An Investigation of
Adaptive Fuzzy Sliding Mode Control for
Robotic Manipulators

by

Xiaosong Lu

A thesis submitted to
The Faculty of Graduate Studies and Research
in partial fulfilment
of the requirements for the degree of
Master of Applied Science in Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
Department of Systems and Computer Engineering

Carleton University
Ottawa, Ontario
May 9, 2007

©Copyright
2007, Xiaosong Lu
NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Abstract

This thesis examines five adaptive fuzzy sliding mode control algorithms and their suitability for use in the control of a two degree of freedom robotic manipulator. Lin and Hsu's algorithm and Shahnazi et al.'s algorithm are redesigned for robotic manipulators. Theoretical proof of the stability and the convergence of the overall system is presented using Lyapunov stability theorem for each algorithm. Simulations of the five algorithms, applied to a two degree of freedom robotic manipulator, are implemented and we have evaluated their efficacy in maintaining Lyapunov stability.
Acknowledgements

I would like to express my extreme gratitude and appreciation to my supervisor, Professor Howard M. Schwartz, for his generous guidance, encouragement and support. Without Professor Schwartz's help, it would not be possible to finish my thesis quickly and successfully.
Contents

1 Introduction
   1.1 Literature Overview ................................................................. 1
   1.2 Thesis Overview and Contributions ............................................ 2

2 Introduction to Sliding Mode Control ........................................... 4
   2.1 Why Sliding Mode Control? .......................................................... 4
   2.2 Classical Sliding Mode Control ..................................................... 5
   2.3 Model of Robotic Manipulators ..................................................... 11
   2.4 Classical Sliding Mode Control for Robotic Manipulators ............. 12
   2.5 Summary ...................................................................................... 21

3 Introduction to Fuzzy Control ......................................................... 22
   3.1 Description and Analysis of Fuzzy Logic Systems ......................... 22
      3.1.1 Basic Concepts ....................................................................... 22
      3.1.2 Fuzzy Systems and Their Properties ....................................... 31
   3.2 Adaptive Fuzzy Control ............................................................... 39
      3.2.1 Why Fuzzy Control? ............................................................... 39
      3.2.2 Why Adaptive Fuzzy Control? ............................................... 40
      3.2.3 Adaptive Fuzzy Control ......................................................... 41
3.3 Summary ................................................................. 42

4 Adaptive Fuzzy Sliding Mode Control ................................. 43
4.1 Description of Algorithms .............................................. 43
  4.1.1 Yoo and Ham’s Algorithm ........................................... 44
  4.1.2 Guo and Woo’s Algorithm .......................................... 46
  4.1.3 Revised Lin and Hsu’s Algorithm ................................. 47
  4.1.4 Revised Shahnazi et al.’s Algorithm ............................. 50
  4.1.5 Medhaffar et al.’s Algorithm ....................................... 51
4.2 Stability Proof of Algorithms .......................................... 53
  4.2.1 Stability Proof of Yoo and Ham’s Algorithm .................. 53
  4.2.2 Stability Proof of Guo and Woo’s Algorithm .................. 55
  4.2.3 Stability Proof of Revised Lin and Hsu’s Algorithm ........ 57
  4.2.4 Stability Proof of Revised Shahnazi et al.’s Algorithm ..... 59
  4.2.5 Stability Proof of Medhaffar et al.’s Algorithm ............... 61
4.3 Comparison of Algorithms ............................................. 62
4.4 Summary ................................................................. 64

5 Simulation Results ........................................................ 66
  5.1 Yoo and Ham’s Algorithm ............................................ 67
  5.2 Guo and Woo’s Algorithm ............................................. 72
  5.3 Revised Lin and Hsu’s Algorithm ................................... 75
  5.4 Revised Shahnazi et al.’s Algorithm ............................... 78
  5.5 Medhaffar et al.’s Algorithm ......................................... 80
  5.6 Stability Analysis of Algorithms in a Practical Sense ........ 83
  5.7 Summary ................................................................. 84

6 Conclusions and Future Work ........................................... 85
List of Figures

2.1 Sliding surface .................................................. 6
2.2 Derivation of bounds ........................................... 8
2.3 Boundary layer with second order system ............. 10
2.4 Control law in the boundary layer ....................... 10
2.5 Two-link robotic manipulator ............................. 11
2.6 (a) Tracking and (b) errors of joint 1 in Classical SMC. Dash line: desired trajectory; solid line: actual trajectory. 16
2.7 (a) Control input and (b) sliding surface of joint 1 in Classical SMC. 16
2.8 (a) Tracking and (b) errors of joint 2 in Classical SMC. Dash line: desired trajectory; solid line: actual trajectory. 17
2.9 (a) Control input and (b) sliding surface of joint 2 in Classical SMC. 17
2.10 (a) Tracking and (b) errors of joint 1 with $\dot{\hat{p}}_1 = 3$, $\dot{\hat{p}}_2 = 7$. Dash line: desired trajectory; solid line: actual trajectory. 18
2.11 (a) Tracking and (b) errors of joint 2 with $\dot{\hat{p}}_1 = 3$, $\dot{\hat{p}}_2 = 7$. Dash line: desired trajectory; solid line: actual trajectory. 18
2.12 (a) Tracking and (b) errors of joint 1 in Classical SMC by using saturation function. Dash line: desired trajectory; solid line: actual trajectory. 19
2.13 (a) Control input and (b) sliding surface of joint 1 in Classical SMC by using saturation function. 19

viii
2.14 (a) Tracking and (b) errors of joint 2 in Classical SMC by using saturation function. Dash line: desired trajectory; solid line: actual trajectory.

2.15 (a) Control input and (b) sliding surface of joint 2 in Classical SMC by using saturation function.

3.1 Membership functions of fuzzy sets

3.2 Gaussian membership function of fuzzy set "Medium"

3.3 An example of the fuzzy reasoning process

3.4 The membership function of the fuzzy set B'

3.5 Basic configuration of fuzzy systems

3.6 Defuzzification of fuzzy set B'

3.7 An example of the center average defuzzifier

3.8 The membership functions of 5 fuzzy sets

3.9 The nonlinear function $f(x)$ and the estimation $\hat{f}(x)$ with 5 rules

3.10 The membership functions of 7 fuzzy sets

3.11 The nonlinear function $f(x)$ and the estimation $\hat{f}(x)$ with 7 rules

5.1 Membership functions of $q_j$

5.2 Membership functions of $q_j$

5.3 Membership functions of $\bar{q}_{ij}$

5.4 Membership functions of $\tilde{q}_j$

5.5 (a) Tracking and (b) errors of joint 1 in Yoo and Ham's algorithm. Dash line: desired trajectory; solid line: actual trajectory.

5.6 (a) Control input and (b) sliding surface of joint 1 in Yoo and Ham's algorithm.

5.7 (a) Tracking and (b) errors of joint 2 in Yoo and Ham's algorithm. Dash line: desired trajectory; solid line: actual trajectory.
5.8 (a) Control input and (b) sliding surface of joint 2 in Yoo and Ham's algorithm. ......................................................... 71
5.9 Membership functions of $s_j$ .............................................. 73
5.10 (a) Tracking and (b) errors of joint 1 in Guo and Woo's algorithm.
Dash line: desired trajectory; solid line: actual trajectory. .......... 73
5.11 (a) Control input and (b) sliding surface of joint 1 in Guo and Woo's algorithm. ......................................................... 74
5.12 (a) Tracking and (b) errors of joint 2 in Guo and Woo's algorithm.
Dash line: desired trajectory; solid line: actual trajectory. .......... 74
5.13 (a) Control input and (b) sliding surface of joint 2 in Guo and Woo's algorithm. ......................................................... 75
5.14 (a) Tracking and (b) errors of joint 1 in revised Lin and Hsu's algorithm.
Dash line: desired trajectory; solid line: actual trajectory. .......... 76
5.15 (a) Control input and (b) sliding surface of joint 1 in revised Lin and Hsu's algorithm. ......................................................... 76
5.16 (a) Tracking and (b) errors of joint 2 in revised Lin and Hsu's algorithm.
Dash line: desired trajectory; solid line: actual trajectory. .......... 77
5.17 (a) Control input and (b) sliding surface of joint 2 in revised Lin and Hsu's algorithm. ......................................................... 77
5.18 (a) Tracking and (b) errors of joint 1 in revised Shahnazi et al.'s algorithm. Dash line: desired trajectory; solid line: actual trajectory. .... 78
5.19 (a) Control input and (b) sliding surface of joint 1 in revised Shahnazi et al.'s algorithm. ......................................................... 79
5.20 (a) Tracking and (b) errors of joint 2 in revised Shahnazi et al.'s algorithm. Dash line: desired trajectory; solid line: actual trajectory. .... 79
5.21 (a) Control input and (b) sliding surface of joint 2 in revised Shahnazi 
et al.'s algorithm. ................................................................. 80
5.22 (a) Tracking and (b) errors of joint 1 in Medhaffar et al's algorithm. 
Dash line: desired trajectory; solid line: actual trajectory. ............. 81
5.23 (a) Control input and (b) sliding surface of joint 1 in Medhaffar et al's 
algorithm. ......................................................................................... 81
5.24 (a) Tracking and (b) errors of joint 2 in Medhaffar et al's algorithm. 
Dash line: desired trajectory; solid line: actual trajectory. ............. 82
5.25 (a) Control input and (b) sliding surface of joint 2 in Medhaffar et al's 
algorithm. ......................................................................................... 82
A-1 Simulink model to simulate a 2 DOF robotic manipulator using clas­
sical sliding mode control method. ...................................................... 89
A-2 Simulink subsystem that implements the desired trajectory of the joint 
for a 2 DOF robotic manipulator. ......................................................... 90
A-3 Simulink subsystem that implements the dynamics of a 2 DOF robotic 
manipulator. ......................................................................................... 90
A-4 Embedded MATLAB function code in Figure A-1. ....................... 91
A-5 Simulink model to simulate a 2 DOF robotic manipulator using clas­
sical sliding mode control method with the saturation function. ....... 92
A-6 Embedded MATLAB function code in Figure A-5. ....................... 93
A-7 Simulink model to simulate a 2 DOF robotic manipulator using Yoo 
and Ham's algorithm. ......................................................................... 94
A-8 Embedded MATLAB function code in Figure A-7(page1) .......... 95
A-9 Embedded MATLAB function code in Figure A-7(page2) .......... 96
A-10 Embedded MATLAB function code in Figure A-7(page3) ......... 97
A-11 Simulink model to simulate a 2 DOF robotic manipulator using Guo and Woo's algorithm .......................................................... 98
A-12 Embedded MATLAB function code in Figure A-11 ....................... 99
A-13 Simulink model to simulate a 2 DOF robotic manipulator using revised Lin and Hsu's algorithm or revised Shahnazi et al.'s algorithm .... 100
A-14 Simulink Subsystem1 for revised Lin and Hsu's algorithm in Figure A-13(same as in Subsystem2) ...................................................... 101
A-15 Embedded MATLAB function code in Figure A-14(same as in Subsystem2) ................................................................. 102
A-16 Simulink Subsystem1 for revised Shahnazi et al.'s algorithm in Figure A-13(same as in Subsystem2) ...................................................... 103
A-17 Embedded MATLAB function code in Figure A-16(same as in Subsystem2) ................................................................. 104
A-18 Simulink model to simulate a 2 DOF robotic manipulator using Medhaffar et al.'s algorithm ............................................................ 105
A-19 Embedded MATLAB function code in Figure A-18 (page 1) ........ 106
A-20 Embedded MATLAB function code in Figure A-18 (page 2) ........ 107
List of Tables

4.1 Comparison of five algorithms .......................... 64
Chapter 1

Introduction

1.1 Literature Overview

Classical sliding mode control is robust to model uncertainties and external disturbances. A sliding mode control method with a switching control law guarantees asymptotic stability of the system, but the addition of the switching control law introduces chattering into the system. One way of attenuating chattering is to insert a saturation function \([1]\) inside of a boundary layer around the sliding surface. Unfortunately, this addition disrupts Lyapunov stability of the closed-loop system. Classical sliding mode control method has difficulty in handling unstructured model uncertainties. One can overcome this problem by combining a sliding mode controller and fuzzy systems together. Fuzzy rules allow fuzzy systems to approximate arbitrary continuous functions. To approximate a time-varying nonlinear system, a fuzzy system requires a large amount of fuzzy rules. This large number of fuzzy rules will cause a high computation load. The addition of an adaptive law to a fuzzy sliding mode controller to online tune the parameters of the fuzzy rules in use will ensure a moderate computational load.

A number of adaptive fuzzy sliding mode control (AFSMC) algorithms \([2]-[10]\)
have been proposed in the last decade. The adaptive laws in these algorithms are
designed based on the Lyapunov stability theorem. Asymptotic stability of the closed
loop system is also proved in the sense of Lyapunov [2]-[10]. However, it is difficult
from the archival literature to differentiate between the algorithms.

1.2 Thesis Overview and Contributions

In this thesis we will highlight the differences between five AFSMC methods [3][5]
[6][9][10]. These five AFSMC algorithms will be analyzed and evaluated on robotic
manipulators. Chapter 2, serves as an introduction to the classical sliding mode con­
trol algorithm and its application to a two degree-of-freedom robotic manipulator.
Chapter 3, presents the background of fuzzy systems with fuzzy if-then rules and
fuzzy membership functions. Chapter 4, introduces and describes the five AFSMC
algorithms and proves Lyapunov stability. In order to achieve the preceding, algo­
rithms [6][9] have been revised to better suit the robotic manipulator application.
Chapter 5 presents both the simulation results of the five algorithms applied to a 2
degree-of-freedom robotic manipulator and the stability analysis in a practical sense
for the algorithms.

The contributions of this thesis are as follows:

1. Redesign adaptation laws of algorithms [6][9], which are originally designed for
an induction servomotor drive, for general robotic manipulators.

2. Rewrite Lyapunov function candidates in [6][9] and prove the asymptotic con­
vergence of the closed-loop system using Lyapunov stability theorem mathematically.

3. Compare and evaluate five AFSMC algorithms under two headings: their use
of fuzzy systems; and, the results they yield when simulated for a two degree
of-freedom robotic manipulator. In regards to the former, we will be looking at
the availability of online tuning methodology and the number of fuzzy if-then
rules inherent to the fuzzy system being used and the corresponding computa­
tional load. Our analysis of the results will be limited to tracking accuracy and
chattering.

4. Illustrate the difficulty in proving Lyapunov stability of the overall system in a
practical sense for the above five algorithms.
Chapter 2

Introduction to Sliding Mode Control

2.1 Why Sliding Mode Control?

Dynamics of robotic manipulators have nonlinear characteristics. There are nonlinear kinematic and dynamic equations in dynamics of robotic manipulators. Conventional linear control technologies are not quite satisfying to control robotic manipulators. Nonlinear control technologies can deal with nonlinear equations in the dynamics of robotic manipulators. Conventional nonlinear control strategies cannot provide good robustness for controlling robotic manipulators. The control system designer is often unsure of the exact value of the manipulator parameters that describe the dynamic behavior of the manipulator. One does not precisely know the value of the manipulator inertial parameters nor the effects of unknown loads. Sliding mode control methods can cope with these uncertainties in the dynamics of the robotic manipulator. Sliding mode controllers are robust controllers for controlling robotic manipulators.
2.2 Classical Sliding Mode Control

This section provides an overview of classical sliding mode control based on [1]. We consider a single input dynamic system:

\[ x^{(n)} = f(x) + b(x)u \]  

(2.1)

where \( x^{(n)} \) is the \( n \)th derivative of \( x \) with respect to time, \( u \) is the control input and \( x = [x, \dot{x}, \ldots, x^{(n-1)}]^{T} \) is the state vector. In (2.1), \( f(x) \) and \( b(x) \) are not known but the upperbound and \( b(x) \) is of known sign. Our aim is to track a desired state \( x_d = [x_d, \dot{x_d}, \ldots, x_d^{(n-1)}]^{T} \) in the presence of model imprecision on \( f(x) \) and \( b(x) \). We define the tracking error vector as \( \tilde{x} = x - x_d = [\tilde{x}, \dot{\tilde{x}}, \ldots, \tilde{x}^{(n-1)}]^{T} \). A time-varying sliding surface \( s(t) \) in the state space \( \mathbb{R}^n \) is given by [1]

\[ s(t) = \left( \frac{d}{dt} + \lambda \right)^{n-1} \tilde{x} \]  

(2.2)

where \( \lambda \) is a positive constant and \( s(t) \) is a scalar.

Given the initial condition \( x_d(0) = x(0) \), the problem of tracking \( x(t) = x_d(t) \) is equivalent to that of remaining on the surface \( s(t) \) for all \( t > 0 \). The unique solution of \( s(t) = 0 \) is \( \tilde{x} = 0 \).

We need to keep \( s(t) \) at zero when tracking is outside of \( s(t) \). We choose control input \( u \) such that outside of \( s(t) \) [1]

\[ \frac{1}{2} \frac{d}{dt} s^2(t) \leq -\eta |s(t)| \]  

(2.3)

where \( \eta \) is a positive constant. Equation (2.3) forces tracking trajectories towards the sliding surface as shown in Figure 2.1 which is called sliding condition.

We assume \( s(0) > 0 \) and define \( t_{\text{reach}} \) as the time that trajectories reach the surface.

We rewrite (2.3) as

\[ \frac{d}{dt} s(t) \leq -\eta . \]  

(2.4)
We integrate (2.4) from \( t = 0 \) to \( t = t_{\text{reach}} \) and get

\[
s(t_{\text{reach}}) - s(0) \leq -\eta(t_{\text{reach}} - 0) .
\]  
(2.5)

We know \( s(t_{\text{reach}}) = 0 \). Equation (2.5) becomes

\[
t_{\text{reach}} \leq s(0)/\eta .
\]  
(2.6)

We can also get \( t_{\text{reach}} \) with \( s(0) < 0 \) which leads to

\[
t_{\text{reach}} \leq |s(0)|/\eta .
\]  
(2.7)

Equation (2.7) guarantees that if trajectories are outside of \( s(t) \), they will reach the sliding surface in a finite time smaller than \( |s(0)|/\eta \). After trajectories reach the surface \( s = 0 \), tracking errors tend to zero \( (x = x_d) \) exponentially with a time constant \( (n - 1)/\lambda \).

An equivalent expression of sliding surface equation (2.2) is integral control which is defined as

\[
s(t) = \left( \frac{d}{dt} + \lambda \right)^n \left( \int_0^t \tilde{x} dT \right) .
\]  
(2.8)
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

We now take an example of constructing a control law. We consider a second-order system

$$\ddot{x} = f + u$$  \hspace{1cm} (2.9)

where $u$ is the control input, $x$ is the scalar output, $f$ is not exactly known, and $\hat{f}$ is the estimate of $f$. We assume that the estimation error is bounded by known $F$

$$|\hat{f} - f| \leq F.$$  \hspace{1cm} (2.10)

We define a sliding surface as

$$s = (d/dt + \lambda) \dot{x} = \dot{x} + \lambda \ddot{x}.$$  \hspace{1cm} (2.11)

We take the derivative of $s$ and get

$$\dot{s} = \ddot{x} + \lambda \ddot{x} = \ddot{x} - \dot{x}_d + \lambda \ddot{x} = f + u - \dot{x}_d + \lambda \ddot{x}.$$  \hspace{1cm} (2.12)

We need $s = 0$ and $\dot{s} = 0$. The best approximation $\hat{u}$ is,

$$\hat{u} = -\hat{f} + \ddot{x}_d - \lambda \ddot{x}.$$  \hspace{1cm} (2.13)

The dynamics of $f$ is uncertain. We add a sign function in (2.13) and get the new control input

$$u = \hat{u} - k \text{sgn}(s)$$  \hspace{1cm} (2.14)

where $\text{sgn}(s)$ is

$$\text{sgn}(s) = \begin{cases} 
1 & (s > 0) \\
-1 & (s < 0) \\
0 & (s = 0) 
\end{cases}.$$  \hspace{1cm} (2.15)

We get the equation from the left hand side of (2.3)

$$\frac{1}{2} \frac{d}{dt} s^2 = \dot{s} s = [f - \hat{f} - k \text{sgn}(s)]s = (f - \hat{f})s - k|s|.$$  \hspace{1cm} (2.16)
Let \( k = F + \eta \), we get (2.3) as desired. Another way is using (2.8) instead of (2.2) to get sliding surface

\[
s = \left( \frac{d}{dt} + \lambda \right)^2 \left( \int_0^t \ddot{x} d\tau \right) = \ddot{x} + 2\lambda \dot{x} + \lambda^2 \int_0^t \ddot{x} d\tau .
\] (2.17)

The approximation \( \dot{u} \) can be written as

\[
\dot{u} = -\ddot{f} + \dddot{x} - 2\lambda \dot{x} - \lambda^2 \ddot{x} .
\] (2.18)

We will also get the same expression as (2.14) and the same \( k = F + \eta \).

Since the control law \( u \) has to be discontinuous across \( s(t) \) and implementation of the control switching could not be perfect, this will lead to chattering. Chattering is highly undesirable because it involves extremely high control activity and may excite high-frequency dynamics neglected in the course of modeling [1]. We introduce a boundary layer to smooth out the control discontinuity neighboring the switching surface

\[
B(t) = \{ x, |s(t)| \leq \Phi \}; \Phi > 0
\] (2.19)

where \( \Phi \) is the boundary layer thickness.

\[
\begin{array}{c}
\text{Figure 2.2: Derivation of bounds}
\end{array}
\]

In Figure 2.2, \( p = d/dt \) is the Laplace operator, and \( y_1 \) is the first filter output.
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

The output \( y_1 \) can be written as

\[
y_1(t) = \int_0^t e^{-\lambda(t-T)} s(T) dT.
\]

Since \(|s| \leq \Phi\), we get

\[
|y_1(t)| \leq \Phi \int_0^t e^{-\lambda(t-T)} dT = \Phi \lambda (1 - e^{-\lambda t}) \leq \Phi / \lambda.
\]

From \(|y_1(t)| \leq \Phi / \lambda\), we can get

\[
y_2(t) = \int_0^t e^{-\lambda(t-T)} y_1(T) dT
\]

\[
|y_2(t)| \leq \Phi / \lambda \int_0^t e^{-\lambda(t-T)} dT \leq \Phi / \lambda^2.
\]

Thus \(|y_{n-1}(t)|\) is described as

\[
|y_{n-1}(t)| = |\tilde{x}| \leq \Phi / \lambda^{n-1} = \varepsilon
\] (2.20)

where \(\varepsilon\) is the boundary layer width. This shows that if we are willing to sacrifice precision, that is, from perfect tracking error \(\tilde{x} = 0\) to tracking within precision \(|\tilde{x}| \leq \varepsilon\), the requirement for control law is reduced from satisfying the sliding condition (2.3) all the time to satisfying the sliding condition only when trajectories are outside of the boundary layer \(B(t)\) [1] as shown in Figure 2.3.

