

ADAPTIVE CONTROL OF MECHANICAL SYSTEMS  
WITH STATIC AND DYNAMIC FRICTION  
COMPENSATION

By

YONGLIANG ZHU

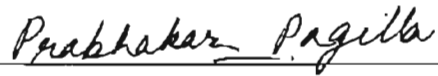
Bachelor of Science  
Zhejiang University  
Hanzhou, Zhejiang, P.R. China  
1988

Master of Science  
University of Science and Technology  
Hefei, Anhui, P.R. China  
1991

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December, 2001

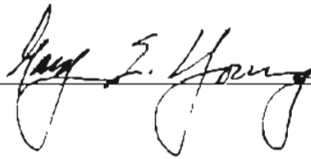
ADAPTIVE CONTROL OF MECHANICAL SYSTEMS  
WITH STATIC AND DYNAMIC FRICTION  
COMPENSATION

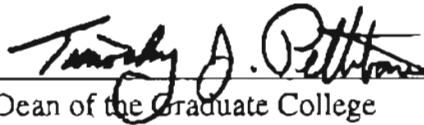
Thesis Approved:



Thesis Adviser







Dean of the Graduate College

## ACKNOWLEDGMENTS

I wish to express my sincerest appreciation to my advisor, Dr. Prabhakar R. Pagilla for intelligent supervision, constructive guidance, inspiration, and friendship throughout the development of this investigation. Without his help and supervision, I would not have completed this work.

I would like to extend my warmest thanks to my master committee members: Dr. Gary Young and Dr. E.A. Misawa for their help and guidance.

I wish to thank my colleagues at Oklahoma State University Robotics Research Lab: Rammishly V. Dividedly, Fu Yet, Gymkhana Nag, Sachem, and L. Prase C. Parera. They are among the finest people I know and are a joy to work with.

## TABLE OF CONTENTS

Chapter	Page
<b>1 Introduction</b>	<b>1</b>
<b>2 Dynamic Model with Friction</b>	<b>4</b>
2.1 Robot Dynamics . . . . .	4
2.2 Static Friction Models . . . . .	6
2.3 The Dahl Model . . . . .	8
2.4 The Bliman-Sorine Model . . . . .	9
2.5 The LuGre Model . . . . .	11
2.6 Complete Dynamic Model . . . . .	12
<b>3 Controller Design</b>	<b>14</b>
3.1 PD Control . . . . .	15
3.1.1 Simulation Results . . . . .	15
3.2 Model Based Adaptive Control . . . . .	17
3.3 Computed Torque Control . . . . .	20
3.4 Model Based Adaptive Control with Static Friction Compensation . . . . .	21
3.4.1 Simulation Results . . . . .	24
3.5 Adaptive Control with Dynamic Friction Compensation . . . . .	26
3.5.1 Simulation Results . . . . .	29
<b>4 Experimental Platform</b>	<b>32</b>
4.1 Hardware . . . . .	33

Chapter	Page
4.1.1 Motor Drivers . . . . .	34
4.1.2 DSP Servo Controller . . . . .	35
4.2 Software . . . . .	36
4.2.1 RPL Software . . . . .	36
4.2.2 Servo Software . . . . .	37
4.3 Manipulator Kinematics . . . . .	37
4.4 Manipulator Dynamics . . . . .	39
<b>5 Experiments</b>	<b>43</b>
5.1 Desired Trajectory Generation . . . . .	43
5.2 Experimental Condition . . . . .	45
5.3 Motor Characteristics . . . . .	46
5.4 Parameter Identification of the LuGre Friction Model . . . . .	48
5.5 Experimental Results . . . . .	51
5.5.1 PD Control . . . . .	51
5.5.2 Model Based Adaptive Control . . . . .	54
5.5.3 Computed Torque Control . . . . .	55
5.5.4 Model Based Adaptive Control with Static Friction Compensation . . . . .	56
5.5.5 Adaptive Control with Dynamic Friction Compensation . . . . .	59
5.6 Comparison of Experimental Results . . . . .	59
<b>6 Summary and Future Work</b>	<b>71</b>
6.1 Summary . . . . .	71
6.2 Future Work . . . . .	74
<b>BIBLIOGRAPHY</b>	<b>76</b>

Chapter	Page
<b>A Manipulator Application Software</b>	<b>79</b>
A.1 Servo.c . . . . .	79
A.2 Rpl.c . . . . .	100
<b>B MATLAB Programs</b>	<b>119</b>
B.1 Pddiff.m . . . . .	119
B.2 Pdcontrolload.m . . . . .	121
B.3 Adptvfpayload1.m . . . . .	123
B.4 Adptvf1.m . . . . .	124
B.5 Lugre.m . . . . .	127
B.6 Lugrediff.m . . . . .	131

## LIST OF FIGURES

Figure	Page	
2.1	Static friction models: (a) Coulomb, (b) Coulomb + Viscous, (c) Coulomb + Viscous + Stiction, (d) Stribeck . . . . .	6
2.2	The Dahl friction model . . . . .	8
2.3	Bliman-Sorine friction model . . . . .	10
3.1	PD control loop of the manipulator . . . . .	15
3.2	Trajectory in Cartesian coordinate . . . . .	16
3.3	Angular position and angular position error . . . . .	16
3.4	Tracking performance in the presence of payload . . . . .	17
3.5	Adaptive control + static friction compensation simulation result: position and tracking error . . . . .	24
3.6	Adaptive control + static friction compensation simulation result: estimated parameters $(\hat{p}_1, \hat{p}_2, \hat{p}_3)$ . . . . .	25
3.7	Adaptive control + static friction compensation simulation result: estimated parameters $(\hat{F}_v(1, 1), \hat{F}_v(2, 2), \hat{F}_c(1, 1), \hat{F}_c(2, 2))$ . . . . .	25
3.8	Adaptive control + static friction compensation simulation result: tracking performance in the presence of payload . . . . .	26
3.9	Simulation: position, velocity and tracking error . . . . .	30
3.10	Simulation: estimated parameters . . . . .	31
4.1	Sketch of the robot system . . . . .	32
4.2	Hardware architecture . . . . .	33

Figure	Page
4.3 Torque mode vs. velocity mode . . . . .	35
4.4 Connection diagram of the robot system . . . . .	36
4.5 Diagram of the work space of the manipulator . . . . .	37
4.6 Diagram of the two-link planar NSK manipulator . . . . .	39
5.1 Circle trajectory . . . . .	43
5.2 Diagram of $q_d, \dot{q}_d, \ddot{q}_d$ and $\tau$ for a circle trajectory . . . . .	44
5.3 Free body diagram of DC motor . . . . .	47
5.4 Identification of $\beta_2$ and $F_c$ for link 1 . . . . .	52
5.5 Identification of $\beta_2$ and $F_c$ for link 2 . . . . .	52
5.6 Identification of $\omega_s$ for link 1 . . . . .	53
5.7 Identification of $\omega_s$ for link 2 . . . . .	53
5.8 Identification of $\beta_0$ for link 1 . . . . .	54
5.9 Identification of $\beta_0$ for link 2 . . . . .	55
5.10 PD control experimental result (circle trajectory) . . . . .	56
5.11 Adaptive control experimental result: velocity and tracking error (circle trajectory) . . . . .	57
5.12 Adaptive control experimental result: estimated parameters ( $\hat{p}_1, \hat{p}_2, \hat{p}_3$ ) (circle trajectory) . . . . .	57
5.13 Computed torque control experimental result: velocity and tracking error (circle trajectory) . . . . .	58
5.14 Adaptive control + static friction compensation experimental result: velocity and tracking error (circle trajectory) . . . . .	58
5.15 Adaptive control + static friction compensation experimental result: estimated parameters ( $\hat{F}_v(1, 1), \hat{F}_v(2, 2), \hat{F}_c(1, 1), \hat{F}_c(2, 2)$ ) (circle trajectory) . . . . .	59



Figure	Page
5.16 Adaptive control with dynamic friction compensation experimental result: velocity and tracking error (circle trajectory) . . . . .	60
5.17 Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{p}_1, \hat{p}_2, \hat{p}_3$ ) (circle trajectory) . . . . .	60
5.18 Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{Q}_0(1, 1), \hat{Q}_1(1, 1), \hat{Q}_3(1, 1)$ ) (circle trajectory) . . . . .	61
5.19 Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{Q}_0(2, 2), \hat{Q}_1(2, 2), \hat{Q}_3(2, 2)$ ) (circle trajectory) . . . . .	61
5.20 Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{z}_{01}, \hat{z}_{11}, \hat{z}_{02}, \hat{z}_{12}$ ) (circle trajectory) . . . . .	62
5.21 Comparison of tracking errors for link 1 (circle trajectory) . . . . .	64
5.22 Comparison of tracking errors for link 2 (circle trajectory) . . . . .	64
5.23 Tracking error and its 2-norm for link 1 (circle trajectory) . . . . .	65
5.24 Tracking error and its 2-norm for link 2 (circle trajectory) . . . . .	65
5.25 Desired trajectories for link 1 and 2 (Sine trajectory) . . . . .	66
5.26 Comparison of tracking errors for link 1 (Sine trajectory) . . . . .	66
5.27 Comparison of tracking errors for link 2 (Sine trajectory) . . . . .	67
5.28 Estimated static friction parameters (Sine trajectory) . . . . .	67
5.29 Estimated dynamic friction parameters ( $\hat{Q}_0, \hat{Q}_1, \hat{Q}_3$ ) for link 1 (Sine trajectory) . . . . .	68
5.30 Estimated dynamic friction parameters ( $\hat{Q}_0, \hat{Q}_1, \hat{Q}_3$ ) for link 2 (Sine trajectory) . . . . .	68
5.31 Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{z}_{01}, \hat{z}_{11}, \hat{z}_{02}, \hat{z}_{12}$ ) (sine trajectory) . . . . .	69
5.32 Tracking error and its 2-norm for link 1 (Sine trajectory) . . . . .	69
5.33 Tracking error and its 2-norm for link 2 (Sine trajectory) . . . . .	70

## NOMENCLATURE

$e = q - q_d$	:	Position tracking error in joint space
$\dot{e} = \dot{q} - \dot{q}_d$	:	Velocity tracking error in joint space
$f$	:	External force
$q$	:	Real position in joint space
$\dot{q}$	:	Real velocity in joint space
$\ddot{q}$	:	Real acceleration in joint space
$q_d$	:	Desired position in joint space
$\dot{q}_d$	:	Desired velocity in joint space
$\ddot{q}_d$	:	Desired acceleration in joint space
$q_r$	:	Reference position in joint space
$\dot{q}_r$	:	Reference velocity in joint space
$\ddot{q}_r$	:	Reference acceleration in joint space
$e_v$	:	Reference velocity error
$M(q)$	:	Mass matrix of robot
$\widehat{M}(q)$	:	Estimate of $M$
$C(q, \dot{q})$	:	Matrix composed of Coriolis and centripetal terms of robot
$\widehat{C}(q, \dot{q})$	:	Estimate of $C$
$g(q)$	:	Gravity vector
$\widehat{g}(q)$	:	Estimate of $g$

- $J(q)$  : Jacobian matrix
- $\tau$  : Control Torque
- $\Lambda_v$  : Derivative feedback gain
- $\Lambda_p$  : Proportional feedback gain
- $\beta$  : Manipulator parameters  $(p_1, p_2, p_3)^T$
- $\hat{\beta}$  : Estimates of  $\beta$
- $\tilde{\beta}$  : Estimates error  $= \hat{\beta} - \beta$
- $Q_i$  : Friction gain
- $\hat{Q}_i$  : Estimate of  $Q_i$
- $F_c$  : Coulomb friction parameter
- $F_v$  : Viscous friction parameter
- $v_s$  : Stribeck velocity
- $\gamma, \alpha$  : Parameters in Dahl friction model
- $x$  : Displacement
- $v$  : Velocity
- $z, z_{ij}$  : Internal state of LuGre friction model
- $\hat{z}, \hat{z}_{ij}$  : Estimate of  $z, z_{ij}$
- $\tilde{z}, \tilde{z}_{ij}$  : Estimate error  $\tilde{z} = \hat{z} - z, \tilde{z}_{ij} = \hat{z}_{ij} - z_{ij}$
- $\beta_i$  : Friction parameter
- $f_f$  : Friction force/torque

# CHAPTER 1

## Introduction

Robotic applications sometimes involve unknown and hazardous environment, which require demanding control performance that cannot be achieved by conventional robot controllers. The requirements of accurate and stable end-effector trajectory tracking of high-speed manipulators under dynamic and payload uncertainties, sensor noise and external disturbances, have initiated the research in adaptive robot controllers; a vast amount of literature exists in the area ([1], [7] [3] [4]). Within this context, two major classes of adaptive control schemes can be identified, namely, the Performance-Based Adaptive Control (non-regressor methods) and the Model-Based Adaptive Control (regressor methods). In the performance-based approach, the controller gains are adjusted based on a system performance index. These controllers ordinarily need very little information about dynamic parameters and system structure. They are simple, requiring little real-time computations and have shown good tracking performance in laboratory experiments.

The model-based approach consists of a nonlinear compensation of the robot model, whose dynamic parameters are estimated through an adaptation law, generally derived from Lyapunov Theory, which guarantees asymptotic stability. Adaptive control techniques are robust to parameter uncertainty, but not robust to unmodeled dynamics and disturbance. More recently, Fuzzy control [19], Neural Network control [20], Learning control ([21],[22]) techniques have been proposed to handle the trajectory tracking problem of the robot.

The turning point in the solution of tracking problems in the adaptive control law was proposed by Slotine and Li ([2], [3]), which consisted of a linear part PD-type and a non-

linear model-based part which utilizes feedforward and feedback actions based on position and velocity data. The control law requires position and velocity measurement, but avoids inverse inertia matrix computation and acceleration measurement which is usually not possible in many robots. The applied torque is usually filtered to avoid high frequency noise before applying to the actuators of the robot.

High accuracy tracking cannot be achieved if friction effect is not considered properly. Friction is a complicated phenomenon, which exhibits coulomb friction, viscous friction and nonlinearity at low velocity. To cancel the friction effect, friction compensation is usually used ([10]-[16]). The classic friction models do not sufficiently describe all the dynamic effects of friction, such as pre-sliding, the friction lag and Stribeck effect. These phenomena all occur in the low velocity and pre-sliding region. Many researchers are still working on friction modelling, yet there is no general agreement on which one is the best. To incorporate more dynamic friction characteristics when designing a friction compensation, three empirical phenomenological models are usually applied to the physical process ([10], [11], [12]). They are the Dahl model, the Bliman-Sorine model, and the LuGre model. The Bliman-Sorine model and the LuGre model are extensions of the Dahl model. The LuGre model gives richer behavioral description of friction phenomena than the Dahl model and the Bliman-Sorine model. The Bliman-Sorine model may be problematic to use because of its poor damping properties at zero crossings of velocity[10].

The previous work dealing with the friction compensation has the following limitations: (1) only static friction was considered, i.e, only static friction behaviors such as viscous friction, Coulomb friction and Stribeck effect are considered, (2) simple mechanical structure like one link structure was considered when dynamic friction compensation was applied, and the dynamic model of the mechanical system studied is assumed without unknown parameters, and further (3) very little has been done to experimentally investigate the proposed designs and methodologies for multi-link robot applications.

This thesis has not only considered theoretical developments but also investigated the

validity of the theoretical developments via extensive experimentation. In the adaptive control with dynamic friction compensation, both the robot model parameter uncertainty and the dynamic friction parameter uncertainty are considered for multi-link robots. Dynamic parameter uncertainties and friction are the principal sources of the tracking error. Robots usually face uncertainties on parameters describing the dynamic properties of the grasped load, such as moments of inertia or exact position of the center of the mass in the end-effector. Simulation and experiments based on the proposed adaptive controller with friction compensation techniques are conducted on a two-link planar manipulator.

This work has the following contributions: (1) adaptive position control with static friction compensation which considers Coulomb and viscous effects was proposed; (2) Adaptive position control with dynamic friction compensation, where the LuGre friction model was applied, was proposed; (3) extensive experiments were conducted to investigate different controllers for robotic tracking task, and their results were compared; (4) different friction models commonly used were investigated; (5) NSK manipulator system software was studied, and some modifications were made. This will facilitate the future work in this manipulator.

The thesis is organized as follows. Chapter 2 describes the robot dynamics and friction dynamics. Chapter 3 introduces five controllers used for investigation in this thesis. The control law for each controller is given and stability is proved. Chapter 4 discusses the control system architecture of the two-link manipulator, on which the proposed controllers are implemented. Chapter 5 provides the experimental results. Summary and future research are given in Chapter 6. Appendix gives the source code written for the experiment and simulation.

## CHAPTER 2

### Dynamic Model with Friction

Dynamic modelling plays an important role in control design. In this chapter, the dynamics of a robot with friction is developed. Section 2.1 discusses the robot dynamics derived based on Euler-Lagrange equations, and some properties of the robot dynamic model, which are essential for control. High performance of position tracking control of robot system cannot be achieved if friction phenomena are not properly taken into account. Failing to compensate for friction in robot applications can lead to large tracking errors and possibly limit cycles when velocity reversals (velocity crossing zero point) are demanded. Understanding of friction is very important for friction compensation design in control systems. However, friction is a very complicated phenomena and its parameters vary with temperature and age. Various static friction models are discussed in Section 2.2. The dynamic friction models are being actively studied, and there is no general agreement on which model is the best to be employed. To incorporate more dynamic friction characteristics, three empirical phenomenological models are usually applied to the physical process ([10], [11], [12]). They are the Dahl model, the Bliman-Sorine model, and the LuGre model. Section 2.3, 2.4 and 2.5 give detailed discussion of these three friction model. The complete dynamic model with friction is presented in Section 2.6.

#### 2.1 Robot Dynamics

Let the kinetic and potential energy function of an  $n$ -link robot be given by  $\mathcal{K}(q, \dot{q}) = \frac{1}{2} \dot{q}^T M(q) \dot{q}$  and  $\mathcal{P}(q)$ , where  $q \in \mathbb{R}^n$ ,  $\dot{q} \in \mathbb{R}^n$  are the generalized position and velocity, respectively, and  $M(q) \in \mathbb{R}^{n \times n}$  is the symmetric positive definite mass matrix. The dynamics

of the robot is given by

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau + J^T(q)f + f_f(q, \dot{q}) \quad (2.1)$$

where  $C(q, \dot{q})$  is the matrix composed of Coriolis and centripetal terms,  $g(q)$  is the gravity vector,  $\tau$  is the vector of generalized forces applied by the motors at each joint of the robot,  $f_f(q, \dot{q})$  is the vector of friction torque.  $f$  represents the vector of external forces and  $J(q)$  is the Jacobian of the manipulator.

