To the Graduate Council:

I am submitting herewith a master thesis written by Naser Hdieb entitled "FUZZY ADAPTIVE VSC SYSTEMS." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

J. C. Hung, Major Professor

We have read this thesis
and recommend its acceptance:

Don Hinton

Igor Alexeff

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

FUZZY ADAPTIVE VSC SYSTEMS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Naser Hdieb

May 1993

DEDICATION

This thesis is dedicated to my father, Ahmad Abdel-Fattah Hdieb, who was my first educator in life, may God rest his soul.

ACKNOWLEDGMENTS

The writer wishes to express his appreciation to his committee members: Dr. I. Alexeff and Dr. D. B. Hinton. A special thanks and a sincere appreciation is extended to my major professor Dr. J. C. Hung for his guidance and inspiration to learn and to think. It has been a pleasure and an honor working for him. A special thanks to my father for his support and inspiration, may God rest his soul. More thanks to my mother, brothers and sisters who provided me with support, love, concern and many words of encouragement. Finally, I would like to thank my friends and colleagues for their encouragement all the way.

ABSTRACT

To design robust control systems, Variable Structure Control (VSC) is one of the modern control methods that could be used. This method is implemented using a discontinuous type control which usually includes linear feedback terms and signum terms. The signum terms are used to neutralize the effects of general perturbations in the system. It is well known in the theory of VSC that large magnitude signum terms increase the amplitude of chattering which exists in VSC systems with time delays in their switching mechanisim. This study proposes a new approach to achieve low chattering in a VSC system.

In this study, the suggested approach is to implement fuzzy adaptive algorthims to the reaching and sliding modes of a VSC system to reduce chattering. The approach incorporates an adaptive adjustment mechanisim and a fuzzy decision mechanisim into the VSC system. Two fuzzy adaptive algorithms are used, one provides a VSC on demand feature and the other reduces the chattering of the state variables caused by switching delay. The result is a low chattering robust VSC system without compromising the robustness property of the system.

TABLE OF CONTENTS

CHAPTER			PAGE
1.	**Introduction**		**1**
	1.1	Fundamentals of VSC	1
	1.2	Design of Sliding Modes	3
		1.2.1 Mathematical Representations for the Dynamics of the Sliding Modes	3
		1.2.2 Design of Sliding Surfaces	5
	1.3	Design of the Reaching Mode Using the Reaching Law Method	7
		1.3.1 Reaching Condition and the Reaching Mode	7
		1.3.2 Design of A VSC Law Using the Reaching Law Method	8
	1.4	Problems of VSC Systems	11
	1.5	Objective of the Study	14
2.	**Adaptive VSC SYSTEM**		**16**
	2.1	General System Configuration	16
		2.1.1 The Plant	16
		2.1.2 VSC	18
		2.1.3 Fuzzy Logic	18
		2.1.4 Perturbation Observer	19
	2.2	Fuzzy Adaptive Sliding Mode	20
	2.3	Fuzzy Adaptive Reaching Mode	24

CHAPTER PAGE

2.4 Fuzzy Adaptive VSC in the Reaching and Sliding Modes 28

3. Experimentation by Simulation 30

3.1 A plant for A Generator Voltage Regulation System 30

3.2 Control by Usual VSC Law 33

3.3 Perturbation Observer 38

3.4 Fuzzy Adaptive VSC System 38

3.4.1 Fuzzy Adaptive VSC in Sliding Mode only 39

3.4.2 Fuzzy Adaptive VSC in Reaching Mode only 50

3.4.3 Fuzzy Adaptive VSC in Both Sliding Mode and Reaching Mode 58

4. SUMMARY 62

BIBLIOGRAPHY 64

APPENDICES 66

A. Listing of Program 1 67

B. Listing of Program 2 85

C. Listing of Program 3 90

VITA 96

LIST OF FIGURES

FIGURE		PAGE
1.1	Chattering caused by a non-ideal reaching	12
1.2	A plot shows a steady state error caused by the stablization of the limit cycle at an equilibrium point where its mean velocity is zero	13
2.1	An overall block disgram of a fuzzy adaptive VSC system	17
2.2	A perturbation observer	20
2.3	A membership function for the normalized variables $\hat{d}$, $\dot{\hat{d}}$, and q	23
2.4	Flowchart of fuzzy logic program which implements fuzzy adaptive VSC in the sliding mode	25
2.5	A membership function for the normalized variables s, $\dot{s}$, and q	27
2.6	Flowchart of fuzzy logic program which implements a fuzzy adaptive VSC in the reaching mode	29
3.1	A Generator Voltage Regulation System	31
3.2	Simulation block diagram for a usual VSC system	34
3.3	Response for VSC, case 1 ($q = 0$ and no perturbation)	35
3.4	Response for VSC, case 2 ($q = 0$, with perturbation)	36
3.5	Response for VSC, case 3 ($q = 12$, with perturbation)	37
3.6	(a) $\hat{d}$, (b) $\dot{\hat{d}}$, and (c) normalized fuzzy region	40
3.7	Example demonstrates a two input fuzzy logic program procedure	41
3.8	Simulation block diagram for fuzzy adaptive VSC in sliding mode only	43

FIGURE **PAGE**

3.9 Fuzzy adaptive VSC in sliding mode only, case 1 44

3.10 Example demonstrates a single input fuzzy logic program procedure 45

3.11 Fuzzy adaptive VSC in the sliding mode only, case 2 47

3.12 Fuzzy adaptive VSC in the sliding mode only, case 3 48

3.13 Phase plane trajectory of (a) case 2, (b) case 3 49

3.14 Response for VSC, case 1 ($q = 1$) 51

3.15 Response for VSC, case 2 ($q = 50$) 52

3.16 Phase plane trajectory (a) case 1, (b) case 2 53

3.17 (a) s, (b) $\dot{s}$, and (c) normalized fuzzy region 54

3.18 Simulation block diagram for a fuzzy adaptive VSC in reaching mode only 55

3.19 Fuzzy adaptive VSC in reaching mode only, case 3 56

3.20 Phase plane trajectory for case 3 57

3.21 Simulation block diagram for a fuzzy adaptive VSC in both sliding mode and reaching mode 59

3.22 A fuzzy adaptive VSC in both sliding mode and reaching mode . 60

3.23 Phase plane trajectory of fuzzy adaptive VSC in both sliding mode and reaching mode 61

LIST OF TABLES

TABLE		PAGE
2.1	Fuzzy system rules for two input variables and one output variable	23
2.2	Fuzzy system rules for two input variables and one output variable	27
3.1	Rules for a single input fuzzy system	46

CHAPTER 1

Introduction

1.1 Fundamentals of VSC

Variable Structure Control (VSC) is a control method for achieving robust control systems. A VSC system is characterized by a discontinuous control action which is switched as the system state crosses certain surfaces in the state space [1],[2]. The control is designed to drive and then constrain the state to lie within a neighborhood of some pre-defined surfaces. The motion of the state in that region is known as the sliding mode . In the sliding mode, the dynamic behavior of the system becomes equivalent to an unforced system of lower order, and under certain conditions [7], the closed-loop response becomes insensitive to changes in plant parameters. In other words, the dynamics of the system may become the dynamics of the sliding mode. The process in which the system state reaches the sliding mode is called the reaching mode. Therefore, the design of VSC systems involves two steps. The first step is the design of appropriate switching surfaces for a desired sliding mode dynamics. The second step is the design of a control function for the reaching mode such that a reaching condition is met.

Ideally, the sliding mode is invariant to system perturbations and external disturbances. On the other hand, the reaching mode does not have invariance

properties. However, the dynamics of the reaching mode can be manipulated and controlled using a VSC design method known as the reaching law method [3]. In general, for the reaching mode, the desired response is usually to reach the switching surfaces in finite time and with no overshoot.

In this chapter, a brief review of VSC design is presented for linear MIMO time-invariant systems. The system is modeled by

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + B\mathbf{d}, \tag{1.1}$$

where $\mathbf{x}$ is n-dimension state vector, $\mathbf{u}$ is an m-dimensional input, $\mathbf{d}$ is m-dimensional and represents the total perturbation in the system, and the matrices A and B are of appropriate dimensions. The switching function for a VSC system is assumed to be m-dimensional and linear. For MIMO systems, the switching function is in the following form:

$$\mathbf{s}(\mathbf{x}) = C\mathbf{x}, \tag{1.2}$$

where $\mathbf{s}(\mathbf{x})$ is m-dimensional and C is mxn.

$$\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}) \;\; s_2(\mathbf{x}) \;\; \ldots \;\; s_m(\mathbf{x})], \tag{1.3}$$

and

$$C^T = [\mathbf{c}_1 \;\; \mathbf{c}_2 \;\; \ldots \;\; \mathbf{c}_m]. \tag{1.4}$$

The above represents m linear switching surfaces and can be put into the following form:

$$\mathbf{s}_i(\mathbf{x}) = \mathbf{c}_i^T\mathbf{x} \quad \text{where} \;\; i = 1, 2, ..., m. \tag{1.5}$$

When the state trajectory of a system reaches a switching surface, then the state is in the sliding mode. The condition under which the state moves toward and reach a sliding surface is called the reaching condition [1].

1.2 Design of Sliding Modes

In designing sliding modes, the main objective is to design ones that have certain characteristics to guarantee global asymptotic stability and desired dynamics. The sliding mode characteristics for linear systems can be designed using one of several techniques such as pole placement, quadratic optimization, and eigen-structure [3],[4].

1.2.1 Mathematical Representations for the Dynamics of the Sliding Modes

The mathematical models of VSC systems are non-linear due to the discontinuous control action. The differential equation for such a system has a finite discontinuous right hand side. To represent the dynamics of VSC systems in the sliding mode, these differential equations must have solutions at all points of discontinuity. However, solutions do not exist due to the existance and uniqueness theorm. Hence, several other methods have been suggested to define the differential equations of a VSC system in the sliding mode. One method is to transform the system model into some canonical form.

For a SISO linear time-invariant systems, transform the system equation into

the canonical form

$$\begin{aligned} \dot{x}_1 &= x_2 \\ &\vdots \\ \dot{x}_{n-1} &= x_n \\ \dot{x}_n &= \sum_{i=1}^{n} -a_i x_i + bu. \end{aligned} \tag{1.6}$$

The switching function s is defined to be

$$c_1 x_1 + c_2 x_2 + c_3 x_3 + ... + x_n = 0. \tag{1.7}$$

where $c_1, c_2, c_3, \ ... \ , c_n$ are coefficients that determine the characteristics of the switching function. By solving (1.7) for x_n and then substituting the results in (1.6), it gives

$$\begin{aligned} \dot{x}_1 &= x_2 \\ &\vdots \\ \dot{x}_{n-1} &= -c_1 x_1 - c_2 x_2 - ... - c_{n-1} x_{n-1}. \end{aligned} \tag{1.8}$$

As a result, the coefficients of (1.7) will define the characteristics of the sliding mode if the system is described in the controllable canonic form.

A linear MIMO time-invariant system is presented by

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{s}(\mathbf{x}) &= C\mathbf{x}, \end{aligned} \tag{1.9}$$

which is the same as the equations presented in (1.1) to (1.4). To describe the dynamics of the sliding mode, the above model has to be transformed into a form

that contains s as m of the state variables. The plant is first transformed by $\mathbf{y} = T_1\mathbf{x}$ to the form [4]

$$\begin{aligned} \dot{\mathbf{y}}_1 &= \bar{A}_{11}\mathbf{y}_1 + \bar{A}_{12}\mathbf{y}_2 \\ \dot{\mathbf{y}}_2 &= \bar{A}_{21}\mathbf{y}_1 + \bar{A}_{22}\mathbf{y}_2 + \bar{B}\mathbf{u} \end{aligned} \tag{1.10}$$

where the dim-$\mathbf{y}_1 = n - m$, dim-$\mathbf{y}_2 = m$, and $det\bar{B} \neq 0$. A second transformation is:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{s} \end{bmatrix} = T_2 \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}. \tag{1.11}$$

Therefore, (1.10) becomes in the following form:

$$\begin{aligned} \dot{\mathbf{y}}_1 &= \bar{A}_{11}\mathbf{y}_1 + \bar{A}_{12}\mathbf{s} \\ \dot{\mathbf{s}} &= \bar{A}_{21}\mathbf{y}_1 + \bar{A}_{22}\mathbf{s} + \bar{B}\mathbf{u}. \end{aligned} \tag{1.12}$$

In the sliding mode ($\mathbf{s} = 0$), the dynamics of the system becomes:

$$\dot{\mathbf{y}}_1 = \bar{A}_{11}\mathbf{y}_1. \tag{1.13}$$

1.2.2 Design of Sliding Surfaces

As was mentioned, different design techniques can be used to design sliding modes. In this study, emphasis is focused on the pole placement design method. Other methods such as quadratic optimization and eigen-structure will be mentioned briefly and a reference will be given.

Consider a plant in a regular form

$$\begin{aligned} \dot{\mathbf{x}}_1 &= A_{11}\mathbf{x}_1 + A_{12}\mathbf{x}_2 \\ \dot{\mathbf{x}}_2 &= A_{21}\mathbf{x}_1 + A_{22}\mathbf{x}_2 + B_2\mathbf{u}, \end{aligned} \tag{1.14}$$

where the dim-$\mathbf{x}_1 = n - m$, dim-$\mathbf{x}_2 = m$, and dim-$\mathbf{u} = m$. The switching function has the form

$$\mathbf{s} = C_1\mathbf{x}_1 + C_2\mathbf{x}_2. \tag{1.15}$$

Solve (1.15) for $\mathbf{x}_2$

$$\mathbf{x}_2 = C_2^{-1}\mathbf{s} - C_2^{-1}C_1\mathbf{x}_1, \tag{1.16}$$

and assume

$$K = C_2^{-1}C_1. \tag{1.17}$$

Substituting (1.16) and (1.17) into (1.14) gives

$$\begin{aligned} \dot{\mathbf{x}}_1 &= (A_{11} - A_{12}K)\mathbf{x}_1 + A_{12}C_2^{-1}\mathbf{s} \\ \dot{\mathbf{s}} &= [C_1A_{11} + C_2A_{21} - (C_1A_{12} + C_2A_{22})K]\mathbf{x}_1 + \\ &\quad (C_1A_{12} + C_2A_{22})c_2^{-1}\mathbf{s} + C_2B_2\mathbf{u}. \end{aligned} \tag{1.18}$$

In the sliding mode $\mathbf{s} = 0$ and $\dot{\mathbf{s}} = 0$, then (1.18) becomes

$$\dot{\mathbf{x}}_1 = (A_{11} - A_{12}K)\mathbf{x}_1, \tag{1.19}$$

$$\mathbf{u} = (C_2B)^{-1}[C_1A_{11} + C_2A_{21} - (C_1A_{12} + C_2A_{22})K]\mathbf{x}_1. \tag{1.20}$$

(1.19) describes the sliding mode dynamics while (1.20) describes the equivalent control function. The characteristic equation of the close-loop system in (1.19) is

$$|zI - (A_{11} - A_{12}K)| = 0. \tag{1.21}$$

The desired characteristic equation can be put into the form

$$(z - \lambda_1)(z - \lambda_2) \ldots (z - \lambda_{n-m}) = 0, \tag{1.22}$$

where λ_i's are the locations of the desired zeros of the closed-loop characteristic equation. By equating the coefficients of (1.21) and (1.22), the elements of the matrix K can be found. This matrix is called the gain matrix that will realize the desired characteristic equation. Finally, in (1.17) C_2 is arbitrary. Assume $C_2 = I$ the identity matrix, then

$$C = [K \quad I] \tag{1.23}$$

which is the desired coefficients of the switching function.

A second method to find the coefficients of the switching function is using the quadratic optimization technique where a cost function is minimized to obtain an optimum value for the elements of the matrix C [4].

The third and final method that can be used is the eigen-structure method. It relies on finding the eigenvectors and their eigenvalues that lie in the space of the switching surfaces and solving for C_1 in terms of the eigenvectors [4].

1.3 Design of the Reaching Mode Using the Reaching Law Method

In this section a discussion of the design of the reaching mode will be presented. The design method is based on the reaching law method [5],[6]. The design considered is for systems presented in the form of (1.1) to (1.4)

1.3.1 Reaching Condition and the Reaching Mode

For the state trajectory of any system to reach a sliding mode, a reaching condition must be satisfied. Several methods were suggested to specify how the reaching

should occur. One important property of such methods is the global reaching property.

One of the earliest methods is called the direct switching function method [1],[2]. The basic idea behind this method is the switching function and its derivative must have opposite signs at all times in the reaching phase. This method requires

$$s_i \dot{s}_i < 0 \quad i = 1, 2, \ \ldots \ , m. \tag{1.24}$$

The method is implemented using a pre-defined discontinuous control action. Although this method is global, it does not guarantee a finite reaching time. It is also very difficult to use for MIMO VSC systems design [5].

In this study a reaching law method is used. The reaching law method makes use of differential equations to specify the dynamics of the switching function [5],[6]. By changing the parameters of the reaching law, one can control the dynamics of the switching function. Therefore, the reaching law assists in controling the reaching mode of a VSC system. It establishes the reaching condition implicitly without the use of a reaching condition as in (1.24).

1.3.2 Design of A VSC Law Using the Reaching Law Method

In Section 1.2 the design of the sliding modes with certain characteristics that guarantees global asymptotic stability was discussed. In this section the design of a control function that drives and constrain the system state to lie and stay within these modes is considered:

The reaching law method can be used to specify the dynamics of the switching

function. This method is implemented using a control law. For an nth-order linear time-invariant system with a single input represented by the form

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}u + \mathbf{b}d, \tag{1.25}$$

where u is the control signal and d is the equivalent perturbation in the system, a control function is to be designed. The equivelant perturbation is modeled as an input to the plant. The dim-$\mathbf{x}$=n, and A and $\mathbf{b}$ are of appropriate dimensions.

Lets assume that a reaching law is in the form

$$\dot{s} = -ks, \tag{1.26}$$

where $k > 0$, and s is the switching function defined by:

$$s = \mathbf{c}^T\mathbf{x}, \tag{1.27}$$

where $\mathbf{c}^T = [c_1, c_2, \ \ldots \ , c_{n-1}, 1]$. Using (1.26), and (1.27)

$$\dot{s} = \mathbf{c}^T\dot{\mathbf{x}} = -k\mathbf{c}^T\mathbf{x}. \tag{1.28}$$

Now, substitute (1.25) into (1.28) and solve for u

$$u = (\mathbf{c}^T\mathbf{b})^{-1}[(-k\mathbf{c}^T - \mathbf{c}^T A)\mathbf{x} - \mathbf{c}^T\mathbf{b}d]. \tag{1.29}$$

In (1.29), the control function u consists of two terms: a linear feedback term and a constant term which neutralizes system perturbations. If d is zero, the control is completely linear feedback control. If d is known (measurable) and has a value. the constant term will cancel it out. Therefore, the reaching condition is satisfied and no steady state errors are present. However, usually, d is unknown

and cannot be measured. Hence, the second term is of no use. To satisfy the reaching condition , i.e., (1.24), a conservative value q should replace d in the control function above, where

$$-q < d < q. \tag{1.30}$$

Therefore, (1.29) becomes:

$$u = (\mathbf{c}^T\mathbf{b})^{-1}[(-k\mathbf{c}^T - \mathbf{c}^T A)\mathbf{x} - \mathbf{c}^T\mathbf{b}q]. \tag{1.31}$$

Substituting, (1.31) into (1.28) and simplifying the results, gives

$$\dot{s} = -ks - \mathbf{c}^T\mathbf{b}(q - d). \tag{1.32}$$

To ensure the reaching condition, the term q must dominate the d term. It is reasonable to assume bounds for d,

$$D_L \leq d \leq D_H. \tag{1.33}$$

Using (1.24) and (1.32), q is chosen according to the following logic.

$$\begin{aligned} &When \;\; s > 0 \;\; and \;\; \dot{s} < 0, \;\; then \; set \;\; q = D_H \\ &When \;\; s < 0 \;\; and \;\; \dot{s} > 0, \;\; then \; set \;\; q = D_L. \end{aligned} \tag{1.34}$$

Define

$$d' = (D_H + D_L)/2 \quad and \quad d'' = (D_H - D_L)/2, \tag{1.35}$$

then, using the logic in (1.34)

$$q = d' + d''sgn(s). \tag{1.36}$$

This term is called the signum term of a VSC system. Substituting (1.36) into (1.31) gives

$$u = (\mathbf{c}^T\mathbf{b})^{-1}[(-k\mathbf{c}^T - \mathbf{c}^T A)\mathbf{x} - \mathbf{c}^T\mathbf{b}(d' + d''sqn(s))]. \tag{1.37}$$

If in (1.35) $D_H = -D_L$, then (1.37) becomes

$$u = (\mathbf{c}^T\mathbf{b})^{-1}[(-k\mathbf{c}^T - \mathbf{c}^T A)\mathbf{x}] - q\ sqn(s). \tag{1.38}$$

The signum term in (1.38) takes care of perturbations present in the system. Moreover, a large magnitude of this term results in a robust system over a wide range of perturbations.