To achieve this smoothing control law instead of the discontinuous control law within the boundary layer, we design a saturation function \( sat(s/\Phi) \) added to the control law and change (2.14) to

\[
u = \dot{u} - k \ sat(s/\Phi)
\] (2.21)

where \( sat(s/\Phi) \) is defined as

\[
sat(s/\Phi) = \begin{cases} 
1 & (s/\Phi > 1) \\
-1 & (s/\Phi \leq -1) \\
s/\Phi & (-1 < s/\Phi \leq 1)
\end{cases}
\] (2.22)

The control input \(u\) is shown in Figure 2.4.
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

Figure 2.3: Boundary layer with second order system

Figure 2.4: Control law in the boundary layer
2.3 Model of Robotic Manipulators

The dynamic equation of an n-link robotic manipulator is defined as

\[ M(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau \]  
\[(2.23)\]

where \( q \in \mathbb{R}^n \) is the vector of joint position, \( M(q) \in \mathbb{R}^{n \times n} \) is the inertial matrix, \( C(q, \dot{q}) \in \mathbb{R}^n \) is the matrix of Coriolis and centrifugal forces, \( G(q) \in \mathbb{R}^n \) is the gravity vector and \( \tau \in \mathbb{R}^n \) is the vector of joint torques.

We use a two-degree-of-freedom robot as our simulated robotic manipulator illustrated in Figure 2.5.

\[ M(q)\ddot{q} + C(q, \dot{q}) = \tau \]  
\[(2.24)\]

where

\[
M(q) = \begin{bmatrix}
    m_1 l^2 + 2m_2 l^2 + 2m_2 l^2 \cos q_2 & m_2 l^2 + m_2 l^2 \cos q_2 \\
    m_2 l^2 + m_2 l^2 \cos q_2 & m_2 l^2
\end{bmatrix},
\]

\[(2.25)\]
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

\[
C(q, \dot{q}) = \begin{bmatrix}
-2m_2l^2\dot{q}_1\dot{q}_2\sin q_2 - m_2l^2\dot{q}_2^2\sin q_2 \\
m_2l^2\dot{q}_1^2\sin q_2
\end{bmatrix}.
\]  \tag{2.26}

2.4 Classical Sliding Mode Control for Robotic Manipulators

Our aim is to track the desired trajectories \(q_d\) of the robotic manipulators (2.24) by using a sliding mode controller. We extract \(\dot{q}\) from \(C(q, \dot{q})\) in (2.24) and rewrite (2.24) as

\[
\tau = M(q)\ddot{q} + C_1(q, \dot{q})\dot{q}
\]  \tag{2.27}\

where

\[
C_1(q, \dot{q}) = \begin{bmatrix}
-m_2l^2\dot{q}_1\sin q_2 & -m_2l^2\dot{q}_1\sin q_2 - m_2l^2\dot{q}_2\sin q_2 \\
m_2l^2\dot{q}_1^2\sin q_2 & 0
\end{bmatrix}.
\]  \tag{2.28}

We define the tracking error as \(e = q - q_d\) \tag{2.29}

We define the tracking error as

\[
\dot{e} = \lambda e
\]  \tag{2.30}

where \(\lambda = \text{diag}[\lambda_1, \lambda_2]\), \(\lambda_1\) and \(\lambda_2\) are chosen as the bandwidth of the robot controller.

We need to choose \(\tau\) to satisfy the sufficient condition (2.3). We define the reference state as

\[
\dot{q}_r = \dot{q} - s = \dot{q}_d - \lambda e.
\]  \tag{2.31}\

Now we pick the control input \(\tau\) as

\[
\tau = \ddot{M}\ddot{q}_r + \dot{C}_r\dot{q} - A\dot{s} - K\text{sgn}(s)
\]  \tag{2.32}
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

where $\hat{M}$ and $\hat{C}_1$ are the estimations of $M(q)$ and $C_1(q, \dot{q})$; $A = \text{diag}[a_1, a_2]$ and $K = \text{diag}[K_1, K_2]$ are diagonal positive definite matrices. From (2.27) and (2.32), we can get

$$\dot{M} \dot{s} + (C_1 + A) s = \Delta f - K \text{sgn}(s)$$

(2.33)

where $\Delta f = \Delta M \ddot{q}_r + \Delta C_1 \dot{q}_r$, $\Delta M = \hat{M} - M$ and $\Delta C_1 = \hat{C}_1 - C_1$. We assume that the bound $|\Delta f_i|_{\text{bound}}$ of $\Delta f_i$ ($i = 1, 2$) is known. We choose $K$ as

$$K_i \geq |\Delta f_i|_{\text{bound}}.$$  

(2.34)

We pick the Lyapunov function candidate to be

$$V = \frac{1}{2} s^T M s.$$  

(2.35)

Since $M$ is positive symmetric definite, $V > 0$ for $s \neq 0$. Take the derivative of $M$ with respect to time in (2.25) and we get

$$\dot{M} = \begin{bmatrix}
-2m_2l^2 \dot{q}_2 \sin q_2 & -m_2l^2 \dot{q}_2 \sin q_2 \\
-m_2l^2 \dot{q}_2 \sin q_2 & 0
\end{bmatrix}. $$  

(2.36)

From (5.11) and (2.36), we get

$$M - 2C_1 = \begin{bmatrix}
0 & 2m_2l^2 \dot{q}_1 \sin q_2 + m_2l^2 \dot{q}_2 \sin q_2 \\
-2m_2l^2 \dot{q}_1 \sin q_2 - m_2l^2 \dot{q}_2 \sin q_2 & 0
\end{bmatrix}. $$  

(2.37)

which is a skew-symmetric matrix satisfying

$$s^T(M - 2C_1)s = 0.$$  

(2.38)

Then $\dot{V}$ becomes

$$\dot{V} = s^T M \dot{s} + \frac{1}{2} s^T \dot{M} s = s^T(M \dot{s} + C_1 s) = s^T[-A s + \Delta f - K \text{sgn}(s)] = \sum_{i=1}^{2} (s_i[\Delta f_i - K_i \text{sgn}(s_i)]) - s^T A s.$$  

(2.39)
For $K_i \geq |\Delta f_i|$, we always get $s_i[\Delta f_i - K_i \text{sgn}(s_i)] \leq 0$. We can describe $\dot{V}$ as

$$
\dot{V} = \sum_{i=1}^{2} (s_i[\Delta f_i - K_i \text{sgn}(s_i)]) - s^T A s \leq -s^T A s < 0 \ (s \neq 0).
$$

(2.40)

To attenuate chattering problem, we introduce a saturation function in the control law instead of the sign function in (2.32). The control law becomes

$$
\tau = \hat{M}\ddot{q}_r + \hat{C}_1 \dot{q}_r - A s - K \text{sat}(s/\Phi).
$$

(2.41)

In this classical sliding mode control method, the model of the robotic manipulator is partly unknown. We know the model structure of the robotic manipulator. But we don’t know the real values of $m_1 l^2$ and $m_2 l^2$ in $M$ and $C_1$. We define $\hat{p}_1 = m_1 l^2$ and $\hat{p}_2 = m_2 l^2$. $\hat{p}_1$ and $\hat{p}_2$ are our estimations of $p_1$ and $p_2$. If the values of $\hat{p}_1$ and $\hat{p}_2$ are away from the real values, we need to increase the control gain $K$ to prevent the tracking error in (2.29) getting bigger. The chattering problem may be eliminated by using the saturation function in (2.41). But there is no theoretical Lyapunov stability proof for using this control law in (2.41).

We simulate this classical sliding mode control method for the two link robotic manipulator in (2.24). We pick $\hat{p}_1 = 1$ and $\hat{p}_2 = 2$. Since robotic manipulators cannot follow a step sequence instantaneously, the desired trajectory will be the output of a filtered sequence of unit steps. We define the transfer function of the pre-filter for each joint of the robotic manipulator [11]:

$$
W_m(s) = \frac{4}{s^2 + 4s + 4}.
$$

(2.42)

The initial values of the robotic manipulators’ joint positions are set to 0.5 radians. The estimated dynamics of the robotic manipulators are $\hat{p}_1 = 1.5$, $\hat{p}_2 = 3$. We pick $\lambda$ in (2.30) as the same bandwidth as in (2.42). We simply pick $A = \text{diag}[1, 1]$ in (2.32) as in [5]. To decide $K$ in (2.32), we need to know the real $\Delta f$ in (2.40). First we choose $K = [20, 10]^T$. We run simulation and plot $\Delta f$. We get $|\Delta f_i| \leq 45$
and $|\Delta f_2| \leq 21$. This will fail the criterion in (2.34). To satisfy this criterion, we choose $K = [45, 21]^T$ and simulate again. Figure 2.6 and Figure 2.8 illustrate that the tracking accuracy is guaranteed. But the chattering problem exists as shown in Figure 2.7 and Figure 2.9. We now change the estimated values to $\hat{p}_1 = 7$ and $\hat{p}_2 = 27$. We run simulation and plot the results. Figure 2.10 and Figure 2.11 show that the tracking errors do not converge. With the values of $|p_1 - \hat{p}_1|$ and $|p_2 - \hat{p}_2|$ getting larger, the estimated dynamics are away from our real dynamics of the robotic manipulator. This results in the increase of the tracking errors.

To attenuate chattering, we use the saturation function described in (2.21). Our control law changes to

$$
\tau = \dot{M}\ddot{q}_r + \dot{C}_1\dot{q}_r - A\ddot{s} - K_{\text{sat}}(s). \quad (2.43)
$$

We simulate with the same parameters of $p_1 = 1.5$, $p_2 = 3$, $K = [45, 21]^T$, $A = \text{diag}[1, 1]$ and $A = \text{diag}[2, 2]$. Our estimated system is $\hat{p}_1 = 1.5$ and $\hat{p}_2 = 3$. We use trial and error for the boundary layer $\Phi_1$ and $\Phi_2$. We try $\Phi_1 = \Phi_2 = 0.1$, simulate and plot the results. Figure 2.13 and Figure 2.15 show that the chattering problem is avoided. From (2.20), we can calculate the tracking errors inside the boundary layer by

$$
|\ddot{x}| \leq \Phi/\lambda^{n-1} = 0.1/2^{2-1} = 0.05 = \varepsilon.
$$

The tracking errors are within the range of 0.05 radians/second as shown in Figure 2.12 and Figure 2.15. We try $\Phi_1 = \Phi_2 = 0.01$ and simulate again. We find that the chattering problem comes out again. There is a trade-off between the chattering problem and the tracking accuracy. When we decrease the boundary layer and the tracking errors getting small, the chattering phenomena appear.
Figure 2.6: (a) Tracking and (b) errors of joint 1 in Classical SMC. Dash line: desired trajectory; solid line: actual trajectory.

Figure 2.7: (a) Control input and (b) sliding surface of joint 1 in Classical SMC.
Figure 2.8: (a) Tracking and (b) errors of joint 2 in Classical SMC. Dash line: desired trajectory; solid line: actual trajectory.

Figure 2.9: (a) Control input and (b) sliding surface of joint 2 in Classical SMC.
Figure 2.10: (a) Tracking and (b) errors of joint 1 with $\hat{p}_1 = 3$, $\hat{p}_2 = 7$. Dash line: desired trajectory; solid line: actual trajectory.

Figure 2.11: (a) Tracking and (b) errors of joint 2 with $\hat{p}_1 = 3$, $\hat{p}_2 = 7$. Dash line: desired trajectory; solid line: actual trajectory.
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

Figure 2.12: (a) Tracking and (b) errors of joint 1 in Classical SMC by using saturation function. Dash line: desired trajectory; solid line: actual trajectory.

Figure 2.13: (a) Control input and (b) sliding surface of joint 1 in Classical SMC by using saturation function.
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

Figure 2.14: (a) Tracking and (b) errors of joint 2 in Classical SMC by using saturation function. Dash line: desired trajectory; solid line: actual trajectory.

Figure 2.15: (a) Control input and (b) sliding surface of joint 2 in Classical SMC by using saturation function.
CHAPTER 2. INTRODUCTION TO SLIDING MODE CONTROL

2.5 Summary

In this chapter we introduced sliding mode control theory. Sliding mode control can effectively deal with nonlinear systems especially for robotic manipulators. The control input forces the robotic manipulators to track the desired trajectories. The chattering problem can be attenuated by using the saturation function in the control law. Lyapunov stability is proved only in the case of the sign function in the control law. Classical sliding mode control can be used to control the robotic manipulator under the condition of partly known model dynamics. In the next chapter we will introduce fuzzy systems which can estimate system dynamics.
Chapter 3

Introduction to Fuzzy Control

Zadeh [12] introduced fuzzy sets in 1965. After 40 years, fuzzy systems have been widely used in different fields, especially on control problems. Fuzzy systems transfer expert knowledge to mathematical models. Fuzzy systems use fuzzy logic to estimate dynamics of our systems. Fuzzy controllers including fuzzy if-then rules are used to control our systems.

3.1 Description and Analysis of Fuzzy Logic Systems

3.1.1 Basic Concepts

We define $X$ as the universe of discourse (or the universe). A fuzzy set $A$ in $X$ is given by [13]

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

(3.1)

where $\mu_A(x)$ is the membership function (MF) of the fuzzy set $A$ and $x$ are elements of $X$. The membership function $\mu_A(x)$ of the fuzzy set $A$ can have values between 0
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

and 1. The membership function maps each element of the universe to a membership grade between 0 and 1. If $\mu_A(x)$ can only have values of 0 or 1, the fuzzy set $A$ is changed to a classical set. Classical sets are also called crisp sets. Figure 3.1 (a) illustrates the typical membership functions of fuzzy sets “small,” “medium,” and “big”. The universe $X$ is defined as “torque.” There are three different fuzzy sets with three membership functions $\mu_{\text{small}}(x)$, $\mu_{\text{medium}}(x)$ and $\mu_{\text{big}}(x)$.

![Membership functions of fuzzy sets](image)

Figure 3.1: Membership functions of fuzzy sets

We introduce six concepts of a fuzzy set: support, fuzzy singleton, symmetry, center, height and normality. The support of a fuzzy set $A$ is the set with $\mu_A(x) > 0$ for all points $x$ in $X$ [13]:

$$\text{support}(A) = \{x \in X|\mu_A(x) > 0\} .$$ (3.2)

A fuzzy singleton is a fuzzy set whose support is a single point in $X$ [13]. Figure 3.1(b) shows the single point $x = 100$ in $X$. A fuzzy set $A$ is symmetric if $\mu_A(x)$ is symmetric around a certain point $x = c$. Figure 3.1(a) shows that the fuzzy set “medium” is symmetric around the point $x = 100$. The center of a fuzzy set $A$ is the mean value of all points $x$ in $X$ that achieve the maximum value of $\mu_A(x)$. The point $x = 100$ is the center of fuzzy set “medium” in Figure 3.1(a). If all the points $x$ in $X$ that achieve the maximum value of $\mu_A(x)$ are infinite, the center of the fuzzy set $A$ is
defined as the largest (smallest) value among these points in $X$. Figure 3.1(a) shows that the point $x = 50$ and the point $x = 150$ are the centers of fuzzy sets “small” and “big”. The height of a fuzzy set $A$ is the largest value of $\mu_A(x)$. The height of fuzzy set “medium” in Figure 3.1(a) is equal to 1. A normal fuzzy set is the set whose height is equal to 1. All sets in Figure 3.1 are normal fuzzy sets.

A triangular membership function of a fuzzy set is defined as [13]

$$
\text{triangle}(x; a, b, c) = \begin{cases} 
0 & x \leq a \\
\frac{x-a}{b-a} & a \leq x \leq b \\
\frac{c-x}{c-b} & b \leq x \leq c \\
0 & c \leq x 
\end{cases}
$$

where the parameters $\{a, b, c\}$ are the corner points of the membership function. Figure 3.1(a) shows that the corner points are $a = 50$, $b = 100$ and $c = 150$ for fuzzy set “medium”.

![Gaussian membership function of fuzzy set "Medium"](image)

A Gaussian membership function is defined as [13]

$$
\text{gaussian}(x; \alpha, \delta) = e^{-\frac{1}{2}(\frac{x-\alpha}{\delta})^2}
$$

where $\alpha$ is the mean and $\delta$ is the variance. Figure 3.2 shows a gaussian membership function with $\alpha = 100$ and $\delta = 30$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

We introduce two operations on fuzzy sets: intersection and union. Assume we have two fuzzy sets \( A \) and \( B \) defined in the same universe of discourse \( U \), the intersection of these two fuzzy sets is a fuzzy set whose membership function is

\[
\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x) \ast \mu_B(x)
\]  

(3.5)

where \( \ast \) is defined as a T-norm operator. We introduce two examples of the T-norm operator:

Minimum : \( T_{\text{min}}(a, b) = \min(a, b) = a \land b \)  

(3.6)

Algebraic product : \( T_{\text{ap}}(a, b) = ab \).

(3.7)

The union of two fuzzy sets \( A \) and \( B \) is a fuzzy set whose membership function is defined as

\[
\mu_{A\cup B}(x) = S(\mu_A(x), \mu_B(x)) = \mu_A(x) \oplus \mu_B(x)
\]  

(3.8)

where \( \oplus \) is defined as a S-norm operator. We introduce two examples of the S-norm operator:

Maximum : \( S_{\text{max}}(a, b) = \max(a, b) = a \lor b \)  

(3.9)

Algebraic sum : \( S_{\text{ap}}(a, b) = a + b - ab \).

(3.10)

We introduce fuzzy relations. A fuzzy relation is a fuzzy set defined in the Cartesian product of \( U_1, U_2, \cdots, U_n \), where \( U_1, U_2, \cdots, U_n \) are universes of discourse. We can define a fuzzy relation \( Q \) in \( U_1 \times U_2 \times \cdots \times U_n \) as

\[
Q = \{(u_1, u_2, \cdots, u_n), \mu_Q(u_1, u_2, \cdots, u_n) \mid (u_1, u_2, \cdots, u_n) \in U_1 \times U_2 \times \cdots \times U_n \}
\]  

(3.11)

where \( \mu_Q(u_1, u_2, \cdots, u_n) \) is the membership function of the fuzzy relation \( Q \). A binary fuzzy relation is a fuzzy set defined in the Cartesian product of two sets \( U_1 \) and \( U_2 \).
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

If we let $X$ and $Y$ be two universes of discourse, a binary fuzzy relation $R$ in $X \times Y$ is defined as

$$R = \{((x, y), \mu_R(x, y)) \mid (x, y) \in X \times Y\}.$$  \hspace{1cm} (3.12)

A fuzzy if-then rule is described as

\[\text{if } x \text{ is } A \text{ then } y \text{ is } B\] \hspace{1cm} (3.13)

where $x$, $y$ are called linguistic variables. $A$ and $B$ are fuzzy sets defined in $X$, $Y$ (domains of $x$, $y$). "$x$ is $A$" is called the premise or antecedent, "$y$ is $B$" is called the consequence or conclusion. A fuzzy if-then rule can be considered as a binary fuzzy relation. The membership function for this fuzzy if-then rule relation is defined as [13]

$$\mu_R(x, y) = \mu_{A \times B}(x, y) = \mu_{A \rightarrow B}(x, y) = \mu_A(x) \# \mu_B(y)$$ \hspace{1cm} (3.14)

where we use $A \rightarrow B$ to interpret the fuzzy if-then rule (the fuzzy relation) and the T-norm operator $\#$ is defined in (3.6) and (3.7).

We define modus ponens (the traditional logic form) as

- premise 1 (rule): if $x$ is $A$ then $y$ is $B$
- premise 2 (fact): $x$ is $A$
- conclusion: $y$ is $B$

Fuzzy reasoning which is referred to the generalized modus ponens. The form of fuzzy reasoning is given by

- premise 1 (rule): if $x$ is $A$ then $y$ is $B$
- premise 2 (fact): $x$ is $A'$
- conclusion: $y$ is $B'$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

Given the fuzzy set \( A' \) and the fuzzy relation \( A \rightarrow B \), the fuzzy set \( B' \) is defined as:

\[
B' = A' \circ R = A' \circ (A \rightarrow B)
\]  

(3.15)

where we define \( A' \circ R \) as the composition of \( A' \) and \( R \). We introduce two composition operations: max-min composition and max-product composition. The membership function of \( B' \) with max-min composition is given by

\[
\mu_{B'}(y) = \max_x \min \left[ \mu_A(x), \mu_R(x, y) \right].
\]  

(3.16)

We use \( \vee \) to denote maximum operation and \( \wedge \) to denote minimum operation. Then (3.16) becomes to

\[
\mu_{B'}(y) = \max_x \min \left[ \mu_A(x), \mu_R(x, y) \right] = \vee_x [\mu_A(x) \wedge \mu_R(x, y)].
\]  

(3.17)

The membership function of \( B' \) with max-product composition is given by

\[
\mu_{B'}(y) = \max_x \mu_A(x) \mu_R(x, y) = \vee_x [\mu_A(x) \mu_R(x, y)].
\]  

(3.18)

The above fuzzy reasoning is used for a single if-then rule \( (A \rightarrow B) \) with a single antecedent \((x \text{ is } A)\). We describe two fuzzy rules with two antecedents as follows:

premise 1 (rule 1): if \( x_1 \) is \( A_1^1 \) and \( x_2 \) is \( A_2^1 \) then \( y \) is \( B^1 \)

premise 2 (rule 2): if \( x_1 \) is \( A_1^2 \) and \( x_2 \) is \( A_2^2 \) then \( y \) is \( B^2 \)

premise 3 (fact): \( x_1 \) is \( A_1' \) and \( x_2 \) is \( A_2' \)

conclusion: \( y \) is \( B' \)

(3.19)

The fuzzy set \( B' \) can be described as

\[
B' = [(A_1' \times A_2') \circ (A_1^1 \times A_2^1 \rightarrow B^1)] \cup [(A_1' \times A_2') \circ (A_1^2 \times A_2^2 \rightarrow B^2)]
\]  

\[
= [(A_1' \times A_2') \circ R_1] \cup [(A_1' \times A_2') \circ R_2].
\]  

\[
= B'_1 \cup B'_2.
\]  

(3.20)
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

If we use max-min composition for \( \circ \) operator, maximum operation \( \vee \) for \( \cup \) operator and minimum operation \( \wedge \) for fuzzy relations defined in (3.6), the membership function of the fuzzy set \( B' \) becomes

\[
\mu_{B'}(y) = \{\vee_{x_1,x_2}[\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2) \wedge \mu_{R_1}(x_1, x_2, y)]\} \vee \\
\{\vee_{x_1,x_2}[\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_2) \wedge \mu_{R_2}(x_1, x_2, y)]\} \\
= \{\vee_{x_1}[\mu_{A'_1}(x_1) \wedge \mu_{A'_1}(x_1)] \wedge \vee_{x_2}[\mu_{A'_2}(x_2) \wedge \mu_{A'_2}(x_2)] \wedge \mu_{B'_1}(y)\} \vee \\
\{\vee_{x_1}[\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_1)] \wedge \vee_{x_2}[\mu_{A'_2}(x_2) \wedge \mu_{A'_2}(x_2)] \wedge \mu_{B'_2}(y)\} \\
= \mu_{B'_1}(y) \vee \mu_{B'_2}(y) \quad (3.21)
\]

