DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
COLLEGE OF ENGINEERING & TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

DESIGN AND IMPLEMENTATION OF FUZZY LOGIC CONTROLLERS

By

Osama A. Abihana, Graduate Research Assistant

Principal Investigator: Oscar R. Gonzalez

Final Report
For the period January 1, 1993

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

Under
P.O. #L20278D
Capt. Gregory W. Walker, Point of Contact
U.S. Army

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508-0369

July 1993

ACKNOWLEDGEMENTS

This is a thesis being submitted in lieu of a final report for the research project entitled "Design and Implementation of Fuzzy Logic Controllers" for the period July 27, 1992 to January 1, 1993. This work was supported by the NASA Langley Research Center through purchase order no. L20278D, point of contact Capt. Gregory W. Walker, U.S. Army, NASA Langley Research Center, Mail Stop 286.

ABSTRACT

The main objectives of our research are to present a self-contained overview of fuzzy sets and fuzzy logic, develop a methodology for control system design using fuzzy logic controllers, and to design and implement a fuzzy logic controller for a real system. In this thesis we first present the fundamental concepts of fuzzy sets and fuzzy logic. Fuzzy sets and basic fuzzy operations are defined. In addition, for control systems, it is important to understand the concepts of linguistic variables, linguistic values, term sets, fuzzy rule base, inference methods, and defuzzification methods. Second, we introduce a four-step fuzzy logic control system design procedure. The design procedure is illustrated via four examples, showing the capabilities and robustness of fuzzy logic control systems. This is followed by a tuning procedure that we developed from our design experience. Third, we present two Lyapunov based techniques for stability analysis. Finally, we present our design and implementation of a fuzzy logic controller for a linear actuator to be used to control the direction of the Free Flight Rotorcraft Research Vehicle at NASA Langley Research Center.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Fuzzy logic controllers (FLCs) are used in control system design for processes that do not admit a mathematical model or where the data is imprecise. FLCs are fuzzy expert systems that can model the human operator of a process. They are based on a linguistic description of the process variables. We first present the fundamental concepts of fuzzy sets and fuzzy logic. Fuzzy sets and basic fuzzy operations are defined. In addition, for control systems, it is important to understand the concepts of linguistic variables, linguistic values, term sets, fuzzy rule bases, inference methods, and defuzzification methods. Second, we introduce a basic four step fuzzy logic control system design procedure. The design procedure is illustrated via four examples showing the capabilities and robustness of fuzzy logic control systems. This is followed by a procedure to tune fuzzy logic controllers. Third, for completeness we present two Lyapunov-based

techniques for stability analysis. This is an important area for future research, since these techniques cannot currently be applied to our control designs. Finally, we present the design and implementation of a fuzzy logic controller for a linear actuator to be used to control the direction of the Free Flight Rotorcraft Research Vehicle in NASA Langley Research Center.

1.2 Overview of Fuzzy Controls

Since the development of fuzzy set theory by Lofti Zadeh (Zadeh, 1965), it was found that it is well suited for control applications. In 1974 the first fuzzy controller was developed by Mamdani to control a small laboratory steam engine (Mamdani, 1974). The purpose was to regulate engine speed and boiler steam pressure by using heat applied to the boiler and the throttle setting on the engine. Because of the successes of these experiments conducted by Mamdani and co-workers, an interest in fuzzy logic control was generated. Since that first application of fuzzy set theory many others have been developed and tested to prove the worthiness of those controllers.

Fuzzy control generated a lot of enthusiasm because it can be applied to processes where other control techniques were not efficient or simply

failed to do the job. Moreover, fuzzy logic controllers require no mathematical model but rather a linguistic description of the process. The control task is achieved through the use of fuzzy sets and a series of IF..THEN rules that capture human expertise. Those rules when applied to a control process would, in turn, give the desired control input to the process.

One drawback of fuzzy control is the lack of systematic techniques to fine tune the controller to attain best results. This tuning process is sometimes difficult and time consuming. In this thesis we introduce a procedure to achieve the desired response when tuning fuzzy controllers. The tuning procedure includes reshaping of the membership functions to achieve the desired response.

1.3 Thesis Structure

The organization of the chapters in the thesis follows.

The theory of fuzzy sets and fuzzy logic as it directly relates to the design of fuzzy logic controllers is presented in Chapter Two. This includes a comparison between two-valued logic and fuzzy logic to help the reader understand and appreciate the theory of fuzzy sets. Moreover, terms that relate to fuzzy control are defined and illustrated graphically. An example

is presented to help in understanding the application of fuzzy sets and fuzzy logic.

Chapter Three is an introduction to the design of fuzzy logic controllers. In this chapter, a procedure for the design of fuzzy logic controllers is presented followed by examples. In the examples, a fuzzy logic control system is designed for single-input, single-output, linear, time invariant, and continuous plants. Second-order type 0 and 1 systems are considered, as well as a non-minimum phase plant. The design of a controller for these examples and the way to approach them is explained. Moreover, techniques for tuning fuzzy controllers are also presented along with experimental data to test the robustness of the controllers.

Chapter Four is an overview of the stability analysis of fuzzy controllers. In this chapter, two approaches to stability analysis are discussed briefly. However, these approaches are all aimed at specific classes of problems. This implies that until now there has been no general method for analyzing the stability of fuzzy logic controllers.

A real-world problem is solved and its controller design is presented in Chapter Five. The design of a fuzzy logic controller to control a linear actuator to be used on the Free Flight Rotorcraft Research Vehicle (FFRRV)

is included with the specifications and system requirements. This is followed by testing the controller under various load conditions to show its robustness.

The conclusion of the thesis and topics for further research are presented in Chapter Six.

Fuzzy sets and the rule base used in the design of the controllers in Chapters Two, Three, and Five are included before and after tuning along with the C code listings that were written for the simulation of the closed-loop systems in appendices at the end of each chapter.

CHAPTER TWO

FUZZY SETS AND FUZZY LOGIC IN CONTROL SYSTEMS

2.1 Introduction

The theory of fuzzy sets was developed in 1965 by Lofti Zadeh of the University of California at Berkeley (Zadeh, 1965). It represents a generalization of conventional set theory, making it more applicable in the solution of real-world problems. In particular, fuzzy sets and fuzzy logic may be used to make decisions with uncertain data. In addition to its real world applicability, fuzzy set theory is now an important area of research in mathematics (Dubois and Prade, 1980; Kandel and Lee, 1979). In this chapter, the main concepts of fuzzy sets and fuzzy logic that are used in control system design are presented.

In order to introduce and appreciate fuzzy sets, consider first the conventional sets which are based on two-valued logic. In conventional set theory, the objects of the universal set belong to or do not belong to specific

sets. This is due to the fact that two-valued logic imposes or dictates that we assign an object to one of two categories, for example, 0-1, good-bad, odd-even, black-white, etc. This type of classification can be easily performed on processes that are precise and well defined. Such a process is the classification of numbers as odd or even. However, many engineering categories are ill-defined, for example, warm, hot, fast, turbulent, near, tall, etc. Notice that the terms in the above example are all relative. For instance what one person might consider tall, another person might consider medium height or what is considered warm by an individual could be classified as very warm by another individual.

A historical example of fuzzy notation comes from an ancient Greek sophism and can best illustrate the classification dilemma (Pedrycz, 1989),

> "...one seed does not constitute a pile nor two nor three .. from the other side everybody agrees that a 100 million seeds constitute a pile. What therefore is the appropriate limit? Can we say that 325,647 seeds don't constitute a pile but 325,648 do?"

From the above discussion we see the need to assign to an object some degree of belonging to a set. The concept of a fuzzy set formulated by Zadeh did just that and is introduced in the next section.

2.2 Fuzzy Sets

2.2.1 Definition of Fuzzy Sets

Fuzzy set theory generalizes the original concept of a set to allow the grade of belonging to the set, which varies from full exclusion (0) up to complete membership (1). The higher the value of the membership of a certain object x to the fuzzy set A, $\mu_A(x)$, the stronger the link of x to the category described by A. For example, defining the fuzzy notion of warm expressed in degrees we can say that a temperature of 80 degrees belongs to the set warm with a grade of 1.0 while the temperature of 70 degrees belongs to the set warm with a grade of 0.50, and the temperature of 50 degrees belongs to the set warm with a grade of 0.0. From the above example, the set of all possible temperatures forms a universal set or a universe of discourse X. The universe of discourse can be discrete or continuous. In this example X=[40°, 100°], the closed interval on the real line, denoting temperature in degrees Fahrenheit.

Formally, a fuzzy set A on a universe of discourse X is characterized by a membership function $\mu_A(x)$ which maps elements of X into the closed real interval [0,1] as follows:

$$\mu_A: X \rightarrow [0, 1],$$

where $\mu_A(x)$ expresses the degree that x belongs to some category A. For simplicity if there is no confusion, the membership function μ_A will be simply denoted by A. A fuzzy set A can be represented by an ordered pair

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

Hence, a fuzzy set can be viewed by plotting x versus $\mu_A(x)$. In addition, for discrete X, a table can also be used to represent a fuzzy set. In conventional set theory, there is also a membership function called the characteristic function which can only take two values 0 or 1, denoting exclusion or inclusion, respectively, of an object in a set.

For example, consider the universe of discourse X of temperatures in [40°, 100°]. We can define two fuzzy sets WARM and HOT on X which are characterized by their respective membership functions. Figure 2.1 shows the membership functions for WARM and HOT. Additional membership functions are left out, such as VERY HOT. Given that x=65° we determine the degree x belongs to each fuzzy set by finding the point at which x intersects the membership function, that is, finding the value of WARM(x) and HOT(x). In this case a temperature of 65° belongs to the fuzzy set WARM with a degree of 0.75 and to HOT with a degree of 0.25. In conventional sets, the characteristic function leads to sharp boundaries as

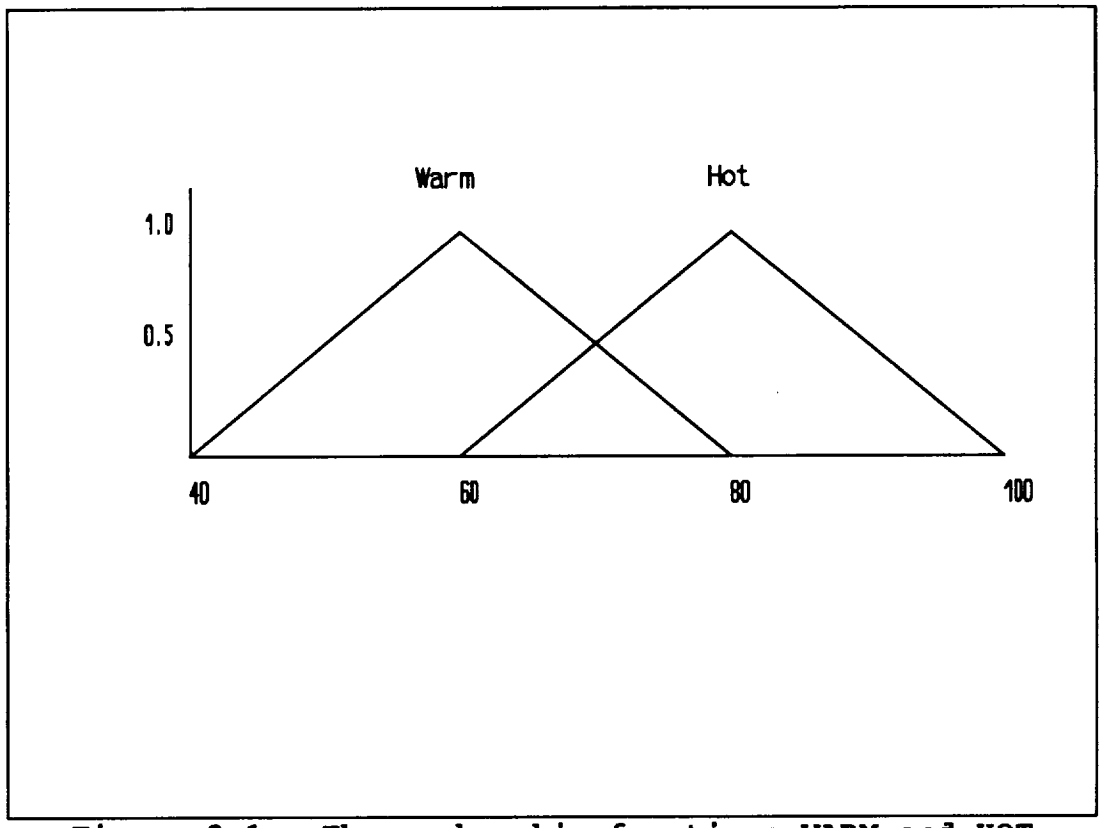

Figure 2.1. The membership functions WARM and HOT.

seen in Figure 2.2. In this example a triangular membership function was used. Triangular and trapezoidal membership functions are commonly used in fuzzy control applications. Other membership functions include gaussian membership functions and monotonically increasing and decreasing functions.

2.2.2 Fuzzy Set Operations

All conventional set operations have been generalized to fuzzy sets. In fact, when a fuzzy set operation is performed on a conventional set, the conventional set result is obtained (Zadeh, 1965; Kandel and Lee, 1979). In this section we present three basic operations including some of their properties.

In set theory the operations of union, intersection, and complement are denoted by $A \cap B$, $A \cup B$, and NOT A, respectively. The operations are defined as follows:

$$A \cup B = \{x \in \mathbf{X} \mid x \in A \text{ or } x \in B\}, \tag{2.1}$$

$$A \cap B = \{x \in \mathbf{X} \mid x \in A \text{ and } x \in B\}, \tag{2.2}$$

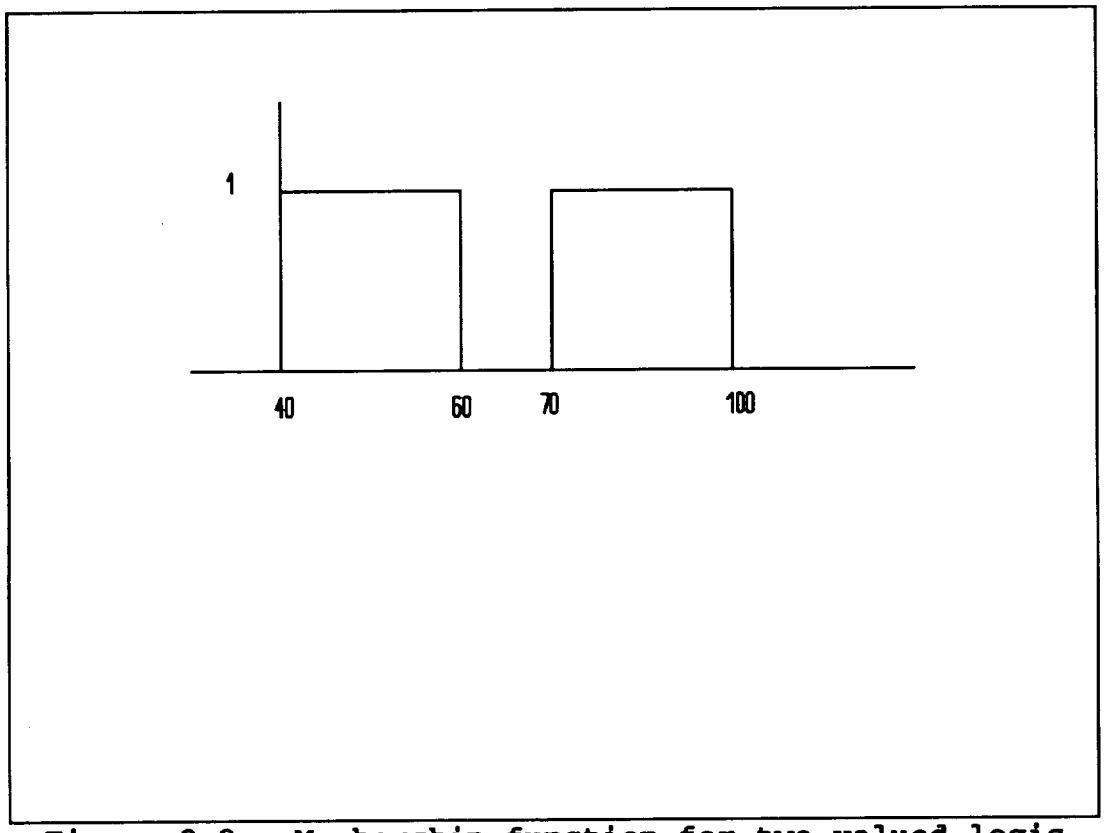

Figure 2.2. Membership function for two valued logic.

and

$$\overline{A} = \{x \in X \mid x \notin A\}. \tag{2.3}$$

Operations in fuzzy sets are defined in terms of their fuzzy membership functions (Zadeh, 1965). In particular, for fuzzy sets A and B on X the above operations in (2.1), (2.2), and (2.3) become

$$(A \cup B)(x) = \max(A(x), B(x)) \quad x \in X, \tag{2.4}$$

$$(A \cap B)(x) = \min(A(x), B(x)) \quad x \in X, \tag{2.5}$$

and

$$\overline{A}(x) = 1 - A(x) \quad x \in X, \tag{2.6}$$

where the union operator corresponds to the OR function, the intersection operator corresponds to the AND function, and the complement operator corresponds to the NOT function. Notice that the result of the three fuzzy set operations in (2.4) - (2.6) is a new fuzzy set. The three fuzzy operations in (2.4) - (2.6) are illustrated in Figures 2.3 - 2.5 for the fuzzy sets WARM

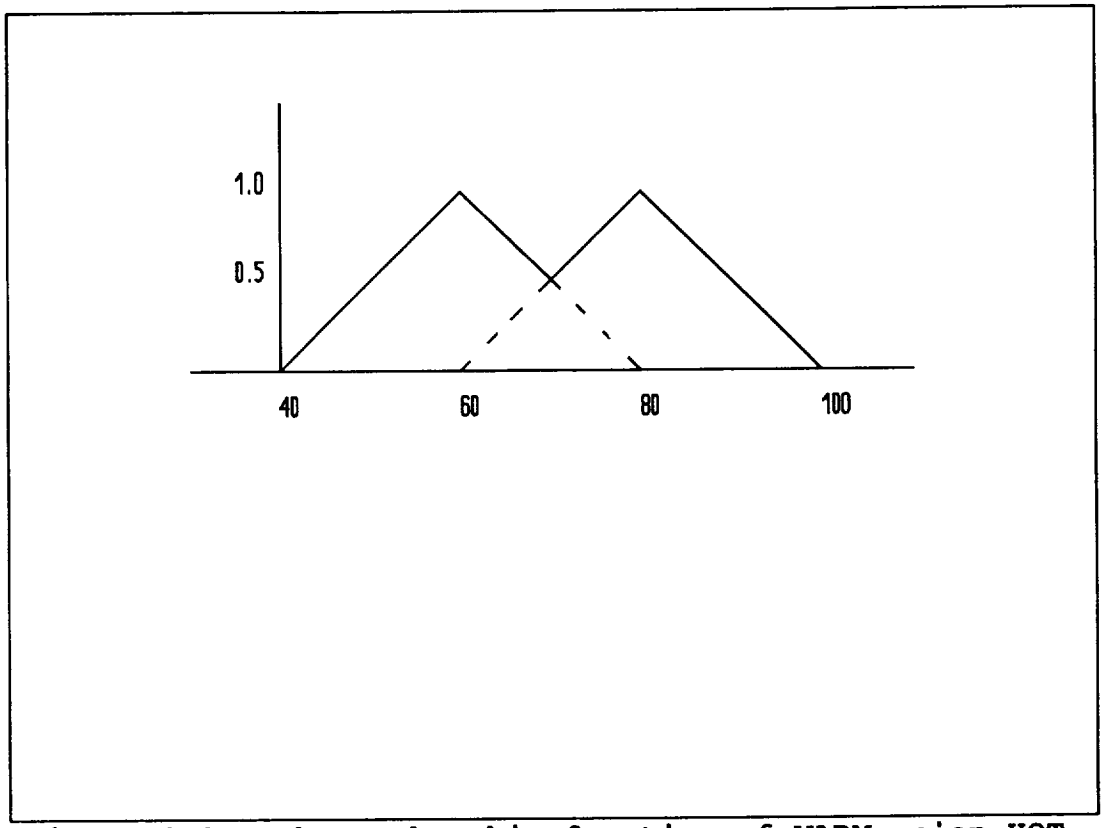

Figure 2.3. The membership function of WARM union HOT.

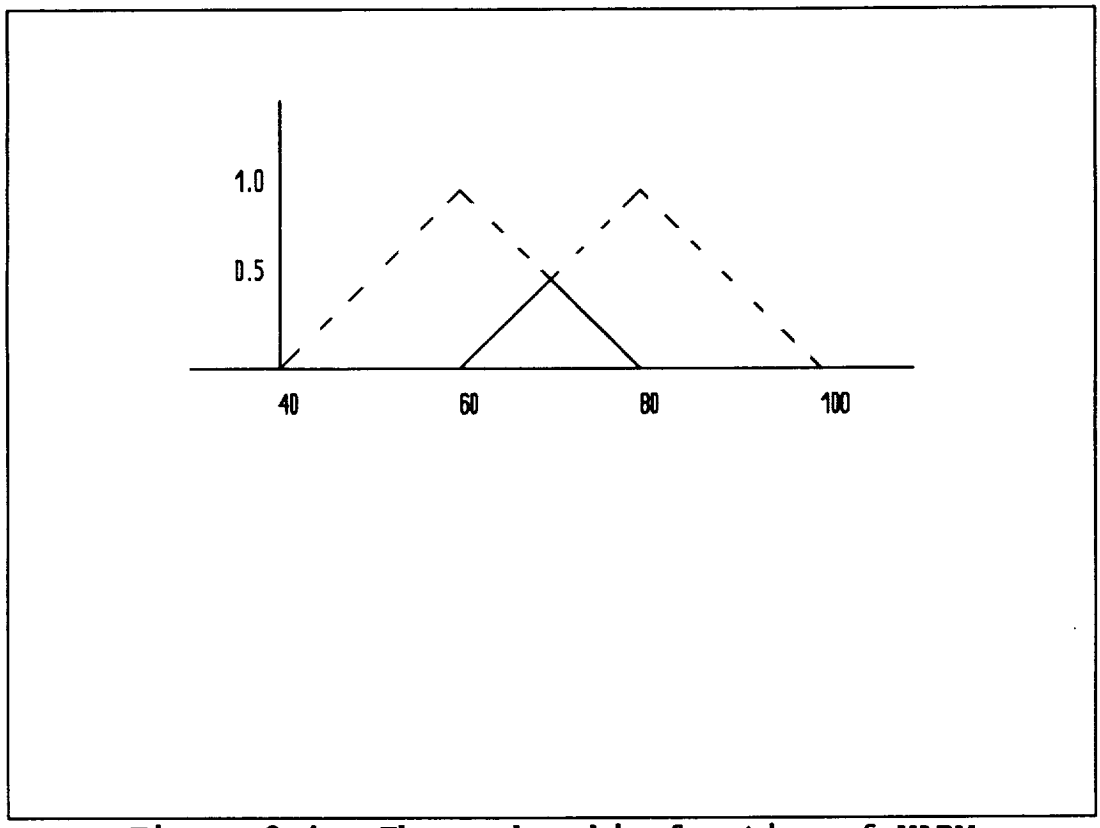

Figure 2.4. The membership function of WARM intersection HOT.

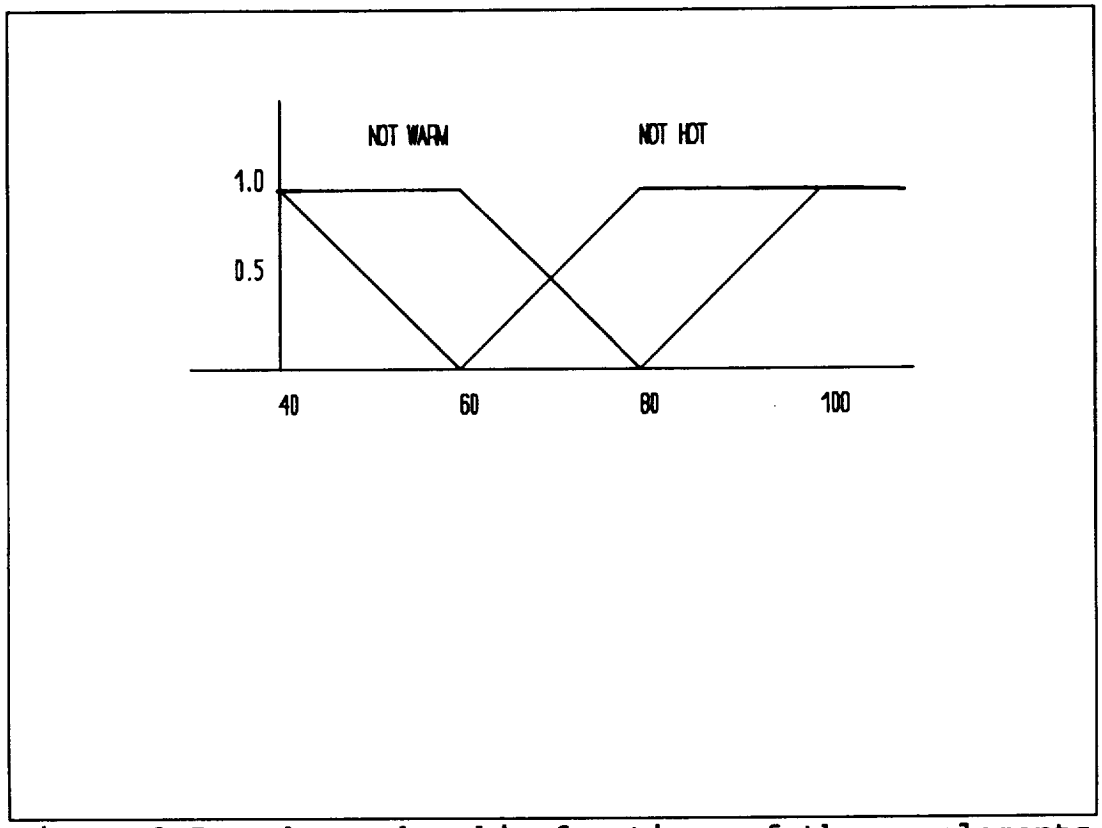

Figure 2.5. The membership functions of the complements of WARM and HOT.

and HOT defined in Section 2.2.1. To aid in the interpretations of Figures 2.3 - 2.5 consider the following subset of temperatures

$$T = \{60^\circ, 65^\circ, 70^\circ, 75^\circ, 80^\circ\}.$$

The degrees of the membership of these temperatures to the fuzzy sets WARM and HOT are given by

$$\text{WARM}(T) = \{1, 0.75, 0.5, 0.25, 0\}$$

and

$$\text{HOT}(T) = \{0, 0.25, 0.5, 0.75, 1\}.$$

The fuzzy operations in (2.4) - (2.6) must be satisfied for all $x \in X$. In particular, they must be satisfied for $x \in T$. The degree of membership of temperatures in T to the fuzzy sets that result from (2.4) - (2.6) is given by

$(\text{WARM} \cup \text{HOT})(T) =$

$\{\max(1,0), \max(0.75,0.25), \max(0.5,0.5), \max(0.25,0.75), \max(0, 1)\}$

$= \{1, 0.75, 0.5, 0.75, 1\};$

$(\text{WARM} \cap \text{HOT})(T) =$

$\{\min(1,0), \min(0.75,0.25), \min(0.5,0.5), \min(0.25, 0.75), \min(0, 1)\}$

$= \{0, 0.25, 0.5, 0.25, 0\};$

$(\text{NOT WARM})(T) = \{0, 0.25, 0.5, 0.75, 1\};$

and

(NOT HOT) (T) = {1, 0.75, 0.5, 0.25, 0}.

For completeness, some fuzzy set properties in the form of theorems are stated below. For further information see Zadeh, (1965) or Pedrycz, (1989). First, two fuzzy sets are equal if and only if their membership functions are identical for all $x \in X$. De Morgan's Laws are given by

$$\overline{(A \cap B)}(x) = \bar{A}(x) \cup \bar{B}(x) \tag{2.7}$$

and

$$\overline{(A \cup B)}(x) = \bar{A}(x) \cap \bar{B}(x). \tag{2.8}$$

De Morgan's laws can be easily proven using the basic operations in (2.4)-(2.6). The first, becomes

$$1 - \text{Max}\{A(x), B(x)\} = \text{Min}\{1 - A(x), 1 - B(x)\}.$$

In order to verify the equality, the two possible cases: $A(x) > B(x)$ and $A(x) < B(x)$ need to be tested. If $A(x) > B(x)$ we have,

$$1 - A(x) = 1 - A(x);$$

while if $A(x) < B(x)$ we have,

$$1 - B(x) = 1 - B(x).$$

The distributive laws are given by

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{2.9}$$

and

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C). \tag{2.10}$$

The properties of absorption and idempotency also hold:

$$(A \cap B) \cup A = A, \tag{2.11}$$

$$(A \cup B) \cap A = A, \tag{2.12}$$

$$A \cup A = A, \tag{2.13}$$

and

$$A \cap A = A. \tag{2.14}$$

However, the following laws are not satisfied,

$$A \cup \overline{A} \neq X \tag{2.15}$$

and

$$A \cap \overline{A} \neq 0. \tag{2.16}$$

This is expected since fuzzy sets do not impose that an object take on one of two values. To illustrate why the laws are not satisfied consider the fuzzy

set WARM. The membership functions of WARM $\cup$ NOT WARM and WARM $\cap$ NOT WARM are given in Figures 2.6 and 2.7. In order for a fuzzy set to be universal, that is, be equal to its universe of discourse, its membership function should be unity on X. In addition, a fuzzy set is empty if and only if its membership function is zero on X.

Before we move on to the application of the above statements we need to define the concept of linguistic variables which is the corner stone of fuzzy logic control.

2.2.3 Linguistic Variables

A linguistic variable is a variable whose value is represented with words rather than with numbers. In control applications, the measured variables are considered to be linguistic variables. For example, in a statement such as the **temperature** is **hot**, we are saying that the linguistic variable **temperature** has the linguistic value **hot**. The linguistic value **hot** is a fuzzy set in the universe of discourse of temperatures. In general, a linguistic variable with universe of discourse X may take on several linguistic values. The set of linguistic values is referred to as the term set of the linguistic variable. Since each linguistic value is a fuzzy set on X, the

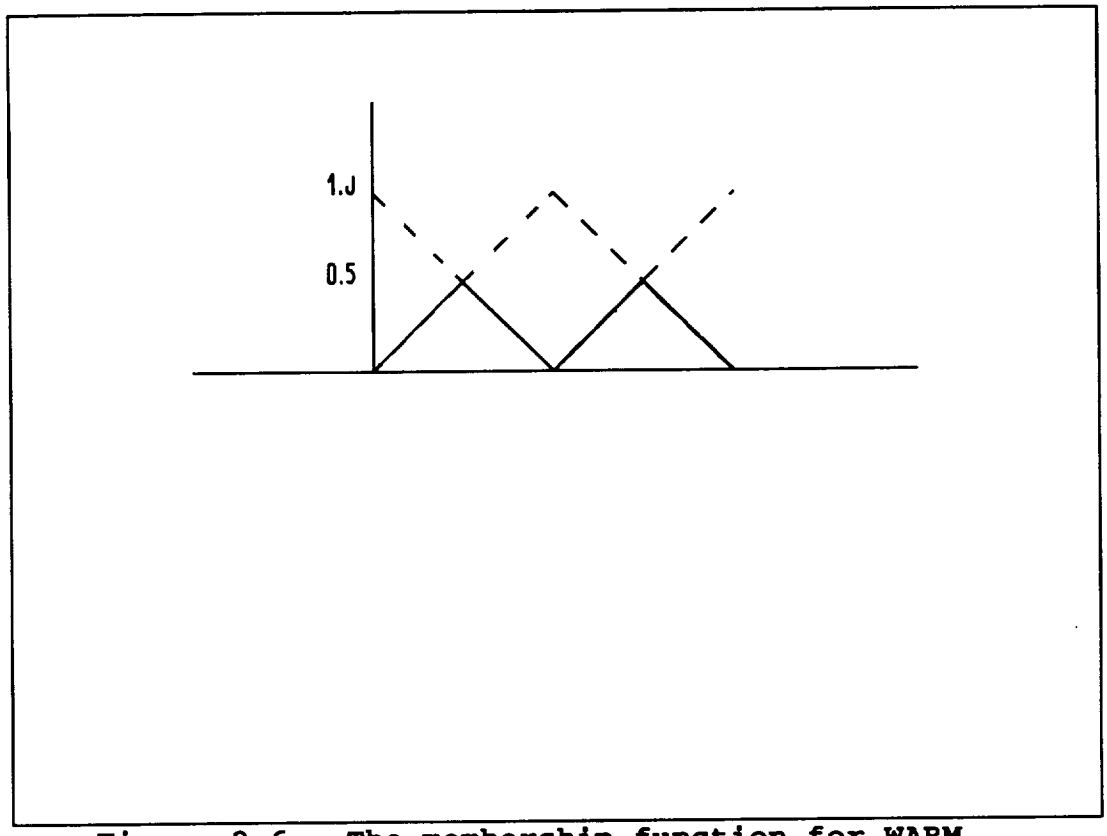

Figure 2.6. The membership function for WARM intersection NOT WARM.