1.4 Problems of VSC Systems

Although VSC systems are useful because of their invariance properties, they result in chattering problems in their state variables. This problem is usually seen at the output of a VSC system. Theoretically, in the analysis of VSC systems, it is assumed that when the state reaches the sliding mode in finite time, the control is switched from one value to another at infinite frequency. In practice, however, it is not realistic to implement. There are always time delays that are either part of the computation processes or part of the switching devices used to realize the switching control action. As a result, when the state reaches the switching surface, it will switch at a time $t_0 + T$ instead of t_0. This will cause a non-ideal reaching which causes chattering (Figure 1.1). The larger T is , the larger the chattering is. Moreover, the larger the magnitude of the signum term in (1.38), the larger

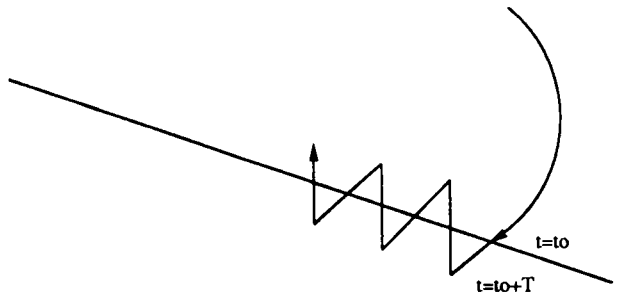


Figure 1.1: Chattering caused by a non-ideal reaching

the chattering is at the system output.

The chattering problem occurs in the sliding and steady state modes of a VSC system. In the steady state, it appears as a high-frequency oscillation about a desired equilibrium point (a limit cycle). This could excite the unmodeled high-frequency dynamics of the system [8],[9],[10]. Furthermore, when the chattering is large and perturbations are present, the limit cycle will stablize at an equilibrium point where its mean velocity is zero. This could result in a steady state error as shown in Figure 1.2.

Two approaches have been used to handle the chattering problem. The first approach is called the continuation method because of the discontinuous relay type actuator is replaced by a high gain device with saturation [10],[11]. Although this method eliminates the high frequency chattering, it also destroys the sliding mode. The second approach deals directly with the reaching process since chattering is caused by the non-ideal reaching at the end of the reaching phase [6]. This

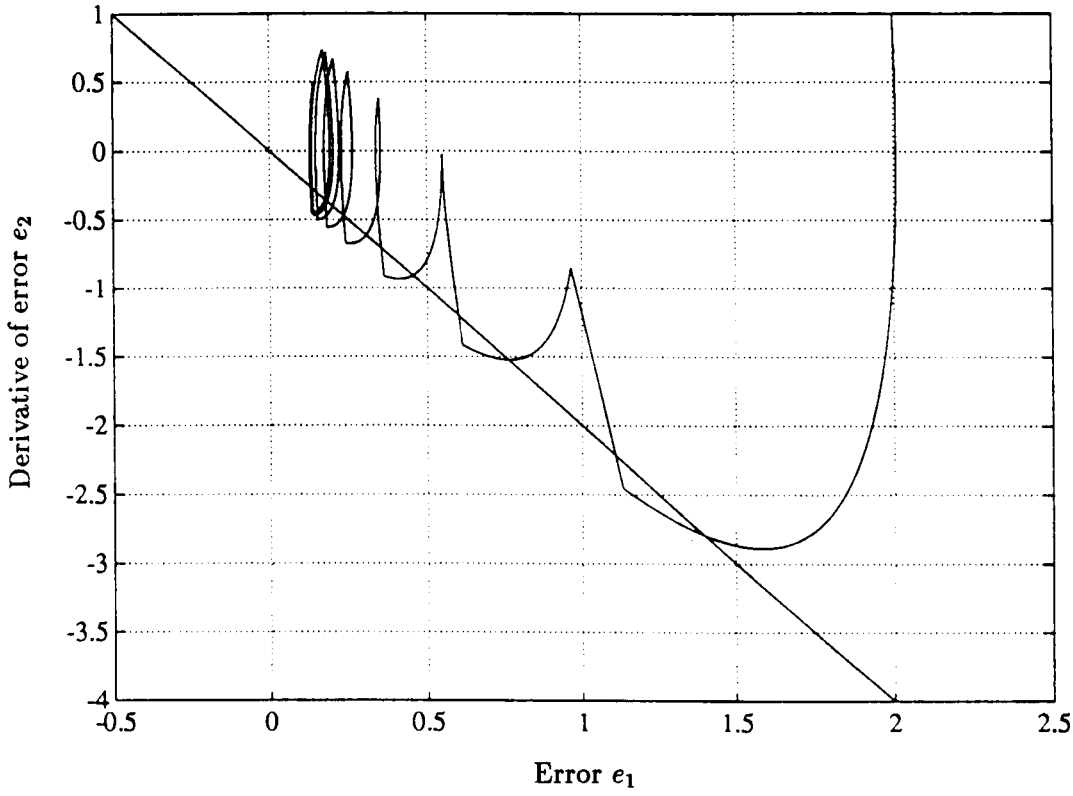


Figure 1.2: A plot shows a steady state error caused by the stablization of the limit cycle at an equilibrium point where its mean velocity is zero

approach is called the reaching law method which already has been presented. The alleviation of chattering is done by reducing the magnitude of the signum term of reaching law. As a result, the range of robustness of a VSC system is narrowed.

1.5 Objective of the Study

It was mentioned above, the control function of a VSC system has usually two terms : a linear feedback term and a signum term. The signum term is used to obtain a robust VSC system. However, a large signum term causes chattering when there is a time delay in the switching function. When perturbations are absent, one can get ideal sliding mode by the linear feedback term for a linear system without having the signum term. However, with perturbations present, the signum term must be included to neutralize their effects. That is, one eliminates the effects of perturbations at the expense of tolerating chattering. Therefore, it is desirable to have the magnitude of the signum term just big enough to handle any perturbation present. When perturbation disappears, the magnitude of the signum term can be made zero. Thus, the design of adaptive magnitude of the signum term based on an estimate of perturbation is encouraging and would result in a low chattering VSC system on the average.

This study proposes a new approach to achieve low chattering in a single input VSC system without sacrificing the robustness of the system. This approach incorporates the fuzzy decision making and adaptive adjustment into the VSC

system. Two fuzzy adaptive algorithm are used. One algorithm provides a feature to reduce the average chattering in the sliding mode (VSC on demand). The other controls the speed of the state in the reaching mode to reduce the chattering caused by the non-ideal reaching. The result is a low chattering VSC system.

CHAPTER 2

Adaptive VSC SYSTEM

2.1 General System Configuration

The key to a fuzzy adaptive VSC is the on-line adjustment of the magnitude of the signum term of the system. When perturbations are present in the system, the signum term must be adjusted to a value which is big enough to maintain the robustness property of the system. Furthermore, in the reaching mode, when the state speed is fast and a delay in the switching mechanisim is present, the value of the signum term again should be adjusted to reduce the chattering effect due to the non-ideal reaching.

The concept of the above approach is depicted in Figure 2.1. The system in the figure includes four different subsystems to be implemented. These subsystems are the plant, a VSC, a fuzzy decision maker, and a perturbation observer. The equivalent perturbation in the system is modeled as an input to the plant and it includes all parameter changes, system uncertainties, and external disturbances. All the blocks and their functions are described below:

2.1.1 The Plant

The plant under consideration is a linear time-invariant SISO which is pre-

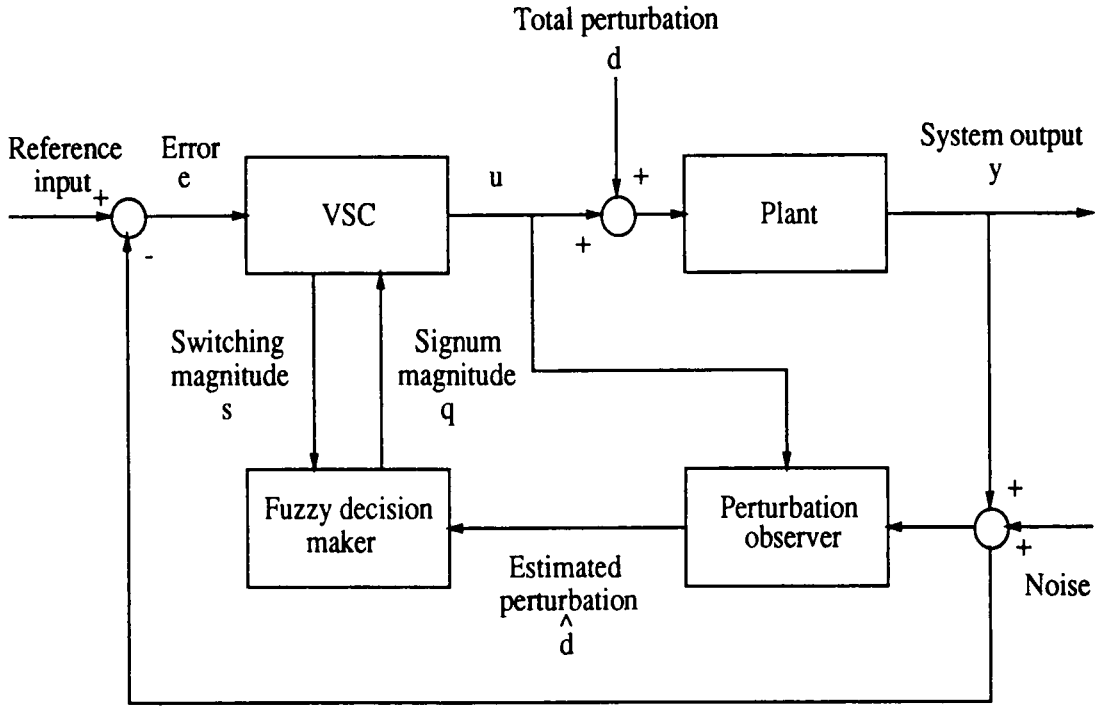


Figure 2.1: An overall block disgram of a fuzzy adaptive VSC system

sented by:

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}u + \mathbf{b}d, \tag{2.1}$$

where the dim-$\mathbf{x} = n$, and A and $\mathbf{b}$ are of appropriate dimensions. d is the total perturbation in the plant and is assumed to enter the plant at its input. The transfer function presentation of the plant in (2.1) is

$$G_p(s) = \frac{b}{(s^n + a_{n-1}s^{n-1} + \ \ldots \ + a_0 s^0)}. \tag{2.2}$$

2.1.2 VSC

A nominal VSC function for the plant in (2.1) is found using (1.38)

$$u = (\mathbf{c}^T\mathbf{b})^{-1}[(-k\mathbf{c}^T - \mathbf{c}^T A)\mathbf{x}] - q\ sqn(s). \tag{2.3}$$

where $s = \mathbf{c}^T\mathbf{x}$ is the switching function. The function of the VSC block includes observation of the plant state, computing the value of switching, and exerting a VSC on the plant based on its control law. Unlike a usual VSC, the magnitude of the signum term is adjusted and may take different values at different times according to the adjustement command coming from the fuzzy decision maker.

2.1.3 Fuzzy Logic

The function of this fuzzy logic system includes receiving the estimated perturbation from the perturbation observer, receiving the value of switching from the VSC, performing necessary data reduction, inferring a decision on the magnitude of the signum term, and issuing the decision to VSC. The details of this block will

be presented when discussing the design of adaptive sliding and reaching modes in the following material.

2.1.4 Perturbation Observer

The function of the perturbation observer block is to estimate a value for the total perturbation using data coming form the system output and VSC output. The design of a perturbation observer is based on finding an inverse of the plant presented in (2.2). The inverse of G_p is simply

$$T(s) = \frac{1}{G_p(s)} = \frac{s^n + a_{n-1}s^{n-1} + \ \ldots \ + a_0 s^0}{b}. \tag{2.4}$$

The design of the perturbation observer is summarized in Figure 2.2. Ideally, the design in Figure 2.2 will give an exact estimate of the perturbation. In practice, however, the inverse plant ($1/G_p$) is not realizable for a second order system or higher due to the unstable characteristics of differentiator Op-Amps used for such realization. The transfer function in (2.4) would be realizable if it were made proper. To make the transfer function proper, one needs to add a low pass filter to it. Therefore, the overall transfer function of the inverse plant becomes:

$$T(s) = \frac{G_{lp}(s)}{G_p(s)} = \frac{s^n + a_{n-1}s^{n-1} + \ \ldots \ + a_0 s^0}{b(\frac{1}{\omega_c^n}s^n + \ \ldots \ + \frac{\sqrt{2}}{\omega_c}s + 1)}, \tag{2.5}$$

where ω_c is the cutoff frequency.

In Figure 2.2, $\hat{d}$ is the estimated perturbation which is different from the real perturbation. This value is fuzzy and contains high noise. This estimated value will be input to a fuzzy logic system. This logic system will handle the fuzziness in $\hat{d}$.

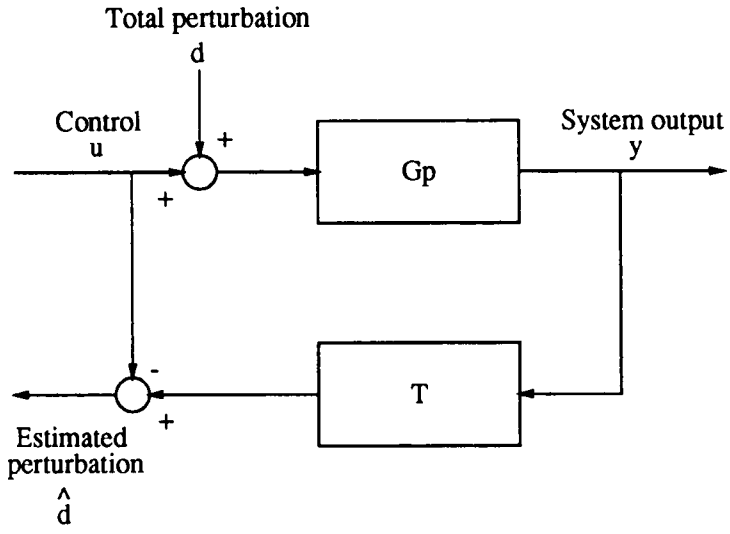


Figure 2.2: A perturbation observer

2.2 Fuzzy Adaptive Sliding Mode

In general, a fuzzy system provides a set of heuristic decision rules based on expert knowledge. It is a method of easily representing analog processes on a digital computer. Usually, the analog processes are uncertain and have no mathematical models as the case of system perturbation. The components of a fuzzy logic system are fuzzifiers, Knowledge respostory (lookup tables), and defuzzifiers [12]. The task carried by a fuzzifier is to convert inputs into their fuzzy representation. On the other hand, defuzzifiers converts the output of a fuzzy logic process into numerical solution variables. The values of the fuzzified inputs execute all rules in the knowledge respostory that have the fuzzified inputs as their premise. This process generates new fuzzy sets representing the solution

variables. Finally, defuzzifiers creates numerical values for the output variables from these newly constructed fuzzy sets.

Rules in a fuzzy system are usually in the form:

- If control variables then solution variables

From Figure 2.1, to design a fuzzy system which results in a robust sliding mode, three fuzzy variables are used. These include two control (input) variables ($\hat{d}$ and $\dot{\hat{d}}$) and one solution (output) variable (q). Each variable is presented by a fuzzy region which includes a number of fuzzy sets. Each fuzzy set has a unique name or label. For example, for the above fuzzy variables, each fuzzy variable region has seven labels that presents seven fuzzy sets which are defined in the following:

- NL: a fuzzy label meaning *Negative* and *Large.*
- NM: a fuzzy label meaning *Negative* and *Medium.*
- NS: a fuzzy label meaning *Negative* and *Small.*
- ZR: a fuzzy label meaning *Zero.*
- PS: a fuzzy label meaning *Positive* and *Small.*
- PM: a fuzzy label meaning *Positive* and *Medium.*
- PL: a fuzzy label meaning *Positive* and *Large.*

The membership function for the normalized variables mentioned above is shown in Figure 2.3.

An understanding of the relationship between the variables is important. The variable q must dominate the variable $\hat{d}$ as was indicated in chapter 1 so that the reaching condition is satisfied. This means

$$-q < \hat{d} < q. \tag{2.6}$$

The relationship between q and $\hat{d}$ is linear. When $\hat{d}$ increases, q must also increase. Using the fuzzy regions and based on the relationships among all fuzzy variables, a fuzzy rule table can be constructed. The number of rules depends on the number of labels and the number of input variables, i.e, for a two input system with seven lable fuzzy regions, 49 rules are constructed. For the above system, some suggesed fuzzy rules are constructed in Table 2.1. For illustration purposes two rules of Table 2.1 are expressed explicitly below:

- If $\hat{d}$ is ZR and $\dot{\hat{d}}$ is PM then q is PS.
- If $\hat{d}$ is PS and $\dot{\hat{d}}$ is PS then q is PS.

The first rule states that if the input value of perturbation $\hat{d}$ is Zero and its derivative $\dot{\hat{d}}$ is PM, then the magnitude of the signum term should be positive and small. The second one states that if $\hat{d}$ is positive small and its derivative $\dot{\hat{d}}$ is positive small, then q is positive small. Using the rules in Table 2.1, a program

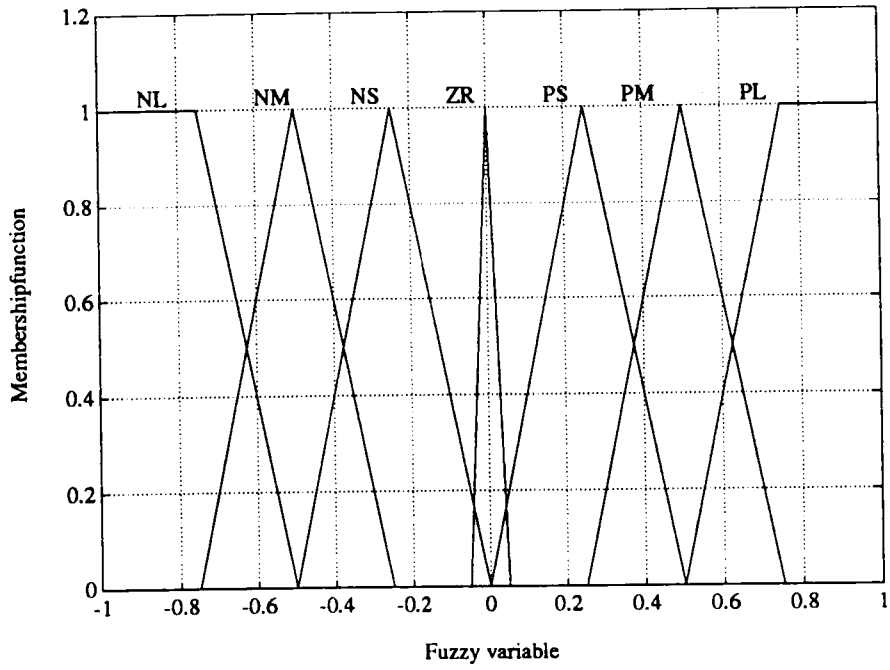


Figure 2.3: A membership function for the normalized variables $\hat{d}$, $\dot{\hat{d}}$, and q

Table 2.1: Fuzzy system rules for two input variables and one output variable

$\dot{\hat{d}}$ \ $\hat{d}$	NL	NM	NS	ZR	PS	PM	PL
NL	NL	NL	NM	NS	ZR	PS	PM
NM	NL	NL	NM	NS	ZR	PS	PM
NS	NL	NM	NS	ZR	PS	PM	PL
ZR	NL	NM	NS	ZR	PS	PM	PL
PS	NL	NM	NS	ZR	PS	PM	PL
PM	NM	NS	ZR	PS	PM	PL	PL
PL	NM	NS	ZR	PS	PM	PL	PL

could be written. The task of this program is to read the value of $\hat{d}$ and its derivative, find the fuzzy sets to which they belong, construct a fuzzy set for the solution variable q, and finally deffuzzify q to a numerical value. The construction of new fuzzy sets is based on the rules presented in Table 2.1. The task of such program is summarized in a flowchart shown in Figure 2.4.

As a conclusion, it should be mentioned that the above fuzzy procedure is not general. Any other procedure may be implemented depending on experimental observaions of the relationships among variables. For example, if the change in perturbation is smooth, then there is no need to use an input variable as $\dot{\hat{d}}$ in the design procedure. Only the variable $\hat{d}$ and q may be used.

2.3 Fuzzy Adaptive Reaching Mode

The signum term magnitude is used to implement a robust control in the sliding mode and also to bring about a very fast reaching to the sliding mode. The design of a low chattering robust VSC systems was completely accomplished in the preceding section by using an adaptive q such that the reaching condition is satisfied at all times. Since in the reaching mode the invariance property is not guaranteed, one needs to control the reaching mode dynamics and force the state to the sliding mode in a very short time. This could be done by increasing the value of q. But if the magnitude of q is large and there exists a time delay in the switching mechanisim, then a serious chattering problem at the system output will exists. Therefore, it is necessary to either eliminate or reduce the chattering.