**Example 3.1** We consider two fuzzy rules with two antecedents described in (3.19). The triangular membership functions of the fuzzy sets are defined as follows:

\[
\mu_{A'_1}(x_1) = \text{triangle}(x_1, 3, 6, 9), \quad \mu_{A'_2}(x_2) = \text{triangle}(x_2, 4.5, 7, 9.5), \\
\mu_{A'_1}(x_1) = \text{triangle}(x_1, 5.5, 8.5, 11.5), \quad \mu_{A'_2}(x_2) = \text{triangle}(x_2, 2, 4.5, 7), \\
\mu_{B'_1}(y) = \text{triangle}(y, 4, 7, 10), \quad \mu_{B'_2}(y) = \text{triangle}(y, 1, 4, 7), \\
\mu_{A'_1}(x_1) = \text{triangle}(x_1, 1, 4, 7), \quad \text{and} \quad \mu_{A'_2}(x_2) = \text{triangle}(x_2, 0, 3.5, 7). \quad \text{Then we can get the membership function} \ \mu_{B'}(y) \ \text{from} \ (3.21):
\]

\[
\mu_{B'}(y) = \{\vee_{x_1}[\mu_{A'_1}(x_1) \wedge \mu_{A'_1}(x_1)] \wedge \vee_{x_2}[\mu_{A'_2}(x_2) \wedge \mu_{A'_2}(x_2)] \wedge \mu_{B'_1}(y)\} \vee \\
\{\vee_{x_1}[\mu_{A'_1}(x_1) \wedge \mu_{A'_2}(x_1)] \wedge \vee_{x_2}[\mu_{A'_2}(x_2) \wedge \mu_{A'_2}(x_2)] \wedge \mu_{B'_2}(y)\} \\
= [0.67 \wedge 0.42 \wedge \mu_{B'_1}(y)] \vee [0.25 \wedge 0.83 \wedge \mu_{B'_2}(y)] \\
= [0.42 \wedge \mu_{B'_1}(y)] \vee [0.25 \wedge \mu_{B'_2}(y)] \\
= \mu_{B'_1}(y) \vee \mu_{B'_2}(y) \quad (3.22)
\]
where

\[
\mu_{B'_1}(y) = \begin{cases} 
0 & y \leq 4 \\
\frac{y-4}{3} & 4 \leq y \leq 5.26 \\
0.42 & 5.26 \leq y \leq 8.74 \\
\frac{10-y}{3} & 8.74 \leq y \leq 10 \\
0 & 10 \leq y
\end{cases}
\]  

(3.23)

\[
\mu_{B'_2}(y) = \begin{cases} 
0 & y \leq 1 \\
\frac{y-1}{3} & 1 \leq y \leq 1.75 \\
0.25 & 1.75 \leq y \leq 6.25 \\
\frac{7-y}{3} & 6.25 \leq y \leq 7 \\
0 & 7 \leq y
\end{cases}
\]  

(3.24)

Then

\[
\mu_{B'}(y) = \mu_{B'_1}(y) \lor \mu_{B'_2}(y) = \begin{cases} 
0 & y \leq 1 \\
\frac{y-1}{3} & 1 \leq y \leq 1.75 \\
0.25 & 1.75 \leq y \leq 4.75 \\
\frac{y-4}{3} & 4.75 \leq y \leq 5.26 \\
0.42 & 5.26 \leq y \leq 8.74 \\
\frac{10-y}{3} & 8.74 \leq y \leq 10 \\
0 & 10 \leq y
\end{cases}
\]  

(3.25)

Figure 3.3 shows the calculation process for Example 3.1. Figure 3.4 shows the membership function of the fuzzy set $B'$.  

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

Figure 3.3: An example of the fuzzy reasoning process

Figure 3.4: The membership function of the fuzzy set B'}
3.1.2 Fuzzy Systems and Their Properties

A whole fuzzy system includes four components: fuzzy rule base, fuzzy inference engine, fuzzifier and defuzzifier [14]. Figure 3.5 shows the fuzzy system where inputs and outputs are real-valued variables [14]. We introduce Mamdani fuzzy inference systems as our fuzzy inference engines. Mamdani fuzzy inference systems use algebraic product in (3.7) as our T-norm operators, and max-product composition in (3.18) as our composition operators. Based on the two rules with two antecedents in (3.19), we use max-product composition for the $\circ$ operator and maximum operation for the $\cup$ operator in (3.20). The membership function of fuzzy set $B'$ is given by

$$\mu_{B'}(y) = \max_{i=1}^{2} \max_{x_1, x_2} \left[ \prod_{i=1}^{2} \mu_{A_i}(x_i) \mu_{B'}(y) \right]. \quad (3.26)$$

A fuzzifier is defined as a mapping from a real-valued crisp point $x_i$ to a fuzzy set $A_i'$. To simplify the computation in (3.26), we introduce the singleton fuzzifier:

$$\mu_{A_i'}(x_i) = \begin{cases} 1 & \text{if } x_i = x_i^* \ (i = 1, 2) \\ 0 & \text{others} \end{cases} \quad (3.27)$$

Since $\mu_{A_1'}(x_1^*) = \mu_{A_2'}(x_2^*) = 1$, we can reduce (3.26) into:

$$\mu_{B'}(y) = \max_{i=1}^{2} \left[ \prod_{i=1}^{2} \mu_{A_i}(x_i^*) \mu_{B'}(y) \right]$$

$$= \max_{i=1}^{2} \left[ \mu_{B'}(y) \right] \quad (3.28)$$
In (3.28), the output $B'$ is a fuzzy set with the membership function $\mu_{B'}(y)$. To use a fuzzy system as a controller, we need a crisp output. We use a defuzzifier to map a fuzzy set to a single point. We introduce two defuzzifiers: center of gravity defuzzifier and center average defuzzifier.

The center of gravity defuzzifier is defined as

$$
y_{\text{cog}} = \frac{\int_y y \mu_{B'}(y) \, dy}{\int_y \mu_{B'}(y) \, dy}.
$$

Figure 3.6 shows the centroid of area $B'$. We can get the point $y_{\text{cog}}$ on Y axis. Since the membership function $\mu_{B'}(y)$ is usually irregular, the integrations in (3.29) are difficult to compute. Instead of the center of gravity defuzzifier, we use its approximation: the center average defuzzifier.

The center average defuzzifier is defined as

$$
y_{\text{ca}} = \frac{\sum_{i=1}^{2} y^i \mu_{B'_i}(y^i)}{\sum_{i=1}^{2} \mu_{B'_i}(y^i)} = \frac{\sum_{i=1}^{2} y^i w_i}{\sum_{i=1}^{2} w_i}.
$$

where $y^1$ and $y^2$ are the centers of $B'_1$ and $B'_2$ in (3.20); $w_1 = \mu_{B'_1}(y^1)$ and $w_2 = \mu_{B'_2}(y^2)$ are shown in Figure 3.6. If all fuzzy membership functions are symmetric, we can get: $y^1$ and $y^2$ are the centers of $B_1$ and $B_2$; $w_1$ and $w_2$ are the heights of fuzzy set $B'$.
related to \(y^1\) and \(y^2\). We derive the membership function of the fuzzy set \(B'_l\) at the point \(y^l\) from (3.28). We can get 
\[
\mu_{B'_l}(y^l) = \mu_{A'_l}(x^*_1)\mu_{A'_2}(x^*_2)\mu_{B^l}(y^l).
\]
Assume the fuzzy set \(B^l\) is normal, we get 
\[
\mu_{B^l}(y^l) = 1.
\]
The membership function of the fuzzy set \(B'_l\) at the point \(y^l\) becomes 
\[
\mu_{B'_l}(y^l) = \mu_{A'_l}(x^*_1)\mu_{A'_2}(x^*_2).
\]
Equation (3.30) becomes 
\[
y_{CA} = \frac{\sum_{i=1}^{2} y^l \mu_{B'_l}(y^l)}{\sum_{i=1}^{2} \mu_{B'_l}(y^l)} = \frac{\sum_{i=1}^{2} y^l \prod_{i=1}^{2} \mu_{A'_i}(x^*_i)}{\sum_{i=1}^{2} \prod_{i=1}^{2} \mu_{A'_i}(x^*_i)}.
\] (3.31)

**Example 3.2** We consider the same fuzzy system as in (3.26). The membership functions of the fuzzy sets are given by:

\[
\begin{align*}
\mu_{A'_1}(x_1) &= \text{triangle}(x_1, 3, 6, 9), \\
\mu_{A'_2}(x_2) &= \text{triangle}(x_2, 4.5, 7, 9.5), \\
\mu_{A'_1}(x_1) &= \text{triangle}(x_1, 5.5, 8.5, 11.5), \\
\mu_{A'_2}(x_2) &= \text{triangle}(x_2, 2, 4.5, 7), \\
\mu_{B'_1}(y) &= \text{triangle}(y, 4, 7, 10), \\
\mu_{B'_2}(y) &= \text{triangle}(y, 1, 4, 7), \\
\mu_{A'_1}(x_1) &= \begin{cases} 1 & \text{if } x_1 = 5 \\
0 & \text{others} \end{cases},
\quad \text{and } \mu_{A'_2}(x_2) = \begin{cases} 1 & \text{if } x_2 = 5.5 \\
0 & \text{others} \end{cases}.
\end{align*}
\]
From (3.31), we get 
\[
y_{CA} = \frac{\sum_{i=1}^{2} y^l \prod_{i=1}^{2} \mu_{A'_i}(x^*_i)}{\sum_{i=1}^{2} \prod_{i=1}^{2} \mu_{A'_i}(x^*_i)} = \frac{\sum_{i=1}^{2} y^l \mu_{A'_1}(5)\mu_{A'_2}(5.5)}{\sum_{i=1}^{2} \mu_{A'_1}(5)\mu_{A'_2}(5.5)} \\
= \frac{y^1 \mu_{A'_1}(5)\mu_{A'_2}(5.5) + y^2 \mu_{A'_1}(5)\mu_{A'_2}(5.5)}{\mu_{A'_1}(5)\mu_{A'_2}(5.5) + \mu_{A'_1}(5)\mu_{A'_2}(5.5)} \\
= \frac{7 \times 0.92 \times 0.42 + 4 \times 0.25 \times 0.6}{0.92 \times 0.42 + 0.25 \times 0.6} \\
= \frac{7 \times 0.386 + 4 \times 0.15}{0.386 + 0.15} \\
= 6.16
\] (3.32)

Figure 3.7 illustrates the computation process of the center average defuzzifier.
We now extend (3.19) to a general form (M rules with n antecedents):

\begin{align*}
\text{Rule 1} & : \text{If } x_1 \text{ is } A_1^{1} \text{ and } \cdots \text{ and } x_n \text{ is } A_n^{1}, \text{ then } y \text{ is } B^1 \\
\text{Rule 2} & : \text{If } x_1 \text{ is } A_1^{2} \text{ and } \cdots \text{ and } x_n \text{ is } A_n^{2}, \text{ then } y \text{ is } B^2 \\
\vdots & \\
\text{Rule M} & : \text{If } x_1 \text{ is } A_1^{M} \text{ and } \cdots \text{ and } x_n \text{ is } A_n^{M}, \text{ then } y \text{ is } B^M \\
\text{Fact} & : x_1 \text{ is } A_1^{'} \text{ and } \cdots \text{ and } x_n \text{ is } A_n^{'} \\
\text{Conclusion} & : y \text{ is } B^{'}
\end{align*}

(3.33)

where the membership functions of the the fuzzy sets \(A_1^{1}, \ldots, A_n^{M}\) and \(B^1, \ldots, B^M\).
are defined as $\mu_{A^l_i}(x)$, $\mu_{B^l_i}(y)$. The fuzzy system in (3.31) is rewritten as [14]

$$y_{CA} = \frac{\sum_{l=1}^{M} y^l [\prod_{i=1}^{n} \mu_{A^l_i}(x_i)]}{\sum_{l=1}^{M} [\prod_{i=1}^{n} \mu_{A^l_i}(x_i)]} = f(x)$$  \hspace{1cm} (3.34)

where $l = 1, \cdots, M$ denote $M$ fuzzy rules and $i = 1, \cdots, n$ denote $n$ antecedents; $y^l$ is the point at which $\mu_{B^l_i}(y)$ achieves its maximum value (we assume $\mu_{B^l_i}(y^l) = 1$); $x$ denotes the crisp inputs $x_1, x_2, \cdots, x_n$; $f(x)$ is one crisp output for specific $x$. In the case where we have a Gaussian membership function $\mu_{A^l_i}(x_i)$ as

$$\mu_{A^l_i}(x_i) = \exp \left[ -\left( \frac{x_i - \alpha_i^l}{\delta_i^l} \right)^2 \right],$$  \hspace{1cm} (3.35)

the fuzzy system $f(x)$ changes to

$$f(x) = \frac{\sum_{l=1}^{M} y^l \left[ \prod_{i=1}^{n} \exp \left( -\left( \frac{x_i - \alpha_i^l}{\delta_i^l} \right)^2 \right) \right]}{\sum_{l=1}^{M} \left[ \prod_{i=1}^{n} \exp \left( -\left( \frac{x_i - \alpha_i^l}{\delta_i^l} \right)^2 \right) \right]}. \hspace{1cm} (3.36)$$

If we only have one antecedent ($n = 1$) for every fuzzy rule in (3.33), the fuzzy system $f(x)$ in (3.34) becomes

$$f(x) = \frac{\sum_{l=1}^{M} y^l [\mu_{A^l_i}(x)]}{\sum_{l=1}^{M} [\mu_{A^l_i}(x)]} = \frac{\sum_{l=1}^{M} y^l \left[ \exp \left( -\left( \frac{x - \alpha_i^l}{\delta_i^l} \right)^2 \right) \right]}{\sum_{l=1}^{M} \exp \left( -\left( \frac{x - \alpha_i^l}{\delta_i^l} \right)^2 \right)}. \hspace{1cm} (3.37)$$

We want to know if the fuzzy systems are good approximators for any arbitrary nonlinear systems. We introduce the following Universal Approximation Theorem.

**Theorem 3.1** For any given real continuous function $g(x)$ on a compact set $U \subset R^n$ and arbitrary $\epsilon > 0$, there exists a fuzzy logic system $f(x)$ in the form of (3.36) such that

$$\sup_{x \in U} |f(x) - g(x)| < \epsilon . \hspace{1cm} (3.38)$$
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

The proof is shown on page 124 in [14].

Example 3.3 We consider a first order nonlinear system. The dynamic equation of the system is given by [15]

\[ \dot{x}(t) = \frac{1-e^{-x(t)}}{1+e^{-x(t)}} + u(t) = f(x) + u(t). \] (3.39)

We define five fuzzy sets over the interval \([-3,3]\): negative medium (NM), negative small (NS), zero (ZE), positive small (PS) and positive medium (PM). The membership functions are given by

\[
\begin{align*}
\mu_{NM}(x) &= \exp(-(x + 1.5)^2) \\
\mu_{NS}(x) &= \exp(-(x + 0.5)^2) \\
\mu_{ZE}(x) &= \exp(-x^2) \\
\mu_{PS}(x) &= \exp(-(x - 0.5)^2) \\
\mu_{PM}(x) &= \exp(-(x - 1.5)^2).
\end{align*}
\] (3.40)

Figure 3.8 shows the five membership functions.

![Figure 3.8](image)

Figure 3.8: The membership functions of 5 fuzzy sets

To estimate the dynamics of the system, the following linguistic descriptions (fuzzy
if-then rules) are given to the designer:

\[ R^1 : \text{ if } x \text{ is near } -1.5 \text{ then } f(x) \text{ is near } -0.6 \]
\[ R^2 : \text{ if } x \text{ is near } -0.5 \text{ then } f(x) \text{ is near } -0.2 \]
\[ R^3 : \text{ if } x \text{ is near } 0 \text{ then } f(x) \text{ is near } 0 \]
\[ R^4 : \text{ if } x \text{ is near } 0.5 \text{ then } f(x) \text{ is near } 0.2 \]
\[ R^5 : \text{ if } x \text{ is near } 1.5 \text{ then } f(x) \text{ is near } 0.6 \]

We set \( y_1 = -0.6, y_2 = -0.2, y_3 = 0, y_4 = 0.2 \) and \( y_5 = 0.6 \). Since there is only one antecedent for every fuzzy if-then rule, we rewrite (3.37) as

\[
\hat{f}(x) = \frac{\sum_{i=1}^{5} y_i [\mu_{A_i}(x)]}{\sum_{i=1}^{5} [\mu_{A_i}(x)]}
\]

\[
= \frac{-0.6\mu_{NM}(x) - 0.2\mu_{NS}(x) + 0.2\mu_{PS}(x) + 0.6\mu_{PM}(x)}{\mu_{NM}(x) + \mu_{NS}(x) + \mu_{ZE}(x) + \mu_{PS}(x) + \mu_{PM}(x)}
\]

\[
= \frac{-0.6e^{-(x+1.5)^2} - 0.2e^{-(x+0.5)^2} + 0.2e^{-(x-0.5)^2} + 0.6e^{-(x-1.5)^2}}{e^{-(x+1.5)^2} + e^{-(x+0.5)^2} + e^{-x^2} + e^{-(x-0.5)^2} + e^{-(x-1.5)^2}}
\]

We use MATLAB to simulate \( f(x) \) and \( \hat{f}(x) \). Figure 3.9 shows \( |f(x) - \hat{f}(x)| < \epsilon = 0.35 \) over the interval \([-3, 3]\).

![Figure 3.9: The nonlinear function f(x) and the estimation \( \hat{f}(x) \) with 5 rules](image)
To make $e$ smaller, we need more specific membership functions and linguistic descriptions in the fuzzy system. We define 7 fuzzy sets over the interval $[-3,3]$: negative big (NB), negative medium (NM), negative small (NS), zero (ZE), positive small (PS), positive medium (PM) and positive big (PB). The membership functions of the fuzzy sets NB and PB are defined as $\mu_{NB}(x) = \exp(-(x+2.5)^2)$, $\mu_{PB}(x) = \exp(-(x-2.5)^2)$. The membership functions of fuzzy sets NM, NS, ZE, PS and PM are the same as in (3.40). Figure 3.10 illustrates the membership functions of the 7 fuzzy sets. The fuzzy if-then rules are given as

$$
R^1: \quad \text{if } x \text{ is near } -2.5 \text{ then } f(x) \text{ is near } -0.85 \\
R^2: \quad \text{if } x \text{ is near } -1.5 \text{ then } f(x) \text{ is near } -0.64 \\
R^3: \quad \text{if } x \text{ is near } -0.5 \text{ then } f(x) \text{ is near } -0.24 \\
R^4: \quad \text{if } x \text{ is near 0 then } f(x) \text{ is near 0} \\
R^5: \quad \text{if } x \text{ is near 0.5 then } f(x) \text{ is near 0.24} \\
R^6: \quad \text{if } x \text{ is near 1.5 then } f(x) \text{ is near 0.64} \\
R^7: \quad \text{if } x \text{ is near 2.5 then } f(x) \text{ is near 0.85}
$$

\((3.43)\)

where $y^1 = -0.85$, $y^2 = -0.64$, $y^3 = -0.24$, $y^4 = 0$, $y^5 = 0.24$, $y^6 = 0.64$ and $y^7 = 0.85$. 

\[\text{Figure 3.10: The membership functions of 7 fuzzy sets}\]
Then the fuzzy system becomes

\[
\hat{f}(x) = \frac{\sum_{i=1}^{7} y_i [\mu_{A_i}(x)]}{\sum_{i=1}^{7} [\mu_{A_i}(x)]} = \frac{-0.85e^{-(x+2.5)^2} - 0.64e^{-(x+1.5)^2} - 0.24e^{-(x+0.5)^2} +}{e^{-(x+2.5)^2} + e^{-(x+1.5)^2} + e^{-(x+0.5)^2} + e^{-x^2} +} \frac{0.24e^{-(x-0.5)^2} + 0.64e^{-(x-1.5)^2} + 0.85e^{-(x-2.5)^2}}{e^{-(x-0.5)^2} + e^{-(x-1.5)^2} + e^{-(x-2.5)^2}}
\]

(3.44)

In Figure 3.11, we find that \(|f(x) - \hat{f}(x)| < 0.15\) over the interval \([-3,3]\).

![Figure 3.11: The nonlinear function \(f(x)\) and the estimation \(\hat{f}(x)\) with 7 rules](image)

### 3.2 Adaptive Fuzzy Control

#### 3.2.1 Why Fuzzy Control?

Conventional control methods use mathematical models to control systems. Fuzzy control methods replace the mathematical models with fuzzy if-then rules and fuzzy membership functions to control systems. Both fuzzy and conventional control methods are designed to meet system requirements of stability and convergence.
CHAPTER 3. INTRODUCTION TO FUZZY CONTROL

When mathematical models are unknown or partially unknown, fuzzy control methods can use fuzzy systems to estimate the unknown models. This is called the model-free approach [14]. Conventional control methods can use adaptive control methods to achieve the model free approach. When system dynamics become more complex, nonlinear systems are difficult to handle by conventional control methods. From the universal approximation theorem in (3.38), fuzzy systems can approximate arbitrary nonlinear systems.

In practical problems, systems can be controlled perfectly by experts. Experts provide linguistic descriptions about systems. Conventional control methods cannot design controllers combined with linguistic information. When linguistic information is important for designing controllers, we need to design fuzzy controllers for our systems. Fuzzy control methods are easy to understand for designers. The design process of fuzzy controllers can be simplified with simple mathematical models.

3.2.2 Why Adaptive Fuzzy Control?

Adaptive control uses a learning method to self-learn the parameters of systems. For systems whose dynamics are varying, adaptive control can learn the parameters of system dynamics. In traditional adaptive control, we need some information about our system such as the structure of the system or the order of the system. In adaptive fuzzy control, we can deal with uncertain systems. Due to the linguistic characteristics, adaptive fuzzy controllers behave like operators: adaptively controlling the system under various conditions. Adaptive fuzzy control provides a good tool for making use of expert knowledge to adjust systems. This is important for a complex unknown system with changing dynamics.
3.2.3 Adaptive Fuzzy Control

We divide adaptive fuzzy control into two categories: direct adaptive fuzzy control and indirect adaptive fuzzy control. A direct adaptive fuzzy controller adjusts the parameters of the control input. An indirect adaptive fuzzy controller adjusts the parameters of the control system based on the estimated dynamics of the plant.

We define fuzzy systems as two different types. The first type of fuzzy systems is given by

\[
M_{f}(x) = \sum_{i=1}^{M} \theta_i \xi_i(x) = \theta^T \xi(x)
\]  

where \( \theta = (\theta^1, \ldots, \theta^M)^T \), \( \xi(x) = (\xi^1(x), \ldots, \xi^M(x))^T \), and \( \xi^i(x) = \prod_{l=1}^{n} \mu_{A_l^i}(x_l) \) defined in (3.34). \( \theta^1, \ldots, \theta^M \) are adjustable parameters.