The dynamic model of the robot is assumed to satisfy the following well known properties:

- The inertia matrix  $M(q)$  is a symmetric positive definite matrix, and is bounded, i.e.,

$$\lambda_{min}I \leq M(q) \leq \lambda_{max}I \quad (2.2)$$

where  $\lambda_{min}$  and  $\lambda_{max}$  denote the strictly positive finite minimum and maximum eigenvalue of  $M(q)$  for all  $q$ , respectively.

- The matrix of  $\frac{1}{2}\dot{M}(q) - C(q, \dot{q})$  is skew-symmetric, that is,

$$z^T \left\{ \frac{1}{2}\dot{M}(q) - C(q, \dot{q}) \right\} z = 0 \quad (2.3)$$

for any vector  $z \in \mathbb{R}^n$

- The left-hand-side of the dynamics given by (2.1) is linear in terms of coupled manipulator inertial parameters, and can be written as

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = Y(q, \dot{q}, \ddot{q})\beta \quad (2.4)$$

where  $\beta \in \mathbb{R}^p$  is a coupled manipulator parameter vector, and  $Y(q, \dot{q}, \ddot{q})$  is a  $n \times p$  regressor matrix.

The first two properties are essential in Lyapunov stability analysis. The adaptive control algorithms are based on the linear parameterization property of the dynamics of robot manipulator.



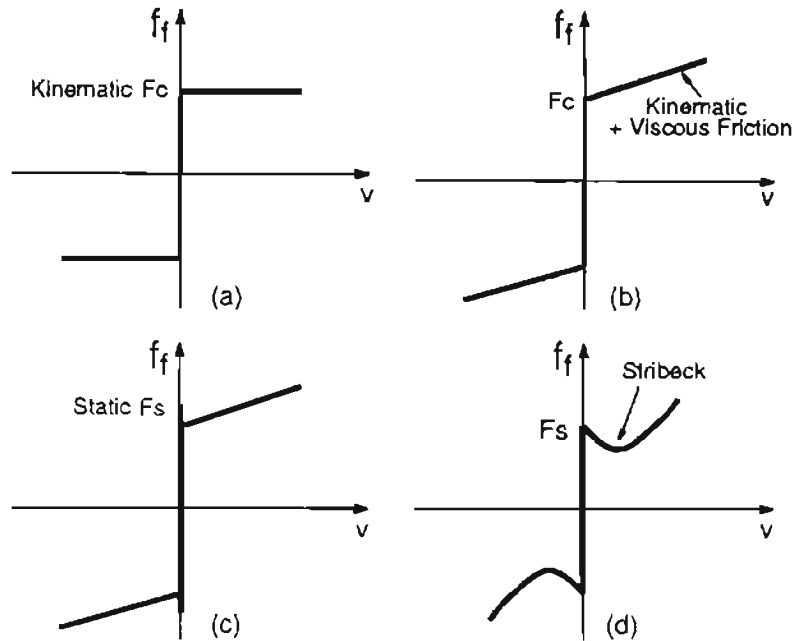


Figure 2.1: Static friction models: (a) Coulomb, (b) Coulomb + Viscous, (c) Coulomb + Viscous + Stiction, (d) Stribeck

## 2.2 Static Friction Models

Figure 2.1 shows the diagram of the four static friction models. In Figure 2.1, (a), (b), (c) are the classic friction models which cover the Coulomb, Coulomb + Viscous, Coulomb + Viscous + Stiction effects, respectively, and the Stribeck effect is included in (d). In dynamic systems, the Coulomb friction is often modelled as a piece-wise function which is a positive function for positive velocity, a negative function for negative velocity, and zero for zero velocity. Although Coulomb friction provides constant friction for non-zero velocity, it introduces discontinuity at zero velocity. It is like a “relay nonlinearity”. This nonlinearity may produce limit cycles in closed-loop control systems. As a result, poor control accuracy is achieved.

Viscous friction is represented as a linear function of velocity. It results from the viscous behavior of a fluid lubricant layer between two rubbing surfaces. Static friction is the force required to initiate motion from rest. The magnitude of the static friction is greater than

that of kinematic friction. Classic friction compensation usually handles only the Coulomb effect because the model is simple and this kind of friction effect is dominant in mechanical systems if the velocity does not frequently cross the zero point.

The friction models shown in Figure 2.2 (a), (b), (c), (d) are mathematically represented in the following equations.

(a) Coulomb friction

$$f_f = F_c \operatorname{sgn}(v) \quad (2.5)$$

(b) Coulomb + Viscous friction

$$f_f = F_c \operatorname{sgn}(v) + F_v v \quad (2.6)$$

(c) Coulomb + Viscous + Stiction friction

$$f_f = \begin{cases} F_c \operatorname{sgn}(v) + F_v v & \text{for } v \neq 0 \\ F_s \operatorname{sgn}(F) & \text{for } v = 0 \end{cases} \quad (2.7)$$

(d) Coulomb + Viscous + Stribeck

$$f_f = F_v v + F_c \operatorname{sgn}(v) + (F_s - F_c) e^{-(v/v_s)^2} \operatorname{sgn}(v) \quad (2.8)$$

where

$F_c$ : Coulomb friction coefficient

$F_s$ : Stiction friction coefficient

$F_v$ : Viscous friction coefficient

$v_s$ : Stribeck velocity constant

$F$ : Applied tangential force

$v$ : Velocity

$\operatorname{sgn}(\cdot)$ : Sign function

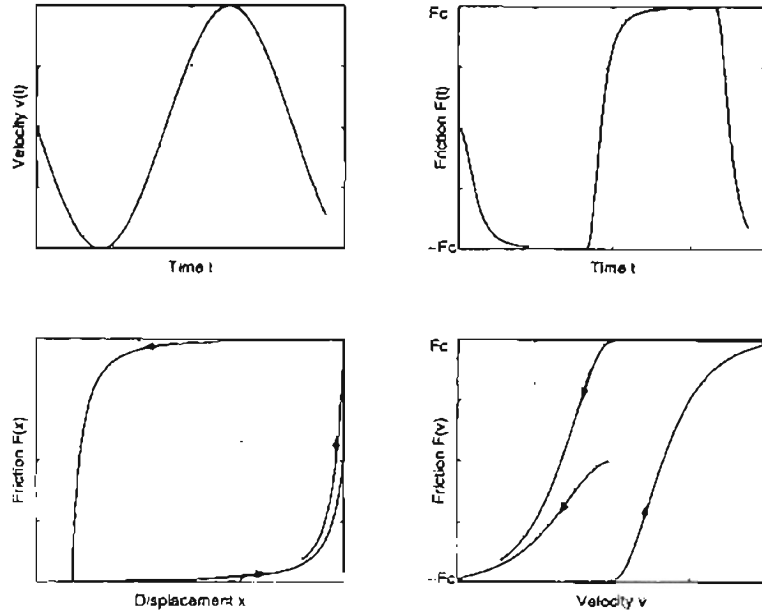


Figure 2.2: The Dahl friction model

### 2.3 The Dahl Model

In the late 1960's, Dahl provided a dynamic friction model (known later as the Dahl model) given by

$$\frac{df_f}{dx} = \sigma \left| 1 - \frac{f_f}{F_c} \operatorname{sgn} \dot{x} \right|^\alpha \quad (2.9)$$

where

- $\sigma$ : Rest stiffness parameter
- $\alpha$ : Solid friction exponent parameter
- $x$ : Relative displacement

In time domain, Equation (2.9) can be written as

$$\frac{df_f}{dt} = \sigma \left| 1 - \frac{f_f}{F_c} \operatorname{sgn}(\dot{x}) \right|^\alpha \dot{x} \quad (2.10)$$

The exponent  $\alpha$  is empirically given by Dahl as  $\alpha \approx 1.5$ .

The Dahl model is inspired by the stress-strain characteristics from solid mechanics. Figure 2.2 shows Dahl's friction force response for a sinusoid velocity. It can be seen that the Dahl model behaves like a spring for small deflection  $x$ , and reaches Coulomb friction

model for large deflection. Friction force will monotonically increase/decrease to  $\pm F_s$ . Coulomb friction model neglects the small deflection zone behavior; and the friction force jumps to  $\pm F_s$  directly.

The steady state version of the Dahl model for  $\sigma=1$  is

$$f_f = F_c \text{sgn}(v) \quad (2.11)$$

which is the Coulomb friction model. The Dahl model accounts for Coulomb friction but it does not include the Stribeck effect.

## 2.4 The Bliman-Sorine Model

Bliman and Sorine developed a dynamic model which emphasizes the importance of rate independence, i.e., the friction force is not explicitly dependent on the velocity, but only on the distance travelled after a velocity zero crossing. The model is expressed as a linear system in the space variable  $s$ , which is given by

$$\begin{aligned} s &= \int_0^t |v(\tau)| d\tau \\ \frac{dx_s}{ds} &= Ax_s + B \text{sgn}(v) \\ f_f &= Cx_s \end{aligned} \quad (2.12)$$

where A, B, C are matrices, and  $v$  is the velocity.

For the second order model, it is suggested to choose the matrices as follows:

$$A = \begin{bmatrix} -\frac{1}{\eta\epsilon_f} & 0 \\ 0 & -\frac{1}{\epsilon_f} \end{bmatrix}, \quad B = \begin{bmatrix} \frac{f_1}{\eta\epsilon_f} \\ -\frac{f_2}{\epsilon_f} \end{bmatrix} \quad \text{and} \quad C = [1 \quad 1] \quad (2.13)$$

In time domain, the time derivative of friction force is

$$\frac{df_f}{dt} = \frac{f_1}{\eta\epsilon_f} \left(1 - \frac{x_{s1}}{f_1} \text{sgn}(v)\right) v - \frac{f_2}{\epsilon_f} \left(1 - \frac{x_{s2}}{f_2} \text{sgn}(v)\right) v \quad (2.14)$$

$$\begin{bmatrix} \frac{dx_{s1}}{dt} \\ \frac{dx_{s2}}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{|v|}{\eta\epsilon_f} & 0 \\ 0 & -\frac{|v|}{\epsilon_f} \end{bmatrix} \begin{bmatrix} x_{s1} \\ x_{s2} \end{bmatrix} + \begin{bmatrix} \frac{f_1}{\eta\epsilon_f} \\ -\frac{f_2}{\epsilon_f} \end{bmatrix} v \quad (2.15)$$

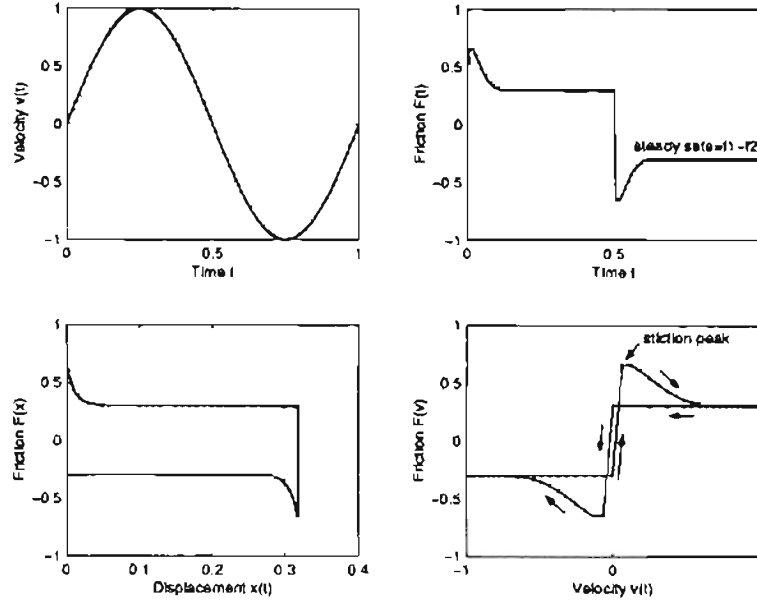


Figure 2.3: Bliman-Sorine friction model

Hence, the second order Bliman-Sorine model can be viewed as a parallel connection of a fast and a slow Dahl model.

Second order Bliman-Sorine model has the following properties:

- The difference  $f_1 - f_2$  determines the amplitude of friction at steady state. This can be observed from Equation (2.15). At steady state,  $x_{s1} = f_1 \text{sgn}(v)$ ,  $x_{s2} = -f_2 \text{sgn}(v)$ , hence  $f_f = x_{s1} + x_{s2} = (f_1 - f_2) \text{sgn}(v)$ .
- Bliman-Sorine model incorporates the stiction effect; this can be deduced by linearizing Equation (2.14) at zero states and zero velocity:

$$\delta f_f = \frac{f_1}{\eta \epsilon_f} \delta v \delta t - \frac{f_2}{\epsilon_f} \delta v \delta t = \left( \frac{f_1}{\eta \epsilon_f} \delta x_{s1} - \frac{f_2}{\epsilon_f} \delta x_{s2} \right) \text{sgn}(\delta v) \quad (2.16)$$

Figure 2.3 shows the characteristics of the Bliman-Sorine model due to a sinusoid velocity input, where the stiction effect and Dahl's effect at low velocity (at small motion around zero crossings of velocity, friction behaves like a spring) can be observed.

## 2.5 The LuGre Model

The LuGre model was presented in [11]. The standard formulation describing the dynamic friction is as follows:

$$\begin{aligned}\frac{dz}{dt} &= v - \beta_0 \frac{|v|}{g(v)} z \\ g(v) &= F_c + (F_s - F_c) e^{-(v/v_s)^2} \\ f_f &= \beta_0 z + \beta_1 \dot{z} + f(v)\end{aligned}\tag{2.17}$$

where

$z$ : The average deflection of the bristles which is unmeasurable

$\beta_0$ : Stiffness coefficient

$\beta_1$ : Velocity dependent damping coefficient

$f(v)$ : Function including effects such as viscous friction and lubrication

$v_s$ : Stribeck velocity

$F_s$ : Stiction friction force constant

$F_c$ : Coulomb friction force constant

The function  $g(v)$  is positive and depends on many factors such as material properties, lubrication, temperature;  $g(v)$  will decrease monotonically from  $g(0)$  when  $v$  increases;  $g(v)$  can be chosen to describe different friction effects;  $g(v)$  chosen in Equation (2.17) characterizes the Stribeck effect. The function  $f(v)$  is usually chosen as  $f(v) = \beta_2 v$ ; it represents viscous friction.

The LuGre model considers the dynamic effects of friction arising out of the deflection of bristles which model the asperities between two contacting surfaces. The contacting surfaces can be visualized as consisting of many asperities at the microscopic level. When a slight tangential force is applied, the bristles will deflect like elastic springs. If the force is large enough, the bristles start to slip.

If  $g(v) = F_c/\beta_0$  and  $\beta_1 = \beta_2 = 0$ , Equation (2.17) becomes

$$\frac{df_f}{dt} = \beta_0 v \left(1 - \frac{f_f}{F_c} \text{sgn}(v)\right)\tag{2.18}$$

the LuGre model reduces to the Dahl model. The steady state of the LuGre model is:

$$z = \frac{g(v)}{\beta_0} \text{sgn}(v) \quad (2.19)$$

$$f_f = g(v) \text{sgn}(v) + f(v)$$

The LuGre model captures many aspects of friction, including stiction, stick slip, stribbeck, hysteresis and zero slip displacement. The LuGre model is characterized by six parameters:  $\beta_0$ ,  $\beta_1$ ,  $\beta_2$ ,  $v_s$ ,  $F_c$  and  $F_s$ . The identification procedure for these parameters is described in [13]. This method is based on a series of experiments in different friction regimes. The constants  $\beta_0$  and  $\beta_1$  are identified in the stick-slip region, and  $v_s$ ,  $F_c$  and  $F_s$  are identified in steady state region. The function  $f(v)$  is identified from break away experiments.

## 2.6 Complete Dynamic Model

Substituting the static and dynamic models, described in Equation (2.6) and Equation (2.17), into the robot dynamics, Equation (2.1), yields the following complete dynamic model. 3.

- Complete dynamics with static friction model is

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + F_c \text{sgn}(\dot{q}) + F_v \dot{q} = \tau \quad (2.20)$$

$$f_f = F_c \text{sgn}(\dot{q}) + F_v \dot{q} \quad (2.21)$$

where  $F_v = \text{diag}(F_{v1}, F_{v2}, \dots, F_{vn})$  and  $F_c = \text{diag}(F_{c1}, F_{c2}, \dots, F_{cn})$  are diagonal matrices,  $F_{vi}$  and  $F_{ci}$  are the Coulomb and viscous friction coefficients of the  $i^{\text{th}}$  link, respectively.

- Complete dynamics with dynamic friction model is

$$\frac{dz}{dt} = \dot{q} - \Psi z \quad (2.22)$$

$$f_f = Q_0 z + Q_1 \dot{z} + Q_2 \dot{q} \quad (2.23)$$

where  $Q_0, Q_1$  and  $Q_2$  are diagonal matrices whose diagonal elements correspond to the friction constants as given by (2.17), and  $\Psi$  is a diagonal matrix and is given by

$$\Psi = \text{diag}(\gamma_1|\dot{q}_1|/g_1(\dot{q}_1), \dots, \gamma_n|\dot{q}_n|/g_n(\dot{q}_n))$$

$$g_i(\dot{q}_i) := F_{ci} + (F_{si} - F_{ci})e^{-(\dot{q}_i/\omega_{si})^2}$$

where  $\text{diag}(a_1, \dots, a_n)$  represents the diagonal matrix with diagonal elements  $a_1$  through  $a_n$  and  $\omega_{si}$  is the Stribeck velocity corresponding to joint  $i$ . Substitution of the dynamic friction model (2.22) and (2.23) into the robot dynamics and simplifying we obtain

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + Q_3\dot{q} + Q_0z - Q_1\Psi z = \tau \quad (2.24)$$

where  $Q_3 = Q_1 + Q_2$ .