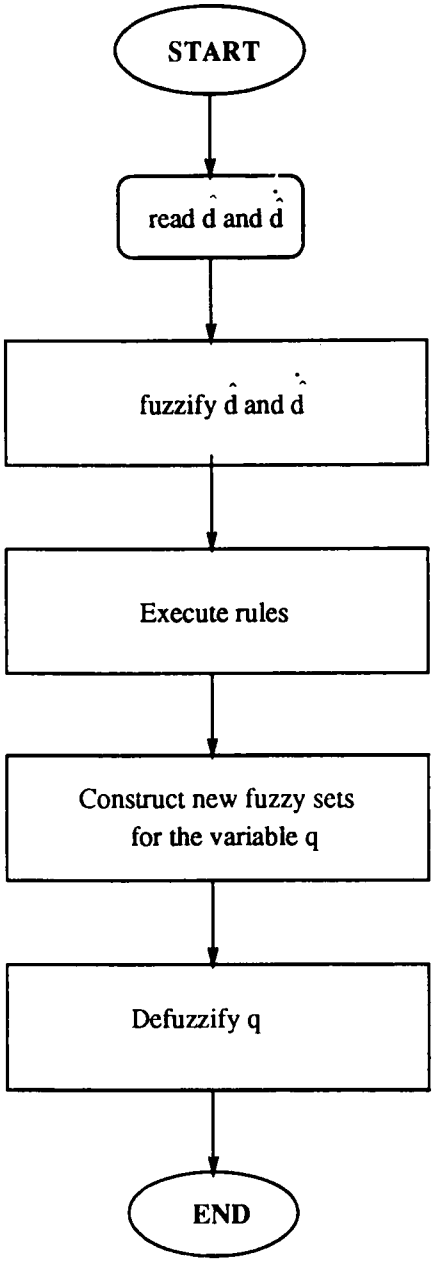


Figure 2.4: Flowchart of fuzzy logic program which implements fuzzy adaptive VSC in the sliding mode

This is done by adjusting magnitude of signum term right before reaching the sliding mode.

Consider the general reaching law in the form

$$\dot{s}_i = -k_i s_i - q_i sgn(s_i) \qquad i = 1, 2, \ ... \ , m. \tag{2.7}$$

When the the switching surface $s_i = 0$, then $|\dot{s}_i| = q_i$. By choosing q_i small, the momentum of the motion will be reduced as the system approaches the switching surface. As a result, the amplitude of chatter will be reduced. However, q_i can not be chosen zero because the reaching time would become infinite. Therefore, by adjusting the the signum term q_i, one can reduce the chattering problem and nearly force an ideal reaching.

Based on (2.7), a fuzzy adaptive algorithm can be implemented. As can be noticed in (2.7), when q increases, the value of s is decreased proportionally. Consequently, increasing q will not increase $\dot{s}$. Based on this observation, a fuzzy adaptive reaching mode is designed. The inputs of the fuzzy system are s and $\dot{s}$, and its output is the a numerical value for the variable q. Each fuzzy variable is presented by three fuzzy sets: Negative (N), Zero (ZR), and Positive (P) as shown in Figure 2.5. The system rules are summarized in Table 2.2. One example of Table 2.2 is described below:

- if s ZR and $\dot{s}$ N then q is ZR

The above states that if s is zero (small) and $\dot{s}$ is negative (fast reaching), then

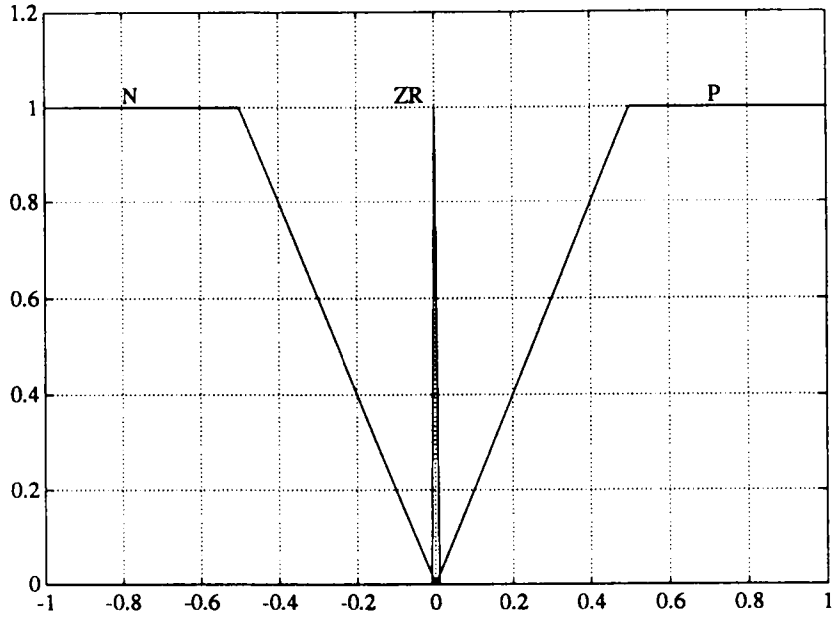


Figure 2.5: A membership function for the normalized variables s, $\dot{s}$, and q

Table 2.2: Fuzzy system rules for two input variables and one output variable

$\dot{s}$ \ s	N	ZR	P
N	ZR	ZR	P
ZR	N	ZR	P
P	N	ZR	ZR

the value of q should be made small but not zero to achieve a finite reaching. Using Table 2.2, a program could be written. The task of the program is summarized in the flowchart shown in Figure 2.6. The result of this algorithm is a fast reaching VSC with low chattering effect.

2.4 Fuzzy Adaptive VSC in the Reaching and Sliding Modes

In the previous two sections, two fuzzy adaptive algorithm were implemented. In this section, these two approaches will be combined into one approach that will handle all objectives (a low chattering, fast, robust VSC system) as follows:

Assume that the adaptive signum term for robust sliding mode is $q_{sliding}$, and for the reaching mode is $q_{reaching}$. To combine the two objectives into one, simply we add the two values for the signum term as the following:

$$q_{total} = q_{sliding} + q_{reaching}, \tag{2.8}$$

where q_{total} is the overall magnitude of the signum term. As a result of the above, a low chattering robust VSC system with fast dynamics is obtained.

In the next chapter, the results of the above design objectives will be shown through computer simulations. It will be shown that a fast robust VSC system which has a little chattering effect could be designed and implemented. As a conclusion, the reaching law method is the essense of designing such VSC systems. Tuning the paramters of the reaching law gives flexibility in the design. This flexibility is obvious when using technologies such as fuzzy logic which acts as a decision maker.

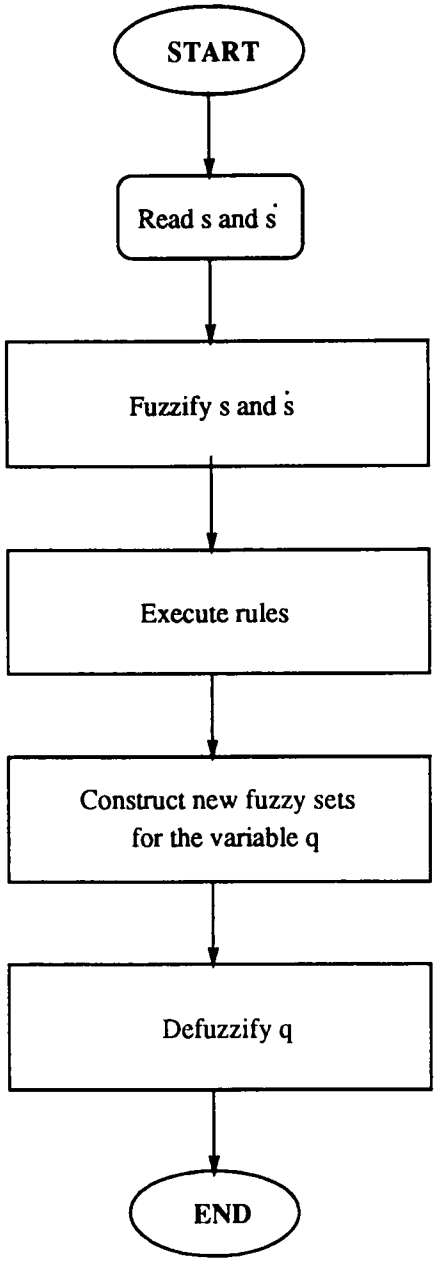


Figure 2.6: Flowchart of fuzzy logic program which implements a fuzzy adaptive VSC in the reaching mode

CHAPTER 3

Experimentation by Simulation

In the previous chapter, the design of a fuzzy adaptive VSC system was presented. In this chapter, the performance of such system is experimented through computer simulations . A dc voltage regualtion system is used to demonstrate the proposed method. All simulations are done using Matlab software.

3.1 A plant for A Generator Voltage Regulation System

The voltage regulation system is shown in Figure 3.1, Using the figure, a mathematical model is derived for the plant G_p. The transfer function of the plant is:

$$G_p(s) = \frac{V_c}{V_a} = \frac{\frac{R_L K_1 K_2}{(R_g+R_L)L_1 L_2}}{s^2 + \frac{(R_1 L_2 + R_2 L_1)}{L_1 L_2} s + \frac{R_1 R_2}{L_1 L_2}}, \tag{3.1}$$

where $R_1 = 100\ ohm$, $R_2 = 4\ ohm$, $L_1 = 20\ H$, $L_2 = 4\ H$, $K_1 = 200\ V/A$, $K_2 = 20\ V/A, R_g = 5\ ohm$, V_a is the control signal, and V_c is the output signal. Substituting the above values in (3.1), the transfer function becomes:

$$G_p(s) = \frac{10\frac{R_L}{5+R_L}}{(1+\frac{1}{5}s)(1+s)}. \tag{3.2}$$

The above transfer function can be written in the following state variable form assuming $V_c = x_1$ and $\dot{V_c} = x_2$:

$$\dot{x1} = x_2$$

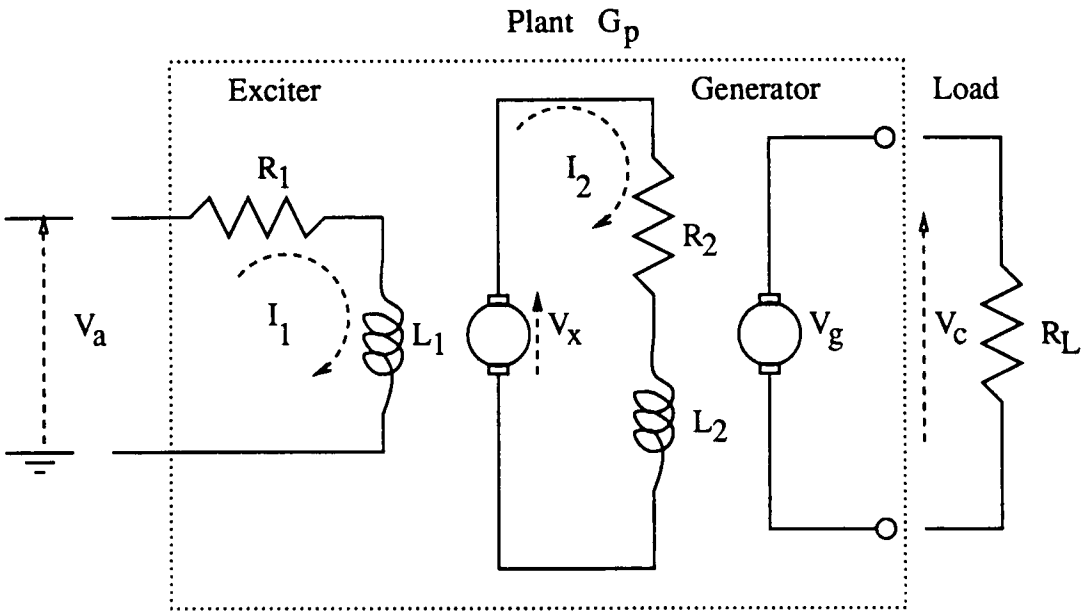


Figure 3.1: A Generator Voltage Regulation System

$$\dot{x_2} = -5x_1 - 6x_2 + 50\frac{R_L}{5 + R_L}u, \tag{3.3}$$

where u is the control signal.

The plant in (3.3) may be under perturbations that should also be included in the plant model. These perturbations includes parameter variations, external disturbances, and any other plant uncertainties. For example, the load R_L may change with time due to some load disturbances. Therefore, if the magnitude of R_L is changing, then the system in (3.3) may be written in the following form:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -5x_1 - 6x_2 + (b + \triangle b)u, \end{aligned} \tag{3.4}$$

where $b = 50\ ohm/sec^2$ (nominal plant with no load), and $\triangle b = \frac{-250}{5+R_L}\ ohm/sec^2$ is the change in b. When $R_L = 0$ (short circuit), $\triangle b = -50$. When $R_L = \infty$ (open circuit), $\triangle b = 0$. Therefore, b may change beween 0 and 50 depending on the load magnitude. An equivalent perturbation d which includes all of the above uncertainties, i.e., $\triangle b$, is modeled into the system, and it is assumed to enter the plant at its input as was mentioned in Section 2.1.1. Thus, (3.4) may be written in the following form:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -5x_1 - 6x_2 + bu + bd. \end{aligned} \tag{3.5}$$

In the following simulations, a specific perturbation profile which changes with time is used.

3.2 Control by Usual VSC Law

For the plant in (3.5), It is desired to design a VSC using the reaching law method. Using (1.38), a control function is found

$$u = \frac{1}{b}[(5 - kc_1)x_1 + (6 - c_1 - k)x_2 - q\ sgn(s)], \tag{3.6}$$

where x_1 and x_2 are the error and its derivative, respectively, assuming a reference input of zero. The value of $s = \mathbf{c}^T\mathbf{x} = \begin{bmatrix} 3 \\ \\ 1 \end{bmatrix}\mathbf{x}$, which is found using pole placement method explained in chapter 1. The value of k is chosen to be 5 so that the input to the plant is not big. Three cases of simulations are performed on this usual VSC system to demonstrate its robustness quality through the use of the signum term. The initial condition used in all the simulations of this chapter is $\mathbf{x}_0 = \begin{bmatrix} 2 \\ \\ 0 \end{bmatrix}$, and a block diagram which represents this VSC system is shown in Figure 3.2.

In case 1, for an unperturbed system, the first term in the control function u (linear feedback) is used. The signum term q is set to zero at all times. The results are shown in Figure 3.3. Since there are no perturbations, the system response is satisfactory and there is no need for the signum term. Case 2 is the same as the first one except a perturbation profile d is inputed into the plant. The results are shown in Figure 3.4, and as can be seen, steady state errors are produced whenever perturbations are present.

Case 3 is the same as the second one except a signum term with a fixed magnitude which is big enough to handle all perturbations is applied at all times.

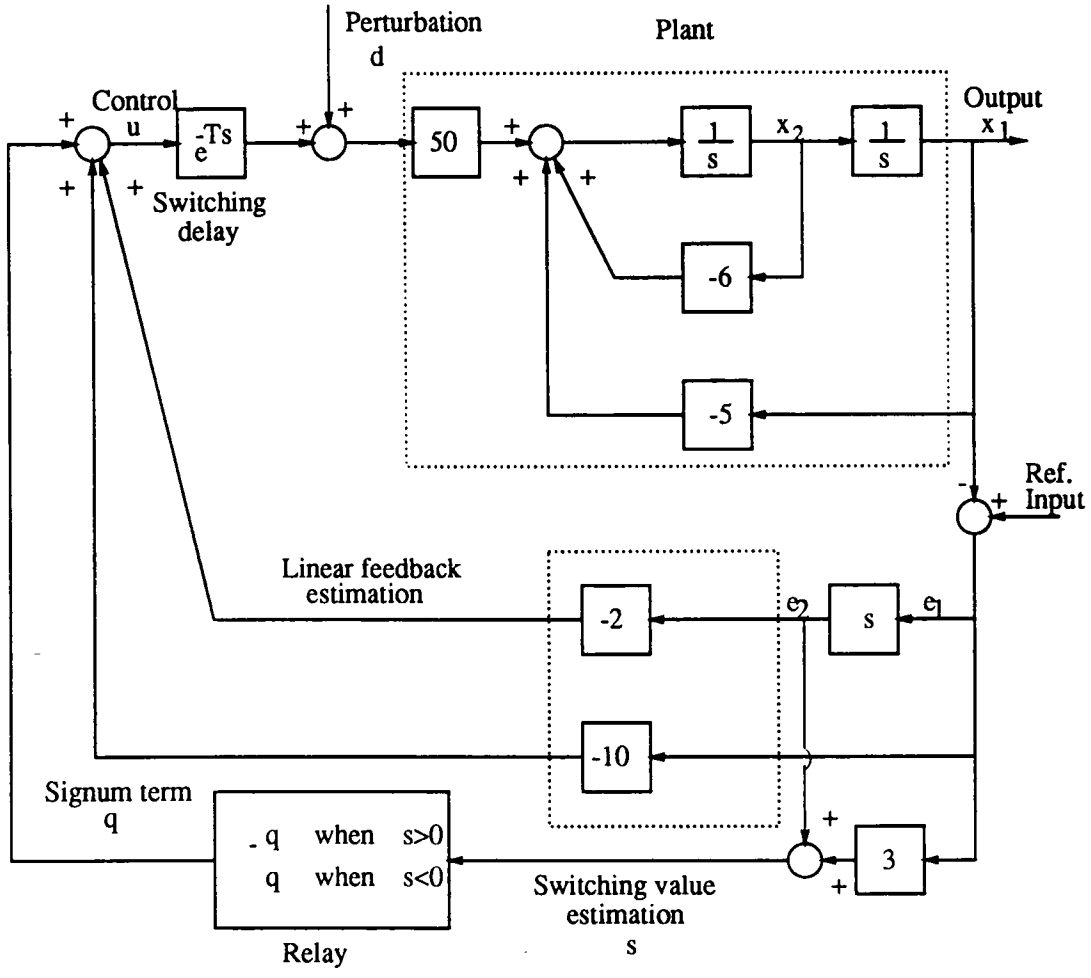


Figure 3.2: Simulation block diagram for a usual VSC system

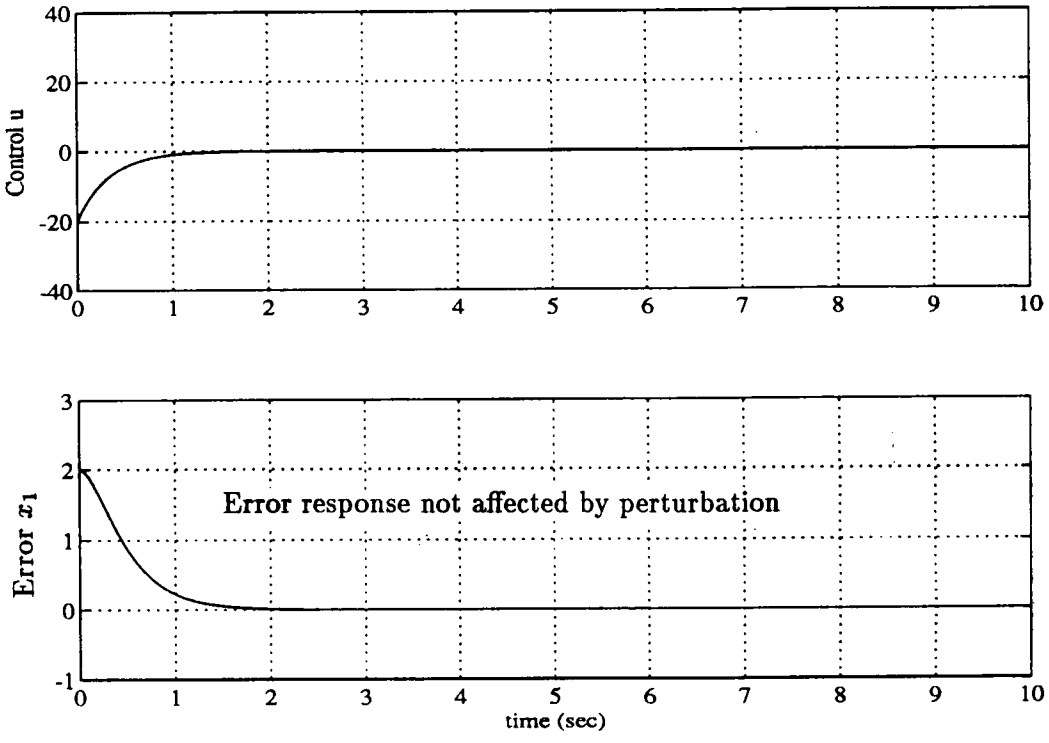


Figure 3.3: Response for VSC, case 1 ($q = 0$ and no perturbation)