From the universal approximation theorem, we know that we can find a fuzzy system to estimate any continuous function. For the first type of fuzzy systems, we can only adjust \( \theta^i \) in (3.45). We define \( \hat{f}(x|\theta) \) as the approximator of the real function \( f(x) \).

\[
\hat{f}(x|\theta) = \theta^T \xi(x)
\]

We define \( \theta^* \) as the values for the minimum error:

\[
\theta^* = \arg \min_{\theta \in \Omega} \sup_{x \in U} |\hat{f}(x|\theta) - g(x)|
\]
where $\Omega$ is a constraint set for $\theta$. For specific $x$, $\sup_{x \in U} |\hat{f}(x|\theta^*) - f(x)|$ is the minimum approximation error we can get.

In Example 3.3, we assume that the five linguistic descriptions (if-then rules) (3.41) are unknown to the designer. We use the first type of fuzzy systems (3.45) to estimate the nonlinear system (3.39). With the given membership functions in (3.40), we can rewrite (3.42) as

$$\hat{f}(x|\theta) = \theta^T \xi(x)$$

$$= \sum_{i=1}^{5} \frac{\theta_i \mu_{A_i}(x)}{\sum_{i=1}^{5} \mu_{A_i}(x)}$$

$$= \frac{\theta_1 e^{-(x+1.5)^2} + \theta_2 e^{-(x+0.5)^2} + \theta_3 e^{-x^2} + \theta_4 e^{-(x-0.5)^2} + \theta_5 e^{-(x-1.5)^2}}{e^{-(x+1.5)^2} + e^{-(x+0.5)^2} + e^{-x^2} + e^{-(x-0.5)^2} + e^{-(x-1.5)^2}}$$

where $\theta_1, \cdots, \theta_5$ are adjusted by an adaptation law. The adaptation law is designed to minimize the parameter errors of $\theta - \theta^*$.

### 3.3 Summary

In this chapter, we introduced basic concepts of fuzzy logic and fuzzy systems. From the universal approximation theorem, we can use fuzzy systems to estimate system dynamics. We also introduced the fuzzy control method and the adaptive fuzzy control method. We introduced two types of fuzzy systems to adaptively control nonlinear systems by adjusting the parameters of fuzzy systems.
Chapter 4

Adaptive Fuzzy Sliding Mode Control

Classical sliding mode control has difficulty in handling unstructured model uncertainties. One can overcome this problem by combining sliding mode control and fuzzy systems together. The addition of an adaptation law to a fuzzy sliding mode controller helps improve the system’s tracking performance by online tuning the parameters of the fuzzy rules. Five different adaptive fuzzy sliding mode control algorithms applied to an m-link robotic manipulator are introduced: Yoo and Ham’s algorithm [3], Guo and Woo’s algorithm, revised Lin and Hsu’s algorithm, revised Shahnazi et al.’s algorithm and Medhaffar et al.’s algorithm.

4.1 Description of Algorithms

The dynamic equation of an m-link robotic manipulator is [11]

\[ M(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau \]  

(4.1)
where \( q = [q_1, \cdots, q_m]^T \) is an \( m \times 1 \) vector of joint position, \( M(q) \) is an \( m \times m \) inertial matrix, \( C(q, \dot{q}) \) is an \( m \times 1 \) matrix of Coriolis and centrifugal forces, \( G(q) \) is an \( m \times 1 \) gravity vector and \( \tau = [\tau_1, \cdots, \tau_m]^T \) is an \( m \times 1 \) vector of joint torques.

### 4.1.1 Yoo and Ham's Algorithm

Yoo and Ham proposed an adaptive fuzzy sliding mode control scheme applied to a robotic manipulator described in [3]. A MIMO (multi-input multi-output) fuzzy system is designed to compensate the uncertainties of the robot manipulator. The parameters of the fuzzy system are adjusted by adaptation laws.

We rewrite (4.1) as

\[
\tau = M(q)\dot{q} + C_1(q, \dot{q})\ddot{q} + G(q)
\]  

(4.2)

The tracking error and the sliding surface are defined as

\[
e = q - q_d
\]  

(4.3)

\[
s = \dot{e} + \lambda e
\]  

(4.4)

We define the reference state as

\[
\dot{q}_r = \dot{q} - s = \dot{q}_d - \lambda e
\]  

(4.5)

\[
\ddot{q}_r = \ddot{q} - \ddot{s} = \ddot{q}_d - \lambda \ddot{e}
\]  

(4.6)

The general MIMO if-then rules are given by [3]

\[
R^l : \text{If } x_1 \text{ is } A^l_1, x_2 \text{ is } A^l_2, \cdots, x_n \text{ is } A^l_n, \text{ then } y_1 \text{ is } B^l_1, \cdots, y_m \text{ is } B^l_m.
\]  

(4.7)

where \( l = 1, 2, \cdots, M \) are fuzzy if-then rules; \( x = (x_1, \cdots, x_n)^T \) and \( y = (y_1, \cdots, y_m)^T \) are the input and output vectors of the fuzzy system. The MIMO fuzzy system is defined as

\[
F(x) = \Theta^T \xi(x)
\]  

(4.8)
where

$$\Theta^T = (\theta_1, \cdots, \theta_m)^T = \begin{bmatrix} \theta_1^1, \theta_1^2, \cdots, \theta_1^M \\ \theta_2^1, \theta_2^2, \cdots, \theta_2^M \\ \vdots \\ \theta_m^1, \theta_m^2, \cdots, \theta_m^M \end{bmatrix} ,$$

(4.9)

$$\xi(x) = (\xi^1(x), \cdots, \xi^M(x))^T , \xi^i(x) = \prod_{i=1}^{n} \mu_{A_i}(x_i) / \sum_{i=1}^{M} (\prod_{i=1}^{n} \mu_{A_i}(x_i)) ,$$

and $$\mu_{A_i}(x_i)$$ is defined in (3.35). To reduce the number of fuzzy rules, we divide the fuzzy system into three parts:

$$F^1(q, \dot{q}) = \Theta^T \xi(q, \dot{q})$$

(4.10)

$$F^2(q, \ddot{q}) = \Theta^T \xi(q, \ddot{q})$$

(4.11)

$$F^3(q, \dot{q}) = \Theta^T \xi(q, \dot{q})$$

(4.12)

The control input is given by

$$\tau = \ddot{\dot{q}} + \dot{q}, \dot{q} + G + F^1(q, \dot{q}) + F^2(q, \ddot{q}) +$$

$$F^3(q, \dot{q}) - K_D \dot{s} - W \text{sgn}(s)$$

(4.13)

where $$\ddot{\dot{q}}, \dot{q}, G$$ are the estimations of $$M(q)$$ and $$C(q, \dot{q}); K_D = \text{diag} [K_{D1}, \cdots, K_{Dm}]$$ and $$K_{D1}, \cdots, K_{Dm}$$ are positive constants; $$W = \text{diag} [W_1, \cdots, W_m]$$ and $$W_1, \cdots, W_m$$ are positive constants. The adaptation law is given by

$$\dot{\theta}_j^1 = -\Gamma_{1j} \xi(q, \dot{q})$$

(4.14)

$$\dot{\theta}_j^2 = -\Gamma_{2j} \xi(q, \ddot{q})$$

$$\dot{\theta}_j^3 = -\Gamma_{3j} \xi(q, \dot{q})$$

where $$j = 1, \cdots, m$$ and $$\Gamma_{1j} - \Gamma_{3j}$$ are positive diagonal matrices.
4.1.2 Guo and Woo's Algorithm

Guo and Woo reduced the MIMO fuzzy system in Section 4.1.1 to a SISO fuzzy system which can approximate the residual coupling effect and alleviate the chattering.

The robotic manipulator used in Guo and Woo's algorithm is defined in (4.2). The tracking error and the sliding surface are defined as:

\[ e = q - q_d \]
\[ s = \dot{s} + \lambda e. \]

We introduce the reference state as

\[ \dot{q}_r = \dot{q} - s = \dot{q}_d - \lambda e \quad (4.15) \]
\[ \ddot{q}_r = \ddot{q} - \dot{s} = \ddot{q}_d - \lambda \dot{e}. \quad (4.16) \]

The control input is given by

\[ \tau = \dot{M}\ddot{q}_r + \dot{C}_r \dot{q}_r + \dot{G} - As - K \quad (4.17) \]

where \( A = \text{diag}[a_1, \cdots, a_m] \) and \( a_1, \cdots, a_m \) are positive constants; \( K = [K_1, \cdots, K_m]^T \) and \( K_j \) is defined as the fuzzy gain estimated by fuzzy systems.

The fuzzy if-then rules for the \( j \)th joint of the robotic manipulator are defined as

\[ R^{(j)}: \text{If } s_j \text{ is } A^{(j)}_l, \text{ then } y \text{ is } B^{(j)}_l. \quad (4.18) \]

where \( j = 1, \cdots, m \) and \( l = 1, \cdots, M \).

We define \( K_j \) by

\[ K_j = \frac{\sum_{l=1}^{M} \theta_j^l [\mu_{A_j}(s_j)]}{\sum_{l=1}^{M} [\mu_{A_j}(s_j)]} = \theta_j^T \xi_j(s_j) \quad (4.19) \]
where

\[ \xi_j(s_j) = \left[ \xi_j^1(s_j), \xi_j^2(s_j), \ldots, \xi_j^M(s_j) \right]^T, \quad (4.20) \]

\[ \xi_j^i(s_j) = \frac{\mu_j^i(s_j)}{\sum_{i=1}^{M} \mu_j^i(s_j)} . \quad (4.21) \]

The membership function \( \mu_j^i(s_j) \) is a Gaussian membership function defined in (3.35):

\[ \mu_j^i(s_j) = \exp \left[ -\left( \frac{s_j - \alpha_j^i}{\sigma_j^i} \right)^2 \right] \quad (j = 1, \ldots, m). \quad (4.22) \]

The adaptation law is expressed by

\[ \dot{\phi}_j = \gamma_{\phi_j} \xi_j(s_j) \quad (4.23) \]

where the constants \( \gamma_{\phi_j} \) is a positive constant.

### 4.1.3 Revised Lin and Hsu’s Algorithm

In Sections 4.1.1 and 4.1.2, we need to define the premise part of fuzzy rules for each input variable of the fuzzy system. Lin and Hsu’s tuning methodology can online adjust both the premise and the consequence parts of the fuzzy rules. Since Lin and Hsu’s algorithm is specifically applied to an induction servomotor drive, we need to revise this algorithm for the control of robotic manipulators which is referred to as the revised Lin and Hsu’s algorithm.

We define the same tracking error, sliding surface and reference state as in Sections 4.1.1 and 4.1.2. The control input is given by

\[ \tau = \hat{M} \ddot{q}_r + \dot{C}_1 \dot{q}_r + \dot{G} - \hat{F}(s) - F_{cp}(s) \quad (4.24) \]

where \( \hat{F}(s) = [\hat{f}_1(s_1), \ldots, \hat{f}_m(s_m)]^T \) and \( F_{cp}(s) = [f_{cp_1}(s_1), \ldots, f_{cp_m}(s_m)]^T \). We define \( F = \Delta \hat{M} \ddot{q}_r + \Delta C_1 \dot{q}_r + \Delta G \) where \( F = [f_1, \ldots, f_m]^T \), \( \Delta \hat{M} = \hat{M} - M \), \( \Delta C_1 = \dot{C}_1 - C_1 \) and \( \Delta G = \dot{G} - G \).
CHAPTER 4. ADAPTIVE FUZZY SLIDING MODE CONTROL

From Universal Approximation Theorem in (3.38), there exists an optimal fuzzy controller \( f_j^* (s_j) \) such that:

\[
f_j = f_j^* (s_j) + \Delta_j
\]

(4.25)

where \( \Delta_j \) is the minimum approximation error.

The fuzzy if-then rules are given in (4.18). In (4.21), we assume \( \sum_{i=1}^{M} \mu_{A_i^j}(s_j) = 1 \) and \( \xi_j^i(s_j) \) becomes

\[
\xi_j^i(s_j) = \frac{\mu_{A_i^j}(s_j)}{\sum_{i=1}^{M} [\mu_{A_i^j}(s_j)]}
\]

(4.26)

where we define \( \Phi_j^i = \mu_{A_i^j}(s_j) \). The membership function \( \mu_{A_i^j}(s_j) \) is a Gaussian membership function represented by

\[
\mu_{A_i^j}(s_j) = \exp\left[-\left(\sigma_j^i(s_j - a_j^i)^2\right)\right].
\]

(4.27)

Then the fuzzy estimator \( \hat{f}_j(s_j) \) is given as

\[
\hat{f}_j(s_j) = \theta_j^T \Phi_j
\]

(4.28)

where \( \theta_j = [\theta_j^1, \cdots, \theta_j^M]^T \), \( \Phi_j = [\Phi_j^1, \Phi_j^2, \cdots, \Phi_j^M]^T \). We define \( \tilde{f}_j \) such that

\[
\tilde{f}_j = f_j - \hat{f}_j(s_j)
\]

\[
= f_j^* (s_j) - \hat{f}_j(s_j) + \Delta_j
\]

\[
= \theta_j^*^T \Phi_j^* - \theta_j^T \Phi_j + \Delta_j
\]

(4.29)

where \( \theta_j^* \) and \( \Phi_j^* \) are the optimal values based on Universal Approximation Theorem in (3.48). We define \( \tilde{\theta}_j = \theta_j^* - \theta_j \), \( \Phi_j = \Phi_j^* - \Phi_j \) and (4.29) is rewritten as

\[
\tilde{f}_j = (\theta_j + \tilde{\theta}_j)^T (\Phi_j + \tilde{\Phi}_j) - \theta_j^T \Phi_j + \Delta_j
\]

\[
= \theta_j^T \Phi_j + \tilde{\theta}_j^T \Phi_j + \theta_j^T \tilde{\Phi}_j + \Delta_j
\]

(4.30)
We take Taylor series expansion of \( \Phi_j \) around two vectors \( \alpha_j \) and \( \sigma_j \) where \( \alpha_j = [\alpha_j^1, \ldots, \alpha_j^M]^T \) and \( \sigma_j = [\sigma_j^1, \ldots, \sigma_j^M]^T \) (\( \alpha_j^i \) and \( \sigma_j^i \) are defined in (4.27)):
\[
\Phi_j^* = \Phi_j + \frac{\partial \Phi_j}{\partial \alpha_j} \dot{\alpha}_j + \frac{\partial \Phi_j}{\partial \sigma_j} \dot{\sigma}_j + \text{h.o.t.} \tag{4.31}
\]
where \( \dot{\alpha}_j = \alpha_j^* - \alpha_j, \dot{\sigma}_j = \sigma_j^* - \sigma_j \) and \( \text{h.o.t.} \) denotes the higher order terms. We rewrite (4.31) as
\[
\begin{align*}
\Phi_j &= \frac{\partial \Phi_j}{\partial \alpha_j} \dot{\alpha}_j + \frac{\partial \Phi_j}{\partial \sigma_j} \dot{\sigma}_j + \text{h.o.t.} \\
&= B_j \dot{\alpha}_j + C_j \dot{\sigma}_j + \text{h.o.t.} \tag{4.32}
\end{align*}
\]
where
\[
B_j = \begin{bmatrix}
\frac{\partial \Phi_j^1}{\partial \alpha_j^1} & \frac{\partial \Phi_j^2}{\partial \alpha_j^1} & \cdots & \frac{\partial \Phi_j^M}{\partial \alpha_j^1} \\
\frac{\partial \Phi_j^1}{\partial \alpha_j^2} & \frac{\partial \Phi_j^2}{\partial \alpha_j^2} & \cdots & \frac{\partial \Phi_j^M}{\partial \alpha_j^2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial \Phi_j^1}{\partial \alpha_j^M} & \frac{\partial \Phi_j^2}{\partial \alpha_j^M} & \cdots & \frac{\partial \Phi_j^M}{\partial \alpha_j^M}
\end{bmatrix}, \quad C_j = \begin{bmatrix}
\frac{\partial \Phi_j^1}{\partial \sigma_j^1} & \frac{\partial \Phi_j^2}{\partial \sigma_j^1} & \cdots & \frac{\partial \Phi_j^M}{\partial \sigma_j^1} \\
\frac{\partial \Phi_j^1}{\partial \sigma_j^2} & \frac{\partial \Phi_j^2}{\partial \sigma_j^2} & \cdots & \frac{\partial \Phi_j^M}{\partial \sigma_j^2} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial \Phi_j^1}{\partial \sigma_j^M} & \frac{\partial \Phi_j^2}{\partial \sigma_j^M} & \cdots & \frac{\partial \Phi_j^M}{\partial \sigma_j^M}
\end{bmatrix}. \tag{4.33}
\]

We substitute (4.32) into (4.30):
\[
\begin{align*}
\tilde{f}_j &= \theta_j^T (B_j \dot{\alpha}_j + C_j \dot{\sigma}_j + \text{h.o.t.}) + \tilde{\theta}_j^T \Phi_j + \tilde{\theta}_j^T \tilde{\Phi}_j + \Delta_j \\
&= \theta_j^T B_j \dot{\alpha}_j + \theta_j^T C_j \dot{\sigma}_j + \tilde{\theta}_j^T \Phi_j + \theta_j^T \text{h.o.t.} + \tilde{\theta}_j^T \tilde{\Phi}_j + \Delta_j \\
&= \theta_j^T B_j \dot{\alpha}_j + \theta_j^T C_j \dot{\sigma}_j + \tilde{\theta}_j^T \Phi_j + \varepsilon_j \tag{4.34}
\end{align*}
\]
where \( \varepsilon_j = \theta_j^T \text{h.o.t.} + \tilde{\theta}_j^T \tilde{\Phi}_j + \Delta_j \) is assumed to be bounded by \(|\varepsilon_j| \leq E_j\). \( E_j \) is a constant and the value of \( E_j \) is uncertain to the designer. We define \( E^\ast \) as the real value and the estimation error is given by
\[
\tilde{E}_j = E_j^\ast - E_j \tag{4.35}
\]

We produce an adaptation law to online tune the following parameters: \( \theta_j \) in (4.28),
\( a_j^i \), \( a_j \) in (4.27) and the bound \( E_j \) in (4.35). The adaptation laws are expressed as

\[
\dot{\theta}_j = \eta_j s_j \Phi_j
\]

\( \alpha_j = \eta_j s_j B_j^T \theta_j \)

\( \sigma_j = \eta_j s_j C_j^T \theta_j \)

\[
f_{cp}(s_j) = E_j \text{sgn}(s_j)
\]

\[
\dot{E}_j = \eta_j |s_j|
\]

where \( \eta_j \) and \( \eta_j \) are positive constants; \( \theta_j = [\theta_1^j, \theta_2^j, \ldots, \theta_j^M]^T \), \( \alpha_j = [\alpha_1^j, \alpha_2^j, \ldots, \alpha_j^M]^T \), \( \sigma_j = [\sigma_1^j, \sigma_2^j, \ldots, \sigma_j^M]^T \), \( B_j \), \( C_j \) are given in (4.33); \( f_{cp}(s_j) \) is the compensation term defined in (4.24).

### 4.1.4 Revised Shahnazi et al.'s Algorithm

Shahnazi et al. designed a PI controller inside a boundary layer in order to attenuate the chattering that occurred in Lin and Hsu's method [6]. Similar to Lin and Hsu's algorithm, Shahnazi et al.'s algorithm deals with a SISO nonlinear servomotor which is not applicable to the robotic manipulator. Therefore, we introduced the modified version of this method which is referred to as the revised Shahnazi et al.'s algorithm.

The tracking error, sliding surface and reference state are the same as in Sections 4.1.1-4.1.3. The control law is defined in (4.24):

\[
\tau = \dot{M} \ddot{q} + \dot{C} \dot{q} + \dot{G} - \dot{F}(s) - F_{cp}(s)
\]

The adaptation law changes to

\[
\dot{\theta}_j = \eta_j s_j \Phi_j(s_j)
\]

\[\dot{\alpha}_j = \eta_j s_j B_j^T \theta_j \]

\[\dot{\sigma}_j = \eta_j s_j C_j^T \theta_j \]

\[
f_{cp}(s_j) = \beta_j^T \Psi_j(s_j)
\]
where $B_j$ and $C_j$ are given in (4.33); $\eta_{j1}, \cdots, \eta_{j5}$ are positive constants; $\Psi_j(s_j) = [s_j \int s_j dt]^T$ and $\beta_j = [K_{Pj} K_{Ij}]^T$. We define a boundary layer with the thickness of $\varphi_i$. We let $f_{cpj} = \beta_j^T \Psi_j(s_j)$ in (4.45) when $|s_j| < \varphi_j$ and $|f_{cpj}| = E_j$ when $|s_j| > \varphi_j$.

### 4.1.5 Medhaffar et al.’s Algorithm

Medhaffar et al. designed an indirect adaptive fuzzy sliding mode control algorithm applied to a two-link robotic manipulator. This algorithm is developed based on Yoo and Ham’s approach [2]. The fuzzy systems used in this approach are MIMO fuzzy systems whose fuzzy rules are reduced by introducing sliding surfaces as the inputs.

