta = k_p \left(\psi_r(t) - \psi(t)\right) - k_d \dot{\psi}(t) \tag{6.50}$$

where k_p and k_d are the proportional and derivative gains, respectively and $\psi_r(t)$ is the desired process output. The dynamical equation which describes the compensated system is expressed

$$\dot{\underline{\psi}} = A_c\underline{\psi} + B_c\psi_r \tag{6.51}$$

where $\underline{\psi} = [\psi \ \dot{\psi}]^T$ and

$$A_c = \begin{bmatrix} 0 & 1 \\ -\frac{K\ k_p}{\tau_1} & -\frac{(1+K\ k_d)}{\tau_1} \end{bmatrix} \tag{6.52}$$

$$B_c = \begin{bmatrix} 0 \\ \frac{K\ k_p}{\tau_1} \end{bmatrix}. \tag{6.53}$$

The reference model for this process may be expressed by the following differential equation

$$\dot{\underline{\psi}}_m = A_m\underline{\psi}_m + B_m\underline{\psi}_r \tag{6.54}$$

where $\underline{\psi}_m = [\psi_m \ \dot{\psi}_m]^T$

$$A_m = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \tag{6.55}$$

$$B_m = \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix}. \tag{6.56}$$

and where ζ is the process damping ration and ω_n is the natural frequency. To be consistent with the FMRLC design we choose $\zeta = 1$ and $\omega_n = 0.05$.

In order to apply the Lyapunov approach to MRAC design, we introduce the reference model error

$$\underline{\psi}_e(t) = \underline{\psi}_m(t) - \underline{\psi}(t) \tag{6.57}$$

where $\underline{\psi}(t)$ now denotes the closed-loop process output. The dynamical equation which describes the error may be expressed by

$$\dot{\underline{\psi}}_e = A_m(t)\underline{\psi}_e + \Phi(t)\underline{\psi} + \Upsilon(t)\underline{\psi}_r \tag{6.58}$$

Recall that the equilibrium point $\underline{\psi}_e = 0$ in Equation (6.58) is asymptotically stable if we choose the adaptation laws to be

$$\dot{A}_c(t) = \gamma P \underline{\psi}_e \underline{\psi}^T, \tag{6.59}$$

$$\dot{B}_c(t) = \gamma P \underline{\psi}_e \underline{\psi}_r^T, \tag{6.60}$$

where $P \in \Re^{n \times n}$ is a symmetric positive definite matrix which is a solution of the Lyapunov equation

$$A_m^T P + P A_m = -Q \ < 0. \tag{6.61}$$

If we assume that Q is a 2×2 identity matrix then we may solve for P. Upon doing this we obtain

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} = \begin{bmatrix} 25.0125 & 200.000 \\ 200.000 & 2005.00 \end{bmatrix}. \tag{6.62}$$

Solving for $\dot{k}_p$ and $\dot{k}_d$ in Equations (6.59) and (6.60), respectively, the adaptation law in Equations (6.59) and (6.60) may be implemented as

$$\dot{k}_p = -\gamma_1(p_{21}\psi_e + p_{22}\dot{\psi}_e)(\psi - \psi_r), \tag{6.63}$$

$$\dot{k}_d = -\gamma_2(p_{21}\psi_e + p_{22}\dot{\psi}_e)\dot{\psi}. \tag{6.64}$$

Of course, Equations (6.63) and (6.64) assume that the plant parameters and disturbance are varying slowly. The reader should note that different values of γ are employed for each parameter update law. However, Narendra and Annaswamy

show that this does not upset the stability of the system. Also, in obtaining Equations (6.63) and (6.64) the term $\frac{K}{\tau_1}$ was absorbed into the adaptation gains γ_1 and γ_2. Recall that for the cargo ship $\frac{K}{\tau_1}$ happens to be a negative quantity. Therefore, both γ_1 and γ_2 must become negative to compensate for this fact. See Narendra and Annaswamy [55] for more details about obtaining Equations (6.63) and (6.64).

Figure 57 below illustrates the complete Lyapunov MRAC design for the cargo ship. For this design, suitable values for γ_1 and γ_2 were found to be -0.005 and -0.1, respectively.

6.4 Simulation Results

All three adaptive control methods presented above (FMRLC, gradient MRAC, and Lyapunov MRAC) have been implemented and simulated using the Fortran programming language. In this section we present the results which were obtained from these simulations. We must point out that these simulations were performed for the non-linear process model given in Equation (6.5) to emulate using the true ship dynamics.

Figure 58 shows the results for the FMRLC controller. Recall that the controller was initially loaded with all zero entries (i.e. an empty rule base). Note in Figure 58 that the FMRLC algorithm was very successful in generating the appropriate control rules for a "good" process response since the reference model and the ship heading track almost perfectly. In fact the maximum deviation between the two signals was observed to be less than 1 degree. As a result, the system exhibits a fast transient response with no overshoot. Also note that the rate of convergence for the FMRLC algorithm was very fast. In the last plot of Figure 58 two large spikes of a magnitude of about 1 degree occur after the first two step input changes. However, as time

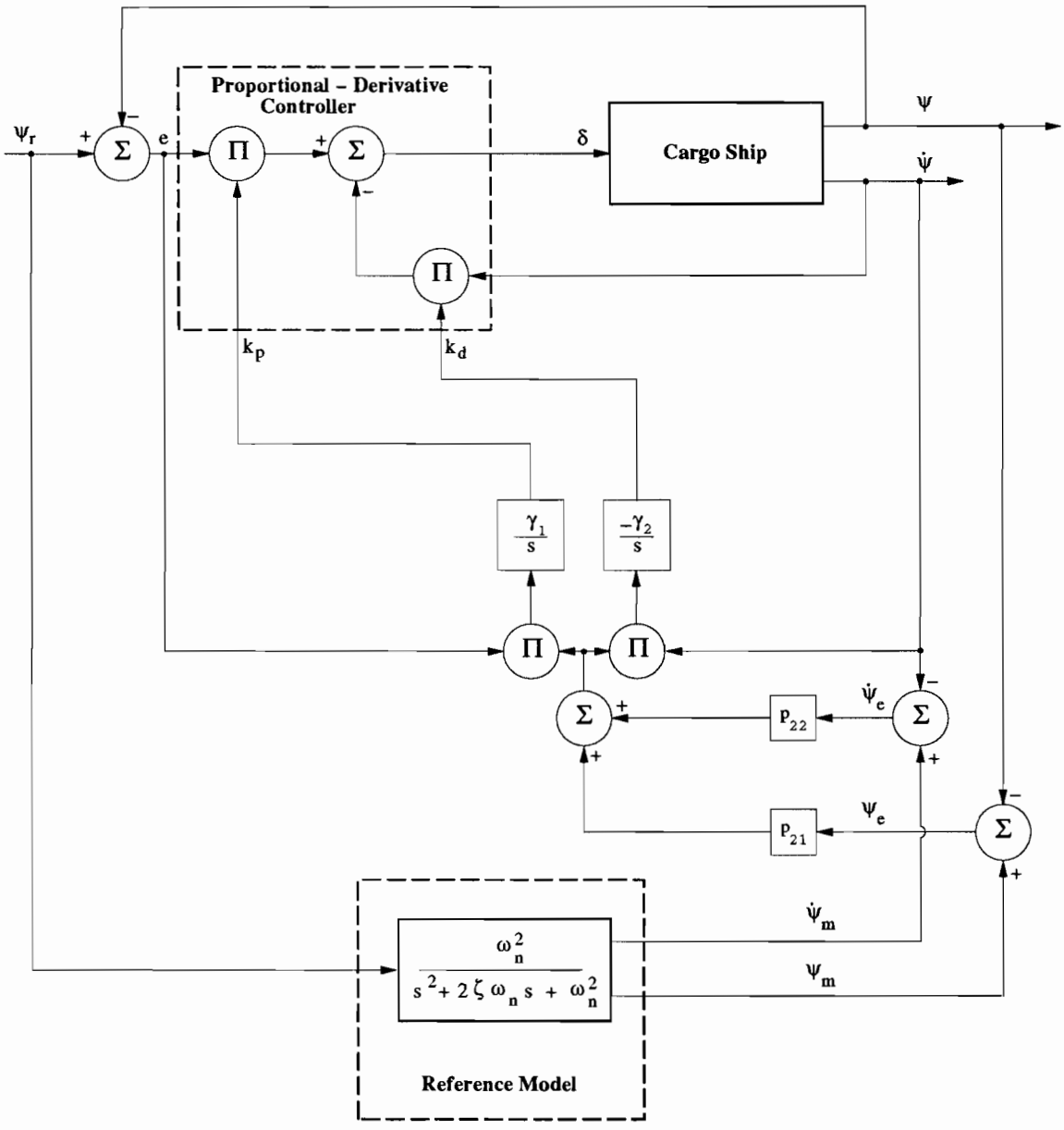


Figure 57: Block diagram for the Lyapunov approach to MRAC for a cargo ship.

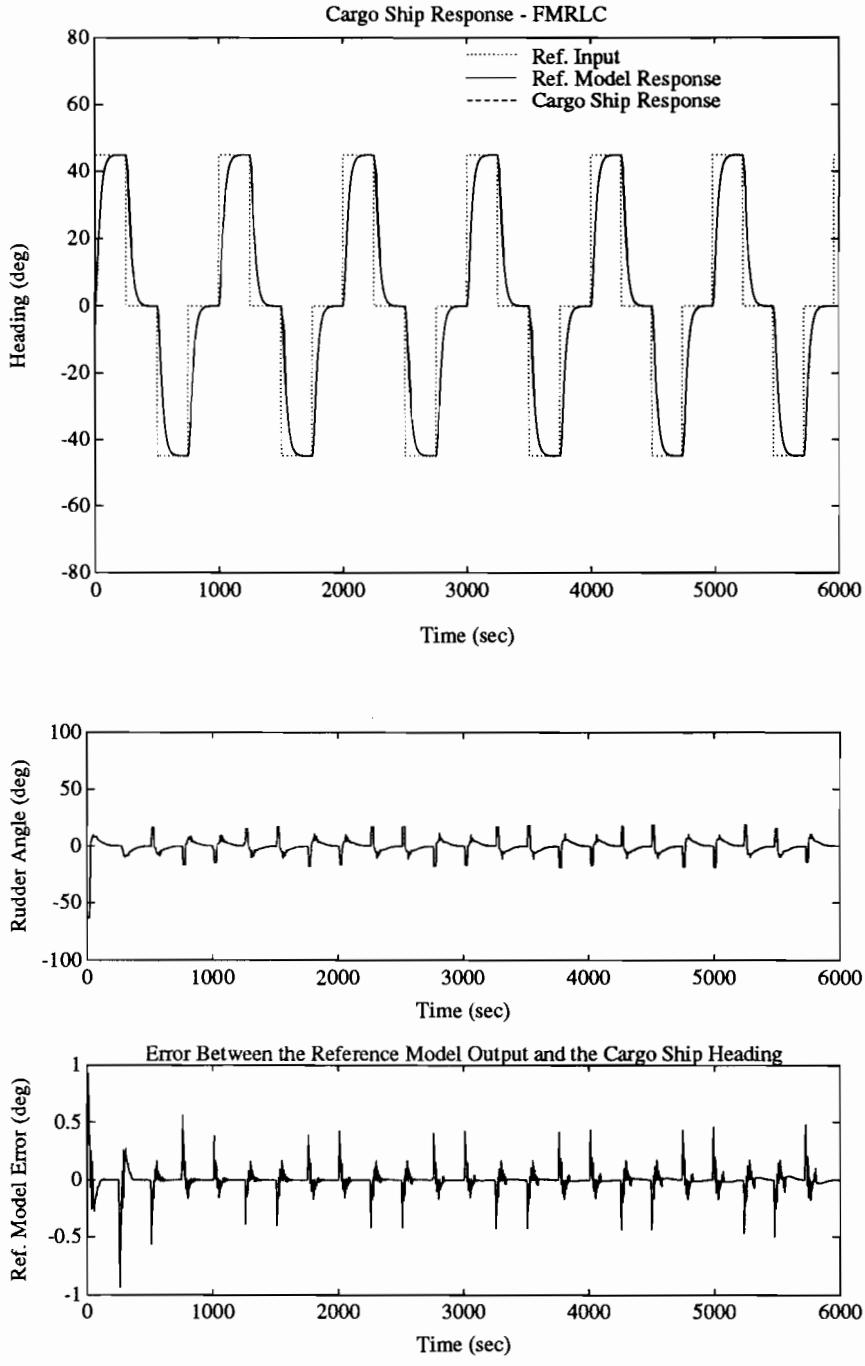


Figure 58: Simulation results for the FMRLC algorithm when employed for a cargo ship.

progresses the spikes resulting from subsequent step input changes are reduced to less than 0.5 degrees as a result of learning.