## CHAPTER 3

### Controller Design

This chapter will investigate various control designs for the complete robot model given in Chapter 2. Five types of control designs are considered: (1) PD controller (**PD**), (2) computed torque control (**CT**), (3) adaptive control (**AC**), (4) adaptive control with static friction compensation (**ACS**) and (5) adaptive control with dynamic friction compensation (**ACD**). Model-based computed torque control and adaptive control are well known in literature[18] and a brief explanation of each is given. The adaptive control design is augmented with static friction compensation and dynamic friction compensation and stability of the closed-loop system is shown in each case. The goal of each designed controller is to get good tracking performance, i.e., achieve the smallest tracking error that is possible.

PD controller is the most commonly used controller in industrial environment because of no dynamic model requirement and easy implementation. Simulation and experimental results of PD controller are shown; these results are used to compare with that of other control design.

In computed torque control, the robot parameters are assumed to be known. The other three controller (**AC**, **ACS**, **ACD**) do not require accurate knowledge of robot parameters. Furthermore, **ACS** and **ACD** consider static friction compensation and dynamic friction compensation, respectively.

Some simulation results on a two-link manipulator (refer to Section 4) are shown in this Chapter. The experimental results are presented in Chapter 5.

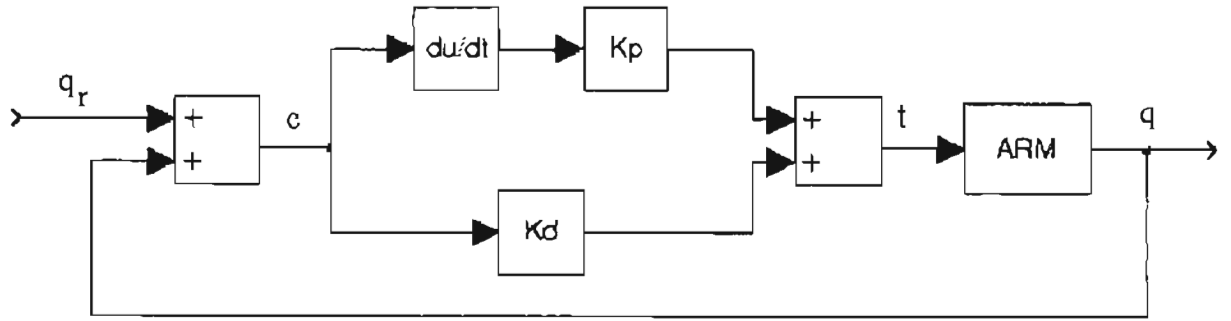


Figure 3.1: PD control loop of the manipulator

### 3.1 PD Control

Numerous control methods such as adaptive control, neural network control, fuzzy control have been studied. However, controllers in industrial and commercial robot manipulators are still employing the classical PID controllers because of their simplicity and ease of implementation.

The block diagram of the PD controller is shown in Figure 3.1. The motor torque  $\tau$  for the PD control law is given by

$$\tau = K_p e + K_d \dot{e} \quad (3.1)$$

where  $e = q_d - q$  is the tracking error,  $q_d$  and  $q$  is the desired and actual angular position respectively.

#### 3.1.1 Simulation Results

This simulation shows that the robot tracks a desired trajectory, which is a circle starting from point (0.58,0)m and ending at this point in a time interval of 4 seconds

PD controller is simulated with fixed parameters  $p_1$ ,  $p_2$  and  $p_3$ . Figure 3.2 to 3.4 show the simulation results.

Figure 3.4 shows the simulation result with a step payload disturbance. The payload

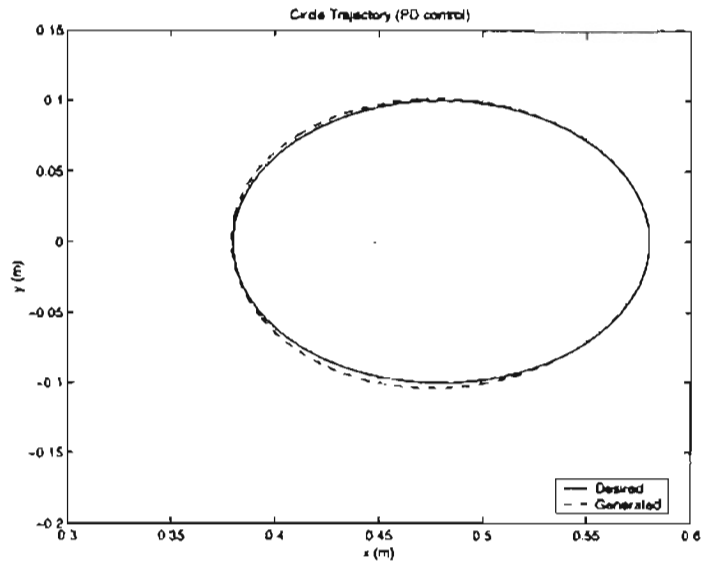


Figure 3.2: Trajectory in Cartesian coordinate

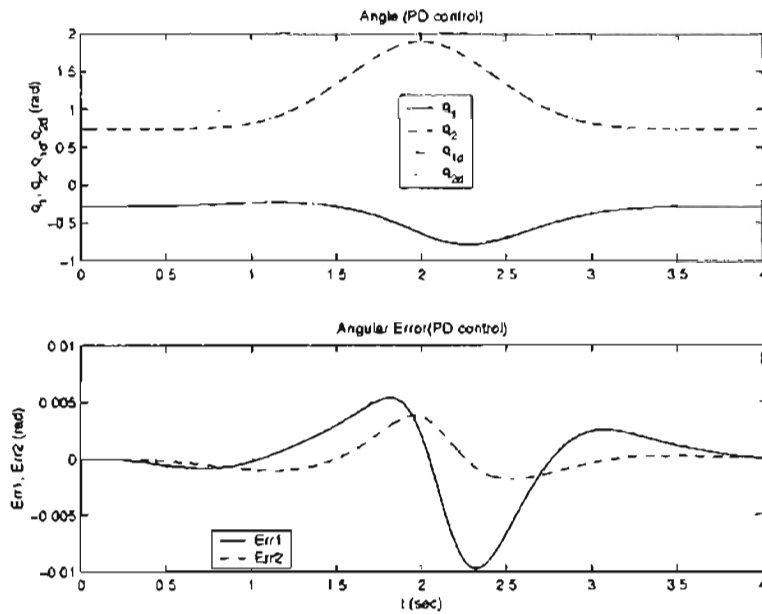


Figure 3.3: Angular position and angular position error

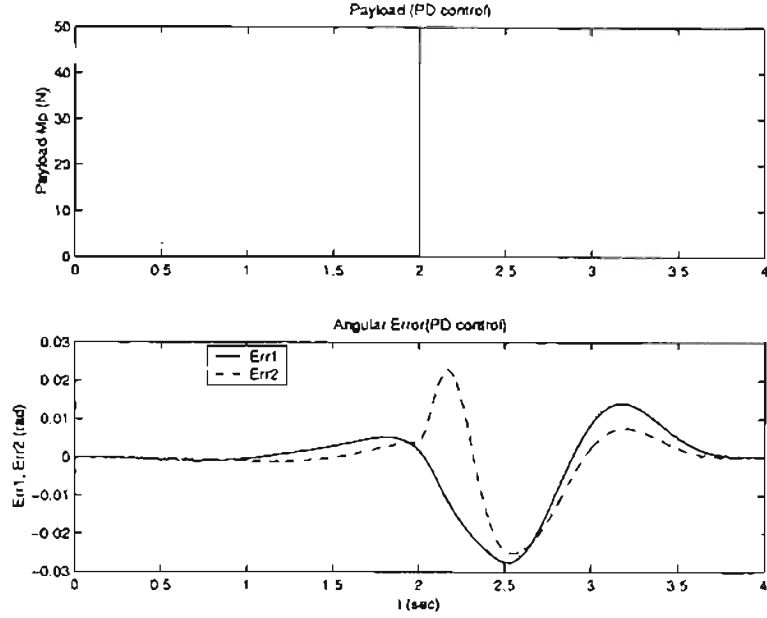


Figure 3.4: Tracking performance in the presence of payload

variation results in the change of the dynamic parameter  $p_1$ ,  $p_2$  and  $p_3$ . It can be seen that the tracking error increases in the presence of the payload variation.

### 3.2 Model Based Adaptive Control

In practice, there are uncertainties in the model parameters and non-ideal efforts such as friction, backlash, dead zones, etc. Adaptive control is effective in handling parameters uncertainties. This section investigates the adaptive control designed for trajectory tracking and gives the simulation results on a two-link planar manipulator.

Recall the dynamic model for the  $n$  degree-of-freedom robot:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + f_f = \tau \quad (3.2)$$

where  $q, \dot{q} \in \mathbb{R}^n$  is the generalized position and velocity vectors, respectively,  $M(q)$  is the generalized mass matrix,  $C(q, \dot{q})$  is the matrix composed of Coriolis and centrifugal terms,  $g(q)$  is the gravity vector,  $\tau \in \mathbb{R}^n$  is the vector of generalized input forces,  $f_f \in \mathbb{R}^n$  is the vector of friction forces.

The adaptive control law is given by

$$\tau = \widehat{M}(q)\ddot{q}_r + \widehat{C}(q, \dot{q})\dot{q}_r + \widehat{g}(q) - \Lambda_v e_v \quad (3.3)$$

where  $\widehat{M}(q)$ ,  $\widehat{C}(q, \dot{q})$ , and  $\widehat{g}(q)$  are the estimates of  $M(q)$ ,  $C(q, \dot{q})$ , and  $g(q)$ , respectively.  $\Lambda_v$  is a positive definite gain matrix, and

$$e_v = \dot{q} - \dot{q}_r \quad (3.4)$$

$$\dot{q}_r = \dot{q}_d - \Lambda_p e \quad (3.5)$$

$$\ddot{q}_r = \ddot{q}_d - \Lambda_p \dot{e} \quad (3.6)$$

$$e = q - q_d \quad (3.7)$$

where  $\Lambda_p$  is a positive definite gain matrix. The reference velocity error  $e_v$  is defined as the difference between actual joint velocity  $\dot{q}$  and reference velocity  $\dot{q}_r$ . Substituting the control law (3.3) into the robot dynamics (3.2) yields

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \widehat{M}(q)\ddot{q}_r + \widehat{C}(q, \dot{q})\dot{q}_r + \widehat{g}(q) - \Lambda_v e_v. \quad (3.8)$$

Subtracting  $[M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q)]$  from both sides of the above equation and simplifying,

$$\begin{aligned} & M(q)(\ddot{q} - \ddot{q}_r) + C(q, \dot{q})(\dot{q} - \dot{q}_r) + \Lambda_v e_v \\ &= (\widehat{M}(q) - M(q))\ddot{q}_r + (\widehat{C}(q, \dot{q}) - C(q, \dot{q}))\dot{q}_r + \widehat{g}(q) - g(q) \end{aligned} \quad (3.9)$$

Using the linear parameterization property yields

$$(\widehat{M}(q) - M(q))\ddot{q}_r + (\widehat{C}(q, \dot{q}) - C(q, \dot{q}))\dot{q}_r + \widehat{g}(q) - g(q) = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta} \quad (3.10)$$

For the planar manipulator considered in this thesis, related parameters are shown in the following.

$$\tilde{\beta} = \hat{\beta} - \beta = \begin{bmatrix} \hat{p}_1 - p_1 \\ \hat{p}_2 - p_2 \\ \hat{p}_3 - p_3 \end{bmatrix} \quad (3.11)$$

$$Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r) = \begin{bmatrix} \ddot{q}_{r1} & \ddot{q}_{r2} & \cos(q_2)(2\ddot{q}_{r1} - \ddot{q}_{r2}) - \sin(q_2)|\dot{q}_2\dot{q}_{r1} + (\dot{q}_1 + \dot{q}_{r2})\dot{q}_{r2}| \\ 0 & \ddot{q}_{r1} + \ddot{q}_{r2} & \cos(q_2)\ddot{q}_{r1} + \sin(q_2)\dot{q}_1\dot{q}_{r1} \end{bmatrix} \quad (3.12)$$

Substituting (3.10) into (3.9) yields error dynamics, which is

$$M(q)\dot{e}_v + C(q, \dot{q})e_v + \Lambda_v e_v = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta}. \quad (3.13)$$

Consider the following Lyapunov function candidate:

$$V(t) = \frac{1}{2}e_v^T M(q)e_v + \frac{1}{2}\tilde{\beta}^T \Gamma \tilde{\beta} > 0 \quad (3.14)$$

The time derivative of the Lyapunov function candidate along the trajectories of (3.13) is

$$\begin{aligned} \dot{V}(t) &= -e_v^T M(q)\dot{e}_v + \frac{1}{2}e_v^T \dot{M}(q)e_v + \tilde{\beta}^T \Gamma \dot{\tilde{\beta}} \\ &= e_v^T [Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta} - C(q, \dot{q})e_v - \Lambda_v e_v] + \frac{1}{2}e_v^T \dot{M}(q)e_v + \tilde{\beta}^T \Gamma \dot{\tilde{\beta}} \\ &= -e_v^T \Lambda_v e_v + e_v^T [\frac{1}{2}\dot{M}(q) - C(q, \dot{q})]e_v + \tilde{\beta}^T \Gamma \dot{\tilde{\beta}} + e_v^T Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta} \end{aligned} \quad (3.15)$$

Since  $[\frac{1}{2}\dot{M}(q) - C(q, \dot{q})]$  is a skew-symmetric matrix,  $e_v^T [\frac{1}{2}\dot{M}(q) - C(q, \dot{q})]e_v = 0$ , hence

$$\dot{V}(t) = -e_v^T \Lambda_v e_v + \tilde{\beta}^T \Gamma \dot{\tilde{\beta}} + e_v^T Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta}. \quad (3.16)$$

If we let

$$e_v^T Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta} + \tilde{\beta}^T \Gamma \dot{\tilde{\beta}} = 0, \quad (3.17)$$

then the time derivative of the estimated parameter error is given by

$$\dot{\tilde{\beta}} = -\Gamma^{-T} Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)e_v. \quad (3.18)$$

Since  $\beta$  is constant,  $\dot{\tilde{\beta}} = \dot{\hat{\beta}}$ , and hence,

$$\dot{\hat{\beta}} = -\Gamma^{-T} Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)e_v, \quad (3.19)$$

that is,

$$\widehat{\beta}(t) = \widehat{\beta}_0 - \int_0^t \Gamma^{-T} Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r) e_v du \quad (3.20)$$

where  $\widehat{\beta}_0$  is the initial value of  $\widehat{\beta}(t)$ .

Equation (3.20) gives the adaptation law for the parameters. Substituting Equation (3.17) into Equation (3.16) and simplifying yields the following equation:

$$\dot{V}(t) = -e_v^T \Lambda_v e_v \leq 0 \quad (3.21)$$

From Equation (3.14) and (3.21), it can be seen that  $V(t)$  is positive definite and  $\dot{V}(t)$  is negative definite, so  $V(t)$  is indeed a Lyapunov function. Because  $V(t)$  is a positive function that is bounded from below,  $e_v$  and  $\tilde{\beta}$  are bounded. Since the regressor matrix  $Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$  in (3.13) remains bounded and manipulator inertia matrix  $M(q)$  is never singular during the motion,  $\dot{e}_v$  is bounded from (3.13). This means that  $e_v$  is uniformly continuous and hence converges to zero asymptotically.

### 3.3 Computed Torque Control

In some robotic applications, robot parameters are known. The computed torque is commonly used to act as a feedforward part in the controller. This computed torque controller is given by

$$\tau = M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q) - \Lambda_v e_v \quad (3.22)$$

where  $\dot{q}_r$ ,  $e_v$ ,  $e$ ,  $q_d$ ,  $\dot{q}_d$  and  $\Lambda_v$  are defined the same as in Section 3.2. Substituting the control in Equation (3.22) into the robot dynamics in Equation (3.2) yields

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q) - \Lambda_v e_v. \quad (3.23)$$

The error dynamics is

$$M(q)\dot{e}_v + C(q, \dot{q})e_v = -\Lambda_v e_v. \quad (3.24)$$

Choose the Lyapunov function candidate

$$V(t) = \frac{1}{2} e_v^\top M(q) e_v > 0. \quad (3.25)$$

The time derivative of  $V(t)$  along the trajectories of error dynamics is

$$\dot{V}(t) = e_v^\top M(q) \dot{e}_v + \frac{1}{2} e_v^\top \dot{M}(q) e_v - e_v^\top \Lambda_v e_v \leq 0 \quad (3.26)$$

Thus,  $V$  is a Lyapunov function, i.e.,  $V$  is positive definite and  $\dot{V}$  is negative definite. Hence, all the internal signals,  $e, e_v$ , are bounded. Further, from (3.24)  $\dot{e}_v \in \mathcal{L}_\infty$ , and from (3.26)  $e_v \in \mathcal{L}_2$ . Therefore,  $e_v, \dot{e}_v \in \mathcal{L}_\infty$  and  $e_v \in \mathcal{L}_2$ , we conclude that  $e_v \rightarrow 0$  as  $t \rightarrow \infty$ . Since  $e_v = \dot{e} + \Lambda_p e$ , we have  $e \rightarrow 0$  as  $t \rightarrow \infty$ .

### 3.4 Model Based Adaptive Control with Static Friction Compensation

The controllers designed in the previous sections do not include friction compensation. High accuracy tracking in mechanical systems cannot be achieved if friction effects are not considered properly in the dynamic model used for control. Failure to compensate for friction in robotic applications may lead to large tracking errors and limit cycles when velocity reversals in the trajectory are required.

Consider robot dynamics in (3.2), and the static friction model

$$f_f = F_v \dot{q} + F_c \text{sgn}(\dot{q}), \quad (3.27)$$

where  $F_v$  is a diagonal matrix with diagonal elements representing static friction constants, and similarly  $F_c$  is a diagonal matrix with diagonal elements representing Coulomb friction constants for each link of the mechanical system.