We rewrite (4.1) as

$$
\ddot{q} = -M^{-1}(q)C(q, \dot{q}) - M^{-1}(q)G(q) + M^{-1}(q)\tau
$$

$$
= F(q, \dot{q}) + B(q)\tau \tag{4.48}
$$

where $F(q, \dot{q}) = -M^{-1}(q)C(q, \dot{q}) - M^{-1}(q)G(q)$ and $B(q) = M^{-1}(q)$. We define the sliding surface as

$$
s = \dot{e} + \lambda e \tag{4.49}
$$

where $e = q - q_d$. We take the derivative of $s$ and get

$$
\dot{s} = \ddot{e} + \lambda \dot{e}
$$

$$
= F(q, \dot{q}) + B(q)\tau - \ddot{q}_d + \lambda \dot{e} \tag{4.50}
$$

We let $\dot{s} = 0$ in (4.50) and get the ideal sliding mode control law $\tau^*$:

$$
\tau^* = B(q)^{-1}(\ddot{q}_d - F(q, \dot{q}) - \lambda \dot{e} - K_M \text{sgn}(s)) \tag{4.51}
$$
where \( K_M = \text{diag}[K_{M1}, \ldots, K_{Mm}] \) and \( K_{M1}, \ldots, K_{Mm} \) are positive constants. The control law is defined as

\[
\tau = \hat{B}^{-1}(s|\theta_B)(\hat{q}_d - \hat{F}(s|\theta_F) - \lambda \hat{e} - \hat{H}(s|\theta_H))
\]

(4.52)

where \( \hat{F}(s|\theta_F), \hat{B}(s|\theta_B) \) and \( \hat{H}(s|\theta_H) \) are fuzzy systems designed to estimate \( F(q, \dot{q}) \), \( B(q) \) and \( K_M \text{sgn}(s) \) in (4.51). The fuzzy system \( \hat{F}(s|\theta_F) \) is given by \( \hat{F}(s|\theta_F) = [\hat{f}_1(s|\theta_{f1}), \ldots, \hat{f}_m(s|\theta_{fm})]^T \) where

\[
\hat{f}_j(s|\theta_{fj}) = \sum_{i=1}^M \theta^j_{fi} \left( \prod_{i=1}^m \mu_{Al}^j(s_i) \right) = \theta^T_{fj} \xi(s)
\]

(4.53)

The fuzzy system \( \hat{B}(s|\theta_B) \) is defined as

\[
\hat{B}(s|\theta_B) = \begin{bmatrix}
\hat{b}_{11}(s|\theta_{b1}), & \ldots, & \hat{b}_{1m}(s|\theta_{b1m}) \\
\vdots & & \vdots \\
\hat{b}_{m1}(s|\theta_{b1}), & \ldots, & \hat{b}_{mm}(s|\theta_{bmm})
\end{bmatrix}
\]

(4.54)

where

\[
\hat{b}_{jk}(s|\theta_{bjk}) = \sum_{i=1}^M \theta^j_{bjk} \left( \prod_{i=1}^m \mu_{Al}^j(s_i) \right) = \theta^T_{bjk} \xi(s)
\]

(4.55)

The fuzzy system \( \hat{H}(s|\theta_H) \) is selected as \( \hat{H}(s|\theta_H) = [\hat{h}_1(s_1|\theta_{h1}), \ldots, \hat{h}_m(s_m|\theta_{hm})]^T \) where

\[
\hat{h}_j(s_j|\theta_{hj}) = \sum_{i=1}^M \theta^j_{hj} \mu_{Al}^j(s_j) = \theta^T_{hj} \xi(s_j)
\]

(4.56)

The adaptation laws are expressed by

\[
\dot{\theta}_{fj} = \chi_{fj} s_j \xi(s)
\]

\[
\dot{\theta}_{bjk} = \chi_{bjk} s_j \xi(s)
\]

\[
\dot{\theta}_{hj} = \chi_{hj} s_j \xi(s_j)
\]

(4.57)
where $\chi_{f_i}$, $\chi_{b_i}$, and $\chi_{b_j}$ are positive constants.

## 4.2 Stability Proof of Algorithms

### 4.2.1 Stability Proof of Yoo and Ham's Algorithm

The Lyapunov function candidate is presented as

$$V = \frac{1}{2} s^T M s + \frac{1}{2} \sum_{j=1}^{m} \frac{1}{\Gamma_{1j}} \phi_j^T \phi_j + \frac{1}{2} \sum_{j=1}^{m} \frac{1}{\Gamma_{2j}} \phi_j^2 \phi_j^2 + \frac{1}{2} \sum_{j=1}^{m} \frac{1}{\Gamma_{3j}} \phi_j^3 \phi_j^3$$  \hspace{1cm} (4.58)

where $\phi^1_j = \theta^1_j - \theta^1_j$, $\phi^2_j = \theta^2_j - \theta^2_j$, and $\phi^3_j = \theta^3_j - \theta^3_j$. We define

$$F(q, \dot{q}, \ddot{q}, \bar{q}) = F^1(q, \dot{q}) + F^2(q, \ddot{q}) + F^3(q, \bar{q})$$  \hspace{1cm} (4.59)

From (4.2) and (4.13), we get

$$M(q) \dot{q} + C_1(q, \dot{q}) \dot{q} + G(q) = \dot{\bar{q}} + \dot{\bar{r}} + \dot{G} + F(q, \dot{q}, \ddot{q}, \bar{q}) - K_D s - W sgn(s)$$  \hspace{1cm} (4.60)

Since $\dot{q}_r = \dot{q} - s$ and $\ddot{q}_r = \ddot{q} - s$ in (4.5) and (4.6), we get

$$M \dot{s} + (C_1 + K_D) s + W sgn(s) = -\Delta F + F(q, \dot{q}, \ddot{q}, \bar{q}) .$$  \hspace{1cm} (4.61)

Then $M \dot{s} + C_1 s$ can be written as

$$M \dot{s} + C_1 s = -\Delta F + F(q, \dot{q}, \ddot{q}, \bar{q}) - K_D s - W sgn(s)$$  \hspace{1cm} (4.62)

where $\Delta F = \dot{\bar{q}} + \dot{\bar{r}} + \dot{G}$, $\bar{M} = M - \dot{M}$, $\dot{C}_1 = C_1 - \dot{C}_1$, and $\dot{G} = G - \dot{G}$.

The derivative of $V$ is

$$\dot{V} = s^T \dot{M} s + \frac{1}{2} s^T \dot{M} s + \sum_{j=1}^{m} \frac{1}{\Gamma_{1j}} \phi_j^T \dot{\phi}_j + \sum_{j=1}^{m} \frac{1}{\Gamma_{2j}} \phi_j^2 \dot{\phi}_j^2 + \sum_{j=1}^{m} \frac{1}{\Gamma_{3j}} \phi_j^3 \dot{\phi}_j^3$$  \hspace{1cm} (4.63)
We know that $s^T M s + \frac{1}{2} s^T \dot{M} s = s^T (M s + C_1 s)$ from (2.38). Then

$$
\dot{V} = -s^T [K_D s + W \text{sgn}(s) + \Delta F - F(q, \dot{q}, \dot{q}_r, \ddot{q})] + \sum_{j=1}^{m} \frac{1}{\Gamma_{1j}} \phi_j^1 \dot{\phi_j}^1 + \sum_{j=1}^{m} \frac{1}{\Gamma_{2j}} \phi_j^2 \dot{\phi_j}^2 + \sum_{j=1}^{m} \frac{1}{\Gamma_{3j}} \phi_j^3 \dot{\phi_j}^3
$$

(4.64)

We define the minimum approximation error as

$$
\omega = \Delta F - [F^1(q, \dot{q}|\Theta^1) + F^2(q, \dot{q}_r|\Theta^2) + F^3(q, \ddot{q}|\Theta^3)]
$$

(4.65)

We plug (4.65) into (4.64):

$$
\dot{V} = -s^T [K_D s + W \text{sgn}(s) + \Delta F - F(q, \dot{q}, \dot{q}_r, \ddot{q})] + \sum_{j=1}^{m} \frac{1}{\Gamma_{1j}} \phi_j^1 \dot{\phi_j}^1 + \sum_{j=1}^{m} \frac{1}{\Gamma_{2j}} \phi_j^2 \dot{\phi_j}^2 + \sum_{j=1}^{m} \frac{1}{\Gamma_{3j}} \phi_j^3 \dot{\phi_j}^3
$$

$$
= -s^T [K_D s + W \text{sgn}(s) + \omega + F^1(q, \dot{q}|\Theta^1) + F^2(q, \dot{q}_r|\Theta^2) + F^3(q, \ddot{q}|\Theta^3)] + \sum_{j=1}^{m} \frac{1}{\Gamma_{1j}} \phi_j^1 \dot{\phi_j}^1
$$

$$
+ \sum_{j=1}^{m} \frac{1}{\Gamma_{2j}} \phi_j^2 \dot{\phi_j}^2 + \sum_{j=1}^{m} \frac{1}{\Gamma_{3j}} \phi_j^3 \dot{\phi_j}^3
$$

$$
= -s^T K_D s - s^T W \text{sgn}(s) - s^T \omega - \sum_{j=1}^{m} s_j \phi_j^T \xi(q, \dot{q}) - \sum_{j=1}^{m} s_j \phi_j^2 \xi(q, \dot{q}_r)
$$

$$
- \sum_{j=1}^{m} s_j \phi_j^3 \xi(q, \ddot{q}) + \sum_{j=1}^{m} \frac{1}{\Gamma_{1j}} \phi_j^1 \dot{\phi_j}^1 + \sum_{j=1}^{m} \frac{1}{\Gamma_{2j}} \phi_j^2 \dot{\phi_j}^2 + \sum_{j=1}^{m} \frac{1}{\Gamma_{3j}} \phi_j^3 \dot{\phi_j}^3
$$

$$
= -s^T K_D s - s^T W \text{sgn}(s) - s^T \omega - \sum_{j=1}^{m} \phi_j^1 \dot{\phi_j}^1 + \sum_{j=1}^{m} \phi_j^2 \dot{\phi_j}^2 + \sum_{j=1}^{m} \phi_j^3 \dot{\phi_j}^3
$$

$$
= -s^T K_D s - s^T W \text{sgn}(s) - s^T \omega - \sum_{j=1}^{m} \phi_j^1 \dot{\phi_j}^1 + \sum_{j=1}^{m} \phi_j^2 \dot{\phi_j}^2 + \sum_{j=1}^{m} \phi_j^3 \dot{\phi_j}^3
$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The adaptation laws are chosen as (4.14). Then \( \hat{V} \) becomes

\[
\hat{V} = -s^T K_D s + s^T W \text{sgn}(s) - s^T \omega
\]

\[
= - \sum_{j=1}^{m} (s_j^2 K_{Dj} + W_j |s_j| + s_j \omega_j)
\]

\[
= - \sum_{j=1}^{m} [s_j (s_j K_{Dj} + \omega_j) + W_j |s_j|]
\]

(4.66)

Since \( \omega_j \) can be as small as possible, we can find \( K_{Dj} \) that \( |s_j K_{Dj}| > |\omega_j| (s_j \neq 0) \).

Therefore, we can get \( s_j (s_j K_{Dj} + \omega_j) > 0 \) for \( s_j \neq 0 \) and \( \hat{V} < 0 (s \neq 0) \).

### 4.2.2 Stability Proof of Guo and Woo’s Algorithm

The Lyapunov function candidate is given by

\[
\hat{V} = \frac{1}{2} s^T M s + \frac{1}{2} \sum_{j=1}^{m} \frac{1}{\gamma_{sj}} \phi_j^T \phi_j
\]

(4.67)

where \( \phi_j = \theta_j^* - \theta_j \). The derivative of \( V \) is

\[
\dot{V} = s^T M \dot{s} + \frac{1}{2} s^T \dot{M} s + \sum_{j=1}^{m} \frac{1}{\gamma_{sj}} \phi_j^T \dot{\phi}_j
\]

(4.68)

Since \( \dot{M} - 2C_1 \) is a skew-symmetric matrix, we can get \( s^T M \dot{s} + \frac{1}{2} s^T \dot{M} s = s^T (M \dot{s} + C_1 s) \). From (4.2) and (4.17), we get

\[
M(q) \ddot{q} + C_1(q, \dot{q}) \dot{q} + G(q) = \dot{\hat{M}} \ddot{q}_r + \dot{\hat{C}}_1 \dot{q}_r + \dot{\hat{G}} - A s - K
\]

(4.69)

Since \( \ddot{q}_r = \ddot{q} - s \) and \( \dot{q}_r = \dot{q} - \dot{s} \) in (4.15) and (4.16), we get

\[
M \dot{s} + (C_1 + A) s = \Delta f - K
\]

(4.70)
where \( \Delta f = \Delta M \ddot{q}_r + \Delta C_1 \dot{q}_r + \Delta G \), \( \Delta M = \dot{M} - M \), \( \Delta C_1 = \dot{C}_1 - C_1 \) and \( \Delta G = \dot{G} - G \). Then \( \dot{V} \) becomes

\[
\dot{V} = s^T (M \dot{s} + C_1 s) + \sum_{j=1}^{m} \frac{1}{\gamma_{sj}} \phi_j^T \phi_j
\]

\[
= s^T (-A s + \Delta f - K) + \sum_{j=1}^{m} \frac{1}{\gamma_{sj}} \phi_j^T \phi_j
\]

\[
= \sum_{j=1}^{m} [s_j (\Delta f_j - K_j)] - s^T A s + \sum_{j=1}^{m} \frac{1}{\gamma_{sj}} \phi_j^T \phi_j
\]

\[
= \sum_{j=1}^{m} (s_j [\Delta f_j - (\theta_j^*)^T \xi_j(s_j)]) - s^T A s + \sum_{j=1}^{m} \frac{1}{\gamma_{sj}} \phi_j^T \phi_j
\]

\[
= \sum_{j=1}^{m} (s_j [\Delta f_j - (\theta_j^*)^T \xi_j(s_j)]) - s^T A s + \sum_{j=1}^{m} \left( \frac{1}{\gamma_{sj}} \phi_j^T \phi_j \right)
\]

We choose the adaptation law \( \dot{\theta}_j = \gamma_{sj} s_j \xi_j(s_j) \). Since \( \dot{\phi}_j = -\dot{\theta}_j = -\gamma_{sj} s_j \xi_j(s_j) \), \( \dot{V} \) becomes

\[
\dot{V} = \sum_{j=1}^{m} (s_j [\Delta f_j - (\theta_j^*)^T \xi_j(s_j)]) - s^T A s \quad (4.71)
\]

We define the minimum approximation error as

\[
\omega_j = \Delta f_j - (\theta_j^*)^T \xi_j(s_j) \quad (4.72)
\]

Then \( \dot{V} \) changes to

\[
\dot{V} = \sum_{j=1}^{m} s_j \omega_j - s^T A s
\]

\[
\leq \sum_{j=1}^{m} |s_j| |\omega_j| - s^T A s
\]

\[
= \sum_{j=1}^{m} (|s_j| |\omega_j| - a_j s_j^2)
\]

\[
= \sum_{j=1}^{m} (|s_j| (|\omega_j| - a_j |s_j|)) \quad (4.73)
\]
CHAPTER 4. ADAPTIVE FUZZY SLIDING MODE CONTROL

According to Universal Approximation Theorem in (3.38), the minimum approximation error $\omega_j$ is as small as possible. We can simply pick $a_j$ to make $a_j|s_j| > |\omega_j|(s_j \neq 0)$. Then we get $\dot{V} < 0$ for $s \neq 0$.

4.2.3 Stability Proof of Revised Lin and Hsu's Algorithm

We define the following Lyapunov function candidate:

$$V = \frac{1}{2} s^T Ms + \frac{1}{2} \sum_{j=1}^{m} \left( \frac{\dot{E}_j^2}{\eta_{j1}} + \frac{\dot{\theta}_j^2}{\eta_{j2}} + \frac{\dot{\alpha}_j^2}{\eta_{j3}} + \frac{\dot{\sigma}_j^2}{\eta_{j4}} \right)$$

(4.74)

We take the derivative of $V$:

$$\dot{V} = s^T Ms + \frac{1}{2} \sum_{j=1}^{m} \left( \frac{\dot{E}_j^2}{\eta_{j1}} + \frac{\dot{\theta}_j^2}{\eta_{j2}} + \frac{\dot{\alpha}_j^2}{\eta_{j3}} + \frac{\dot{\sigma}_j^2}{\eta_{j4}} \right)$$

(4.75)

where $\dot{E}_j = E_j^* - E_j$, $\dot{\theta}_j = \theta_j^* - \theta_j$, $\dot{\alpha}_j = \alpha_j^* - \alpha_j$, $\dot{\sigma}_j = \sigma_j^* - \sigma_j$. From (4.2) and (4.24), we get

$$M(q)\ddot{q} + C_1(q, \dot{q})\dot{q} + G(q) = \hat{M}\ddot{q} + \hat{C}_1\dot{q} + \hat{G} - \hat{F}(s) - F_{cp}(s)$$

(4.76)

Since $\hat{M} - 2C_1$ is a skew-symmetric matrix, we can get $s^T Ms + \frac{1}{2}s^T Ms = s^T (Ms + C_1 s)$. From (4.15) and (4.16), we get

$$Ms + C_1 s = F - \hat{F}(s) - F_{cp}(s)$$

(4.77)

where $F = \Delta M \ddot{q} + \Delta C_1 \dot{q} + \Delta G$, $\Delta M = \hat{M} - M$, $\Delta C_1 = \hat{C}_1 - C_1$ and $\Delta G = \hat{G} - G$. Then $\dot{V}$ becomes

$$\dot{V} = s^T (Ms + C_1 s) - \sum_{j=1}^{m} \left( \frac{\dot{E}_j^2}{\eta_{j1}} + \frac{\dot{\theta}_j^2}{\eta_{j2}} + \frac{\dot{\alpha}_j^2}{\eta_{j3}} + \frac{\dot{\sigma}_j^2}{\eta_{j4}} \right)$$

$$= s^T (F - \hat{F}(s) - F_{cp}(s)) - \sum_{j=1}^{m} \left( \frac{\dot{E}_j^2}{\eta_{j1}} + \frac{\dot{\theta}_j^2}{\eta_{j2}} + \frac{\dot{\alpha}_j^2}{\eta_{j3}} + \frac{\dot{\sigma}_j^2}{\eta_{j4}} \right)$$

$$= \sum_{j=1}^{m} s_j (f_j - \hat{f}_j(s_j) - f_{cp}) - \sum_{j=1}^{m} \left( \frac{\dot{E}_j^2}{\eta_{j1}} + \frac{\dot{\theta}_j^2}{\eta_{j2}} + \frac{\dot{\alpha}_j^2}{\eta_{j3}} + \frac{\dot{\sigma}_j^2}{\eta_{j4}} \right)$$
\[
\sum_{j=1}^{m} s_j (\theta_j^T B_j \Delta \dot{\alpha}_j + \theta_j^T C_j \Delta \dot{\sigma}_j + \theta_j^T \Phi_j + \varepsilon_j - f_{cpj})
\]
\[
- \sum_{j=1}^{m} \left( \frac{\dot{E}_j \dot{E}_j}{\eta_{j1}} + \frac{\dot{\theta}_j \dot{\theta}_j}{\eta_{j2}} + \frac{\Delta \dot{\alpha}_j \dot{\alpha}_j}{\eta_{j3}} + \frac{\Delta \dot{\sigma}_j \dot{\sigma}_j}{\eta_{j4}} \right)
\]
\[
= \sum_{j=1}^{m} \left[ \theta_j^T \left( s_j \Phi_j - \frac{\dot{\theta}_j}{\eta_{j2}} \right) + \alpha_j^T \left( s_j B_j^T \theta_j - \frac{\dot{\alpha}_j}{\eta_{j3}} \right) + \sigma_j^T \left( s_j C_j^T \theta_j - \frac{\dot{\sigma}_j}{\eta_{j4}} \right) \right]
\]
\[
+ \sum_{j=1}^{m} \left( s_j \varepsilon_j - s_j f_{cpj} - \frac{\dot{E}_j \dot{E}_j}{\eta_{j1}} \right)
\]
(4.78)

We substitute the adaptation law (4.36)-(4.40) into (4.78):
\[
\dot{V} = \sum_{j=1}^{m} \left[ s_j \varepsilon_j - s_j E_j sgn(s_j) - \dot{E}_j |s_j| \right]
\]
\[
= \sum_{j=1}^{m} \left[ s_j \varepsilon_j - s_j E_j sgn(s_j) - (E_j^* - E_j) s_j sgn(s_j) \right]
\]
\[
= \sum_{j=1}^{m} \left[ s_j \varepsilon_j - E_j^* s_j sgn(s_j) \right]
\]
\[
\leq \sum_{j=1}^{m} \left[ |s_j| |\varepsilon_j| - E_j^* |s_j| \right]
\]
\[
= \sum_{j=1}^{m} \left[ |s_j| (|\varepsilon_j| - E_j^*) \right] \leq 0
\]
(4.79)

where \( \dot{V} \) is negative semidefinite. We define \( \dot{V}_j = |s_j(t)|(|\varepsilon_j| - E_j^*) \). From \( \dot{V}_j \leq 0 \), we can get \( s_j(t) \) is bounded. We assume \( |s_j(t)| \leq \eta_s \) and rewrite \( |s_j(t)|(E_j^* - |\varepsilon_j|) \leq -\dot{V}_j \) as
\[
|s_j(t)| \leq \frac{1}{E_j^*} |s_j(t)||\varepsilon_j| - \frac{1}{E_j^*} \dot{V}_j \leq \frac{\eta_s}{E_j^*} |\varepsilon_j| - \frac{1}{E_j^*} \dot{V}_j
\]
(4.80)

Then we take the integral on both sides of (4.80):
\[
\int_{0}^{t} |s_j(\nu)|d\nu \leq \frac{\eta_s}{E_j^*} \int_{0}^{t} |\varepsilon_j|d\nu + \frac{1}{E_j^*} (V_j(0) - V_j(t))
\]
\[
\leq \frac{\eta_s}{E_j^*} \int_{0}^{t} |\varepsilon_j|d\nu + \frac{1}{E_j^*} (|V_j(0)| + |V_j(t)|)
\]
(4.81)
CHAPTER 4. ADAPTIVE FUZZY SLIDING MODE CONTROL

If \( \varepsilon_j \in L_1 \), we can get \( s_j \in L_1 \) from (4.81). Since we can prove \( \dot{s}_j \) is bounded (see proof in [15]), we have \( \dot{s}_j \in L_\infty \). We introduce the following Barbalat's lemma [16].

**Lemma 4.1** Let \( \phi : R \rightarrow R \) be a uniformly continuous function on \([0, \infty)\). Suppose that \( \lim_{t \rightarrow \infty} \int_0^t \phi(\tau) d\tau \) exists and is finite. Then,

\[
\lim_{t \rightarrow \infty} \phi(t) = 0
\]

Therefore, by using Barbalat's lemma, we can get \( \lim_{t \rightarrow \infty} s_j(t) = 0 \) and \( \lim_{t \rightarrow \infty} e_j(t) = 0 \).