Compare the results for the FMRLC with those obtained for the gradient and the Lyapunov approach to MRAC shown in Figure 59 and 60, respectively. The controller gains k_p and k_d for both MRAC algorithms where initially chosen to be 5. This choice of initial controller gains happens to be an unstable case for the linearized second order process model. However, we felt this to be a fair comparison since the fuzzy controller is initially chosen with no control rules. We would have chosen both controller gains to be 0; but, this choice resulted in a very slow convergence rate for the MRAC. It is easy to show that the compensated second order process model in Equation (6.51) is equal to the reference model in Equation (6.54) if the controller gains are chose to be $k_p = -3.8$ and $k_d = -143.7$. In Table 21 below, we summarize the final values of the process gains shown in Figures 59 and 60. Although the process

Table 21: Final values of controller gains k_p and k_d in the simulation results for the the gradient and Lyapunov approach to MRAC.

	MRAC Approach	
Controller Parameter	**Gradient**	**Lyapunov**
$k_p(6000)$	-4.7752	-3.0974
$k_d(6000)$	-171.8580	-105.0242

gains for both algorithms converged to values relatively close to the optimal values, they do not match exactly. This reason for this may be explained by a few simple facts:

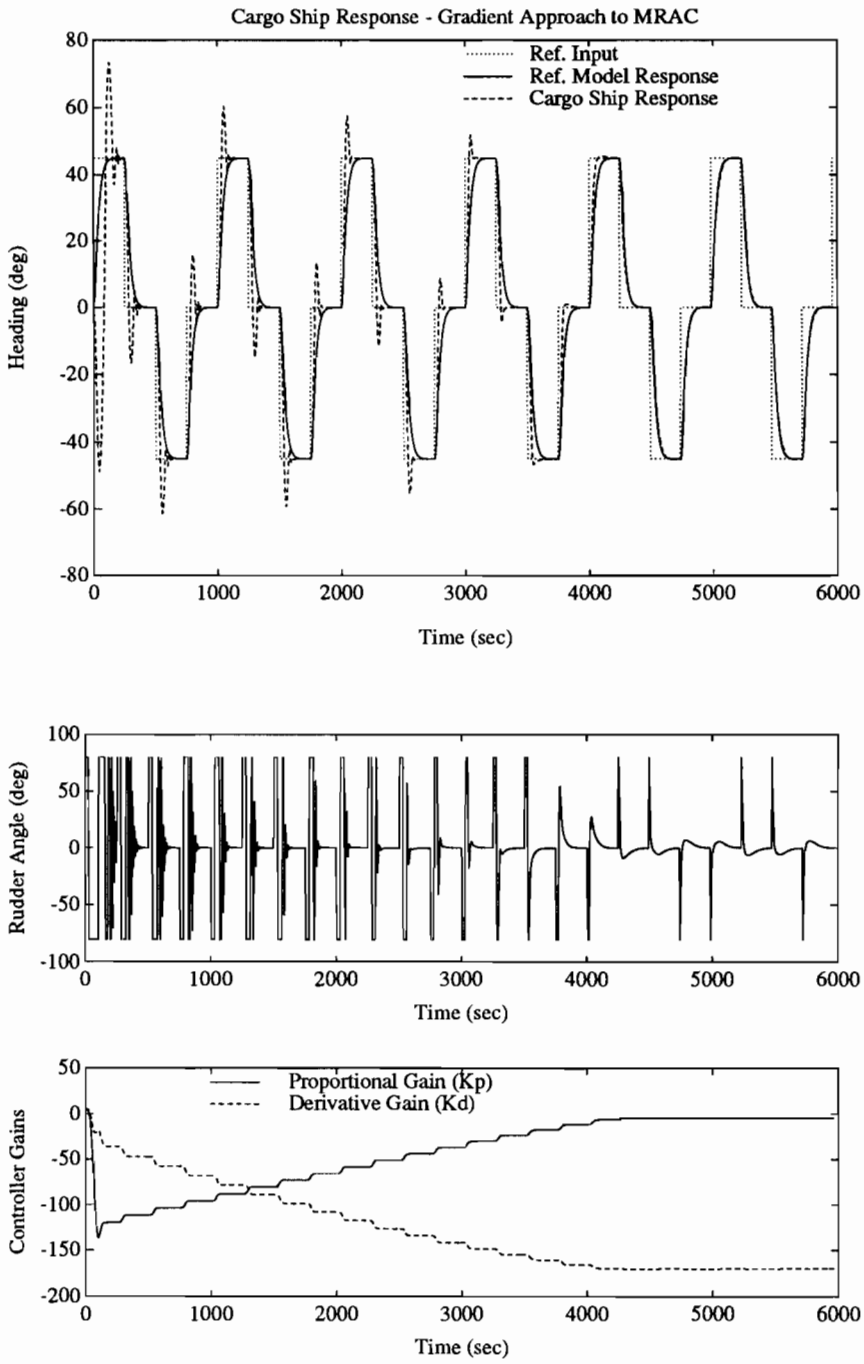


Figure 59: Simulation results for the gradient approach to MRAC when employed for a cargo ship.

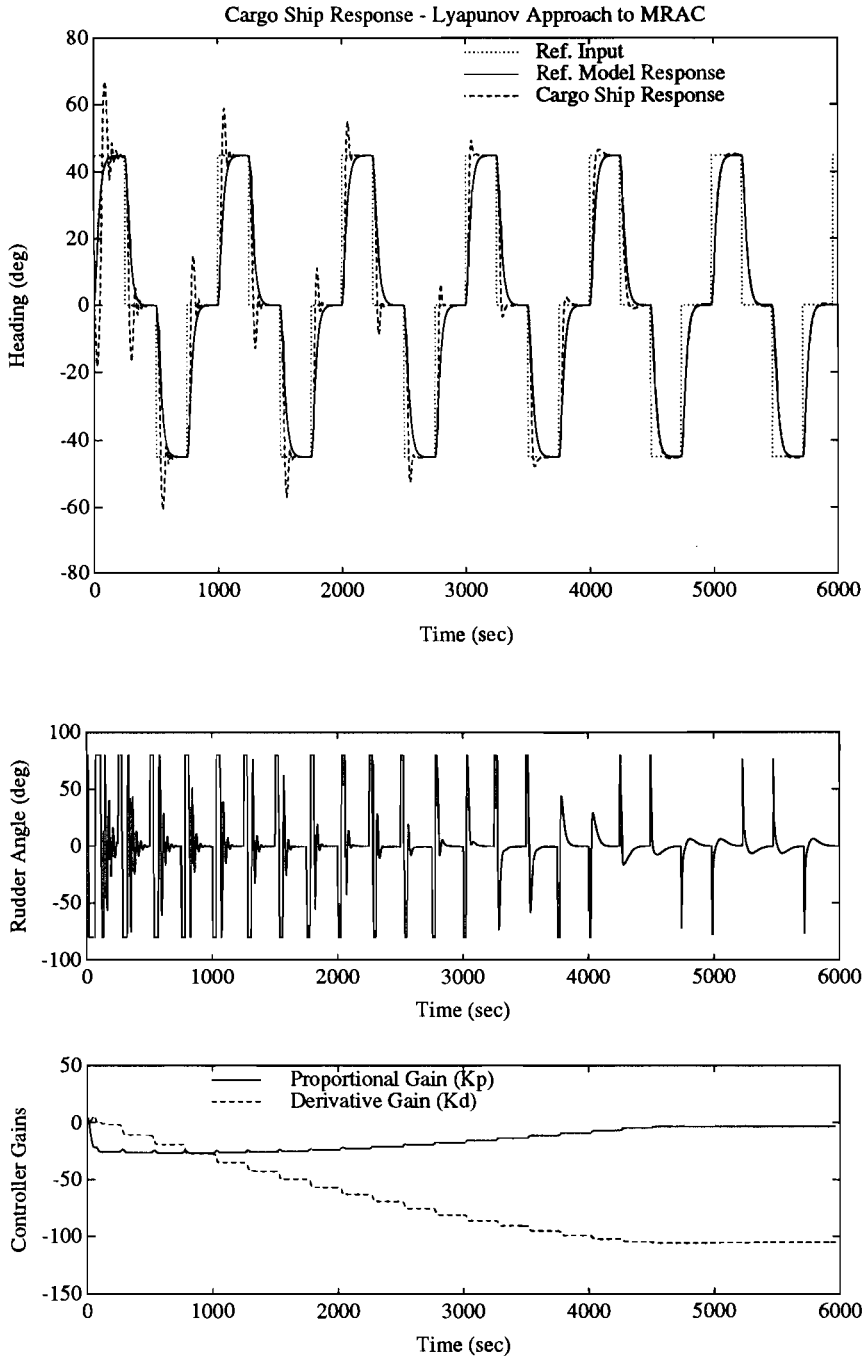


Figure 60: Simulation results for the Lyapunov approach to MRAC when employed for a cargo ship.

- Both MRAC algorithms are designed to minimize the error signal ψ_e. This does not necessarily mean that the process parameters will also converge.
- The reduced second order linear process model, for which the controller designs and the "optimal" process gains are based, is not a completely accurate characterization of the third order non-linear process model used in simulation.

For both the gradient approach and the Lyapunov approach, the system response converged to track the reference model. However, the convergence rate of both algorithms was significantly slower than the FMRLC method.

Another significant advantage of the FMRLC algorithm may be seen in the amount of input energy which was spent at the system input to obtain an accurate tracking with the reference model. Due to the fact that the magnitude of the rudder angle is generally larger for both MRAC approaches than for the FMRLC algorithm, we may suspect that the input energy for the FMRLC is significantly less. The rudder angle plots shown in Figures 58, 59, and 60 for each of the adaptive control algorithms represents only 1200 sampled data points. We may obtain a measure of the input energy if we think of these data points as a vector $\underline{\delta} = [\delta(0)\ \delta(T)\ \delta(2T)\ ...\ \delta(1199T)]^T$, where the energy is the square of the 2-norm for this vector (i.e., $energy = \underline{\delta}^T\underline{\delta}$). Upon performing this for the data shown in Figures 58, 59, and 60 in radians rather than degrees, we obtain the result shown in Table 22 below. As expected the process input energy for the FMRLC was significantly less than that obtained for both MRAC approaches. However, we should point out that sometimes the input energy of the FMRLC is not always small. For several application considered in this thesis, we have observed that often the FMRLC algorithm behaves similar to a variable structure controller by providing large positive/negative process inputs when the process error is positive and large negative/positive process inputs when the process

Table 22: Comparison of the process input energy for the FMRLC, the gradient MRAC, and the Lyapunov MRAC when employed as an autopilot for a cargo ship.