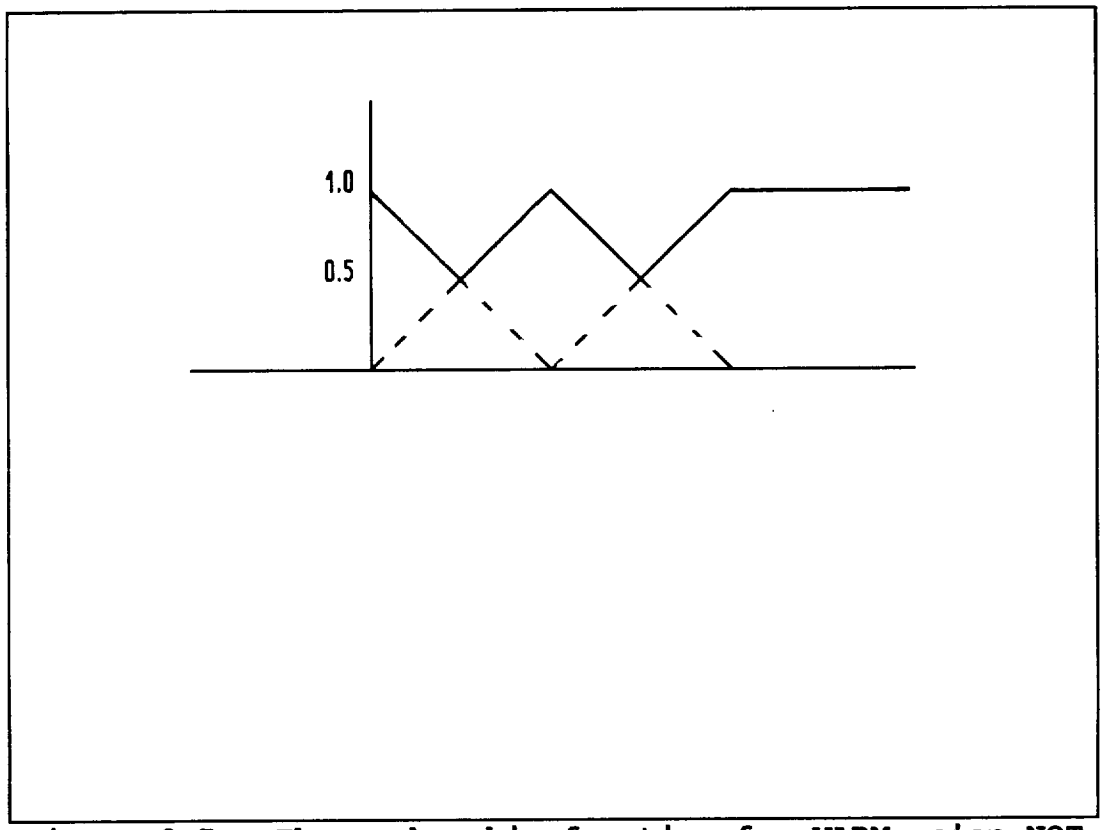

Figure 2.7. The membership function for WARM union NOT WARM.

term set represents a fuzzy partitioning of X, where the membership functions of the linguistic values are made to overlap. An example of some linguistic values are:

PB: positive big,

PS: positive small,

Z: zero,

NS: negative small,

and

NB: negative big.

A finer granularity can be obtained by considering more linguistic values.

2.3 Fuzzy Logic

2.3.1 Fuzzy Rule Base

In order to apply fuzzy sets, a fuzzy rule base needs to be specified. This rule base can be provided to the system, for example, by a human expert or learned by an artificial neural network. Some of the methods used to derive a fuzzy rule base for control system applications are presented in Lee (1990) and Kosko (1992). The fuzzy control or production rules in the rule base are of the form (Hill, Horstkotte, and Teichrow, 1990):

IF premise THEN consequence, **(2.17)**

where the premise is a set of conditions to be specified and the consequence is a set of actions to be taken. The premise and the consequence are fuzzy relations represented by linguistic variables and their linguistic values. For example, let L_1, L_2, and L_3 be three linguistic variables defined on x_1, x_2, and x_3, respectively. Let x_1, x_2, and x_3 be samples of L_1, L_2, and L_3, respectively, where x_1 and x_2 are known, and x_3 is to be determined. In addition, let the term sets for L_1, L_2, and L_3 be given by $\{A_1,..., A_{n1}\}$, $\{B_1,..., B_{n2}\}$, and $\{C_1,..., C_{n3}\}$, respectively. Then the i^{th} fuzzy rule is of the form

$$R_i: \text{IF } L_1 \text{ is } A^i \text{ AND } L_2 \text{ is } B^i \text{ THEN } L_3 \text{ is } C^i \qquad (2.18)$$

Additional linguistic variables in the premise and consequence can be easily taken into account. The premise of R_i is a fuzzy relation defined on the cartesian product $X_1 \times X_2$. This fuzzy relation is a fuzzy set and it can be represented by

$$\text{Premise} = \{((x_1,x_2), \text{Premise}(x_1,x_2)) \mid (x_1,x_2) \in X_1 \times X_2\}, \qquad (2.19)$$

where we can take

$$\text{Premise}(x_1, x_2) = \min (A^i(x_1), B^i(x_2)). \qquad (2.20)$$

In this case, it is assumed that the fuzzy sets in the premise are combined conjunctively with the AND operation. It is also possible to use, for

example, the OR operation to combine the fuzzy sets. The operation depends on the inference method chosen. This is discussed further in the next section.

An example of a rule to control a linear actuator is

R_1: IF E is PB AND ΔE is PS

THEN control input is PB.

Here E, ΔE, and control input are the linguistic variables and PB, PS, and PB linguistic values in the term sets of each linguistic variable.

2.3.2 Inference Methods

The process of applying the degree of membership computed for a production rule premise to the rule's conclusion to determine the action to be taken is called performing an inference. One is inferring the action to be taken from the premise. In the example in the previous section, each fuzzy rule R can be considered to be a fuzzy implication

$$A_i \times B_i \rightarrow C_k, \tag{2.21}$$

which is a fuzzy set. This fuzzy implication is a fuzzy set in $X_1 \times X_2 \times X_3$ with membership function

$$R(x_1, x_2, x_1) = L_1(x_1) * L_2(x_2) * L_3(x_3), \tag{2.22}$$

where the most commonly used operations for * are product and union (Lee, 1990).

The two main inference methods that are used in applying fuzzy logic are the max-min inference method and the max-dot inference method (Hill, Horstkotte, and Teichrow, 1990). In Kosko (1992) these two inference methods are referred to as correlation-minimum and correlation-product encoding, respectively. In either inference method, the basic concept is that the value to be assigned to the output is either scaled (max-dot) or clipped (max-min) to the degree of membership for the premise. All the clipped or scaled sets for all the rules that set this output are then combined together to form the final output membership function. In reality, both methods give very similar results. However, when it comes to computer implementation the max-dot method is preferred because it is much faster computationally than the max-min method. Both inference methods are illustrated in Figures 2.8 and 2.9 (Hill, Horstkotte, and Teichrow, 1990). In Figures 2.8 and 2.9 the output can be computed by taking the samples Temperature and Pressure and observing to which sets they belong. From the appropriate rules will fire and we can find the degree of fulfillment of each rule by applying the AND (min) or the OR (max) operations. The output fuzzy set is then scaled or

clipped by the result of the min or max operation. Figures 2.8 and 2.9 show both the AND (min) and the OR (max) operations on both inference methods.

The following comments apply to the max-dot inference method. The membership function of the i^{th} fuzzy implication is given by

$$R_i(x_1,x_2,x_3) = \min(\ A^i(x_1),\ B^i(x_2))\times C^i, \tag{2.23}$$

which is the membership function of the consequence scaled by the weight $w_i = \min(A^i(x_1),\ B^i(x_2))$. If there are n fuzzy rules, then there are n fuzzy sets R_i. A method is needed to determine a combined fuzzy set for the rule base which will be called the output fuzzy set. One approach is to let

$$O = R_1 U R_2 U \ldots U R_n. \tag{2.24}$$

A better approach (Kosko, 1992) is to add the membership functions as follows

$$\mu_O = \sum_{i=1}^{n} w_i C^i. \tag{2.25}$$

The latter approach is preferred, in particular, as the number of fuzzy rules in the rule base grows. Note that the universe of discourse of the output fuzzy set is X_3.

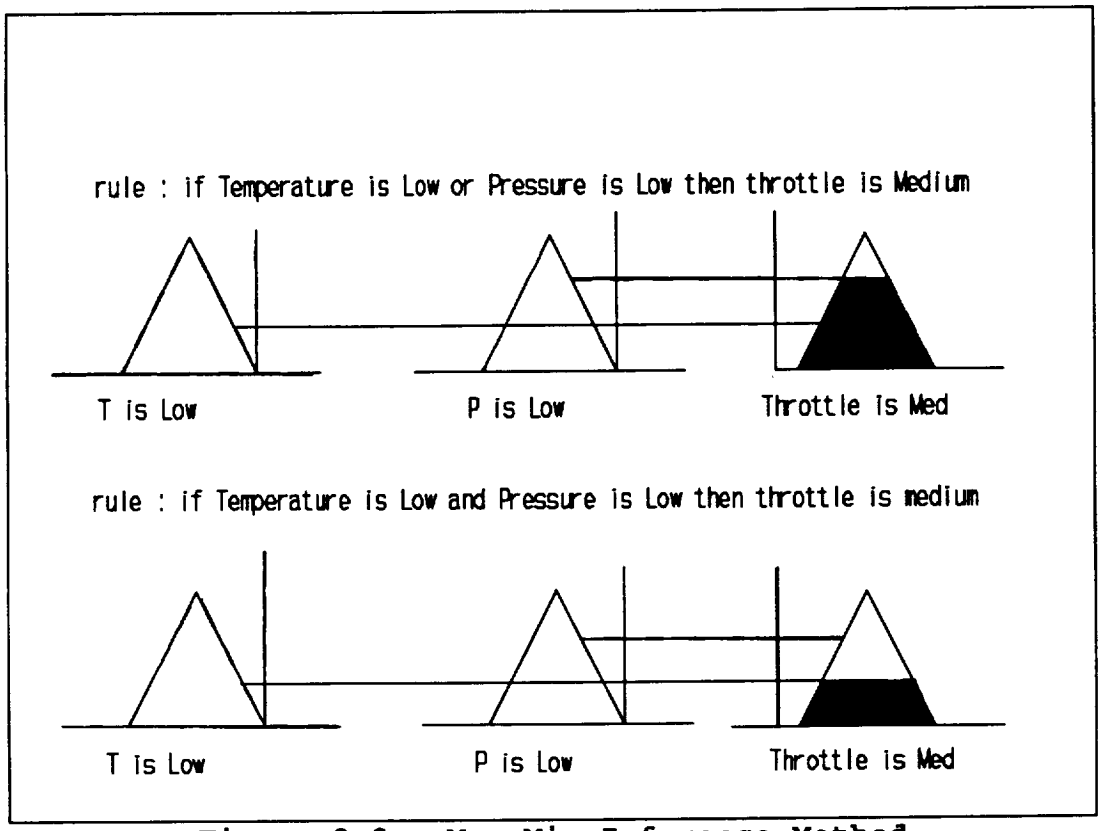

Figure 2.8. Max-Min Inference Method.

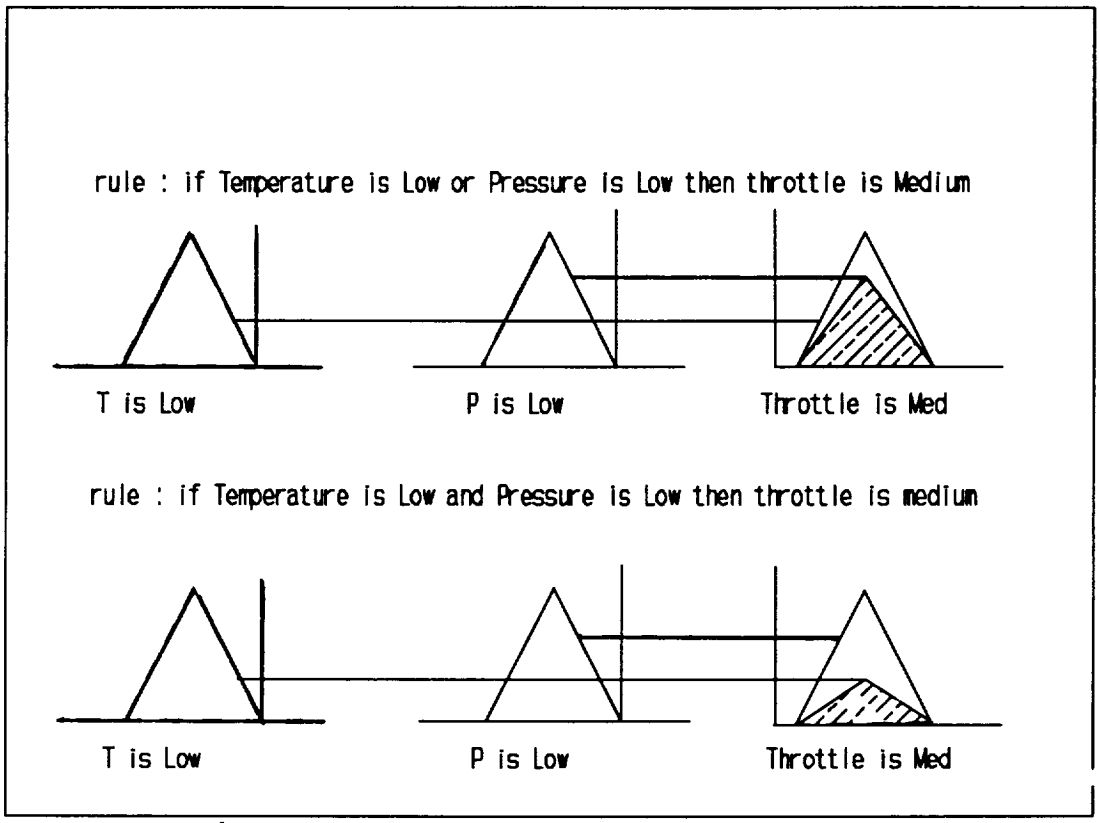

Figure 2.9. Max-Dot Inference Method.

2.4 Fuzzy Systems in Control Systems

2.4.1 Defuzzification

The previous section has described the main components of fuzzy systems. A fuzzy system consists of a fuzzy rule base and an inference engine. The fuzzy system maps a given set of fuzzy inputs, represented by linguistic values, into an output fuzzy set O. More inputs can be handled one at a time by forming several multi-input, single-output (MISO) fuzzy systems. Fuzzy logic is used to determine which rules fire and together with the inference method chosen, determine the output fuzzy set.

In control system applications, the crisp vector of measured variables needs to be fuzzified first. This is accomplished by considering each measured variable to be a linguistic variable with an appropriate term set. The fuzzification process consists in finding the degree of membership of each measured variable to the fuzzy sets in the corresponding term sets. Continuing with the example of the previous section, we may find that

$$x_1 \text{ is } A_1,\ x_1 \text{ is } A_2,\ x_2 \text{ is } B_5, \text{ and } x_2 \text{ is } B_6.$$

The fuzzy sets for which the degree of membership is non-zero are used as inputs to the fuzzy system. The output of the fuzzy system is a fuzzy set which corresponds, for example, to the degree of membership of the control

inputs. In control systems applications, the output fuzzy set needs to be converted into a crisp numerical value. This process is called defuzzification and it is described in the next section. A more detailed description of fuzzy logic controllers is given in the next chapter.

2.4.2 Defuzzification Method

The output of the fuzzy controller obtained using one of the inference methods described above is a fuzzy set of controls. However, a process requires a nonfuzzy value. This establishes the need for a defuzzification stage. In order to arrive at a crisp output value, a method is needed to pick a value that best represents the membership function. There are several methods for performing defuzzification. Two of the methods that are used are the center of gravity method and the max-procedure defuzzification method. The former method is discussed below. It yields, in general, better steady-state performance than the latter method (Lee, 1990). The max-procedure is hardly used because it does not consider the shape of the membership function (Hill, Horstkotte, and Teichrow, 1990).

2.4.3 Center Of Gravity Method

The center of gravity method picks the output value corresponding to the center of gravity of the output membership function as the crisp value for an output. In other words, in this method the action is given by the center of the summed area, which is contributed by the inputs (the premise part of the IF..THEN rule).

The output of the fuzzy system is the control input u. If the output fuzzy set is denoted by O then the control input u, determined by projecting the centroid to the axis corresponding to the universe of discourse of the control input, is given by

$$u = \frac{\int_{-\infty}^{+\infty} \tau O(\tau)\, d\tau}{\int_{-\infty}^{+\infty} O(\tau)\, d\tau}, \tag{2.26}$$

whenever

$$\int_{-\infty}^{+\infty} O(\tau)\, d\tau \neq 0, \tag{2.27}$$

These integrals are well defined in our applications since the output fuzzy membership function is non zero only over a finite range of values. The numerical evaluation of the integrals makes the center of gravity method more complex than the max method. Fortunately, it is not necessary to evaluate the integrals at all.

As described in the previous section, the membership function of the output fuzzy set is given by

$$\mu_o = \sum_{i=1}^{n} w_i C^i = \sum_{i=1}^{n} R_i . \tag{2.28}$$

where C^i corresponds to one of the fuzzy values in the term set $\{C_1, ..., C_{n3}\}$ of the control input linguistic variable (denoted by L_3 in the previous sections). It is possible to show that the centroid of the output fuzzy set is directly related to the centroids of C^i. The control input can be evaluated using (Kosko, 1992)

$$u = \frac{\sum_{i=1}^{n} w_i c_i I_i}{\sum_{i=1}^{n} w_i I_i} . \tag{2.29}$$

where n is the number of fuzzy rules that have fired, and for the ith fuzzy rule that has fired, $w_i = \min(A^i(x_1), B^i(x_2))$, c_i is the control input corresponding to the centroid of C^i, and I_i is the area under C^i. If the fuzzy sets are symmetric with respect to a vertical line passing through it, then c_i is simply the value of the control input on the axis of symmetry. This value is easily computed. In addition, if the area under each fuzzy set C_i is the same then $I = I_i$ for all i and it can be canceled from the equation. Therefore, the control input is simply given by

$$u = \frac{\sum_{i=1}^{n} w_i c_i}{\sum_{i=1}^{n} w_i}. \qquad \textbf{(2.30)}$$

If the fuzzy sets C_i are unimodal with peak belief values over its centroid (as in triangular or trapezoidal membership functions) and the area under each fuzzy set is the same, then the control input is also given by

$$u = \frac{\sum_{i=1}^{n} c_i O(c_i)}{\sum_{i=1}^{n} O(c_i)}. \qquad \textbf{(2.31)}$$

2.5 Example

Suppose we have a system with two inputs and one output. The two inputs are error and derror (the change-of-error) and the output is the control input of the plant. The fuzzy sets for both inputs and the output are all normalized with the universe of discourse [-6, 6]. In addition, the term sets for inputs and output are identical in this case.

In order to illustrate how to obtain a crisp output, consider an error of -1 and a change-of-error of 1.75. In addition, suppose that the rules that get fired are the following ones:

R_1 : IF error is Z AND derror change is PS

THEN control input is NS;

R_2 : IF error is Z AND derror change is Z

THEN control input is Z;

R_3 : IF error is NS AND derror change is NS

THEN control input is PS;

and

R_4 : IF error is NS AND derror change is Z

THEN control input is PB.

Figures 2.10 and 2.11 illustrate the fuzzification of the controller inputs, the

max-dot inference method, and the center of gravity defuzzification method. Figure 2.10 show the fuzzy sets Z and NS for error, PS, Z, and NS for error change, and the fuzzy sets NS, Z, PS, and PB for the control input. The additional lines illustrate how to find the scaling used in the max-dot inference method. The scaling factors for the four rules are given by w_1 = 0.5, w_2 = 0.25, w_3 = 0, and w_4 = 0.25, respectively. Figure 2.11 shows the resulting scaled membership functions contributed by the rules. The combined output membership function can be obtained using Equation (2.25). The resulting crisp value for the control input corresponds to the centroid of the combined output membership function. It can be easily solved using Equation (2.30):

$$I=\frac{[0.5\times(-2)+0.25\times0+0\times0+0.25\times4]}{(0.5+0.25+0+0.25)}=0.0, \qquad (2.32)$$

where the centroids of the NS, Z, PS, and PB membership functions are given by c_1 = -2, c_2 = 0, c_3 = 2, and c_4 = 4, respectively.

These calculations can be implemented on a computer. After each calculation, each error and error change will give a corresponding control input. These results can then be stored in the form of a look-up table in such a way that given an error and error change we can then look up the

control input. An example of a look up-table is given in Table 2.1.

Error	Error change								
	-4	-3	-2	-1	0	1	2	3	4
-4	5	4	4	3	3	2	1	1	-1
-3	5	4	3	2	2	1	0	0	-2
-2	4	3	3	2	1	1	0	-1	-3
-1	4	3	2	1	1	0	1	-2	-3
0	3	3	2	1	0	-1	-2	-3	-3
1	3	2	1	0	-1	-1	-2	-3	-4
2	3	1	0	-1	-1	-2	-3	-4	-4
3	2	0	0	-1	-2	-2	-3	-4	-5
4	1	-1	-1	-1	-2	-3	-4	-4	-5

Table 2.1. Sample look-up table.

The following procedure shows how a control input is determined from the look up table.

- Suppose the set point = 1.
- Output of system at t_1 = 4.
- Output of system at t_2 = 2.
- Error at t_1 = 4 - 1 = 3.
- Error at t_2 = 2 - 1 = 1.

- Error change = 1 - 3 = -2.

From the look-up table we can find the control input to the process at time t_2 to be 1. For errors and error changes that do not appear in the table, linear interpolation will be necessary to find the control input. This example shows the simplicity of determining the control input to a process. Moreover, the use of a computer makes the calculation process simple and easy to implement in real time.

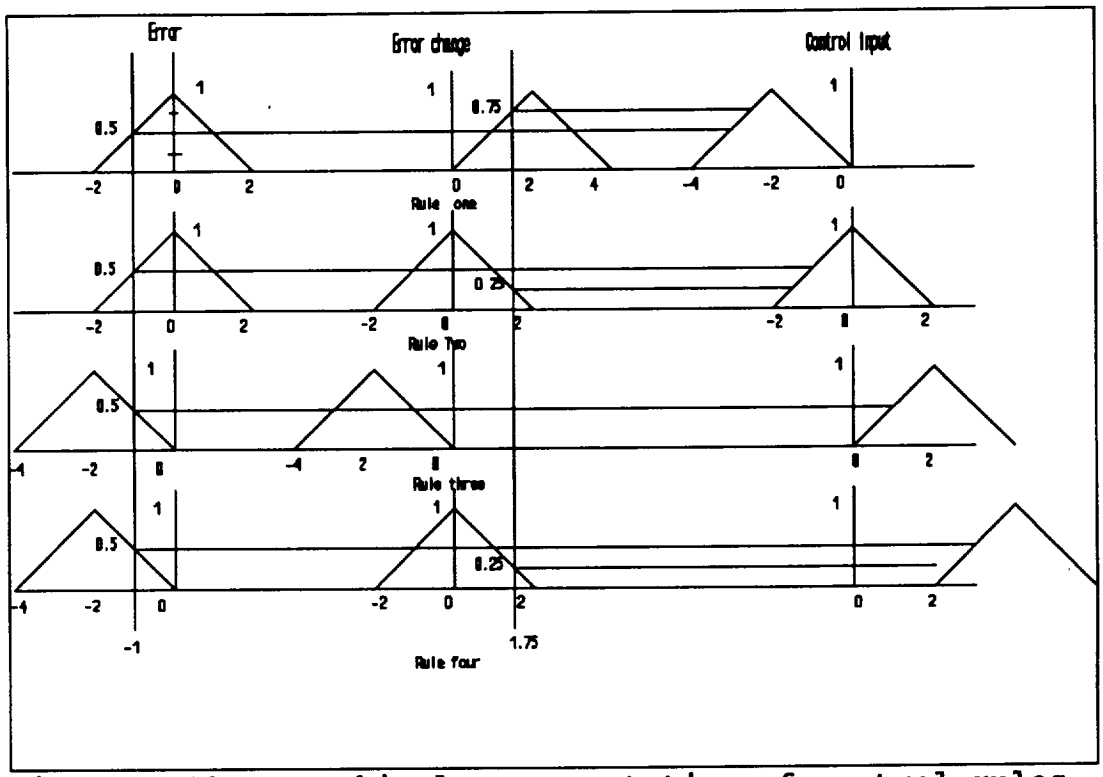

Figure 2.10. Graphical representation of control rules.

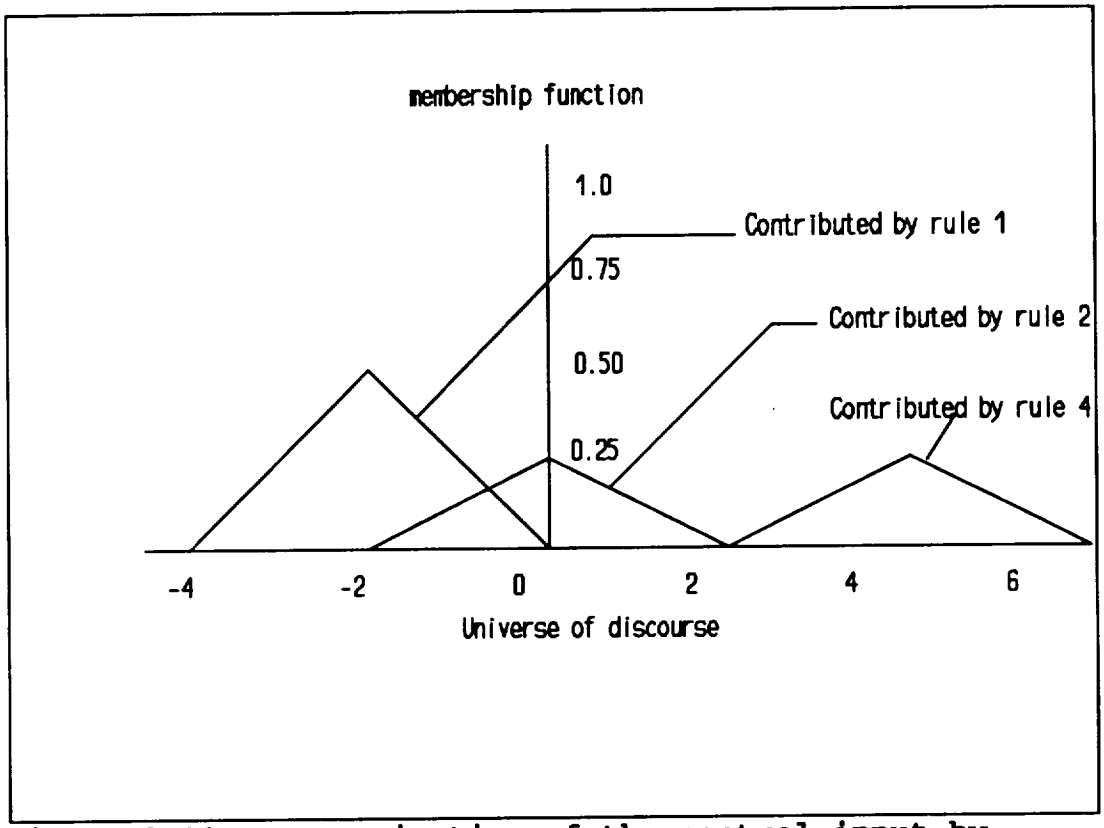

Figure 2.11. Determination of the control input by means of center of gravity method.

2.6 Conclusions

In this chapter, we presented an overview of fuzzy sets and fuzzy logic as they apply to control systems design. Moreover, we illustrated the operations on fuzzy sets analytically and graphically. Fuzzy logic was introduced and the concepts of a rule base, inference methods, and defuzzification were illustrated.

In conclusion, this chapter served as an introduction to fuzzy sets, fuzzy logic and how they can be applied to design fuzzy logic controllers. It is expected to be a useful research guide, together with the references, to researchers new to the field.

CHAPTER 3

FUZZY LOGIC CONTROLLERS

3.1 Introduction

A fuzzy logic controller is a fuzzy expert system which is a generalization of the expert systems widely used in artificial intelligence (AI) applications. The main difference between fuzzy expert systems and AI expert systems is in the way they handle uncertainty. In an AI expert system, uncertainty is handled using a probabilistic approach. A fuzzy expert system attempts to handle uncertainty in the way humans do, using linguistic variables and fuzzy sets. In fact, one interpretation of a fuzzy logic controller is that it models a human operator of a control process. The knowledge of the human operator is embedded in the fuzzy rule base. The inference engine and the defuzzifier approximate the response of the human operator to a given set of inputs.

Initially fuzzy logic controllers were applied in process control applications characterized by slow time constants and lacking the mathematical models of the process, but reasonably controlled by a human

operator. More recently, fuzzy logic controllers have been applied to more typical electrical engineering control problems such as motor control (Li and Lau, 1989), the inverted pendulum problem (Kosko, 1992), roll control of aircraft (Chiu and Chand, 1989) and many more applications. All of the mentioned applications show that fuzzy logic controllers can be used with faster time constant systems. In addition to single-input, single-output (SISO) processes, fuzzy logic controllers have been applied to multivariable problems such as a turbo fan engine (Heisemer, 1992). Furthermore, all of these applications offered comparisons between fuzzy logic controllers (FLCs), and proportional integral (PI), proportional integral derivative (PID), proportional derivative (PD), model reference adaptive control (MRAC) and other classical control techniques to prove that FLCs are as good or in many cases better than classical control techniques.

The goal of this chapter is to present our design procedure which includes new general guidelines for the tuning of fuzzy logic controllers. This will be followed up with examples to illustrate the capabilities and robustness of fuzzy logic controllers.

3.2 Fuzzy Logic Control System Design

A basic block diagram for fuzzy logic control is given in Figure 3.1 where r(kT) is a sequence of command requests, e(kT) = r(kT) - y(kT) is a sequence of tracking errors, u(t) is the control input signal, and y(t) is the output of the plant. This is a typical sampled-data implementation of fuzzy control systems that is useful for analysis and comparison to other control techniques. The controller consists of two main blocks. First, a filter is used to construct the change of error sequence $\Delta e(kT) = e(kT) - e((k-1)T)$, which is a rough first order approximation of the rate of change of error. A block diagram implementation of the filter is shown in Figure 3.2. Other filters are also possible (Langari and Berenji, 1992). The fuzzy logic controller (FLC) consists of four blocks illustrated in Figure 3.7; their design is described next.

A Design Procedure

Suppose that the control configuration in Figure 3.1 is chosen and that an appropriate sampling period T is chosen. The following main steps are typically used to design fuzzy logic controllers.

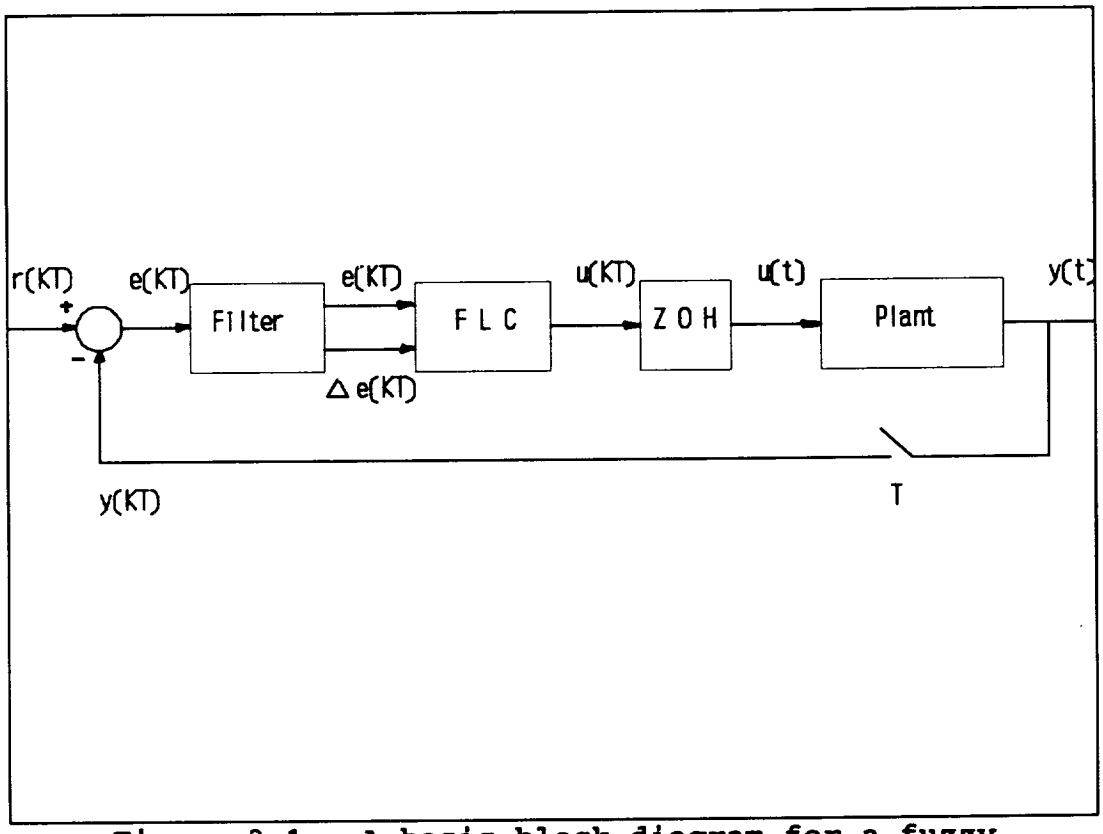

Figure 3.1. A basic block diagram for a fuzzy controlled system.

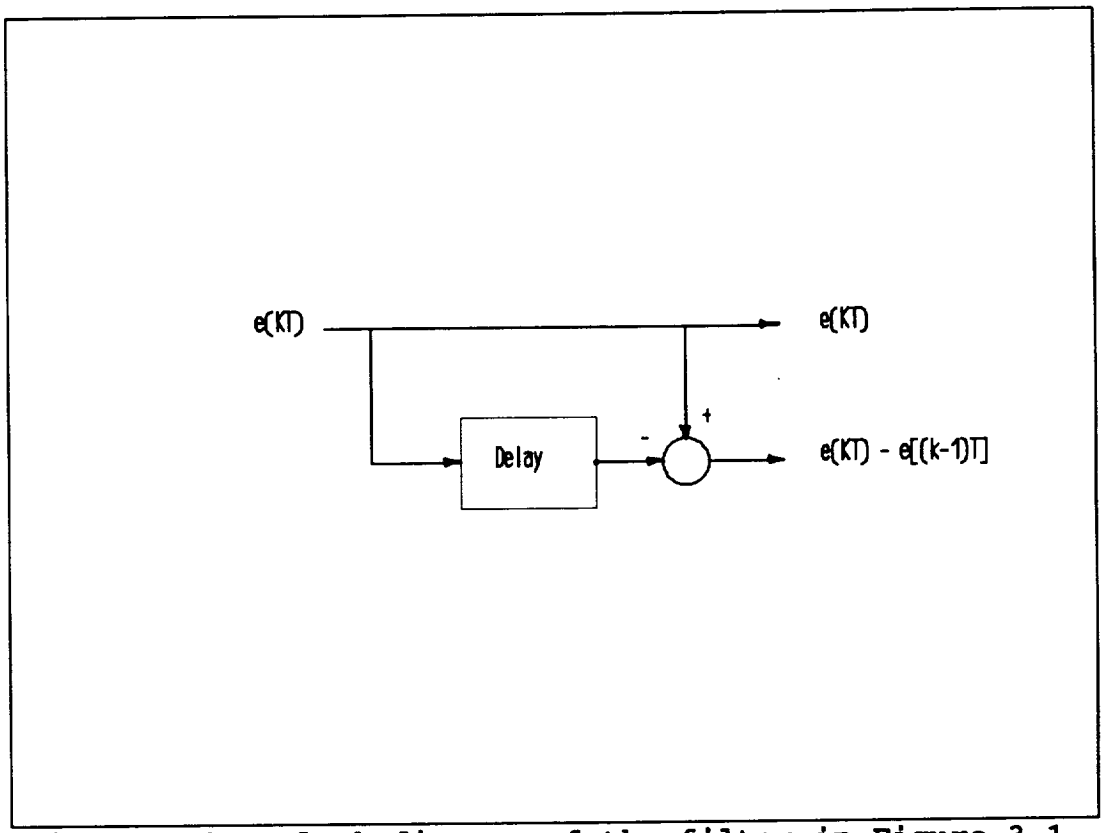

Figure 3.2. Block diagram of the filter in Figure 3.1.