The following adaptive control law is chosen for the robot based on the static friction model:

$$\tau = \widehat{M}(q) \ddot{q}_r + \widehat{C}(q, \dot{q}) \dot{q}_r + \widehat{g}(q) + \widehat{F}_v \dot{q}_r + \widehat{F}_c \text{sgn}(\dot{q}_r) - \Lambda_v e_v \quad (3.28)$$



where  $\widehat{F}_v$  and  $\widehat{F}_c$  are the estimates of  $F_v$  and  $F_c$ , respectively,  $\widehat{M}(q)$ ,  $\widehat{C}(q, \dot{q})$  and  $\widehat{g}(q)$  are the estimates of  $M(q)$ ,  $C(q, \dot{q})$ , and  $g(q)$ , respectively. Substituting the control law (3.28) into the robot dynamics and simplifying yields the following error dynamics:

$$\begin{aligned} M(q)\dot{e}_v + C(q, \dot{q})e_v + (F_v + \Lambda_v)e_v + F_c [\text{sgn}(\dot{q}) - \text{sgn}(\dot{q}_r)] \\ =: Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta} + Y_{fs}(\dot{q}_r)\tilde{\beta}_{fs} \end{aligned} \quad (3.29)$$

where  $\tilde{\beta} = \widehat{\beta} - \beta$ ,  $\tilde{\beta}_{fs} = \widehat{\beta}_{fs} - \beta_{fs}$  and  $Y_{fs}(\dot{q}_r) = [Y_1, Y_2]$ , where

$$Y_1 = \text{diag}(\dot{q}_{r1}, \dots, \dot{q}_{rn}),$$

$$Y_2 = \text{diag}(\text{sgn}(\dot{q}_{r1}), \dots, \text{sgn}(\dot{q}_{rn})).$$

Notice that since  $F_v$  and  $F_c$  are diagonal, the elements of the parameter vector  $\beta_{fs}$  are the diagonal elements of  $F_v$  followed by the diagonal elements of  $F_c$ .

Stability of the solutions of the closed-loop error dynamics (3.29) can be shown by considering the following Lyapunov function candidate:

$$V = \frac{1}{2}e_v^T M(q)e_v + \frac{1}{2}\tilde{\beta}^T \Gamma^{-1}\tilde{\beta} + \frac{1}{2}\tilde{\beta}_{fs}^T \Gamma_{fs}^{-1}\tilde{\beta}_{fs} \quad (3.30)$$

where  $\Gamma$  and  $\Gamma_{fs}$  are diagonal gain matrices. The time derivative of the Lyapunov function candidate along the trajectories of the error dynamics is given by

$$\begin{aligned} \dot{V} &= e_v^T M(q)\dot{e}_v + \frac{1}{2}e_v^T \dot{M}(q)e_v + \tilde{\beta}^T \Gamma^{-1}\dot{\tilde{\beta}} + \tilde{\beta}_{fs}^T \Gamma_{fs}^{-1}\dot{\tilde{\beta}}_{fs} \\ &= -e_v^T (F_v + \Lambda_v)e_v - e_v^T F_c (\text{sgn}(e_v + \dot{q}_r) - \text{sgn}(\dot{q}_r)) \\ &\quad + (e_v^T Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r) + \tilde{\beta}^T \Gamma^{-1})\tilde{\beta} \\ &\quad + (e_v^T Y_{fs}(\dot{q}_r) + \tilde{\beta}_{fs}^T \Gamma_{fs}^{-1})\tilde{\beta}_{fs} \end{aligned}$$

where the skew-symmetry property of the matrix  $[\frac{1}{2}\dot{M}(q) - C(q, \dot{q})]$  has been used in the simplification.

Notice that

$$\begin{aligned}
& e_v^T F_c(\text{sgn}(e_v + \dot{q}_r) - \text{sgn}(\dot{q}_r)) \\
&= \sum_{i=1}^n e_{vi} f_{ci} (\text{sgn}(\dot{q}_i) - \text{sgn}(\dot{q}_{ri})) \\
&= \sum_{i=1}^n f_{ci} (\dot{q}_i - \dot{q}_{ri}) (\text{sgn}(\dot{q}_i) - \text{sgn}(\dot{q}_{ri})) \\
&= \sum_{i=1}^n f_{ci} (\dot{q}_i \text{sgn}(\dot{q}_i) - \dot{q}_i \text{sgn}(\dot{q}_{ri}) + \dot{q}_{ri} \text{sgn}(\dot{q}_{ri}) - \dot{q}_{ri} \text{sgn}(\dot{q}_i)) \\
&= \sum_{i=1}^n f_{ci} (|\dot{q}_i| - \dot{q}_i \text{sgn}(\dot{q}_{ri}) + |\dot{q}_{ri}| - \dot{q}_{ri} \text{sgn}(\dot{q}_i)) \\
&\geq 0
\end{aligned}$$

Further if we let

$$e_v^T Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r) \tilde{\beta} + \tilde{\beta}^T \Gamma^{-1} \tilde{\beta} = 0,$$

$$e_v^T Y_{fs}(\dot{q}_r) \tilde{\beta}_{fs} + \tilde{\beta}_{fs}^T \Gamma_{fs}^{-1} \tilde{\beta}_{fs} = 0,$$

and since the parameter vectors  $\beta$  and  $\beta_{fs}$  are constant, the update law for the estimated manipulator and static friction parameters are given by

$$\dot{\hat{\beta}} = -\Gamma^T Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r) e_v, \quad (3.31)$$

$$\dot{\hat{\beta}}_{fs} = -\Gamma_{fs}^T Y_{fs}^T(\dot{q}_r) e_v. \quad (3.32)$$

As a result, the time derivative of the Lyapunov function candidate simplifies to

$$\begin{aligned}
\dot{V} &= -e_v^T (\Lambda_v + F_v) e_v - e_v^T F_c(\text{sgn}(e_v + \dot{q}_r) - \text{sgn}(\dot{q}_r)) \\
&\leq 0
\end{aligned} \quad (3.33)$$

Thus,  $V$  is a Lyapunov function, i.e.,  $V$  is positive definite and  $\dot{V}$  is negative definite. Hence, all the internal signals,  $e, e_v, \hat{\beta}, \hat{\beta}_{fs}$ , are bounded. Further, from (3.29)  $\dot{e}_v \in \mathcal{L}_\infty$ , and from (3.33)  $e_v \in \mathcal{L}_2$ . Therefore,  $e_v, \dot{e}_v \in \mathcal{L}_\infty$  and  $e_v \in \mathcal{L}_2$ , we conclude that  $e_v \rightarrow 0$  as  $t \rightarrow \infty$ . Since  $e_v = \dot{e} + \Lambda_p e$ , we have  $e \rightarrow 0$  as  $t \rightarrow \infty$ .

### 3.4.1 Simulation Results

The simulation of the proposed model-based adaptive controller with static friction compensation is conducted before this controller is applied to real plant. Figure 3.5 shows the tracking error for the circle trajectory. Figure 3.6 and 3.7 show the estimated parameters. The tracking error due to the payload introduced at time  $t = 2$ sec. at the end-effector is shown in Figure 3.8.

Compared to the corresponding results shown in Figure 3.2, Figure 3.3 and Figure 3.4 of *PD* control, it can be observed that the proposed adaptive controller offers better tracking error in the face of payload disturbance. Figure 3.8 and 3.4 show that the *PD* controller with fixed *P* and *D* gains is very sensitive to the dynamics uncertainties caused by the changing payload. However, by using adaptive controller, the tracking error increase due to the introduction of the payload is much smaller than that in *PD* control.

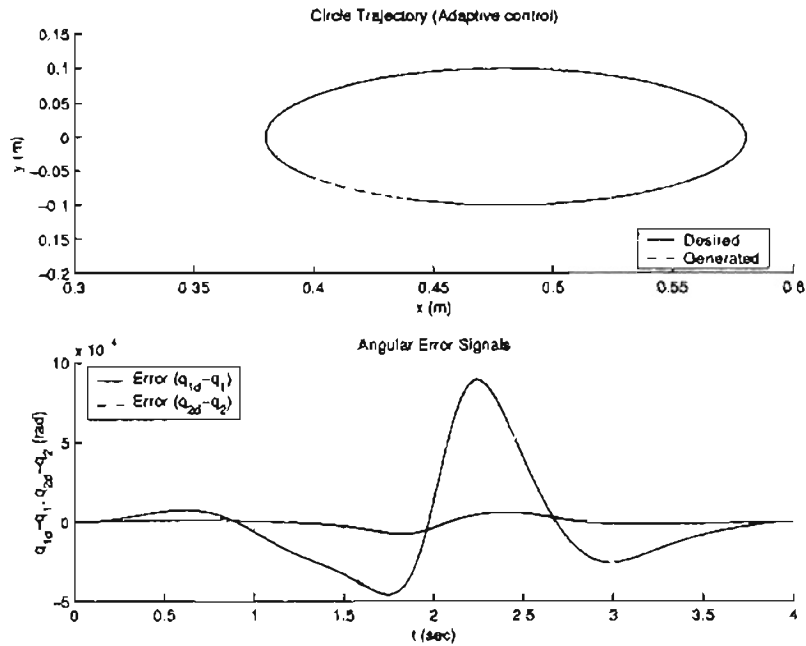


Figure 3.5: Adaptive control + static friction compensation simulation result: position and tracking error

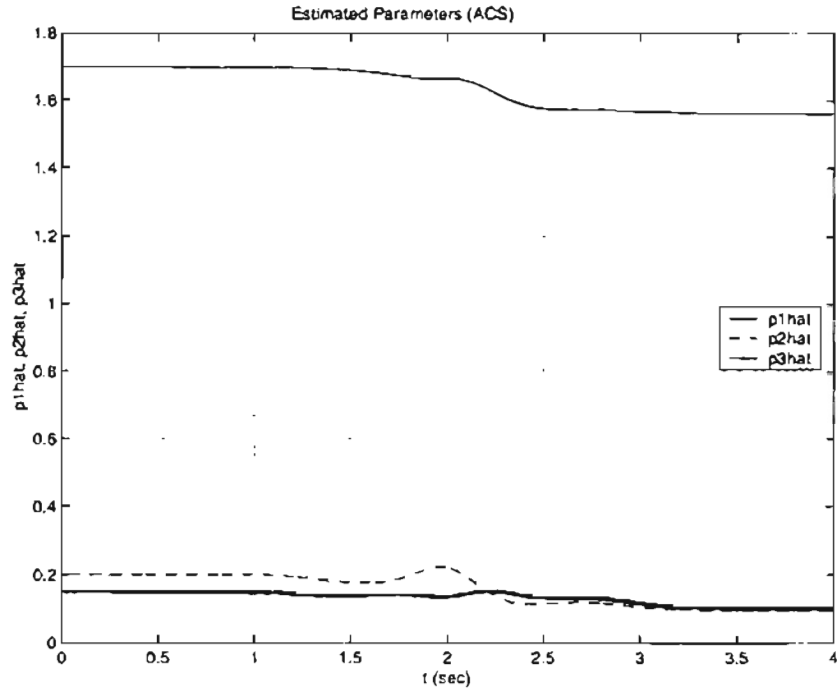


Figure 3.6: Adaptive control + static friction compensation simulation result: estimated parameters ( $\hat{p}_1$ ,  $\hat{p}_2$ ,  $\hat{p}_3$ )

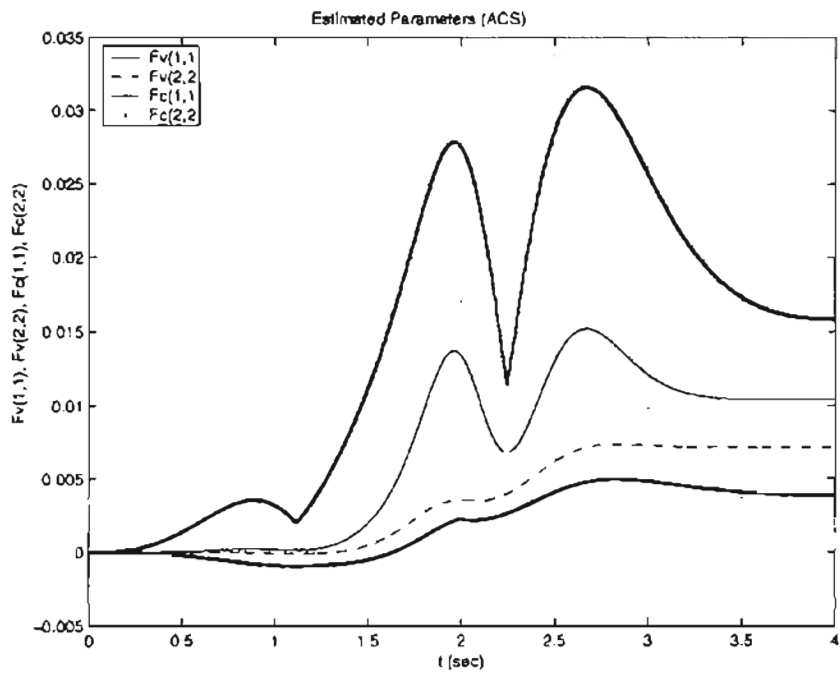


Figure 3.7: Adaptive control + static friction compensation simulation result: estimated parameters ( $\hat{F}_v(1, 1)$ ,  $\hat{F}_v(2, 2)$ ,  $\hat{F}_c(1, 1)$ ,  $\hat{F}_c(2, 2)$ )

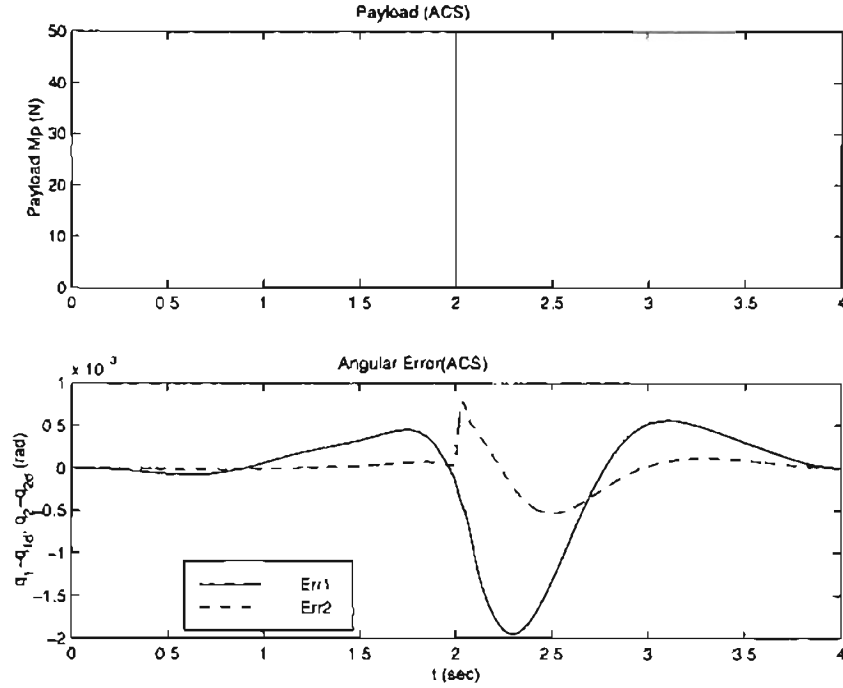


Figure 3.8: Adaptive control + static friction compensation simulation result: tracking performance in the presence of payload

### 3.5 Adaptive Control with Dynamic Friction Compensation

The classic static friction models cannot sufficiently describe all the dynamic effects of friction, such as pre-sliding, the friction lag and Stribeck effect. These phenomena all occur in the low velocity and pre-sliding region. To incorporate changing friction characteristics with velocity and for designing efficient friction compensation techniques, several friction compensations utilizing LuGre model are proposed ([11] - [16]). The basic concept is to improve the tracking performance, where the unmeasurable friction state  $z$  is estimated by an observer which is driven by the tracking error. Constant parameters in LuGre model are identified off-line. However the friction force varies with normal force, temperature and other factors. This variation can be considered as the effect of parameter uncertainties. A friction compensation algorithm with three unknown parameters  $\beta_1, \beta_2$  and  $\beta_3$  was proposed in [16]. This report will extend the research result and applied it to the multi-link robot

application. The robot dynamics and friction dynamics derived in Section 2.6 are

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + Q_3\dot{q} + Q_0z - Q_1\Psi z = \tau \quad (3.34)$$

$$\frac{dz}{dt} = \dot{q} - \Psi z \quad (3.35)$$

and

$$\Psi = \text{diag}(\gamma_1|\dot{q}_1|/g_1(\dot{q}_1), \dots, \gamma_n|\dot{q}_n|/g_n(\dot{q}_n))$$

$$g_i(\dot{q}_i) = F_{ci} + (F_{si} - F_{ci})e^{-(\dot{q}_i/\omega_{si})^2}$$

Consider the following control law

$$\tau = \widehat{M}(q)\ddot{q}_r + \widehat{C}(q, \dot{q})\dot{q}_r + \widehat{g}(q) - \Lambda_v e_v + \widehat{Q}_3\dot{q}_r = \widehat{Q}_0\widehat{z}_0 - \widehat{Q}_1\Psi\widehat{z}_1$$

where  $\widehat{z}_0$  and  $\widehat{z}_1$  are estimates of the internal state  $z$  in the LuGre model. Observers are necessary because  $z$  is unmeasurable. The observer dynamics is given by

$$\dot{\widehat{z}}_0 = \dot{q} - \Psi\widehat{z}_0 - e_v \quad (3.36)$$

$$\dot{\widehat{z}}_1 = \dot{q} - \Psi(\widehat{z}_1 - e_v) \quad (3.37)$$

Substituting the control law into the robot dynamics and simplifying results in the following closed-loop error dynamics:

$$M(q)\dot{e}_v + C(q, \dot{q})e_v + (\Lambda_v + Q_3)e_v = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\widetilde{\beta} + \widetilde{Q}_3\dot{q}_r + Q_0\widetilde{z}_0 + \widetilde{Q}_0\widehat{z}_0 - Q_1\Psi\widetilde{z}_1 - \widetilde{Q}_1\Psi\widehat{z}_1. \quad (3.38)$$

where  $\widetilde{Q}_i = \widehat{Q}_i - Q_i$  and  $\widetilde{z}_i = \widehat{z}_i - z_i$ . Since  $Q_i$  is a diagonal matrix, define a column vector  $\theta_i$  whose elements are the diagonal elements of  $Q_i$ . Also, define

$$Y_{0d}(\widehat{z}_0) = \text{diag}(\widehat{z}_{01}, \dots, \widehat{z}_{0n}),$$

$$Y_{1d}(\widehat{z}_1) = \text{diag}(\widehat{z}_{11}, \dots, \widehat{z}_{1n}),$$