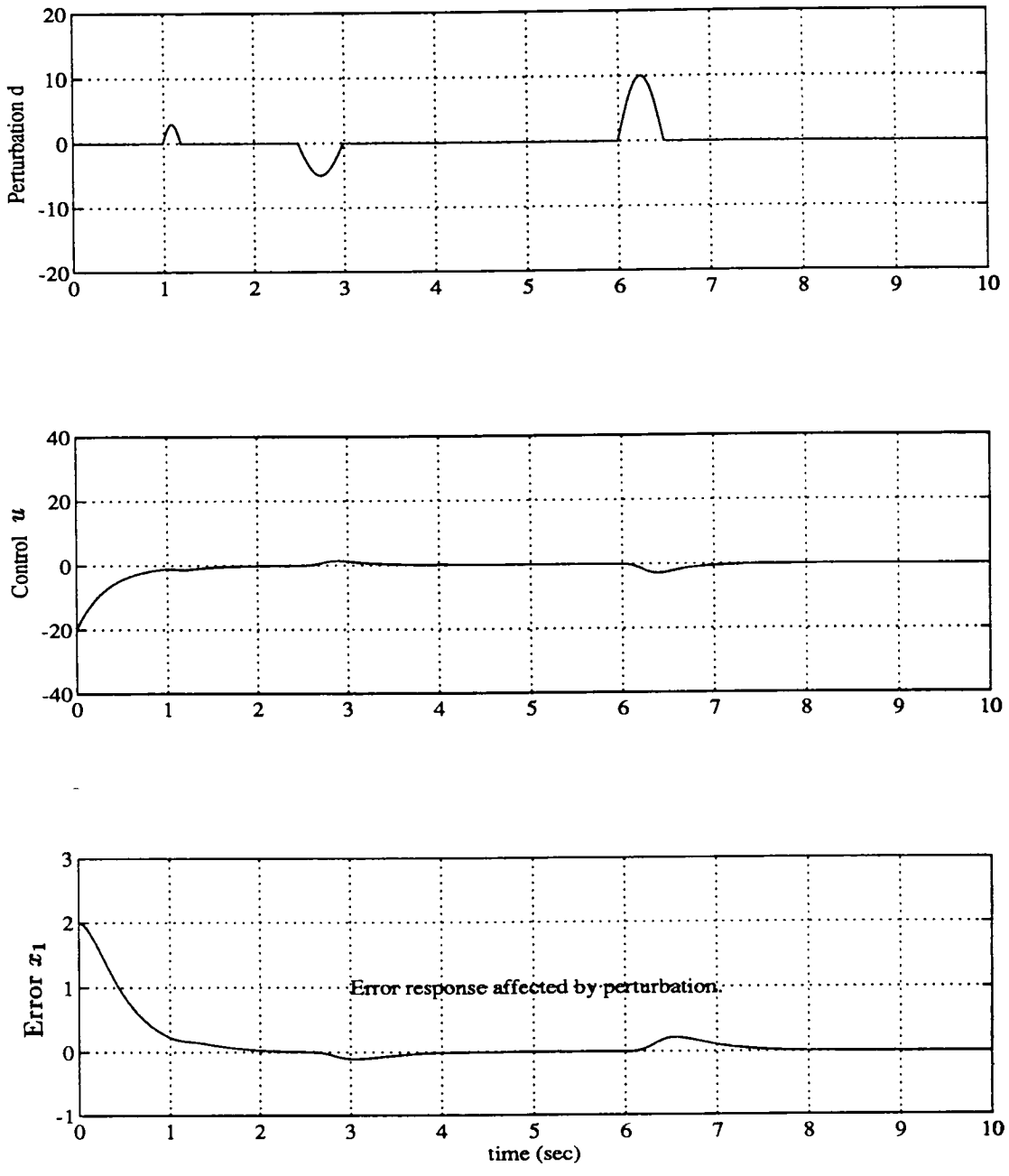


Figure 3.4: Response for VSC, case 2 ($q = 0$, with perturbation)

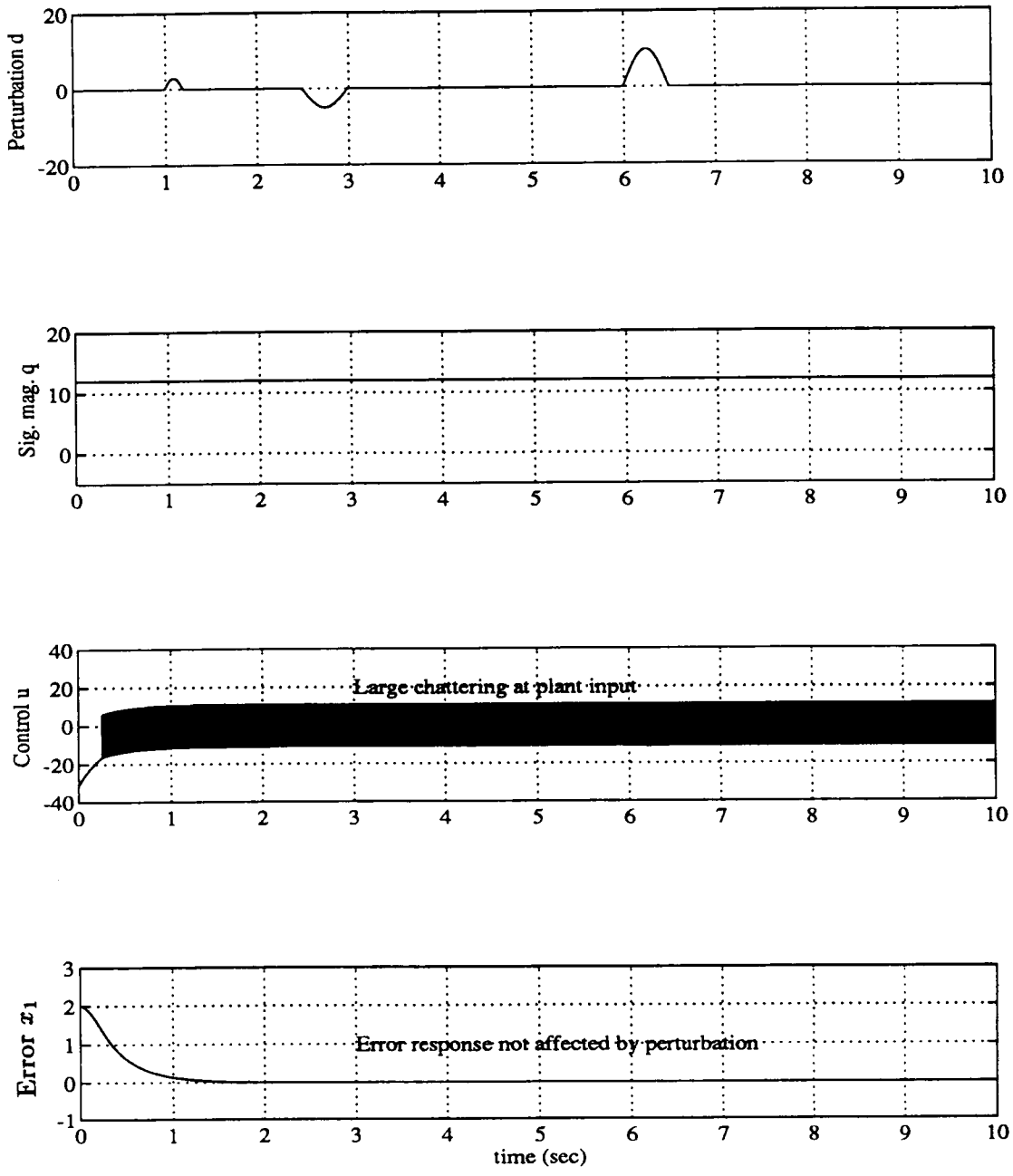


Figure 3.5: Response for VSC, case 3 ($q = 12$, with perturbation)

This suggests that the reaching condition is satisfied at all times. Therefore, no steady state errors are present. This is shown in Figure 3.5. In this simulation, however, if delays are present in the system, the chattering effect will be on at all times which is undesirable.

3.3 Perturbation Observer

Referring to section 2.1.3, a perturbation observer is found for the plant in (3.2),

$$T(s) = \frac{G_{lp}(s)}{G_p(s)} = \frac{(s^2 + 6s + 5)/b}{\frac{1}{w_c^2}s^2 + \frac{\sqrt{2}}{w_c}s + 1}, \tag{3.7}$$

where $w_c = 20$ rad/sec and $b = 50$. The noise present at plant output which is entering the observer is assumed to be white with low amplitude. The output of this observer is the estimated perturbation $\hat{d}$ which is fuzzy. The estimated perturbation will be the input to a fuzzy logic program that will make the adjustment on the magnitude of signum term.

3.4 Fuzzy Adaptive VSC System

In the following , simulations will be performed to reveal the characteristics of fuzzy adaptive VSC systems. This is done by dividing the simulations into three simulation studies. The first is to show fuzzy robust sliding modes by adjusting an adaptive signum term. The second is to show the application of fuzzy adaptive techniques to reaching modes to obtain fast reaching with low chattering. Finally, in the third study, the methods of the above studies are combined.

3.4.1 Fuzzy Adaptive VSC in Sliding Mode only

In this simulation study, three cases of simulation are presented. In case 1, an adaptive signum term which is based on the decision of a fuzzy logic system is used. Based on the data in the perturbation profile used in the simulation studies, a fuzzy system is designed. In this case, a two input fuzzy system is used. The input variables are $\hat{d}$ and its first derivative $\dot{\hat{d}}$, and the output is a numerical value for q ($\dot{\hat{d}}$ was found using a Matlab function which finds the derivatives of mathematical functions). Fuzzy regions which present all fuzzy variables are shown in Figure 3.6. Using Table 2.1 (rule table), a program to implement the above fuzzy system is written and is listed in Appendix A.

To illustrate the procedure of the above program, an example is given. Assume that the magnitude of perturbation $\hat{d}$ is 2 at some time t, and the derivative $\dot{\hat{d}}$ at that time instant is 37. The main task of the program is to find a value of the signum term q based on the data in $\hat{d}$ and $\dot{\hat{d}}$. The first step of the procedure is to normalize these input values into their fuzzy presentation. The normalization factor for $\hat{d}$ is 10 and for $\dot{\hat{d}}$ is 100. Hence, the fuzzy value of $\hat{d}_n$ is 0.2 which is a member of PS fuzzy set, and for $\dot{\hat{d}}_n$ is 0.37 which is a member of two fuzzy sets, PS and PM (Figure 3.7.a). The membership function of $\hat{d}$ in PS is 0.8, and the membership function of $\dot{\hat{d}}$ in PS is 0.52 and in PM is 0.48. As shown in Table 2.1, two rules are in effect. These rules are:

- If $\hat{d}$ is PS and $\dot{\hat{d}}$ is PS then q is PS.
- If $\hat{d}$ is PS and $\dot{\hat{d}}$ is PM then q is PM.

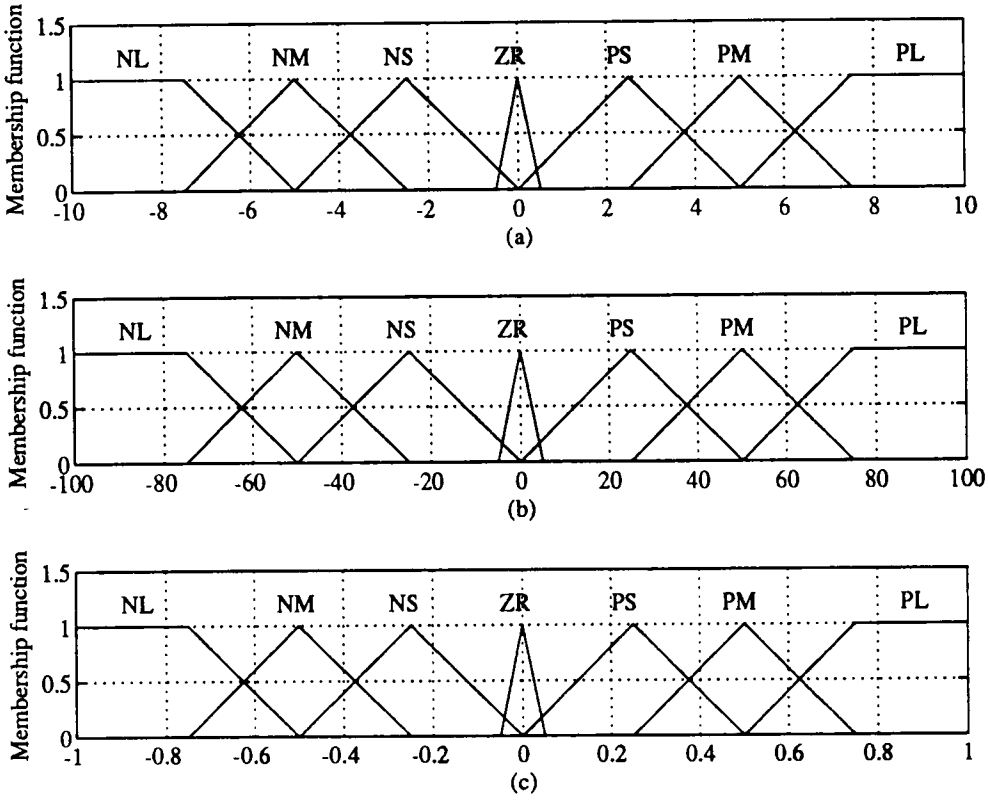


Figure 3.6: (a) $\hat{d}$, (b) $\dot{\hat{d}}$, and (c) normalized fuzzy region

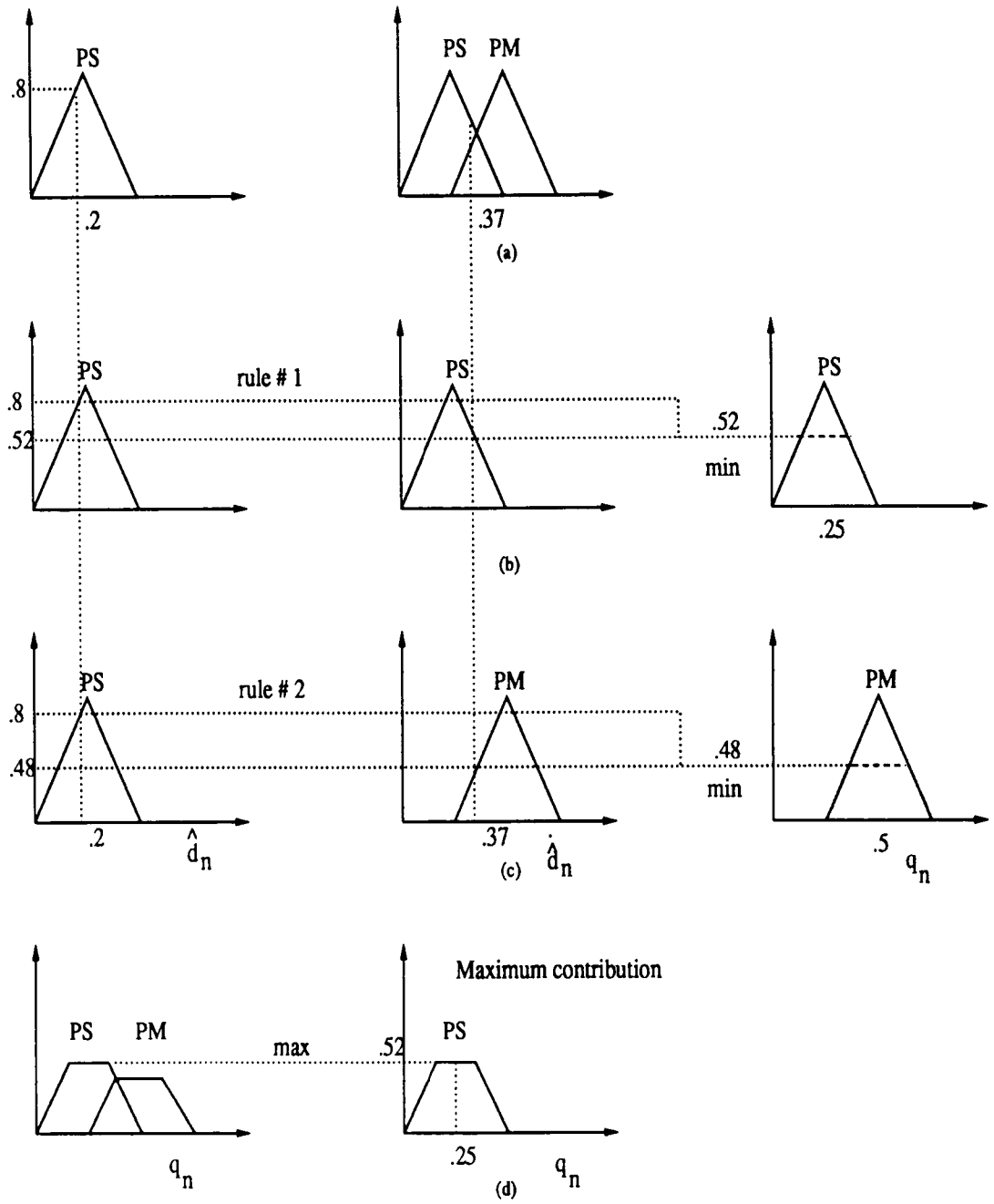


Figure 3.7: Example demonstrates a two input fuzzy logic program procedure

Each rule will contribute to the solution of the fuzzy variable q_n (Figures 3.7.b and 3.7.c). Using the composite maximum method, the set that has the maximum contribution is considered, in this case PS (Figure 3.7.d). Thus, the value of q_n is 0.25. Scaling this fuzzy value by a factor of 15, then the numerical value of q is 3.75 which is larger than $\hat{d}$ and the reaching condition is satisfied, therefore, the system is robust. A simulation block diagram is shown in Figure 3.8, and the results of this case are shown in Figure 3.9.

In case 2, a single input fuzzy system is used to replace the one presented in the previous case. Since the perturbation profile which is used for this specific problem is smooth and does not have large changes in its values, the use of its derivative as an input to the fuzzy system is insignificant. Thus, a single input fuzzy system is sufficient. The input variable is $\hat{d}$ and the output variable is q. The fuzzy rules are constrcuted and shown in Table 3.1. Based on these rules, a fuzzy system program is written. The program is listed in Appendix B.

To illustrate the procedure of the above program, an example is given. Assume at time t the magnitude of perturbation is 6. The first step of the procedure is to normalize the input value of $\hat{d}$ into a fuzzy value that fits into the range of the fuzzy region (Universe of Discourse) as in Figure 3.6.c. The normalization factor is 10. Hence, the fuzzy value is $\hat{d}_n = 0.6$ which is a member of two fuzzy sets, PM and PL (Figure 3.10.a). The membership function of $\hat{d}_n$ in PM set is 0.6 and in PL is 0.4. Two rules are applied in this case and each rule will contribute to the solution variable q_n (Figure 3.10.b and 3.10.c). Using the composite maximum method, the set that has the maximum contribution is considered, in this case