Step 1. Acquire plant information.

Step 2. Select term sets for the linguistic variables.

Step 3. Form a fuzzy rule base.

Step 4. Tune the fuzzy controller.

These steps are described in more detail next.

3.2.1 Step 1. Acquire Plant Information.

The design of fuzzy logic controllers begins by obtaining the range of operation of the process. This is done by defining the numerical ranges of the signals r(kT), u(t), and y(t) using appropriate units. These ranges must be valid from startup to shutdown of the process. Knowledge of a range for r(kt) defines a range for y(kT) and y(t) at steady-state. Since it is possible for the system to experience overshoot or undershoot during the transient behavior, the range for y(t) could be bigger than for r(kT). If these situations are not possible, then both ranges can be the same. The range for u(t) can be determined by considering limitations on the actuators and on power availability. These ranges are simple to obtain with the help of an expert operator of the system. In the absence of such an expert, system identification techniques can be used. For example, suppose that the range

of command inputs, r(kT), is [0, 100] units. The following table explains how to set up the ranges for the other signals where Δ is the expected maximum percent overshoot and y_{min} is the negative minimum expected undershoot such that $|y_{min}| < 100$.

Signal	Conservative Range	Usual Range
Output: y(t)	$[y_{min}, (1+\Delta)100]$	[0, 100]
Error: e(KT)	$[-(1+\Delta)100, (1+\Delta)100]$	[-100, 100]
Derror:Δe(kT)	$[-2(1+\Delta)100, 2(1+\Delta)100]$	[-200, 200]

Table 3.1. Range of signals

In addition, in this step it is important to acquire additional plant information such as the presence of nonlinearities, for example, dead zones. This information is useful in determining the term sets and the rule base.

3.2.2 Step 2. Select Term Sets For The Linguistic Variables.

For the control problems of interest in this study, the linguistic variables are the error samples, e(kT), the approximation of the rate of change of error samples, Δe(kT), and the output of the fuzzy logic controller,

u(kT). The role of the first stage of the fuzzy controller is to convert the crisp numerical data for the first two signals into linguistic values. This stage is called fuzzification. The set of possible linguistic values for each linguistic variable constitutes a term set or fuzzy partitioning of the universe of discourse. The universe of discourse was obtained in Step 1.

There are two main design steps in the fuzzification stage. The first step is to decide on the number of values which are fuzzy. Recall that fuzzy values are fuzzy sets. A good rule of thumb is to start with three fuzzy sets in their term sets; for example, low, medium, and high. If the system performance is not as desired, such as large steady state error, then the number of membership functions is increased as necessary. Figure 3.3 shows examples of three and five membership function sets. In all the examples in this thesis, we have used five membership function sets.

In order to evaluate the system performance, the second design step needs to be accomplished. The second design step is to determine the shape of the membership functions for each linguistic value. Choosing the shape of the membership functions of the fuzzy sets is based on heuristics and depends entirely on the designer and the expert. The lack of an expert opens the door for experimentation on the shape of these fuzzy sets and on their

degree of overlap. In the literature, triangular shapes for the fuzzy sets have been advocated. Moreover, the triangular shapes have been shown to provide good control action provided that the sets overlap adequately (Kosko, 1992). The designer must decide what shape works best for the problem at hand. Regarding the overlapping of the sets, a good rule of thumb is that adjacent membership functions should overlap approximately 25 percent (Kosko, 1992). Different problems might require more or less overlap. The overlapping of the fuzzy sets is essential because it provides continuous control action since it guarantees that more than one rule will fire at one time. On the other hand, non-overlapping sets do not provide very good control since only one rule can be applied at any time instant.

The following rules of thumb can be used in the absence of a real human expert. As a starting point the membership functions for a particular input or output should be symmetrical triangles of the same width peaking at belief values of one, except for one covering the lowest values of the input or output and one covering the highest value of the input and output (Hill, Horstkotte, and Teichrow, 1990); see Figure 3.3. These two membership functions should be shouldered ramps with peak belief value of one.

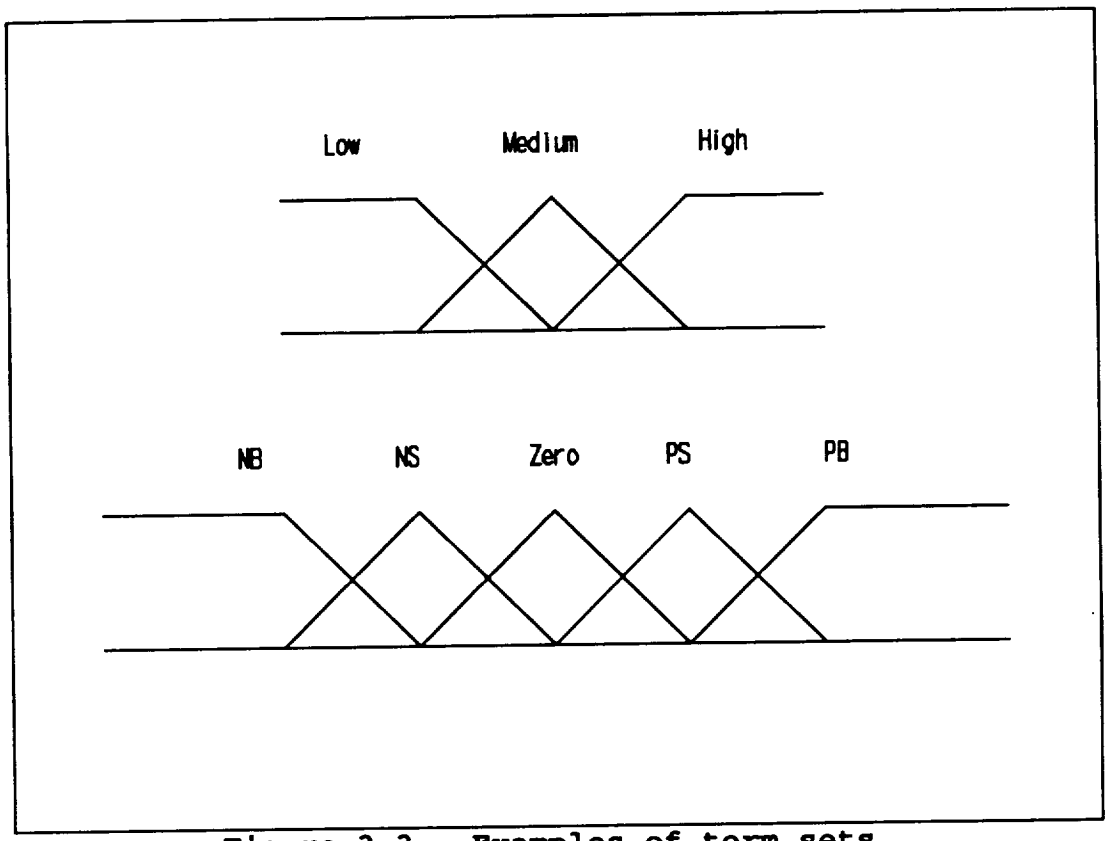

Figure 3.3. Examples of term sets.

Furthermore, the membership function width should initially be chosen so that each value of the input or output is contained in at least two membership functions. This helps ensure that more than one rule applies to each value of the input or the output. The overlapping of the membership functions will make the control of the system smoother. This is illustrated in the next example.

3.2.2.1 Example.

Consider the two rules:

1) IF error is Zero AND derror is Negative Small
   THEN output is Positive Small

and

2) IF error is Negative Small AND derror is Negative Big
   THEN output is Positive Big.

Figure 3.4 shows that an error of -1 and derror of -2.5 will fire both rules provided that the fuzzy sets associated with the linguistic variables overlap. On the other hand, the second rule will not fire in the case of the non-overlapping rules as in Figure 3.5. In the case of Figure 3.4, we calculate the control input by using the center of gravity method as follows. The -1

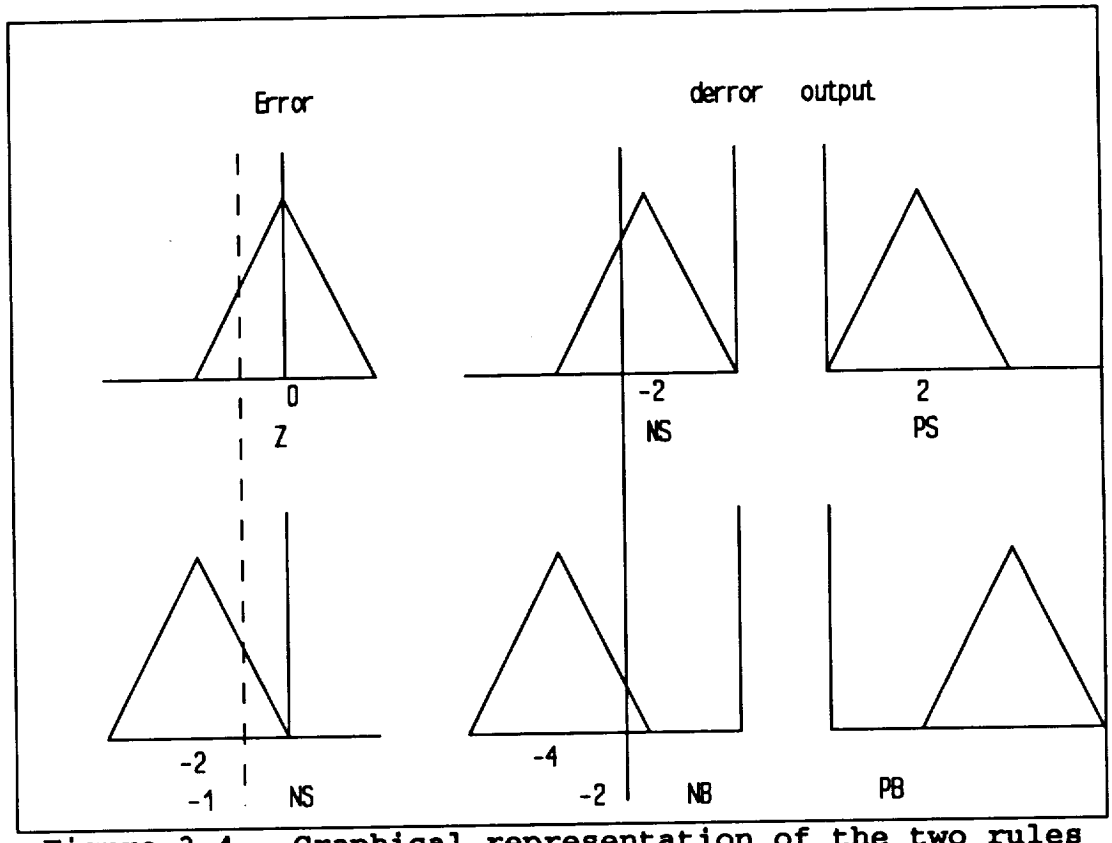

Figure 3.4. Graphical representation of the two rules with overlapping sets.

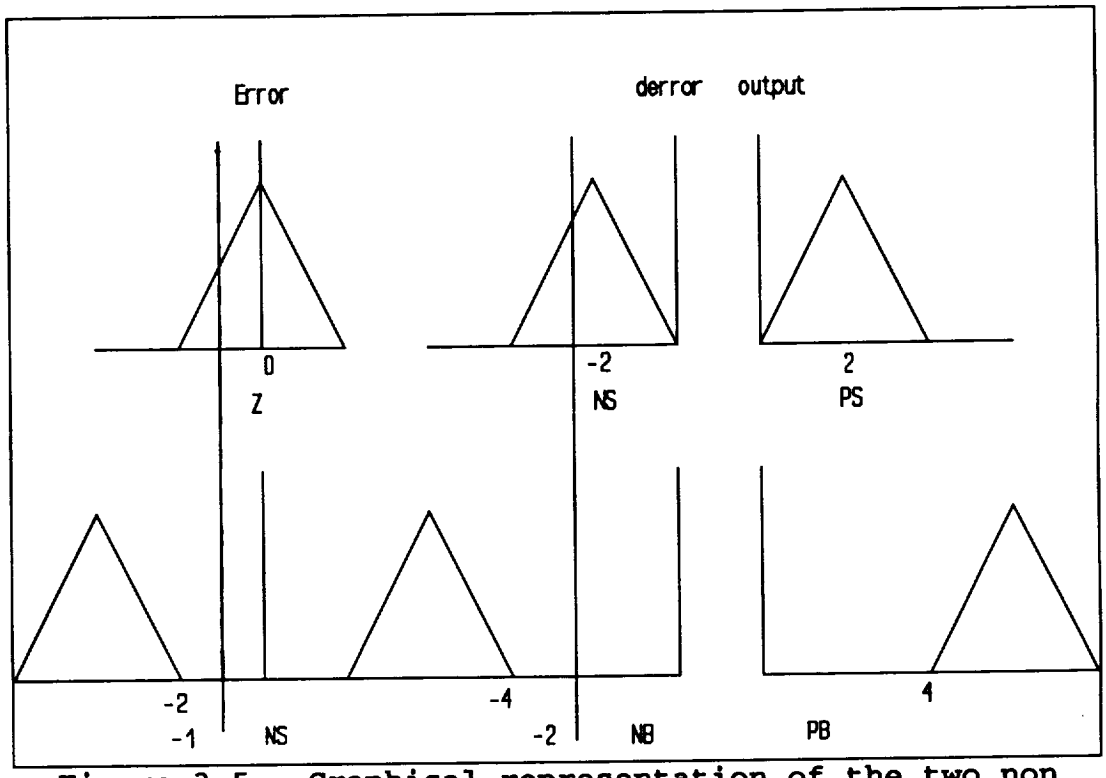

Figure 3.5. Graphical representation of the two non overlapping rules.

input belongs to the fuzzy set Zero with degree of 0.5 and to the fuzzy set Negative Small with a degree of 0.5. On the other hand, the rate of change of error belongs to the set Negative Small with a degree of 0.75 and to the set Negative Big with a degree of 0.25. By using Equations (2.4) -(2.8) we can find the control input or the output value of the controller. We do that by first finding the scaling of the output membership functions, w_i, and the control inputs that correspond to the centroids of the output membership functions, c_i.

$$w_1 = \min(0.5,0.75) = 0.5,$$

$$w_2 = \min(0.5,0.25) = 0.25,$$

$$c_1 = 2,$$

and

$$c_2 = 4.$$

Using Equation (2.30),

$$\text{Output} = \frac{0.5 \times 2 + 0.25 \times 4}{0.5 + 0.25} = 2.666.$$

On the other hand, in Figure 3.5 we see that the second rule does not get fired by the inputs of -1 and -2.5. Similarly, we can calculate the control input as we did for Figure 3.4. The resulting control input will be 2.

3.2.3 Step 3. Form The Fuzzy Rule Base

The main source of knowledge in the construction of fuzzy controllers comes from the human operator. This knowledge is represented in terms of sentences that describe the situation at hand and the action to be taken in light of this information. These sentences consist of a set of conditional IF..THEN statements where the first part of each contains the condition (premise) and the second part deals with the conclusion (action). To build the linguistic protocol, two main types of questions are relevant in the construction of fuzzy logic controllers (Pedrycz, 1989):

- questions about the operators own behavior; for example, what would you do in such and such situation?

and

- questions about the behavior of the process; for example, why does such and such situation occur?

From these type of questions the designer will be able to construct the rule base for the fuzzy controller. At times, one will find that not all the rules in the rule base are needed and thus will delete those unused rules. This improves the efficiency of the controller at run time since it has less rules to search.

The following is an example of a rule base that represents the entire process for a second-order system controller. From the table we can find what the control input will be, given the error and change of error. For example, entry 1,1 is rule 1:

R_1: IF error is PB AND derror is NB THEN control input is PS.

Error	Change of Error				
	NB	NS	Z	PS	PB
PB	PS	PS	PB	PB	PB
PS	NS	Z	PS	PS	PB
Z	NS	NS	Z	PS	PS
NS	NB	NS	NS	Z	PS
NB	NB	NB	NB	NS	NS

Table 3.2. Example of a rule base.

The acronyms used above are defined as follows:

PB: Positive Big,

PS: Positive Small,

Z : Zero,

NS: Negative Small,

and

NB: Negative Big.

A physical interpretation of the fuzzy values is explained next. Consider the statement "near the set point from the positive direction." This statement can be represented by the linguistic value Positive Small (PS). The statement "still moving towards the set point" can be represented by the linguistic value Negative Small (NS) and the control input (u) can be represented by the linguistic value Zero (Z). The above description of what the controller is to do in this situation can be expressed by the following rule:

IF error is PS AND derror is NS THEN u is Z.

The rule base need not be symmetric unless the process is symmetric. For example, if one expects the setpoint to have positive and negative values, then this dictates that the rule base be symmetric. Otherwise, symmetry is not required. Moreover, symmetry makes sense when the term sets have equal numbers of partitions.

3.2.4 Step 4. Tune the Fuzzy Controller

The issue of tuning a fuzzy logic controller is important and is an essential part of the design process of fuzzy logic controllers. Tuning of fuzzy logic controllers is required to achieve the desired control specifications. Tuning the controller consists, in general, in re-executing design steps 1 - 3. In practice, the controller is first tuned by modifying the term sets of the linguistic variables and the fuzzy rule base. Systematic techniques to perform tuning is an area of research. Through our design experience we have arrived at the following set of guidelines to modify the membership functions of the term sets and their relation to control specifications. These guidelines are:

- Making the Negative Big and Positive Big sets dominant will provide faster control action and may result in overshoot.
- Making the Negative Small and Positive Small sets dominant will provide for faster settling time with little or no overshoot.
- Making the Zero set dominant will insure that no overshoot will occur, but this may result in some steady state error.

In order to meet the control specifications of a particular application, the membership functions of the term sets should be modified using the above

guidelines. In addition, the fuzzy rule base may also be modified.

The methodology for fuzzy logic control system design introduced in this first two sections is applied to four examples.

3.3 Design Example One

Consider the plant modeled by the following Laplace transform transfer function

$$G(s) = \frac{5}{(5s+1)(4s+1)}.$$

It is seen that the plant is open-loop stable, but it has a slow settling time of 25 seconds. This plant was used in an example in Pedrycz (1989). In the design in Pedrycz (1989), the fuzzy logic controller has three inputs: error, change-of-error, and sum-of-errors. The third input was added to approximate integral action in the fuzzy logic controller. This was required because, without it, there was a steady-state error in the response. We will show that our guidelines to tune the controller can be used to yield zero steady-state error when the controller has only two inputs: error and change-of-error. In addition, we will speed up the response of the system and set the D.C. gain of the closed-loop system to one. The knowledge of the plant

transfer function is used only for simulation purposes.

The first step in the design process is to acquire plant information. In this step the linguistic variables are chosen and universes of discourse are obtained. The main linguistic variables are the inputs and outputs of the controller. The inputs are the error and change-of-error. The latter is found by first order approximation of the derivative of error (Franklin, Powell, and Workman, 1990). The output is the control input which is dictated by the command request signal.

In order to determine the universes of discourse, a range or universe of discourse for the command inputs needs to be determined. For example, to compare the response with the open-loop response unit step response, consider a step input of amplitude equal to five. Thus, the range for the output should be [0, 5] if the closed loop system response has no overshoot or undershoot. This range is only important, in so far as it is used to determine the range of the error linguistic variable. From this output information we can deduce the universe of discourse for the error linguistic variable to be [-5, +5]. For example, if the output is 0 and the reference input is 5 then the error will be

$$e = 5 - 0 = 5.$$

On the other extreme, if the output is +5 and the reference input is 0 then the error will be

$$e = 0 - 5 = -5.$$

The second step is to determine the term sets for the linguistic variables. The first issue is to decide the number of membership functions that will best represent the inputs (error, derror), and the output (u) of the controller. A good rule of thumb is to start with three and see if the controller behaves in satisfactory way. If a large steady state error occurs then increasing membership functions from three to five or seven will reduce the steady state error and provide better control. In the case of this example we choose to represent both inputs and the output (the control input) as in Pedrycz (1989) with five membership functions each. These membership functions are then branded with a linguistic value that defines it. The linguistic values are the same ones used in Table 3.2. The fuzzy sets for the error (e), change-of-error (de), and the control input (u) are chosen as explained in the design procedure. They are given in Appendix 3A at the end of this chapter. These membership functions are different from the ones used by Pedrycz (1989).

The third step is to form the rule base. In order to be able to compare our results to those in Pedrycz (1989), we used his same rule base as given in Table 3.3.

Error	Change of Error				
	NB	NS	Z	PS	PB
PB	NB	NB	NB	NS	PB
PS	NB	NB	NS	PS	PB
Z	NB	NS	Z	PS	PB
NS	NB	NS	PS	PB	PB
NB	NB	PS	PB	PB	PB

Table 3.3. Rule base for example.

The rules are applied in the following manner. The controller takes two numerical inputs, mainly the error and the change of error. The inference engine then determines to how many membership functions the two inputs belong to. If it is found that they belong to four membership function sets

then those four sets will apply and the corresponding rules contribute to the control input. This is done through the use of the max-min or the max-dot inference methods and consequently through the use of a defuzzification stage to produce the crisp output. This process is repeated until the output has reached the desired set-point with minimum steady state error. Figure 3.6 shows the stages of a fuzzy controller.

In order to simulate the closed-loop control system, we need to make a continuous to discrete conversion of the plant using a zero-order hold. The resulting difference equation for the plant becomes

$$y[i+2] = 1.9955y[i+1] - 0.9955y[i] + 0.0000148u[i+1] + 0.00001246u[i].$$

The sampling period for this process was arbitrarily chosen to be 0.01 seconds. The fuzzy sets and the rule base for this example are included in Appendix 3A. The C code listing used for the simulation is also included.

The controller we obtained for the second order plant worked very well in speeding the response of the system. The responses we obtained depended on the choices for the error and the control input sets. From the guidelines of Step Four, manipulation of the membership functions was done to achieve different responses. In this particular example, we found that the system response was fast at the expense of having a 34% overshoot. To

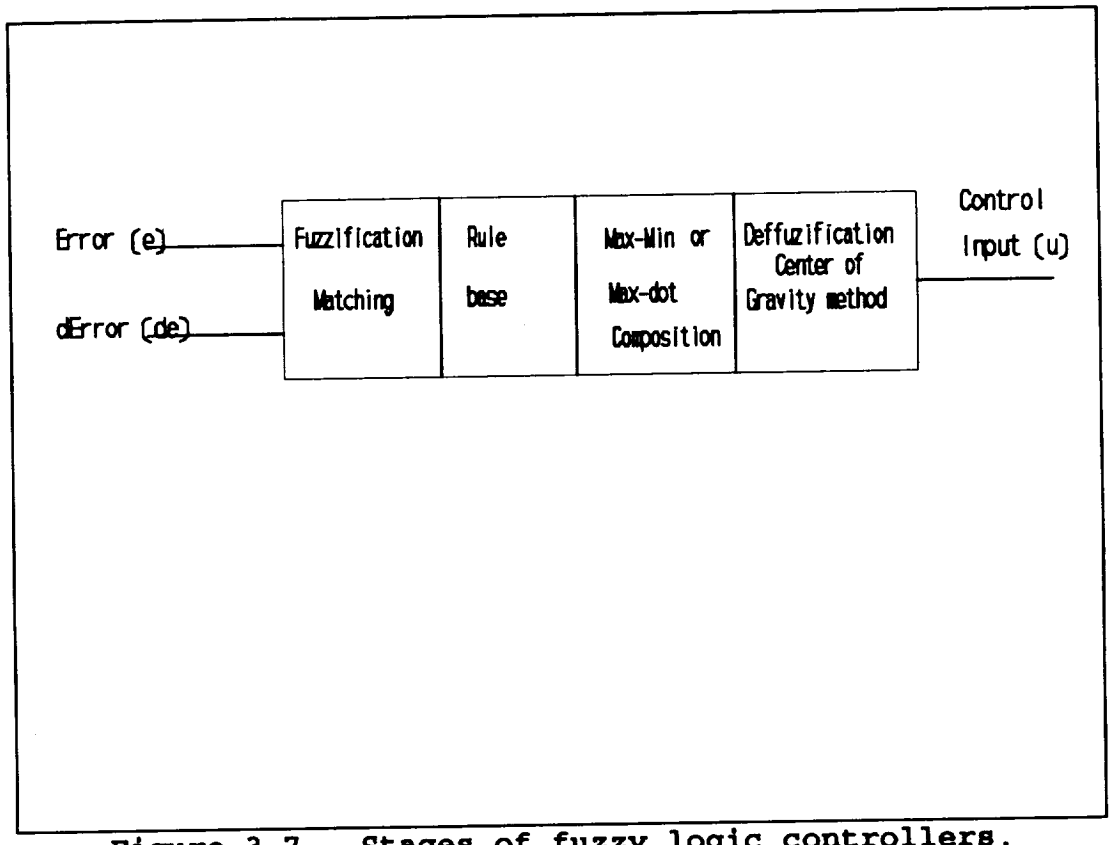

Figure 3.7. Stages of fuzzy logic controllers.

reduce the overshoot, we follow the outlined guidelines by making the membership functions for the NB and PB smaller and not dominant. However, this implies that the PS and NS will be dominant if no changes are made to the zero set. After making these changes the overshoot dropped to 21%. However, the system response slowed down slightly. Further manipulation of the shape of the membership function sets can result in no overshoot. This is also accomplished at the expense of the time response.

In all three cases the overall performance of the system was improved. The fuzzy controller that we designed after the second tuning appears to have faster response time than the one designed by Pedrycz (1989) with three controller inputs. In addition, all three of our designs had zero steady-state errors for the nominal plant. Our control inputs are bigger initially but they all go to zero. The control input in Pedrycz's design (1989) appears to settle to a steady-state value of one. Figures 3.7 through 3.9 show the response of the system to a step input of 5 before and after the first and second modifications of the sets. Note that all three responses are valid responses and it is entirely up to the designer to choose which is best for the application at hand.

To test the robustness of the controller, variation of the command input, variation of the dc gain, and the location of the poles was done. The poles, the dc gain, and the set points were varied by plus and minus 20% for the controller after the first tuning. The results are presented in Tables 3.4 through 3.6. Table 3.4 illustrates the responses to variation of the command input. Table 3.5 illustrates the responses to variation on the location of the poles and Table 3.6 illustrates the response to variation of the dc gain. Furthermore, these responses are illustrated graphically in Figures 3.10 through 3.12.

From these results it is seen that variations of plant parameters produced some steady-state error. There are currently no analytical tools to explain these responses. Variation in plant parameters were modeled by shifting the poles of the transfer function to +20% and to -20% and increasing and decreasing the dc gain of the plant by 20%. It is shown that variation of plant parameters by 20%, created a steady state error of less than 5% and had very little effect on the time response of the plant and controller.

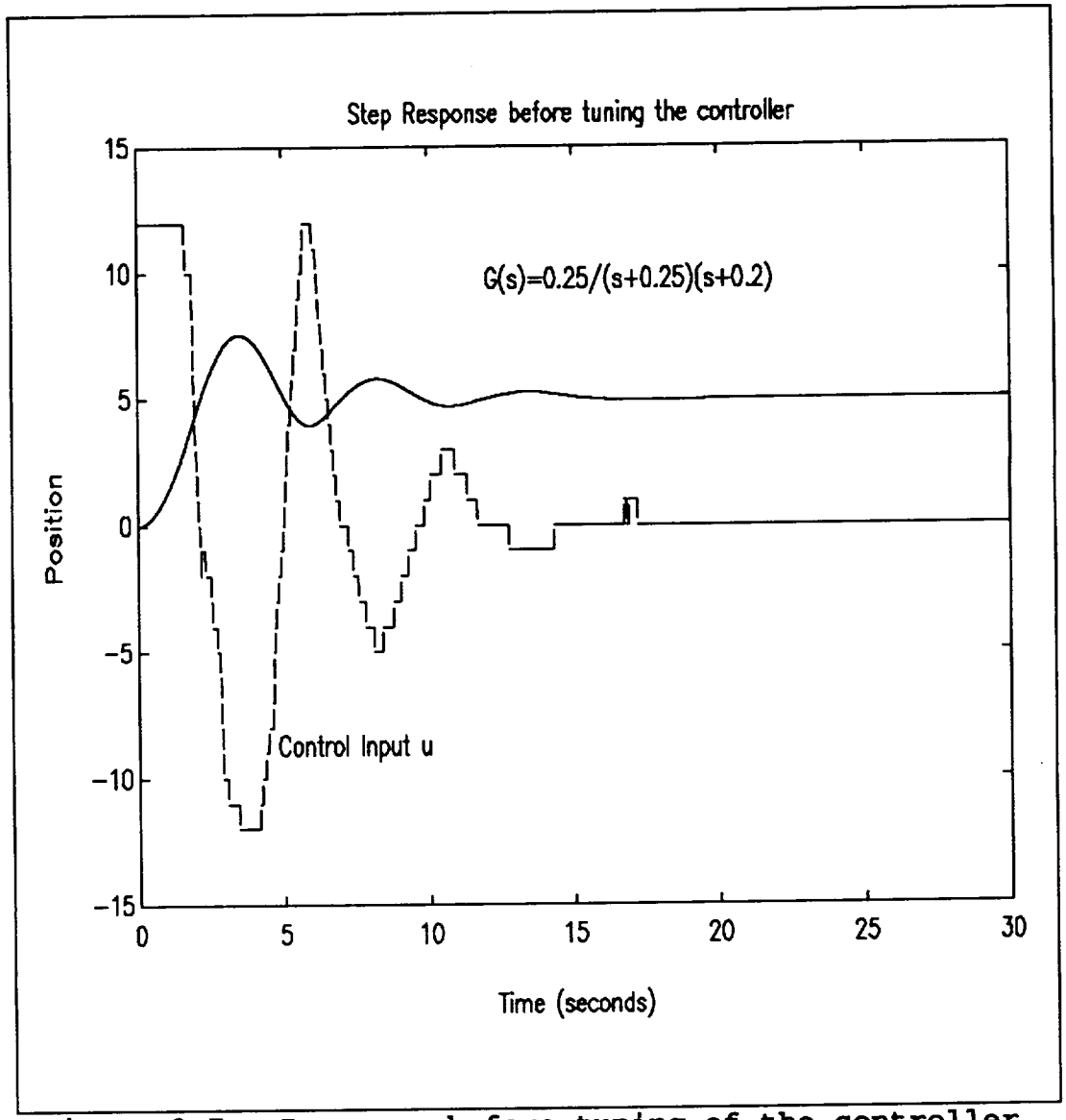

Figure 3.7. Response before tuning of the controller, control input included.

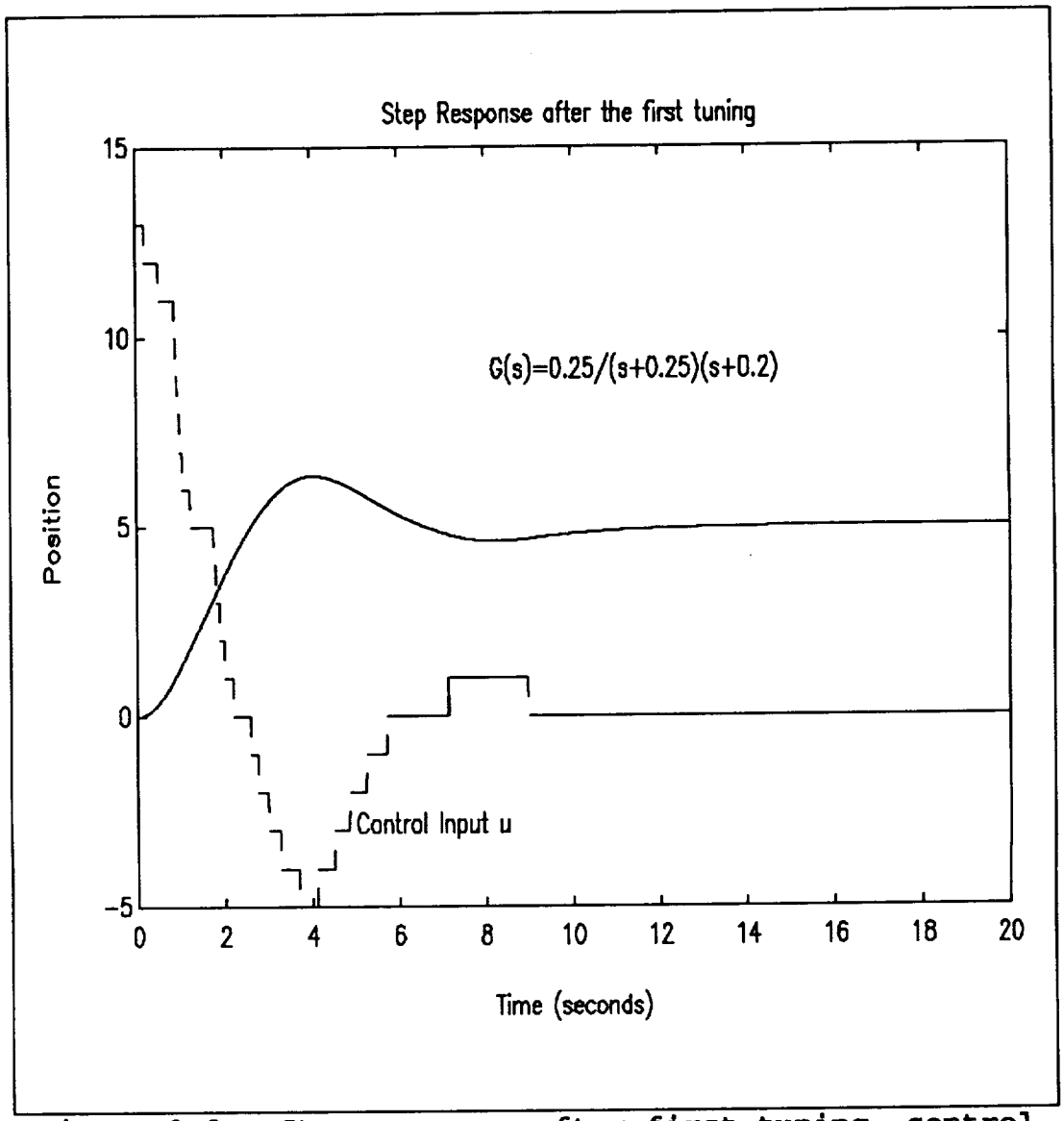

Figure 3.8. Step response after first tuning, control input included.