Process Input Energy ($\|\|\delta\|\|^2$)		
FMRLC	**Gradient MRAC**	**Lyapunov MRAC**
17.3368	458.6324	425.7801

error is negative. When this occurs the input energy can become quite large. However, even when this situation occurs the input energy may still be relatively small when compared with conventional adaptive control methods.

The final set of experiments performed for this process were designed to illustrate the ability of the adaptive controller to compensate for large disturbances at the process input. Figure 61 illustrates the results obtained for this simulation. The disturbance added at the rudder was chosen to be be a sinusoidal with a frequency of one cycle per minute and a magnitude of two degrees with a D.C. bias of one degree. The effect of this disturbance is similar to that of a gusting wind acting upon the ship since wind effects are generally modeled as a rudder disturbance. To provide a fair comparison with the FMRLC algorithm, we initially loaded the PD controllers in both MRAC algorithms with the with the controller gains shown in Table 21, which were previously found by each method. However, the knowledge base of the fuzzy controller in the FMRLC algorithm was initialized with all zeros.

Notice that the FMRLC algorithm was nearly able to completely cancel the effects of the disturbance input. However, the gradient and the Lyapunov approaches to MRAC where not nearly so successful.

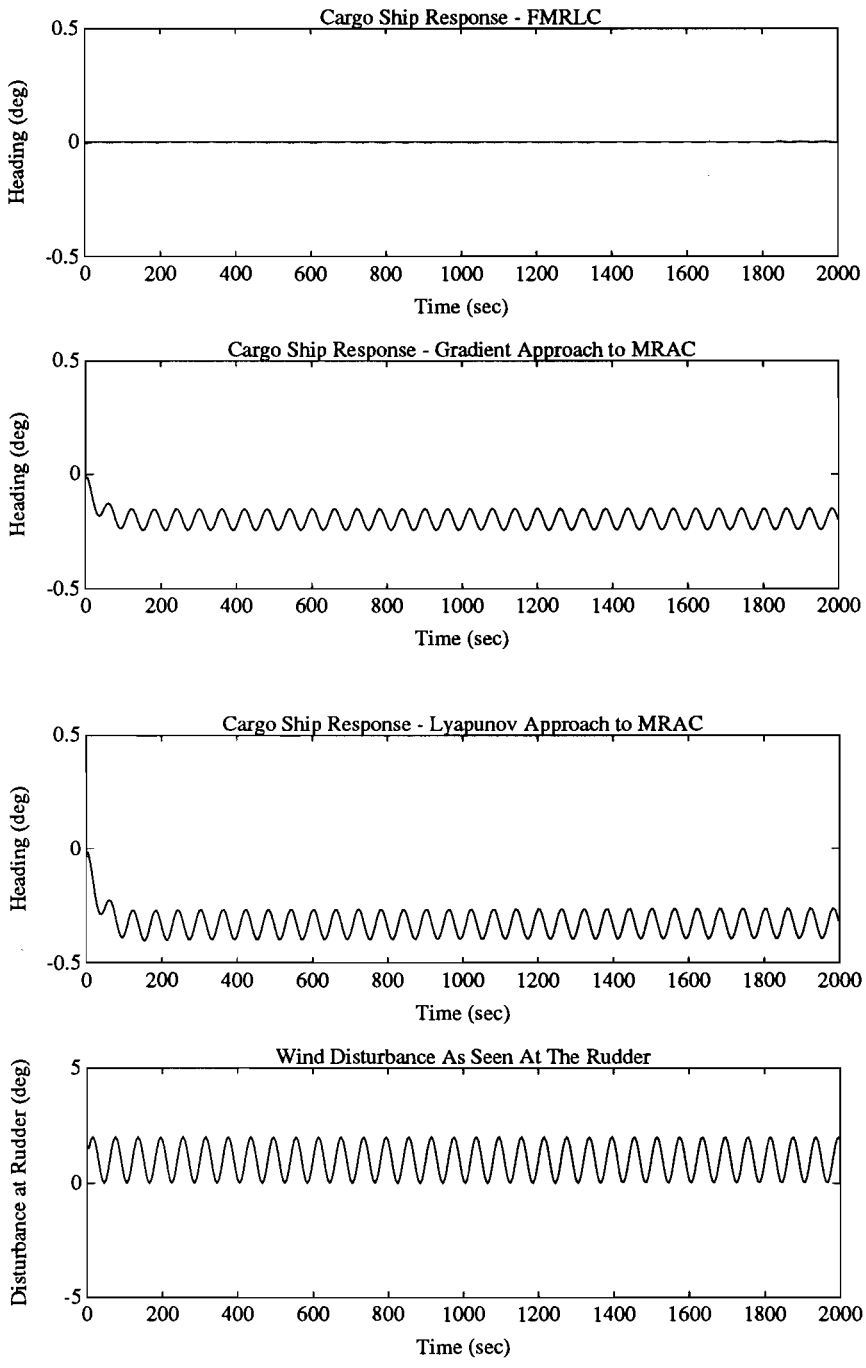


Figure 61: Simulation results compare disturbance rejection for the FMRLC, the gradient approach to MRAC, and the Lyapunov approach to MRAC.

6.5 Chapter Summary

In this chapter we have illustrated the application of the FMRLC algorithm for a real world application involving the directional heading control of a cargo ship. Also, since the FMRLC algorithm fits into the framework for direct adaptive control, the results obtain from applying the FMRLC to the cargo ship were compared with both the gradient and Lyapunov approaches to MRAC.

The results of simulations performed on these adaptive systems indicate that the FMRLC algorithm is likely to exhibit a faster rate of convergence. Also, for a cargo ship application the FMRLC provided a control signal whose energy content was significantly less than that produced by the conventional methods. In addition the FMRLC provided a greater degree of process input disturbance rejection than the conventional methods. Most astounding in the fact that these advantages of the FMRLC were obtained even though less information about the process was needed to perform the FMRLC design.

The results presented in this chapter only represent a start at simulation based studies. The advantages of the FMRLC which were observed from simulation are only for two types of adaptive control; certainly there are other techniques that may provide better results. For example, we could consider gain scheduling and self-tuning regulators. Also, we did not have the opportunity to implement and test this algorithm on a real ship. However, based on our initial studies serious consideration should be given to the FMRLC.

CHAPTER VII

Conclusions

One objective of this thesis was to briefly introduce fuzzy set and system theory and its application in control. Further, in this thesis we have provided a qualitative comparison of fuzzy controllers with conventional linear control schemes. This comparison illustrates the potential advantages of the fuzzy controller over the conventional control methods. Moreover, we have illustrated the practical application of a direct fuzzy controller for a motor positioning system.

The primary objective of this thesis was to introduce the FMRLC and discuss how it relates to the linguistic SOC of Procyk and Mamdani. In achieving this objective we have illustrated some of the motivations for the FMRLC algorithm. Also we have illustrated the design methodology and the application of both the linguistic SOC and the FMRLC algorithms for a cart and pendulum system. This application of the FMRLC algorithm illustrates the effectiveness of both the linguistic SOC and the FMRLC algorithm. The simulation experiments performed for this system have shown that the FMRLC provides certain advantages over the linguistic SOC algorithm. For example, by utilizing a reference model we provide an improved method for specifying the desired performance of the controlled process. Furthermore, we provide a greater degree of flexibility in the inverse model design by employing a

fuzzy system.

Other applications of the FMRLC algorithm considered in this thesis include: a solid propellant rocket, a two degree-of-freedom robot manipulator, an automotive anti-skid braking system, and a cargo ship autopilot. The rocket application was provided to illustrate the effectiveness of the FMRLC algorithm for controlling highly time-varying process. The rocket process is time-varying since the mass of the rocket changes as the fuel is burned. The two degree-of-freedom robot manipulator application illustrates the effectiveness of the FMRLC algorithm when controlling a non-linear, time-varying, MIMO process. The anti-skid application was considered as a practical alternative to other control techniques which are often employed for this application. The simulation results for the ABS system indicate that the FMRLC provides a "relatively good" method of compensating for process variations resulting from changing road surfaces and road inclines. Based on the simulation results obtained with the ABS system, we feel that the FMRLC algorithm should be given serious consideration for ABS systems.

The final application considered in this thesis was an autopilot for a cargo ship. For this application, the FMRLC was compared with both the gradient and Lyapunov approaches to MRAC. The results of simulations on these systems indicate that the FMRLC provides a faster rate of convergence than the MRAC approach so that the process tracks the reference model within a shorter time frame. Furthermore, we have illustrated that the energy at the input of the process is likely to be smaller for the FMRLC than for both the gradient and Lyapunov approaches to MRAC. Also for the cargo ship, we have observed that the FMRLC algorithm provides better rejection of a class of disturbances which occur at the process input.

Based on the design and simulation results for the above applications, we sum-

marize below some of the motivations and reasons why one should consider the FMRLC algorithm for controlling processes.

- A detailed analytical model of the process is not needed.
- It provides an autonomous method for generating and modifying fuzzy control rules while ensuring that the system behaves in a desirable fashion.
- Since the fuzzy controller is continually updated in response to process parameter variations and/or disturbances, it is hoped that the FMRLC provides "robust" control.

In addition, by combining learning control concepts with fuzzy system theory, we have developed a control scheme which has a fast rate of convergence and often provides an accurate "optimal" mapping between controller inputs and outputs.

Despite the many advantages of the FMRLC algorithm, several drawbacks do exist:

- The entire design process tends to be *ad hoc*,
- The linguistically expressed fuzzy inverse model may not be available,
- A given fuzzy inverse model may not be reliable for all process states or conditions,
- Conditions for stability and convergence of the algorithm are yet to be found, and
- The FMRLC algorithm is somewhat computationally intensive.

These disadvantages provide several future research directions. For example, future research involving the FMRLC algorithm should include a mathematical analysis of the controller to better quantify the effect of controller design parameters on

the response and stability of the system. Recently, several papers have appeared which investigates the stability of direct fuzzy controllers including: Langari [43, 44], Kiszka, Gupta, and Nikiforuk [41], Ray and Majumder [60], and Ray, Ghosh and Majumder [61]. It may be possible to incorporate these concepts into the knowledge base modification portion of the FMRLC algorithm to ensure that any controller modification results in a stable system. Also, research must be directed towards determining faster algorithms for FMRLC computations. Furthermore, note that the FMRLC algorithm may be classified as a "direct" adaptive control algorithm since the *a priori* information about the process behavior is incorporated into the fuzzy inverse model. This information may be obtained off-line and/or on-line by employing the fuzzy identification algorithm such as the one proposed by E. Czogała and W. Pedrycz in [16]. Alternatively, since the rule base modification algorithm is essentially a "controller identification algorithm", we may modify this algorithm slightly to provide the function of process identification. A Fuzzy process identification algorithm would provide a mechanism for updating the fuzzy inverse model in the event that the current inverse model becomes invalid due to process variations or disturbances. In addition to the above, future research should be directed toward analyzing and quantify *persistent excitation* of the FMRLC algorithm. Finally, the FMRLC algorithm must be employed in real world applications to determine if computer capabilities are sufficient for implementation of this algorithm.

APPENDIX A

Computer Simulation Software

In this appendix we present examples of computer software used for implementation and simulation of the direct fuzzy control systems, the linguistic SOC system, the FMRLC systems which are presented throughout this thesis. The motor positioning system and the cart and pendulum system were chosen as typical examples of these control algorithms.