### 4.2.4 Stability Proof of Revised Shahnazi et al.'s Algorithm

We define the following Lyapunov function candidate:

\[
V = \frac{1}{2} s^T M s + \frac{1}{2} \sum_{j=1}^{m} \left( \frac{\tilde{E}_j^T \tilde{\theta}_j}{\eta_{j1}} + \frac{\tilde{\alpha}_j^T \tilde{\alpha}_j}{\eta_{j2}} + \frac{\tilde{\sigma}_j^T \tilde{\sigma}_j}{\eta_{j3}} + \frac{\tilde{\beta}_j^T \tilde{\beta}_j}{\eta_{j4}} \right) \quad (4.82)
\]

We take the derivative of \( V \):

\[
\dot{V} = s^T M \dot{s} + \frac{1}{2} s^T M s + \sum_{j=1}^{m} \left( \frac{\tilde{E}_j \dot{\tilde{E}}_j}{\eta_{j1}} + \frac{\tilde{\theta}_j \dot{\tilde{\theta}}_j}{\eta_{j2}} + \frac{\tilde{\alpha}_j \dot{\tilde{\alpha}}_j}{\eta_{j3}} + \frac{\tilde{\sigma}_j \dot{\tilde{\sigma}}_j}{\eta_{j4}} + \frac{\tilde{\beta}_j \dot{\tilde{\beta}}_j}{\eta_{j5}} \right) \quad (4.83)
\]

where \( \tilde{E}_j = E_j^* - E_j, \tilde{\theta}_j = \theta_j^* - \theta_j, \tilde{\alpha}_j = \alpha_j^* - \alpha_j, \tilde{\sigma}_j = \sigma_j^* - \sigma_j, \tilde{\beta}_j = \beta_j^* - \beta_j \). From (4.76) and (4.77), \( \dot{V} \) can be rewritten as

\[
\dot{V} = s^T (M \dot{s} + C_1 s) - \sum_{j=1}^{m} \left( \frac{\tilde{E}_j \dot{E}_j}{\eta_{j1}} + \frac{\tilde{\theta}_j \dot{\theta}_j}{\eta_{j2}} + \frac{\tilde{\alpha}_j \dot{\alpha}_j}{\eta_{j3}} + \frac{\tilde{\sigma}_j \dot{\sigma}_j}{\eta_{j4}} + \frac{\tilde{\beta}_j \dot{\beta}_j}{\eta_{j5}} \right)
\]

\[
= s^T (F - \dot{F}(s) - F_{cp}(s)) - \sum_{j=1}^{m} \left( \frac{\tilde{E}_j \dot{E}_j}{\eta_{j1}} + \frac{\tilde{\theta}_j \dot{\theta}_j}{\eta_{j2}} + \frac{\tilde{\alpha}_j \dot{\alpha}_j}{\eta_{j3}} + \frac{\tilde{\sigma}_j \dot{\sigma}_j}{\eta_{j4}} + \frac{\tilde{\beta}_j \dot{\beta}_j}{\eta_{j5}} \right)
\]

\[
= \sum_{j=1}^{m} s_j (f_j - \dot{f}_j(s_j) - f_{cp}) - \sum_{j=1}^{m} \left( \frac{\tilde{E}_j \dot{E}_j}{\eta_{j1}} + \frac{\tilde{\theta}_j \dot{\theta}_j}{\eta_{j2}} + \frac{\tilde{\alpha}_j \dot{\alpha}_j}{\eta_{j3}} + \frac{\tilde{\sigma}_j \dot{\sigma}_j}{\eta_{j4}} + \frac{\tilde{\beta}_j \dot{\beta}_j}{\eta_{j5}} \right)
\]

\[
= \sum_{j=1}^{m} s_j (\theta_j^T B_j \alpha_j + \theta_j^T C_j \sigma_j + \theta_j^T \Phi_j + \varepsilon_j + \beta_j^T \Psi_j - f_{cp}) - \sum_{j=1}^{m} \left( \frac{\tilde{E}_j \dot{E}_j}{\eta_{j1}} + \frac{\tilde{\theta}_j \dot{\theta}_j}{\eta_{j2}} + \frac{\tilde{\alpha}_j \dot{\alpha}_j}{\eta_{j3}} + \frac{\tilde{\sigma}_j \dot{\sigma}_j}{\eta_{j4}} + \frac{\tilde{\beta}_j \dot{\beta}_j}{\eta_{j5}} \right)
\]
\[
\begin{align*}
\frac{\theta_j^T}{\eta_j2} \dot{\theta}_j + \frac{\alpha_j^T}{\eta_j3} \dot{\alpha}_j + \frac{\delta_j^T}{\eta_j4} \dot{\delta}_j + \frac{\beta_j^T}{\eta_j5} \dot{\beta}_j \\
= \sum_{j=1}^{m} \left[ \dot{\theta}_j^T \left( s_j \Phi_j - \frac{\dot{\theta}_j}{\eta_j2} \right) + \dot{\alpha}_j^T \left( s_j B_j^T \theta_j - \frac{\dot{\alpha}_j}{\eta_j3} \right) + \dot{\delta}_j^T \left( s_j C_j^T \theta_j - \frac{\dot{\delta}_j}{\eta_j4} \right) + \dot{\beta}_j^T \left( s_j \Psi_j - \frac{\dot{\beta}_j}{\eta_j5} \right) \right] + \sum_{j=1}^{m} \left[ s_j (\varepsilon_j - f_{cpj}^*) - \frac{\dot{E}_j}{\eta_j2} \right]
\end{align*}
\] (4.84)

We substitute the adaptation law (4.42)-(4.47) into (4.84):

\[
\begin{align*}
\dot{V} &= \sum_{j=1}^{m} \left[ s_j (\varepsilon_j - f_{cpj}^*) - \frac{\dot{E}_j}{\eta_j2} \right] \\
&= \sum_{j=1}^{m} \left[ s_j (\varepsilon_j - f_{cpj}^*) - E_j^* |s_j| + E_j |s_j| \right] \\
&\leq \sum_{j=1}^{m} \left[ |s_j| (|\varepsilon_j| - E_j^*) + E_j |s_j| - s_j f_{cpj}^* \right]
\end{align*}
\] (4.85)

The minimum approximation error is given by

\[
\omega_{rj} = E_j \text{sgn}(s_j) - f_{cpj}^*
\] (4.86)

We plug (4.86) into (4.85):

\[
\begin{align*}
\dot{V} &\leq \sum_{j=1}^{m} \left[ |s_j| (|\varepsilon_j| - E_j^*) + E_j |s_j| - s_j f_{cpj}^* \right] \\
&= \sum_{j=1}^{m} \left[ |s_j| (|\varepsilon_j| - E_j^*) + s_j \omega_{rj} \right] \\
&\leq \sum_{j=1}^{m} \left[ |s_j| (|\varepsilon_j| - E_j^*) + |s_j| \omega_{rj} \right] \\
&= \sum_{j=1}^{m} \left[ |s_j| (|\varepsilon_j| - E_j^* + \omega_{rj}) \right]
\end{align*}
\] (4.87)

According to Universal Approximation Theorem in (3.38), the minimum approximation error $\omega_{rj}$ is as small as possible. Since $|\varepsilon_j| \leq E_j^*$ by assumption, we can get

\[
\dot{V} = \sum_{j=1}^{m} \left[ |s_j| (|\omega_{rj}| - (E_j^* - |\varepsilon_j|)) \right] \leq 0
\] (4.88)
where $\dot{V}$ is negative semidefinite. Similar to the stability proof in Section 4.2.4, we can prove $\lim_{t \to \infty} s_j(t) = 0$. The stability of the system is guaranteed and the error asymptotically converges to zero.

### 4.2.5 Stability Proof of Medhaffar et al.’s Algorithm

We define the minimum approximation errors as

$$
\omega_F = F(q, q) - \hat{F}(s|\theta_F^*) = [\omega_{f_1}, \cdots, \omega_{f_n}]^T
$$

$$
\omega_B = B(q) - \hat{B}(s|\theta_B^*) = \omega_{b_{jk}} = b_{jk}(q) - \hat{b}_{jk}(s|\theta_{b_{jk}})
$$

$$
\omega_H = K_M \text{sgn}(s) - \hat{H}(s|\theta_H^*) = [\omega_{h_1}, \cdots, \omega_{h_n}]^T
$$

where $K_M = \text{diag}[K_{M1}, \cdots, K_{Mm}]$ and $K_{M1}, \cdots, K_{Mm}$ are positive constants. From (4.52) we get

$$
\dot{q}_d = \dot{B}(s|\theta_B)\tau + \dot{F}(s|\theta_F^*) + \lambda \dot{e} + \dot{H}(s|\theta_H^*)
$$

We plug (4.92) into (4.50):

$$
\dot{s} = F(q, q) + B(q)\tau - \left(\dot{B}(s|\theta_B)\tau + \dot{F}(s|\theta_F^*) + \lambda \dot{e} + \dot{H}(s|\theta_H^*)\right) + \lambda \dot{e}
$$

$$
= F(q, q) - \dot{F}(s|\theta_F^*) + \left(\dot{B}(s|\theta_B) - \dot{B}(s|\theta_B^*)\right)\tau - \dot{H}(s|\theta_H^*)
$$

$$
= \dot{F}(s|\theta_F^*) - \dot{F}(s|\theta_F) + \omega_F + \left(\dot{B}(s|\theta_B^*) - \dot{B}(s|\theta_B) + \omega_B\right)\tau
$$

$$
+ \dot{H}(s|\theta_H^*) - \dot{H}(s|\theta_H) - \dot{H}(s|\theta_H^*)
$$

For each $s_j$, we can rewrite as

$$
\dot{s}_j = \dot{f}_j(s|\theta_{f_j}^*) - \dot{f}_j(s|\theta_{f_j}) + \omega_{f_j} + \sum_{k=1}^m \left(\dot{b}_{jk}(s|\theta_{b_{jk}}^*) - \dot{b}_{jk}(s|\theta_{b_{jk}}) + \omega_{b_{jk}}\right)\tau_j
$$

$$
+ \dot{h}_j(s_j|\theta_{h_j}) - \dot{h}_j(s_j|\theta_{h_j}^*) - \dot{h}_j(s_j|\theta_{h_j}^*)
$$

$$
= \phi_{f_j}^T \xi(s) + \omega_{f_j} + \sum_{k=1}^m \left(\phi_{b_{jk}}^T \xi(s) + \omega_{b_{jk}}\right)\tau_j + \phi_{h_j}^T \xi(s) - \dot{h}_j(s_j|\theta_{h_j}^*)
$$

(4.94)
where \( \phi_{f_j} = \theta'_{f_j} - \theta_{f_j} \), \( \phi_{b_{jk}} = \theta'_{b_{jk}} - \theta_{b_{jk}} \) and \( \phi_{h_j} = \theta'_{h_j} - \theta_{h_j} \). We define the following Lyapunov function \( V_j \) for each joint of the robotic manipulator:

\[
V_j = \frac{1}{2} \left( \dot{s}_j^2 + \frac{1}{\chi_{f_j}} \phi_{f_j}^T \phi_{f_j} + \sum_{k=1}^{m} \frac{1}{\chi_{b_{jk}}} \phi_{b_{jk}}^T \phi_{b_{jk}} + \frac{1}{\chi_{h_j}} \phi_{h_j}^T \phi_{h_j} \right)
\]

(4.95)

The derivative of \( V_j \) is given by

\[
\dot{V}_j = s_j \ddot{s}_j + \frac{1}{\chi_{f_j}} \phi_{f_j}^T \dot{\phi}_{f_j} + \sum_{k=1}^{m} \frac{1}{\chi_{b_{jk}}} \phi_{b_{jk}}^T \dot{\phi}_{b_{jk}} + \frac{1}{\chi_{h_j}} \phi_{h_j}^T \dot{\phi}_{h_j}
\]

\[
= s_j \left( \phi_{f_j}^T \xi(s) + \sum_{k=1}^{m} \phi_{b_{jk}}^T \xi(s) + \phi_{h_j}^T \xi(s) \right) + \frac{1}{\chi_{f_j}} \phi_{f_j}^T \dot{\phi}_{f_j} + \sum_{k=1}^{m} \frac{1}{\chi_{b_{jk}}} \phi_{b_{jk}}^T \dot{\phi}_{b_{jk}}
\]

\[
+ \frac{1}{\chi_{h_j}} \phi_{h_j}^T \dot{\phi}_{h_j} + s_j (\omega_{f_j} + \sum_{k=1}^{m} \omega_{b_{jk}} \tau_j) - s_j \dot{\theta}_j (s_j \theta'_{h_j})
\]

\[
= \frac{1}{\chi_{f_j}} \phi_{f_j}^T \left( \chi_{f_j} s_j \xi(s) + \dot{\phi}_{f_j} \right) + \sum_{k=1}^{m} \frac{1}{\chi_{b_{jk}}} \phi_{b_{jk}}^T \left( \chi_{b_{jk}} s_j \xi(s) + \dot{\phi}_{b_{jk}} \right)
\]

\[
+ \frac{1}{\chi_{h_j}} \phi_{h_j}^T \left( \chi_{h_j} s_j \xi(s) + \dot{\phi}_{h_j} \right) + s_j \Omega_j - K_{M_j} s_j
\]

(4.96)

where \( \Omega_j = \omega_{f_j} + \sum_{k=1}^{m} \omega_{b_{jk}} \tau_j + \omega_{h_j} \). The adaptation laws are given in (4.57). Then we can rewrite (4.96) as

\[
\dot{V}_j = s_j \Omega_j - K_{M_j} s_j \leq (\|\Omega_j\| - K_{M_j})|s_j|
\]

(4.97)

According to Universal Approximation Theorem in (3.48), \( \Omega_j \) is as small as possible. Therefore, we can simply pick \( K_{M_j} > |\Omega_j| \) and get \( \dot{V}_j < 0(s_j \neq 0) \).

### 4.3 Comparison of Algorithms

Yoo and Ham [3] developed a MIMO fuzzy system to help estimate the torque coupling. In the case of the m-link robotic manipulator, if we define \( k_1 \) membership
functions for each input variable, the number of fuzzy rules applied for each joint is $3k_1^{2m}$ which will result in a high computational load. In addition to the computational load problems, the switching control term in (4.13) also produces chattering.

Guo and Woo downgraded their initial MIMO fuzzy system to a SISO fuzzy system which, in addition to helping estimate the parameter estimates and the residual cross coupling similar to Section 4.1.1, helps to reduce both high computational load and chattering problems encountered in Section 4.1.1. First, if we define $k_2$ membership functions for each input variable, the number of fuzzy rules applied for each joint is $k_2$ which is obviously decreased. Second, by substituting the switching control term in (4.13) for the fuzzy compensator in (4.17) we alleviate chattering.

The algorithms in Sections 4.1.1-4.1.2 can only tune the consequence part of the fuzzy rules, which places the onus of designing the premise part upon the designer. Lin and Hsu surpassed this limitation by presenting a number of adaptation laws that are capable of online tuning both the premise and consequence parts of the fuzzy rules in question. A fuzzy controller and a compensation controller are designed to construct a control law. Unfortunately, this algorithm falls prey to chattering.

Shahnazi et al. proposed a direct AFSMC scheme based on Lin and Hsu's algorithm. In [9], a PI controller is applied inside a boundary layer to attenuate chattering. The parameters and bounds of this PI controller are online adjusted by adaptation laws.

Medhaffar et al. designed an indirect adaptive fuzzy sliding mode controller to control a robotic manipulator. In this approach, MIMO fuzzy systems are applied to estimate the dynamic equations of the robotic manipulator. If we define $k_2$ membership functions for each input variable, the number of fuzzy rules for each joint of the robotic manipulator is $(m + 1) \times k_2^m + k_2$. Compared with Yoo and Ham's algorithm in Section 4.1.1, the number of the fuzzy rules is reduced by introducing
Table 4.1: Comparison of five algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Availability of tuning method</th>
<th>Number of fuzzy rules</th>
<th>Computational load</th>
<th>Chattering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yoo and Ham [3]</td>
<td>consequence part</td>
<td>$3 \times k_1^{2m}$</td>
<td>high</td>
<td>exist</td>
</tr>
<tr>
<td>Guo and Woo</td>
<td>consequence part</td>
<td>$k_2$</td>
<td>low</td>
<td>attenuated</td>
</tr>
<tr>
<td>Lin and Hsu</td>
<td>premise and consequence parts</td>
<td>$k_2$</td>
<td>low</td>
<td>exist</td>
</tr>
<tr>
<td>Shahnazi et al.</td>
<td>premise and consequence parts</td>
<td>$k_2$</td>
<td>low</td>
<td>attenuated</td>
</tr>
<tr>
<td>Medhaffar et al.</td>
<td>consequence part</td>
<td>$m \times k_2^{m+1}$</td>
<td>high</td>
<td>attenuated</td>
</tr>
</tbody>
</table>

sliding surfaces as the inputs of the fuzzy systems. However, the number of the fuzzy rules will become large quickly when we increase the links of the m-link robotic manipulator. Similar to Sections 4.1.1-4.1.2, there is no tuning method for the premise part of the fuzzy rules. The chattering is attenuated by substituting a fuzzy system for the switch control term.

Table 4.1 is a synopsis of the aforementioned five algorithms along four axes: availability of tuning methods, number of fuzzy rules for each joint of the robotic manipulator, computational load and chattering.

4.4 Summary

In this chapter, we introduced five AFSMC algorithms with the adaptation laws derived in the Lyapunov sense. The stability of the closed-loop system is proved mathematically based on the Lyapunov method. Yoo and Ham proposed a MIMO fuzzy system in [3] to compensate for the model uncertainties of the system, but produced

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
chattering. Guo and Woo proposed a SISO fuzzy system to alleviate chattering and to estimate the control gain in the control law. Since there is no tuning method to adjust the premise part of fuzzy rules in [3] and [5], Lin and Hsu presented a scheme to online tune both premise and consequence parts of fuzzy rules. To attenuate the chattering phenomenon in Lin and Hsu’s algorithm [6], Shahnazi et al. developed a PI controller inside a boundary layer and the parameters of the PI controller are online tuned by adaptation laws. Medhaffar et al. proposed a decoupled AFSMC algorithm with MIMO fuzzy systems to control a robotic manipulator whose dynamic equations are totally unknown to the designer. In Medhaffar et al.’s approach, the chattering is avoided, but we can only tune the consequence part of the fuzzy rules.
Chapter 5

Simulation Results

The dynamic equations for the 2-link robotic manipulator are given as

\[
\tau = M(q)\ddot{q} + C(q, \dot{q}) = M(q)\ddot{q} + C_1(q, \dot{q})\dot{q} 
\]

\[
M(q) = \begin{bmatrix}
m_1 l^2 + 2m_2 l^2 + 2m_2 l^2 \cos q_2 & m_2 l^2 + m_2 l^2 \cos q_2 \\
 m_2 l^2 + m_2 l^2 \cos q_2 & m_2 l^2 
\end{bmatrix} 
\]

\[
= \begin{bmatrix}
P_1 + 2P_2 + 2P_2 \cos q_2 & P_2 + P_2 \cos q_2 \\
 P_2 + P_2 \cos q_2 & P_2 
\end{bmatrix} 
\] (5.1)

\[
C(q, \dot{q}) = \begin{bmatrix}
-2m_2 l^2 \dot{q}_1 \dot{q}_2 \sin q_2 - m_2 l^2 \dot{q}_2^2 \sin q_2 \\
 m_2 l^2 \dot{q}_1^2 \sin q_2 
\end{bmatrix} 
\]

\[
= \begin{bmatrix}
-2P_2 \dot{q}_1 \dot{q}_2 \sin q_2 - P_2 \dot{q}_2^2 \sin q_2 \\
 P_2 \dot{q}_1^2 \sin q_2 
\end{bmatrix} 
\]

\[
C_1(q, \dot{q}) = \begin{bmatrix}
-P_2 \dot{q}_2 \sin q_2 - P_2 \dot{q}_1 \sin q_2 - P_2 \dot{q}_2 \sin q_2 \\
 P_2 \dot{q}_1 \sin q_2 
\end{bmatrix} 
\]

\[
= \begin{bmatrix}
-P_2 \dot{q}_2 \sin q_2 - P_2 \dot{q}_1 \sin q_2 - P_2 \dot{q}_2 \sin q_2 \\
 P_2 \dot{q}_1 \sin q_2 
\end{bmatrix} 
\] (5.2)

where \( m_1, m_2 \) are mass; links 1 and 2 have the same length of \( l \); \( q_1 \) and \( q_2 \) are joint positions of the links 1 and 2. The values of the parameters are chosen as \( P_1 = m_1 l^2 = 1.0 \) and \( P_2 = m_2 l^2 = 2.0 \). The desired model transfer function for each
joint of the robotic manipulator is specified as [11]

\[ W_m(s) = \frac{4}{s^2 + 4s + 4}. \]  

(5.5)

The desired trajectory is the output of a filtered sequence of unit steps. The initial values of the robotic manipulators' joint positions are set to 0.5 radians. For all five algorithms, we choose the parameter \( \lambda_j \) in the sliding surface as same as the bandwidth of our desired model. The predefined adaptation gains in adaptation laws are chosen as a result of trial and error. The number and the type of membership functions for each input variable are chosen to be consistent and comparable with the algorithms proposed in [3], [5], [6], [9] and [10].

5.1 Yoo and Ham's Algorithm

Yoo and Ham applied a MIMO fuzzy system to estimate the torque coupling effect for the 2-link robotic manipulator. The input variables of the fuzzy systems in (4.10)-(4.12) are defined as \( F^1(q, \dot{q}) : \{q_1, q_2, \dot{q}_1, \dot{q}_2\} \), \( F^2(q, \ddot{q}_r) : \{q_1, q_2, \ddot{q}_r, \ddot{q}_2\} \) and \( F^3(q, \ddot{q}) : \{q_1, q_2, \ddot{q}, \ddot{q}_2\} \). For each input variable, we define 5 fuzzy sets: NB NS ZO PS PB. The parameters of the fuzzy membership functions for \( q_j, \dot{q}_j, \ddot{q}_rj \) and \( \ddot{q}_j \) in Figure 5.1-Figure 5.4 are selected based on the simulation results in Section 2.4. The membership functions for \( q_j (j = 1, 2) \) are defined as

\[
\begin{align*}
\mu_{q_j, NB} &= \frac{1}{1 + \exp((q_j + 0.75)/0.1)^2} \\
\mu_{q_j, NS} &= \exp(-(q_j + 0.5)/0.3)^2) \\
\mu_{q_j, ZO} &= \exp(-(q_j/0.3)^2) \\
\mu_{q_j, PS} &= \exp(-(q_j - 0.5)/0.3)^2) \\
\mu_{q_j, PB} &= \frac{1}{1 + \exp(-(q_j - 0.75)/0.1)^2}
\end{align*}
\]  

(5.6)
CHAPTER 5. SIMULATION RESULTS

The membership functions for \( \dot{q}_j \) \((j = 1, 2)\) are defined as

\[
\begin{align*}
\mu_{\dot{q}_jNB} &= \frac{1}{1 + [\exp((\dot{q}_j + 1.13)/0.2)]^2} \\
\mu_{\dot{q}_jNS} &= \exp(-((\dot{q}_j + 0.75)/0.45)^2) \\
\mu_{\dot{q}_jSO} &= \exp(-((\dot{q}_j/0.45)^2) \\
\mu_{\dot{q}_jPS} &= \exp(-((\dot{q}_j - 0.75)/0.45)^2) \\
\mu_{\dot{q}_jPB} &= \frac{1}{1 + [\exp(-(\dot{q}_j - 1.13)/0.2)]^2}
\end{align*}
\]

(5.7)

The membership functions for \( \ddot{q}_{rj} \) \((j = 1, 2)\) are defined as

\[
\begin{align*}
\mu_{\ddot{q}_{rj}NB} &= \frac{1}{1 + [\exp((\ddot{q}_{rj} + 7)/1.5)]^2} \\
\mu_{\ddot{q}_{rj}NS} &= \exp(-((\ddot{q}_{rj} + 5)/5)^2) \\
\mu_{\ddot{q}_{rj}ZO} &= \exp(-((\ddot{q}_{rj}/5)^2) \\
\mu_{\ddot{q}_{rj}PS} &= \exp(-((\ddot{q}_{rj} - 5)/5)^2) \\
\mu_{\ddot{q}_{rj}PB} &= \frac{1}{1 + [\exp((\ddot{q}_{rj} - 7)/1.5)]^2}
\end{align*}
\]

(5.8)

The membership functions for \( \ddot{q}_j \) \((j = 1, 2)\) are defined as

\[
\begin{align*}
\mu_{\ddot{q}_jNB} &= \frac{1}{1 + [\exp((\ddot{q}_j + 60)/20)]^2} \\
\mu_{\ddot{q}_jNS} &= \exp(-((\ddot{q}_j + 40)/30)^2) \\
\mu_{\ddot{q}_jZO} &= \exp(-((\ddot{q}_j/30)^2) \\
\mu_{\ddot{q}_jPS} &= \exp(-((\ddot{q}_j - 40)/30)^2) \\
\mu_{\ddot{q}_jPB} &= \frac{1}{1 + [\exp((\ddot{q}_j - 60)/20)]^2}
\end{align*}
\]

(5.9)

We choose \( K_D = \text{diag}[10, 10] \) and \( W = \text{diag}[150, 100] \) in (4.13). The estimated mass and Coriolis matrices are given by

\[
\hat{M} = \begin{bmatrix}
\hat{P}_1 + 2\hat{P}_2 + 2\hat{P}_2\cos q_2 & \hat{P}_2 + \hat{P}_2\cos q_2 \\
\hat{P}_2 + \hat{P}_2\cos q_2 & \hat{P}_2
\end{bmatrix}
\]

(5.10)
CHAPTER 5. SIMULATION RESULTS

\[
\dot{C}_1 = \begin{bmatrix}
-\dot{P}_2 q_2 \sin q_2 & -\dot{P}_2 \dot{q}_1 \sin q_2 - \dot{P}_2 \dot{q}_2 \sin q_2 \\
\dot{P}_2 q_1 \sin q_2 & 0
\end{bmatrix}
\]  \hspace{1cm} (5.11)

where \( \dot{P}_1 = 2, \dot{P}_2 = 4 \). The values of the diagonal elements of \( \Gamma_{11}, \Gamma_{21} \) and \( \Gamma_{31} \) in (4.14) are all set to 300 and the values of the diagonal elements of \( \Gamma_{12}, \Gamma_{22} \) and \( \Gamma_{32} \) in (4.14) are all set to 100 by trial and error.