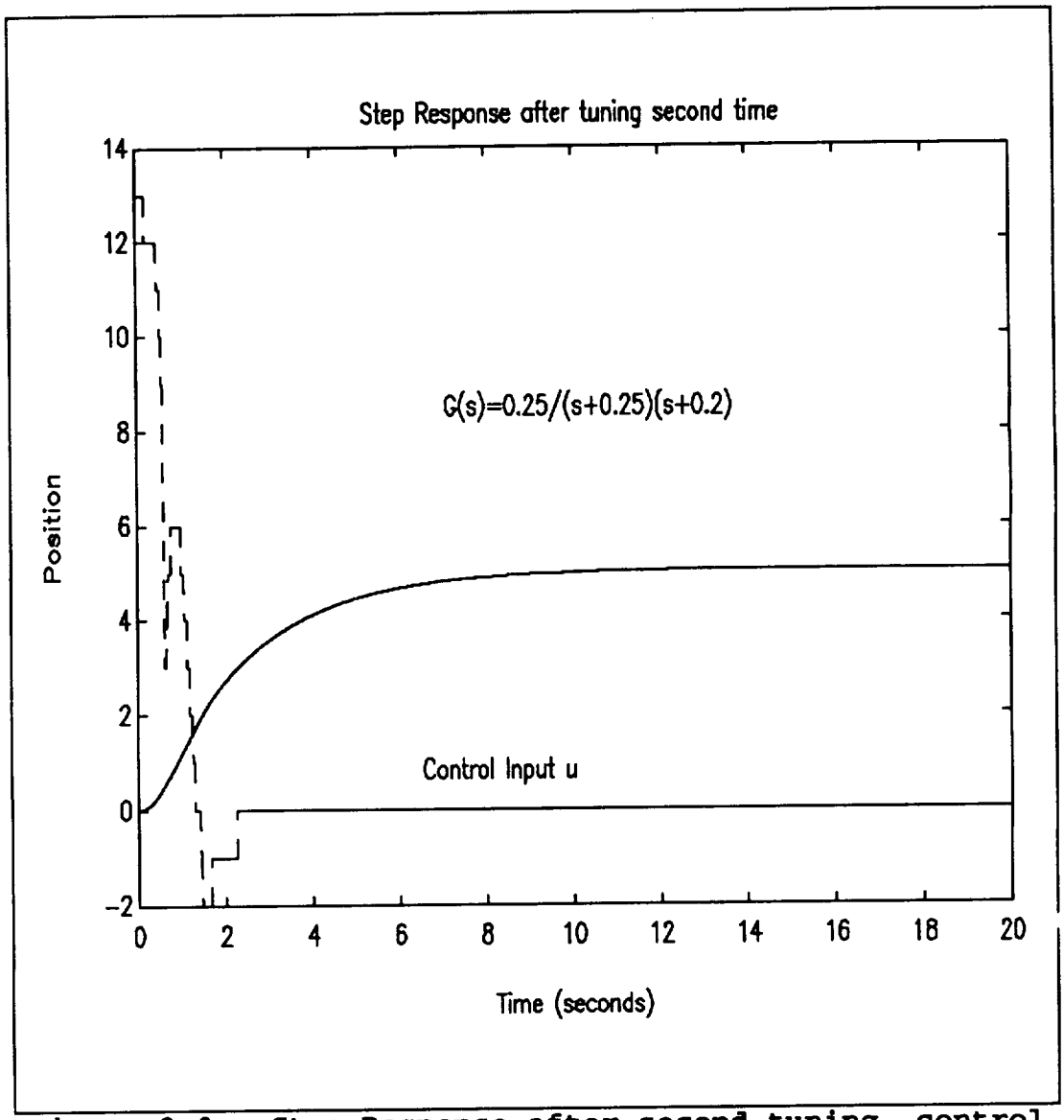

Figure 3.9. Step Response after second tuning, control input included.

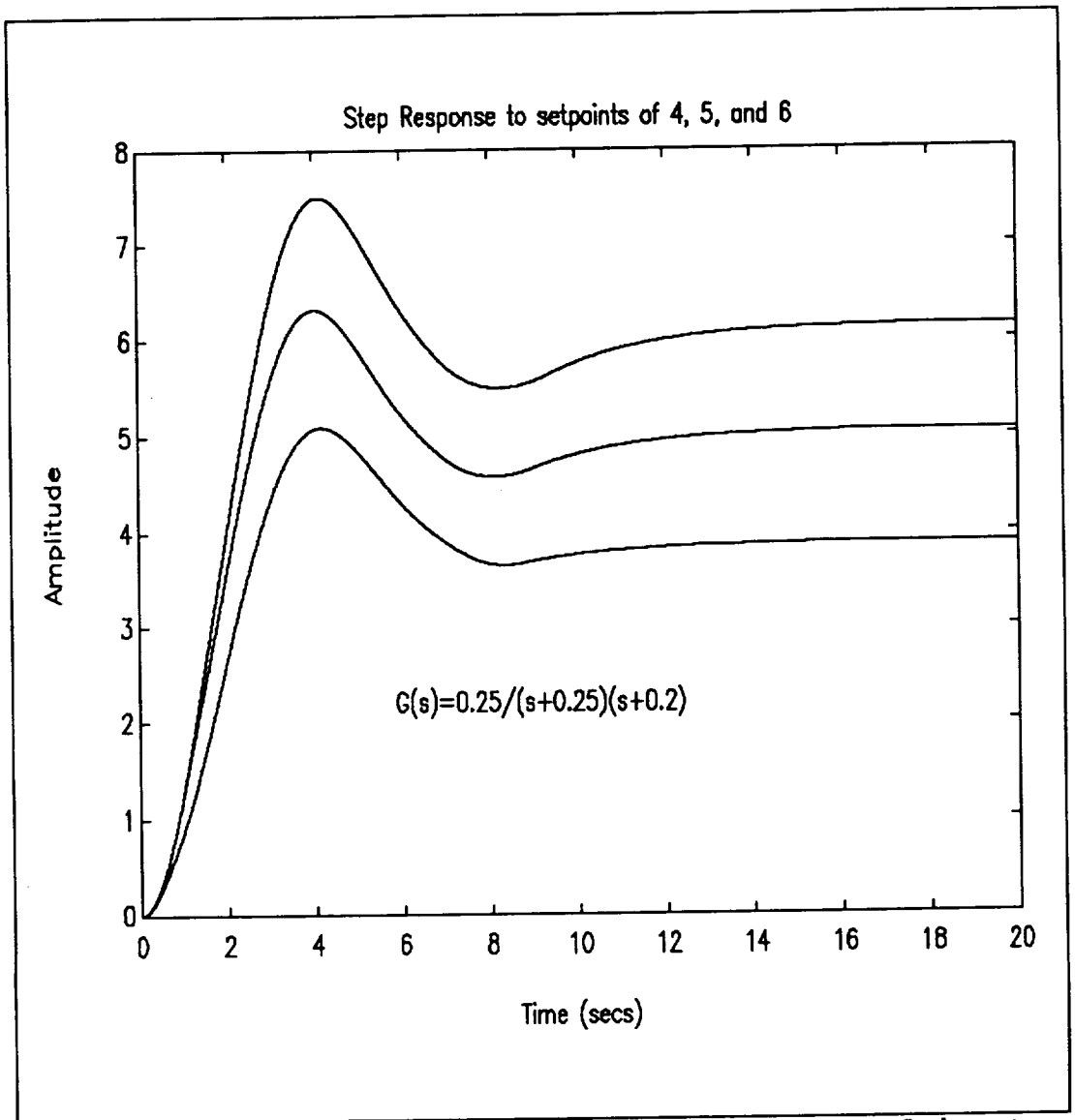

Figure 3.10. Step response to different command inputs.

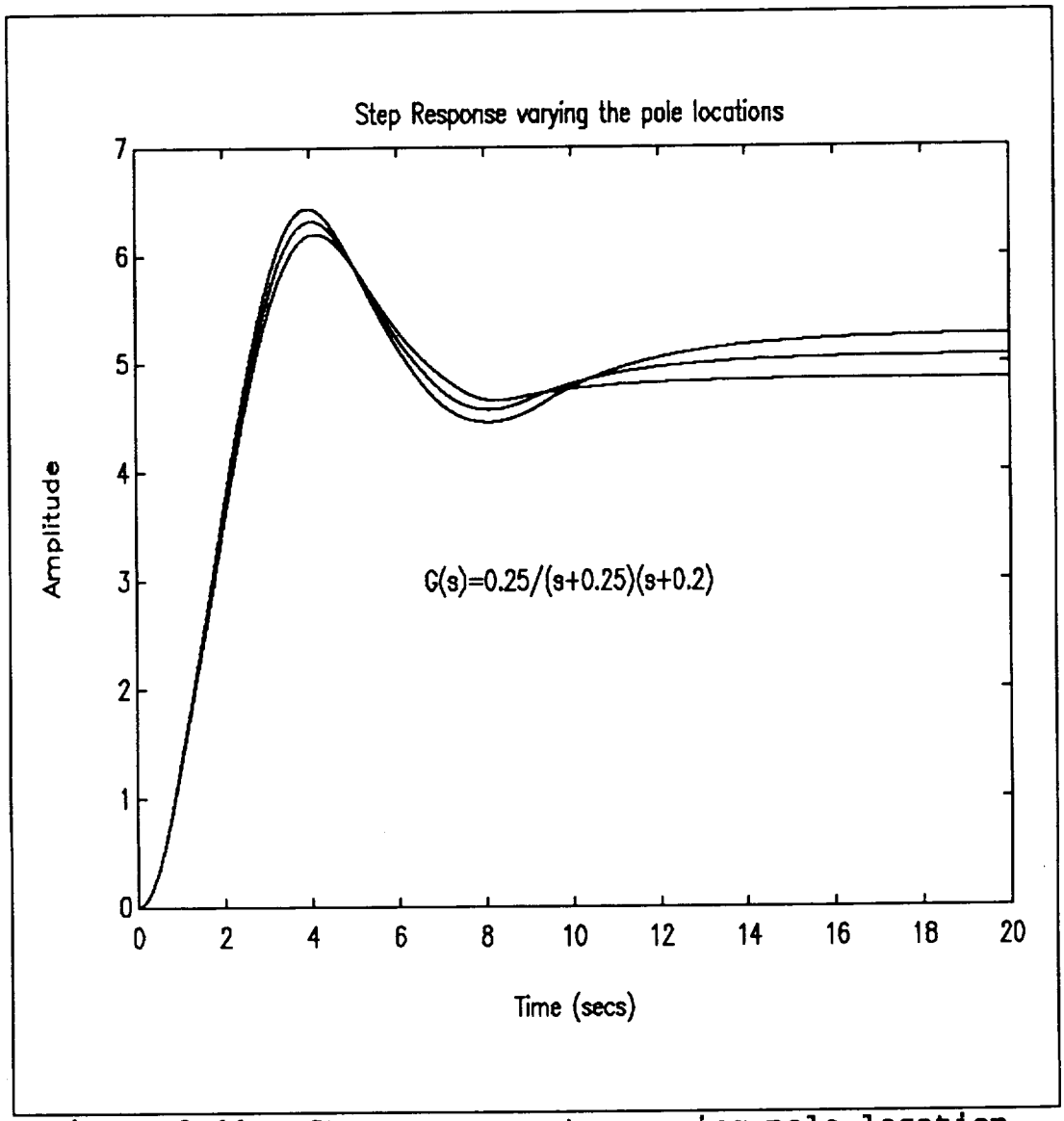

Figure 3.11. Step response to varying pole location.

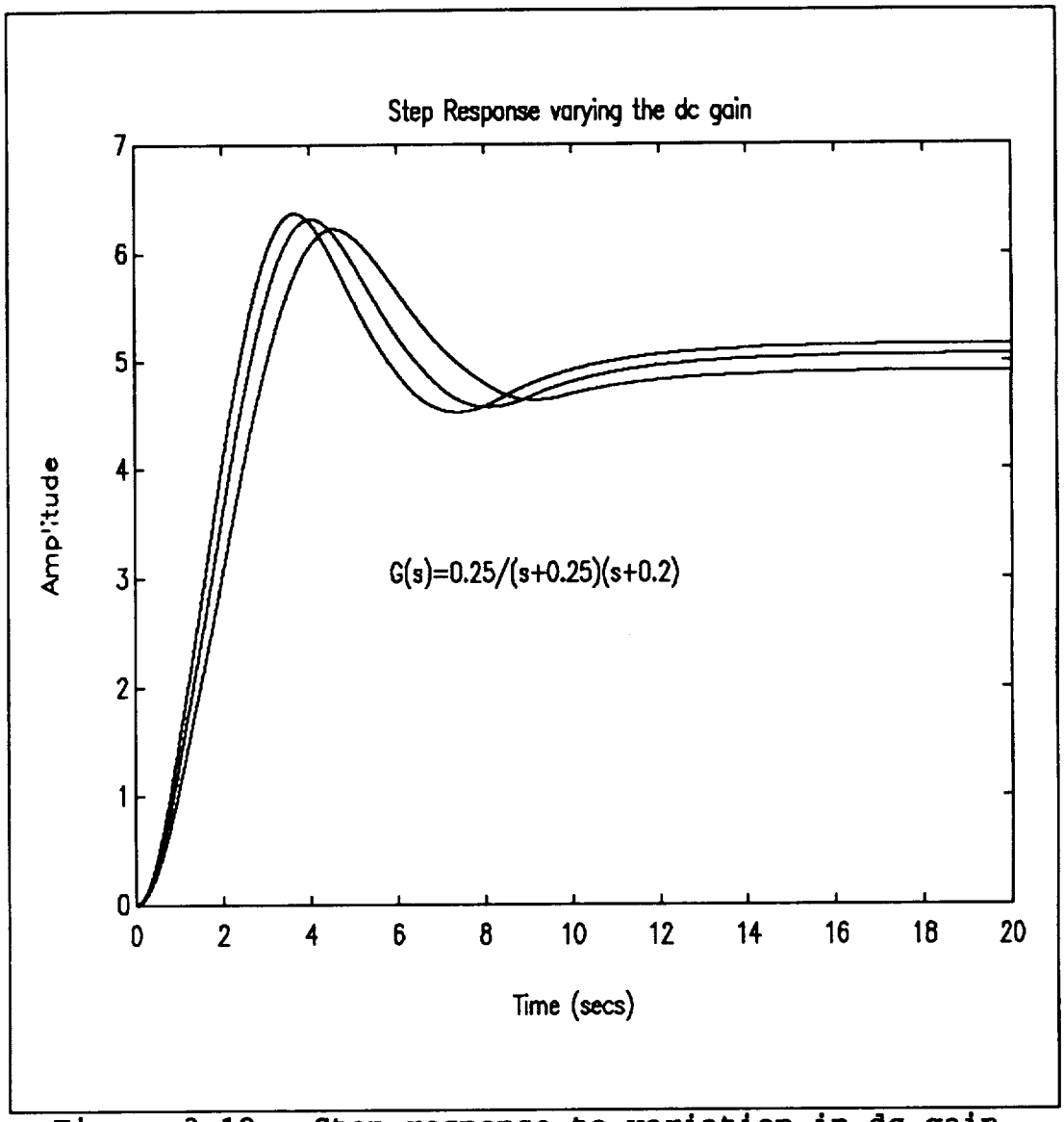

Figure 3.12. Step response to variation in dc gain.

$G(s) = \frac{0.25}{(s+0.25)(s+0.2}$	%O.S	e_{ss}	t_s(sec)	t_r(sec)
Command Input r = 5	26.4	0.0	11.89	1.81
Command Input r = 4	27	0.12	12.29	1.91
Command Input r = 6	25	0.14	13.47	1.87

Table 3.4. Results of varying the command input.

G(s)	% O.S	e_{ss}	t_s (sec)	t_r (sec)
$\frac{0.25}{(s+0.25)(s+0.2)}$	26.4	0.0	11.8	1.81
$\frac{0.25}{(s+0.2)(s+0.2)}$	28.8	0.25	14.7	1.76
$\frac{0.25}{(s+0.25)(s+0.25)}$	24	0.15	11.24	1.86

Table 3.5. Results of varying the pole locations by 20%.

G(s)	%O.S	e_{ss}	t_s (sec)	t_r (sec)
$\frac{0.25}{(s+0.25)(s+0.2)}$	26	0.0	11.89	1.81
$\frac{0.20}{(s+0.25)(s+0.2)}$	24	0.1	12.91	2.33
$\frac{0.30}{(s+0.25)(s+0.2)}$	27	0.15	12.91	1.79

Table 3.6. Results of varying the dc gain by 20%.

3.5 Design Example Two

In this example a fuzzy controller for a type-one plant was designed and simulated. Unlike the first plant which was stable, this type-one system is not stable, it is marginally stable. This implies that we need a controller that will bring the plant under control and improve the response time.

In this example we tuned the controller three times to reduce the overshoot. The results after first and second tuning are presented here. The rule base and membership functions are given in Appendix 3B.
The final controller produced an overshoot of less than 10% with a rise time of 0.5 seconds and zero steady-state error. If no overshoot is desired this can also be accomplished by tuning the controller some more. The controller was tuned by following the guidelines presented in this chapter. The controller was also tested for robustness by varying the location of the left-half plane pole. Moreover, the dc gain was varied as well as the command input to the plant. Responses to these variations are listed in Tables 3.7 through 3.9.

The graphical response of the plant and controller are also given. Figure 3.13 shows the response of the system after first tuning the controller. Figures 3.14 through 3.16 show the response of the controller to variations

in the plant parameters. These responses show that variation in plant parameters in only one direction produced some steady-state error. This steady-state error could not be explained and should be investigated in future research. Figure 3.17 shows the response of the system to a ramp command. Notice the apparent lack of a transient response which cannot be explained at this point.

To test the controller robustness, disturbances at the input and output were simulated. To simulate a disturbance at the input, a unit step response disturbance was added at the input to the plant after it had reached steady state error at approximately t = 1.75 seconds. Similarly, a step disturbance was simulated at the output and then at both the input and output. Figure 3.18 shows the response to a step disturbance at the input, Figure 3.19 shows the response to a step disturbance at the output, and Figure 3.20 shows the response to disturbances at the input and output simultaneously. These responses show that the controller has good disturbance rejection capability.

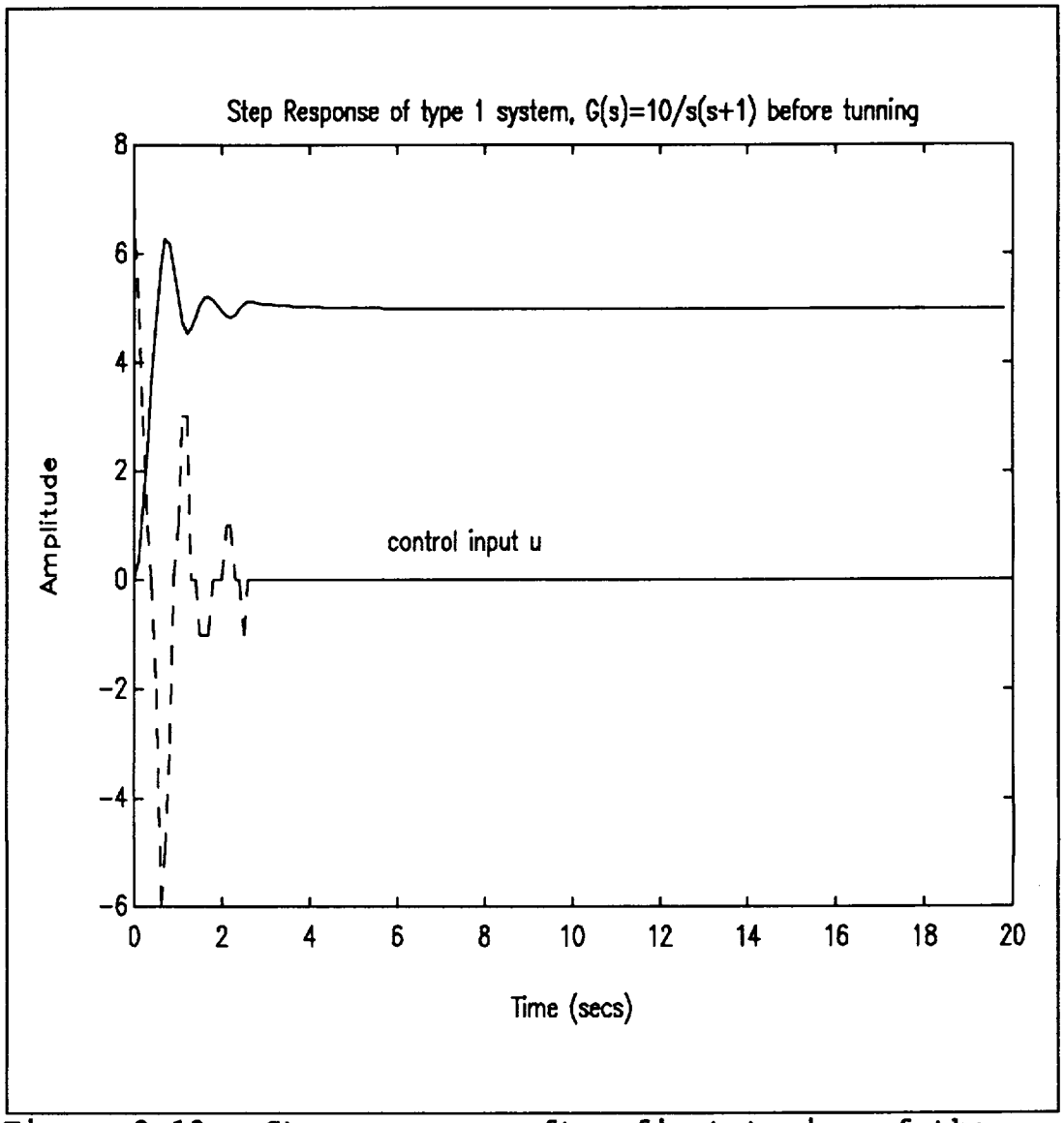

Figure 3.13. Step response after first tuning of the controller, control input included.

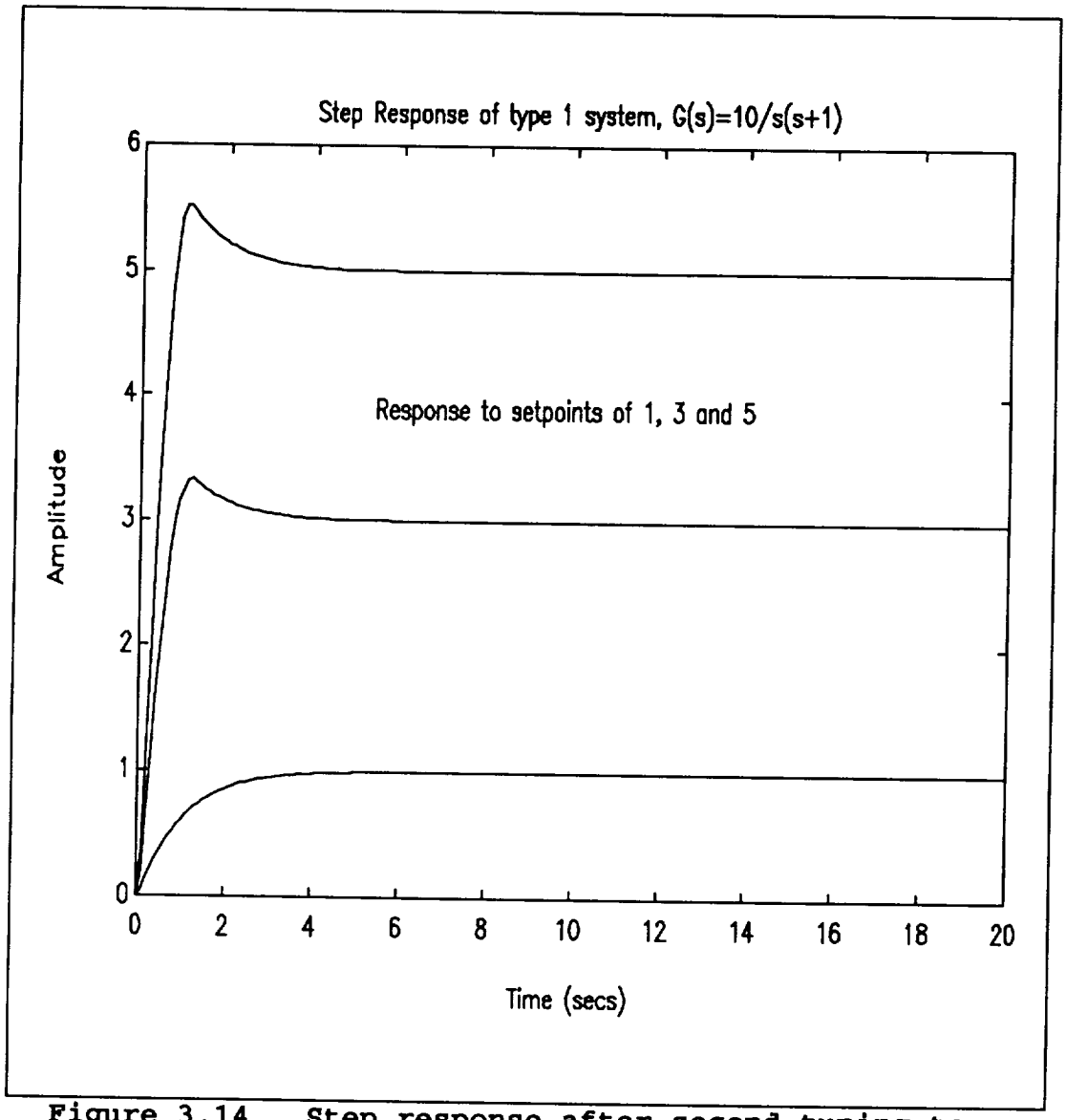

Figure 3.14. Step response after second tuning to different command inputs.

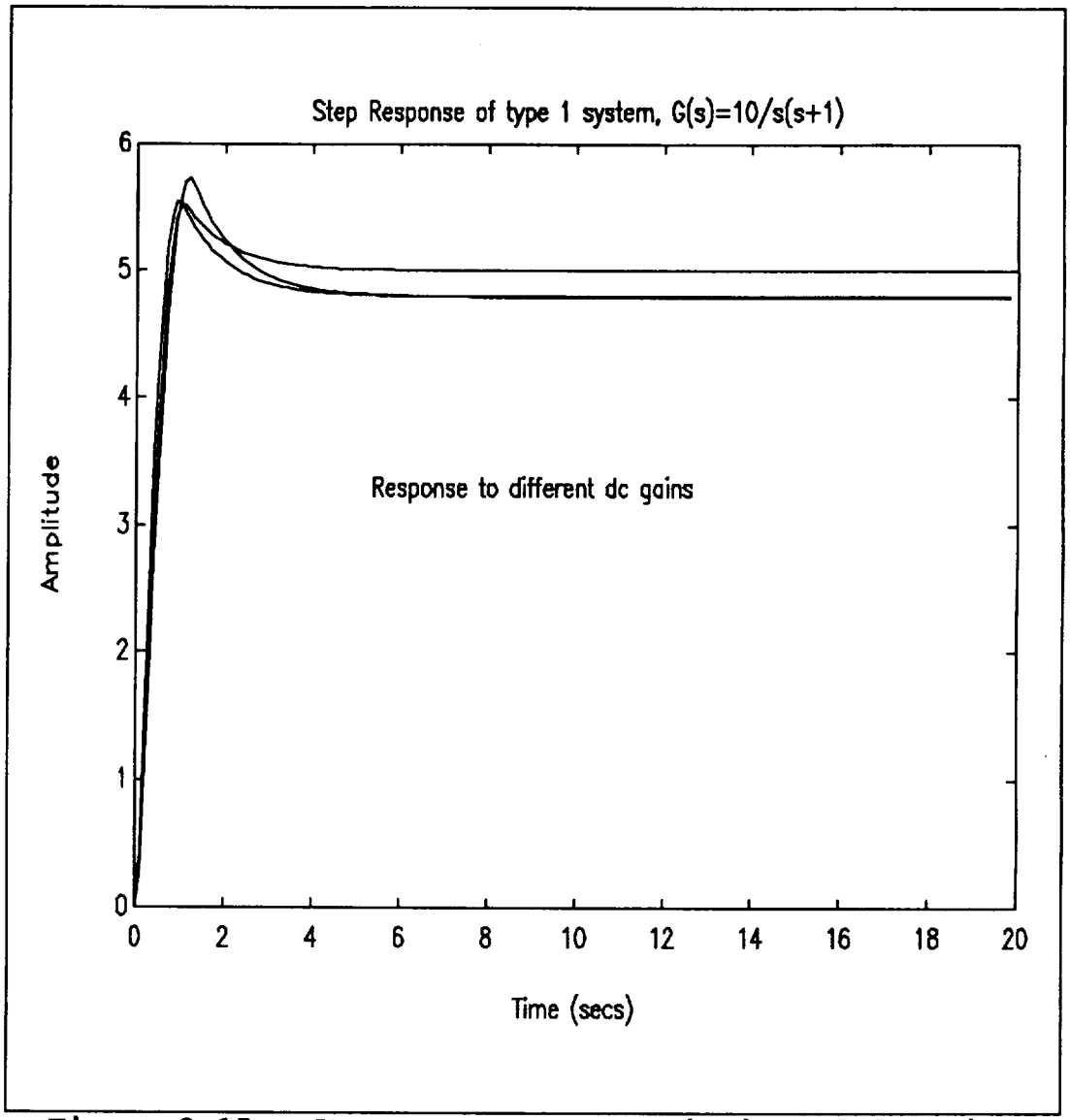

Figure 3.15. Step response to variation of dc gain.

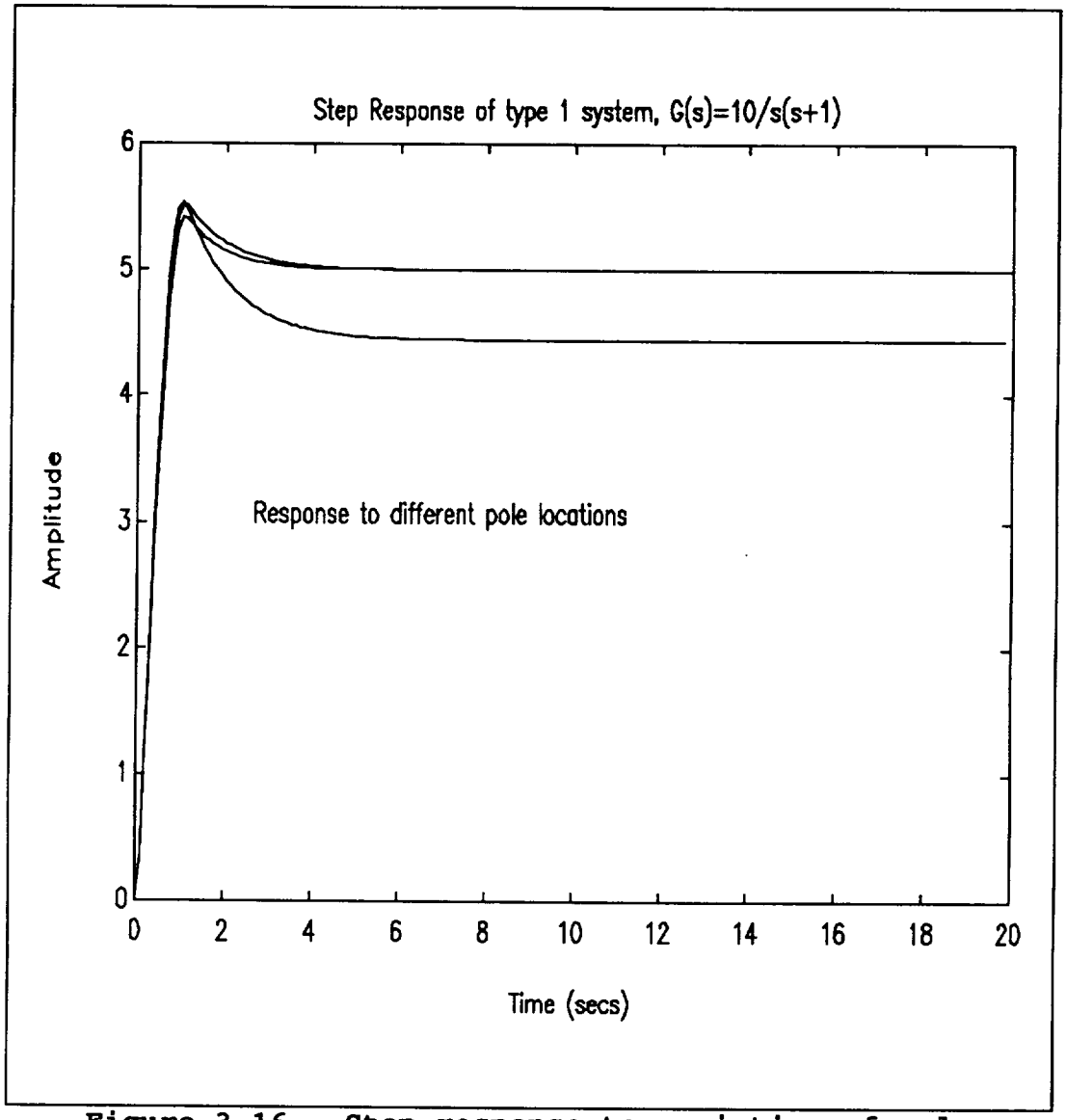

Figure 3.16. Step response to variation of pole location.