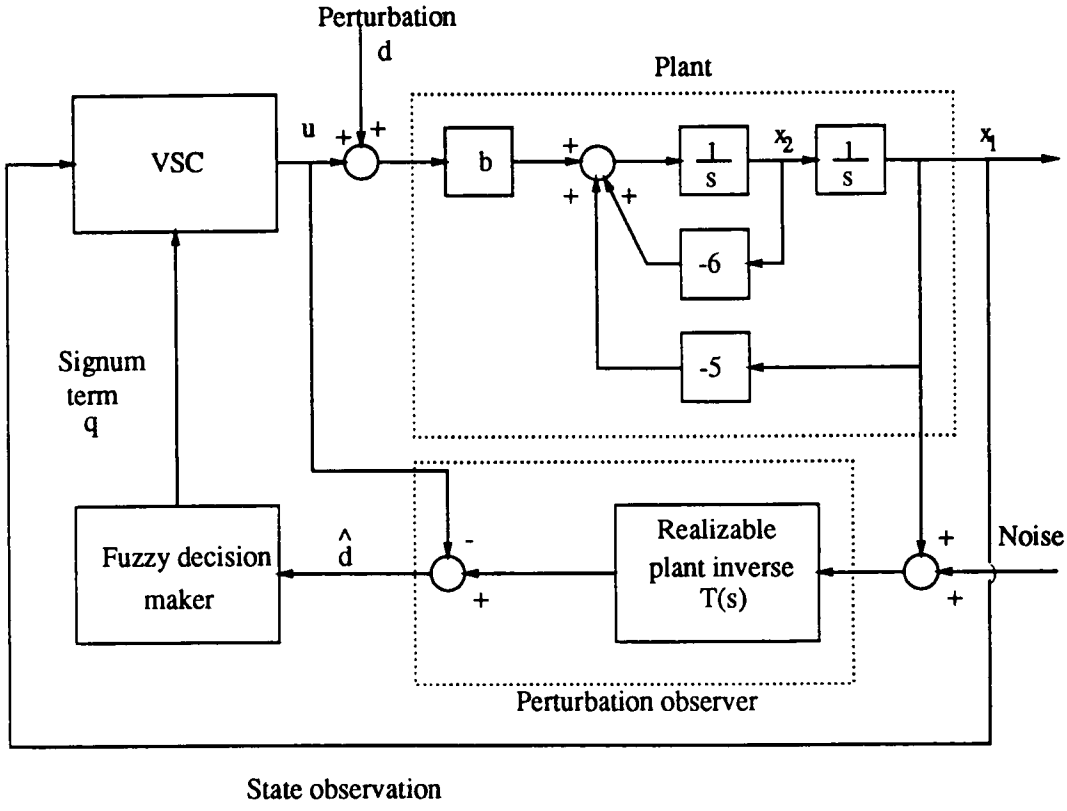


Figure 3.8: Simulation block diagram for fuzzy adaptive VSC in sliding mode only

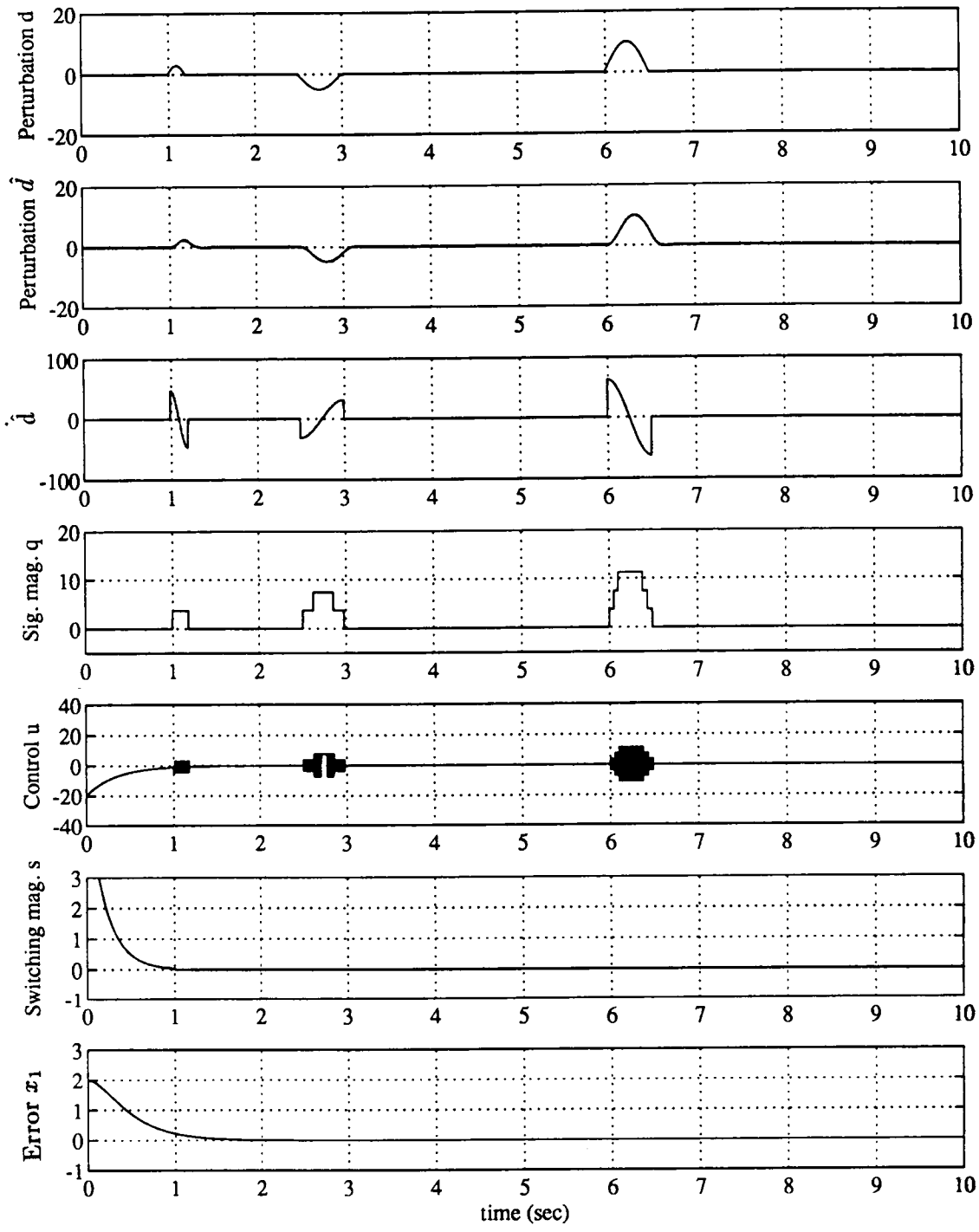


Figure 3.9: Fuzzy adaptive VSC in sliding mode only, case 1

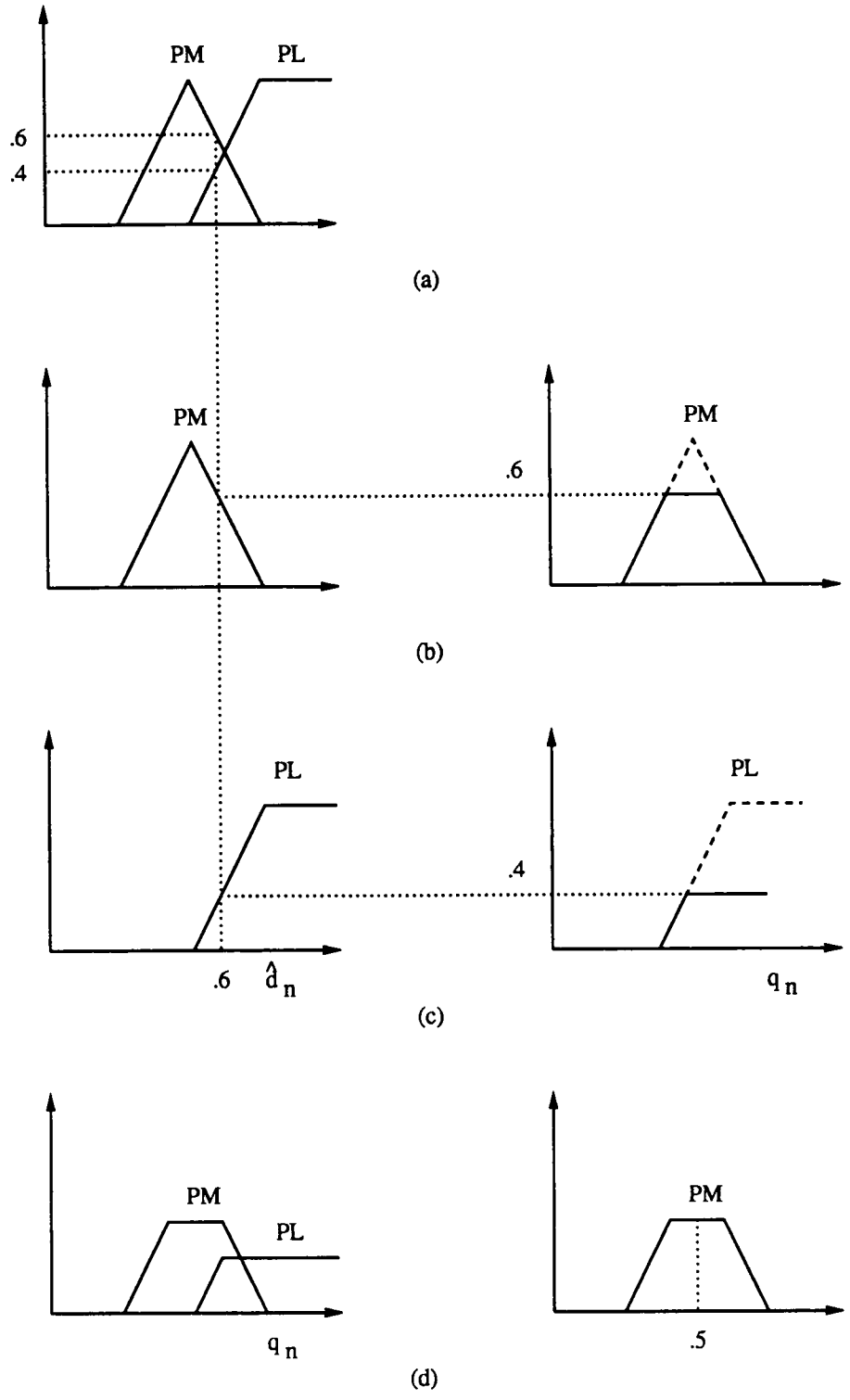


Figure 3.10: Example demonstrates a single input fuzzy logic program procedure

Table 3.1: Rules for a single input fuzzy system

$\hat{d}$	NL	NM	NS	ZR	PS	PM	PL
q	NL	NM	NS	ZR	PS	PM	PL

the set PM. The value of fuzzy q_n is .5 (Figure 3.10.d). Hence, scaling the fuzzy q_n will result in a numerical value. In this program the scaling factor for q_n was chosen to be 15. Therefore, q is 7.5 which is greater than $\hat{d}$ and the reaching condition is satisfied. A gain, the result is a robust control system. The results of this case is shown in Figure 3.11 where no time delays were modeled into the switching mechanisim.

Case 3 is the same as case 2 except a delay of .005 second is added. The results of the simulation are shown in Figure 3.12. As a result, in case 2, there were no chattering in the sliding mode, this is the ideal case (no delays are present). On the other hand, in case 3, there exists chattering in the sliding mode which is tolerable on the average. The chattering is not seen at the system output (x_1) becuase it is quite small. If the magnitude of the time delay were large, then the chattering would be large. The phase trajectories of case 2 and 3 are shown in Figure 3.13.

As a conclusion, in a fuzzy adaptive sliding mode, the signum term has a magnitude when a perturbation is present. Thus, the chattering is reduced on the average compared to the chattering produced in a usual VSC which has a fixed signum term magnitude as was shown in case 3 of the usual VSC system.

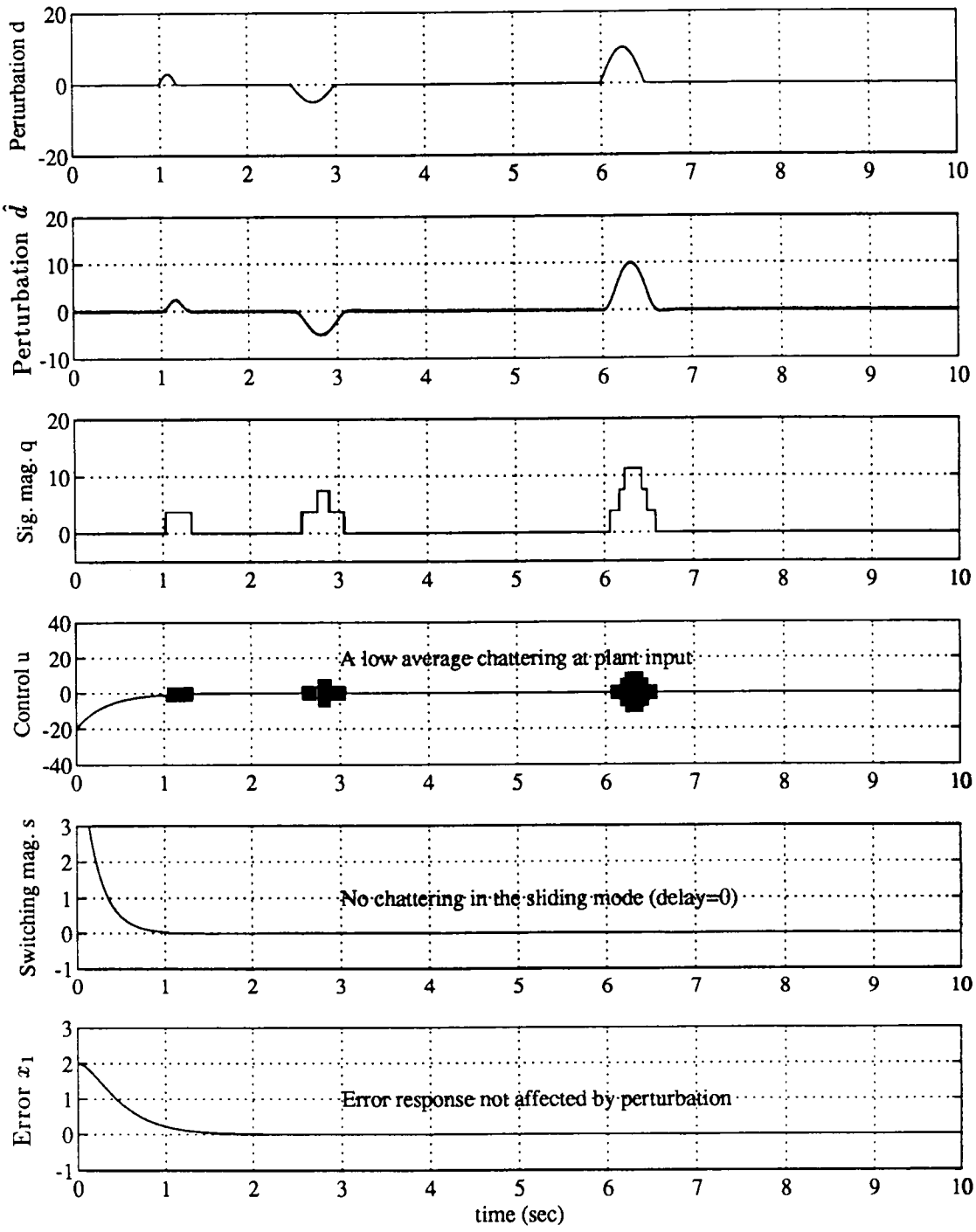


Figure 3.11: Fuzzy adaptive VSC in the sliding mode only, case 2

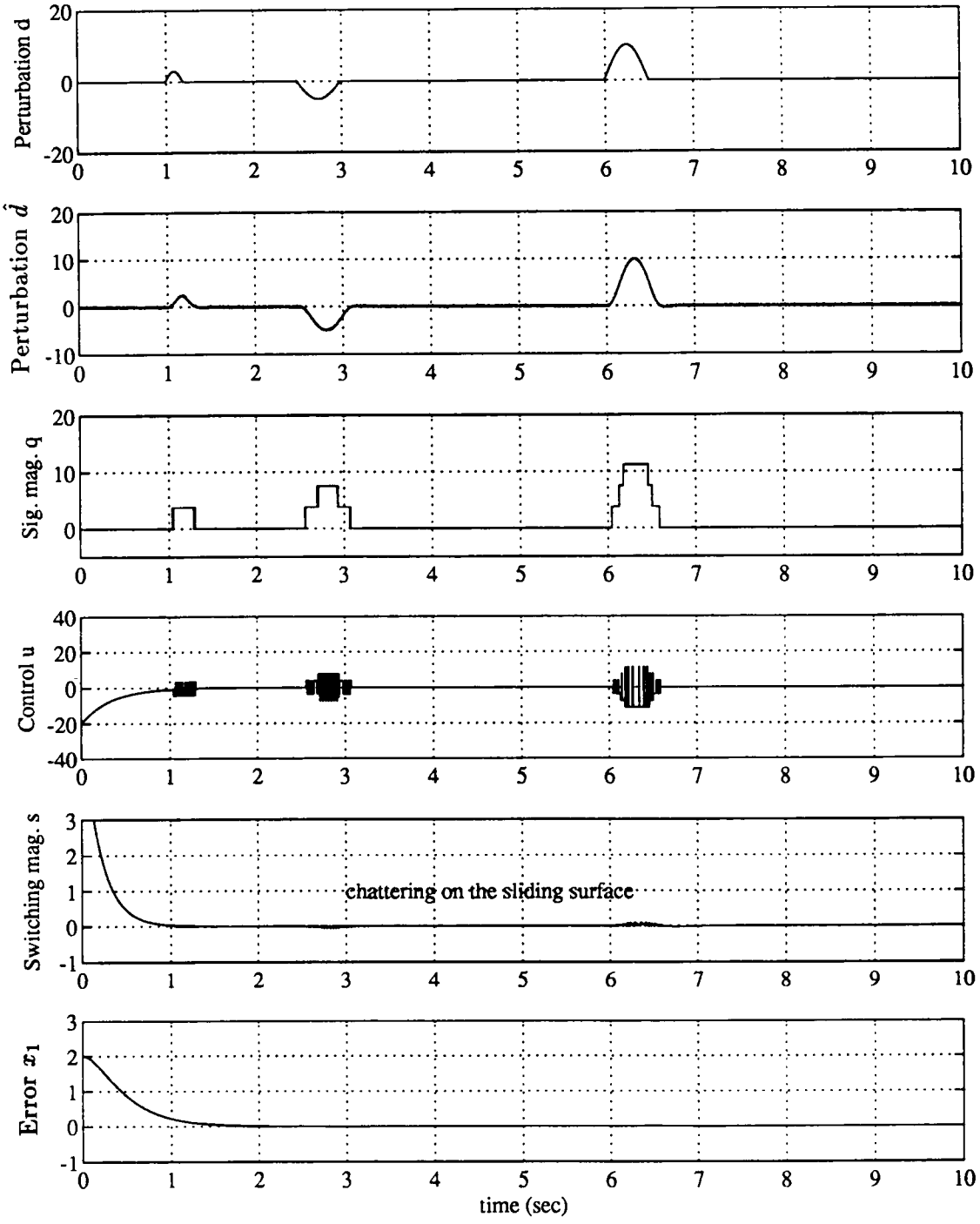
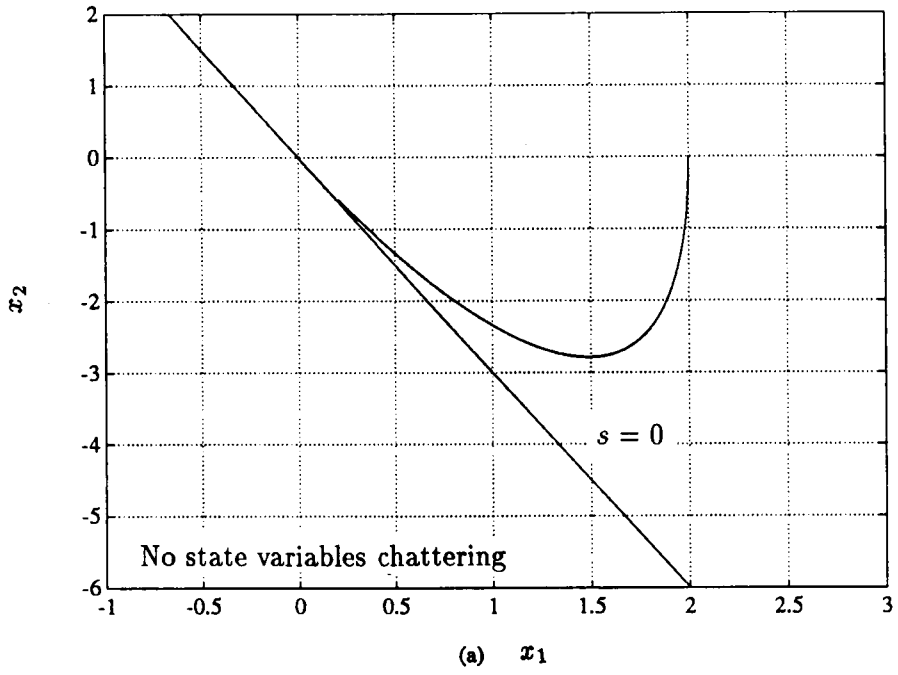


Figure 3.12: Fuzzy adaptive VSC in the sliding mode only, case 3



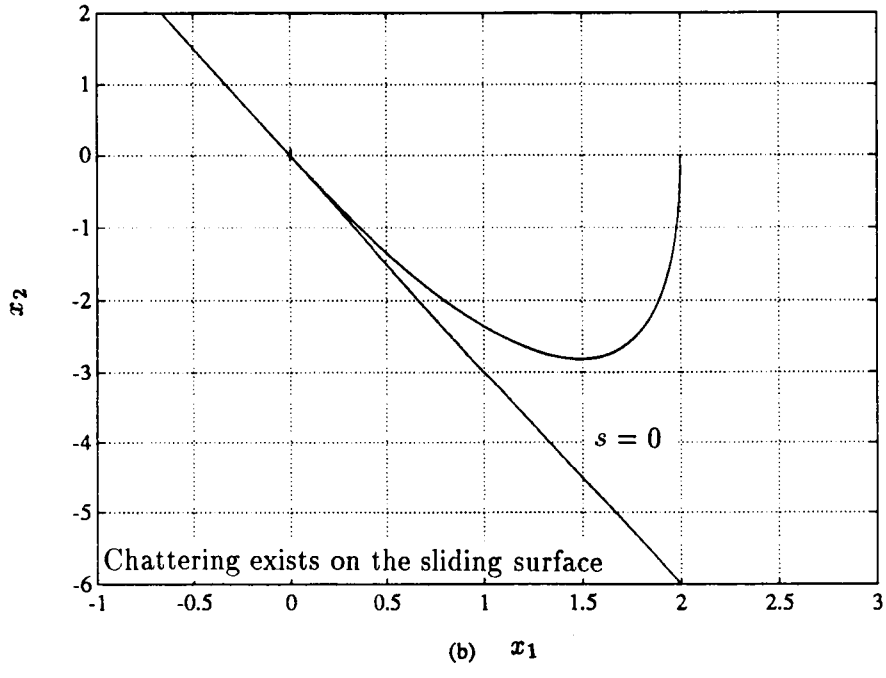


Figure 3.13: Phaze plane trajectory of (a) case 2, (b) case 3 of the fuzzy adaptive VSC in sliding mode

3.4.2 Fuzzy Adaptive VSC in Reaching Mode only

In this study of simulation, the results of designing fuzzy adaptive reaching mode are shown. Three simulation cases are presented with a delay of 0.001 second and no perturbations. Case 1 is to use a usual VSC with a signum term that has a small fixed value ($q = 1$). This value results in a very slow reaching (Figure 3.14). In case 2, q is given a large fixed value (50) which results in a fast reaching with chattering effects (Figure 3.15). The phase plane trajectory for case 1 and 2 are shown in Figure 3.16 (notice there exists chattering on the sliding surface in Figure 3.16.b).

In case 3, an adaptive signum term is used in the reaching mode. The design of the fuzzy system is the same as the one presented in Chapter 2 (section 2.3) where a two input fuzzy system was implemented. In Figure 3.17, the membership functions of the fuzzy variables are shown. Based on the normalized fuzzy region of Figure 3.17.c and the rule table presented in section in 2.3 (Table 2.2), a program is written and is listed in Appendix C. In the program, the defuzzification process was implemented using the center of gravity technique, and the scaling factor used for the value of q was chosen to be 100 to achieve a maximum value of 50 (100 x 0.5) when the state is far from the sliding surface. The simulation block diagram is shown in Figure 3.18. As a result, the chattering is reduced (see Figure 3.19). Phase plane trajectory of the above case is shown in Figure 3.20 (notice the chattering has disappeared even though the reaching is almost as fast as the one in case 2).