While Figure 5.5 and Figure 5.7 demonstrate that the tracking accuracy is satisfied within the range \([-0.025, 0.025]\) radians, Figure 5.6 and Figure 5.8 show us that chattering remains a problem.

![Figure 5.1: Membership functions of \( q_j \)](image1)

![Figure 5.2: Membership functions of \( \dot{q}_j \)](image2)

![Figure 5.3: Membership functions of \( \ddot{q}_{q_j} \)](image3)

![Figure 5.4: Membership functions of \( \ddot{q}_{j} \)](image4)
CHAPTER 5. SIMULATION RESULTS

Figure 5.5: (a) Tracking and (b) errors of joint 1 in Yoo and Ham's algorithm. Dash line: desired trajectory; solid line: actual trajectory.

Figure 5.6: (a) Control input and (b) sliding surface of joint 1 in Yoo and Ham's algorithm.
CHAPTER 5. SIMULATION RESULTS

Figure 5.7: (a) Tracking and (b) errors of joint 2 in Yoo and Ham’s algorithm. Dash line: desired trajectory; solid line: actual trajectory.

Figure 5.8: (a) Control input and (b) sliding surface of joint 2 in Yoo and Ham’s algorithm.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 5. SIMULATION RESULTS

5.2 Guo and Woo’s Algorithm

Instead of using a MIMO fuzzy system to estimate the torque coupling effect in Section 4.1.2, Guo and Woo reduced the MIMO fuzzy system to a SISO fuzzy system which can approximate the same residual coupling effect and alleviate the chattering. We choose the same $\hat{M}$ and $\hat{C}_1$ as in Section 5.1. We pick $A = \text{diag}[250, 100]$ in (4.17) and $\gamma_1 = 10^4, \gamma_2 = 5 \times 10^3$ in (4.23) by trial and error. According to Table 4.1, the number of fuzzy rules is reduced from $3 \times k_1^m$ in Yoo and Ham’s algorithm to $k_2$ in Guo and Woo’s algorithm. Therefore we can choose the value of $k_2$ larger than the value of $k_1$. Since we pick $k_1 = 5$ in Section 5.1, we can simply define $k_2 = 7$ fuzzy sets for each input variable: NB NM NS ZO PS PM PB. The parameters of the fuzzy membership functions for $s_j$ are selected based on the simulation results of the sliding surface $s_j$ in Section 2.4. The membership functions for $s_j$ ($j = 1, 2$) in Figure 5.9 are expressed as

\[
\begin{align*}
\mu_{s_j \text{NB}} &= \frac{1}{1 + \exp((s_j + 1.25)/0.1)^2} \\
\mu_{s_j \text{NM}} &= \exp(-(s_j + 1)/0.3)^2 \\
\mu_{s_j \text{NS}} &= \exp(-(s_j + 0.5)/0.3)^2 \\
\mu_{s_j \text{ZO}} &= \exp(-(s_j/0.3)^2) \\
\mu_{s_j \text{PS}} &= \exp(-(s_j - 0.5)/0.3)^2 \\
\mu_{s_j \text{PM}} &= \exp(-(s_j - 1)/0.3)^2 \\
\mu_{s_j \text{PB}} &= \frac{1}{1 + \exp(-(s_j - 1.25)/0.1)^2}.
\end{align*}
\] (5.12)

Figure 5.10 and Figure 5.12 show us that the tracking errors are within the range $[-0.01, 0.01]$ radians, which is comparable to that of Yoo and Ham’s algorithm. However, unlike Yoo and Ham’s algorithm, we see in Figure 5.11 and Figure 5.13 that chattering has been attenuated.
CHAPTER 5. SIMULATION RESULTS

Figure 5.9: Membership functions of $s_j$

Figure 5.10: (a) Tracking and (b) errors of joint 1 in Guo and Woo’s algorithm. Dash line: desired trajectory; solid line: actual trajectory.
CHAPTER 5. SIMULATION RESULTS

Figure 5.11: (a) Control input and (b) sliding surface of joint 1 in Guo and Woo's algorithm.

Figure 5.12: (a) Tracking and (b) errors of joint 2 in Guo and Woo's algorithm. Dash line: desired trajectory; solid line: actual trajectory.
5.3 Revised Lin and Hsu’s Algorithm

In Section 5.1 and Section 5.2, we need to define the premise part of fuzzy rules for each input variable of the fuzzy system. The revised Lin and Hsu’s tuning methodology can online adjust both the premise and the consequence parts of the fuzzy rules. We choose the same $\hat{M}$ and $\hat{C}_1$ as in Section 5.1. We define five membership functions for each input variable in (4.27) ($l = 1, \cdots, 5$). The parameters of the adaptation laws in (4.36)-(4.40) are selected as $\eta_{j_2} = \eta_{j_3} = \eta_{j_4} = 100$, $\eta_{j_1} = 200 (j = 1, 2)$ and $\sigma_1(0) = \sigma_2(0) = [1, 1, 1, 1]^T$ by trial and error.

It can be seen from Figure 5.14 and Figure 5.16 that the tracking errors are with the range $[-0.022, 0.012]$ radians after 4 seconds. The tuning methodology utilized in the premise and consequence parts of the fuzzy rules allows the fuzzy system to approximate the control input. The drawback is the chattering phenomenon appearing in Figure 5.15 and Figure 5.17.
Figure 5.14: (a) Tracking and (b) errors of joint 1 in revised Lin and Hsu's algorithm. Dash line: desired trajectory; solid line: actual trajectory.

Figure 5.15: (a) Control input and (b) sliding surface of joint 1 in revised Lin and Hsu's algorithm.
Figure 5.16: (a) Tracking and (b) errors of joint 2 in revised Lin and Hsu's algorithm. Dash line: desired trajectory; solid line: actual trajectory.

Figure 5.17: (a) Control input and (b) sliding surface of joint 2 in revised Lin and Hsu's algorithm.
5.4 Revised Shahnazi et al.’s Algorithm

To attenuate the chattering in Section 5.3, a PI controller is applied in the revised Shahnazi et al.’s algorithm in Section 4.1.4. We choose the same $\eta_{j1}-\eta_{j4}$ and $\sigma_j(0)$ as in Section 5.3. We define $\eta_{j5} = 500$, $K_{P1}(0) = K_{P2}(0) = 20$, $K_{I1}(0) = 300$, $K_{I2}(0) = 100$ and $\varphi_j = 1(j = 1, 2)$ in (4.36)-(4.40) by trial and error.

Figure 5.18 and Figure 5.20 show that the tracking errors are within the range [-0.04,0.015] radians. The chattering is eliminated by using the PI controller inside the boundary layer as illustrated by Figure 5.19 and Figure 5.21.

![Figure 5.18: (a) Tracking and (b) errors of joint 1 in revised Shahnazi et al.’s algorithm. Dash line: desired trajectory; solid line: actual trajectory.](image-url)
CHAPTER 5. SIMULATION RESULTS

Figure 5.19: (a) Control input and (b) sliding surface of joint 1 in revised Shahnazi et al.'s algorithm.

Figure 5.20: (a) Tracking and (b) errors of joint 2 in revised Shahnazi et al.'s algorithm. Dash line: desired trajectory; solid line: actual trajectory.
CHAPTER 5. SIMULATION RESULTS

Figure 5.21: (a) Control input and (b) sliding surface of joint 2 in revised Shahnazi et al.'s algorithm.

5.5 Medhaffar et al.'s Algorithm

Medhaffar et al. introduced an indirect AFSMC approach to online identify the dynamic equations of the 2-link robotic manipulator in (5.1). This approach is suitable for systems whose dynamic equations are uncertain to the designer. Similar to Guo and Woo's algorithm, the chattering is avoided by adding fuzzy compensators in the control law. Compared with Yoo and Ham's algorithm, the fuzzy rules are reduced by introducing the sliding surfaces as the input variables of the fuzzy systems.

We define the same fuzzy sets for each input variable as in (5.12). Since there is an inverse matrix $\hat{B}^{-1}(s|\theta_B)$ in the control law in (4.52), we have to define an initial condition for the fuzzy systems $\hat{B}(s|\theta_B)$. We choose $\hat{B}(s(0)|\theta_B(0)) = \text{diag}[1,2]$ by trial and error. The parameters of the adaptation laws in (4.57) are selected as $\chi_{f_j} = 5 \times 10^3(j = 1,2)$, $\chi_{b_{jk}} = 5 \times 10^3(j, k = 1,2)$ and $\chi_{b_j} = 0.01(j = 1,2)$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.22: (a) Tracking and (b) errors of joint 1 in Medhaffar et al's algorithm. Dash line: desired trajectory; solid line: actual trajectory.

Figure 5.23: (a) Control input and (b) sliding surface of joint 1 in Medhaffar et al's algorithm.
CHAPTER 5. SIMULATION RESULTS

Figure 5.24: (a) Tracking and (b) errors of joint 2 in Medhaffar et al's algorithm. Dash line: desired trajectory; solid line: actual trajectory.

Figure 5.25: (a) Control input and (b) sliding surface of joint 2 in Medhaffar et al's algorithm.
CHAPTER 5. SIMULATION RESULTS

Figure 5.22 and Figure 5.24 show that the tracking errors are within the range $[-0.01, 0.01]$ radians. From Figure 5.23 and Figure 5.25, we can see that the chattering is alleviated by using the fuzzy systems as the compensators in the control law.

5.6 Stability Analysis of Algorithms in a Practical Sense

The stability of the closed loop systems with the five AFSMC algorithms are guaranteed in Section 4.2.1-Section 4.2.5. Since these proofs are only considered in the mathematical sense, we need to discuss the convergence of the overall system in a practical sense.

To meet the requirement of Lyapunov stability in Section 4.2.1, we need to choose a constant $K_{D_j}$ to make $|s_j K_{D_j}| > |\omega_j|(s_j \neq 0)$ where $\omega_j$ is defined as the minimum approximation error in (4.65). According to the universal approximation theorem in (3.38), we can find a fuzzy system to estimate any given real continuous function and the approximation error $\omega_j$ is as small as possible. However, in practice, we can only define a fuzzy system with finite membership functions and finite fuzzy rules. Then $\omega_j$ will no longer be as small as possible in the real application. When $s_j$ is close to zero and $|s_j K_{D_j}|$ becomes very small if not equal to zero, the condition $|s_j K_{D_j}| > |\omega_j|(s_j \neq 0)$ will no longer hold. The same situation happened in Guo and Woo's algorithm. In Section 4.2.2, we need to choose a positive constant $a_j$ to make $a_j |s_j| > |\omega_j|(s_j \neq 0)$ in (4.73). Since $\omega_j$ can not be arbitrarily small, we will always get $a_j |s_j| < |\omega_j|(s_j \neq 0)$ when $s_j$ gets close to the sliding surface and $a_j |s_j|$ is very small if not equal to zero.

To verify Lyapunov stability in the revised Lin and Hsu's algorithm, we need to find the optimal values $\theta_j^*, \alpha_j^*$ and $\sigma_j^*$ in (4.29). Since the optimal values are unknown
to the designer, one may obtain these values by running the simulation first and checking the values from the simulation results. From the adaptation laws in (4.36)-(4.38), we find that $\dot{\theta}_j$, $\dot{\alpha}_j$ and $\dot{\sigma}_j$ do not converge to zero due to the chattering problem in the sliding surface $s_j$. Therefore, we can not get the optimal values through the simulation results.

Section 4.2.4 and Section 4.2.5 use a fuzzy system to estimate the discontinuous function $\text{sgn}(s_j)$ in (4.86) and (4.91). Since the universal approximation theorem can only deal with the real continuous function, we may not find an optimal fuzzy system and obtain the minimum approximation error described in (4.86) and (4.91). Thus, both Medhaffar et al.'s algorithm and revised Shahnazi et al.'s algorithm have difficulty proving the convergence of the overall system in a practical sense.

5.7 Summary

In this chapter, we applied the five algorithms to a two degree of freedom robotic manipulator and simulated for the period of 20 seconds. Simulation results show that the tracking performance with small tracking errors is satisfied. The chattering phenomenon appeared in Section 5.1 and Section 5.3 are attenuated in Section 5.2, Section 5.4 and Section 5.5. In addition to the adaptation gains, we need to predefined the initial conditions of the fuzzy rules by a trial and error method. The analysis of Lyapunov stability in a practical sense shows that all five algorithms suffer from a lack of a convergence proof of the closed loop system in the real application.
Chapter 6

Conclusions and Future Work

Five adaptive fuzzy sliding mode control algorithms for robotic manipulators are investigated in this thesis. Yoo and Ham’s algorithm [3] utilizes MIMO fuzzy systems to estimate the cross-coupling effects in robotic manipulators and gets perfect tracking accuracy. However, the switching control term in the control law causes chattering and there is no methodology to tune the premise part of the fuzzy rules. Guo and Woo’s algorithm attenuated the chattering problem very well by substituting a fuzzy compensator for the switching control term. The number of fuzzy rules is also reduced by abandoning MIMO fuzzy systems and using SISO fuzzy systems instead. But we still need to predefine the premise part of the fuzzy rules. The revised Lin and Hsu’s algorithm created a methodology of learning both the premise and the consequence part of the fuzzy rules. But chattering is inevitable. Based on the knowledge of the revised Lin and Hsu’s learning method, the revised Shahnazi et al.’s algorithm uses the PI controller inside the bound layer to attenuate the chattering and gets good results. Medhaffar et al.’s algorithm brings us an indirect adaptive fuzzy sliding mode controller which can estimate the dynamic equations of the robotic manipulators and chattering is alleviated by using the compensator proposed by Guo and Woo.

The stability and the convergence of the five algorithms for the m-link robotic
manipulator is proved theoretically using Lyapunov stability theory. However, in practice, all five algorithms applied to robotic manipulators fail to show the asymptotic convergence of the overall systems and Lyapunov stability criteria are no longer satisfied in a practical sense.

All five algorithms have predefined adaptation gains in the adaptation laws which are highly related to the performance of our controllers. In future research, a tuning methodology to online learn these adaptation gains is recommended. Since Lyapunov stability proof in practice does not hold, the study of convergence for robotic manipulators in practice is considered to be an issue for future work. Furthermore, since all algorithms are only evaluated and compared via computer simulation, experimental studies of these algorithms are needed in the future.
References


87
REFERENCES


Appendix A: Lists of Simulink Diagrams

Figure A-1: Simulink model to simulate a 2 DOF robotic manipulator using classical sliding mode control method.
Figure A-2: Simulink subsystem that implements the desired trajectory of the joint for a 2 DOF robotic manipulator.

Figure A-3: Simulink subsystem that implements the dynamics of a 2 DOF robotic manipulator.
function [u1, u2, s1, s2, e1, e2, deltaf1, deltaf2, V, Vdot] = fcn(q, dq, d2q, q1, q2, qldot, q2dot)

Pl=1;
P2=2;
Ple=3;
P2e=7;
lambda=2;
k1=45;
k2=21;

%define the sliding surface
s1=qldot-dq+lambda*(q1-q);
s2=q2dot-dq+lambda*(q2-q);

%the dynamic equations of the robotic manipulator
H_e=[Ple+2*P2e+2*P2e*cos(q2) P2e+P2e*cos(q2);
P2e+P2e*cos(q2) P2e];
h_e=[-P2e*q2dot*sin(q2) -P2e*qldot*sin(q2)-P2e*q2dot*sin(q2);
P2e*qldot*sin(q2) 0];

%the estimated dynamic equations of the robotic manipulator
H=[P1+2*P2+2*P2*cos(q2) P2+P2*cos(q2);
P2+P2*cos(q2) P2];
h=[-P2*q2dot*sin(q2) -P2*qldot*sin(q2)-P2*q2dot*sin(q2);
P2*qldot*sin(q2) 0];

%define the control law
ue=H_e*( [d2q; d2q] - 2*([qldot-dq; q2dot-dq]) ) + h_e*( [dq; dq] - 2*([q1-q; q2-q]) ) - [1 0; 0 1]*[s1; s2];
uue=[k1 0; 0 k2]*[sign(s1); sign(s2)];

%calculate the Lyapunov function candidate V & Vdot
deltaH=H_e-H;
deltah=h_e-h;
deltaf=[deltaH*( [d2q; d2q] - 2*([qldot-dq; q2dot-dq])) + deltah*( [dq; dq] - 2*([q1-q; q2-q]))];
deltaf1=deltaf(1,1);
deltaf2=deltaf(2,1);
Vdot=s1*(deltaf1-k1*sign(s1))+s2*(deltaf2-k2*sign(s2))-[s1 s2]*[1 0; 0 1]*[s1; s2];

%define the tracking error
el=q1-q;
e2=q2-q;
ul=u(1,1);
ul2=u(2,1);

Figure A-4: Embedded MATLAB function code in Figure A-1.
Appendix A: Lists of Simulink Diagrams

Figure A-5: Simulink model to simulate a 2 DOF robotic manipulator using classical sliding mode control method with the saturation function.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
function [s1, s2, u1, u2, e1, e2, s1_phi1, s2_phi2] = fcn(sats1, sats2, q, dq, d2q, q1, q2, q1dot, q2dot)

P1e=2;
P2e=4;
lambda=2;
k1=45;
k2=21;

phi1=0.1;
phi2=0.1;

% define the sliding surface
s1=q1dot-dq+lambda*(q1-q);
s2=q2dot-dq+lambda*(q2-q);

s1_phi1=s1/phi1;
s2_phi2=s2/phi2;

% define the estimated dynamic equations of the 2-link robotic manipulator
H_e=[P1e+2*P2e+2*P2e*cos(q2) P2e+P2e*cos(q2); P2e+P2e*cos(q2) P2e];
h_e=[-P2e*q2dot*sin(q2) -P2e*q1dot*sin(q2)-P2e*q2dot*sin(q2); P2e*q1dot*sin(q2) 0];

ue=H_e*([d2q;d2q]-2*([dq;d2q]-2*([q1-q;q2-q])); h_e*([dq;d2q]-2*([q1-q;q2-q]))-[1 0; 0 1]*[s1; s2];

% define the control law
u=[k1 0; 0 k2]*[sats1; sats2];

e1=q1-q;
e2=q2-q;

u1=u(1,1);
u2=u(2,1);

Figure A-6: Embedded MATLAB function code in Figure A-5.
Figure A-7: Simulink model to simulate a 2 DOF robotic manipulator using Yoo and Ham's algorithm.
Appendix A: Lists of Simulink Diagrams

function[s1,s2,e1,e2,u1,u2,theta dot11,theta dot12,theta dot13,theta dot21,theta dot22,theta dot23]=fcn(qdl,qd2,dq,d2q,q1,q2,qldot,q2dot,q1ddot,q2ddot,theta_11,theta_12,theta_13,theta_21,theta_22,theta_23)

P1e=2;P2e=4;
lambdal=2;lambda2=2;
%define the sliding surface
s1=qldot-dq+lambdal*(q1-qdl);
s2=q2dot-dq+lambda2*(q2-qd2);
%define the reference state
qrdot1=qrdot(1,1);
qrdot2=qrdot(2,1);
%define the estimated dynamic equations Mq_e & C_e of the 2-link robotic manipulator
Mq_e=[Ple+2*P2e+2*P2e*cos(q2) P2e+P2e*cos(q2);P2e+P2e*cos(q2) P2e+P2e*cos(q2)];
C_e=[-2*P2e*qldot*sin(q2) -P2e*qldot*sin(q2);P2e*q2dot*sin(q2) 0];
%define membership functions of 5 fuzzy if-then rules
NBq=-0.75; deltaNBq=0.1;
NSq=-0.5; deltaNSq=0.3;
ZOq=0; deltaZOq=0.5;
PSq=0.5; deltaPSq=0.3;
PBq=0.75; deltaPBq=0.1;

NBqdot=-1.13; deltaNBqdot=0.2;
NSqdot=-0.75; deltaNSqdot=0.45;
ZOqdot=0; deltaZOqdot=0.5;
PSqdot=0.5; deltaPSqdot=0.45;
P Bjdot=1.13; deltaPBjdot=0.2;

NBqddot=-60; deltaNBqddot=20;
NSqddot=-40; deltaNSqddot=30;
ZOqddot=0; deltaZOqddot=30;
PSqddot=40; deltaPSqddot=30;
P Bqddot=60; deltaPBqddot=20;

NBqrrddot=-7; deltaNBqrrddot=1.5;
NSqrrddot=-5; deltaNSqrrddot=5;
ZOqrrddot=0; deltaZOqrrddot=5;
PSqrrddot=5; deltaPSqrrddot=5;
P Bqrrddot=7; deltaPBqrrddot=1.5;

A1=ones(5,1); A2=ones(5,1); A3=ones(5,1); A4=ones(5,1); B1=ones(5,1); B2=ones(5,1);

A1(1)=1./((1+exp((-q1-NBq)/deltaNBq)).^2);
A1(2)=exp((-q1-NBq)/deltaNBq).*2);
A1(3)=exp(-((q1-NSq)/deltaNSq).*2);
A1(4)=exp(-((q1-ZOq)/deltaZOq).*2);
A1(5)=1./((1+exp((-q1-PBq)/deltaPBq)).^2);