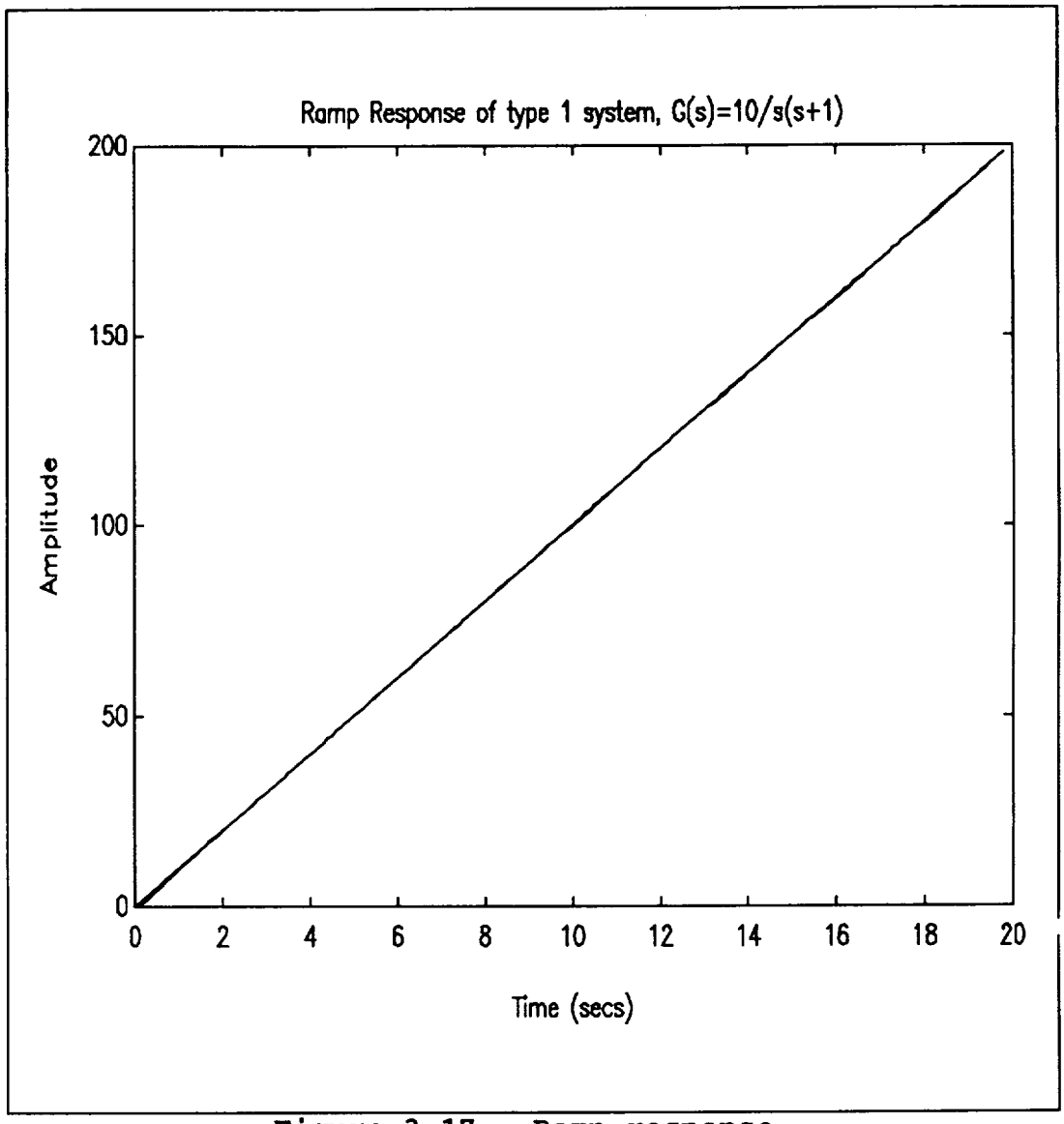

Figure 3.17. Ramp response.

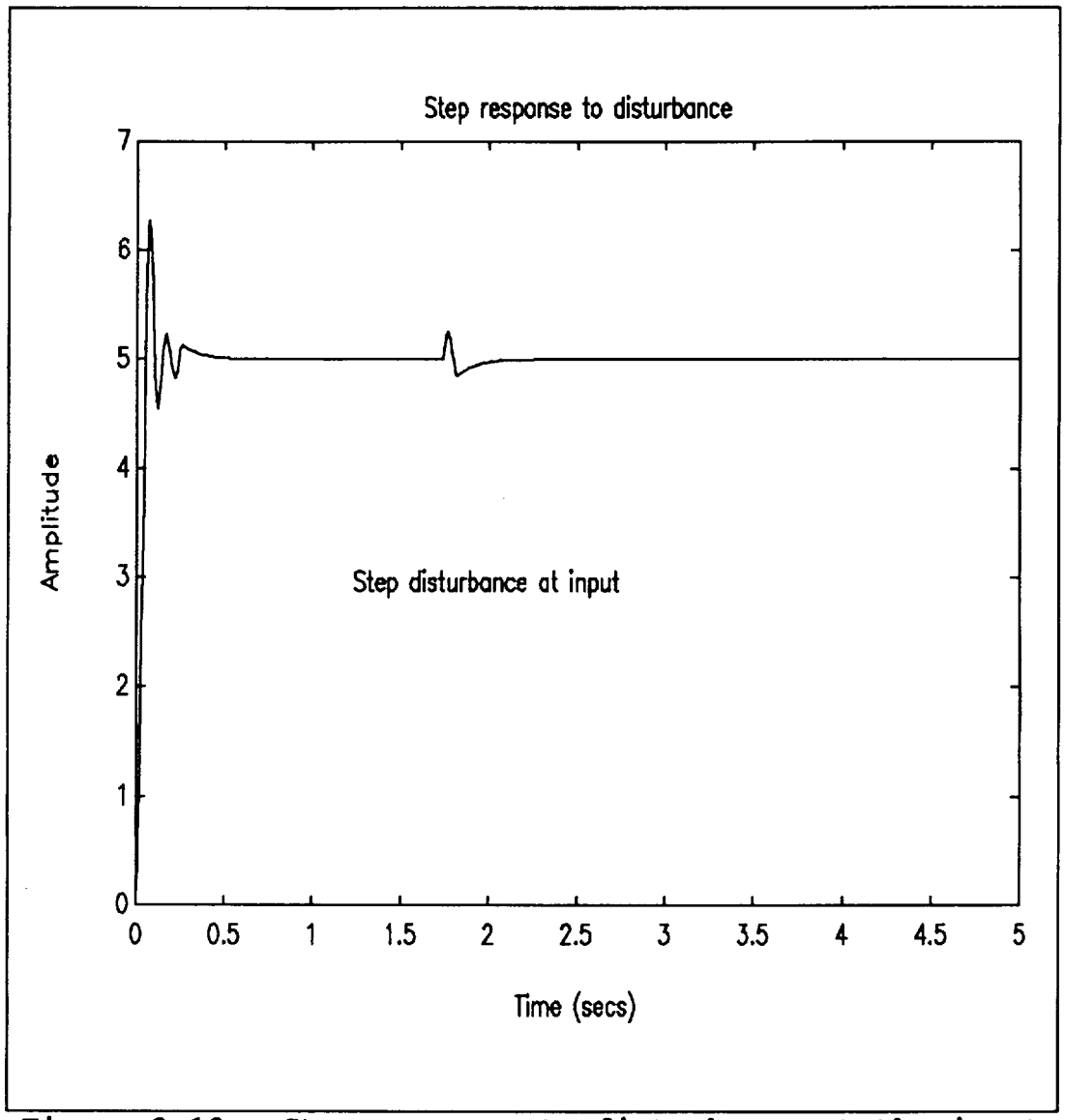

Figure 3.18. Step response to disturbance at the input of the plant.

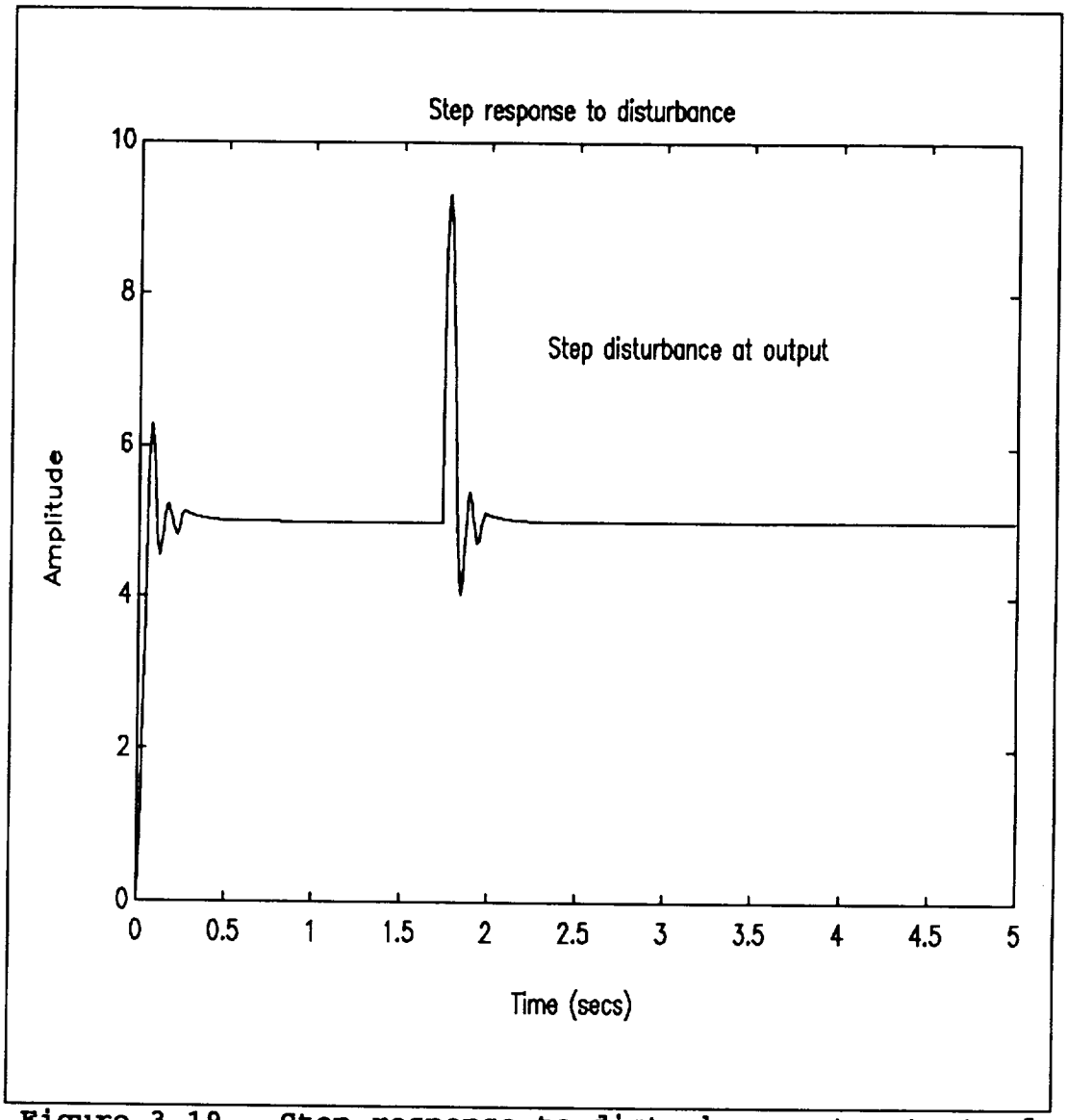

Figure 3.19. Step response to disturbance at output of the plant.

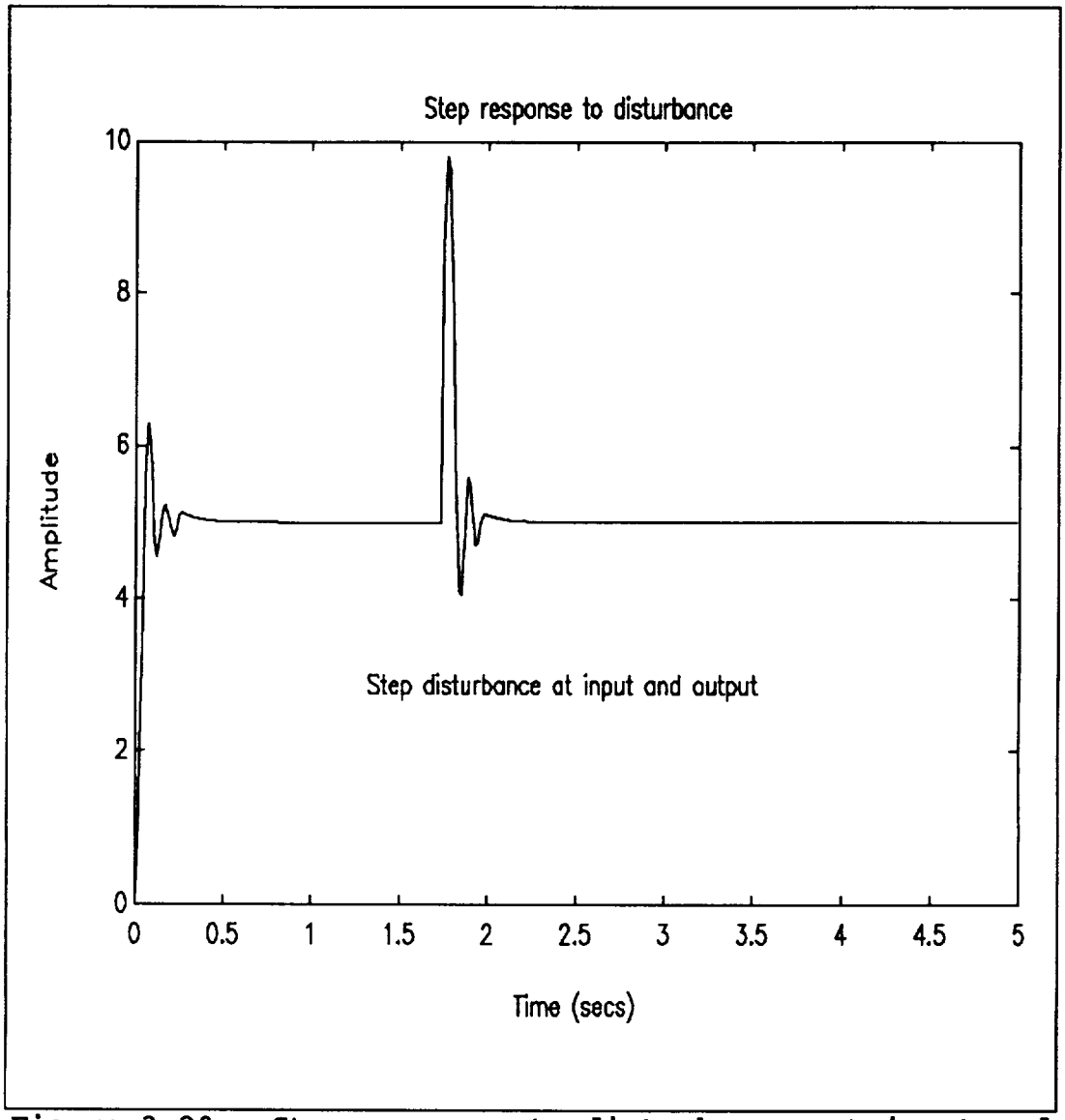

Figure 3.20. Step response to disturbances at input and output of the plant.

$G(s) = \frac{10}{s(s+1)}$	%O.S	e_{ss}	t_s(sec)	t_r(sec)
Command Input r = 5	10.0	0.0	3.4	0.5
Command Input r = 3	10.0	0.0	3.4	0.5
Command Input r = 1	10.0	0.0	3.4	0.5

Table 3.7. Results of varying the command input.

G(s)	% O.S	e_{ss}	t_s (sec)	t_r (sec)
$\frac{10}{s(s+1)}$	10.0	0.0	3.4	0.5
$\frac{10}{s(s+0.8)}$	10.6	0.56	4.6	0.5
$\frac{10}{s(s+1.2)}$	10	0.0	3.4	0.5

Table 3.8. Results of varying the pole locations by 20%.

G(s)	%O.S	e_{ss}	t_s (sec)	t_r (sec)
$\frac{10}{s(s+1)}$	10.0	0.0	3.4	0.5
$\frac{8}{s(s+1)}$	14.4	0.21	4.6	0.5
$\frac{12}{s(s+1)}$	9.4	0.27	3.9	0.5

Table 3.9. Results of varying the dc gain by 20%.

3.6 Design Example Three

In this example integral action is added to the loop gain of Design Example One by putting an integrator in series with the plant. This cascade connection yields a new third order type one system

$$G(s) = \frac{0.25}{s(s+0.2)(s+0.25)}.$$

The transfer function of this plant is the same as of that of Example

One except it has an additional pole at zero. The reason for this is to investigate whether an integrator will yield zero steady-state errors as the plant parameters vary. This is the classical linear control approach. This approach can also be compared to the way Pedrycz (1989) introduced integral action. This was done by adding a third input to the controller: a sequence of sum of errors. For this example, we show that changing the pole locations by plus or minus 20% has no effect on the response of the system. In fact, the response of the system will not change at all. Only insignificant variation in the transient response of the system is noticeable, everything else is almost identical.

Figures 3.21 and 3.22 show the step response of the closed-loop system to variations in pole location and variations in dc gain. The sampling period for this system was T = 0.1 seconds. On the other hand, these responses are significantly slower than the open-loop system. The approach in Pedrycz seems to yield better responses, but they did not test the robustness properties.

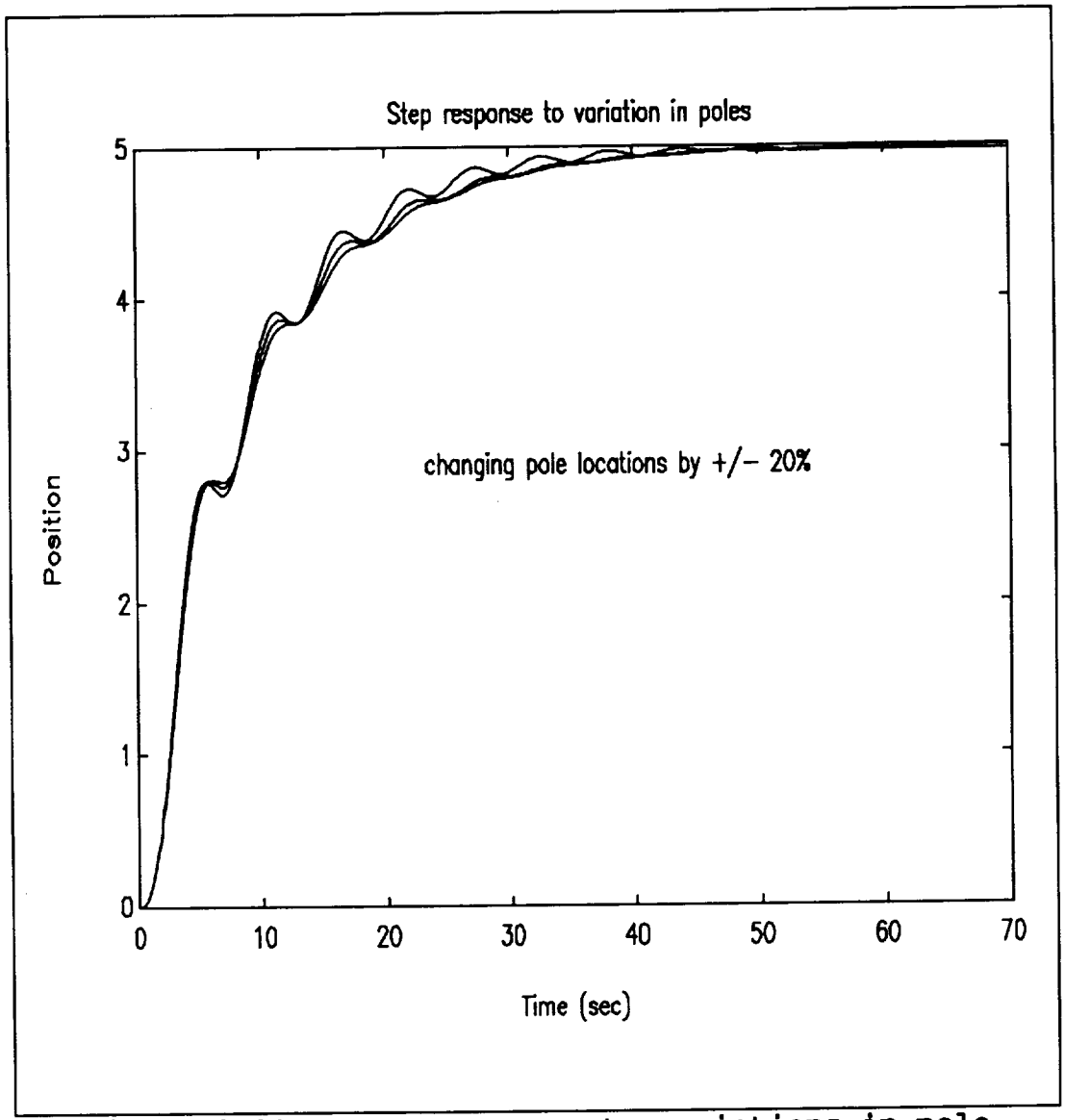

Figure 3.21. Step response to variations in pole location.

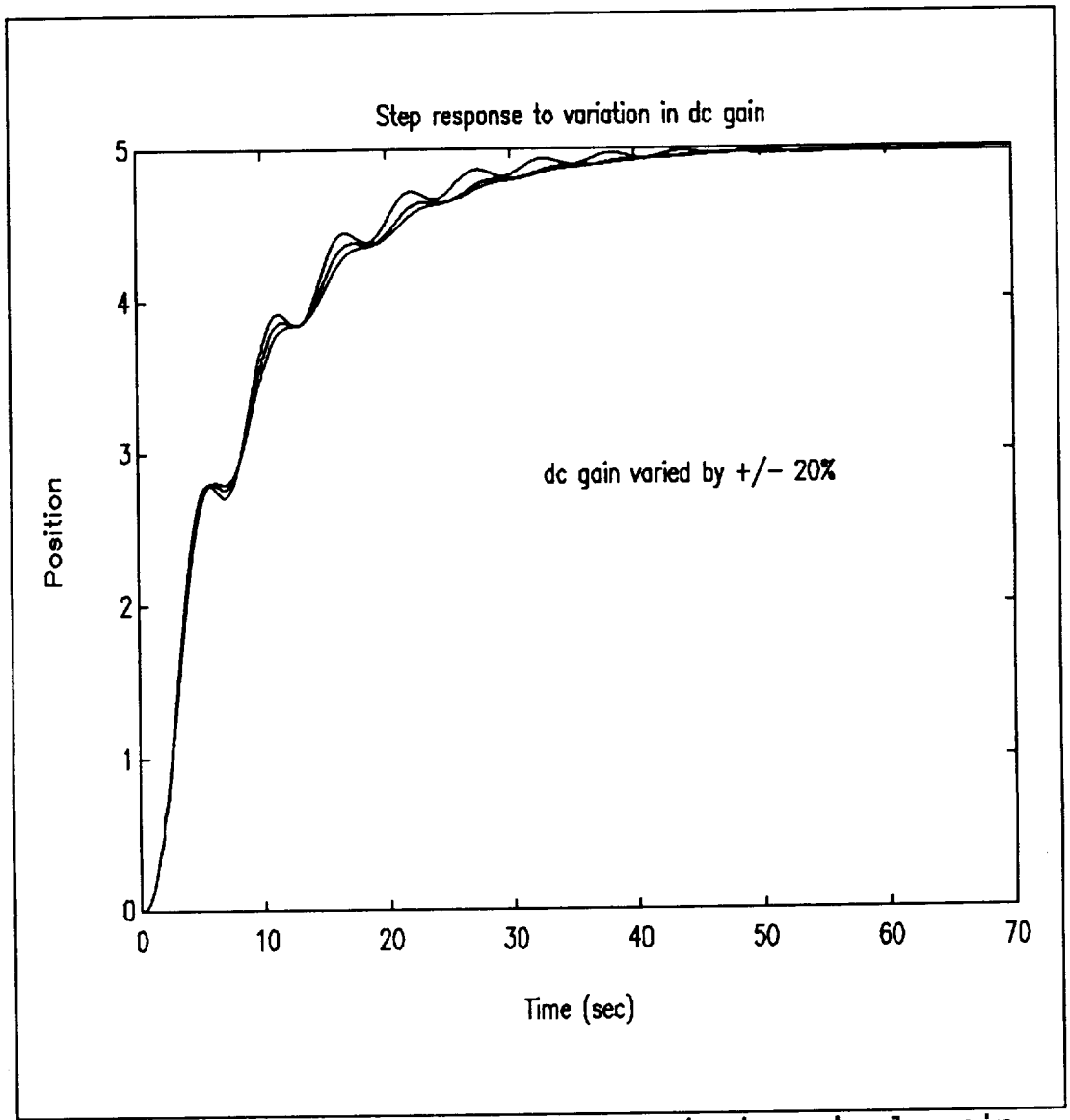

Figure 3.22. Step response to variations in dc gain.

3.7 Design Example Four

Until now we have shown that fuzzy logic controllers work well for stable plants regardless of the order and the type. In addition, we established that fuzzy controllers are robust with regard to uncertainties in plant parameters and to disturbances at the input, output, or both. In this example we will show that fuzzy controllers can do more than control stable plants. In particular, we will show that fuzzy controllers can handle non-minimum phase plants as effectively as they do stable plants. In this example, the plant is given by its transfer function

$$G(s) = \frac{(s-1)}{(s-2)(s+1)}.$$

This transfer function has a right-half plane zero and a right-half plane pole. Moreover, this type of plant is hard to control using classical control techniques. The plant was discretized at a sampling period of 0.001. Using a fuzzy controller, this plant was brought under control fairly quickly. Figure 3.23 shows the step response of the controlled unstable plant. Notice the complete absence of undershoot in the response. Such undershoot would be present if any stabilizing linear controller had been used.

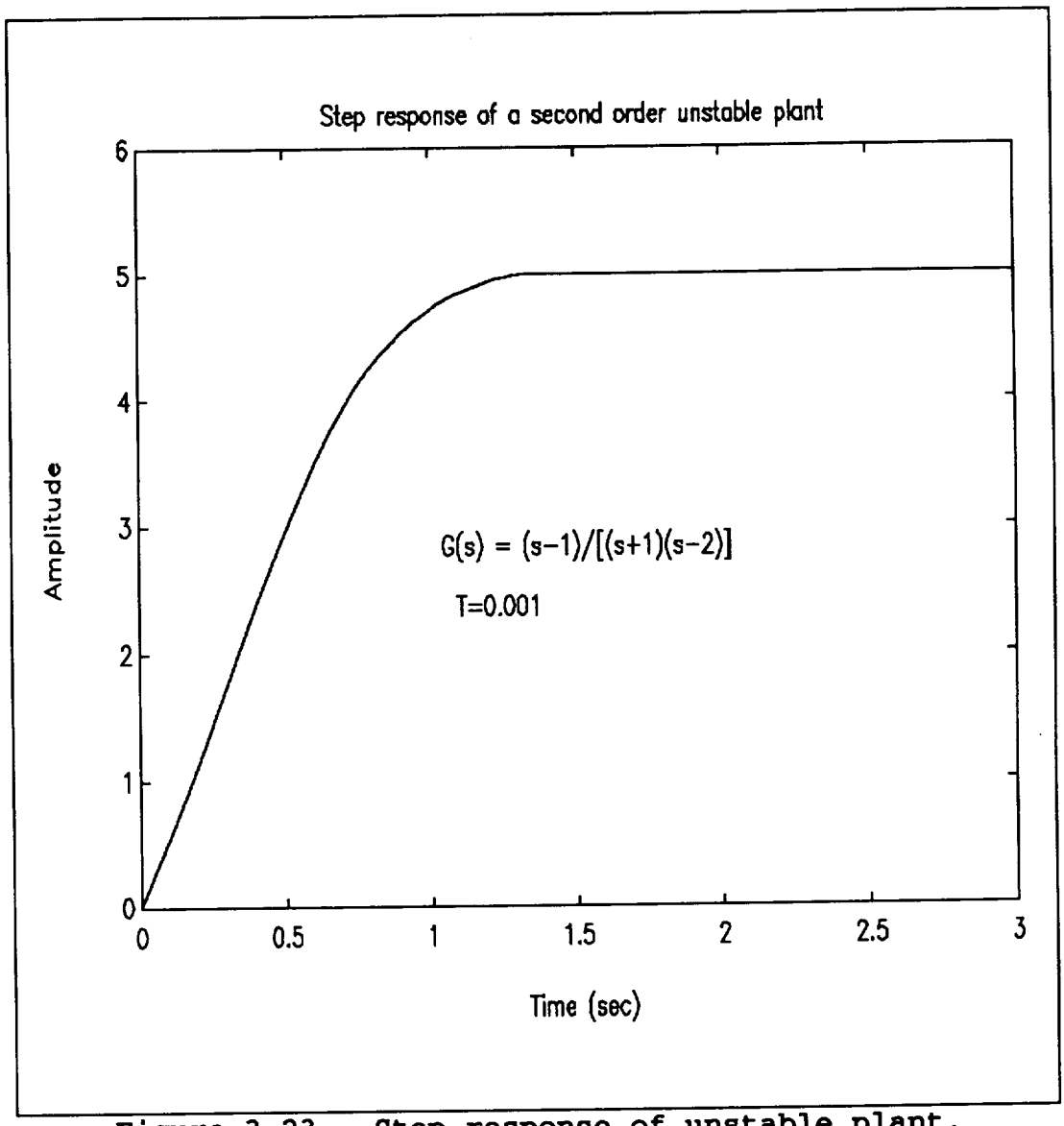

Figure 3.23. Step response of unstable plant.

3.8 Conclusions

In this chapter we introduced a basic fuzzy logic control system design procedure. The procedure includes new tuning rules. Four examples are given that illustrate the application of the design procedure. These examples also show the capabilities and robustness of fuzzy logic control systems.

The lack of analytical tools kept us from further analysis of the results. However, some peculiarities of fuzzy logic control systems are noted. For example, variations of plant parameters in one direction caused no steady-state errors, but variations in the opposite direction did introduce steady-state errors. Also, the ramp response showed no transients at all. Finally, the control design for the non-minimum phase plant shows that fuzzy logic controllers are not bounded by the limitations of linear controllers. These examples highlight the importance of developing analytical tools to analyze fuzzy logic control systems and demonstrate their limitations.

The examples showed that the closed-loop system satisfied both properties of scaling and adaptivity, that is, superposition over the range considered.

Appendix 3A

Rule Base and Fuzzy Sets For Example One

$$G(s) = \frac{5}{(5s+1)(4s+1)}$$

Rule Base

IF (error IS PB) AND (derror IS NB) THEN output=PB

IF (error IS PB) AND (derror IS NS) THEN output=PB

IF (error IS PB) AND (derror IS Z) THEN output=PB

IF (error IS PB) AND (derror IS PS) THEN output=PS

IF (error IS PB) AND (derror IS PB) THEN output=PB

IF (error IS PS) AND (derror IS NB) THEN output=Z

IF (error IS PS) AND (derror IS NS) THEN output=PS

IF (error IS PS) AND (derror IS Z) THEN output=PB

IF (error IS PS) AND (derror IS PS) THEN output=PB

IF (error IS PS) AND (derror IS PB) THEN output=PB

IF (error IS Z) AND (derror IS NB) THEN output=NS

IF (error IS Z) AND (derror IS NS) THEN output=NS

IF (error IS Z) AND (derror IS Z) THEN output=Z

IF (error IS Z) AND (derror IS PS) THEN output=PS

IF (error IS Z) AND (derror IS PB) THEN output=PB

IF (error IS NS) AND (derror IS NB) THEN output=PB

IF (error IS NS) AND (derror IS NS) THEN output=PS

IF (error IS NS) AND (derror IS Z) THEN output=NS

IF (error IS NS) AND (derror IS PS) THEN output=NB

IF (error IS NS) AND (derror IS PB) THEN output=NB

IF (error IS NB) AND (derror IS NB) THEN output=NB

IF (error IS NB) AND (derror IS NS) THEN output=NS

IF (error IS NB) AND (derror IS Z) THEN output=NB

IF (error IS NB) AND (derror IS PS) THEN output=NB

IF (error IS NB) AND (derror IS PB) THEN output=NB

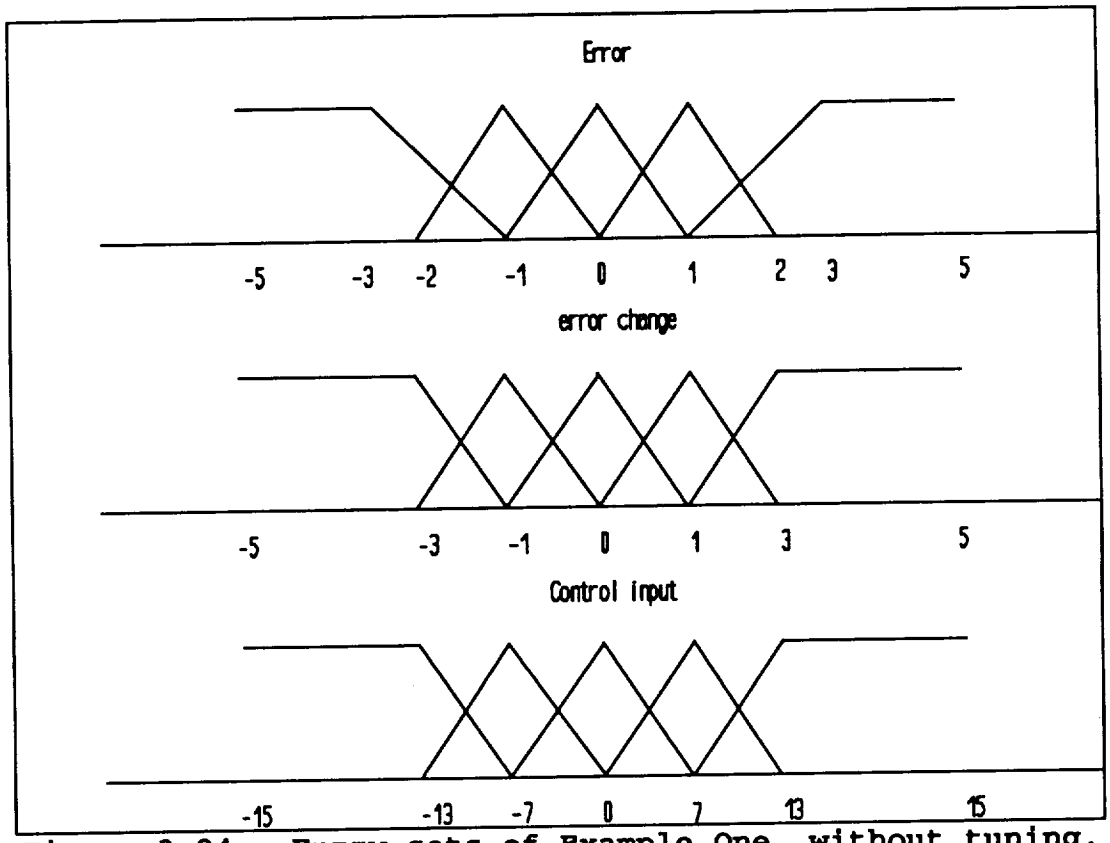

Figure 3.24. Fuzzy sets of Example One, without tuning.