$$Y_{3d}(\dot{q}_r) = \text{diag}(\dot{q}_{r1}, \dots, \dot{q}_{rn}),$$

where  $\tilde{z}_{0i}, \tilde{z}_{1i}, \dot{q}_{ri}, i = 1 : n$ , represent the elements of the  $n$ -vectors  $\tilde{z}_0, \tilde{z}_1$ , and  $\dot{q}_r$ , respectively. Therefore, the error dynamics, (3.38), can be written as

$$\begin{aligned} M(q)\dot{e}_v + C(q, \dot{q})e_v + (\Lambda_v + Q_3)e_v &= Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\tilde{\beta} \\ &+ Y_{3d}(\dot{q}_r)\tilde{\theta}_3 + Y_{0d}(\tilde{z}_0)\tilde{\theta}_0 - \Psi Y_{1d}(\tilde{z}_1)\tilde{\theta}_1 \\ &- Q_1\Psi\tilde{z}_1 + Q_0\tilde{z}_0. \end{aligned} \quad (3.39)$$

where  $\tilde{\theta}_i = \hat{\theta}_i - \theta_i$ . Subtracting each of (3.36) and (3.37) from (3.35), we obtain the observer error dynamics as

$$\dot{\tilde{z}}_0 = -\Psi\tilde{z}_0 - e_v, \quad (3.40)$$

$$\dot{\tilde{z}}_1 = -\Psi(\tilde{z}_1 - e_v). \quad (3.41)$$

Consider the following Lyapunov function candidate:

$$\begin{aligned} V &= \frac{1}{2}e_v^T M(q)e_v + \frac{1}{2}\tilde{\beta}^T \Gamma^{-1}\tilde{\beta} + \frac{1}{2}\tilde{z}_0^T Q_0\tilde{z}_0 + \frac{1}{2}\tilde{z}_1^T Q_1\tilde{z}_1 \\ &+ \frac{1}{2}\tilde{\theta}_0^T \Gamma_0^{-1}\tilde{\theta}_0 + \frac{1}{2}\tilde{\theta}_1^T \Gamma_1^{-1}\tilde{\theta}_1 + \frac{1}{2}\tilde{\theta}_3^T \Gamma_3^{-1}\tilde{\theta}_3 \end{aligned} \quad (3.42)$$

where  $\Gamma, \Gamma_0, \Gamma_1$  and  $\Gamma_3$  are diagonal positive gain matrices. The time derivative of the Lyapunov function candidate along the trajectories of the error dynamics is given by

$$\begin{aligned} \dot{V} &= e_v^T M(q)\dot{e}_v + \frac{1}{2}e_v^T \dot{M}(q)e_v + \tilde{\beta}^T \Gamma^{-1}\dot{\tilde{\beta}} + \dot{\tilde{z}}_0^T Q_0\tilde{z}_0 \\ &+ \dot{\tilde{z}}_1^T Q_1\tilde{z}_1 + \dot{\tilde{\theta}}_0^T \Gamma_0^{-1}\tilde{\theta}_0 + \dot{\tilde{\theta}}_1^T \Gamma_1^{-1}\tilde{\theta}_1 + \dot{\tilde{\theta}}_3^T \Gamma_3^{-1}\tilde{\theta}_3 \end{aligned} \quad (3.43)$$

Substituting the closed-loop error dynamics and the observer error dynamics,

$$\begin{aligned} \dot{V} &= -e_v^T (\Lambda_v + Q_3)e_v + (\Gamma^{-1}\dot{\tilde{\beta}} + e_v^T Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r))\tilde{\beta} \\ &- \dot{\tilde{z}}_0^T Q_0\Psi\tilde{z}_0 + \dot{\tilde{z}}_0^T Q_0e_v - e_v^T Q_0\tilde{z}_0 \\ &- \dot{\tilde{z}}_1^T Q_1\Psi\tilde{z}_1 - \dot{\tilde{z}}_1^T Q_1\Psi e_v + e_v^T Q_1\Psi\tilde{z}_1 \\ &+ (\Gamma_0^{-1}\dot{\tilde{\theta}}_0 + e_v^T Y_{0d}(\tilde{z}_0))\tilde{\theta}_0 \\ &+ (\Gamma_1^{-1}\dot{\tilde{\theta}}_1 - e_v^T Y_{1d}(\tilde{z}_1))\tilde{\theta}_1 \\ &+ (\Gamma_3^{-1}\dot{\tilde{\theta}}_3 + e_v^T Y_{3d}(\dot{q}_r))\tilde{\theta}_3 \end{aligned}$$

Since  $Q_0, Q_1$  and  $\Psi$  are diagonal,  $\tilde{z}_0^T Q_0 e_v - e_v^T Q_0 \tilde{z}_0 = 0$  and  $-\tilde{z}_1^T Q_1 \Psi e_v + e_v^T Q_1 \Psi \tilde{z}_1 = 0$ .

Further, if we let

$$\begin{aligned}\Gamma^{-1} \dot{\tilde{\beta}}^T + e_v^T Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r) &= 0 \\ \Gamma_0^{-1} \dot{\tilde{Q}}_0^T + e_v^T Y_{0d}(\hat{z}_0) &= 0 \\ \Gamma_1^{-1} \dot{\tilde{Q}}_1^T - e_v^T Y_{1d}(\hat{z}_1) &= 0 \\ \Gamma_3^{-1} \dot{\tilde{Q}}_3^T + e_v^T Y_{3d}(\dot{q}_r) &= 0\end{aligned}$$

and since  $\dot{\tilde{\beta}} = \dot{\hat{\beta}}$  and  $\dot{\tilde{\theta}}_i = \dot{\hat{\theta}}_i, i = 0, 1, 3$ , the update laws for the estimated manipulator and dynamic friction parameters are:

$$\dot{\hat{\beta}} = -\Gamma Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r) e_v \quad (3.44)$$

$$\dot{\hat{\theta}}_0 = -\Gamma_0 Y_{0d}^T(\hat{z}_0) e_v \quad (3.45)$$

$$\dot{\hat{\theta}}_1 = \Gamma_1 Y_{1d}^T(\hat{z}_1) e_v \quad (3.46)$$

$$\dot{\hat{\theta}}_3 = -\Gamma_3 Y_{3d}^T(\dot{q}_r)^T e_v \quad (3.47)$$

As a result, the time derivative of the Lyapunov function candidate simplifies to

$$\dot{V} = -e_v^T (\Lambda_v + Q_3) e_v - \tilde{z}_0^T Q_0 \Psi \tilde{z}_0 - \tilde{z}_1^T Q_1 \Psi \tilde{z}_1 \quad (3.48)$$

Thus,  $V$  is a Lyapunov function. Hence, all the internal signals are bounded. Since  $\Lambda_v, Q_0, Q_1, Q_3$  and  $\Psi$  are positive definite matrices, using the same arguments as before we conclude that  $e, e_v, \tilde{z}_0, \tilde{z}_1, \hat{\beta}, \hat{\theta}_0, \hat{\theta}_1, \hat{\theta}_3$  are bounded, and  $e_v, e, \tilde{z}_0, \tilde{z}_1 \rightarrow 0$  as  $t \rightarrow \infty$ .

### 3.5.1 Simulation Results

The simulation results of the dynamic controller with dynamic friction compensation are shown in Figure 3.9 and 3.10.

Figure 3.9 shows the controlled angular position, velocity and the tracking error. It can be seen that the tracking errors have peaks at the zero-velocity points. For joint 1, these points take place at time  $t = 1.2\text{sec}$  and  $t = 2.25\text{sec}$ . For joint 2, it takes place at  $t = 2\text{sec}$ .



Around the low velocity area ( $\dot{q} \approx 0$ ), the controlled system exhibits larger tracking errors than at high velocity. This is because the friction force has a quick jump with a directional reverse at zero-velocity crossing point, and exhibits the Stribeck effect. As a result, tracking error increases. At low velocity, the friction force has a lot of variation, whereas the friction force is almost constant when the velocity is high enough. When friction compensation is used, the controlled manipulator is forced to the desired position quickly. The estimated parameters  $\hat{Q}_0, \hat{Q}_1, \hat{Q}_3$  are shown in Figure 3.10. These parameters converge to constant values.

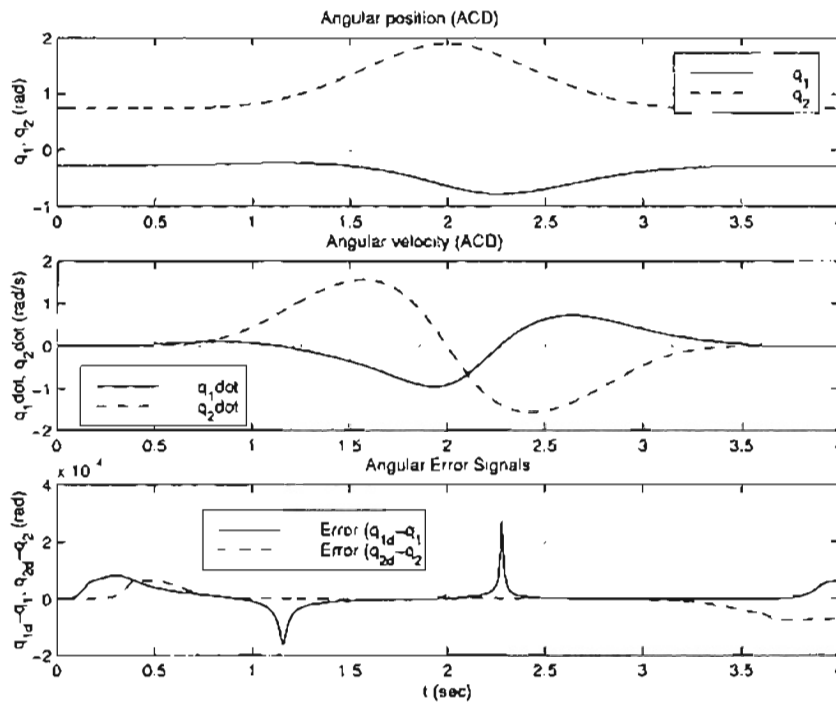


Figure 3.9: Simulation: position, velocity and tracking error

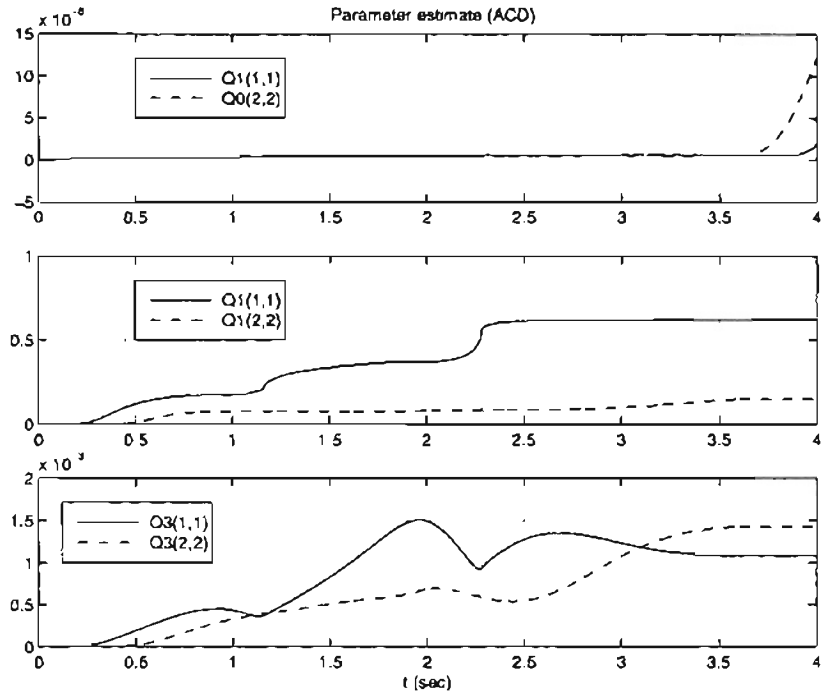


Figure 3.10: Simulation: estimated parameters

## CHAPTER 4

### Experimental Platform

This chapter gives a detailed description of the experimental platform, on which the designed controllers are tested. The open architecture feature of the experimental platform makes it easier for user to write various control algorithms and to introduce any additional specific hardware (such as Force/Torque sensor, gripper) to the manipulator.

The experimental platform consists of a two-axis direct drive manipulator as shown in Figure 4.1. The direct drive manipulator operates in the absence of the undesirable factors such as mechanical backlash and gear train compliance. It eliminates the need for gear reduction, so repeatability is limited only by the resolution of the position feedback. Each axis of the manipulator is driven by an NSK Megatorque direct drive servo-motor.

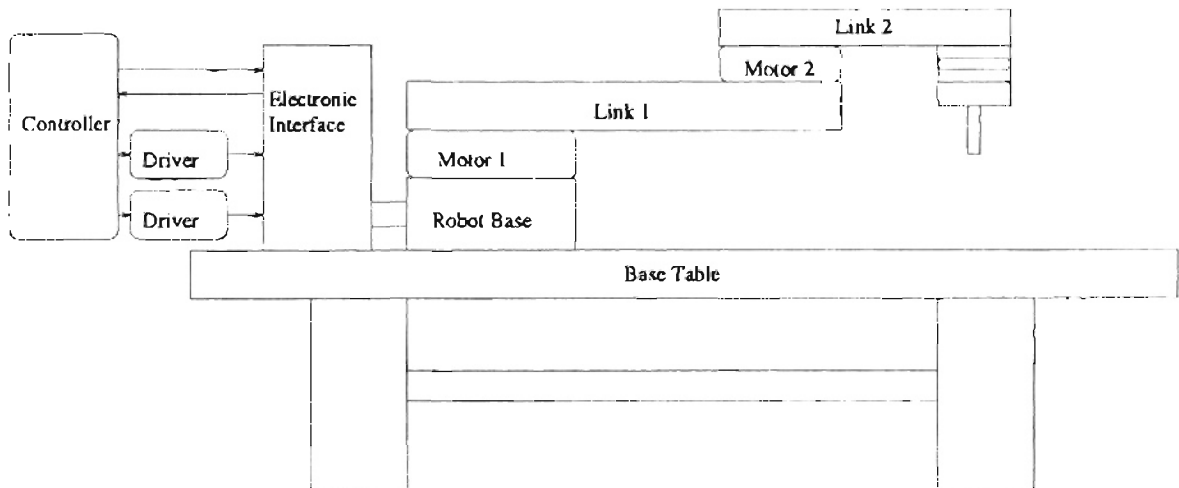


Figure 4.1: Sketch of the robot system

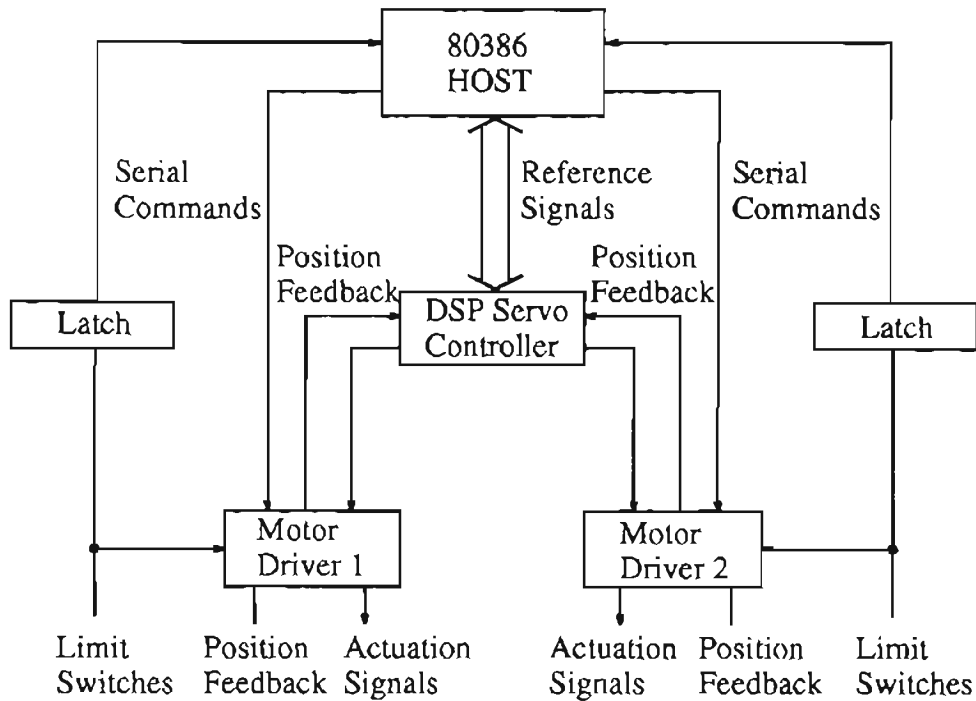


Figure 4.2: Hardware architecture

#### 4.1 Hardware

Figure 4.2 shows the block diagram of the hardware architecture of the control workstation. In standard operation, user interface, inverse kinematics and trajectory generation (interpolation) take place on the 80386 based host computer, while servo control is implemented on the Spectrum Signal Processing TMS320C30 DSP Processor Board. This scheme takes advantage of the hardware floating point capability of the 80386/80387 system and the computational power and efficient I/O handling of the TMS320C30 DSP. The motor drivers perform digital communication to and from the direct drive motors, and can optionally provide closed loop velocity control.

The NSK-Megatorque motor system consists of motor and its driver unit. This is a stand-alone system that contains all the elements needed for a complete closed-loop servo motor control. The NSK motor consists of a high torque direct drive brushless actuator, a high-resolution brushless resolver, and a heavy precision bearing. The servo motors

are capable of up to 3 revolutions per second maximum velocity and position feedback resolution of up to 156,400 counts per resolution. The base motor delivers up to 245 N-m of torque output, and the elbow motor produces up to 40 N-m torque output.