A.1 Motor Simulation Program - Fuzzy Control

```
c**********************************************************************
c
      program motor_simulation
c
c**********************************************************************
c**********************************************************************
c
c     Programmed by       : Jeffery R. Layne
c     Date                : 9/5/90
c     Purpose             :
c
c     This program simulate the position response of a U12M4T D.C.
c     motor for which a fuzzy system controller is employed in the
c     feedback loop. The inputs to the fuzzy controller are the position
c     error and the position error derivative.
c
c     Subroutines called  :
c               1) subroutine flc(n,u,gu,fs,y,gy,by,rule_base,applied_fs)
c
c     Files read          :
c               1) flc_parameters.motor
c
c     Output files   :
c               1) motor.dat
c
```

```
c     Input files
c
c
c     Motor Parameters    :
c
c               1) bm   - rotational damping on the motor.
c               2) jm   - internal inertia on the motor.
c               3) rm   - internal d.c. resistance of the motor.
c               4) nm   - turn ratio of the internal gears.
c               5) kt   - torque constant.
c               6) kb   - back emf constant.
c               7) tf   - rotational friction torque.
c
c*********************************************************************
c
c Define Controller Variables
c
      integer  n, fs(5)
      real     flc_input(5), gu(5), flc_output, gy, by
      real     rule_base(161051), applied_fs(5,2,2)
c
c Define motor parameters
c
      real  bm, jm, nm, rm, kt, kb, tf, theta, theta_dot, vin, y
c
c Define program variables
c
      integer          j, k, iter_per_flc, iterations, iter_per_sp
      real             dt, e, ce, sp, t, pi, sp1, sp2, sp3
      real             time(10000), thetah, dtheta, dthetah, e_prev
      real             theta_doth, dtheta_dot
      real             dtheta_doth, motor_sp(10000)
      real             motor_output(10000), motor_input(10000)
      character*80     header(7)
      logical          exist
c
c Initialize motor parameters
c
      bm = 0.000303
      jm = 0.000233
      rm = .75
      nm = 102
      kt = .101
      kb = .101
      tf = 0.0424
      theta = 0.0
      theta_dot = 0.0
      y = theta
c
c Initialize program variables
c
      pi = 3.1416
      dt = 1.0/100.0
      t = 0.0
c
c Initialize controller variables
c
      open(unit=10,status='old',file='flc_parameters.motor',err=1000)
      do 10 j = 1,6
         read(10,'(a80)') header(j)
 10   continue
      read(10,*,err=1001)n
```

```
      read(10,*,err=1001)fs(1)
      read(10,*,err=1001)fs(2)
      read(10,*,err=1001)gu(1)
      read(10,*,err=1001)gu(2)
      read(10,*,err=1001)gy
      read(10,*,err=1001)by
      read(10,*,err=1001)iter_per_flc
      read(10,*,err=1001)sp1
      read(10,*,err=1001)sp2
      read(10,*,err=1001)sp3
      read(10,*,err=1001)iterations
      read(10,*,err=1001)iter_per_sp
      read(10,'(a80)') header(7)
      do 20 j = 1,fs(1)
         read(10,*,err=1001)
     #     (rule_base(k),k=(1+((j-1)*fs(2))),(((j-1)*fs(2)))+fs(2))
 20   continue
      close(10)
c
c Determine the initial input to the process
c
      sp = sp1
      e = sp - y
      ce = 0
      flc_input(1) = e
      flc_input(2) = ce
      call flc(n,flc_input,gu,fs,flc_output,gy,by,rule_base,applied_fs)
      vin = flc_output
c
c ******************************Begin computing
c
      do 30 k = 1,iterations
         if (jmod(k,iter_per_sp) .eq. 0) then
            if (sp .eq. sp1) then
               sp = sp2
            else if (sp .eq. sp2) then
               sp = sp3
            else if (sp .eq. sp3) then
               sp = sp1
            end if
         end if
         time(k) = t
         motor_input(k) = vin
         motor_output(k) = y
         motor_sp(k) = sp
c
c ******************************predict next states
c
         dtheta = theta_dot
         dtheta_dot = theta_dot*(-rm*bm - kb*kt)/(rm*jm) +
     #          vin*kt/(nm*rm*jm) - tf/(nm*jm)*sign(1.0,theta_dot)
         thetah = theta + dt*dtheta
         theta_doth = theta_dot + dt*dtheta_dot
c
c ******************************correct next states
c
         dthetah = theta_doth
         dtheta_doth = theta_doth*(-rm*bm - kb*kt)/(rm*jm) +
     #          vin*kt/(nm*rm*jm) - tf/(nm*jm)*sign(1.0,theta_doth)
         theta = theta + ((dt/2)*(dthetah+dtheta))
         theta_dot = theta_dot + ((dt/2)*(dtheta_doth+dtheta_dot))
c
```

```
c ****************************compute next output
c
        y = theta
c
c ******************Compute next controller parameters if needed
c
        if (jmod(k,iter_per_flc) .eq. 0) then
           e_prev = e
           e = sp - y
           ce = (e - e_prev)/(dt*iter_per_flc)
           print*,'e = ',e
           print*,'ce = ',ce
           flc_input(1) = e
           flc_input(2) = ce
c
c ***********************Compute next process input if needed
c
           call
     #     flc(n,flc_input,gu,fs,flc_output,gy,by,rule_base,applied_fs)
           vin = flc_output
         end if
c
c *********************************Increment time
c
        t = t + dt
c
c ****************************Go back to the beginning
c
 30   continue
c
c Output results
c
      inquire(file='motor.mat',exist = exist)
      if (exist) then
         open(unit=11,status='old',file='motor.mat', form = 'formatted')
      else
         open(unit=11,status='new',file='motor.mat', form = 'formatted')
      end if
      do 40 k = 1,iterations
      write(11,*) time(k), motor_output(k), motor_input(k), motor_sp(k)
 40   continue
      close(11)
      goto 2000
c
c Error Comments
c
 1000 print*,'$****Rule base data file does not exist !!!'
      goto 2000
 1001 print*,'$****Check controller parameters data file !!!'
      goto 2000
c
c End program
c
 2000 stop
      end
```

A.1.1 File "flc_parameters.motor" for the Motor Simulation Program

```
c*****************************************************************
c           CONTROLLER PARAMETERS FOR THE MOTOR SIMULATION
c*****************************************************************
```

```
c       This file contains the fuzzy controller parameters used
c       in the motor simulation program.
c*****************************************************************
 2                      % controller inputs
 11                     % fuzzy sets on error u.o.d.
 11                     % fuzzy sets on error_change u.o.d
 0.159155               % gain applied to error
 0.02                   % gain applied to change in error
 220.0                  % controller output gain
 0.4                    % normalized width of fs on output u.o.d.
 5                      % iterations before flc
 1.5708                 % 1st set point - desired position of the motor
 0.0                    % 2nd set point - desired position of the motor
 -1.5708                % 3rd set point - desired position of the motor
 1500                   % total number of integration iterations
 300                    % iterations before a set point change
c    The following is the rule base for the fuzzy controller
 -1.0  -1.0  -1.0  -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0
 -1.0  -1.0  -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2
 -1.0  -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4
 -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6
 -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8
 -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0
 -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0
 -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0
 -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0  +1.0
 -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0  +1.0  +1.0
 +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0  +1.0  +1.0  +1.0
```

A.2 Cart and Pendulum Simulation Program - Linguistic SOC

```
c*********************************************************************
c
      program SOC_of_cart_and_pendulum
c
c*********************************************************************
c*********************************************************************
c
c     Programmed by       : Jeffery R. Layne
c     Date                : 9/5/91
c     Purpose             :
c
c     This program simulate the control of an inverted pendulum
c     for which a linguistic self-organizing control system as described
c     by Procyk and Mandami is employed. The input to the fuzzy
c     system is the position error and the change in position error.
c
c     Subroutines called  :
c               1) subroutine flc(n,u,gu,fs,y,gy,by,rule_base,applied_fs)
c               2) subroutine rbm(n,fs,y_change,gy,rule_base,applied_fs)
c
c     Files read          :
c               1) cart_parameter.soc
c
c     Pendulum Parameters :
c
c               1) lp   - half pole length
c               2) mc   - mass of the cart
c               3) mp   - mass of the pole
c               4) g    - gravitation constant
```

```
c
c*********************************************************************
c
c Define Controller Variables
c
      integer  n, fs(5)
      real     flc_input(5), gu(5), flc_output, gy, by
      real     rule_base(161051), applied_fs(5,2,2)
c
c Define performance evaluator parameters - Note that those not defined
c                 are the same as that defined in the controller
c
      real     y_change, gy_perf
      real     rule_base_perf(161051), garbage(5,2,2)
c
c Define cart and pendulum parameters
c
      real  lp, mc, mp, g, theta, theta_dot, fc, y
c
c Define program variables
c
      integer          j, k, iter_per_flc, iterations,try,tries
      integer          iter_per_sp, iter_per_output
      logical          exist
      real             dt, e, ce, sp, t, f_change, theta_initial
      real             sp1, sp2, sp3, set_point(10000)
      real             time(10000), thetah, dtheta, dthetah, e_prev
      real             theta_doth, dtheta_dot, dtheta_doth
      real             theta_output(10,10000), force_input(10,10000)
      character*80     header(30)
c
c Define inverse model parameter
c
      real model_inverse
c
c Initialize cart  and pendulum parameters
c
      open(unit=10,status='old',file='cart_parameter.soc',err=1000)
      do 10 j = 1,8
         read(10,'(a80)') header(j)
 10   continue
      read(10,*,err=1001)lp
      read(10,*,err=1001)mc
      read(10,*,err=1001)mp
      read(10,*,err=1001)theta_initial
      g = 9.8
c
c Initialize program variables
c
      read(10,'(a80)') header(9)
      read(10,*,err=1001)sp1
      read(10,*,err=1001)sp2
      read(10,*,err=1001)sp3
      read(10,*,err=1001)iter_per_sp
      read(10,*,err=1001)dt
      read(10,*,err=1001)iterations
      read(10,*,err=1001)iter_per_output
      read(10,*,err=1001)tries
c
c Initialize controller variables
c
      read(10,'(a80)') header(10)
```

```
      read(10,*,err=1001)n
      read(10,*,err=1001)fs(1)
      read(10,*,err=1001)fs(2)
      read(10,*,err=1001)gu(1)
      read(10,*,err=1001)gu(2)
      read(10,*,err=1001)gy
      read(10,*,err=1001)by
      read(10,*,err=1001)iter_per_flc
      read(10,'(a80)') header(11)
      do 20 j = 1,fs(1)
         read(10,*,err=1001)
     #     (rule_base(k),k=(1+((j-1)*fs(2))),(((j-1)*fs(2)))+fs(2))
 20   continue
c
c Initialize performance fuzzy system parameters
c
      read(10,'(a80)') header(12)
      read(10,*,err=1001)gy_perf
      read(10,'(a80)') header(13)
      do 25 j = 1,fs(1)
         read(10,*,err=1001)
     #     (rule_base_perf(k),k=(1+((j-1)*fs(2))),(((j-1)*fs(2)))+fs(2))
 25   continue
      y_change = 0.0
c
c Initialize inverse model parameter
c
      read(10,'(a80)') header(14)
      read(10,*,err=1001)model_inverse
      close(10)
c
c Determine the initial input to the process
c
      do 50 try = 1,tries
      print*,'$New Try !!!!!!!!'
      theta= theta_initial
      theta_dot = 0.0
      t = 0.0
      sp = sp1
      y = theta
      e = sp - y
      ce = 0
      flc_input(1) = e
      flc_input(2) = ce
      call flc(n,flc_input,gu,fs,flc_output,gy,by,rule_base,applied_fs)
      fc = flc_output
c
c ****************************Begin computing
c
      do 30 k = 1,iterations
         if (jmod(k,iter_per_sp) .eq. 0) then
            if (sp .eq. sp1) then
               sp = sp2
            else if (sp .eq. sp2) then
               sp = sp3
            else if (sp .eq. sp3) then
               sp = sp1
            end if
         end if
         set_point(k) = sp
         time(k) = t
         force_input(try,k) = fc
```

```
         theta_output(try,k) = y
c
c ******************************predict next states
c
         dtheta = theta_dot
         dtheta_dot = (g*sin(theta) +
     #       cos(theta)*((-fc-mp*lp*(theta_dot**2)*sin(theta))/(mc+mp)))/
     #       (lp*(4/3 - (mp*(cos(theta)**2))/(mc+mp)))
         thetah = theta + dt*dtheta
         theta_doth = theta_dot + dt*dtheta_dot
c
c ******************************correct next states
c
         dthetah = theta_doth
         dtheta_doth = (g*sin(thetah) +
     #     cos(thetah)*((-fc-mp*lp*(theta_doth**2)*sin(thetah))/(mc+mp)))/
     #     (lp*(4/3 - (mp*(cos(thetah)**2))/(mc+mp)))
c
         theta = theta + ((dt/2)*(dthetah+dtheta))
         theta_dot = theta_dot + ((dt/2)*(dtheta_doth+dtheta_dot))
c
c ******************************compute next output
c
         y = theta
c
c *Modify controller rule base and Compute next controller parameters if needed
c
         if (jmod(k,iter_per_flc) .eq. 0) then
            e_prev = e
            e = sp - y
            ce = (e - e_prev)/(dt*iter_per_flc)
            print*,'e = ',e
            print*,'ce = ',ce
            flc_input(1) = e
            flc_input(2) = ce
c
c ******for better performance correct previously applied fs in rule base
c
            call
     #        flc(n,flc_input,gu,fs,y_change,gy_perf,
     #          by,rule_base_perf,garbage)
            f_change = model_inverse * y_change
            call rbm(n,fs,f_change,gy,rule_base,applied_fs)
c
c ***********************Compute next process input if needed
c
            call
     #      flc(n,flc_input,gu,fs,flc_output,gy,by,rule_base,applied_fs)
            fc = flc_output
          end if
c
c *********************************Increment time
c
         t = t + dt
c
c ****************************Go back to the beginning
c
 30   continue
 50   continue
c
c Output the time in a file called 'cart_t.mat'
c
```

```
      inquire(file='cart_t.mat',exist = exist)
      if (exist) then
      open(unit=11,status='old',file='cart_t.mat', form = 'formatted')
      else
      open(unit=11,status='new',file='cart_t.mat', form = 'formatted')
      end if
      do 60 k = 1,iterations,iter_per_output
      write(11,*) time(k)
 60   continue
      close(11)
c
c Output force applied to the cart in a file 'cart_i.mat'
c
      inquire(file='cart_i.mat',exist = exist)
      if (exist) then
      open(unit=12,status='old',file='cart_i.mat', form = 'formatted')
      else
      open(unit=12,status='new',file='cart_i.mat', form = 'formatted')
      end if
      do 70 k = 1,iterations,iter_per_output
      write(12,*)(force_input(try,k),try = 1,tries)
 70   continue
      close(12)
c
c Output position of the pendulum in a file 'cart_o.mat'
c
      inquire(file='cart_o.mat',exist = exist)
      if (exist) then
      open(unit=13,status='old',file='cart_o.mat', form = 'formatted')
      else
      open(unit=13,status='new',file='cart_o.mat', form = 'formatted')
      end if
      do 80 k = 1,iterations,iter_per_output
      write(13,*) (theta_output(try,k),try = 1,tries)
 80   continue
      close(13)
c
c Output the final rule base to 'rb.mat'
c
      inquire(file='rb.mat',exist = exist)
      if (exist) then
      open(unit=14,status='old',file='rb.mat', form = 'formatted')
      else
      open(unit=14,status='new',file='rb.mat', form = 'formatted')
      end if
      do 90 j = 1,fs(1)
         write(14,*)
     #     (rule_base(k),k=(1+((j-1)*fs(2))),(((j-1)*fs(2)))+fs(2))
 90   continue
      close(14)
c
c Output the desired set point for the system to 'cart_sp.mat'
c
      inquire(file='cart_sp.mat',exist = exist)
      if (exist) then
      open(unit=15,status='old',file='cart_sp.mat', form = 'formatted')
      else
      open(unit=15,status='new',file='cart_sp.mat', form = 'formatted')
      end if
      do 100 k = 1,iterations,iter_per_output
      write(15,*) set_point(k)
 100  continue
```

```
      close(15)
      goto 2000
c
c Error Comments
c
 1000 print*,'$****Cart parameters data file does not exist !!!'
      goto 2000
 1001 print*,'$****Check Cart parameters data file !!!'
      goto 2000
c
c End program
c
 2000 stop
      end
```