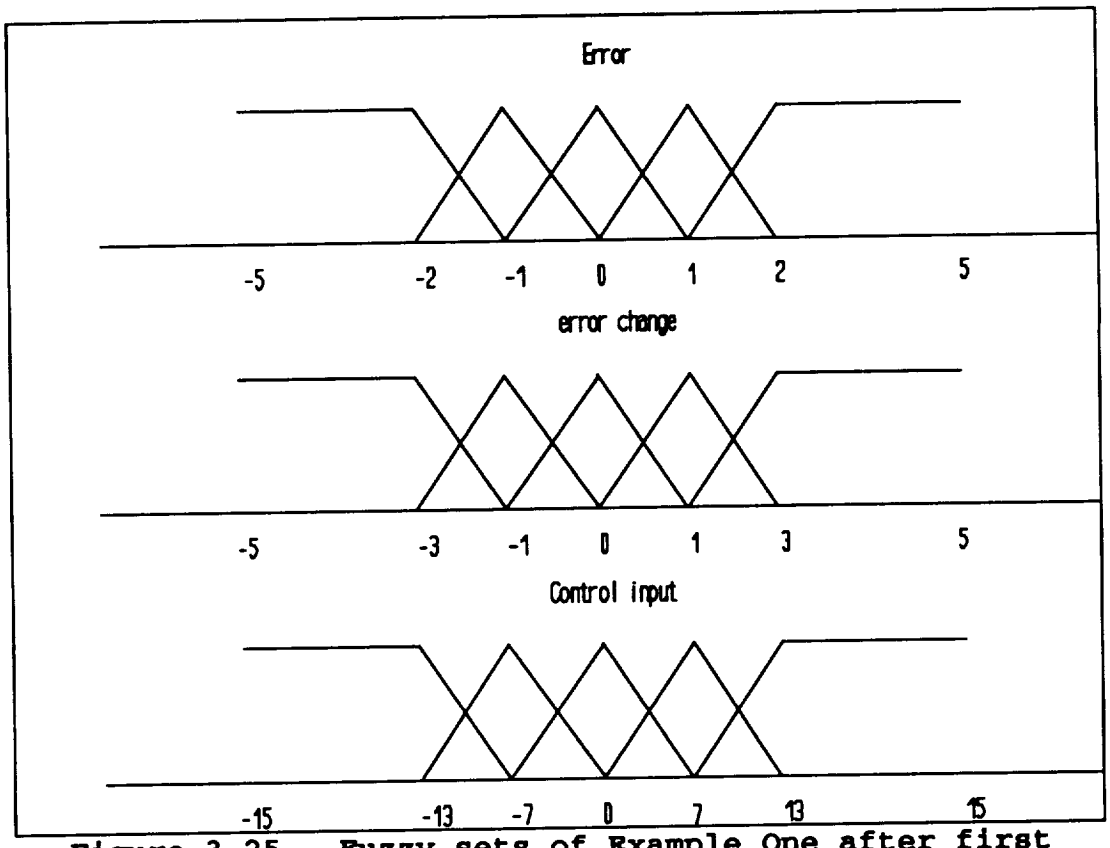

Figure 3.25. Fuzzy sets of Example One after first tuning.

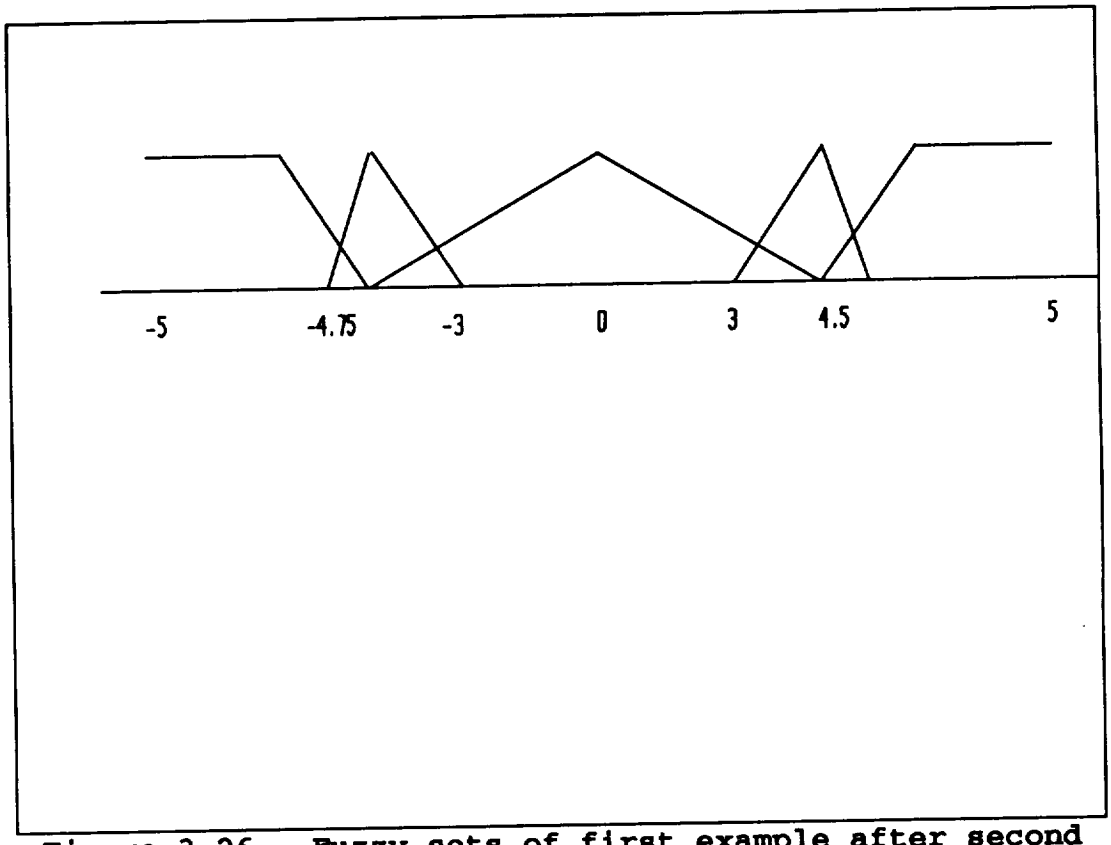

Figure 3.26. Fuzzy sets of first example after second tuning.

Appendix 3B

The Rule Base and The Fuzzy Sets For Example Two

$$G(s) = \frac{10}{s(s+1)}$$

Rule Base

IF (error IS PB) AND (derror IS NB) THEN output=PS

IF (error IS PB) AND (derror IS NS) THEN output=PS

IF (error IS PB) AND (derror IS Z) THEN output=PB

IF (error IS PB) AND (derror IS PS) THEN output=PB

IF (error IS PB) AND (derror IS PB) THEN output=PB

IF (error IS PS) AND (derror IS NB) THEN output=Z

IF (error IS PS) AND (derror IS NS) THEN output=Z

IF (error IS PS) AND (derror IS Z) THEN output=PS

IF (error IS PS) AND (derror IS PS) THEN output=PB

IF (error IS PS) AND (derror IS PB) THEN output=PB

IF (error IS Z) AND (derror IS NB) THEN output=NS

IF (error IS Z) AND (derror IS NS) THEN output=NS

IF (error IS Z) AND (derror IS Z) THEN output=Z

IF (error IS Z) AND (derror IS PS) THEN output=PS

IF (error IS Z) AND (derror IS PB) THEN output=PB

IF (error IS NS) AND (derror IS NB) THEN output=NB

IF (error IS NS) AND (derror IS NS) THEN output=NB

IF (error IS NS) AND (derror IS Z) THEN output=NS

IF (error IS NS) AND (derror IS PS) THEN output=Z

IF (error IS NS) AND (derror IS PB) THEN output=PS

IF (error IS NB) AND (derror IS NB) THEN output=NB

IF (error IS NB) AND (derror IS NS) THEN output=NB

IF (error IS NB) AND (derror IS Z) THEN output=NB

IF (error IS NB) AND (derror IS PS) THEN output=NS

IF (error IS NB) AND (derror IS PB) THEN output=NS

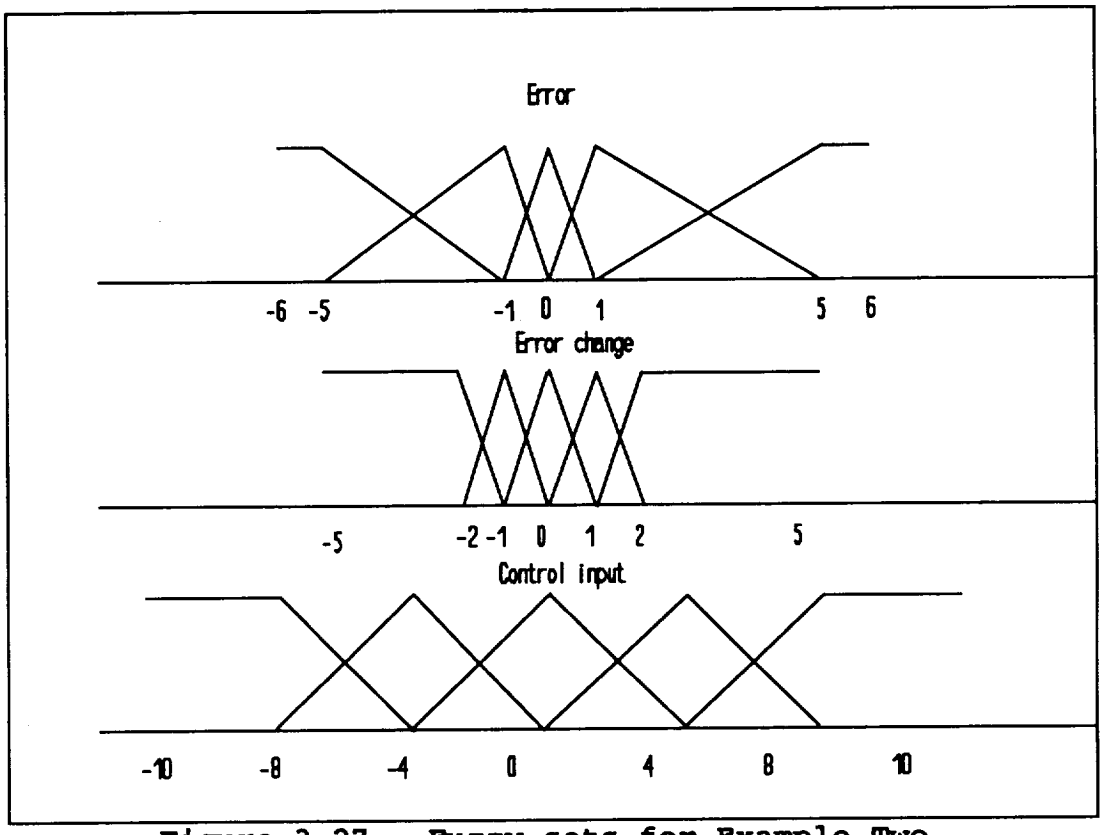

Figure 3.27. Fuzzy sets for Example Two.

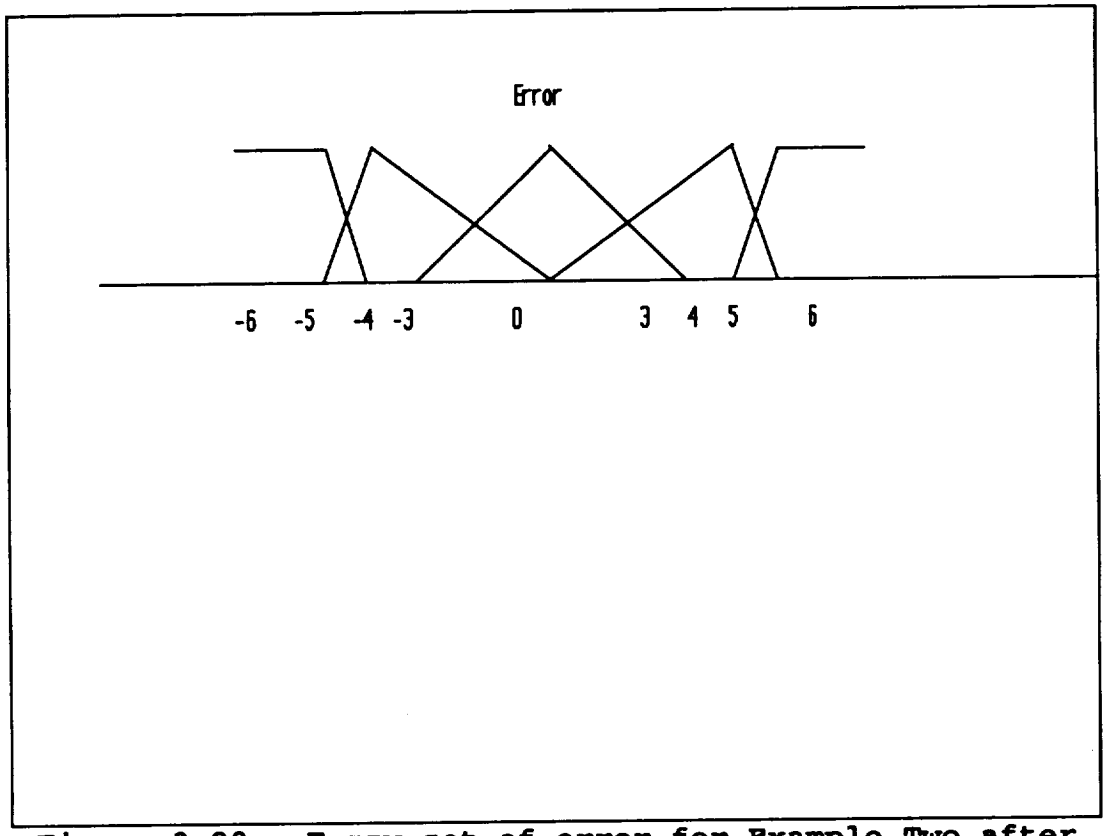

Figure 3.28. Fuzzy set of error for Example Two after tuning.

Appendix 3C

The Rule Base and The Fuzzy Sets For Example Four

$$G(s) = \frac{(s-1)}{(s+1)(s-2)}$$

IF (error IS PB) AND (derror IS NB) THEN output=Z

IF (error IS PB) AND (derror IS NS) THEN output=PS

IF (error IS PB) AND (derror IS Z) THEN output=PB

IF (error IS PB) AND (derror IS PS) THEN output=PB

IF (error IS PB) AND (derror IS PB) THEN output=PB

IF (error IS PS) AND (derror IS NB) THEN output=Z

IF (error IS PS) AND (derror IS NS) THEN output=Z

IF (error IS PS) AND (derror IS Z) THEN output=PS

IF (error IS PS) AND (derror IS PS) THEN output=PS

IF (error IS PS) AND (derror IS PB) THEN output=PB

IF (error IS Z) AND (derror IS NB) THEN output=NB

IF (error IS Z) AND (derror IS NS) THEN output=NS

IF (error IS Z) AND (derror IS Z) THEN output=Z

IF (error IS Z) AND (derror IS PS) THEN output=PS

IF (error IS Z) AND (derror IS PB) THEN output=PB

IF (error IS NS) AND (derror IS NB) THEN output=NB

IF (error IS NS) AND (derror IS NS) THEN output=NS

IF (error IS NS) AND (derror IS Z) THEN output=NS

IF (error IS NS) AND (derror IS PS) THEN output=Z

IF (error IS NS) AND (derror IS PB) THEN output=Z

IF (error IS NB) AND (derror IS NB) THEN output=NB

IF (error IS NB) AND (derror IS NS) THEN output=NB

IF (error IS NB) AND (derror IS Z) THEN output=NB

IF (error IS NB) AND (derror IS PS) THEN output=NS

IF (error IS NB) AND (derror IS PB) THEN output=Z

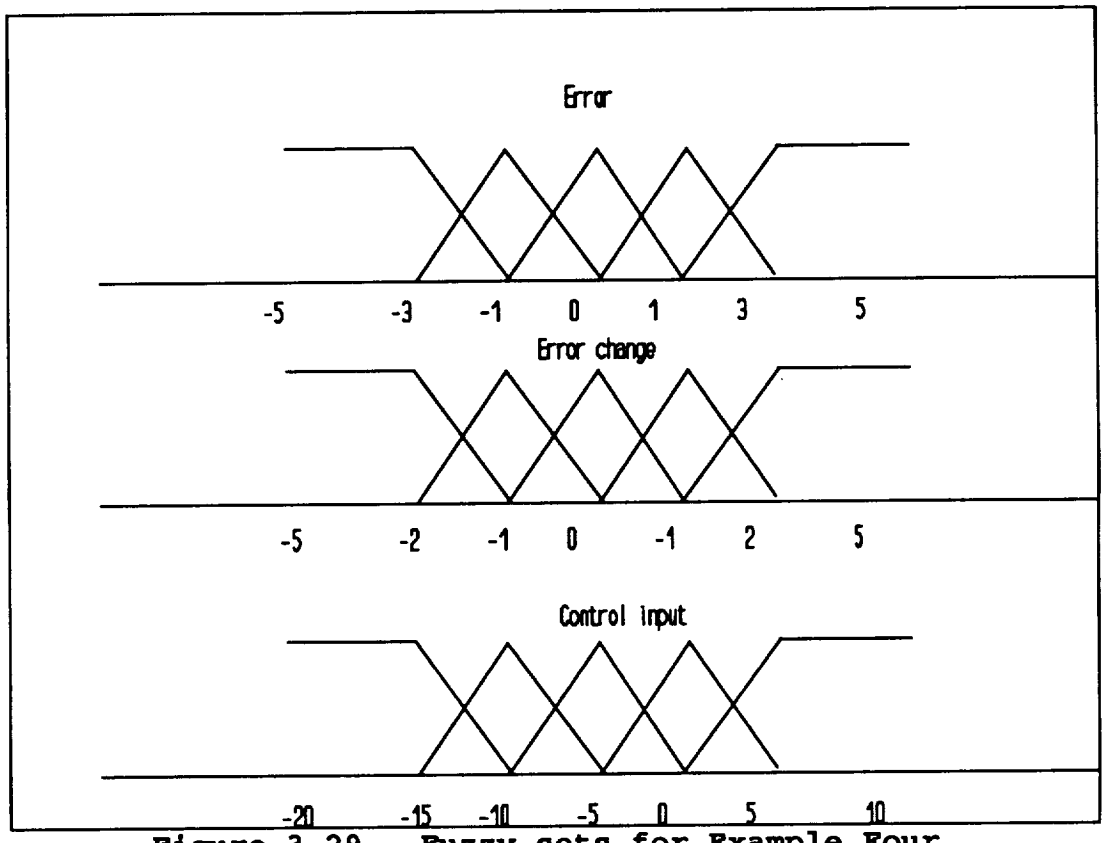

Figure 3.29. Fuzzy sets for Example Four.

Appendix 3D

C code used for simulation for all examples

```
static void Dummy_main ()
{
}
void main()
{

int i;
double derror,old_error,error;
double y[3000];
int u[3000];
int setpoint;
double output;
FILE *fout;
FILE *fopen();
fout = fopen("uns.m", "w");
```

```
for (i=0;i<3000;i++){
y[i]=0;
u[i]=0;
}
y[0]=0;
y[1]=0;
u[0]=0;
u[1]=0;
i=0;
setpoint=5;
y[i+2] = 2.001*y[i+1] - 1.001*y[i] + 0.001*u[i+1] - 0.001*u[i];
old_error = setpoint - y[i+2];
printf("i       ERROR          dERROR          u       Y \n");
for(i=1;i<3000;i++)       {
y[i+2] = 2.001*y[i+1] - 1.001*y[i] + 0.001*u[i+1] - 0.001*u[i];
error = setpoint - y[i+2];
if (error >= 5) error=5;
if (error <=-5) error=-5;
derror = (error - old_error)/0.01;
```

```
if (derror >= 5) derror=5;

if (derror <=-5) derror=-5;

fuzzrule(derror, error, &output);

u[i+2]= output;

printf("%d\t%2f\t%2f\t%6d\t%2f\t\n",i,error,derror,u[i+2],y[i+2]);

fprintf(fout,"%2f\n",y[i+2]);

old_error = error;

}

}
```

CHAPTER FOUR

STABILITY ANALYSIS OF FUZZY LOGIC CONTROLLERS

4.1 Introduction

The issue of stability of fuzzy control systems is of a great deal of importance as with that of any other control design technique. Nevertheless, stability analysis is the least addressed issue in connection with the design and implementation of fuzzy control systems. The reason is that tools that are used in classical control analysis simply do not lead to any results in the area of fuzzy control. This is due to the nature of the controller which is not described by dynamical equations but rather by linguistic statements that describe the inputs and evaluates the output accordingly using a nonlinear defuzzification method, for example, the center of gravity method. However, stability results are starting to emerge giving the designer of fuzzy control systems tools to aid in the analysis and design of fuzzy controllers. Some of the analysis tools, however, are based on assumptions that are impractical. In this chapter we will discuss some of these stability analysis

tools and point out their limitations and advantages. In particular, we will discuss two Lyapunov approaches to establish stability in a fuzzy control system

4.2 Stability Analysis Using Lyapunov's Direct Method

The stability of fuzzy control systems can be studied using Lyapunov's direct method (Kowamato, Tada, Ishigame, and Taniguchi, 1992). The fuzzy model for the system is assumed to be a linear combination of its inputs and has the following form (Tanaka and Sugeno 1985):

R_i : IF X(k) is A^i_1 AND ... AND X(k-n+m) is A^i_n

THEN $X^i(k+1) = a^i_1 X(k) + ... + a^i_n X(k-n+m)$,

where R_i, i=1,2,....m, denotes the i^{th} implication, m is the number of fuzzy implications, $X^i(k+1)$ is the output from the i^{th} implication, a^i_p's, p=0,1,2,..,n, are the consequent parameters, x(k) through x(k-n+m) are state variables, and the A^i_p's are fuzzy sets. The linear subsystem in the consequent part of the i^{th} implication can be written in matrix form

$$A_i X(k),$$

where

$$X(k) = [X(k), X(k-1), ..., X(k-n+m)]^T$$

and

$$A_i = \begin{bmatrix} a^i_1 & a^i_2 & \dots & a^i_{n-1} & a^i_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & . \\ . & 0 & \dots & . & . \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix},$$

The output of the fuzzy system is inferred as follows.

$$X(k+1) = \frac{\sum_{i=1}^{l} W^i A_i X^i(k)}{\sum_{i=1}^{l} W^i},$$

where the W^i's are the weights of each rule that has been applied. For example, weights are used to emphasize the effect of some rules. Next we state Lyapunov's stability theorem.

Theorem 4.1 (Franklin, Powell, and Workman, 1990)

V(x) is a Lyapunov function for the system

$$X(k+1) = f(x),\ f(0)=0, \tag{4.1}$$

if the following conditions hold (V(x) is a scalar function):

1. $V(0) = 0$, and $V(x)$ is continuous in x.

2. $V(x)$ is positive definite; that is $V(x) \leq 0$ with $V(x) = 0$ only if $x = 0$.

3. $\Delta V(x) = V(f(x)) - V(x)$ is negative definite; that is $V(f(x)) - V(x) \leq 0$ with $\Delta V(x) = 0$ only if $x = 0$.

4. $V(x)$ approaches infinity as $\| x(k) \| \rightarrow \infty$.

The solution X(k)=0 for the system given by (4.1) is globally asymptotically stable if there exists a Lyapunov function in x; that is, if V(x) is a Lyapunov function that satisfies conditions 1, 2, 3, and 4.

Theorem 4.2 (Tanaka and Sugeno, 1990)

The equilibrium of a fuzzy system is asymptotically stable in the large if there exists a common positive definite matrix P such that

$$A_i^T P A_i - P < 0$$

for i=1, 2,..., m; that is, for all subsystems.

Definition 4.1

A fuzzy system such that all the A_i's are stable matrices is said to be locally stable. A fuzzy system is globally stable if there exists a common positive definite matrix P for all subsystems such that A_i is stable and nonsingular and (4.2) is satisfied.

To find P consider the following relations (Kowamato, Tada, Ishigame, and Taniguchi, 1992)

$$A_1^T P A_1 - P = - Q_1 < 0,$$

$$A_2^T P A_2 - P = - Q_2 < 0,$$

$$\vdots$$

$$A_m^T P A_m - P = - Q_m < 0, \qquad (4.2)$$

where $Q_1, Q_2, ..., Q_m > 0$. In this approach it is assumed that the common positive definite matrix P is 2×2 with real entries, that is

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

P can be modified for computational ease, as follows:

$$P = \begin{bmatrix} p_1 & 1 \\ 1 & p_2 \end{bmatrix},$$

where $p_1 = p_{11}/|p_{12}|$ and $p_2 = P_{22}/|p_{12}|$.

Since P is 2×2, A and Q are also 2×2 and of the form

$$A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$$

and

$$Q = \begin{bmatrix} q_1 & q_2 \\ q_2 & q_4 \end{bmatrix}.$$

Substituting these forms in equation (4.2) yields

$$q_1 = - \{ (a_1^2 - 1)P_1 + a_3^2 p_2 + 2a_1 a_3 \},$$

$$q_2 = - \{ a_1 a_2 p_1 + a_3 a_4 p_2 + (a_1 a_4 + a_2 a_3 - 1) \},$$

and

$$q_3 = - \{ a_2^2 P_1 + (a_4^2 - 1)p_2 + 2a_2 a_4 \}.$$

Since we need $P > 0$, we have

$$p_1 > 0 \tag{4.3}$$

and

$$p_1 p_2 > 1. \tag{4.4}$$

The condition $Q_1 > 0$ implies $q_1 > 0$ by rewriting it as

$$(a_1^2 - 1)P_1 + a_3^2 p_2 + 2a_1 a_3 < 0. \tag{4.5}$$

The condition $q_1 q_3 - q_2^2 > 0$ gives us

$$a_2^2 p_1^2 + a_3^2 p_2^2 - \{(a_1 a_4 - a_2 a_3)^2 - (a_1^2 + a_4^2) + 1\} p_1 p_2 + 2a_2(a_4 - a_1)p_1 +$$

$$2a_3(a_1 - a_4)p_2 + \{(a_1 a_4 - a_2 a_3)^2 - 2(a_1 a_4 + a_2 a_3) + 1\} < 0. \tag{4.6}$$

Now we can construct the P - region that satisfies equations (4.3) - (4.6).

4.2.1 Example

Suppose we have two systems described by

$$\text{System 1: } X_1(k+1) = A_1 X_1(k)$$

and

$$\text{System 2: } X_2(k+1) = A_2 X_2(k)$$

where

$$A_1 = \begin{bmatrix} 0.9999 & 0.009949 \\ -0.01989 & 0.9899 \end{bmatrix}$$

and

$$A_1 = \begin{bmatrix} 0.9999 & 0.009949 \\ -0.01989 & 0.9899 \end{bmatrix}$$

By applying Equations (4.3) - (4.6) we can plot those equations to find if a common positive definite P matrix exists. The plots for the A matrices are given in Figure 4.1. Since the curves do not intersect, there exists no P region that is common to both A's.

Suppose that the A matrices are changed to

$$A_1 = \begin{bmatrix} 0.9999 & 0.009949 \\ -0.01989 & 0.9899 \end{bmatrix} .$$

and

$$A_1 = \begin{bmatrix} 0.9997 & 0.009949 \\ -0.05969 & 0.9898 \end{bmatrix}$$

The plot for these systems is given in Figure 4.2. In this case, there exists a critical P region, since the curves are tangent to each other.

Finally, if

$$A_1 = \begin{bmatrix} 0.9999 & 0.009949 \\ -0.01989 & 0.9899 \end{bmatrix}$$

and

$$A_1 = \begin{bmatrix} 0.9998 & 0.009949 \\ -0.02985 & 0.9898 \end{bmatrix}$$

There exists a P region that is common to both A's. This is shown in Figure 4.3.

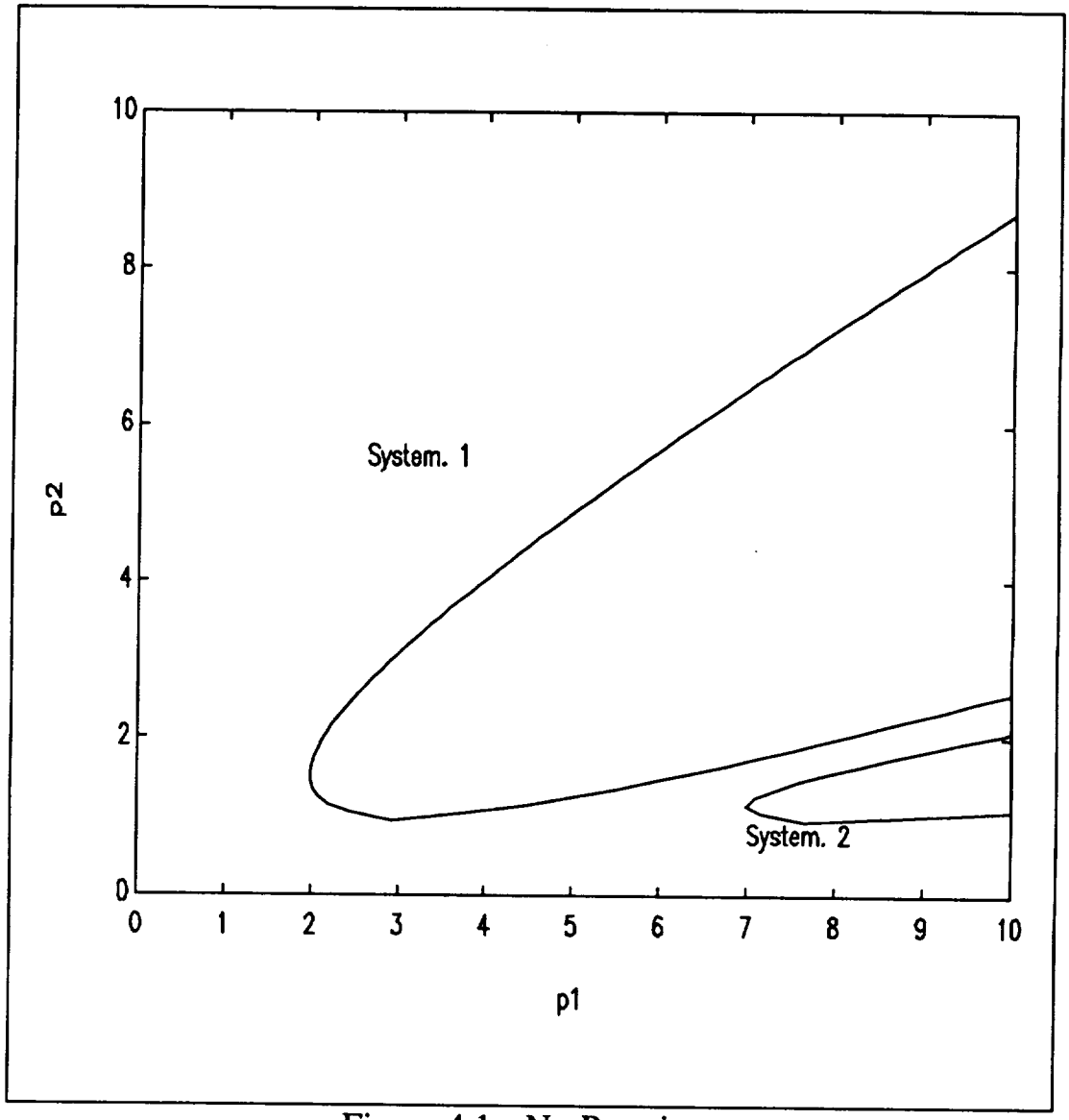

Figure 4.1. No P-region.

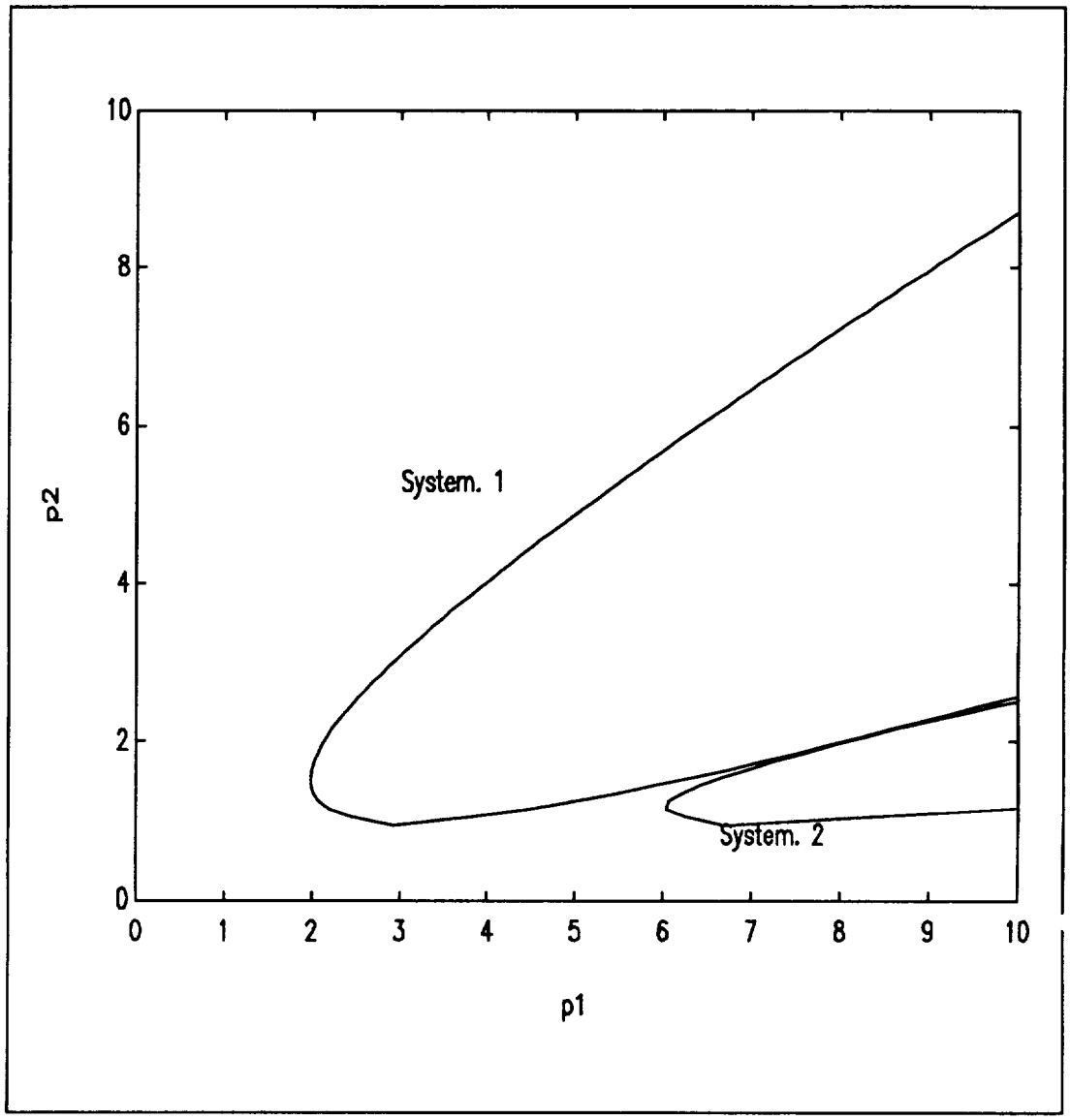

Figure 4.2. Critical P-region.