#### 4.1.1 Motor Drivers

The motor driver consists of the following components:

- **Power Amplifier(PA):** Amplifies the controller output to a level which is capable of driving the direct drive motors. AC power and the temperature of PA are monitored, and corresponding steps are taken to protect PA from damage.
- **Resolver Interface:** Position and velocity feedback signals are provided by resolver interface circuit.
- **Digital Signal Processor:** A 16 bit microprocessor system receives commands from outside world in either analog form or digital form. The command parameters can be position, velocity, or torque. The digital signal processor receives its feedback information from the resolver interface(position) and limit switch signals. Many techniques are applied in this part to improve the repeatability and eliminate mechanical resonances, including
  - Using a digital integrating function to improve the repeatability of the motors.
  - To avoid mechanical resonances, digital notch filters is employed to cut out certain frequencies.
  - A digital low-pass filter is employed to modify motor frequency response and make the motor smooth and quite.
  - An RS-232C serial port is provided for user to communicate with the digital signal processor directly.

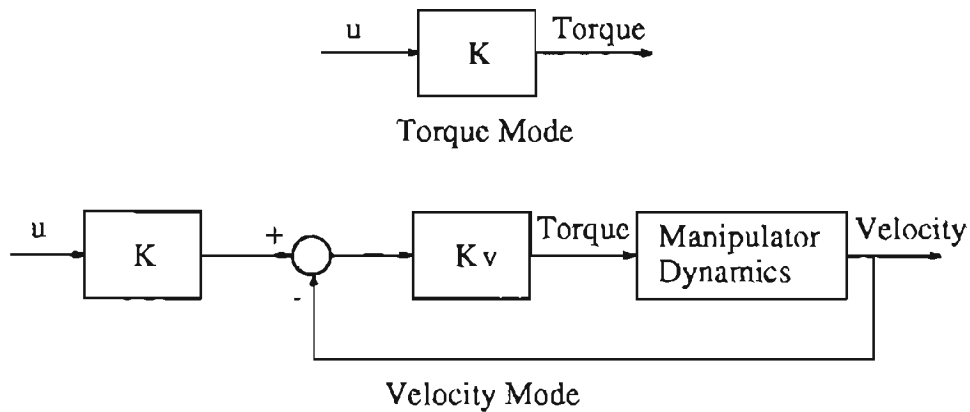


Figure 4.3: Torque mode vs. velocity mode

Two types of control modes (torque mode and velocity mode) selected by user are implemented in the motor driver unit. Referring to Figure 4.3, it can be seen that the driver simply behaves like a current amplifier, producing a motor torque which is proportional to the input signal. In velocity mode, the amplifier provides closed-loop velocity control. Closed-loop velocity control operates with a sample time of  $550\mu s$ .

#### 4.1.2 DSP Servo Controller

DSP Servo Controller is implemented by a Spectrum TMS320C30 DSP Processor Board. This board is installed on the bus of the host computer and interfaced to the motor drivers via a DS2 motion control interface board. The user servo algorithm is executed in DSP servo controller. The DSP/DS2 combination provides the TMS320C30 DSP, 128 KW RAM, 2 D/A converter, 2 A/D converter, 2 shaft encoder interface and 4-bit parallel I/O (refer to Figure 4.4).

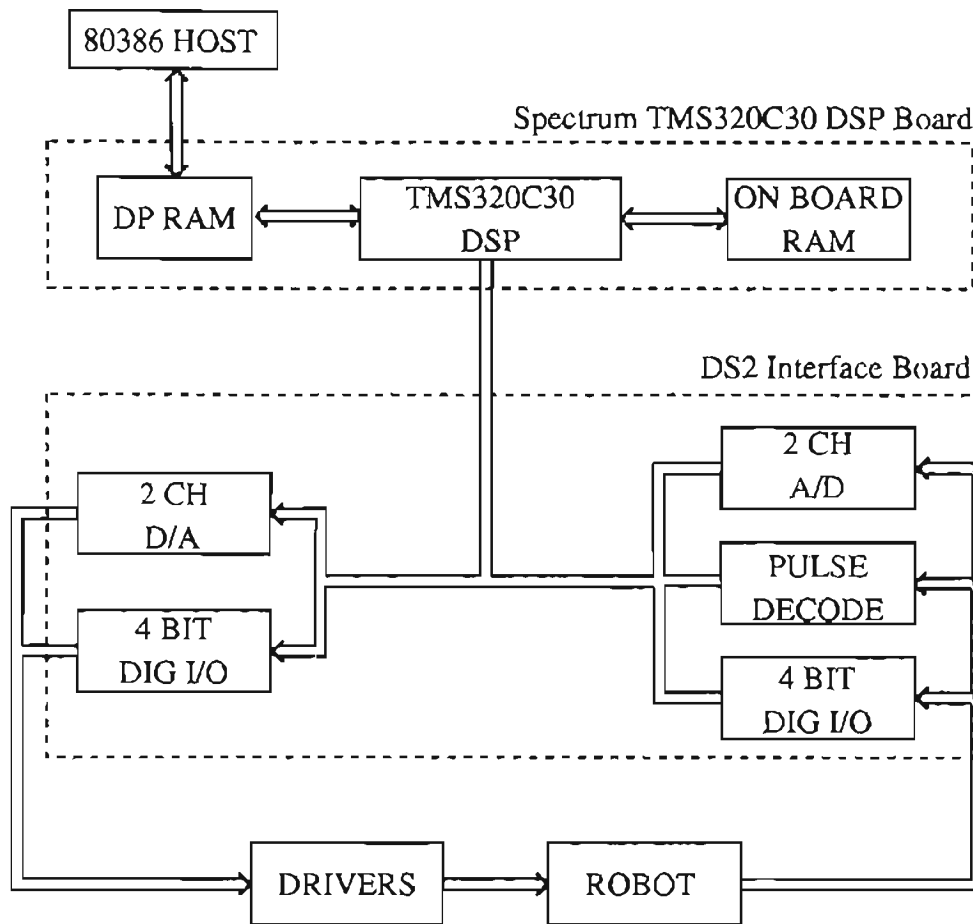


Figure 4.4: Connection diagram of the robot system

## 4.2 Software

### 4.2.1 RPL Software

An RPL(Robot Programming Language) program is a C-language program which is created by the user through an editor, then compiled by executing the Compile RPL File command. When the RPL program is executed, the execution of the controls software is temporarily suspended, and the RPL program is spawned as a separate process which runs on the host computer.

### 4.2.2 Servo Software

The servo software (control algorithm) is written in programming language C in the host computer, then downloaded to the DSP board after compilation by the TMS320C30C compiler. The user can monitor the desired signals when the control algorithm is running. The run time executive module handles the communication between the host computer and the servo DSP and delivers the output signals to the motor driver units via two DS2 I/O interface cards.

### 4.3 Manipulator Kinematics

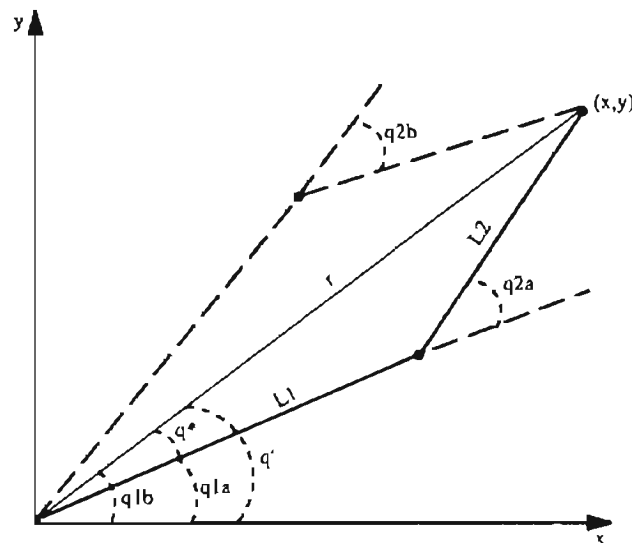


Figure 4.5: Diagram of the work space of the manipulator

Figure 4.5 shows the diagram of the work space of the manipulator. The transformation from joint space to work space is:

$$X = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) \\ L_1 \sin(q_1) + L_2 \sin(q_1 + q_2) \end{bmatrix} \quad (4.1)$$

where  $x$  and  $y$  are the work space coordinates of the end effector and  $q_1$  and  $q_2$  are the base angle and elbow angle, respectively.



Taking the derivative of (4.1) gives the work space velocity  $(\dot{x}, \dot{y})^T$  as a function of the joint space velocity  $(\dot{q}_1, \dot{q}_2)^T$ , which is

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J\dot{q} \quad (4.2)$$

where

$$J(q) = \begin{bmatrix} \frac{\partial h_1}{\partial q_1} & \frac{\partial h_1}{\partial q_2} \\ \frac{\partial h_2}{\partial q_1} & \frac{\partial h_2}{\partial q_2} \end{bmatrix} = \begin{bmatrix} -L_1 \sin(q_1) - L_2 \sin(q_1 + q_2) & -L_2 \sin(q_1 + q_2) \\ L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) & L_2 \cos(q_1 + q_2) \end{bmatrix} \quad (4.3)$$

is the Jacobian matrix of the manipulator. Since the determinant of  $J(q)$  is given by

$$\det(J(q)) = L_1 L_2 \sin(q_2) \quad (4.4)$$

It is obvious that the manipulator has singularity at any point where  $q_2 = 0$  or  $q_2 = \pi$ . Physically, if a constant, nonzero work space velocity is desired as the manipulator approaches singularity point, the joint space velocities required to achieve such a work space velocity are given by

$$\begin{aligned} \dot{q} &= J^{-1} \dot{X} \\ &= \frac{1}{\det(J)} \text{cof}^T(J) \dot{X} \\ &= \frac{\begin{bmatrix} L_2 \cos(q_1 + q_2) & L_2 \sin(q_1 + q_2) \\ -L_1 \cos(q_1) - L_2 \cos(q_1 + q_2) & -L_1 \sin(q_1) - L_2 \sin(q_1 + q_2) \end{bmatrix}}{L_1 L_2 \sin(q_2)} \dot{X} \end{aligned} \quad (4.5)$$

Since  $\det(J)$  approaches zero as the manipulator approaches the singularity point, the required joint space velocities become unbounded. As a result, during work space motions the end effector should always be kept away from the singularity regions where  $r = L_1 - L_2$  or  $r = L_1 + L_2$ .

Referring to Figure 4.5, the joint angles in terms of Cartesian coordinates are

$$q_2 = g(x, y) = \cos^{-1}\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1 L_2}\right) \quad (4.6)$$

and

$$q_1 = \tan^{-1}\left(\frac{y}{x}\right) - \sin^{-1}\left(\frac{L_2 \sin(q_2)}{r}\right). \quad (4.7)$$

Equation (4.6) and (4.7) give the inverse kinematics of the manipulator, i.e.,

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{y}{x}\right) \mp \sin^{-1}\left\{ \frac{L_2 \sin\left[\cos^{-1}\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1 L_2}\right)\right]}{(x^2 + y^2)^{1/2}} \right\} \\ \pm \cos^{-1}\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1 L_2}\right) \end{bmatrix} \quad (4.8)$$

Notice that (4.6) has two distinct solutions  $q_{2a}$  and  $q_{2b}$  where  $q_{2a} = -q_{2b}$  and an infinite number of solutions  $2n\pi q_{2a,b}$ . The transformation is not unique. Using equation (4.7) to express  $q_1$  as a function of  $q_2$  and  $q' = \tan^{-1}(y/x)$  reveals that  $q_{2a}$  and  $q_{2b}$  each yield a different value of  $q_1$ . This is represented graphically in Figure 4.5. The dashed and solid line represents elbow-up and elbow-down configuration of the arms, respectively.

#### 4.4 Manipulator Dynamics

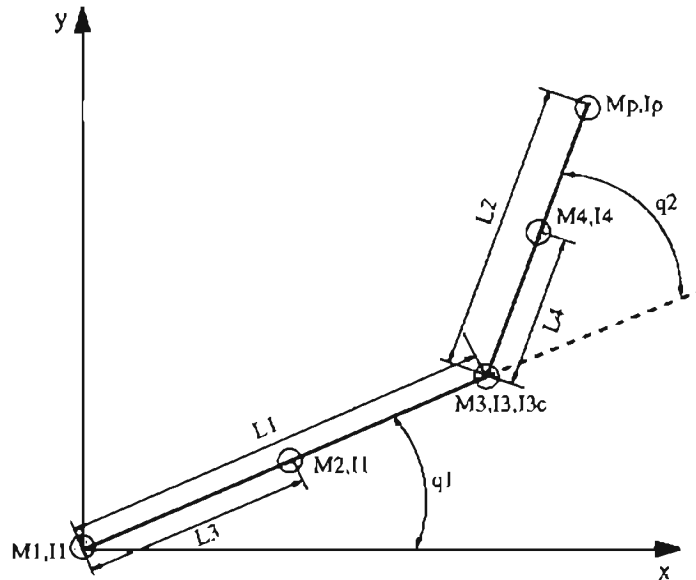


Figure 4.6: Diagram of the two-link planar NSK manipulator

Figure 4.6 shows the diagram of the NSK manipulator, which is a direct drive manipulator. Each axis of the manipulator is driven by an NSK-Megatorque direct drive servo motor.

$I_1$	Motor 1 rotor inertia
$M_1$	Motor 1 mass
$I_2$	Link 1 moment of inertia
$M_2$	Link 1 mass
$I_3$	Motor 2 rotor inertia
$M_3$	Motor 2 mass
$I_{3c}$	Motor 2 stator inertia
$M_4$	Link 2 mass
$I_4$	Link 2 moment of inertia
$M_p$	Payload mass
$I_p$	Payload moment of inertia
$q_1$	Angle of link 1 with respect to the horizontal
$q_2$	Angle of link 2 with respect to link 1
$l_1$	Length of link 1
$l_2$	Length of link 2
$l_3$	Distance of CG of link 1 from axis rotation
$l_4$	Distance of CG of link 2 from axis of rotation
$\tau_1$	Torque applied by motor 1
$\tau_2$	Torque applied by motor 2
$f_{c1}$	Axis 1 friction force
$f_{c2}$	Axis 2 friction force

The parameters defined in Figure 4.6 are shown in the above table. The equations of motion for the manipulator can be derived by using the Euler-Lagrange equations,

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = Q_i, \quad i = 1, 2 \quad (4.9)$$

where  $\mathcal{L} = \mathcal{K} - \mathcal{P}$ ,  $\mathcal{K}$  and  $\mathcal{P}$  are the kinetic and potential energy of the robot, respectively.  $q$  is a vector of generalized coordinates which completely describe the configuration of the manipulator and  $Q$  is a vector of generalized forces which are applied to the manipulator. In this case,

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad (4.10)$$

$$Q = \tau - f_f = \begin{bmatrix} \tau_1 - f_{f1} \\ \tau_2 - f_{f2} \end{bmatrix} \quad (4.11)$$

The dynamic equation for the two-link planar manipulator is:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = \tau(t) - f_f \quad (4.12)$$

where

$$C(q, \dot{q}) = \begin{bmatrix} -\dot{q}_2 p_3 \sin(q_2) & -(q_1 + q_2) p_3 \sin(q_2) \\ \dot{q}_1 p_3 \sin(q_2) & 0 \end{bmatrix} \quad (4.13)$$

is a vector of Coriolis terms and

$$M(q) = \begin{bmatrix} p_1 + 2p_3 \cos(q_2) & p_2 + p_3 \cos(q_2) \\ p_2 + p_3 \cos(q_2) & p_2 \end{bmatrix} \quad (4.14)$$

is the state varying inertia matrix.

Notice that there is no gravity term for NSK manipulator, due to the way in which it is constructed.

The left side of Equation (4.12) can be parameterized as

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = Y(q, \dot{q}, \ddot{q})\beta \quad (4.15)$$

where

$$Y(q, \dot{q}, \ddot{q}) = \begin{bmatrix} \ddot{q}_1 & \ddot{q}_2 & (2\ddot{q}_1 + \ddot{q}_2)\cos(q_2) - \ddot{q}_1^2 \sin(q_2) \\ 0 & \ddot{q}_1 + \ddot{q}_2 & (\sin(q_2) + \cos(q_2))\ddot{q}_1 \end{bmatrix} \quad (4.16)$$

is the regressor matrix, and

$$\beta \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \quad (4.17)$$

is the system parameter vector, where

$$p_1 = I_1 + I_2 + I_3 + I_{3c} + I_4 + I_p + (M_3 + M_4 + M_p)L_1^2 + M_2L_3^2 + M_4L_4^2 + M_pL_2^2 \quad (4.18)$$

$$p_2 = I_3 + I_4 + I_p + M_4L_4^2 + M_pL_2^2 \quad (4.19)$$

$$p_3 = M_4L_1L_4 + M_pL_1L_2 \quad (4.20)$$

## CHAPTER 5

### Experiments

#### 5.1 Desired Trajectory Generation

In this section, a desired trajectory in Cartesian space is generated for simulation and experiment. Suppose the desired trajectory is a circle with a radius of  $r$  and center at  $(x_c, y_c)$ , as shown in Figure 5.1. The desired trajectory should have smooth joint space trajectories such that  $\dot{q}_d(t) \in \mathcal{L}_\infty$  and  $\ddot{q}_d(t) \in \mathcal{C}^3$ , i.e.,  $\dot{q}_d(t)$ ,  $\ddot{q}_d(t)$  and  $\dddot{q}_d(t) \in \mathcal{L}_\infty$ . In

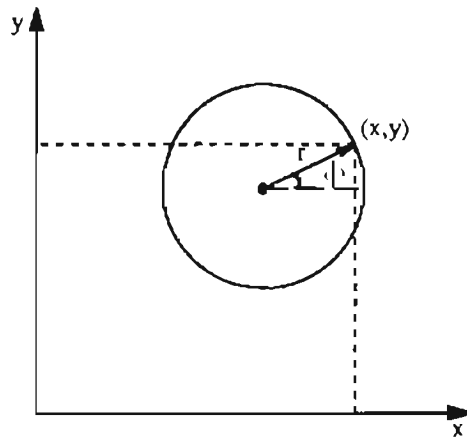


Figure 5.1: Circle trajectory

Cartesian space, the coordinate of the end-effector:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} x_c + r \cos(\phi(t)) \\ y_c + r \sin(\phi(t)) \end{bmatrix} \quad (5.1)$$

Let

$$\phi(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6 + a_7 t^7 \quad (5.2)$$

with the initial condition

$$\phi(0) = 0, \dot{\phi}(0) = 0, \ddot{\phi}(0) = 0, \dddot{\phi}(0) = 0 \quad (5.3)$$

and the end condition

$$\phi(t_f) = 2\pi, \dot{\phi}(t_f) = 0, \ddot{\phi}(t_f) = 0, \dddot{\phi}(t_f) = 0 \quad (5.4)$$

where  $t_f$  is the instant when the circle trajectory ends. From (5.2), we can get