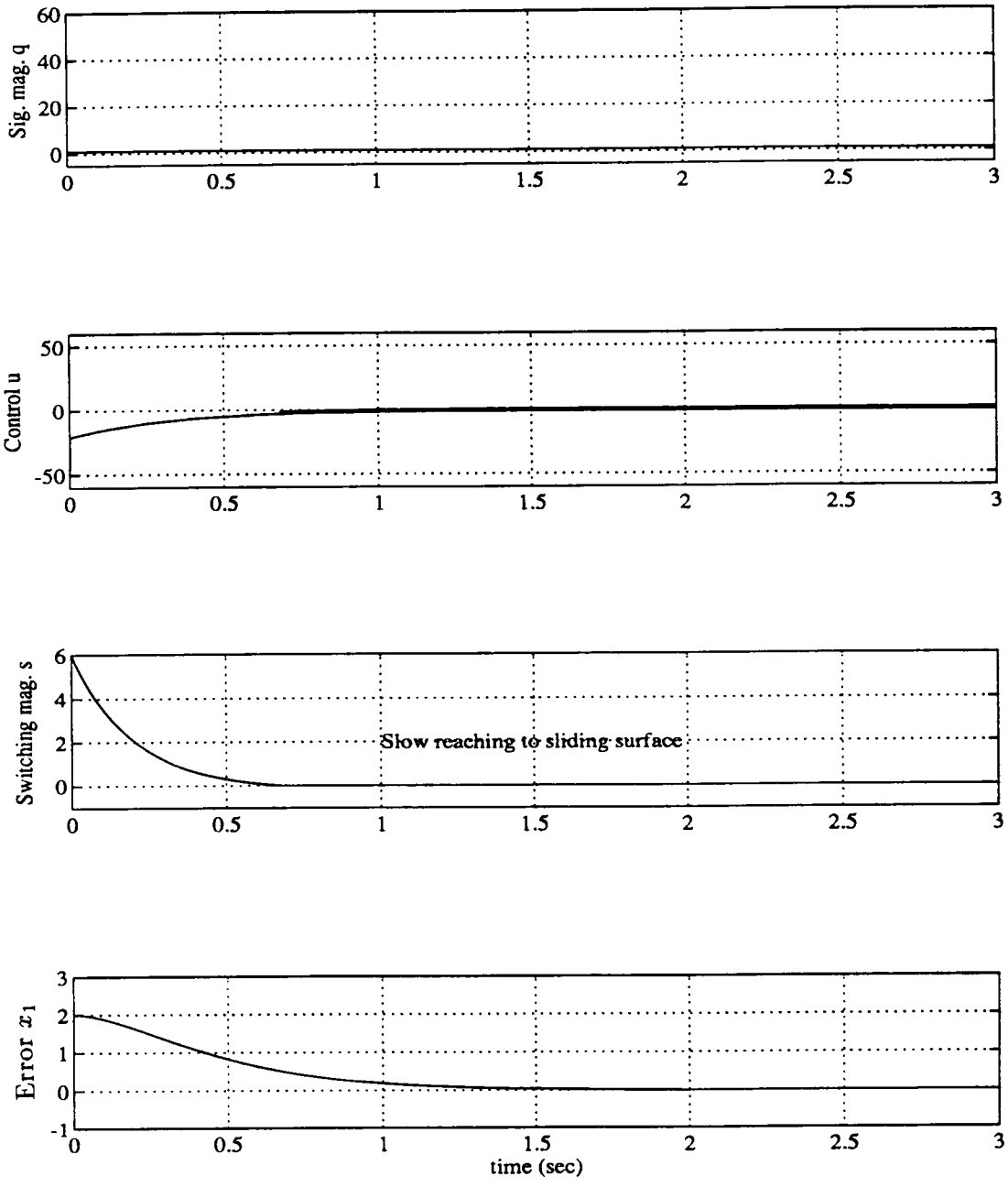


Figure 3.14: Response for VSC, case 1 ($q = 1$)

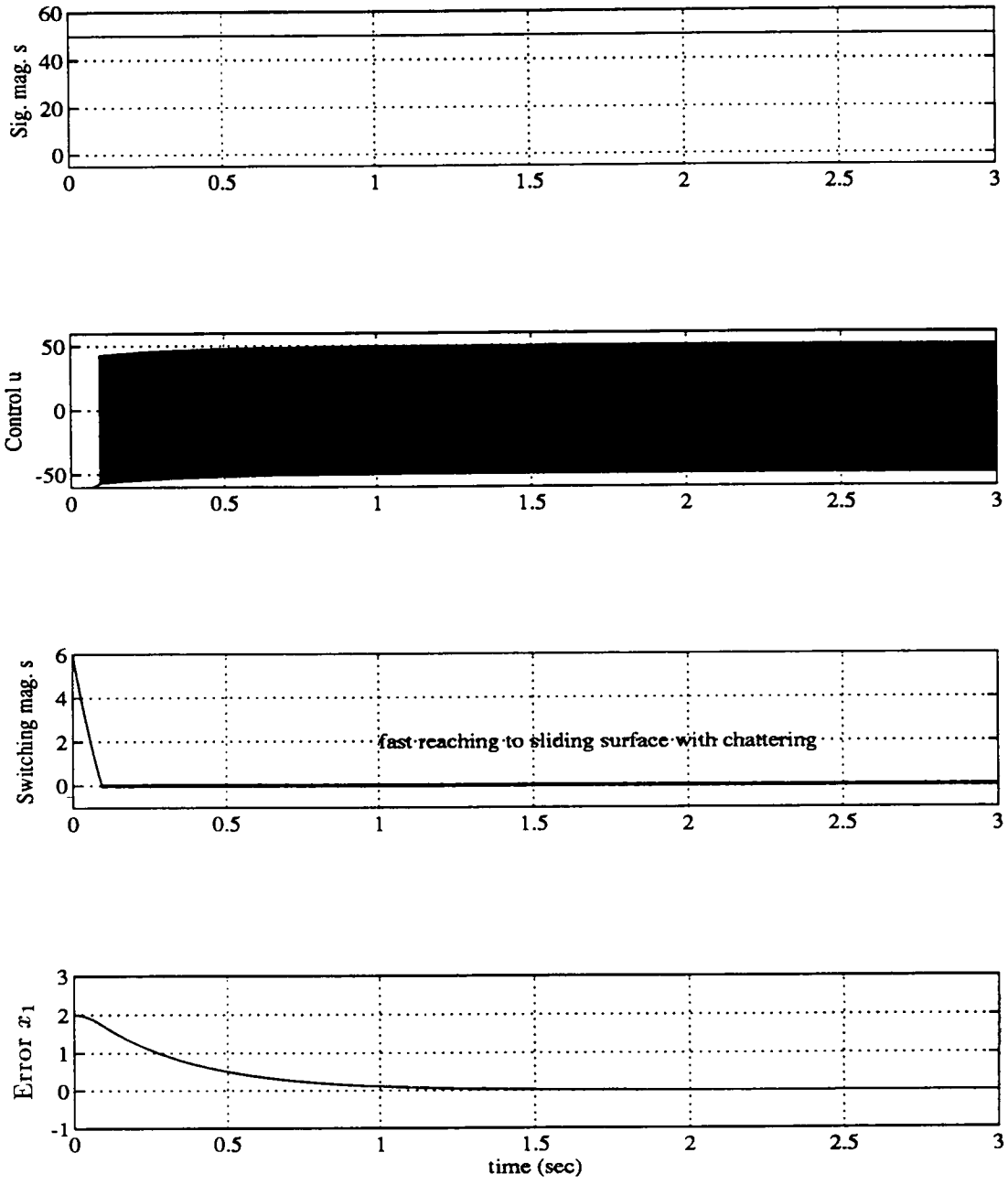
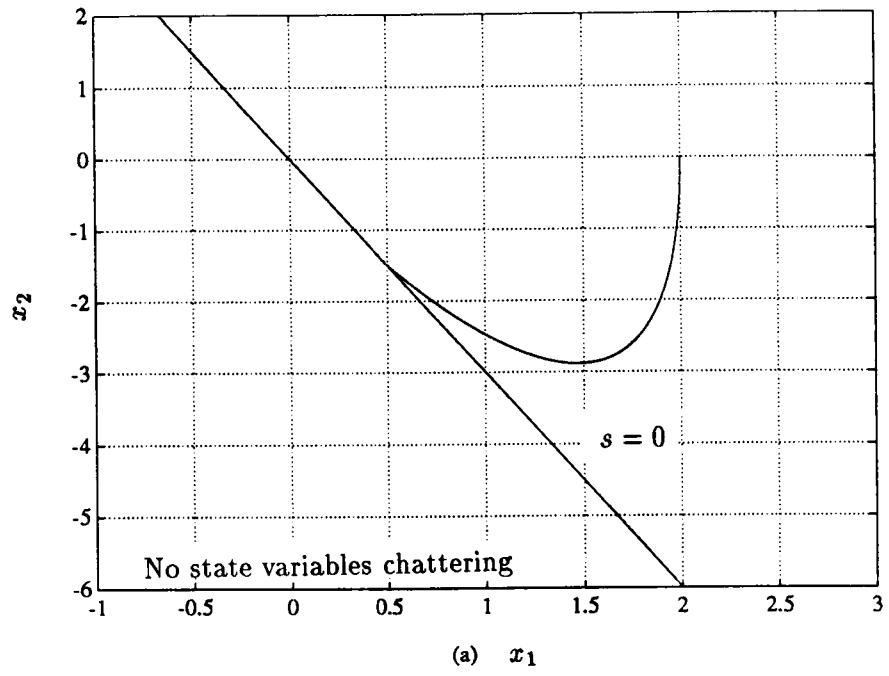


Figure 3.15: Response for VSC, case 2 ($q = 50$)



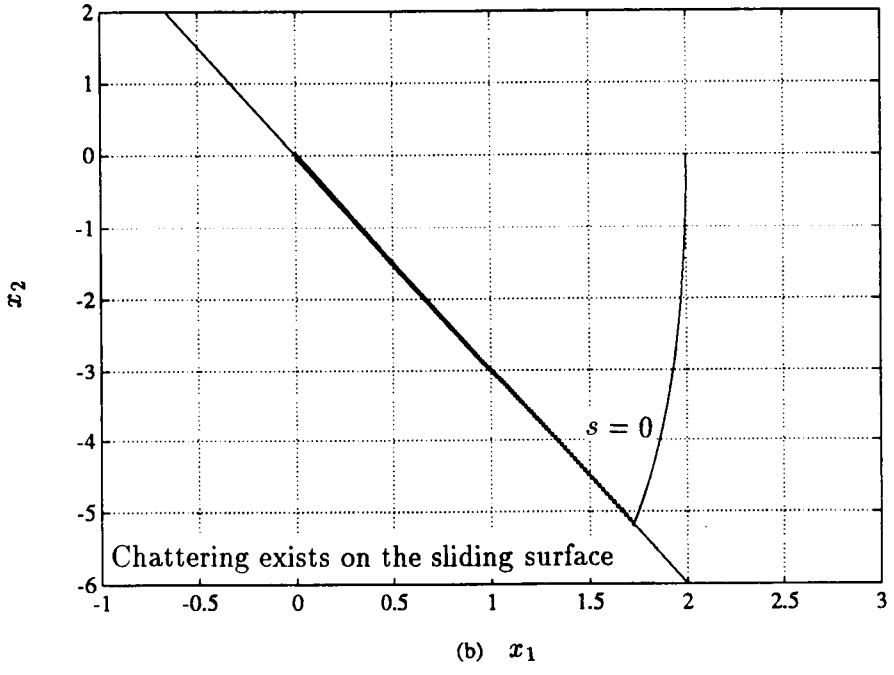


Figure 3.16: Phaze plane trajectory (a) case 1, (b) case 2

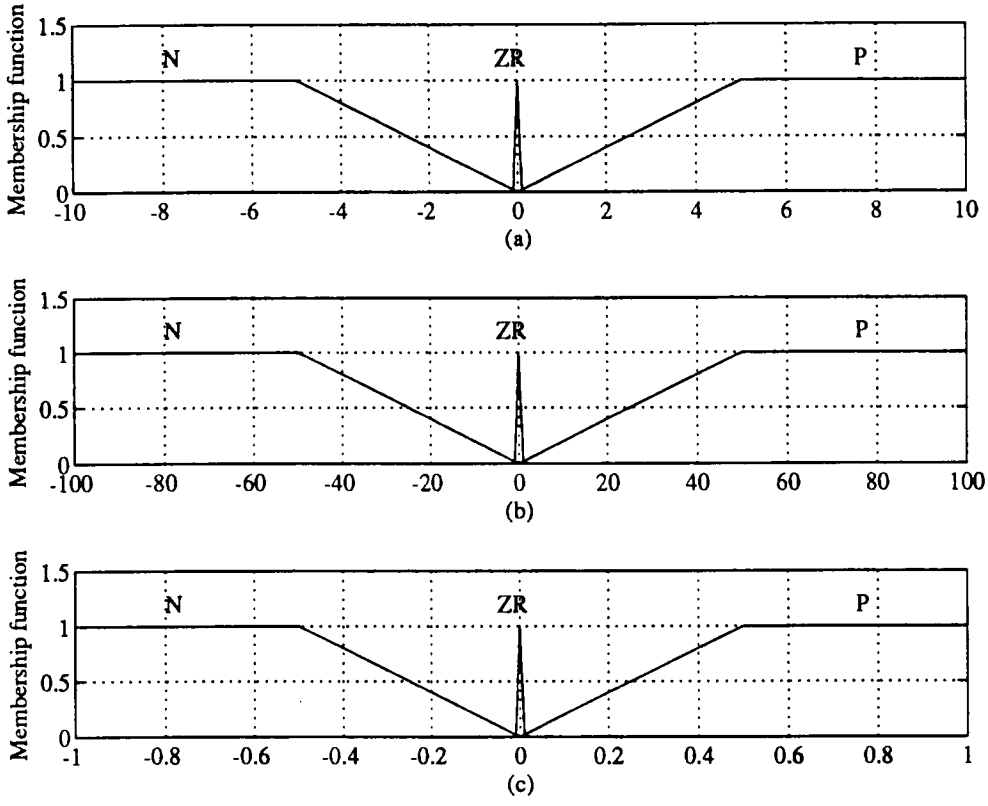


Figure 3.17: (a) s, (b) $\dot{s}$, and (c) normalized fuzzy region

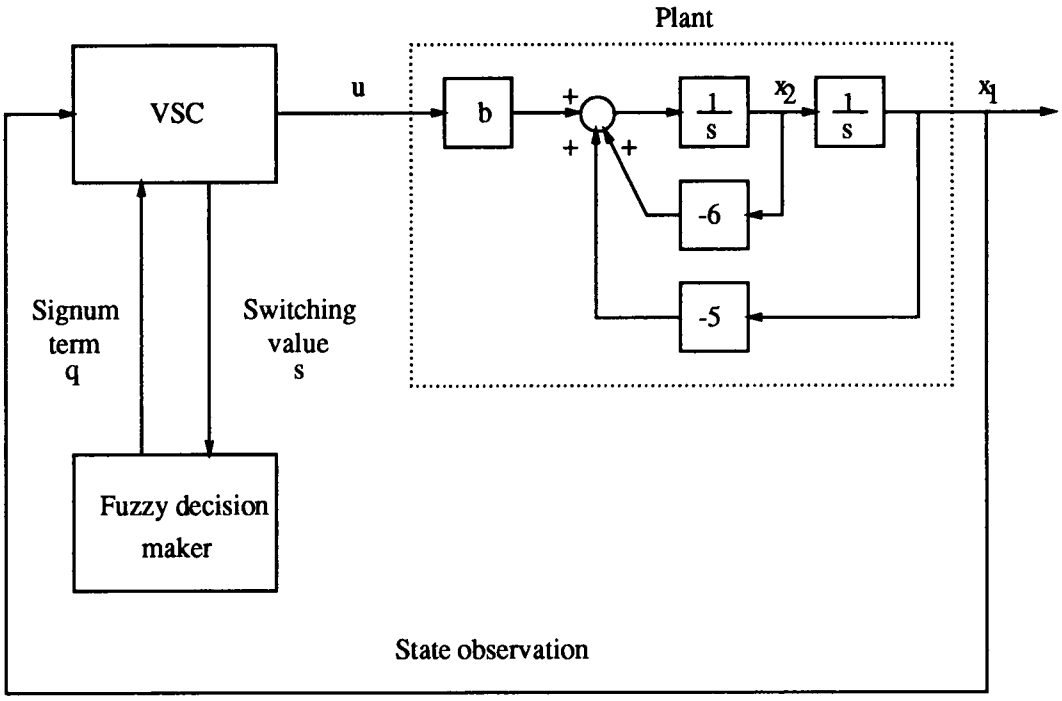


Figure 3.18: Simulation block diagram for a fuzzy adaptive VSC in reaching mode only

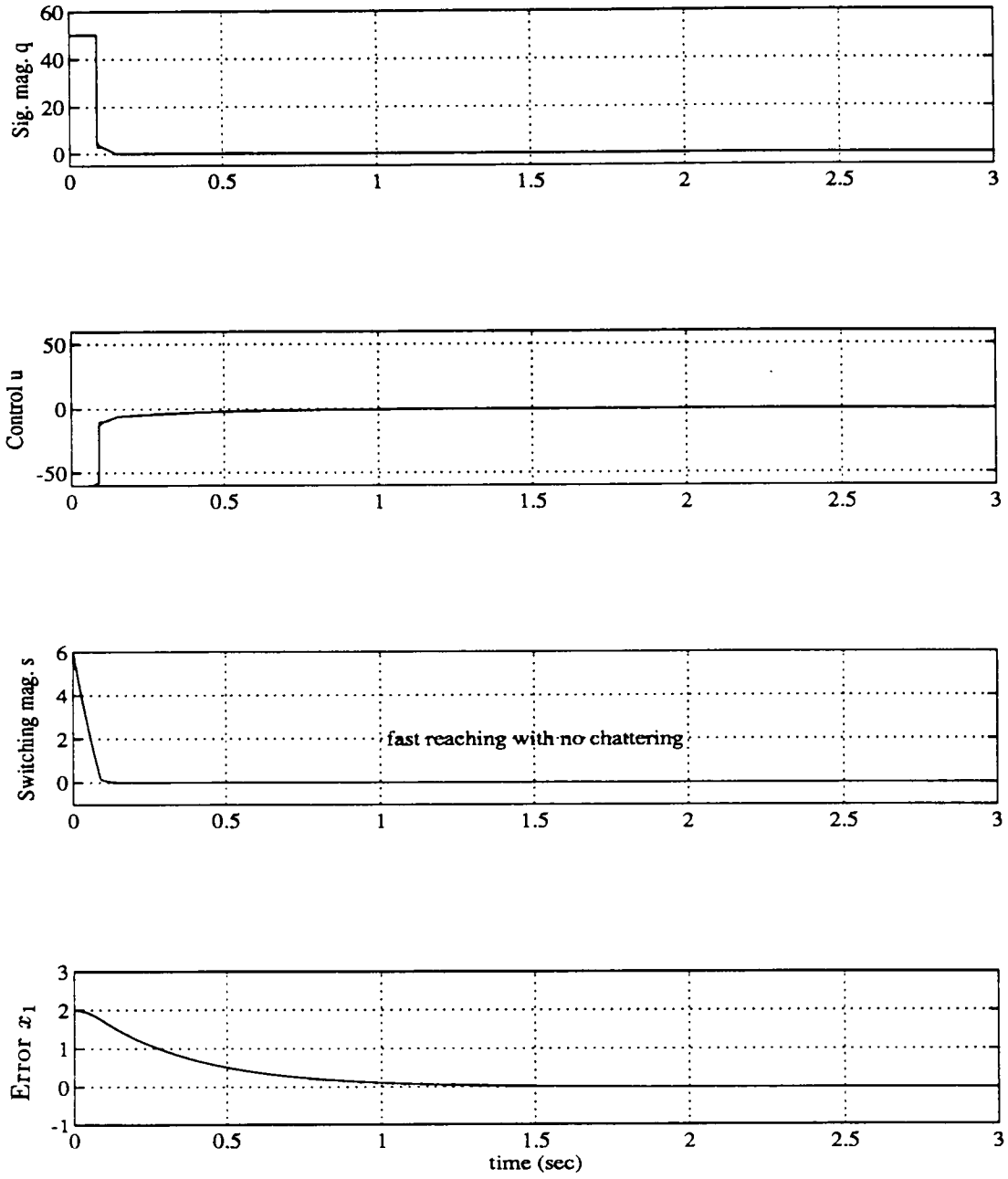


Figure 3.19: Fuzzy adaptive VSC in reaching mode only, case 3

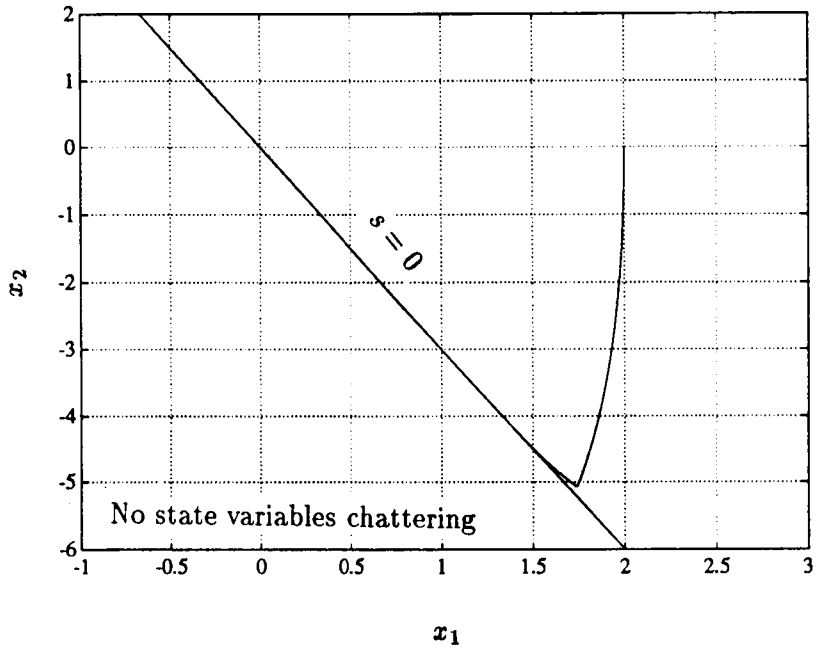


Figure 3.20: Phaze plane trajectory for case 3

3.4.3 Fuzzy Adaptive VSC in Both Sliding Mode and Reaching Mode

In this study, the methods of the above two studies are combined into one design method. A time delay of 0.005 second is used. The simulation block diagram is shown in Figure 3.21 and the results are shown in Figure 3.22. The phase plane trajectory is shown in Figure 3.23. The results can be compared to Figures 3.12 and 3.13. As can be noticed, a fast robust VSC system with minimum chattering effects is obtained.