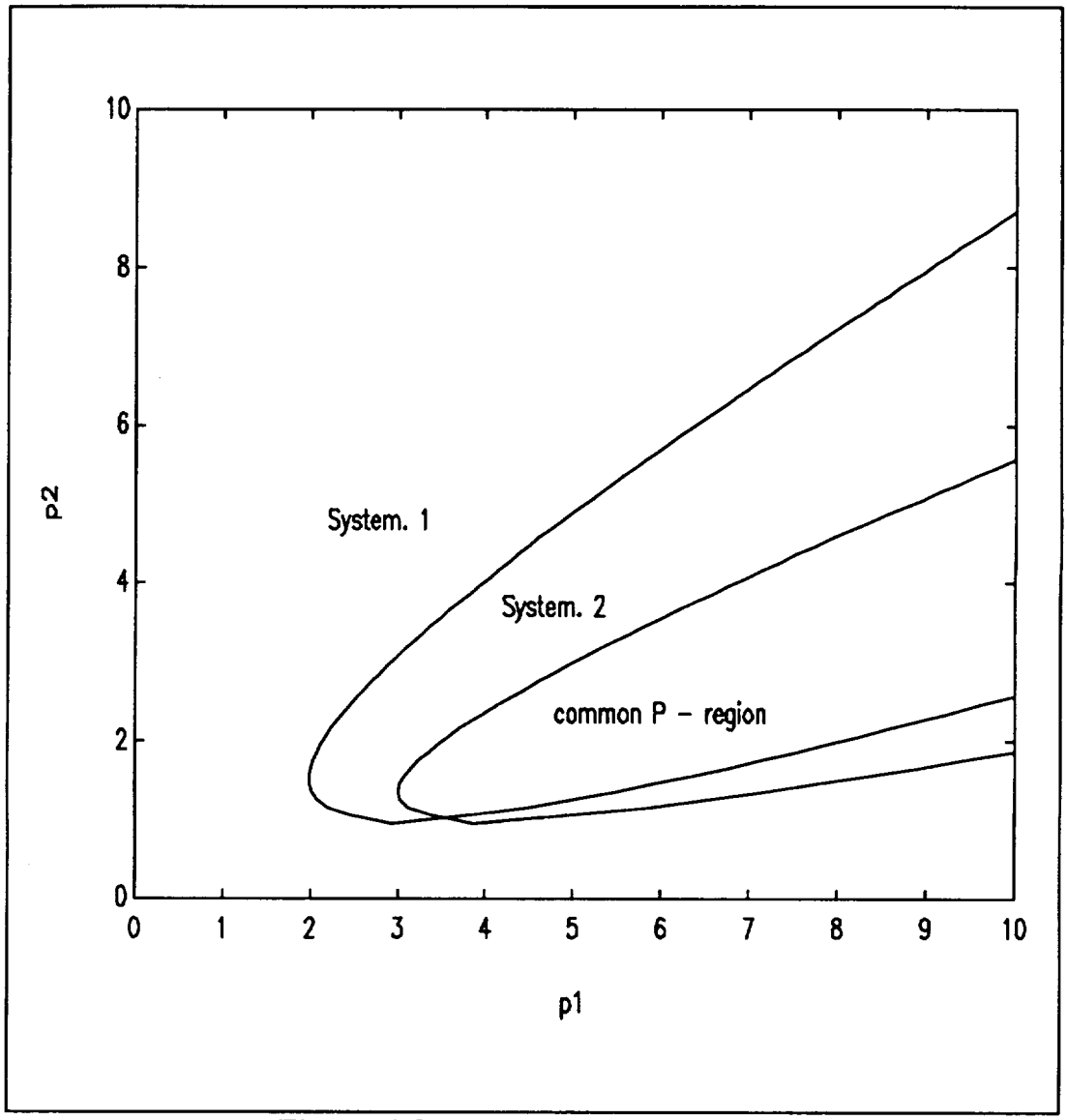

Figure 4.3. The P-region appears.

4.2.2 Conclusions

The stability analysis tools that were presented in this section make stability analysis using Lyapunov's method simple and easy to compute. However, there are disadvantages to this method. The main disadvantage is that it is assumed that the output of the system is a linear combination of the inputs. This is the most important reason for not using this method for stability analysis. In actual fuzzy control system design, the defuzzification algorithm is nonlinear and prohibits the use of this method. Another reason is system order. Once the system order is greater than two, then the calculations to find a positive definite matrix P become too computationally involved.

4.3 Another Approach Using Lyapunov's Method

The method of analysis presented in this section relies on having an accurate mathematical description of the system under control along with the control parameters. This method of stability analysis of fuzzy control systems is based on a partitioning of the state space and applying Lyapunov's criterion to each partition. This approach utilizes only the minimum and maximum bounds on the control output within the partitioned

regions of the state space (Chiu and Chand, 1991). Moreover, this method does not impose any constraints on the rule structure. The fuzzy control system is comprised of the input and output sets and the fuzzy rules. The control rules are of the following form:

$$\text{IF } X_1 \text{ is } A_{i,1} \text{ AND } X_2 \text{ is } A_{i,2} \text{ THEN Y is } B_i,$$

where X_1 and X_2 are the inputs to the controller and Y is the output, the A's and B's are membership functions, and the subscript i denotes the rule number. Therefore, given the inputs X_1 and X_2, we can find the degree of fulfillment w_i of the i^{th} rule which is given by

$$w_i = \min [A^i{}_{,1}(x_1), A^i{}_{,2}(x_2)].$$

The output can be computed using the center of gravity method.

4.3.1 Stability criterion

Consider the linear discrete time model of a controlled plant described by

$$X(k+1) = A\ x(k) + B\ u(k).$$

Define a Lyapunov function candidate V(x) of the form

$$V(x) = X^T P X,$$

where P is a positive definite matrix. Then

$$\Delta V = V(k+1) - V(k) = X^T(k+1)\, P\, X(k+1) - X^T(k)\, P\, X(k)$$

$$= X^T(k)\, [\, A^TPA - P\,]\, X(k) + 2u^T(k)B^TPAX(k) +$$

$$U^T(k)B^TPBu(k).$$

For global asymptotic stability, the following condition must be satisfied:

$$\frac{X^T(-A^TPA + P)X}{X^TX} > \frac{(2u^TB^TPAX) + (u^TB^TpB\,U)}{X^TX}.$$

This implies

$$\lambda_{min}(-A^TPA + P) > \frac{(2u^TB^TPAX) + (u^TB^TPBu)}{|x|^2}. \qquad (4.7)$$

where $A^TPA - P = -Q$.

Equation 4.7 simplifies to

$$\lambda_{min}(-A^TPA + P) > \frac{|u|^2}{|x|^2}\left(2B^TPA\frac{X}{u} + B^TPB\right). \qquad (4.8)$$

If the condition given by Equations (4.7) - (4.8) holds in every region of the state space, then the system is asymptotically stable. It is possible that this method will give inconclusive results in some cells. In particular, in the

following example, this method holds for all the regions of the state space except those near the origin. To account for those cells near the origin, further partitioning of the states near the origin is necessary. If we assume that the controller provides a simple proportional plus derivative control (Mizumoto, 1992),

In continuous time the control law is

$$u = k_1(x,\dot{x})x + k_2(x,\dot{x})\dot{x}. \tag{4.9}$$

4.3.2 Example

To illustrate the application of stability to fuzzy logic controllers, consider a first order system given by its state space representation

$$\dot{x} = -x + u.$$

The discrete form at a sampling period of 0.01 becomes

$$x[k+1] = 0.99x[k] + 0.01u[k]. \tag{4.10}$$

In terms of the error and the rate of change of error, (4.12) becomes

$$e[k] = x[k]$$

and

$$de[k] = x[k] - x[k-1].$$

The state space representation of the discrete system in terms of error and it rate of change becomes

$$\begin{bmatrix} e(k+1) \\ de(k+1) \end{bmatrix} = \begin{bmatrix} 0.99 & 0 \\ -0.01 & 0 \end{bmatrix}$$

To verify the stability of the controller we will need the fuzzy sets of error, rate of change of error, and the control input. Table 4.1 gives the maximum and minimum output condition for each cell. The top horizontal line is the error and the left vertical line gives the rate of change.

	5	0.1	0	-0.1	-5
-5	6 0.37	0.37 -6	-6 -6	-6 -6	-6 -6
-1	6 0.37	0.37 -2.65	-2.65 -2.9	-2.9 -2.6	-6 -2.9
0	6 2.94	2.9 0	0 -2.94	-2.9 -6	-6 -2.9
1	6 2.94	2.9 2.65	2.66 -0.37	-0.37 -2.65	-6 -0.37
5	6 6	6 6	6 -0.37	-0.37 -6	-6 -0.37

Table 4.1. Partitioned state space.

The stability of the controller is verified for all cells except the ones that are shaded. The conservative bounds given by (4.12) fail to establish

stability for that region. However, it has been established that a fuzzy controller behaves as a proportional plus derivative controller near the equilibrium point (Mizumoto, 1992). Using this result and the continuous time state-space equation for the system and substituting the control law given by equation (4.11) we have

$$\dot{x} = (-1 + k_1)x + k_2\dot{x}$$

or

$$\dot{x} = \frac{-1 + k_1}{1 - k_2}x.$$

Using a Lyapunov function of the form $V=x^Tx$, it can be shown that the system is asymptotically stable if

$$\frac{(-1 + k_1)}{(1 - k_2)} < 0.$$

This implies that both k_1 and k_2 are either larger or less than one. Figure 4.4 shows the step response of the system. The fuzzy sets and the complete rule

base for this example are given at the end of this chapter.

4.4 Conclusion

This second method of stability analysis also has significant limitations. The first and major limitation of this method is the importance of an accurate model to do the analysis. Second, we have shown that to do stability analysis on a controller that takes the error and its rate of change as input, requires us to take the rate of change of the system equations as we have done in the example. This implies that the system order will double. As we go to systems greater than one the computational difficulties would be immense.

These approaches to stability analysis are only meant as an overview. More work needs to be done in this area.

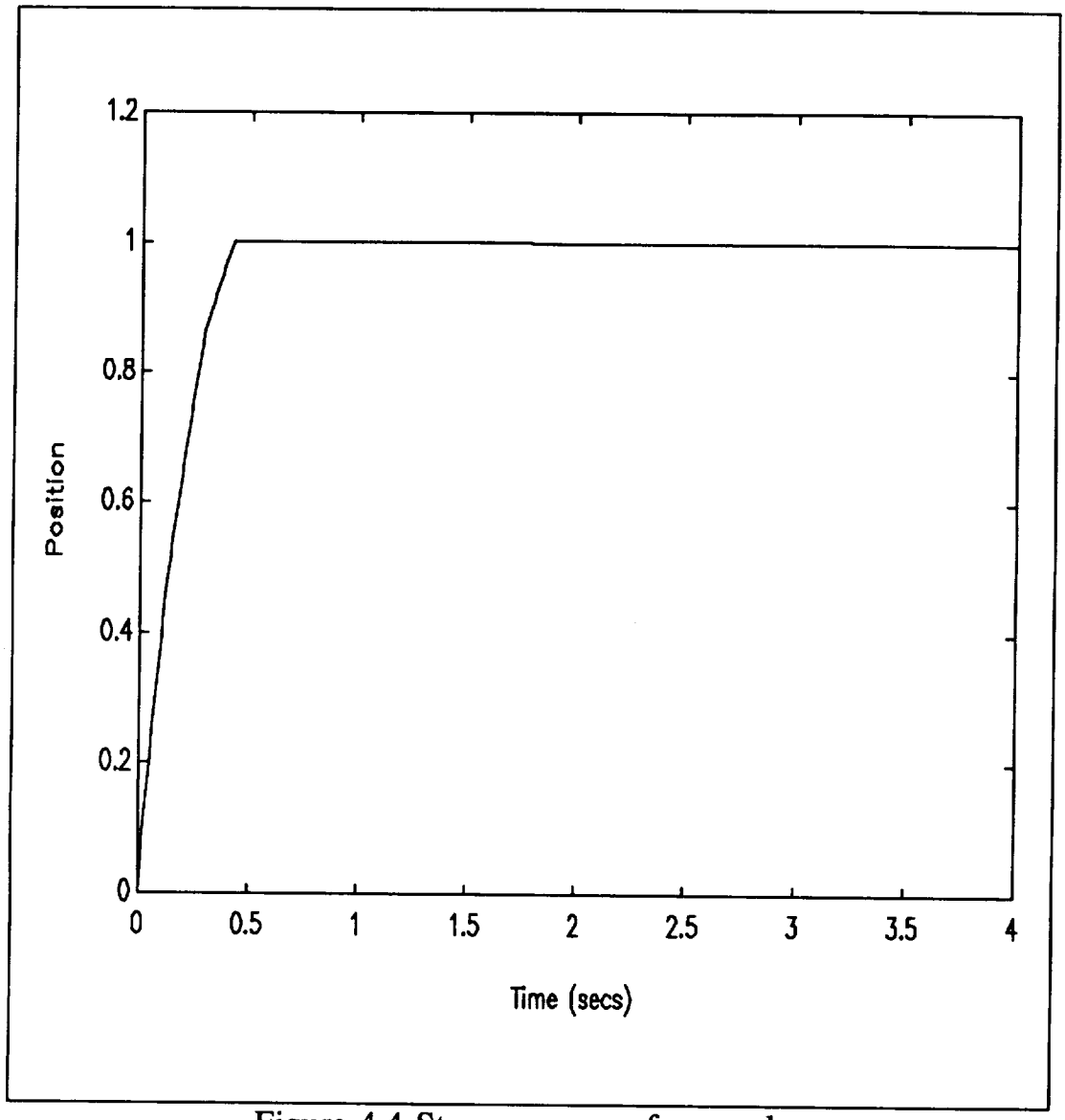

Figure 4.4 Step response of example.

Appendix 4B

The Rule Base And The Fuzzy Sets For Example

$$G(s) = \frac{1}{(s+1)}.$$

RULE BASE

IF (error IS PB) AND (derror IS ANY) THEN output=PB

IF (error IS PS) AND (derror IS NB) THEN output=Z

IF (error IS PS) AND (derror IS NS) THEN output=Z

IF (error IS PS) AND (derror IS Z) THEN output=PS

IF (error IS PS) AND (derror IS PS) THEN output=PS

IF (error IS PS) AND (derror IS PB) THEN output=PB

IF (error IS Z) AND (derror IS NB) THEN output=NB

IF (error IS Z) AND (derror IS NS) THEN output=NS

IF (error IS Z) AND (derror IS Z) THEN output=Z

IF (error IS Z) AND (derror IS PS) THEN output=PS

IF (error IS Z) AND (derror IS PB) THEN output=PB

IF (error IS NS) AND (derror IS NB) THEN output=NB

IF (error IS NS) AND (derror IS NS) THEN output=NS

IF (error IS NS) AND (derror IS Z) THEN output=NS

IF (error IS NS) AND (derror IS PS) THEN output=Z

IF (error IS NS) AND (derror IS PB) THEN output=Z

IF (error IS NB) AND (derror IS ANY) THEN output=NB

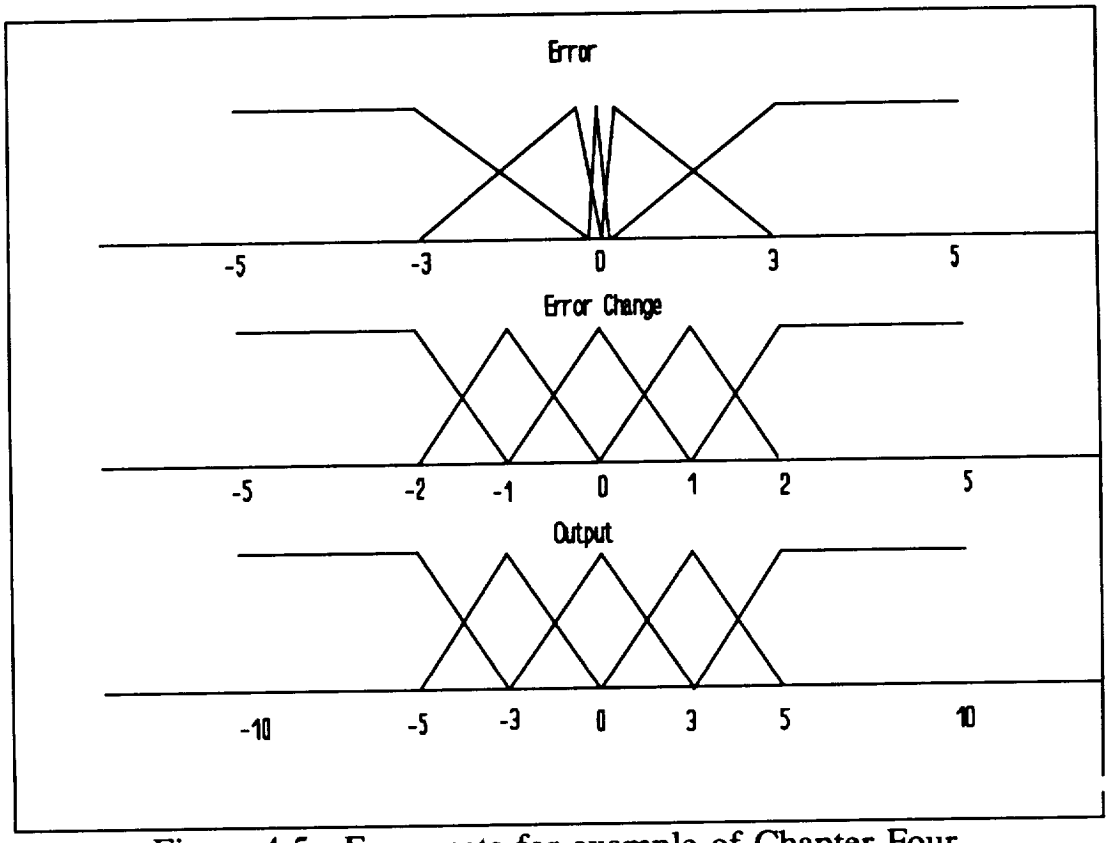

Figure 4.5. Fuzzy sets for example of Chapter Four.

CHAPTER FIVE

ACTUATOR DESIGN FOR THE FREE FLIGHT ROTORCRAFT RESEARCH VEHICLE

5.1 Introduction

The National Aeronautics and Space Administration (NASA) is currently developing a tele-operated unmanned rotorcraft at Langley Research Center. The purpose of the rotorcraft is to allow testing of dynamic stability, maneuverability, and agility in a wind tunnel environment. Before, these tests could not be conducted due to the limitations that are both technical and economical (Phelps and Walker 1992).

Part of our research objectives were to investigate the use of fuzzy logic controllers to control the linear actuators and the throttle governor on the helicopter. The linear actuators are used to control the direction of the helicopter. These directions are left-right movement, forward-backward movement, up-down movement, and the nose direction of the helicopter. The rotor control mechanism consists of a swash plate, connecting links, and

blades which rotate in their sockets (Gessow and Myers, 1985). The swash plate consists of a central non rotating disk and an outer disk which rotates with the rotor. The central and the outer disks are connected by a ball bearing. The inner swash plate is universally mounted and connected to a linkage which allows it to move up and down and tilt in any direction. Blades are connected to the swash plate by links so that the pitch of the blades is determined by the plane of the swashplate. The three linear actuators are mounted on the inner non-rotating swash plate 120 degrees apart. Therefore, tilting the inner swash plate in any direction will force the outer swashplate to tilt in the same direction and the rotor blade to be in such configuration as to make the rotorcraft move in that direction; that is, if the plate is tilted to the right, then the swashplate will follow and the helicopter will move to the right.

The task before us is to design a fuzzy controller for one of the three actuators and test it extensively. Once one of the fuzzy controllers has been tested and proved to be good, the control algorithm can be duplicated for the rest of the actuators in a similar manner and the final control task can be solved geometrically. Another approach is to treat it as a multivariable control problem and therefore design a universal controller that controls all

three actuators.

The underlying reason for having a fuzzy controller instead of another type of digital controller is the ease of modification of the controller parameters. This will allow modifications of the physical characteristics of the swashplates without having to have an accurate model to perform these modifications as one would with a conventional digital controller.

5.2 System Description and Specifications

As discussed in Chapter Three, the first step to design a fuzzy controller is to acquire plant information. In addition, it is essential that we understand how the controller behaves under different conditions; these conditions being the maximum and minimum inputs to the controller and the maximum and minimum outputs of the controller. To arrive at these results, one needs to know the physical as well as the mechanical and electrical aspects and characteristics of the plant.

5.2.1 Physical Characteristics

The electric linear actuators that are to be used on the FFRRV are 4" to 6" inches in length with a rectangular cross section of 1.75" by 3.25" and

weigh no more than 40 oz.

5.2.2 Mechanical Characteristics

The actuator has a minimum linear displacement of 1.5" with a maximum speed of 10" per second. It has an output force of 50 lbs when moving at 10"/sec. Its static or stall force is 75 lbs.

5.2.3 Electrical Characteristics

The actuator is driven by an internal power amplifier. This amplifier is an H-bridge circuit that takes inputs that are TTL and CMOS compatible. Its output is the appropriate current to drive the motor in a chosen direction. The signal to the H-bridge is a single variable duty cycle signal in which is encoded both direction and amplitude information. A 50% duty cycle represents zero drive, since the net value of the voltage integrated over one period is zero. The voltage supply to the H-bridge is 28 volts.

Feedback from the actuator to the controller is accomplished via a Linear Variable Differential Transformer (LVDT) that is mounted on it. The LVDT measures the actuator's absolute position. Mounted on the outside of the actuator are two limiting switches that are located at the bottom and top

of the actuator shaft. These switches are located in those positions in order to sense when the actuator has reached the two extreme positions. When a situation is reached, such as the actuator shaft has extended or contracted to one of those points, the sensors will output a high TTL signal that can be used in conjunction with another logic circuit to prevent the two extreme situations from occurring. The logic circuit and the sensors will only interfere with the operation of the actuator when one of the two extremes occur.

5.3 Design Procedure

The first step is to determine ranges of operation for the signals in the closed-loop system. These ranges are used to form the universes of discourse of the linguistic variables. In the actual implementation, the output of the plant is the actual absolute position of the controller measured to the nearest one thousandths of an inch. The actuator position as given by the range of operation of the LVDT is between 0 and 1467, where 0 represent a position of 0" and 1467 represents a position of 1.467". This is obtained from converting the analog signal of the LVDT via an ac analog to digital converter. Note that, the command input is also between 0 and 1467. This

implies that the error is between -1467 and +1467. As a first guess, the rate of change of error is estimated to be between -2000 and +2000. The control input to the plant, i.e, the output of the controller, is given to be between 0 and +1024, 0 corresponds to 0% duty cycle at the H-bridge circuit which means that the motor will be driven counter-clockwise at full speed (actuator shaft is contracting). On the other hand, 1024 implies a signal of 100% duty cycle that drives the motor in the other direction (actuator shaft extending).

In this problem the linguistic variables are the error (e), rate of change of error (de), and the control input (u). Their universes of discourse have been selected to be X_e = [-1467, 1467], X_{de} = [-2000, 2000], and X_u = [0, 1024], respectively.

The second step is to determine the term sets of each linguistic variable. We assigned to each term set five linguistic values labeled as

PB: Positive Big,

PS: Positive Small,

Z: Zero,

NS: Negative Small,

and

NB: Negative Big.

The membership functions are triangular and are equally spaced with equal overlapping.

The third step is to determine a rule base. Some of the heuristics used to develop it are given next. The specification of how the controller should react to actuator position as measured by the LVDT can be described as follows: If the command input to the closed loop system is 1 inch, then it is desired that the plant follow that command input and have a position of 1 inch. Furthermore, if the plant is initially in a different position, then the error measured in the closed loop system will be either in the negative or positive direction where an error in the negative direction implies that the output is at a position greater than that of the command input. On the other hand, a positive error indicates that the output is less than that of the command input.

Depending on what sign the error carries, we want the controller output to have a value compatible with the error sign. This implies that if the error is negative then we need a negative output from the controller to bring it down to the desired position, therefore reducing the error. In particular, we need to cause the actuator shaft to contract by presenting 0% duty cycle. More specifically, taking the rate of change of error into account

we can describe the controller action to the output of the plant more accurately as follows.

- If the error is big or small and it is moving still further from the set point, then we want a big or small input to bring the actuator to the desired position.
- If the error is small and it is being reduced then we need no input (50% duty cycle) to the plant. This takes into account the momentum that will bring the actuator into the desired position with little or no overshoot.
- If the error is zero and it is not changing then do nothing. A complete listing of the controller files, fuzzy sets,and rule base are given at the end of the chapter.

5.4 Test Results

The controller was extensively tested under various loads and various load conditions. The initial test of the controller without tuning the fuzzy sets showed that the controller was performing with some steady state error and was not as fast as desired. From the evaluation of the data, and in particular from evaluating the control output we found that the maximum

output we were getting from the controller is 892 instead of the 1024 we designed for. In addition, the fuzzy sets PB and NB were not extended enough to cover errors that we felt should be covered under the PB and NB sets. Figures 5.1 show the response of the closed loop system.

Extending the range of the PB and NB error sets makes the controller react faster. To make the maximum output of the controller 1024, the range of the controller was extended such that the center value of the PB set of the output was 1024. After data analysis, the controller was tuned to achieve a maximum output of 1024 and to make the error sets of PB and NB have wider ranges. This improved the response of the controller dramatically. Specifically, the rise time, settling time, and steady state error were noticeably much better. Under a no load condition, the controller responded with 0.075 seconds of rise time, 0.0 steady-state error, and 9.6% overshoot. Moreover, the control input also achieved its maximum value of 512.

Figures 5.2 shows the response of the closed loop system after tuning with no load. Next we tested the controller with the plant loaded with weights. These tests required the actuator to push and pull weights that were attached to a box. The controller pushed and pulled the box and the response was then evaluated to see if it accomplished the task without any

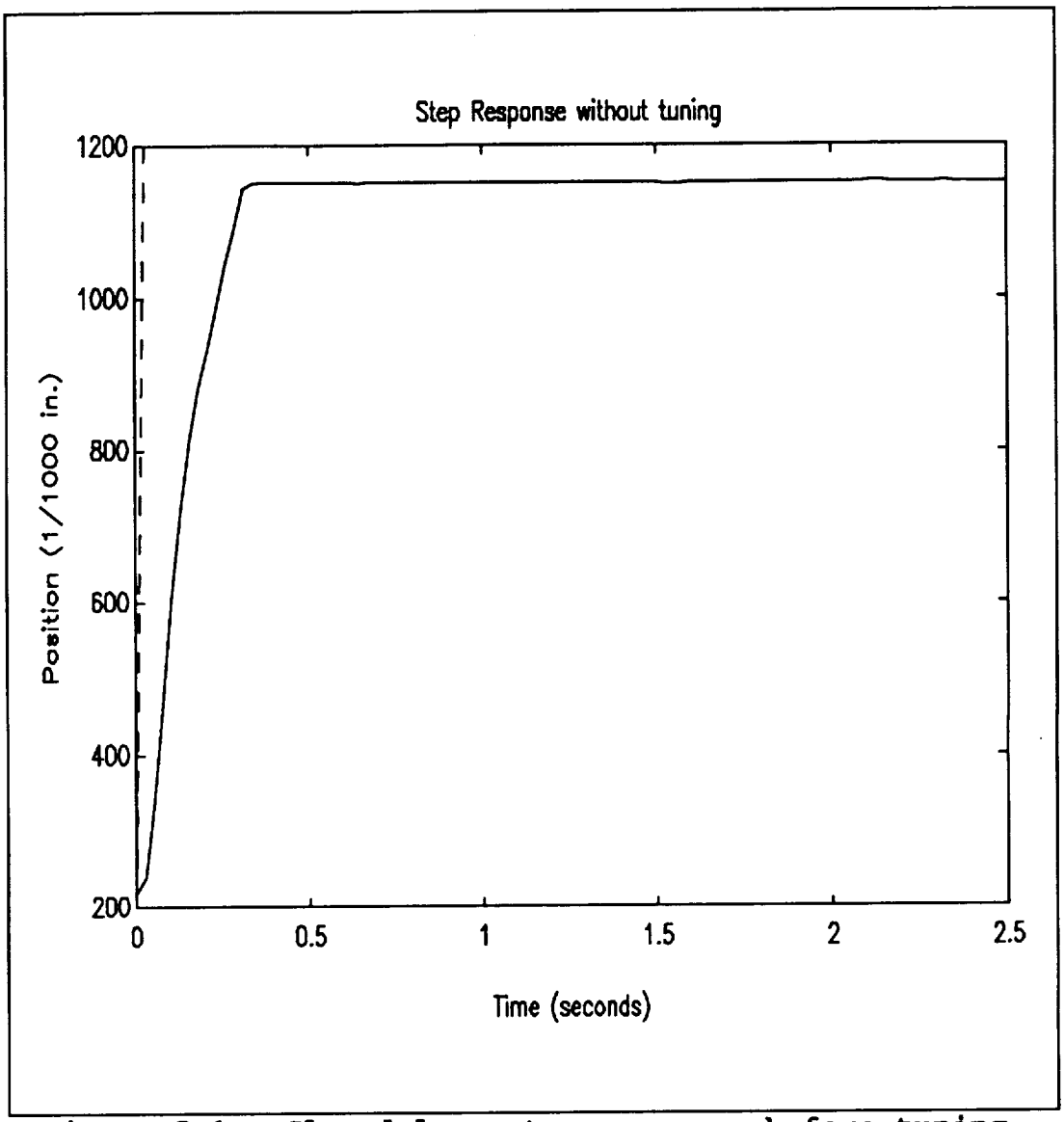

Figure 5.1. Closed loop step response before tuning.

significant change in the time response. An initial weight of 20 lbs was experienced by the plant. The response was not changed significantly. Figures 5.3 shows the closed loop response to horizontal loading and Figure 5.4 shows the response of the closed loop system under vertical loading.

Different controllers were designed and tested to observe the effect of tightening the zero set and the extension of the big sets. The step response of the three tuned control systems are given in Figures 5.5 through 5.6. In Figure 5.5, the step response of the closed loop system under light load is given. Figure 5.6 is the response of the closed loop system when the plant is heavily loaded to the same three controllers.

Notice that the closed-loop system under heavy load shows bigger steady-state errors for the first two designs. We were able to reduce the steady-state error of the third design by allowing larger steady-state error under light loads. An interpretation of the variation in steady-state errors is not available at this time. All modifications of the initial controllers are given in Appendix 5A at the end of this chapter.

5.5 Conclusions

A fuzzy controller was designed and tested for the FFRRV. This section illustrated the simplicity of designing and tuning a fuzzy logic controller for a real plant whose mathematical model was not known. The results section showed that the fuzzy logic controlled system was able to meet several of the design specifications. The fact that not all the specifications could be met was due to plant limitations; the plant did not meet specifications.

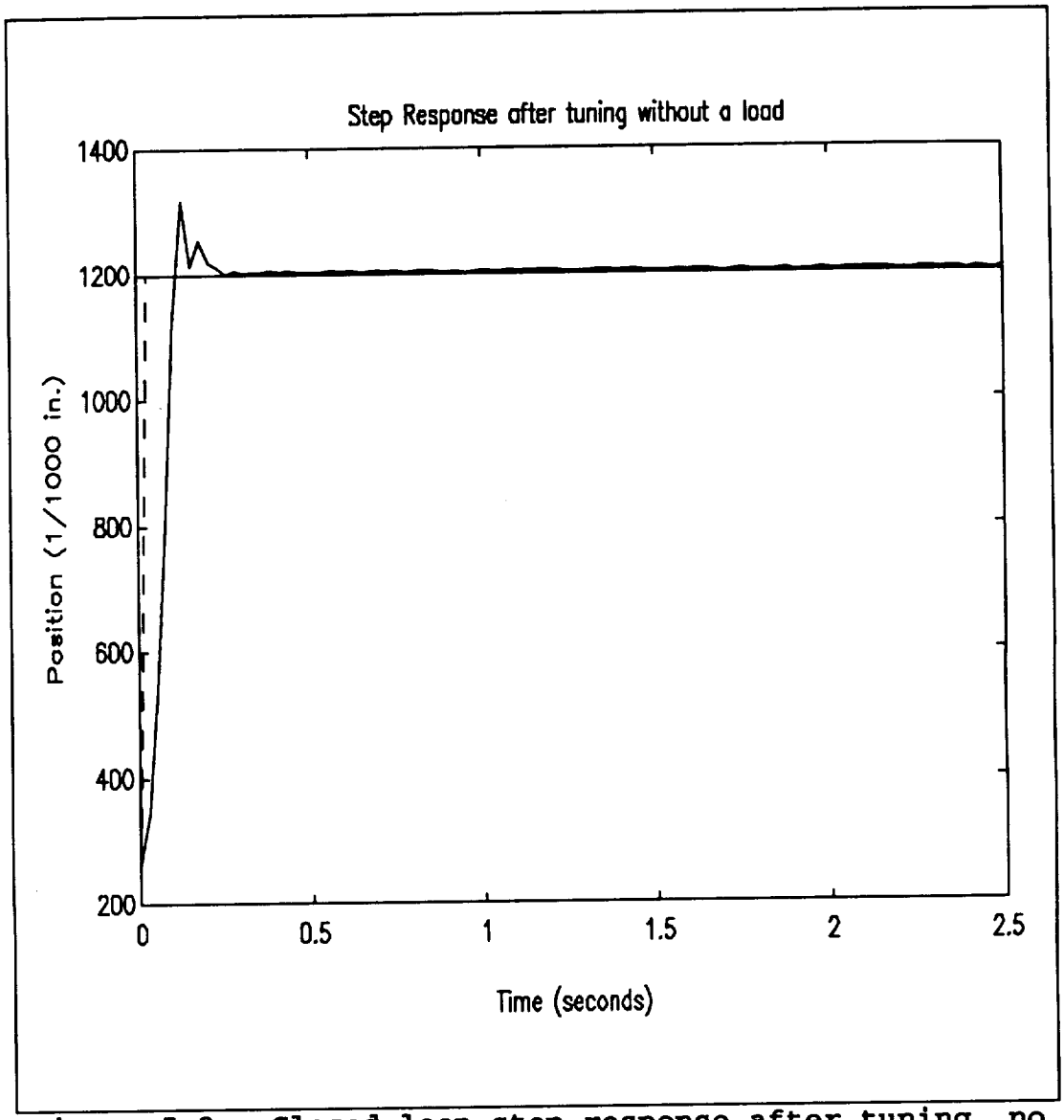

Figure 5.2. Closed loop step response after tuning, no load.