A.2.1 File "cart_parameters.soc" for SOC of the Cart and Pendulum Program

```
c*****************************************************************
c      PARAMETERS FOR THE INVERTED PENDULUM SIMULATION WITH SOC
c*****************************************************************
c This file contains the cart parameters, the fuzzy controller
c       parameters, and performance fuzzy system parameters used
c in the inverted pendulum simulation program.
c*****************************************************************
c Initial cart and pendulum parameters
 0.5                      % half pole length in meters
 1.0                      % mass of the cart in kg
 0.5                      % mass of the pole in kg
 -3.1416                  % initial position of the pole in rad.
c Initial Program parameters
 0.0                      % 1st desired pole position set point
 -0.4363                  % 2nd desired pole position set point
 0.4363                   % 3rd desired pole position set point
 1001                     % iteration before a set point change
 0.001                    % integration step time
 1000                     % total number of integration iterations
 5                        % data reduction factor in the output data
 5                        % tries for better performance
c fuzzy controller parameters
 2                        % controller inputs
 11                       % fuzzy sets on error u.o.d.
 11                       % fuzzy sets on error_change u.o.d
 0.159155                 % gain applied to error
 0.02                     % gain applied to change in error
 200.0                    % controller output gain
 0.4                      % normalized width of fs on output u.o.d.
 5                        % integration iterations before flc
c   The following is the initial rule base for the fuzzy controller
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
c Performance fuzzy system parameters
```

```
 220.0                  % output gain of performance fuzzy system
c    The following is the rule base for the performance fuzzy system
 -1.0  -1.0  -1.0  -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0
 -1.0  -1.0  -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2
 -1.0  -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4
 -1.0  -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6
 -1.0  -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8
 -1.0  -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0
 -0.8  -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0
 -0.6  -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0
 -0.4  -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0  +1.0
 -0.2  +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0  +1.0  +1.0
 +0.0  +0.2  +0.4  +0.6  +0.8  +1.0  +1.0  +1.0  +1.0  +1.0  +1.0
c Inverse model Parameter
 -0.9                   % Inverse model constant
```