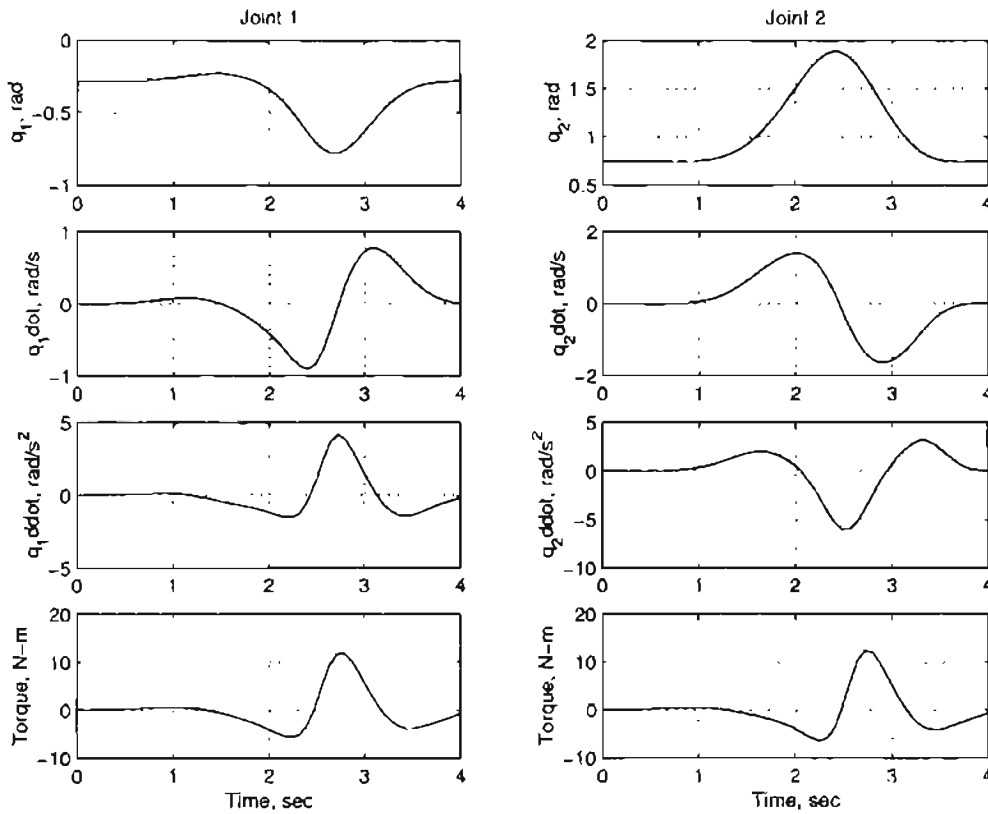


Figure 5.2: Diagram of  $q_d, \dot{q}_d, \ddot{q}_d$  and  $\tau$  for a circle trajectory

$$\begin{bmatrix} \phi(t) \\ \dot{\phi}(t) \\ \ddot{\phi}(t) \\ \dddot{\phi}(t) \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 & t^5 & t^6 & t^7 \\ 0 & 1 & 2t & 3t^2 & 4t^3 & 5t^4 & 6t^5 & 7t^6 \\ 0 & 0 & 2 & 6t & 12t^2 & 20t^3 & 30t^4 & 42t^5 \\ 0 & 0 & 0 & 6 & 24t & 60t^2 & 120t^3 & 210t^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} \quad (5.5)$$

Substituting the initial and end conditions into (5.5) and solving the matrix equation results in

$$a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 & t_f^6 & t_f^7 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 & 6t_f^5 & 7t_f^6 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 & 30t_f^4 & 42t_f^5 \\ 0 & 0 & 0 & 6 & 24t_f & 60t_f^2 & 120t_f^3 & 210t_f^4 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2\pi \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (5.6)$$

Figure 5.2 shows the open-loop response of joint angle, velocity and acceleration, as well as computed torque when a circle trajectory is generated. The circle is selected with a radius of  $0.1m$ , center at  $(0.48, 0)m$ . It can be seen that the joint angle, velocity, acceleration and torque are smooth. Motor torques are within the range of the maximum allowed ( $\tau_{1max}=240N-m$ ,  $\tau_{2max}=40N-m$ ).

## 5.2 Experimental Condition

All the five controllers (PD, AC, CT, ACS and ACD) designed in Section 3 were implemented on the experimental platform described in Section 4. The tracking performance of



the five controllers are compared. Two desired trajectories are chosen: (1) circle trajectory introduced in Section 5.1, (2) sinusoidal trajectory of desired joint position for both links of the manipulator. The amplitude of the sinusoidal trajectory is 0.8 radians and frequency is 0.2 Hz. The sinusoidal trajectory tests the tracking performance of each controller for low velocity ( $<1$  rad/s) and velocity reversal. The control sampling period is chosen to be 2 milli-seconds. The payload on the end-effector is constant. Since the frequency of the sinusoid is 0.2 Hz, the period of each cycle is 5 seconds. Six cycles of data was collected for each experiment.

The following gain matrix are chosen in the experiments:  $\Lambda_p = \text{diag}(15, 15)$ ,  $\Lambda_v = \text{diag}(100, 25)$ ,  $\Gamma = \text{diag}(1, 0.2, 0.2)$ ,  $\Gamma_{fs} = \text{diag}(1, 1, 1, 1)$ ,  $\Gamma_0 = \text{diag}(100, 25)$ ,  $\Gamma_1 = \text{diag}(100, 25)$ ,  $\Gamma_3 = \text{diag}(100, 25)$ . The initial values of estimated parameters are set as  $\hat{\beta}(0) = [1.7, 0.5, 0.5]^T$ ,  $\hat{\beta}_{fs}(0) = [0, 0, 0, 0]^T$ ,  $\hat{Q}_i(0) = \text{diag}[0, 0], i = 0, 1, 3$ .

To consistently compare the performance of different controllers, the common gain parameters in different controllers are kept the same for all experiments. The robot is running under PD control in the normal operation. A particular controller (CT, AC, ACS, or ACD) can be chosen by pressing the assigned key on the keyboard.

### 5.3 Motor Characteristics

The motors consist of a high brushless actuator, a high resolution brushless resolver, and a heavy duty precise NSK bearing. The high torque actuator eliminates the need for gear reduction, while the built-in resolver usually makes feedback components, such as encoders or tachometers unnecessary. Finally the heavy duty bearing eliminates the need for separate mechanical support since the motor casing can very often support the load directly in most applications. Figure 5.3 gives the free body diagram of a *DC* motor. If the rotor and shaft are assumed to be rigid, the motor torque,  $T$ , is related to the armature current,  $i$ , by a constant factor  $K_t$ . The back emf,  $e$ , is related to the rotational velocity by

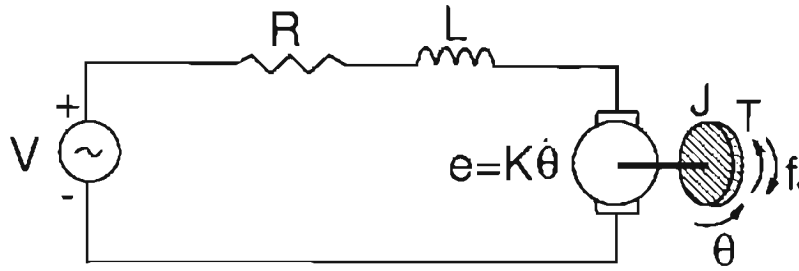


Figure 5.3: Free body diagram of DC motor

the following equations:

$$T = K_t i, \quad (5.7)$$

$$e = K_e \dot{\theta}. \quad (5.8)$$

From the Figure 5.3 we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J\ddot{\theta} + f_f \dot{\theta} = T \quad (5.9)$$

$$L \frac{di}{dt} + Ri = V - K_e \dot{\theta} \quad (5.10)$$

where

$J$ : moment of inertia of the rotor

$K_t$ : motor constant

$K_e$ : motor constant

$R$ : electric resistance

$L$ : electric inductance

$V$ : source voltage

$\theta$ : position of shaft

$T$ : motor torque

$f_f$ : friction torque

Applying Laplace Transform, the above equations can be expressed in terms of  $s$  and eliminating  $I(s)$  we can get the following transfer function, where the rotating position is

the output and the voltage is the input.

$$\theta(s) = \frac{K_t}{Js^2(Ls + R) + K_tK_e s} V(s) - \frac{Ls + R}{Js^2(Ls + R) + K_tK_e s} f_f(s) \quad (5.11)$$

From Equation (5.11), it can be seen that the motor dynamics depends on friction. If only viscous friction is considered, i.e.,  $f_f = F_v \dot{\theta}$ , Equation (5.11) can be rewritten as

$$\frac{\theta(s)}{V(s)} = \frac{K_t}{(Js + F_c)(Ls + R)s + K_tK_e s} \quad (5.12)$$

Considering different friction models results in different motor dynamics.

#### 5.4 Parameter Identification of the LuGre Friction Model

For the two link manipulator, the friction parameters of each link are identified separately. Recall that the LuGre friction model for each link is given by

$$\frac{dz}{dt} = \dot{q} - \beta_0 \frac{|\dot{q}|}{g(\dot{q})} z, \quad (5.13)$$

$$g(\dot{q}) = F_c + (F_s - F_c) e^{-(\dot{q}/\omega_s)^2}, \quad (5.14)$$

$$f_f = \beta_0 z + \beta_1 \dot{z} + \beta_2 \dot{q}, \quad (5.15)$$

and the motor dynamics is

$$J\ddot{q} = T - f_f. \quad (5.16)$$

We can identify friction parameters in Equation (5.13) and Equation (5.14) by following steps.

- Identify  $F_c$  and  $\beta_2$

When  $\dot{q} \gg \omega_s$ , we can consider steady-state friction only, i.e.,

$$f_f \approx F_c \text{sgn}(\dot{q}) + \beta_2 \dot{q}. \quad (5.17)$$

Hence, the motor model becomes

$$J\ddot{q} = T - F_c \text{sgn}(\dot{q}) - \beta_2 \dot{q}. \quad (5.18)$$

Rewriting Equation (5.18) yields

$$y = \varphi^T \theta, \quad (5.19)$$

where

$$y = T - J\ddot{q}, \theta = \begin{bmatrix} \beta_2 \\ F_c \end{bmatrix} \text{ and } \varphi = \begin{bmatrix} \dot{q} \\ \text{sgn}(\dot{q}) \end{bmatrix}. \quad (5.20)$$

Using the least square estimation, we get

$$\theta = (\phi\phi^T)^{-1}\phi Y, \quad (5.21)$$

where

$$\phi = \begin{bmatrix} \varphi(1), \dots, \varphi(n) \end{bmatrix} \text{ and } Y = \begin{bmatrix} y(1), \dots, y(n) \end{bmatrix}^T. \quad (5.22)$$

where  $n$  is the number of data points.

Figure 5.4 and 5.5 show the experimental results for link 1 and link 2, respectively. A sinusoidal torque is applied to each link. Only the data with the angular velocity  $|\dot{q}| > \omega_s$  are taken into the estimation of  $F_c$  and  $\beta_2$ . The experiments indicate that  $F_c$  for link 1 and link 2 are 1.857  $N\cdot m$  and 0.7805  $N\cdot m$ , respectively, and  $\beta_2$  are 1.9291  $N\cdot m\cdot s/rad$  and 0.38  $N\cdot m\cdot s/rad$ , respectively.

- Identify  $F_s$

$F_s$  can be identified by the break away experiment, i.e., the torque is increased from a small value and registered when the link just starts to move.  $F_s$  for link 1 and link 2 are 11  $N\cdot m$  and 1.6  $N\cdot m$ , respectively.

- Identify Stribeck velocity,  $\omega_s$

When the link moves at sufficiently small velocity, the friction is given by

$$f_f = \left[ F_c + (F_s - F_c e^{(\frac{\dot{q}}{\omega_s})^2}) \right]. \quad (5.23)$$

Equation (5.23) substituted into the motor dynamics results in

$$J\ddot{q} = T - \left[ F_c + (F_s - F_c e^{(\frac{\dot{q}}{\omega_s})^2}) \right]. \quad (5.24)$$

The linear parameterization on the unknown parameter  $\omega_s$  is

$$y = \varphi^T \theta, \quad (5.25)$$

where,

$$y = \dot{q}^2, \quad \theta = \omega_s^2, \quad \varphi = \ln \gamma \quad (5.26)$$

and

$$\gamma = \frac{F_s - F_c}{(T - J\ddot{q})\text{sgn}(\dot{q}) - F_c}. \quad (5.27)$$

The real solution for  $\omega_s$  only exists for  $\gamma \geq 1$ . The experimental results for link 1 and link 2 are shown in Figure 5.6 and 5.7, respectively. The Stribeck velocities  $\omega_s$  for link 1 and link 2 are  $0.14 \text{ rad/s}$  and  $0.096 \text{ rad/s}$ , respectively.

- Identify  $\beta_0$

When the motion is in the pre-sliding region where elastic effects dominate over the plastic effects, the friction is given by

$$f_f = \beta_0 q. \quad (5.28)$$

Hence

$$y = \varphi^T \theta, \quad (5.29)$$

where

$$y = T, \quad \theta = \beta_0 \text{ and } \varphi = q. \quad (5.30)$$

Figure 5.8 and 5.9 show the experiments for both links. The rest stiffness  $\beta_0$  for link 1 and link 2 are  $3410 \text{ N.m/rad}$  and  $838 \text{ N.m/rad}$ , respectively.

In the above identification procedures, the accuracy of the identified parameters depends on the parameters which are assumed known. The inertia  $J$  and the constant  $v_2 t$  between the applied voltage and generated torque by the motor are the major factors that

affect the identification results. Because the torque  $T$  generated by motor cannot be measured directly in the experimental platform, it is computed by the applied voltage  $V$  from the  $D/A$  board using the relation  $T = v2c * V$ . Both  $J$  and  $v2c$  are selected as nominal values in the experiments. However the actual  $J$  and  $v2c$  may be different from the nominal values that are chosen, which will result in identification error. The joint position  $q$ , velocity  $\dot{q}$  and acceleration  $\ddot{q}$  are assumed accurate. This is reasonable because of the high resolution of the resolver which measures the joint position;  $\dot{q}$  and  $\ddot{q}$  are computed from  $q$  using finite difference approximation. As a result,  $\dot{q}$  and  $\ddot{q}$  may be noisy. This disadvantage can be overcome by applying a low pass filter to filter the high frequency noise. The resolution of the resolver affects the accuracy of  $\beta_0$  as the experiment for identifying it requires measuring very small joint position  $q$ . More accurate parameter identification methods need further investigations with a very high resolution resolver. As the adaptive control with dynamic friction compensation use some of the identified friction parameters, the effect of inaccurate friction parameters on the closed-loop system is also a problem which requires future study.

## 5.5 Experimental Results

### 5.5.1 PD Control

The experimental results for PD control are shown in Figure 5.10. The experiment is conducted without payload disturbance and with a fixed proportional and derivative gain. The first column of Figure 5.10 shows the joint position, joint velocity and joint position error for link 1, and the second column shows results for link 2. From the figure it can be observed that tracking errors are small without disturbance by using only PD controller. Observe that the maximum tracking errors occurs at the points where the joint velocity reverses direction, which is due to low velocity friction.