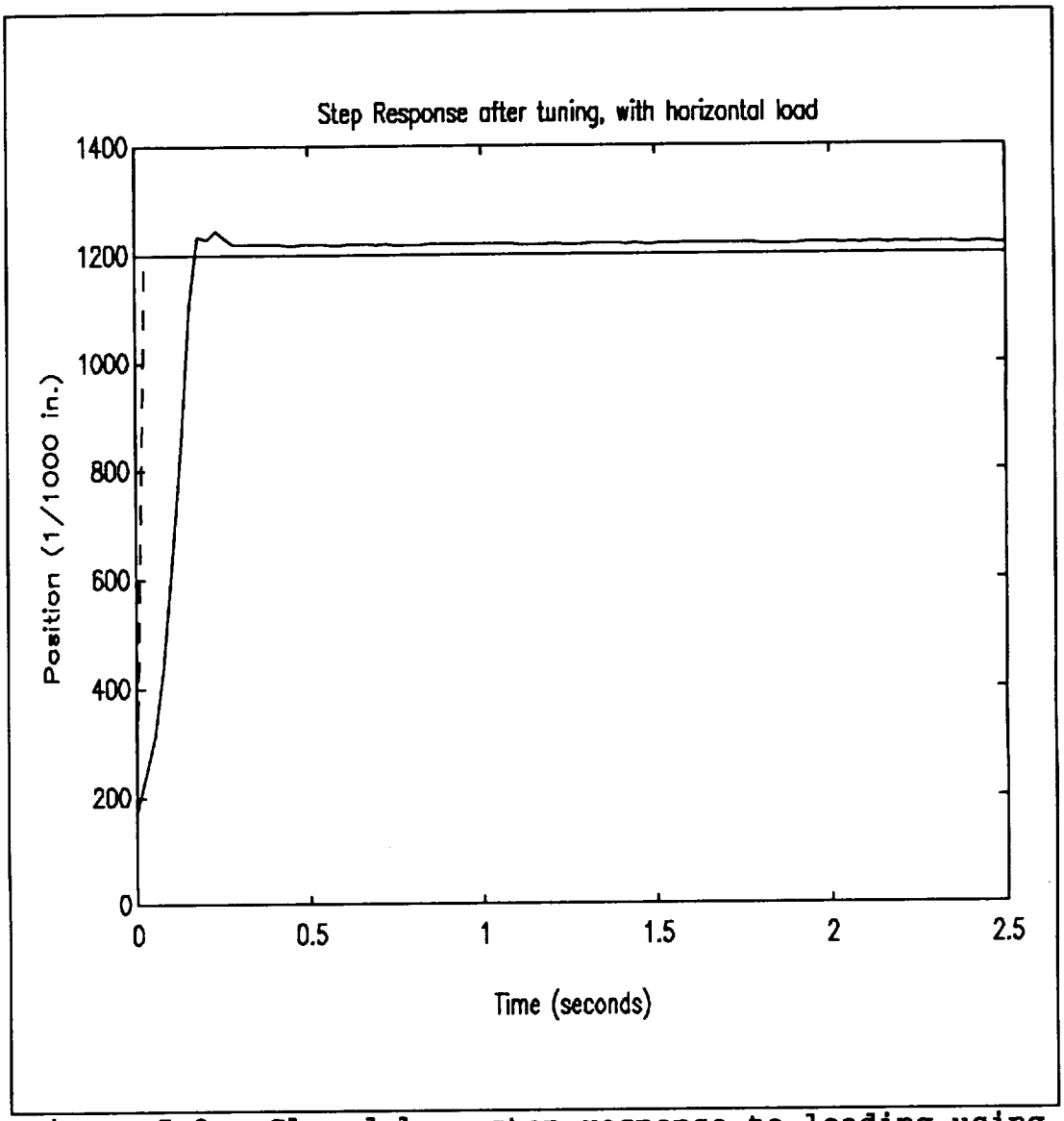

Figure 5.3. Closed loop step response to loading using horizontal test bed.

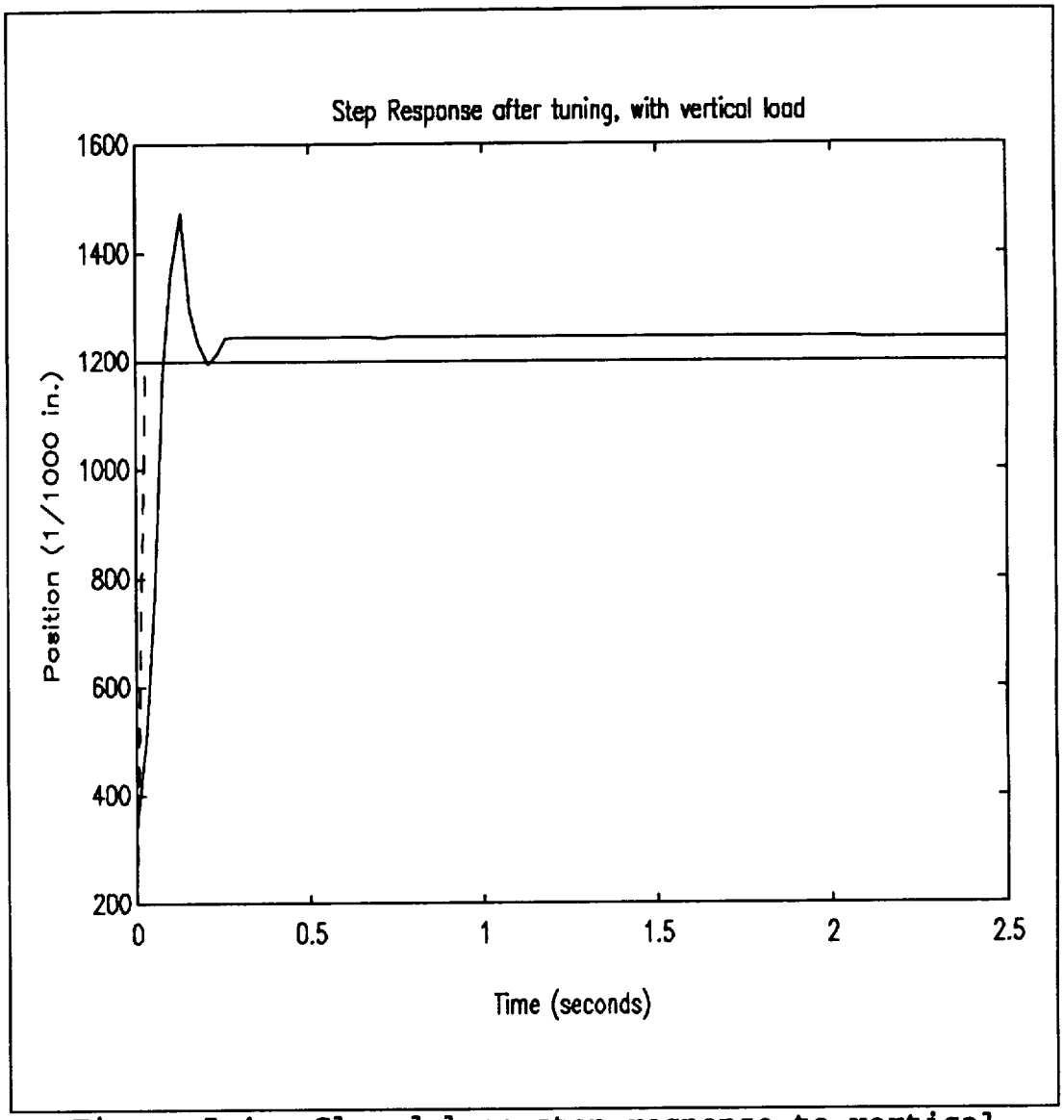

Figure 5.4. Closed loop step response to vertical loading.

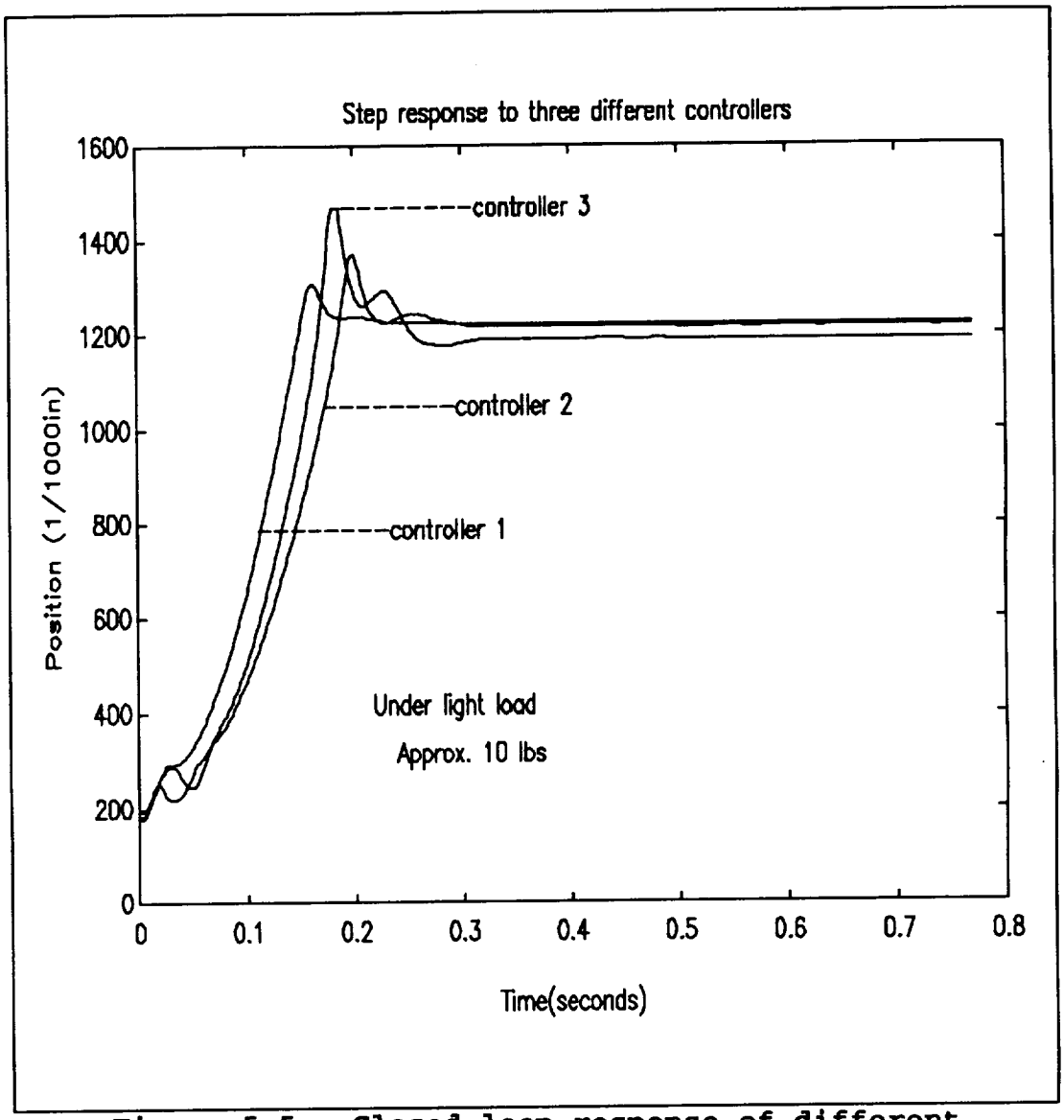

Figure 5.5. Closed loop response of different controllers.

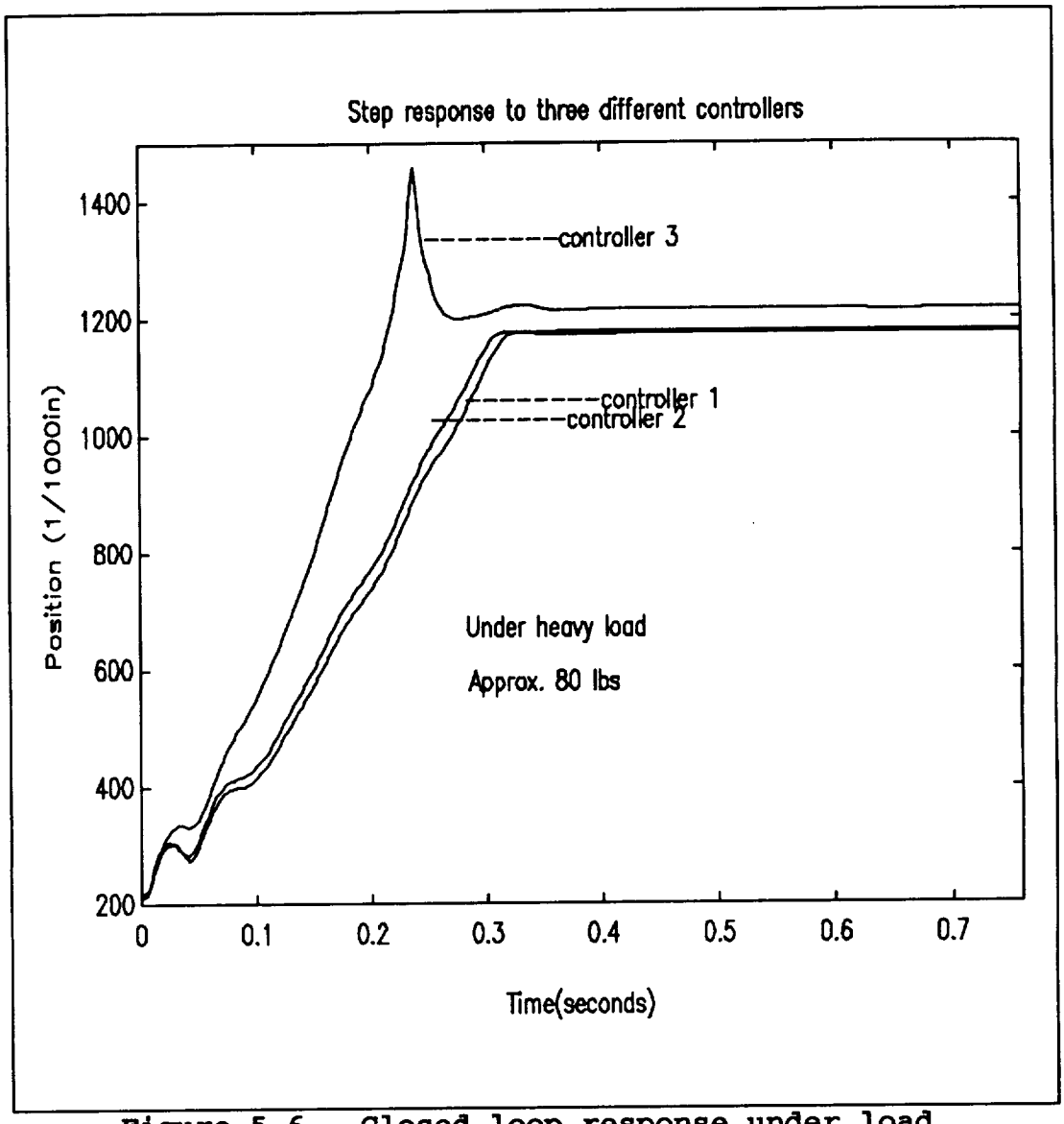

Figure 5.6. Closed loop response under load.

APPENDIX 5A

The Rule Base and Fuzzy Sets For The Linear Actuator

IF (Error IS PB) AND (derror IS ANY) THEN output=Fast_Reverse

IF (Error IS PS) AND (derror IS PB) THEN output= Fast_Reverse

IF (Error IS PS) AND (derror IS PS) THEN output=Reverse

IF (Error IS PS) AND (derror IS Z) THEN output=Reverse

IF (Error IS PS) AND (derror IS NS) THEN output=IDLE

IF (Error IS PS) AND (derror IS NB) THEN output= Forward

IF (Error IS Z) AND (derror IS PB) THEN output=Reverse

IF (Error IS Z) AND (derror IS PS) THEN output=Reverse

IF (Error IS Z) AND (derror IS Z) THEN output=IDLE

IF (Error IS Z) AND (derror IS NS) THEN output=Forward

IF (Error IS Z) AND (derror IS NB) THEN output=Fast_Forward

IF (Error IS NS) AND (derror IS PB) THEN output=Fast_Forward

IF (Error IS NS) AND (derror IS PS) THEN output=Forward

IF (Error IS NS) AND (derror IS Z) THEN output=Forward

IF (Error IS NS) AND (derror IS NS) THEN output=Forward

IF (Error IS NS) AND (derror IS NB) THEN output=Fast_Forward

IF (Error IS NB) AND (derror IS ANY) THEN output=Fast_Forward

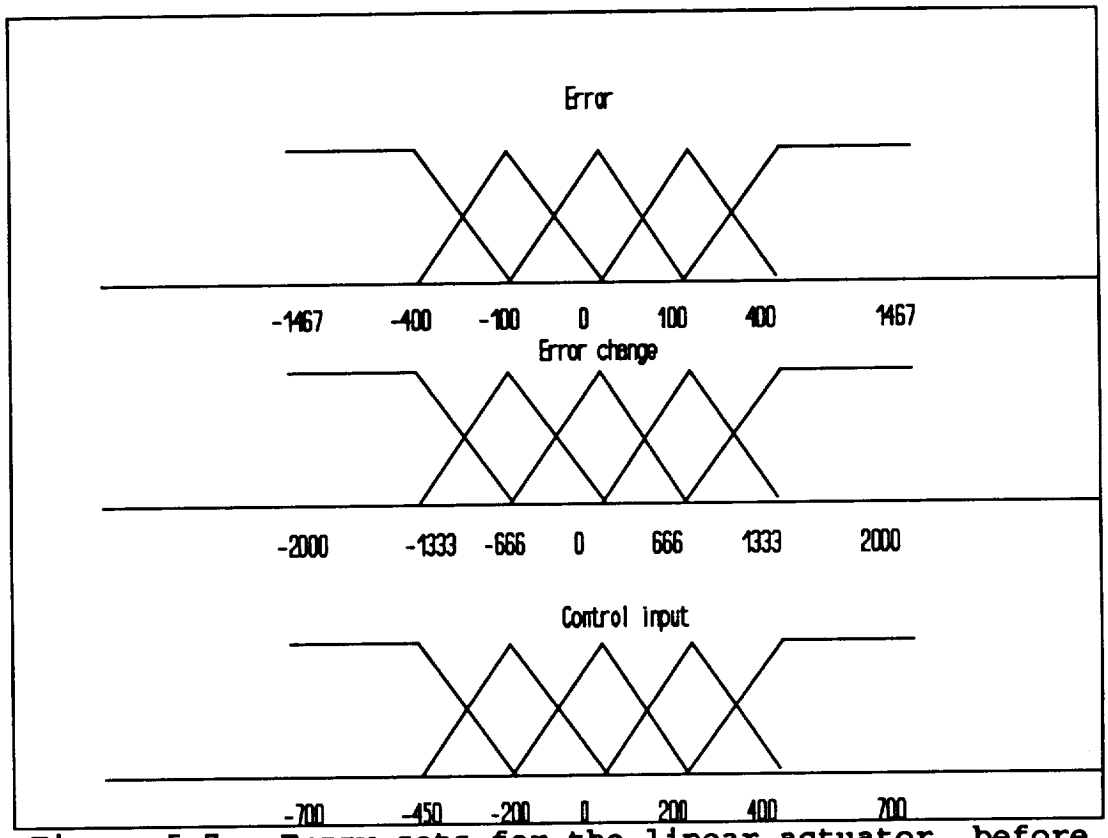

Figure 5.7. Fuzzy sets for the linear actuator, before tuning.

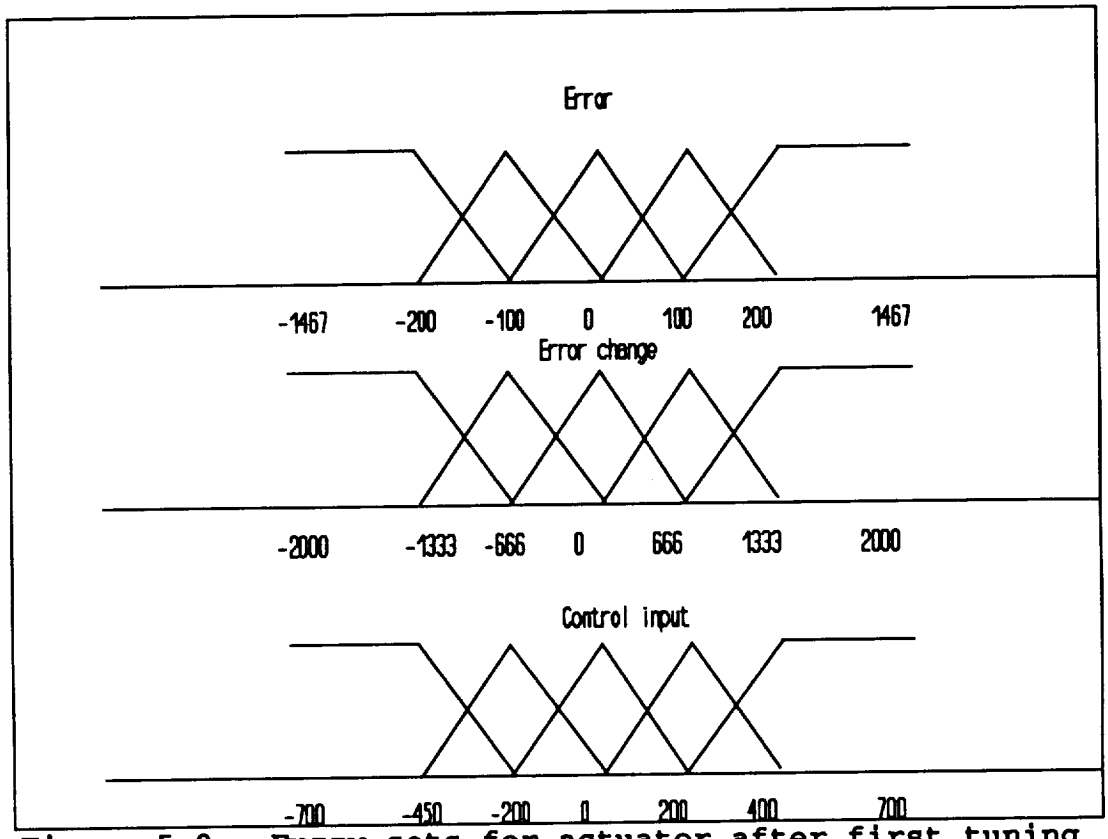

Figure 5.8. Fuzzy sets for actuator after first tuning.

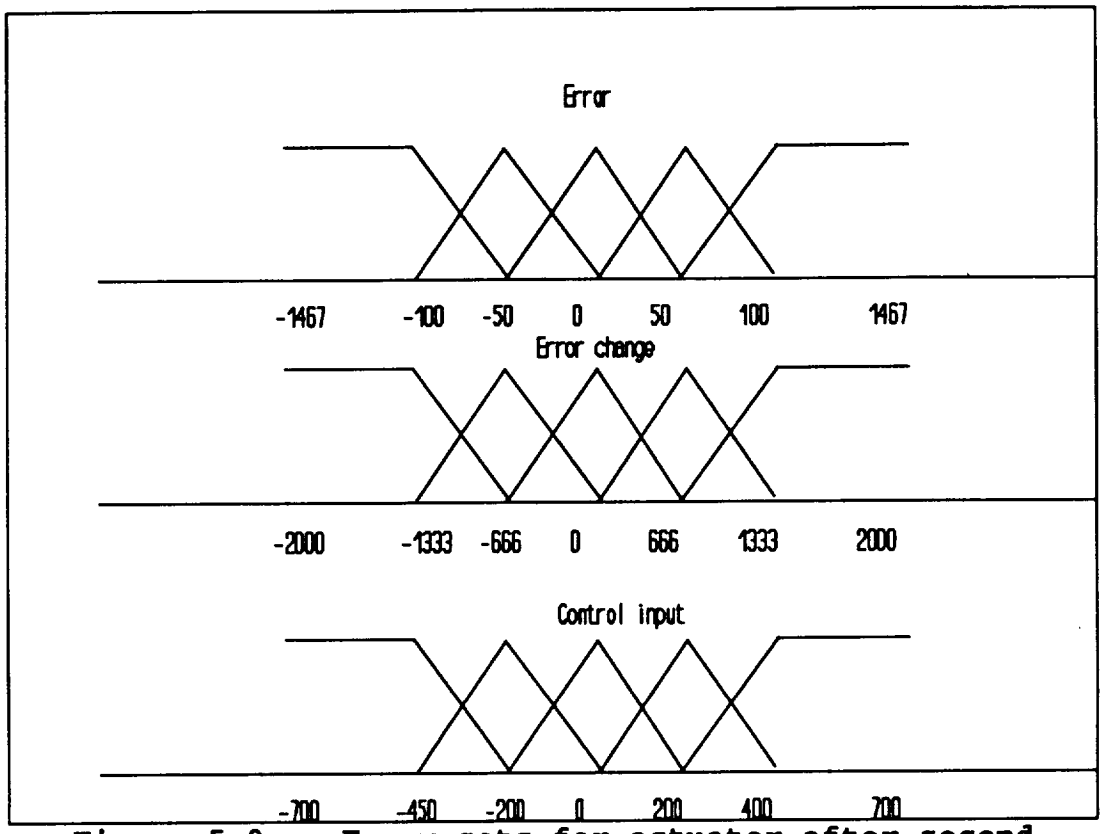

Figure 5.9. Fuzzy sets for actuator after second tuning.

CHAPTER SIX

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

6.1 Conclusions

In this thesis we first present a tutorial on fuzzy logic and fuzzy sets as they relate to control system design. The concept of a fuzzy set and operations on a fuzzy set are presented in comparison to conventional sets to help the reader understand the concept more easily.

Second, we introduce a basic design procedure. New guidelines to tune the fuzzy logic controllers are included in the procedure. This procedure was derived from our experience in designing fuzzy logic controllers.

Third, we present four design examples to illustrate the application of the design procedure and to highlight some of the characteristics of fuzzy logic control systems. The design approach is simple and was applied to

four examples. Moreover, the examples illustrate the success of the tuning guidelines in modifying both transient and steady state responses.

One of the characteristics of a fuzzy logic control system is that it satisfies the superposition property over the range of operations. The scaling property is demonstrated by varying the set points. The additive property is demonstrated by finding the responses to two separate and simultaneous unit-step disturbances. This was unexpected.

A second characteristic investigated was the robustness to plant parameter variations. It is shown that some level of robustness is attained. The fact that the responses are more affected by variations in one direction versus the other one was unexpected. A third characteristic observed was that the ramp response is instantaneous; that is no transient response was observed. A fourth characteristic observed was that a fuzzy logic control system can cancel the effect of right-half plane zeros, unlike a linear controller. All these characteristics were unexpected. They point to the need to develop analytical tools to study fuzzy logic control systems.

Fourth, an overview of stability analysis approaches were presented for completeness. The analysis methods presented are not yet useful to practical

fuzzy logic control design. This overview may be used as a base for further research.

Finally, we solved a real world problem using the methodology introduced in this thesis. Fuzzy logic controllers were designed and implemented to control one of the linear actuators to be used in a research helicopter. This chapter illustrates the ideal environment in which fuzzy logic controllers may be used; that is, plants that do not have a mathematical representation.

6.2 Suggestions For Further Research

This thesis serves as a foundation for understanding fuzzy logic control systems. As presented in Section 6.1, our results highlight the need for further research. Some of the suggestions are given next.

It is important to develop analysis tools with similar capabilities as those available for linear and nonlinear controllers.

The stability analysis of fuzzy control systems is another area that needs to be further researched. Presently, there are few methods of stability analysis. These methods include assumptions that are impractical, making it impossible to study the stability of practical fuzzy control systems. These

analysis tools should explain the characteristics of fuzzy control systems as explained in Section 6.1. For example, analysis tools to study the sensitivity to plant parameter variations.

These analysis tools can then be translated into design tools. For example, for robust control should an integrator be added in series with the plant or should integral action be included in the fuzzy logic controller as in Pedrycz (1989)?

Another important area of further research is to determine methods to make fuzzy logic control systems adaptable. The adaptivity of fuzzy controllers is extremely important for several reasons. One reason is that changes in the dynamics of the plant could be automatically accounted for by an adaptive algorithm incorporated in the existing fuzzy control system. The adaptive algorithm would also be used to achieve predefined goals in the system. Such goals are the rise time, steady-state error, overshoot, and settling time.

BIBLIOGRAPHY

Bernard, J., (1988), "Use of a rule-based system for process control," International Conference on Industrial Electronics, Control, and Instrumentation, 3-12.

Chand, S. and Chiu, S., (1991), "Robustness analysis of fuzzy control systems with application to aircraft roll control," AIAA Guidance, Navigation, and Control.

Chand, S. and Hansen, S., (1989), "Energy based stability analysis of a fuzzy controller design for a flexible aircraft wing," Proceeding of the IEEE International Conference on Decision and Control, 705-709.

Chiu, S. and Chand, S., (1991), "Fuzzy controller design and stability analysis for an aircraft model," American Control Conference, 89-95.

Dubois, D. and Prade, H., (1984), "Fuzzy logic and the generalized modus ponens revisited," Cybernetic Systems, Volume 15, 3-4.

Franklin, G., Powell, D., J., and Workman, M., (1990), Digital Control of Dynamic Systems, Addison Wesley Publishing Company, Reading, MA, 559-561.

Gessow, A. and Myers, G., (1985), "Aerodynamics of the Helicopter," Library of Congress catalog card number:67-26126, 30-36.

Heisener, S., G., (1992), "Fuzzy control applied to an aircraft engine," Proceedings of International Federation of Automatic Control Nonlinear Control System Design Symposium, 94-99

Hill, G., Horstkotte, E., and Teichrow, J., (1991), TILShell Users Manual, Togai Infra Logic, Irvine, CA.

Jacob, M. and J. (1988), Industrial control electronics, applications and design, Perentice Hall, 90-98.

Jang, R. and (1992), "Fuzzy controller design without domain experts," Proceedings of the IEEE International Conference on Fuzzy Systems, 289-296.

Kalman, E. and R., Bertman, E., J., (1960), "Control systems analysis and design via the second method of Lyapunov," American Society of Mechanical Engineering, 58-87

Kandel, A. and Lee, S., C. (1979), Fuzzy Switching and Automata: Theory and Application, New York, NY, Grane, Russak and Company, Inc.

Kowamoto, S., Tada, K., Ishigame, A., and Taniguchi, T., (1992), "An approach to stability analysis of second order fuzzy systems," Proceedings of the IEEE International Conference on Fuzzy Systems, 1427-1434.

Langari, G. and Tomizuka, M., (1990), "Stability of fuzzy linguistic control systems," Proceedings of the IEEE Conference on Decision and Control, 2185-2190.

Langari, G. and Tomizuka, M. (1990), "Fuzzy linguistic model based control," Proceedings of the IEEE International Symposium on Intelligent Control.

Lee, C., C., (1990), "Fuzzy logic in control systems: Fuzzy logic controller," IEEE Transactions on Systems, Man, and Cybernetics, 404-435.

Mamdani, E., H., (1974), "Application of fuzzy algorithms for control of simple dynamic plant," Proceedings of the IEEE, 121 (12), 1585-1588.

Mizumoto, M., (1992), "Realization of PID controllers by fuzzy control methods," Proceedings of the IEEE International Conference on Fuzzy Systems, 709-715.

Li, Y., F. and Lau, C., C., (1989), "Development of fuzzy algorithms for servo systems," IEEE Control Systems Magazine, 65-72.

Pedrycz, W., (1989), Fuzzy Control and Fuzzy Systems, New York, John Wiley and Sons Inc.

Schneider, D., Wang, P., and Togai, M., (1992), "Design of a fuzzy logic controller for a target tracking system," Proceedings of the IEEE International Conference on Fuzzy Systems, 1131-1138.

Sugeno, M., Murofushi, T., Nishino, J., and Miwa, H., (1991), "Helicopter flight control based on fuzzy logic," International Fuzzy Engineering Symposium.

Tanaka, K. and Sugeno, M., (1990), "Stability analysis of fuzzy control systems using Lyapuvov's direct method," North American Fuzzy International Proceedings, 133-136.

Tanaka, K. and Sugeno, M., (1985), "Fuzzy identification of systems and its application to modeling and controls," IEEE Transactions on Systems, Man, and Cybernetics, 116-132.

Tseng, C., H., Hwang, V., and Lui, S., L., (1992), "Fuzzy servo controller: The hierarchical approach," Proceedings of the IEEE International Conference on Fuzzy Systems, 623-631.

Walker, G., W. and Phelps, A., E., (1992), "A teleoperated unmanned rotorcraft flight test technique," National Telesystems Conference.

Wang, B., H. and Vachtsevanos, G., (1992), "Learning fuzzy control: An indirect approach," Proceedings of the IEEE International Conference on Fuzzy Systems, 297-304.

Xiangchu, T. and Chengyuan, T., (1988), "A new approach to fuzzy control," Decision and Control, 307-315.

Ying, H., Siler, W., and Buckley, J., (1990), "Fuzzy control theory: A nonlinear case," Automatica, 513-520.

Zadeh, L., A., (1965), "Fuzzy Sets," Information and Control, 338-353.

Zheng, L., (1992), "A practical guide to tune PI like fuzzy controllers," Proceedings of the IEEE International Conference on Fuzzy Systems, 633-640.

APPENDIX A

TILShell and Fuzzy Programming Language (FPL)

TILShell is a software development tool which provides a way to describe and design fuzzy expert systems and then compile this description into the output code necessary to implement the system. It is designed to run with Windows.

The major components of the TILSell are the object editors which consist of

- The Project Editor
- The Rule Editor
- The Membership Function Editor
- The Package Editor
- The Source/Fragment Editor
- The Var Editor

Once you log on to TILShell, you would see all the above editors listed in icons on the left hand of the screen. To start with the design of fuzzy expert system, one would need to define the input and output variables. This is

accomplished by clicking on the variable icon. In the variable icon, you would be able to define the universe of discourse for each variable and the membership function sets. With regard to the membership functions TILShell allows the user to specify the number of membership functions that is needed. The default for the membership functions is three. Moreover, the membership functions are equally space, equally overlapped triangles. However, the spacing and the overlap can be easily modified to give best results. Once the input and output variables are defined, the rule icon can be invoked and the rules specified accordingly. The input variables are connected to the rule base through the connect icon and in turn the rule base is connected to the output variable also through the connect icon.

The user has the ability to define his own code in the source/fragment editor. The source editor allows the user to write the code only in C. Moreover, it allows the user to manipulate variables and do tasks that are not included in the initial package. For example, the source editor allows the user to write C code specific to the simulation of the controller. The TILShell file can also be generated using a regular ASCII editor to modify the variables, rule base, and the universe of discourse. To best illustrate the use of TILShell consider the examples given in the previous chapters.