A.3 Cart and Pendulum Simulation Program - FMRLC

```
c*************************************************************************
c
      program FMRLC_of_cart_and_pendulum
c
c*************************************************************************
c*************************************************************************
c
c     Programmed by       : Jeffery R. Layne
c     Date                : 9/15/91
c     Purpose             :
c
c     This program simulate the control response of an inverted pendulum
c     for which a fuzzy model reference learning control scheme is
c     employed. This fuzzy model reference adaptive control scheme can
c     be viewed as an extension of the technique described by Procyk and
c     Mamdani. The input to the fuzzy system is the position error and
c     the change in position error.
c
c     Subroutines called  :
c               1) subroutine flc(n,u,gu,fs,y,gy,by,rule_base,applied_fs)
c               2) subroutine rbm(n,fs,y_change,gy,rule_base,applied_fs)
c
c     Files read          :
c               1) cart_parameter.fmrlc
c
c     Pendulum Parameters :
c
c               1) lp   - half pole length
c               2) mc   - mass of the cart
c               3) mp   - mass of the pole
c               4) g    - gravitation constant
c
c*************************************************************************
c
c Define Controller Variables
c
      integer  n, fs(5)
      real     flc_input(5), gu(5), flc_output, gy, by
      real     rule_base(161051), applied_fs(5,2,2)
c
c Define Parameters for the model reference system
c
```

```
      real     am, bm, gm, ym, ymh, dym, dymh, ym1, y_change
      real     y_change_d, y_change_prev
c
c Define cart and pendulum parameters
c
      real  lp, mc, mp, g, theta, theta_dot, fc, y
c
c Define program variables
c
      integer          j, k, iter_per_flc, iterations,try,tries
      integer          iter_per_sp, iter_per_output
      logical          exist
      real             dt, e, ce, sp, t, f_change, theta_initial
      real             sp1, sp2, sp3
      real             set_point(10000), model_output(10000)
      real             time(10000), thetah, dtheta, dthetah, e_prev
      real             theta_doth, dtheta_dot, dtheta_doth
      real             theta_output(10,10000), force_input(10,10000)
      character*80     header(30)
c
c Define inverse model parameter
c
      integer  n_imodel, imodel_fs(5)
      real     imodel_input(5), imodel_gu(5), imodel_output
      real     imodel_gy, imodel_by
      real     imodel_rule_base(161051), trash(5,2,2)
c
c Initialize cart  and pendulum parameters
c
      open(unit=10,status='old',file='cart_parameter.fmrlc',err=1000)
      do 10 j = 1,8
         read(10,'(a80)') header(j)
 10   continue
      read(10,*,err=1001)lp
      read(10,*,err=1001)mc
      read(10,*,err=1001)mp
      read(10,*,err=1001)theta_initial
      g = 9.8
c
c Initialize program variables
c
      read(10,'(a80)') header(9)
      read(10,*,err=1001)sp1
      read(10,*,err=1001)sp2
      read(10,*,err=1001)sp3
      read(10,*,err=1001)iter_per_sp
      read(10,*,err=1001)dt
      read(10,*,err=1001)iterations
      read(10,*,err=1001)iter_per_output
      read(10,*,err=1001)tries
c
c Initialize controller variables
c
      read(10,'(a80)') header(10)
      read(10,*,err=1001)n
      read(10,*,err=1001)fs(1)
      read(10,*,err=1001)fs(2)
      read(10,*,err=1001)gu(1)
      read(10,*,err=1001)gu(2)
      read(10,*,err=1001)gy
      read(10,*,err=1001)by
      read(10,*,err=1001)iter_per_flc
```

```
      read(10,'(a80)') header(11)
      do 20 j = 1,fs(1)
         read(10,*,err=1001)
     #     (rule_base(k),k=(1+((j-1)*fs(2))),(((j-1)*fs(2)))+fs(2))
 20   continue
c
c Initialize reference model parameters
c
      read(10,'(a80)') header(12)
      read(10,*,err=1001)am
      read(10,*,err=1001)gm
      read(10,*,err=1001)ym1
      bm = -gm*am
c
c Initialize fuzzy inverse model parameter
c
      read(10,'(a80)') header(13)
      read(10,*,err=1001)n_imodel
      read(10,*,err=1001)imodel_fs(1)
      read(10,*,err=1001)imodel_fs(2)
      read(10,*,err=1001)imodel_gu(1)
      read(10,*,err=1001)imodel_gu(2)
      read(10,*,err=1001)imodel_gy
      read(10,*,err=1001)imodel_by
      read(10,'(a80)') header(14)
      do 25 j = 1,imodel_fs(1)
         read(10,*,err=1001)
     #     (imodel_rule_base(k),k=(1+((j-1)*imodel_fs(2))),
(    #                 (((j-1)*imodel_fs(2)))+imodel_fs(2))
 25   continue
      close(10)
c
c Determine the initial input to the process
c
      do 50 try = 1,tries
      print*,'$New Try !!!!!!!!'
      theta= theta_initial
      theta_dot = 0.0
      y_change_prev = 0
      y_change_d = 0
      t = 0.0
      ym = ym1
      sp = sp1
      y = theta
      e = sp - y
      ce = 0
      flc_input(1) = e
      flc_input(2) = ce
      call flc(n,flc_input,gu,fs,flc_output,gy,by,rule_base,applied_fs)
      fc = flc_output
c
c ******************************Begin computing
c
      do 30 k = 1,iterations
         if (jmod(k,iter_per_sp) .eq. 0) then
            if (sp .eq. sp1) then
               sp = sp2
            else if (sp .eq. sp2) then
               sp = sp3
            else if (sp .eq. sp3) then
               sp = sp1
            end if
```

```
         end if
         set_point(k) = sp
         time(k) = t
         force_input(try,k) = fc
         theta_output(try,k) = y
         model_output(k) = ym
c
c ****************************predict next states
c
         dtheta = theta_dot
         dtheta_dot = (g*sin(theta) +
     #      cos(theta)*((-fc-mp*lp*(theta_dot**2)*sin(theta))/(mc+mp)))/
     #      (lp*(4/3 - (mp*(cos(theta)**2))/(mc+mp)))
         thetah = theta + dt*dtheta
         theta_doth = theta_dot + dt*dtheta_dot
c
c ****************************correct next states
c
         dthetah = theta_doth
         dtheta_doth = (g*sin(thetah) +
     # cos(thetah)*((-fc-mp*lp*(theta_doth**2)*sin(thetah))/(mc+mp)))/
     # (lp*(4/3 - (mp*(cos(thetah)**2))/(mc+mp)))
c
         theta = theta + ((dt/2)*(dthetah+dtheta))
         theta_dot = theta_dot + ((dt/2)*(dtheta_doth+dtheta_dot))
c
c ****************************compute next output
c
         y = theta
c
c ****************************compute next reference model states
c
         dym = am*ym + bm*sp
         ymh = ym + dt*dym
         dymh = am*ymh + bm*sp
         ym = ym + ((dt/2)*(dymh+dym))
c
c *Modify controller rule base and Compute next controller parameters if needed
c
         if (jmod(k,iter_per_flc) .eq. 0) then
c
c ******for better performance correct previously applied fs in rule base
c
            y_change_prev = y_change
            y_change = ym - y
            y_change_d = (y_change - y_change_prev)/(dt*iter_per_flc)
            imodel_input(1) = y_change
            imodel_input(2) = y_change_d
            call flc(n_imodel,imodel_input,imodel_gu,imodel_fs,
     #         imodel_output,imodel_gy,imodel_by,imodel_rule_base,trash)
            f_change = imodel_output
            if (abs(y) .le. 1.571) then
               f_change = f_change
            else
               f_change = -f_change
            end if
            print*,'f_change = ',f_change
            call rbm(n,fs,f_change,gy,rule_base,applied_fs)
c
c ************************Compute next process input if needed
c
            e_prev = e
```

```
            e = sp - y
            ce = (e - e_prev)/(dt*iter_per_flc)
            print*,'e = ',e
            print*,'ce = ',ce
            flc_input(1) = e
            flc_input(2) = ce
            call
     #      flc(n,flc_input,gu,fs,flc_output,gy,by,rule_base,applied_fs)
            fc = flc_output
          end if
c
c ***********************************Increment time
c
         t = t + dt
c
c ****************************Go back to the beginning
c
 30   continue
 50   continue
c
c Output the time in a file called 'cart_t.mat'
c
      inquire(file='cart_t.mat',exist = exist)
      if (exist) then
      open(unit=11,status='old',file='cart_t.mat', form = 'formatted')
      else
      open(unit=11,status='new',file='cart_t.mat', form = 'formatted')
      end if
      do 60 k = 1,iterations,iter_per_output
      write(11,*) time(k)
 60   continue
      close(11)
c
c Output force applied to the cart in a file 'cart_i.mat'
c
      inquire(file='cart_i.mat',exist = exist)
      if (exist) then
      open(unit=12,status='old',file='cart_i.mat', form = 'formatted')
      else
      open(unit=12,status='new',file='cart_i.mat', form = 'formatted')
      end if
      do 70 k = 1,iterations,iter_per_output
      write(12,*)(force_input(try,k),try = 1,tries)
 70   continue
      close(12)
c
c Output position of the pendulum in a file 'cart_o.mat'
c
      inquire(file='cart_o.mat',exist = exist)
      if (exist) then
      open(unit=13,status='old',file='cart_o.mat', form = 'formatted')
      else
      open(unit=13,status='new',file='cart_o.mat', form = 'formatted')
      end if
      do 80 k = 1,iterations,iter_per_output
      write(13,*) (theta_output(try,k),try = 1,tries)
 80   continue
      close(13)
c
c Output the final rule base to 'rb.mat'
c
      inquire(file='rb.mat',exist = exist)
```

```
      if (exist) then
      open(unit=14,status='old',file='rb.mat', form = 'formatted')
      else
      open(unit=14,status='new',file='rb.mat', form = 'formatted')
      end if
      do 90 j = 1,fs(1)
         write(14,*)
     #     (rule_base(k),k=(1+((j-1)*fs(2))),(((j-1)*fs(2)))+fs(2))
 90   continue
      close(14)
c
c Output the desired set point for the system to 'cart_sp.mat'
c
      inquire(file='cart_sp.mat',exist = exist)
      if (exist) then
      open(unit=15,status='old',file='cart_sp.mat', form = 'formatted')
      else
      open(unit=15,status='new',file='cart_sp.mat', form = 'formatted')
      end if
      do 100 k = 1,iterations,iter_per_output
      write(15,*) set_point(k)
 100  continue
c
c Output the reference model output to 'ref.mat'
c
      inquire(file='ref.mat',exist = exist)
      if (exist) then
      open(unit=16,status='old',file='ref.mat', form = 'formatted')
      else
      open(unit=16,status='new',file='ref.mat', form = 'formatted')
      end if
      do 110 k = 1,iterations,iter_per_output
            write(16,*) model_output(k)
 110  continue
      close(16)
      goto 2000
c
c Error Comments
c
 1000 print*,'$****Cart parameters data file does not exist !!!'
      goto 2000
 1001 print*,'$****Check Cart parameters data file !!!'
      goto 2000
c
c End program
c
 2000 stop
      end
```

A.3.1 File "cart_parameters.fmrlc" for FMRLC of the Cart and Pendulum Program

```
c*****************************************************************
c      PARAMETERS FOR THE INVERTED PENDULUM SIMULATION WITH FMRLC
c*****************************************************************
c       This file contains the cart parameters, the fuzzy controller
c       parameters, and performance fuzzy system parameters used
c       in the inverted pendulum simulation program.
c*****************************************************************
c Initial cart and pendulum parameters
 0.5                    % half pole length in meters
```

```
 1.0                    % mass of the cart in kg
 0.5                    % mass of the pole in kg
 -1.0472                % initial position of the pole in rad.
c Initial Program parameters
 0.0                    % 1st desired pole position set point
 -0.4363                % 2nd desired pole position set point
 0.4363                 % 3rd desired pole position set point
 1000                   % iteration before a set point change
 0.001                  % integration step time
 10000                  % total number of integration iterations
 5                      % data reduction factor in the output data
 1                      % tries for better performance
c fuzzy controller parameters
 2                      % controller inputs
 11                     % fuzzy sets on error u.o.d.
 11                     % fuzzy sets on error derivative u.o.d
 0.159155               % gain applied to error
 0.02                   % gain applied to error derivative
 200.0                  % controller output gain
 0.4                    % normalized width of fs on output u.o.d.
 5                      % integration iterations before flc
c   The following is the initial rule base for the fuzzy controller
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
 +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0  +0.0
c fuzzy model parameters
 -10                    % pole location for 1st order model
 1                      % gain of the first order system
 -1.0472                % initial output of the model
c Inverse process model Parameter
 2                      % inverse model inputs
 11                     % fuzzy sets on reference error u.o.d
 11                     % fuzzy sets on reference error derivative u.o.d
 0.159155               % gain applied to reference error
 0.1                    % gain applied to reference error derivative
 200.0                  % inverse model fuzzy system output gain
 0.4                    % normalized width of fs on inverse fuzzy system
c   The following is the rule base for the performance fuzzy system
 +1.0  +1.0  +1.0  +1.0  +1.0  +1.0  +0.8  +0.6  +0.4  +0.2  +0.0
 +1.0  +1.0  +1.0  +1.0  +1.0  +0.8  +0.6  +0.4  +0.2  +0.0  -0.2
 +1.0  +1.0  +1.0  +1.0  +0.8  +0.6  +0.4  +0.2  +0.0  -0.2  -0.4
 +1.0  +1.0  +1.0  +0.8  +0.6  +0.4  +0.2  +0.0  -0.2  -0.4  -0.6
 +1.0  +1.0  +0.8  +0.6  +0.4  +0.2  +0.0  -0.2  -0.4  -0.6  -0.8
 +1.0  +0.8  +0.6  +0.4  +0.2  +0.0  -0.2  -0.4  -0.6  -0.8  -1.0
 +0.8  +0.6  +0.4  +0.2  +0.0  -0.2  -0.4  -0.6  -0.8  -1.0  -1.0
 +0.6  +0.4  +0.2  +0.0  -0.2  -0.4  -0.6  -0.8  -1.0  -1.0  -1.0
 +0.4  +0.2  +0.0  -0.2  -0.4  -0.6  -0.8  -1.0  -1.0  -1.0  -1.0
 +0.2  +0.0  -0.2  -0.4  -0.6  -0.8  -1.0  -1.0  -1.0  -1.0  -1.0
 +0.0  -0.2  -0.4  -0.6  -0.8  -1.0  -1.0  -1.0  -1.0  -1.0  -1.0
```