A2(1)=1./((1+exp((q1-NBq)/deltaNBqdot)).^2);
A2(2)=exp((-q1-NBq)/deltaNBqdot).*2);
A2(3)=exp(-((q1-NSq)/deltaNSqdot).*2);
A2(4)=exp(-((q1-ZOq)/deltaZOqdot).*2);
A2(5)=1./((1+exp((-q1-PBq)/deltaPBqdot)).^2);

Figure A-8: Embedded MATLAB function code in Figure A-7(page1).
Appendix A: Lists of Simulink Diagrams

A3(1)=1./(1+exp((q1ddot-NBqddot)/deltaNBqddot).^2);
A3(2)=exp(-((q1ddot-2Oqddot)/delta2Oqddot).^2);
A3(4)=exp(-((q1ddot-PSqddot)/deltaPSqddot).^2);
A3(5)=1./(1+exp(-((q1ddot-PBqddot)/deltaPBqddot)).^2);

A4(1)=1./(1+exp((qrddotl-NBqrddot)/deltaNBqrddot).^2);
A4(2)=exp(-((qrddotl-NSqrddot)/deltaNSqrddot).^2);
A4(3)=exp(-((qrddotl-ZOqrddot)/deltaZOqrddot).^2);
A4(4)=exp(-((qrddotl-PSqrddot)/deltaPSqrddot).^2);
A4(5)=1./(1+exp(-((qrddotl-PBqrddot)/deltaPBqrddot)).^2);

B1(1)=1./(1+exp((q2-NBq)/deltaNBq).^2);
B1(2)=exp(-((q2-NSq)/deltaNSq).^2);
B1(3)=exp(-((q2-ZOq)/deltaZOq).^2);
B1(4)=exp(-((q2-PSq)/deltaPSq).^2);
B1(5)=1./(1+exp(-((q2-PBq)/deltaPBq)).^2);

B2(1)=1./(1+exp((q2dot-NBqdot)/deltaNBqdot).^2);
B2(2)=exp(-((q2dot-NSqdot)/deltaNSqdot).^2);
B2(3)=exp(-((q2dot-2Oqdot)/delta2Oqdot).^2);
B2(4)=exp(-((q2dot-PSqdot)/deltaPSqdot).^2);
B2(5)=1./(1+exp(-((q2dot-PBqdot)/deltaPBqdot)).^2);

B3(1)=1./(1+exp((q2ddot-NBqddot)/deltaNBqddot).^2);
B3(2)=exp(-((q2ddot-NSqddot)/deltaNSqddot).^2);
B3(3)=exp(-((q2ddot-2Oqddot)/delta2Oqddot).^2);
B3(4)=exp(-((q2ddot-PSqddot)/deltaPSqddot).^2);
B3(5)=1./(1+exp(-((q2ddot-PBqddot)/deltaPBqddot)).^2);

B4(1)=1./(1+exp((qrddot2-NBqrddot)/deltaNBqrddot).^2);
B4(2)=exp(-((qrddot2-NSqrddot)/deltaNSqrddot).^2);
B4(3)=exp(-((qrddot2-2Oqrddot)/delta2Oqrddot).^2);
B4(4)=exp(-((qrddot2-PSqrddot)/deltaPSqrddot).^2);
B4(5)=1./(1+exp(-((qrddot2-PBqrddot)/deltaPBqrddot)).^2);

El=ones(625,1);k=1;
for l=1:5
    for m=1:5
        for i=1:5
            for j=1:5
                E1(k)=A1(l)*A2(m)*B1(i)*B2(j);
            end
        end
    end
end
SUMEl=0;
for n=1:625
    SUMEl=SUMEl+E1(n);
end
psil1=E1./SUMEl;

E2=ones(625,1);k=1;
for l=1:5
    for m=1:5
        for i=1:5

Figure A-9: Embedded MATLAB function code in Figure A-7(page2).
for j=1:5
    El(k)=A1(1) * A4(m) * B1(i) * B4(j);
k=k+1;
end
end
end
SUME2=0;
for n=1:625
    SUME2=SUME2+E2(n);
end
psi2=E2./SUME2;
E3=ones(625,1); k=1;
for l=1:5
    for m=1:5
        for i=1:5
            for j=1:5
                El(k)=A1(1) * A3(m) * B1(i) * B3(j);
k=k+1;
            end
        end
    end
end
SUME3=0;
for n=1:625
    SUME3=SUME3+E3(n);
end
psi3=E3./SUME3;

% define the adaptation gains
theta_dot11=3000*s1*psi1;
theta_dot12=-3000*s1*psi2;
theta_dot13=-3000*s1*psi3;
theta_dot21=-1000*s2*psi1;
theta_dot22=-1000*s2*psi2;
theta_dot23=-1000*s2*psi3;

% define the tracking errors
el=q1-qd1;
e2=q2-qd2;

F11=theta_11'*psi1;
F12=theta_12'*psi2;
F13=theta_13'*psi3;
F21=theta_21'*psi1;
F22=theta_22'*psi2;
F23=theta_23'*psi3;

ue=Mq_e*qrddot+C_e*qrdot+
    [F11(1,1);F21(1,1)]+
    [F12(1,1);F22(1,1)]+
    [F13(1,1);F23(1,1)]
    -[10 0 10]*[s1;s2]-[150 0 100]*[sign(s1);sign(s2)];

% define the control laws
u1=ue(1,1);
u2=ue(2,1);

Figure A-10: Embedded MATLAB function code in Figure A-7(page3)
Figure A-11: Simulink model to simulate a 2 DOF robotic manipulator using Guo and Woo's algorithm.
function [ul, u2, theta_dotkl, theta_dotk2, el, e2, si, s2] = fen(qdl, qd2, dq, d2q, ql, q2, qldot, q2dot, theta_kl, theta_k2)

PLE = 2; P2E = 4; 
lambda = 2;

%define the sliding surface si & s2
sl = qldot - dq + lambda * (ql - qdl);
s2 = q2dot - dq + lambda * (q2 - qd2);

%define estimated dynamic equations M_e & C_e of the 2-link robotic manipulator
M_e = [PLE + 2 * P2E + 2 * P2E * cos(q2); P2E + P2E * cos(q2);
P2E + P2E * cos(q2)] ;

%define membership functions of fuzzy if-then rules
NBs = -1.25; deltaNBs = 0.1;
NMs = -1; deltaNMs = 0.3;
NSs = -0.5; deltaNSs = 0.3;
ZEs = 0; deltaZEs = 0.3;
PSSs = 0.5; deltaPSSs = 0.3;
PMs = 1; deltaPMs = 0.3;
PBS = 1.25; deltaPBS = 0.1;

muAsl = ones(7, 1);
mua1(1, 1) = 1 ./ (1 + exp(((sl - NBs) / deltaNBs).^2));
mua1(2, 1) = exp(-((sl - NMs) / deltaNMs).^2);
mua1(3, 1) = exp(-((sl - NSs) / deltaNSs).^2);
mua1(4, 1) = exp(-((sl - ZEs) / deltaZEs).^2);
mua1(5, 1) = exp(-((sl - PSSs) / deltaPSSs).^2);
mua1(6, 1) = exp(-((sl - PMs) / deltaPMs).^2);
mua1(7, 1) = 1 ./ (1 + exp(-((sl - PBSs) / deltaPBSs).^2));

muAs2 = ones(7, 1);
mua2(1, 1) = 1 ./ (1 + exp(((s2 - NBs) / deltaNBs).^2));
mua2(2, 1) = exp(-((s2 - NMs) / deltaNMs).^2);
mua2(3, 1) = exp(-((s2 - NSs) / deltaNSs).^2);
mua2(4, 1) = exp(-((s2 - ZEs) / deltaZEs).^2);
mua2(5, 1) = exp(-((s2 - PSSs) / deltaPSSs).^2);
mua2(6, 1) = exp(-((s2 - PMs) / deltaPMs).^2);
mua2(7, 1) = 1 ./ (1 + exp(-((s2 - PBSs) / deltaPBSs).^2));

sum_mua1 = mua1(1, 1) + mua1(2, 1) + mua1(3, 1) + mua1(4, 1) + mua1(5, 1) + mua1(6, 1) + mua1(7, 1);
psi_1 = mua1 ./ sum_mua1;

sum_mua2 = mua2(1, 1) + mua2(2, 1) + mua2(3, 1) + mua2(4, 1) + mua2(5, 1) + mua2(6, 1) + mua2(7, 1);
psi_2 = mua2 ./ sum_mua2;

ue = M_e .* ((d2q * dq2) - lambda * ((q2dot - dq; q2dot - dq))) + C_e .* ((dq; dq - lambda * (q1 - ql; q2 - q2q2))) - [250 0 0 100] * [sl; s2];
ue1 = ue; 
ue2 = ue(2, 1);

%define adaptation laws
theta_dotkl = 10000 * sl * psi_1;
theta_dotk2 = 5000 * s2 * psi_2;

%define the tracking errors
el = ql - qdl;
e2 = q2 - qd2;

kl = theta_k1 * psi_1;
k2 = theta_k2 * psi_2;

%define control laws
ul = ue1 - kl(1, 1);
u2 = ue2 - k2(1, 1);
Figure A-13: Simulink model to simulate a 2 DOF robotic manipulator using revised Lin and Hsu's algorithm or revised Shahnazi et al.'s algorithm.
Figure A-14: Simulink Subsystem1 for revised Lin and Hsu's algorithm in Figure A-13 (same as in Subsystem2)
function[alpha1ldot, alpha2ldot, alpha3ldot, alpha4ldot, alpha5ldot, clldot, cl2dot, cl3dot, cl4dot, cl5dot, wlldot, wl2dot, wl3dot, wl4dot, wl5dot, Eldot, ul, si, el]

fcn(q1, q2, qldot, q2dot, ql, dq, d2q, alpha1, alpha2, alpha3, alpha4, alpha5, cl1, cl2, cl3, cl4, cl5, wll, wl2, wl3, wl4, wl5, El)
P1=2; P2e=6; lambda=2;
% define the sliding surface si
si=qldot-dq+lambda*(ql-qdl);
% define the estimated dynamic equations Mq_e & C_e of the 2-link robotic manipulators
Mq_e=[Ple+2*P2e+2*P2e*cos(q2) P2e+P2e*cos(q2) P2e; P2e+P2e*cos(q2) P2e+P2e*cos(q2) P2e; 0 P2e+P2e*cos(q2) 0];
C_e=[-P2e*q2dot*sin(q2) -P2e*qldot*sin(q2)-P2e*q2dot*sin(q2); P2e*qldot*sin(q2) 0];
% define the adaptation gains
Etal=100; Eta2=100; Eta3=100; Eta4=100;
% define the adaptation laws
alpha1ldot=Eta2*si'*exp(-wll*(si-cll))*alpha1;
alpha2ldot=Eta2*si'*exp(-wl2*(si-cl2))*alpha2;
alpha3ldot=Eta2*si'*exp(-wl3*(si-cl3))*alpha3;
alpha4ldot=Eta2*si'*exp(-wl4*(si-cl4))*alpha4;
alpha5ldot=Eta2*si'*exp(-wl5*(si-cl5))*alpha5;
clldot=Eta3*si*2*wll*(si-cll)*exp(-wll*(si-cll))*alpha1;
cl2dot=Eta3*si*2*wl2*(si-cl2)*exp(-wl2*(si-cl2))*alpha2;
cl3dot=Eta3*si*2*wl3*(si-cl3)*exp(-wl3*(si-cl3))*alpha3;
cl4dot=Eta3*si*2*wl4*(si-cl4)*exp(-wl4*(si-cl4))*alpha4;
cl5dot=Eta3*si*2*wl5*(si-cl5)*exp(-wl5*(si-cl5))*alpha5;
wlldot=Eta4*si*(-2*wll*(si-cll))*alpha1;
w2dot=Eta4*si*(-2*wl2*(si-cl2))*alpha2;
w3dot=Eta4*si*(-2*wl3*(si-cl3))*alpha3;
w4dot=Eta4*si*(-2*wl4*(si-cl4))*alpha4;
w5dot=Eta4*si*(-2*wl5*(si-cl5))*alpha5;
Eldot=Etal*abs(si);

ucpl=El*sign(si);

ur=Mq_e*(\[d2q; d2q\]-lambda*\[ql-dq; q2dot-dq\])+C_e*(\[dq; dq\]-lambda*\[ql-dq; q2dot-dq\]);
% define the control law
ul=ur(1,1); -alpha1*exp(-wll*(si-cll))+alpha2*exp(-wl2*(si-cl2))+alpha3*exp(-wl3*(si-cl3))+alpha4*exp(-wl4*(si-cl4))+alpha5*exp(-wl5*(si-cl5))-ucpl;
% define the tracking error
el=ql-qdl;

Figure A-15: Embedded MATLAB function code in Figure A-14 (same as in Subsystem2)
Figure A-16: Simulink Subsystem1 for revised Shahnazi et al.'s algorithm in Figure A-13 (same as in Subsystem2)
function[Kpldot,Kldot,alpha1dot,alpha12dot,alpha13dot,alpha14dot,alpha15dot,cl1dot,cl2dot, cl3dot,cl4dot,cl5dot,wldot,wld2dot,wld3dot,wld4dot,wld5dot,Eldot,ul,si,el]=fcn(Kp1,K11,s1,q1,q 2,q1dot,q1ddot,q1dddot,q1,q2,q2dot,q2ddot,q2d2q,alphais1,alphais2,alphais3,alphais4,alphais5,cl1,cl2,cl3, cl4,cl5,wl1,wl2,wl3,wl4,wl5,El)

Pl=1+P2; P1e=2+P2e=4;
lambda=2;

%define sliding surface sl
sl=q1ddot-q2+lambda*(q1-q2); sl
M=[P1*2*P2+2*P2*cos(q1) P2+P2*cos(q2); P2+P2*cos(q2) P2]; M =
C=[-P2*q1ddot*sin(q2) -P2*q2ddot*sin(q2); P2*q2ddot*sin(q2) 0]; C =

%estimated dynamic equations M_q and C of robotic manipulator
M_e=[P1*2*P2e+2*P2e*cos(q2) P2e+P2e*cos(q2); P2e+P2e*cos(q2) P2e]; M_e =
C_e=[-P2e*q1ddot*sin(q2) -P2e*q2ddot*sin(q2); P2e*q2ddot*sin(q2) 0]; C_e =

%define adaptation gains
Eta1=200; Eta2=100; Eta3=100; Eta4=100; Eta5=500;

%define adaptation laws
alphais1dot=Eta2*si*exp(-wllA2*(sl-cll)*A2);
alphais2dot=Eta2*si*exp(-wl2A2*(sl-cl2)*A2);
alphais3dot=Eta2*si*exp(-wl3A2*(sl-cl3)*A2);
alphais4dot=Eta2*si*exp(-wl4A2*(sl-cl4)*A2);
alphais5dot=Eta2*si*exp(-wl5A2*(sl-cl5)*A2);

clldot=Eta3*si*2*wllA2*(sl-cll)*exp(-wllA2*(sl-cll)*A2)*alphais1;
cl2dot=Eta3*si*2*wl2A2*(sl-cl2)*exp(-wl2A2*(sl-cl2)*A2)*alphais2;
cl3dot=Eta3*si*2*wl3A2*(sl-cl3)*exp(-wl3A2*(sl-cl3)*A2)*alphais3;
cl4dot=Eta3*si*2*wl4A2*(sl-cl4)*exp(-wl4A2*(sl-cl4)*A2)*alphais4;
cl5dot=Eta3*si*2*wl5A2*(sl-cl5)*exp(-wl5A2*(sl-cl5)*A2)*alphais5;

wlldot=Eta4*si*(-2*wll*(sl-cll)*exp(-wllA2*(sl-cll)*A2))*alphais1;
wld2dot=Eta4*si*(-2*wl2*(sl-cl2)*exp(-wl2A2*(sl-cl2)*A2))*alphais2;
wld3dot=Eta4*si*(-2*wl3*(sl-cl3)*exp(-wl3A2*(sl-cl3)*A2))*alphais3;
wld4dot=Eta4*si*(-2*wl4*(sl-cl4)*exp(-wl4A2*(sl-cl4)*A2))*alphais4;
wld5dot=Eta4*si*(-2*wl5*(sl-cl5)*exp(-wl5A2*(sl-cl5)*A2))*alphais5;

Eldot=Eta1*abs(sl);
Kpldot=Eta5*si; Kldot=Eta5*si;

if(sl>l)
ucpl=El;
elseif(sl<—1)
ucpl=-El;
else
ucpl=Kpl*sl+Kld*sl;
end

ur=M_e:*([q1ddot-q1dddot] -lambda*([q1-q2] -lambda*([q1-q1] -q2-q2d2q])); ur

%define the control law for joint1
ul=ur((1,1)-alphais1*exp(-wllA2*(sl-cll)*A2)+alphais2*exp(-wl2A2*(sl-cl2)*A2)+alphais3*exp(-wl3A2*(sl-cl3)*A2)+alphais4*exp(-wl4A2*(sl-cl4)*A2)+alphais5*exp(-wl5A2*(sl-cl5)*A2))-ucpl;

%define joint1 error
el=q1-q2d2q;

Figure A-17: Embedded MATLAB function code in Figure A-16 (same as in Subsystem2)
Figure A-18: Simulink model to simulate a 2 DOF robotic manipulator using Medhaffar et al.'s algorithm
Appendix A: Lists of Simulink Diagrams

function[u1,u2,theta_dotf1,theta_dotf2,theta_dotb11,theta_dotb12,theta_ dotb21,theta_dotb22,theta_doth1,theta_doth2,s1,s2,e1,e2]=fcn(qd1,qd2,dq ,dq2,q1,q2,qdot,q2dot,q1ddot,q2ddot,theta_f1,theta_f2,theta_h1,theta_ h2)

P1=1;P2=2;
P1e=2;P2e=4;
lambda=2;

%define the sliding surface sl & s2
sl=qldot-dq+lambda*(ql-qdl);
s2=q2dot-dq+lambda*(q2-qd2);

%define real dynamic equations Mq and C of the 2-link robotic

Mq=[P1+2+P2+2*P2*cos(q2) P2+P2*cos(q2);
P2+P2*cos(q2) P2];
C=[-P2*q2dot*sin(q2) -P2*q2dot*sin(q2)-P2*q2dot*sin(q2);
P2*q2dot*sin(q2) 0];

%define membership functions of 7 fuzzy if-then rules
NBs=-1.25;
NMs=-1;
NSs=-0.5;
ZEs=0;
PSs=0.5;
PMs=1;
PBs=1.25;

deltaNBs=0.1;
deltaNMs=0.3;
deltaNSs=0.3;
deltaZEs=0.3;
deltaPSs=0.3;
deltaPMs=0.3;
deltaPBs=0.1;

A1(1)=1./(1+exp((sl-NBs)/deltaNBs).^2);
A1(2)=exp((-((sl-NMs)/deltaNMs).^2));
A1(3)=exp((-((sl-NSs)/deltaNSs).^2));
A1(4)=exp((-((sl-ZEs)/deltaZEs).^2));
A1(5)=exp((-((sl-PSs)/deltaPSs).^2));
A1(6)=exp((-((sl-PMs)/deltaPMs).^2));
A1(7)=1./(1+exp((-((sl-PBs)/deltaPBs)).^2));

A2(1)=1./(1+exp((s2-NBs)/deltaNBs).^2);
A2(2)=exp((-((s2-NMs)/deltaNMs).^2));
A2(3)=exp((-((s2-NSs)/deltaNSs).^2));
A2(4)=exp((-((s2-ZEs)/deltaZEs).^2));
A2(5)=exp((-((s2-PSs)/deltaPSs).^2));
A2(6)=exp((-((s2-PMs)/deltaPMs).^2));
A2(7)=1./(1+exp((-((s2-PBs)/deltaPBs)).^2));

Figure A-19: Embedded MATLAB function code in Figure A-18 (page 1)
Appendix A: Lists of Simulink Diagrams

\[ E_l = \text{ones}(49,1); k = 1; \]
for \( i = 1:7 \)
    for \( m = 1:7 \)
        \[ E_l(k) = A_1(1) * A_2(m); \]
        \[ k = k + 1; \]
    end
end

\[ \text{SUM} E_1 = 0; \]
for \( n = 1:49 \)
    \[ \text{SUM} E_1 = \text{SUM} E_1 + E_l(n); \]
end

\[ \text{psil} = E_l / \text{SUM} E_1; \]
\[ \text{thetaDotf1} = 5000 * \text{psil}; \]
\[ \text{thetaDotf2} = 5000 * \text{psil}; \]
\[ F_1 = \text{thetaDotf1} * \text{psil}; \]
\[ F_2 = \text{thetaDotf2} * \text{psil}; \]
\[ B_{11} = \text{thetaB11} * \text{psil}; \]
\[ B_{12} = \text{thetaB12} * \text{psil}; \]
\[ B_{21} = \text{thetaB21} * \text{psil}; \]
\[ B_{22} = \text{thetaB22} * \text{psil}; \]
\[ F = [F_1(1,1); F_2(1,1)]; \]
\[ B = [B_{11}(1,1); B_{12}(1,1); B_{21}(1,1); B_{22}(1,1)]; \]
\[ \text{summuAs1} = A_1(1) + A_1(2) + A_1(3) + A_1(4) + A_1(5) + A_1(6) + A_1(7); \]
\[ \text{psisl} = A_1 / \text{summuAs1}; \]
\[ \text{summuAs2} = A_2(1) + A_2(2) + A_2(3) + A_2(4) + A_2(5) + A_2(6) + A_2(7); \]
\[ \text{psisl2} = A_2 / \text{summuAs2}; \]
\[ \text{thetaDoth1} = 5000 * \text{psisl} * \text{psisl}; \]
\[ \text{thetaDoth2} = 5000 * \text{psisl} * \text{psisl}; \]
\[ h_1 = \text{thetaH1} * \text{psisl}; \]
\[ h_2 = \text{thetaH2} * \text{psisl2}; \]
\[ H = [h_1(1,1); h_2(1,1)]; \]
\% define the control law
\[ u = \text{inv}(B) * \{(d2q; d2q) - \lambda * (\{q1dot - dq; q2dot - dq\} - F - H); \]
\[ u_1 = u(1,1); \]
\[ u_2 = u(2,1); \]
\[ \text{thetaDotb11} = 0.01 * \text{psil} * u_1; \]
\[ \text{thetaDotb12} = 0.01 * \text{psil} * u_2; \]
\[ \text{thetaDotb21} = 0.01 * \text{s2} * \text{psil} * u_1; \]
\[ \text{thetaDotb22} = 0.01 * \text{s2} * \text{psil} * u_2; \]
\% define tracking errors
\[ e_1 = q_1 - qd_1; \]
\[ e_2 = q_2 - qd_2; \]

Figure A-20: Embedded MATLAB function code in Figure A-18 (page 2)