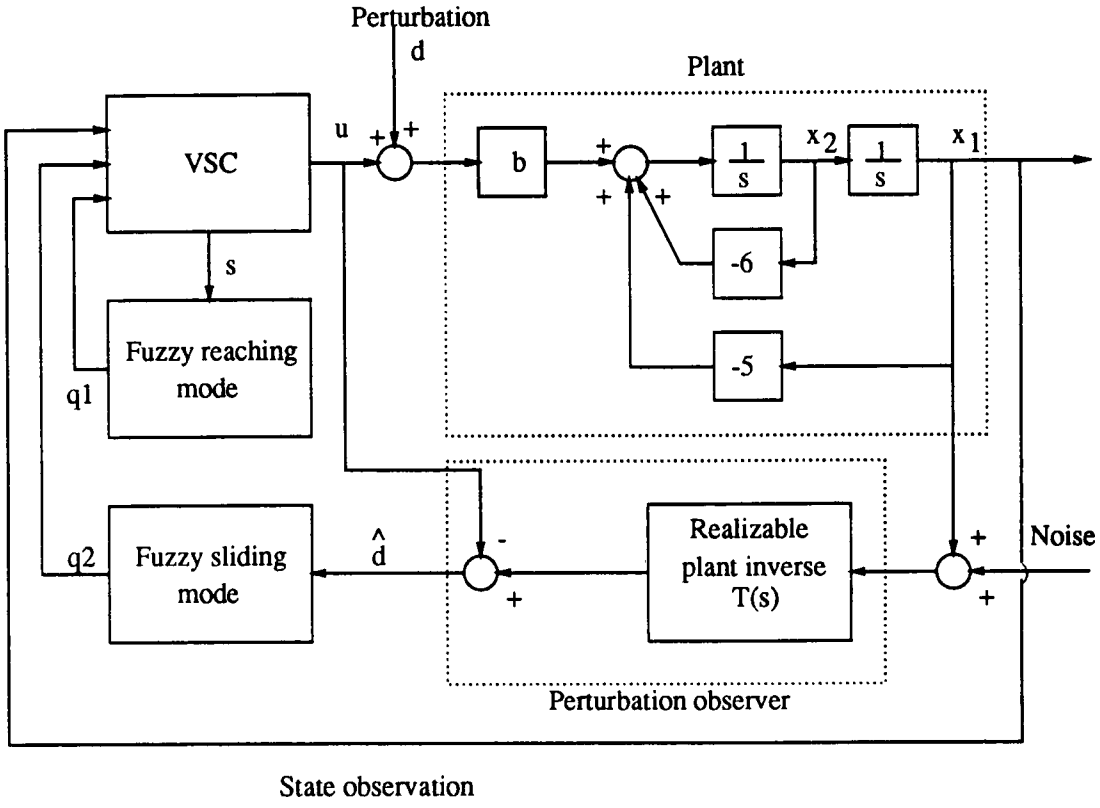


Figure 3.21: Simulation block diagram for a fuzzy adaptive VSC in both sliding mode and reaching mode

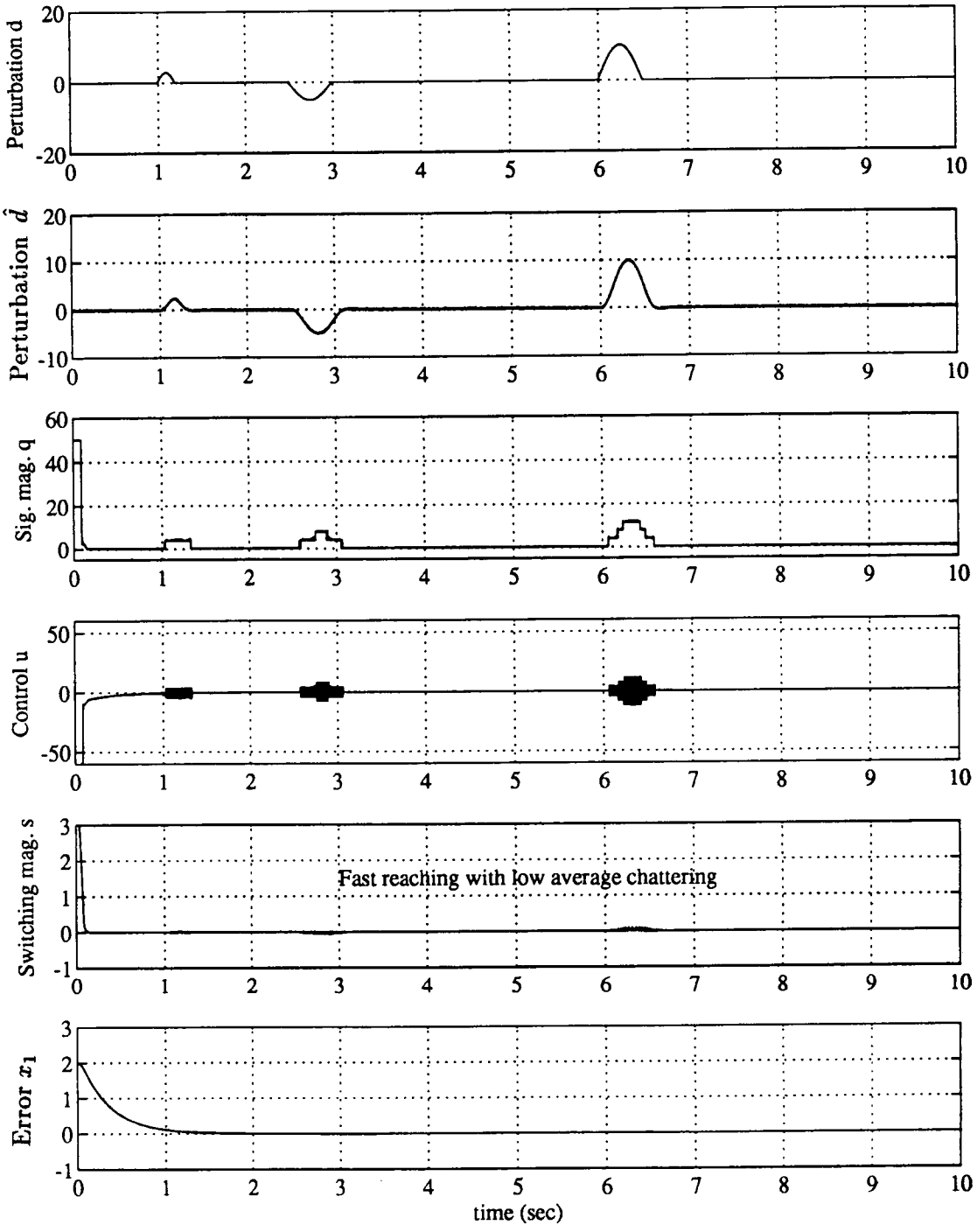


Figure 3.22: A fuzzy adaptive VSC in both sliding mode and reaching mode

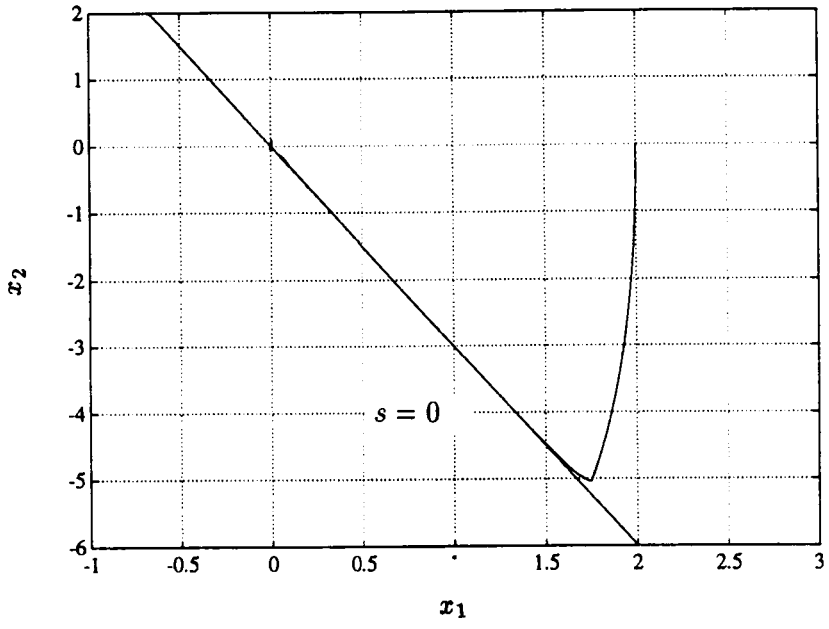


Figure 3.23: Phaze plane trajectory of fuzzy adaptive VSC in both sliding mode and reaching mode

CHAPTER 4

SUMMARY

In this study, the theory of VSC was utilized to design robust control systems. To reduce the effect of chattering in a VSC system, two fuzzy adaptive algorithms were used. One algorithm was used to design an adaptive robust sliding mode by the on-line adjustment of the magnitude of the signum term so that the reaching condition is satisfied at all times. The other was used to control the speed of the state in the reaching mode to reduce chattering by adjusting the magnitude of the signum term. The result was a low chattering robust VSC system.

In the first part of the study, the design of sliding and reaching modes was presented. A VSC law was found using the reaching law technique. It was explained that the reaching law specifies the dynamics of the switching function, and hence, it controls the dynamics of the system in the reaching mode. This control has two separate terms: a linear feedback term and a signum term to neutralize the effects of perturbation. It was also shown that a large magnitude of a signum term increases the amplitude of chatter which exits in VSC systems with time delays in their switching mechanisms. In order to have a VSC system that is robust and has a low chattering effect, a fuzzy adaptive approach was suggested.

In the second part, the suggested approach was discussed. The approach considered was to design a fuzzy adaptive VSC. This system included four subsystem:

the plant, a VSC, a fuzzy logic decision maker, and a perturbation observer. Two fuzzy adaptive algorithms were used. One algorithm was used to design a fuzzy robust sliding mode (VSC on demand). This was implemented using a fuzzy logic system which recieves the estimated perturbation from a perturbation observer and sends a decision on the magnitude of the signum term to the VSC. A magnitude of the signum term is given such that the reaching condition is satisfied at all times. The result was a low chattering VSC on the average (the signum term is only utilized when perturbations are present in the system).

The second algorithm was used to design a fuzzy adaptive reaching mode to reduce the chattering problem. This also was implemented using a fuzzy logic system which recieves the value of the switching function from the VSC. Based on this value and its rate of change a command on the magnitude of the signum term is issued to the VSC. For example, when the magnitude of the switching is small (the state is about to reach the sliding surface), and its rate of change is fast (large momentum), then the value of the signum term should be made small to reduce the momentum. This would nearly force an ideal-reaching, and hence, lower the chattering magnitude.

The third part of the study was to combine the above two algorithms into one algrorithm. This was done by adding the magnitudes of the signum terms resulted form each of the two fuzzy adaptive algorithms used.

Computer simulations were performed to show the results of the fuzzy adaptive approach. A practical problem was given, and all the simulations were performed on the same problem. The result was a robust low chattering VSC system.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Y. Itkis, *Control Systems of Variable Structure*, New York: Wiley, 1976.

[2] V.A. Utkin, "Discontinuous control system: State of the art in theory and applications," in *Proc. 10th IFAC World Congr. on Automatic Contr.*, vol. 1, pp. 25-44, 1987.

[3] V.A. Utkin, "Variable structure systems with sliding mode," *IEEE Trans. Automat. Contr.*, vol. AC-22, no. 2, pp. 212-222, 1977.

[4] V.A. Utkin and K.-K. D. Young, "Method for constructing discontinuity planes in multidimensional variable structure systems," *Automation and Remote Contr.*, vol. 39, no. 10, pp. 1466-1470, 1978.

[5] J.Y. Hung, W.B. Gao, and J.C. Hung " Variable Structure Control: A survey," *IEEE Trans. on Ind. Electron.*, vol. 40, no.1, pp. 2-22, 1993.

[6] W.B. Gao, and J.C. Hung " Variable Structure control of nonlinear systems: A new approach," *IEEE Trans. on Ind. Electron.*, vol. 40, no.1, pp. 45-55, 1993.

[7] B. Drazenovic, "The invariance conditions in variable structure systems" *Automatica*, vol. 15, pp. 287-295, 1969.

[8] I. Flugge-Lotz, *Discontinuous Automatic Control.* Princeton, NJ: Princeton University Press, N.I.

[9] H. G. Kwatny and T. L. Siu, "Chattering in variable structure feedback systems," in *Proc. IFAC 10th World Congr.*, Vol. 8, PP. 307-314, 1987.

[10] J. -J. Slotine and W. Li, *Applied Nonlinear Control.* Englewood Cliffs, NJ: Prentice-Hall.

[11] J. -J. Slotine, " Sliding Control Design for Nonlinear Systems," *Int. J. Contr.*, vol. 40, no. 2, pp. 421- 434, 1984.

[12] L. A. Zadeh, "Fuzzy Sets," Inform. Contr., vol. 8, pp. 338-353, 1965.