A.4 Fuzzy System Subroutine

```
c**********************************************************************
c
      subroutine flc(n,u,gu,fs,y,gy,by,rule_base,applied_fs)
c
c**********************************************************************
c**********************************************************************
c
c     Programmed  by     : Jeffery R. Layne
c     Date               : 8/29/91
c     Purpose            :
c
c     This subroutine is used to generate the input-output relationship
c     for a fuzzy system which  has symmetrical triangular shaped membership
c     functions on the input and output universes of discourse. The COG
c     defuzzification strategy is employed to generate the crisp  output.
c     This routine is capable of dealing with a maximum of 161051 rules
c     for a maximum of 5 inputs which for completeness is equivalent
c     to an average of 11 fuzzy sets associated with each of 5 inputs.
c     The input and output parameters to this subroutine are as follows:
c
c     Input Parameters
c
c         n         - The number of inputs to the fuzzy system.
c                     Maximum is 5 inputs. (integer)
c
c         u(i)      - The numerical value of input i. (real)
c
c         gu(i)     - The gain multiplying input u(i) so that u(i) maps to a
c                     normalized universe of discourse [-1,+1] (real). Any
c                     input which is mapped outside of the interval [-1,+1]
c                     is assumed to be mapped to +1 or -1 depending which
c                     is nearest.
c
c         fs(i)     - The odd number of fuzzy set defined on the normalized
c                     universe of discourse associated with input u(i).
c                     Fuzzy sets are assumed to be distributed such that
c                     all triangular membership functions intersect at 0.5.
c                     Maximum is 21 fuzzy sets on a given universe of
c                     discourse. (integer)
c
c         gy        - The gain multiplying the normalized output of the
c                     fuzzy system. (real)
c
c         by        - The normalized width of the base of the membership
c                     functions for all fuzzy sets associated with the
c                     output universe of discourse. (real)
c
c         rule_base(m) -
c                     Contains the center value of the symmetric triangular
c                     shaped membership functions implied by fuzzy sets
c                     fsi(j) associated with input u(i) where fsi(j) is
c                     a set of integers  such that
c                     -int(fs(i)/2)<=fsi<=int(fs(i)/2)
c                     for fs(i) odd. If a, b, c, d, and e denote the index
c                     for fuzzy sets associated by inputs 1 though 5,
c                     respectively, then m is computed by
c                     m = (a + int(fs(1)/2)*(fs(2)*...*fs(5)) +
c                     (b+int(fs(2)/2))*(fs(3)*...*fs(5)) +
c                     (c+int(fs(3)/2))*(fs(4)*fs(5)) +
c                     (d+int(fs(4)/2))*(fs(5)) + (e+int(fs(5)/2)) + 1.
```

```
c
c     Output Parameters
c
c          y         - The output of the fuzzy system. The maximum is gu. (real)
c
c          applied_fs(i,j,k) -
c                    Contains the index for all input fuzzy sets in
c                    which input u(i) has membership greater
c                    than 0.0 when k = 1 and the respective membership
c                    value when k=2. Give the way that the membership
c                    functions are distributed a particular input can be
c                    characterized by at most 2 fuzzy sets. Therefore j
c                    may only take on values 1 and 2. However, in the
c                    case that an input is described by only one
c                    fuzzy set both applied fuzzy set indexes will be
c                    the same. (real)
c
c*********************************************************************
c
c Define all variable
c
      integer  n, fs(5)
      real     u(5), gu(5), y, gy, by
      real     rule_base(161051), applied_fs(5,2,2)

      integer  input, count, index, index_fs(5),index_rb, index_add
      real     normal_u(5), center, truth_value, moment_sum, area_sum
      real     height, mult, area, moment, fs_step
c
c scale the inputs to the normalized universe of discourse
c
      do 10 input = 1,n
         normal_u(input) = gu(input) * u(input)
         if (normal_u(input) .gt. 1) normal_u(input) = 1
         if (normal_u(input) .lt. -1) normal_u(input) = -1
 10   continue
c
c find the array called applied_fs
c
      do 20 input = 1,n
         fs_step = 1/aint(floatj(fs(input))/2.0)
         index = int(normal_u(input)/fs_step)
         center = index*fs_step
         truth_value = 1.0 - abs((center-normal_u(input))/fs_step)
         applied_fs(input,1,1) = index
         applied_fs(input,1,2) = truth_value
         if (truth_value .eq. 1.0) then
            index = index
         else if (normal_u(input) .ge. 0.0) then
            index = index + 1
         else if (normal_u(input) .lt. 0.0) then
            index = index - 1
         end if
         center = index*fs_step
         truth_value = 1.0 - abs((center-normal_u(input))/fs_step)
         applied_fs(input,2,1) = index
         applied_fs(input,2,2) = truth_value
 20   continue
c
c find the "sum of the moments" and the "sum of the areas" to compute COG
c
      moment_sum = 0.0
```

```
      area_sum = 0.0
      do 30 count = 0,((2**n)-1)
         do 40 input = 1,n
            if (btest(count,(input-1))) then
               index_fs(input) = 1
            else if (.not. btest(count,(input-1))) then
               index_fs(input) = 2
            end if
            if ((index_fs(input) .eq. 2)
     #         .and. (applied_fs(input,1,1) .eq. applied_fs(input,2,1)))
     #            goto 30
 40         continue
            height = 1.0
            index_rb = 0
            do 50 input = 1,n
               height = amin1(height,applied_fs(input,index_fs(input),2))
               index_add = int(applied_fs(input,index_fs(input),1)
     #             + int(fs(input)/2))
               do 60 mult = (input + 1),n
                  index_add = index_add*fs(mult)
 60            continue
               index_rb = index_rb + index_add
 50         continue
            index_rb = index_rb + 1
            area = by*(height - (height*height)/2)
            moment = area*rule_base(index_rb)
            area_sum = area_sum + area
            moment_sum = moment_sum + moment
 30      continue
c
c compute the output of the fuzzy system
c
      y = moment_sum/area_sum * gy
      return
      end
```

A.5 Knowledge Base Modifier Subroutine

```
c**********************************************************************
c
      subroutine rbm(n,fs,y_change,gy,rule_base,applied_fs)
c
c**********************************************************************
c                        KNOWLEDGE BASE MODIFIER
c**********************************************************************
c
c     Programmed  by     : Jeffery R. Layne
c     Date               : 9/12/91
c     Purpose            :
c
c     Given all implication resulting from combinations of the input fuzzy
c     sets indexed in the applied_fs array, this subroutine modifies
c     the knowledge_base such that the implied output, when
c     COG defuzzification is employed, is changed by an amount given
c     by y_change. When using this subroutine for fuzzy adaptive control, the
c     applied_fs array contains all previously applied fuzzy sets which
c     created the process input and which must be change by an amount dictated
c     by y_change so that
c
c
```

```
c     Input Parameters
c
c         n        - The number of inputs to the fuzzy system.
c                    Maximum is 5 inputs. (integer)
c
c         fs(i)    - The odd number of fuzzy set defined on the normalized
c                    universe of discourse associated with the i^{th}
c                    input to the fuzzy system. Fuzzy sets are assumed
c                    to be distributed such that all triangular
c                    membership functions intersect at 0.5. (integer)
c
c         gy       - The gain multiplying the normalized output of the
c                    fuzzy system. (real)
c
c         y_change - The amount that the output of the fuzzy system
c                    must be changed given a COG defuzzification of all
c                    implied fuzzy sets resulting from all combinations
c                    of the indexes in the applied_fs array.(real)
c
c         applied_fs(i,j,k) -
c                    Similar to that defined in the "flc" subroutine.
c                    However, in this case it represents all the index for
c                    the input fuzzy sets such  that the resulting implied
c                    output must be changed by an amount given by y_change.
c                    (real)
c
c     Output Parameters
c
c         rule_base(m) -
c                    Contains the center value of the symmetric triangular
c                    shaped membership functions implied by fuzzy sets
c                    fsi(j) associated with input u(i) where fsi(j) is
c                    a set of integers  such that
c                    -int(fs(i)/2)<=fsi<=int(fs(i)/2)
c                    for fs(i) odd. If a, b, c, d, and e denote the index
c                    for fuzzy sets associated by inputs 1 though 5,
c                    respectively, then m is computed by
c                    m = (a + int(fs(1)/2)*(fs(2)*...*fs(5)) +
c                    (b+int(fs(2)/2))*(fs(3)*...*fs(5)) +
c                    (c+int(fs(3)/2))*(fs(4)*fs(5)) +
c                    (d+int(fs(4)/2))*(fs(5)) + (e+int(fs(5)/2)) + 1.
c
c*************************************************************************
c
c Define all variable
c
      integer  n, fs(5)
      real     y_change, gy, rule_base(161051), applied_fs(5,2,2)

      integer  input, count, index_fs(5), index_rb, index_add
      real     normal_y_change, mult, new_fs_center
c
c scale the change in output to the normalized universe of discourse
c

         normal_y_change= y_change/gy
c
c modify the appropriate entries in the rule base
c
      do 30 count = 0,((2**n)-1)
         do 40 input = 1,n
            if (btest(count,(input-1))) then
```

```
            index_fs(input) = 1
         else if (.not. btest(count,(input-1))) then
            index_fs(input) = 2
         end if
         if ((index_fs(input) .eq. 2)
     #      .and. (applied_fs(input,1,1) .eq. applied_fs(input,2,1)))
     #          goto 30
40       continue
         index_rb = 0
         do 50 input = 1,n
            index_add = int(applied_fs(input,index_fs(input),1)
     #          + int(fs(input)/2))
            do 60 mult = (input + 1),n
               index_add = index_add*fs(mult)
60          continue
            index_rb = index_rb + index_add
50       continue
         index_rb = index_rb + 1
         new_fs_center = rule_base(index_rb) + normal_y_change
         if (new_fs_center .gt. 1) new_fs_center = 1
         if (new_fs_center .lt. -1) new_fs_center = -1
         rule_base(index_rb) = new_fs_center
30    continue

      return
      end
```

BIBLIOGRAPHY

[1] J. Amerongen and A. Cate, "Model reference adaptive autopilots for ships," *Automatica*, vol. 11, pp. 441–449, 1975.

[2] K. Anderson, G. Blankenship, and L. Lebow, "A rule-based adaptive PID controller," *Proceedings, 1988 IEEE Conference on Decision and Control*, pp. 564–569, Austin, Texas, December 1988.

[3] K. Åström and B. Wittenmark, eds., *Adaptive Control.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1989.

[4] M. Barrére, A. Jaumotte, B. Veubeke, and J. Vandenkerckhove, eds., *Rocket Propulsion.* New York, New York: Elsevier Publishing Company, 1960.

[5] G. Bartolini, G. Casalino, F. Davoli, R. M. M. Mastretta, and E. Morten, "Development of performance adaptive fuzzy controllers with applications to continuous casting plants," *Industrial Application of Fuzzy Control*, pp. 73–86, 1985.

[6] C. Batur, A. Srinivasan, and C. Chan, "Automatic rule based model generation for uncertain complex dynamical systems," *Proceedings, 1991 IEEE International Symposium on Intelligent Control*, pp. 275–279, 1991.

[7] M. Bech and L. Smitt, "Analogue simulation of ship maneuvers," tech. rep., Hydro-og Aerodynmisk Laboratorium, Lyngby, Denmark, 1969.

[8] H. Berenji, Y. Chen, and R. Yager, "Using new aggregation operators in rule-based intelligent control," *Proceedings, 1990 IEEE Conference on Decision and Control*, pp. 2198–2203, Honolulu, Hawaii, December 1990.

[9] J. Bernard, "Use of a rule-based system for process control," *IEEE Control Systems Magazine*, vol. 8, pp. 3–13, October 1988.

[10] J. Buckley and H. Ying, "Linear fuzzy controller: It is a linear non–fuzzy controller," *Information Sciences*, vol. 51, pp. 183–192, 1990.

[11] R. Cannon, ed., *Dynamics of physical systems.* New York, New York: McGraw-Hill Book Company, 1967.

[12] Y. Chen and T. Tsao, "A description of the dynamical behavior of fuzzy systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, pp. 745–755, July/August 1989.

[13] S. Chiu, S. Chand, and D. Moore, "Fuzzy logic for control of roll and moment for a flexible wing aircraft," *IEEE Control Systems*, pp. 42–48, June 1981.

[14] A. Cumani, "On a possibilistic approach to the analysis of fuzzy feedback systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 12, pp. 417–422, May/June 1982.

[15] E. Czogała and W. Pedrycz, "On identification in fuzzy systems and its applications in control problems," *Fuzzy Sets and Systems*, vol. 6, pp. 73–83, 1981.

[16] E. Czogała and W. Pedrycz, "Control problems in fuzzy systems," *Fuzzy Sets and Systems*, vol. 7, pp. 257–273, 1982.

[17] H. Decker, R. Emig, and H. Goebels, "Anti-lock braking system for commercial vehicles," *SAE paper 881821*, 1988.

[18] S.Daley and K. F. Gill, "Comparison of a fuzzy logic controller with a P+D control law," *Journal of Dynamical System, Measurement, and Control*, vol. 111, pp. 128–137, June 1989.

[19] S.Daley and K. F. Gill, "Altitude control of a spacecraft using an extended self-organizing fuzzy logic controller," *Proc. I. Mech. E.*, vol. 201, no. 2, pp. 97–106, 1987.

[20] S.Daley and K. F. Gill, "A design study of a self-organizing fuzzy logic controller," *Proc. I. Mech. E.*, vol. 200, pp. 59–69, 1986.

[21] C. de Silva, "Simulation studies of analytical fuzzy tuner for a PID servo," *Proceedings of the 1991 American Control Conference*, pp. 2100–2105, June 1991.

[22] C. de Silva, "Fuzzy adaptation and control of a class of dynamical systems," *Proceedings, 5th IEEE International Symposium on Intelligent Control*, pp. 304–309, 1990.

[23] C. de Silva, "An analytical framework for knowledge-based tuning of servo controllers," *Engineering Applications of Artificial Intelligence*, vol. 4, no. 3, pp. 177–189, 1991.

[24] J. Farrell and W. Baker, "Learning control systems," in *An Introduction to Intelligent and Autonomous Control Systems* (P. Antsaklis and K. Passino, eds.), Kluwer Academic Publishing, 1992. (to be published).

[25] R. Fling and R. Fenton, "A describing-function approach to antiskid design," *IEEE Transactions on Vehicular Technology*, vol. 30, pp. 134–144, August 1981.

[26] B. Friedland, ed., *Control System Design.* New York, New York: McGraw Hill Book Company, 1986.

[27] E. Garcia-Benitez, S. Yurkovich, and K. Passino, "A fuzzy supervisor for flexible manipulator control," *Proceedings, 1991 IEEE International Symposium on Intelligent Control*, pp. 37–42, Arlington, Virginia, August 1991.

[28] E. Garcia-Benitez, S. Yurkovich, and K. Passino, "Rule-based supervisory control of a two-link flexible manipulator," *Journal of Intelligent and Robotic Systems*, (to appear in 1992).

[29] P. Graham and R. Newell, "Fuzzy adaptive control of a first-order process," *Fuzzy Sets and Systems*, vol. 31, pp. 47–65, 1989.

[30] P. Graham and R. Newell, "Fuzzy identification and control of a liquid level rig," *Fuzzy Sets and Systems*, vol. 8, pp. 255–273, 1988.

[31] R. Grimm, R. Bremer, F. Jain, and W. Levijoki, "Evaluation of vehicle installed wheel lock control hardware with a hybrid computer simulation," *SAE paper 770098*, 1977.

[32] R. Guntur and H. Ouwerkerk, "Adaptive brake control system," *Proceedings of the Institution of Mechanical Engineers*, vol. 186, pp. 855–880, 1972.

[33] M. Gupta, G. Saridis, and B. Gaines, eds., *Fuzzy Automation and Decision Process*. Amsterdam, the Netherlands: North Holland Publishing Company, 1977.

[34] J. Harned, L. Johnston, and G. Scharpf, "Measurement of tire brake force characteristics as related to wheel slip (antilock) control system design," *SAE paper 690214*, 1986.

[35] S. Hussain, "Digital algorithm design for wheel lock control system," *SAE paper 860509*, 1986.

[36] S. Isaka, A. Sebald, A. Karimi, N. Smith, and M. Quinn, "On the design and performance evaluation of adaptive fuzzy controllers," *Proceedings, 1988 IEEE Conference on Decision and Control*, pp. 1068–1069, Austin, Texas, December 1988.

[37] A. Jones, A. Kaufmann, and H. Zimmermann, eds., *Fuzzy Set Theory and Applications*. Dororecht, Holland: D. Reidel Publishing Company, 1986.

[38] H. Kang and G. Vachtsevanos, "Model reference fuzzy control," *Proceedings, 1989 IEEE Conference on Decision and Control*, pp. 751–755, Tampa, Florida, December 1989.

[39] W. Kickert and H. V. N. Lemke, "Application of a fuzzy controller in a warm water plant," *Automatica*, vol. 12, no. 4, pp. 301–308, 1976.

[40] W. Kickert and E. Mamdani, "Analysis of a fuzzy logic controller," *Fuzzy Sets and Systems*, vol. 1, pp. 29–44, 1978.

[41] J. Kiszka, M. Gupta, and P. Nikiforuk, "Energetistic stability of fuzzy dynamical systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, pp. 783–792, November/December 1985.

[42] M. Kokar, "Learning control methods, needs and architectures," in *An Introduction to Intelligent and Autonomous Control Systems* (P. Antsaklis and K. Passino, eds.), Kluwer Academic Publishing, 1992. (to be published).

[43] G. Langari, *A Framework for Analysis and Synthesis of Fuzzy Linguistic Control Systems*. PhD thesis, University of California, Berkeley, 1990.

[44] G. Langari, "Stability of fuzzy linguistic control systems," *Proceedings, 1990 IEEE Conference on Decision and Control*, pp. 2185–2190, December 1990.

[45] E. Leaphart, "A DSP based hybrid simulator for evaluating anti–lock brake system control designs," Master's thesis, The Ohio State University, 1991.

[46] C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller-part I," *IEEE Trans. on Systems, Man. and Cybernetics*, vol. 20, pp. 404–418, March/April 1990.

[47] C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller-part II," *IEEE Trans. on Systems, Man. and Cybernetics*, vol. 20, pp. 419–435, March/April 1990.

[48] H. Leiber and A. Czinczel, "Anti-skid system for passenger cars with a digital electronic control unit," *SAE paper 790458*, 1979.

[49] H. Leiber and A. Czinczel, "Four years of experience with 4–wheel antiskid brake systems (ABS)," *SAE paper 830481*, 1983.

[50] Y. Li and C. Lau, "Development of fuzzy algorithms for servo systems," *IEEE Control Systems Magazine*, vol. 9, pp. 65–72, April 1989.

[51] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Intl. Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[52] E. Mamdani, "Advances in the linguistic synthesis of fuzzy controllers," *Intl. Journal of Man-Machine Studies*, vol. 8, no. 6, pp. 669–678, 1976.

[53] G. Mandell, G. Caporaso, and W. Bengen, eds., *Topics in Advanced Model Rocketry.* Cambridge, Massachusetts: The MIT Press, 1973.

[54] D. Michie and R. Chambers, "Boxes: an experiment in adaptive control," in *Machine Intelligence 2* (E. Dale and D. Michie, eds.), American Elsevier Publishing Company, Inc., 1969.

[55] K. Narendra and A. Annaswamy, eds., *Stable Adaptive Systems.* Englewood Cliffs, New Jersey: Prentice Hall, 1989.

[56] H. van Nauta Lemke and W. De-zhao, "Fuzzy PID supervisor," *Proceedings, 1985 IEEE Conference on Decision and Control*, pp. 602–608, Ft. Lauderdale, FL, December 1985.

[57] P. Oliveira, P. Lima, and J. Sentieiro, "Fuzzy supervision of direct controllers," *Proceedings, 5th IEEE International Symposium on Intelligent Control*, pp. 638–643, 1990.

[58] A. Ollero and A. Garcia-Cerezo, "Direct digital control, auto-tuning and supervision using fuzzy logic," *Fuzzy Sets and Systems*, vol. 30, no. 2, pp. 135–153, 1989.

[59] T. Procyk and E. Mamdani, "A linguistic self-organizing process controller," *Automatica*, vol. 15, no. 1, pp. 15–30, 1979.

[60] K. Ray and D. Majumder, "Application of the circle criteria for the stability of linear SISO and MIMO systems associated with fuzzy logic controllers," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 14, pp. 345–349, March/April 1984.

[61] K. Ray, A. Ghosh, and D. Majumder, "L_2-stability and the related concepts for SISO linear systems associated with fuzzy logic controllers," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 14, pp. 932–939, November/December 1984.

[62] S. Rhee, "Friction coefficient of automotive friction materials – its sensitivity to load speed and temperature," *SAE paper 740415*, 1974.

[63] G. N. Saridis, ed., *Self-Organizing Control of Stochastic Systems.* New York, New York: Marcel Dekker, Inc., 1977.

[64] E. Scharf and N. Mandic, "The application of a fuzzy controller to the control of a multi-degree-of-freedom robot arm," in *Industrial Applications of Fuzzy Control*, pp. 41–62, Amsterdam, the Netherlands: M. Sugeno (ed.), 1985.

[65] K. Self, "Designing with fuzzy logic," *IEEE Spectrum*, pp. 42–105 and 105, November 1990.

[66] S. Shao, "Fuzzy self-organizing controller and its application for dynamic processes," *Fuzzy Sets and Systems*, vol. 26, pp. 151–164, 1988.

[67] C. Shih, W. Gruver, and Y. Zhu, "Fuzzy logic force control for a biped robot," *Proceedings, 1991 IEEE International Symposium on Intelligent Control*, pp. 269–274, Arlington, Virginia, August 1991.

[68] J. Slotine and W. Li, eds., *Applied Nonlinear Control.* Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[69] S. Smith and D. Comer, "Automated calibration of a fuzzy logic controller using a cell state space algorithm," *IEEE Control Systems*, vol. 11, pp. 18–28, August 1991.

[70] E. Stewart and L. Bowler, "Road testing of wheel slip control systems in the laboratory," *SAE paper 690215*, 1969.

[71] M. Sugeno, "An introductory survey of fuzzy control," *Information Sciences*, vol. 36, no. 1, pp. 59–83, 1985.

[72] M. Sugeno and M. Nishida, "Fuzzy control of model car," *Fuzzy Sets and Systems*, vol. 16, pp. 103–113, 1985.

[73] T. Tabo, N. Ohka, H. Kuraoka, and M. Ohba, "Automotive antiskid system using modern control theory," *IECON*, pp. 390–395, 1985.

[74] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[75] H. Takahashi and Y. Ishikawa, "Anti-skid braking control system based on fuzzy inference." *U.S. Patent*, Number*4842342*, June 1989.

[76] H. Takahashi, "Automatic speed control device using self-tuning fuzzy logic," *1988 IEEE Workshop on Automotive Applications of Electronics*, pp. 65–71, Dearborn, Michigan, October 1988.

[77] H. Tan and M. Tomizuka, "A discrete-time robust vehicle traction controller design," *1988 American Controls Conference*, vol. 3, pp. 1053–1058, Pittsburgh, Pennsylvania, June 1989.

[78] H. Tan and M. Tomizuka, "Discrete-time controller design for robust vehicle traction," *IEEE Control Systems Magazine*, pp. 107–113, April 1990.

[79] R. Tanscheit and E. Scharf, "Experiments with the use of a rule-based self-organising controller for robotics applications," *Fuzzy Sets and Systems*, vol. 26, pp. 195–214, 1988.

[80] R. M. Tong, "A control engineering review of fuzzy systems," *Automatica*, vol. 13, pp. 559–569, November 1977.

[81] R. M. Tong, "Some properties of fuzzy feedback systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, pp. 327–330, June 1980.

[82] S. Tzafestas and N. Papanikolopoulos, "Incremental fuzzy expert PID control," *IEEE Transactions on Industrial Electronics*, vol. 37, pp. 365–371, October 1990.

[83] L. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *Proceedings, 1991 IEEE International Symposium on Intelligent Control*, pp. 263–268, Arlington, Virginia, August 1991.

[84] F. V. D. Rhee, H. V. N. Lemke, and J. Dijkman, "Knowledge based fuzzy control of systems," *IEEE Transactions on Automatic Control*, vol. 35, pp. 148–155, February 1990.

[85] J. Vijayvargiya, "Identification and control of automotive brake systems," Master's thesis, The Ohio State University, 1991.

[86] T. Yamazaki, *An improved algorithm for a self-organizing controller and its experimental analysis.* PhD thesis, London University, 1982.

[87] H. Ying, W. Siler, and J. Buckley, "Fuzzy control theory: A nonlinear case," *Automatica*, vol. 26, no. 3, pp. 513–520, 1990.

[88] S. Yoneda, Y. Naitoh, and H. Kigoshi, "Rear brake lock-up control system of Mitsubishi Starion," *SAE paper 830482*, 1983.

[89] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. on Systems, Man. and Cybernetics*, vol. 3, no. 1, pp. 28–44, 1973.

[90] L. A. Zadeh, "Fuzzy logic," *IEEE Computer*, pp. 83–93, April 1988.