APPENDICES

APPENDIX A

Listing of Program 1

```
%*****************************************************************
% Written by : Naser A. Hdieb.                                    *
% Date       : January/10/93.                                     *
%*****************************************************************

%*****************************************************************
% This program is written to implement a fuzzy logic system      *
% for fuzzy adaptive VSC in sliding mode only. The system has     *
% two inputs. The inputs are the estimated perturbation           *
% ( call it d ) and its derivative ( call it dd ). The task of    *
% this program is to fuzzify these input values and find their    *
% membership functions, then based on the given fuzzy rules,      *
% it will construct a new fuzzy variable which is the solution    *
% variable ( call it qn ). A deffuzification process is used      *
% to produce a numerical value for the solution variable          *
% ( call this value q ). In matlab, this program is called        *
% a user built in Matlab function. It will be used in matlab      *
% simulation block diagram ( Simulab ) as a block that inputs     *
% the variables (  d & dd ) and outputs a value of the signum     *
% term ( q ). The name of this function will be called            *
% 2_iput_fuzzy.                                                   *
%*****************************************************************

function q = 2_input_fuzzy(u)
                                % this is how one can build his
                                % own matlab function. This
                                % will read the two inputs.

d=u(1)/10 ;                     % Normalize the input  d
dd=u(2)/100;                    % Normalize its derivative dd

%*****************************************************************
% (1) A fuzzy region which presents the fuzzy variable d          *
%*****************************************************************

% A normalized fuzzy set which presents the values of d that
% are negative large ( NL ):
```

```
if d <= -.75
        mb_d1=1.0;          %The membership function of d in NL.
        NL1=1;              % A flag which indicates that d belongs
                            % to NL fuzzy set to be used in rules.
elseif (-.75 < d & d <=-.5)
        mb_d1=-4*d-2;
        NL1=1;
        else
                mb_d1=0.0;
                NL1=0;
end

% A normalized fuzzy set which presents the values of d that
% are negative Medium ( NM ):

if -.75 <= d & d <=-.5
        mb_d2=4*d+3;        % The membership function of d in NM.
        NM1=1;              % A flag which indicates that d belongs
                            % to NM fuzzy set to be used in rules.
elseif -.5 < d & d <=-.25
        mb_d2=-4*d-1;
        NM1=1;
        else
                mb_d2=0.0;
                NM1=0;
end

% fuzzy set ( NS ):

if -.5 <= d & d <= -.25  % Same comments as the above sets
        mb_d3=4*d+2;
        NS1=1;
elseif -.25 < d  & d <=0
        mb_d3=-4*d;
        NS1=1;
        else
                mb_d3=0.0;
                NS1=0;
end

% fuzzy set ( ZR ):
```

```
if -.05<= d & d < 0
        mb_d4=20*d+1;
        ZR1=1;
elseif 0 <= d  & d <= .05
        mb_d4=-20*d+1;
        ZR1=1;
        else
                mb_d4=0.0;
                ZR1=0;
end
% fuzzy set ( PS ):
if 0 <= d & d <= .25
        mb_d5=4*d;
        PS1=1;
elseif .25 < d & d <=.5
        mb_d5=-4*d+2;
        PS1=1;
        else
                mb_d5=0.0;
                PS1=0;
end
% fuzzy set ( PM ):
if .25 <= d & d <= .5
        mb_d6=4*d-1;
        PM1=1;
elseif .5 < d & d <= .75
        mb_d6=-4*d+3;
        PM1=1;
        else
                mb_d6=0.0;
                PM1=0;
end
% fuzzy set ( PL )
if .5 <= d & d <= .75
        mb_d7=4*d-2;
        PL1=1;
elseif .75 < d
        mb_d7=1;
        PL1=1;
        else
                 mb_d7=0.0;
                PL1=0;
end
```

```
%*************************************************************
% (2) A fuzzy region which presents the fuzzy variable dd     *
%*************************************************************

% A fuzzy set which present value off dd that are negative and
% large ( NL ):

if dd <= -.75
        mb_dd1=1.0;         % The membership function of dd
        NL2=1;              % A flag which indicates  dd belongs to
                            % the fuzzy set NL to be used in rules.
elseif  -.75 < dd & dd <=-.5
        mb_dd1=-4*dd-2;
        NL2=1;
        else
                mb_dd1=0.0;
                NL2=0;
end

% fuzzy set ( NM ):

if -.75 <= dd & dd <=-.5
        mb_dd2=4*dd+3;
        NM2=1;
elseif -.5 < dd & dd <=-.25
        mb_dd2=-4*dd-1;
        NM2=1;
        else
                mb_dd2=0.0;
                NM2=0;
end

% fuzzy det ( NS ):

if -.5 <= dd & dd <= -.25
        mb_dd3=4*dd+2;
        NS2=1;
elseif -.25 < dd  & dd <=0
        mb_dd3=-4*dd;
        NS2=1;
        else
                mb_dd3=0.0;
                NS2=0;
end

% fuzzy det ( ZR ):
```

```
if -.05<= dd & dd < 0
        mb_dd4=20*dd+1;
        ZR2=1;
elseif 0 <= dd  & dd <= .05
        mb_dd4=-20*dd+1;
        ZR2=1;
        else
                mb_dd4=0.0;
                ZR2=0;
end

% fuzzy set ( PS ):

if 0 <= dd & dd <= .25
        mb_dd5=4*dd;
        PS2=1;
elseif .25 < dd & dd <=.5
        mb_dd5=-4*dd+2;
        PS2=1;
        else
                mb_dd5=0.0;
                PS2=0;
end

% fuzzy set ( PM ):

if .25 <= dd & dd <= .5
        mb_dd6=4*dd-1;
        PM2=1;
elseif .5 < dd & dd <= .75
        mb_dd6=-4*dd+3;
        PM2=1;
        else
                mb_dd6=0.0;
                PM2=0;
end

% fuzzy set ( PL ):

if .5 <= dd & dd <= .75
        mb_dd7=4*dd-2;
        PL2=1;
elseif .75 < dd
        mb_dd7=1;
        PL2=1;
        else
                 mb_dd7=0.0;
                 PL2=0;
end
```

```
%*************************************************************
% The construction of the solution variable ( Fuzzy signum    *
% term q_n ). The follwing rules are the base rules for the   *
% fuzzy system shown in Table 2.1.                            *
%*************************************************************

% The contribuations of all rules are stored in a vector called
% mb_q which has the values of the membership functions for all
% constructed q_n's, and all the q_n's are stored in another
% vector called q_n.

u=[-.75 -.5 -.25 0 .25 .5 .75];
                        % This is a vector which contains values
                        % of the Normalized fuzzy variables
                        % ( Universe of Discourse )

%*************************************************************
%                          rule # 1                           *
%*************************************************************

if NL1 & NL2            % If d is NL and dd is NL then q_n is
                        % NL.
        mb_q(1)=min(mb_d1,mb_dd1);
                        % membership function of the newly
                        % constructed fuzzy variable.
else
        mb_q(1)=0;
end
q_n(1)=u(1);            % The normalized q_n ( NL ) which is the
                        % first element of the universe of
                        % discourse u, i.e., q_n(1)=-0.75.
%*************************************************************
%                          rule # 2                           *
%*************************************************************

if NL1 & NM2            % If d is NL and dd is NM then q_n is
                        % NL.
        mb_q(2)=min(mb_d1,mb_dd2);
else
        mb_q(2)=0;
end
q_n(2)=u(1);            % the second element of the vector q_n
                        %  is  also -0.75 ( NL ).
%*************************************************************
%                          rule # 3                           *
%*************************************************************
```

```
if NL1 & NS2                % If d is NL and dd is NS then q_n is
                            % NL.
        mb_q(3)=min(mb_d1,mb_dd3);
else
        mb_q(3)=0;
end
q_n(3)=u(1);

%****************************************************************
%                          rule # 4                             *
%****************************************************************

if NL1  & ZR2
        mb_q(4)=min(mb_d1,mb_dd4);
else
        mb_q(4)=0;

end
q_n(4)=u(1);

%****************************************************************
%                          rule # 5                             *
%****************************************************************

if NL1 & PS2
        mb_q(5)=min(mb_d1,mb_dd5);
else
        mb_q(5)=0;
end
q_n(5)=u(1);

%****************************************************************
%                          rule # 6                             *
%****************************************************************

if NL1 & PM2
        mb_q(6)=min(mb_d1,mb_dd6);
else
        mb_q(6)=0;
end
q_n(6)=u(7);

%****************************************************************
%                          rule # 7                             *
%****************************************************************
```

```
if NL1 & PL2
        mb_q(7)=min(mb_d1,mb_dd7);
else
        mb_q(7)=0;
end
q_n(7)=u(2);

%*****************************************************************
%                          rule # 8                                *
%*****************************************************************

if NM1 & NL2
        mb_q(8)=min(mb_d2,mb_dd1);
else
        mb_q(8)=0;
end
q_n(8)=u(1);

%*****************************************************************
%                          rule # 9                                *
%*****************************************************************

if NM1 & NM2
        mb_q(9)=min(mb_d2,mb_dd2);
else
        mb_q(9)=0;
end
q_n(9)=u(1);

%*****************************************************************
%                          rule # 10                               *
%*****************************************************************

if NM1 & NS2
        mb_q(10)=min(mb_d2,mb_dd3);
else
        mb_q(10)=0;
end
q_n(10)=u(2);

%*****************************************************************
%                          rule # 11                               *
%*****************************************************************

if NM1 & ZR2
        mb_q(11)=min(mb_d2,mb_dd4);
else
        mb_q(11)=0;
```

```
end
q_n(11)=u(2);

%*****************************************************************
%                          rule # 12                              *
%*****************************************************************

if NM1 & PS2
        mb_q(12)=min(mb_d2,mb_dd5);
else
        mb_q(12)=0;

end
q_n(12)=u(2);

%*****************************************************************
%                          rule # 13                              *
%*****************************************************************

if NM1 & PM2
        mb_q(13)=min(mb_d2,mb_dd6);
else
        mb_q(13)=0;
end
q_n(13)=u(3);

%*****************************************************************
%                          rule # 14                              *
%*****************************************************************

if NM1 & PL2
        mb_q(14)=min(mb_d2,mb_dd7);
else
        mb_q(14)=0;
end
q_n(14)=u(3);

%*****************************************************************
%                          rule # 15                              *
%*****************************************************************

if NS1 & NL2
        mb_q(15)=min(mb_d3,mb_dd1);
else
        mb_q(15)=0;
end
q_n(15)=u(2);
```

```
%*****************************************************************
%                          rule # 16                                *
%*****************************************************************

if NS1 & NM2
        mb_q(16)=min(mb_d3,mb_dd2);
else
        mb_q(16)=0;

end
q_n(16)=u(2);

%*****************************************************************
%                          rule # 17                                *
%*****************************************************************

if NS1 & NS2
        mb_q(17)=min(mb_d3,mb_dd3);
else
        mb_q(17)=0;
end
q_n(17)=u(3);

%*****************************************************************
%                          rule # 18                                *
%*****************************************************************

if NS1 & ZR2
        mb_q(18)=min(mb_d3,mb_dd4);
else
        mb_q(18)=0;

end
q_n(18)=u(3);

%*****************************************************************
%                          rule # 19                                *
%*****************************************************************

if NS1 & PS2
        mb_q(19)=min(mb_d3,mb_dd5);
else
        mb_q(19)=0;

end
q_n(19)=u(3);
```

```
%*****************************************************************
%                          rule # 20                              *
%*****************************************************************

if NS1 & PM2
        mb_q(20)=min(mb_d3,mb_dd6);
else
        mb_q(20)=0;
end
q_n(20)=u(4);

%*****************************************************************
%                          rule # 21                              *
%*****************************************************************

if NS1 & PL2
        mb_q(21)=min(mb_d3,mb_dd7);
else
        mb_q(21)=0;
end
q_n(21)=u(4);

%*****************************************************************
%                          rule # 22                              *
%*****************************************************************

if ZR1 & NL2
        mb_q(22)=min(mb_d4,mb_dd1);
else
        mb_q(22)=0;
end
q_n(22)=u(3);

%*****************************************************************
%                          rule # 23                              *
%*****************************************************************

if ZR1 & NM2
        mb_q(23)=min(mb_d4,mb_dd2);
else
        mb_q(23)=0;

end
q_n(23)=u(3);
```

```
%****************************************************************
%                          rule # 24                             *
%****************************************************************

if ZR1 & NS2
        mb_q(24)=min(mb_d4,mb_dd3);
else
        mb_q(24)=0;

end
q_n(24)=u(4);

%****************************************************************
%                          rule # 25                             *
%****************************************************************

if ZR1 & ZR2
        mb_q(25)=min(mb_d4,mb_dd4);
else
        mb_q(25)=0;

end
q_n(25)=u(4);

%****************************************************************
%                          rule # 26                             *
%****************************************************************

if ZR1 & PS2
        mb_q(26)=min(mb_d4,mb_dd5);
else
        mb_q(26)=0;

end
q_n(26)=u(4);

%****************************************************************
%                          rule # 27                             *
%****************************************************************

if ZR1 & PM2
        mb_q(27)=min(mb_d4,mb_dd6);
else
        mb_q(27)=0;
end
q_n(27)=u(5);
```

```
%*****************************************************************
%                          rule # 28                              *
%*****************************************************************

if ZR1 & PL2
        mb_q(28)=min(mb_d4,mb_dd7);
else
        mb_q(28)=0;
end
q_n(28)=u(5);

%*****************************************************************
%                          rule # 29                              *
%*****************************************************************

if PS1 & NL2
        mb_q(29)=min(mb_d5,mb_dd1);
else
        mb_q(29)=0;
end
q_n(29)=u(4);

%*****************************************************************
%                          rule # 30                              *
%*****************************************************************

if PS1 & NM2
        mb_q(30)=min(mb_d5,mb_dd2);
else
        mb_q(30)=0;
end
q_n(30)=u(4);

%*****************************************************************
%                          rule # 31                              *
%*****************************************************************

if PS1 & NS2
        mb_q(31)=min(mb_d5,mb_dd3);
else
        mb_q(31)=0;

end
q_n(31)=u(5);
```

```
%*****************************************************************
%                          rule # 32                              *
%*****************************************************************

if PS1 & ZR2
        mb_q(32)=min(mb_d5,mb_dd4);
else
        mb_q(32)=0;

end
q_n(32)=u(5);

%*****************************************************************
%                          rule # 33                              *
%*****************************************************************

if PS1 & PS2
        mb_q(33)=min(mb_d5,mb_dd5);
else
        mb_q(33)=0;

end
q_n(33)=u(5);

%*****************************************************************
%                          rule # 34                              *
%*****************************************************************

if PS1 & PM2
        mb_q(34)=min(mb_d5,mb_dd6);
else
        mb_q(34)=0;
end
q_n(34)=u(6);

%*****************************************************************
%                          rule # 35                              *
%*****************************************************************

if PS1 & PL2
        mb_q(35)=min(mb_d5,mb_dd7);
else
        mb_q(35)=0;
end
q_n(35)=u(6);
```

```
%****************************************************************
%                          rule # 36                             *
%****************************************************************

if PM1 & NL2
        mb_q(36)=min(mb_d6,mb_dd1);
else
        mb_q(36)=0;

end
q_n(36)=u(5);

%****************************************************************
%                          rule # 37                             *
%****************************************************************

if PM1 & NM2
        mb_q(37)=min(mb_d6,mb_dd2);
else
        mb_q(37)=0;

end
q_n(37)=u(5);

%****************************************************************
%                          rule # 38                             *
%****************************************************************

if PM1 & NS2
        mb_q(38)=min(mb_d6,mb_dd3);
else
        mb_q(38)=0;
end
q_n(38)=u(6);

%****************************************************************
%                          rule # 39                             *
%****************************************************************

if PM1 & ZR2
        mb_q(39)=min(mb_d6,mb_dd4);
else
        mb_q(39)=0;

end
q_n(39)=u(6);
```

```
%****************************************************************
%                          rule # 40                             *
%****************************************************************

if PM1 & PS2
        mb_q(40)=min(mb_d6,mb_dd5);
else
        mb_q(40)=0;

end
q_n(40)=u(6);

%****************************************************************
%                          rule # 41                             *
%****************************************************************

if PM1 & PM2
        mb_q(41)=min(mb_d6,mb_dd6);
else
        mb_q(41)=0;
end
q_n(41)=u(7);

%****************************************************************
%                          rule # 42                             *
%****************************************************************

if PM1 & PL2
        mb_q(42)=min(mb_d6,mb_dd7);
else
        mb_q(42)=0;
end
q_n(42)=u(7);

%****************************************************************
%                          rule # 43                             *
%****************************************************************

if PL1 & NL2
        mb_q(43)=min(mb_d7,mb_dd1);
else
        mb_q(43)=0;

end
q_n(43)=u(6);
```

```
%****************************************************************
%                          rule # 44                             *
%****************************************************************
if PL1 & NM2
        mb_q(44)=min(mb_d7,mb_dd2);
else
        mb_q(44)=0;

end
q_n(44)=u(6);

%****************************************************************
%                          rule # 45                             *
%****************************************************************

if PL1 & NS2
        mb_q(45)=min(mb_d7,mb_dd3);
else
        mb_q(45)=0;

end
q_n(45)=u(7);

%****************************************************************
%                          rule # 46                             *
%****************************************************************

if PL1 & ZR2
        mb_q(46)=min(mb_d7,mb_dd4);
else
        mb_q(46)=0;

end
q_n(46)=u(7);

%****************************************************************
%                          rule # 47                             *
%****************************************************************

if PL1 & PS2
        mb_q(47)=min(mb_d7,mb_dd5);
else
        mb_q(47)=0;

end
q_n(47)=u(7);
```

```
%****************************************************************
%                          rule # 48                            *
%****************************************************************

if PL1 & PM2
        mb_q(48)=min(mb_d7,mb_dd6);
else
        mb_q(48)=0;
end
q_n(48)=u(7);

%****************************************************************
%                          rule # 49                            *
%****************************************************************

if PL1 & PL2
        mb_q(49)=min(mb_d7,mb_dd7);
else
        mb_q(49)=0;
end
q_n(49)=u(7);

%****************************************************************
% The defuzzification process using the Composite Maximum      *
% Method                                                        *
%****************************************************************

[mb_q,i]=sort(mb_q);    % sort is a Matlab function which sorts
                        % all the values in the vector mb_q in
                        % ascending order.  The maximum
                        % contribution becomes the last
                        % element of mb_q. The corresponding
                        %  index is also returned ( the last
                        % element of the vector i ) and subtituted
                        % in the vector q_n to find the
                        %  corresponding fuzzy value q_n.

q=15*q_n(i(49));        % Scaling the fuzzy q_n by 15.
```

APPENDIX B

Listing of Program 2

```
%*****************************************************************
% Written  by : Naser A. Hdieb                                    *
% Date        : January/13/93                                     *
%*****************************************************************

%*****************************************************************
% This program is another Matlab function that will eventually*
% be a block in a Matlab simulation diagram which  will take  *
% an input value of the estimated perturbation d and output a *
% numerical value of the signum q in the the VSC system. The  *
% objective is to design a fuzzy adaptive VSC in              *
% the sliding mode.                                           *
%*****************************************************************

function q = 1_input_fuzzy(d)
                        % this how one can build his own
                        % Matlab function. This will read
                        % one input which is d.
d=d/10;                 % Normalize the estimated perturbation d

%*****************************************************************
% Fuzzy regions which present the fuzzy variables             *
%*****************************************************************

% A fuzzy set which presents values of d which are negative
% and large ( NL ):

if d <= -.75
        mb_d1=1.0;      % The membership function for the fuzzy
                        % value d
        NL=1;           % A flag that indicates d is in the fuzzy
                        % set NL
elseif (-.75 < d & d <=-.5)
        mb_d1=-4*d-2;
        NL=1;
        else
                mb_d1=0.0;
                NL=0;
end
```

```
% fuzzy set ( NM ):

if -.75 <= d & d <=-.5 % Same as the comments above but for NM
        mb_d2=4*d+3;
        NM=1;
elseif -.5 < d & d <=-.25
        mb_d2=-4*d-1;
        NM=1;
        else
                mb_d2=0.0;
                NM=0;
end

% fuzzy set ( NS ):

if -.5 <= d & d <= -.25
        mb_d3=4*d+2;
        NS=1;
elseif -.25 < d  & d <=0
        mb_d3=-4*d;
        NS=1;
        else
                mb_d3=0.0;
                NS=0;
end

% fuzzy set ( ZR ):

if -.05< d & d < 0
        mb_d4=20*d+1;
        ZR=1;
elseif 0 <= d  & d < .05
        mb_d4=-20*d+1;
        ZR=1;
        else
                mb_d4=0.0;
                ZR=0;
end

% fuzzy set ( PS ):
```

```
if 0 <= d & d <= .25
        mb_d5=4*d;
        PS=1;
elseif .25 < d & d <=.5
        mb_d5=-4*d+2;
        PS=1;
        else
                mb_d5=0.0;
                PS=0;
end

% fuzzy set (PM):

if .25 <= d & d <= .5
        mb_d6=4*d-1;
        PM=1;
elseif .5 < d & d <= .75
        mb_d6=-4*d+3;
        PM=1;
        else
                mb_d6=0.0;
                PM=0;
end

% fuzzy set ( PL ):

if .5 <= d & d <= .75
        mb_d7=4*d-2;
        PL=1;
elseif .75 < d
        mb_d7=1;
        PL=1;
        else
                mb_d7=0.0;
                PL=0;
end

%*****************************************************************
% Fuzzy rules to implement the fuzzy system                      *
%*****************************************************************

q_n=[-.75 -.5 -.25 0 .25 .5 .75]; % the universe of discourse
                                  %  of solution variable q_n

% The contributions of all the rules are saved in a vector mb_q
% that has the membership functions of all the constructed fuzzy
% values q_n's.
```

```
%*****************************************************************
%                              rule # 1                           *
%*****************************************************************

if (NL)
        mb_q(1)=mb_d1;
end

%*****************************************************************
%                              rule # 2                           *
%*****************************************************************

if (NM)
        mb_q(2)=mb_d2;
end

%*****************************************************************
%                              rule # 3                           *
%*****************************************************************

if (NS)
        mb_q(3)=mb_d3;
end

%*****************************************************************
%                              rule # 4                           *
%*****************************************************************

if (ZR)
        mb_q(4)=mb_d4;
end

%*****************************************************************
%                              rule # 5                           *
%*****************************************************************

if (PS)
        mb_q(5)=mb_d5;
end

%*****************************************************************
%                              rule # 6                           *
%*****************************************************************

if (PM)
        mb_q(6)=mb_d6;
end
```

```
%*****************************************************************
%                                  rule # 7                       *
%*****************************************************************

if (PL)
        mb_q(7)=mb_d7;
end

%*****************************************************************
% Defuzzification of q_n using the Composite Maximum method     *
%*****************************************************************

[i,mb_q]=sort(mb_q);     % Find the maximum contribution using
                         %  the sort function.
q=15*q_n(i(7));          % Scale the fuzzy value q_n by 15.
```

APPENDIX C

Listing of Program 3

```
%*****************************************************************
% Written by : Naser A. Hdieb                                     *
% Date       : January/20/93                                      *
%*****************************************************************

%*****************************************************************
% This program is to  implement a fuzzy adaprive VSC in  the      *
% reaching mode  to reduce the problem of chattering. The         *
% inputs of the system the magnitude  of the switching value s    *
% and its derivative ds. The output is a numerical for the        *
% signum term q in the VSC system.                                *
%*****************************************************************

function q =fuzzy_reaching(u) % A user built in Matlab function.

s=u(1)/10;                    % Normalize the switching value s
ds=u(2)/100;                  % Normalize the derivative ds

%*****************************************************************
% A fuzzy region for switching values (s) which inludes the       *
% fuzzy sets: N, ZR, and P.                                       *
%*****************************************************************

%fuzzy set ( N ):

if ( s <= -.5)
        mb_s1=1.0;            % The membership function for the
                              % fuzzy value s.
        N1=1;                 % A flag which indicates that s is
                              % in fuzzy set N.
elseif -.5< s & s <=0
        mb_s1=-2*s;
        N1=1;
        else
                mb_s1=0;
                N1=0;
end

% fuzzy set ( ZR ):
```

```
if -.01<= s & s  <= 0
        mb_s2=100*s+1;
        ZR1=1;
elseif 0 < s  & s <= .01
        mb_s2=-100*s+1;
        ZR1=1;
        else
                mb_s2=0.0;
                ZR1=0;
end

% fuzzy set ( P ):

if 0 <= s & s <= .5
        mb_s3=2*s;
        P1=1;
elseif s > .5
        mb_s3=1;
        P1=1;
        else
                mb_s3=0.0;
                P1=0;
end

%*****************************************************************
% the second fuzzy region for derivative of switching (ds)      *
% which also has three fuzzy sets: N, ZR, and P.                *
%*****************************************************************

%fuzzy set ( N ):

if ( ds <= -.5)
        mb_ds1=1.0;
        N2=1;

elseif -.5< ds & ds <=0
        mb_ds1=-2*ds;
        N2=1;
        else
                mb_ds1=0;
                N2=0;
end

% fuzzy set ( ZR ):
```

```
if -.01 <= ds & ds  <= 0
        mb_ds2=100*ds+1;
        ZR2=1;
elseif  0 < ds  & ds <= .01
        mb_ds2=-100*ds+1;
        ZR2=1;
        else
                mb_ds3=0.0;
                ZR2=0;
end

% fuzzy set ( P ):

if 0 <= ds & ds <= .5
        mb_ds3=2*ds;
        P2=1;
elseif .5 < ds
        mb_ds3=1;
        P2=1;
        else
                mb_ds3=0.0;
                P2=0;
end

%*****************************************************************
% Fuzzy rules to implement the fuzzy system                      *
%*****************************************************************

u=[-.5  0 .5];          % the universe of discourse of
                        % of all fuzzy variables.

% The contributions of all rules are saved in a vector mb_q that
% has the membership functions of newly constructed q_n's, and
% all the q_n's are stored in another vector called q_n.

%*****************************************************************
%                          rule # 1                              *
%*****************************************************************

if N1 & N2
        mb_q(1)=min(mb_s1,mb_ds1);
else
        mb_q(1)=0;
end
q_n(1)=u(1);
```

```
%*****************************************************************
%                          rule # 2                                *
%*****************************************************************

if N1 & ZR2
        mb_q(2)=min(mb_s1,mb_ds2);
else
        mb_q(2)=0;
end
q_n(2)=u(1);

%*****************************************************************
%                          rule # 3                                *
%*****************************************************************

if N1 & P2
        mb_q(3)=min(mb_s1,mb_ds3);
else
        mb_q(3)=0;
end
q_n(3)=u(1);

%*****************************************************************
%                          rule # 4                                *
%*****************************************************************

if ZR1 & N2
        mb_q(4)=min(mb_s2,mb_ds1);
else
        mb_q(4)=0;
end
q_n(4)=u(2);

%*****************************************************************
%                          rule #5                                 *
%*****************************************************************

if ZR1 & ZR2
        mb_q(5)=min(mb_s2,mb_ds2);
else
        mb_q(5)=0;

end
q_n(5)=u(2);
%*****************************************************************
%                          rule # 6                                *
%*****************************************************************
```

```
if ZR1 & P2
        mb_q(6)=min(mb_s2,mb_ds3);
else
        mb_q(6)=0;

end
q_n(6)=u(2);
%*****************************************************************
%                          rule # 7                                *
%*****************************************************************

if P1 & N2
        mb_q(7)=min(mb_s3,mb_ds1);
else
        mb_q(7)=0;

end
q_n(7)=u(3);

%*****************************************************************
%                          rule # 8                                *
%*****************************************************************

if P1 & ZR2
        mb_q(8)=min(mb_s3,mb_ds2);
else
        mb_q(8)=0;

end
q_n(8)=u(3);

%*****************************************************************
%                          rule # 9                                *
%*****************************************************************

if P1 & P2
        mb_q(9)=min(mb_s3,mb_ds3);
else
        mb_q(9)=0;

end
q_n(9)=u(3);

%*****************************************************************
% the defuzzification process is done using the center of        *
% gravity technique                                              *
%*****************************************************************
```

```
% the center of gravity formula

for i=1:9
num(i)=mb_q(i)*q_n(i);
end
fuzzy_q=sum(num)/(sum(mb_q));  % The sum is a Matlab function
                               % adds the elements of its
                               % input vector.
q=0.1+abs(100*fuzzy_q);        % Scale q by a factor of 100
                               % and a small value ( 0.1 ) so
                               % a finite reaching is achieved.
```

VITA

Naser Hdieb was born in Amman, Jordan, on May 19, 1965. He attended elementary and secondary schools in Amman, Jordan, and graduated from El-husain High school in 1983. In Fall 1984, he entered the University of Tennessee at Knoxville where he received a Bachelor of Science Degree in Electrical and Computer Engineering in August, 1989. In Fall 1990, he enrolled in the Graduate School of the University of Tennessee, Knoxville where he received his Master of Science Degree in Electrical and Computer Engineering in May, 1993.