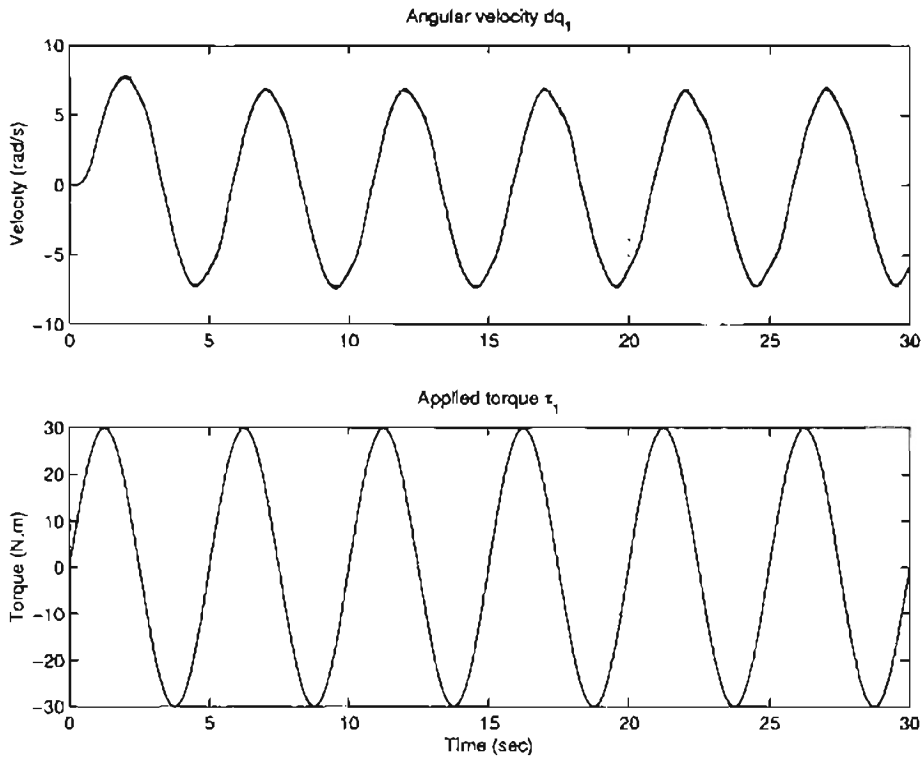


Figure 5.4: Identification of  $\beta_2$  and  $F_c$  for link 1

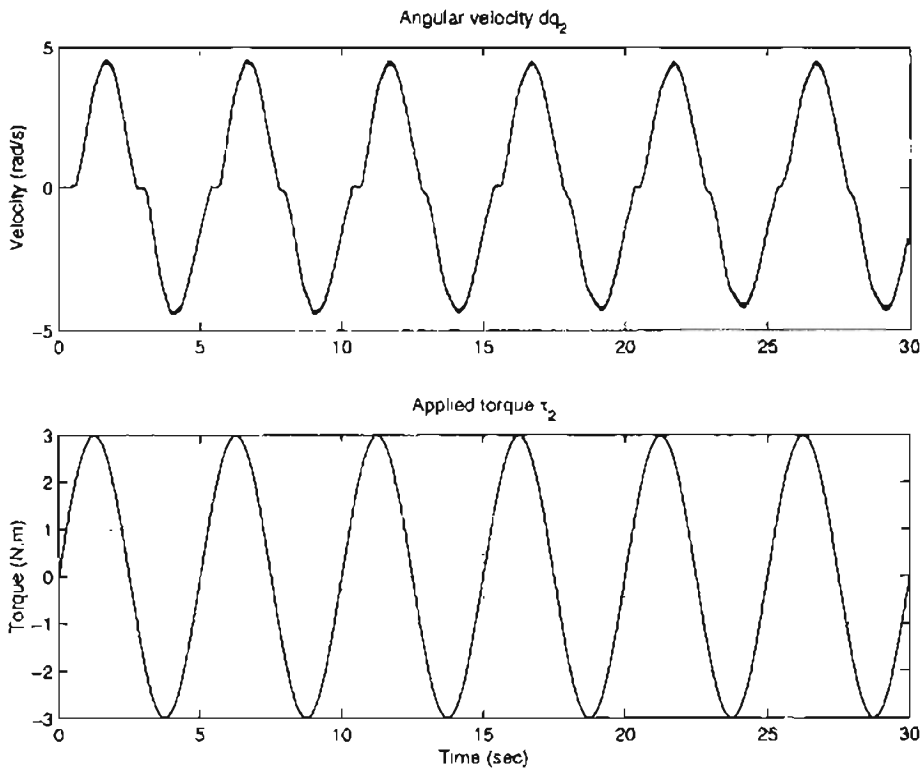


Figure 5.5: Identification of  $\beta_2$  and  $F_c$  for link 2

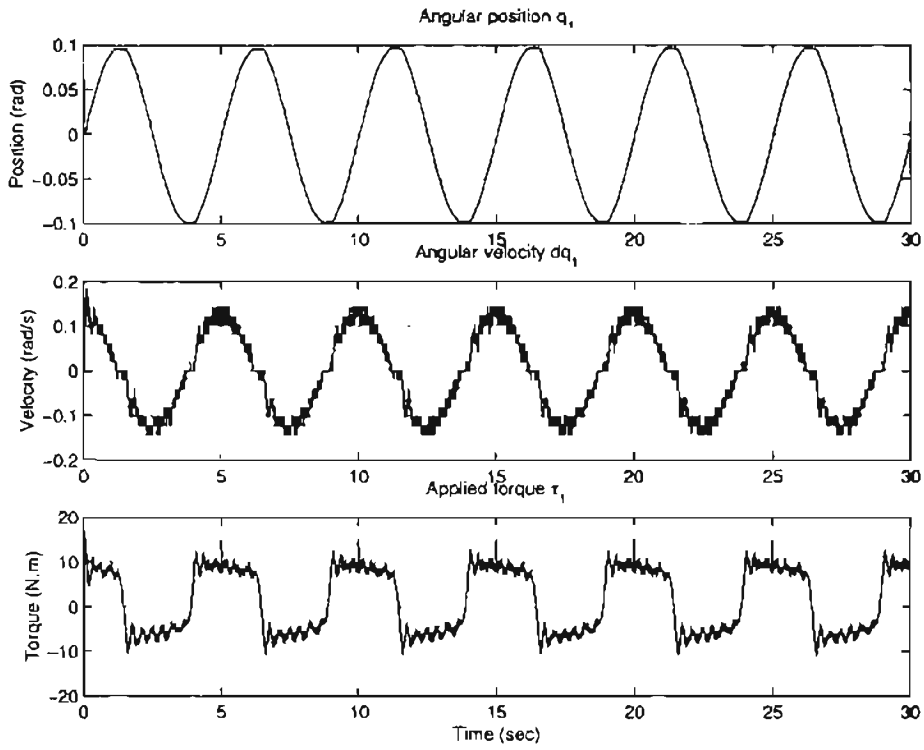


Figure 5.6: Identification of  $\omega_s$  for link 1

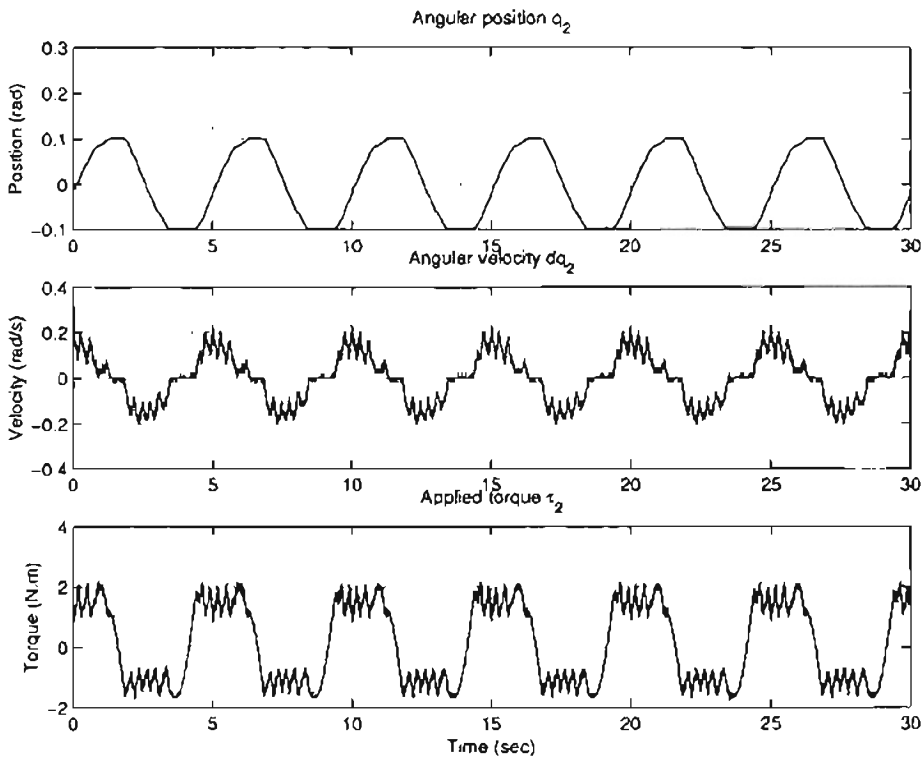


Figure 5.7: Identification of  $\omega_s$  for link 2



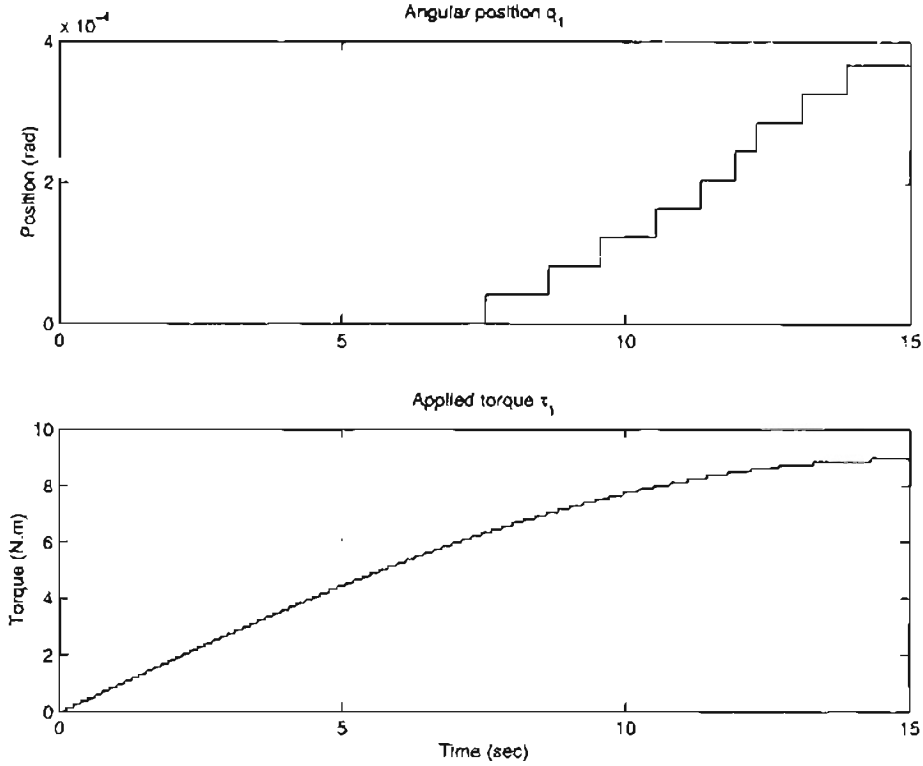


Figure 5.8: Identification of  $\beta_0$  for link 1

### 5.5.2 Model Based Adaptive Control

The experimental results of the model-based adaptive controller are shown in Figures 5.11 and 5.12. From the Figure 5.11, it can be seen that the tracking error is smaller than that of PD control, especially for link 2. The tracking error of link 2 in adaptive control is much smaller than in PD control. Figure 5.12 shows the estimated parameters  $\hat{p}_1$ ,  $\hat{p}_2$  and  $\hat{p}_3$ .

In the experiment, the initial values of parameter estimate  $\hat{\beta}$  are taken to be half of their nominal value. The adaptive controller therefore starts as a PD controller with a feedforward part. The feedforward part plays an important role as the parameter adaptation is driven by the tracking error. Increasing the gain matrix  $\Lambda_v$  will speed the tracking process;  $\Lambda_v$  in the adaptive controller takes the same role as gain  $K_p$  does in PD controller.

However, although the tracking error is guaranteed to converge to zero, the adaptation law does not guarantee the parameters converging to their true values. In the absence

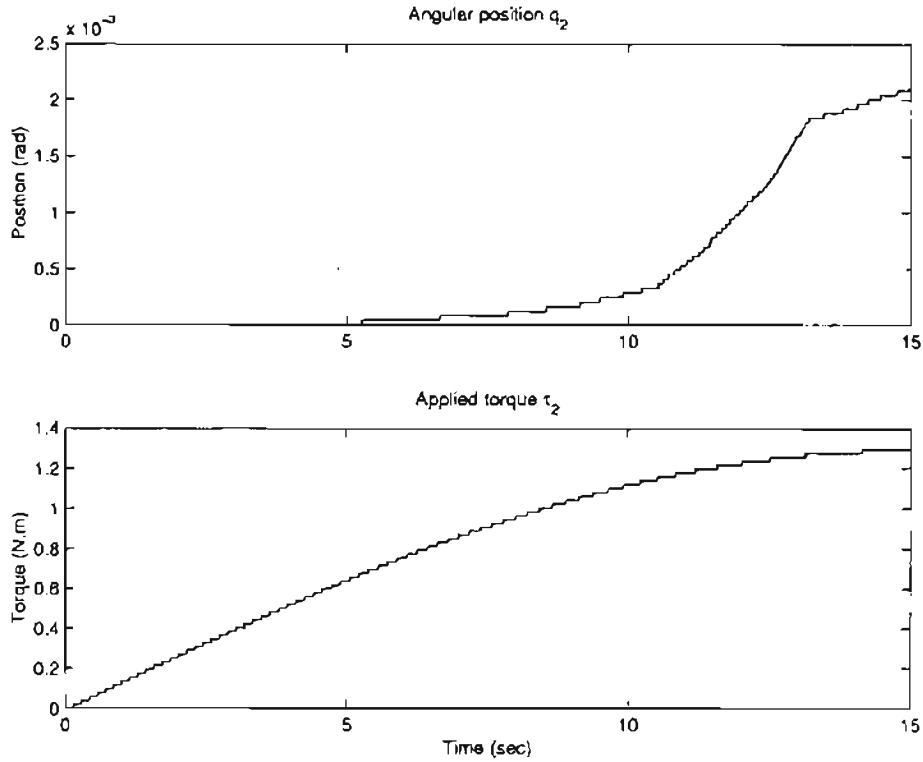


Figure 5.9: Identification of  $\beta_0$  for link 2

of friction, the condition for the estimated parameters asymptotically converging to the true values is that the matrix  $Y(q_d, \dot{q}_d, \ddot{q}_d)$  is persistently exciting (PE) and uniformly continuous, i.e., there must exist positive constants  $\delta$ ,  $\alpha_1$  and  $\alpha_2$  such that for all  $t_1 \geq 0$

$$\alpha_1 I \leq \int_{t_1}^{t_1 + \delta} Y^T(q_d, \dot{q}_d, \ddot{q}_d) Y(q_d, \dot{q}_d, \ddot{q}_d) dt \leq \alpha_2 I \quad (5.31)$$

This adaptive controller does not need the measurement of angular acceleration and the use of the reference model. This advantage makes it easy to be applied to an industrial robot without much computation. The parameter adaptation process should be stopped if the estimated parameters reach a priori known bounds.

### 5.5.3 Computed Torque Control

Figure 5.13 gives the experimental result for computed torque control. Comparing Figures 5.13 and 5.11, it can be seen that the computed torque control and the adaptive

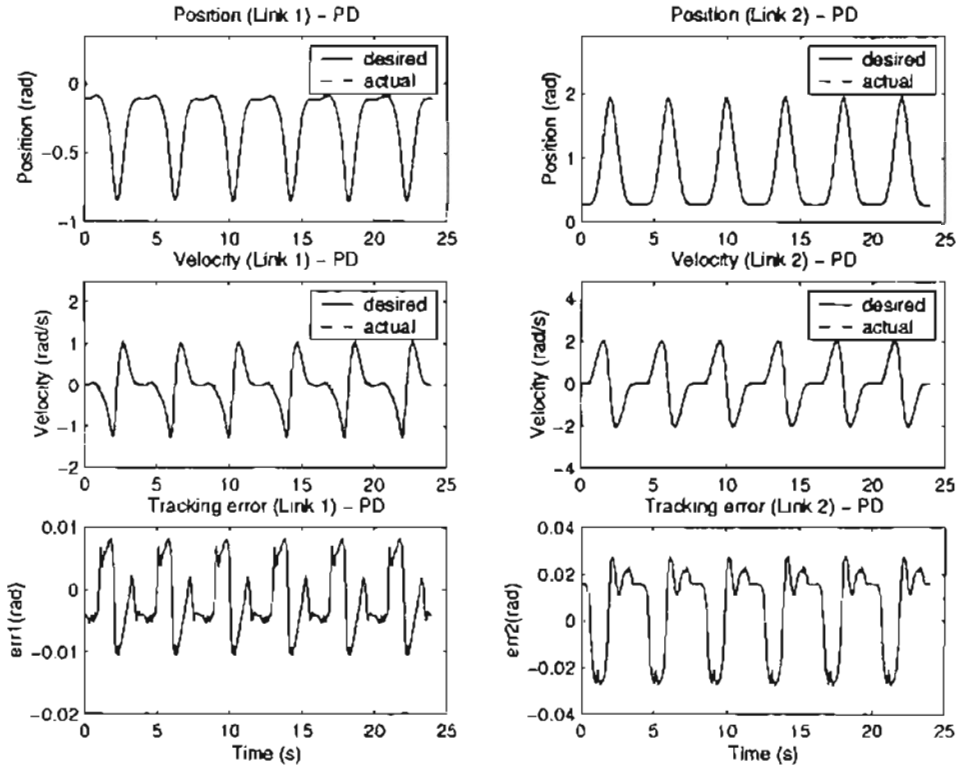


Figure 5.10: PD control experimental result (circle trajectory)

control achieve almost the same level of tracking error. Because the computed torque acts as a feedforward part, small PD gain can be achieved compared with the PD control. High PD gain may generate oscillation which should be avoided.

#### 5.5.4 Model Based Adaptive Control with Static Friction Compensation

Tracking performance of adaptive control with static friction compensation is shown in Figure 5.14 and Figure 5.15. Figure 5.14 shows that the tracking errors decrease with time. At the beginning of the first trajectory cycle, the tracking error is maximum at the zero velocity crossing points. As the friction compensation takes effect, i.e., estimates of the friction parameter matrix  $F_v$  and  $F_c$  take non-zero values, these peaks decrease and even disappear in link 2. The estimated parameters  $\hat{F}_v(1, 1)$ ,  $\hat{F}_v(2, 2)$ ,  $\hat{F}_c(1, 1)$  and  $\hat{F}_c(2, 2)$  are shown in Figure 5.15.

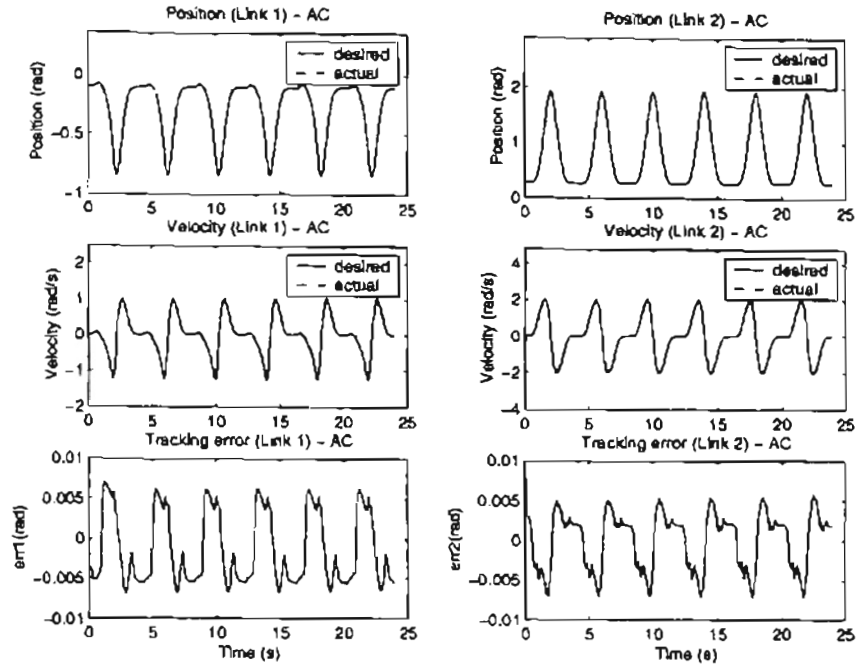


Figure 5.11: Adaptive control experimental result: velocity and tracking error (circle trajectory)

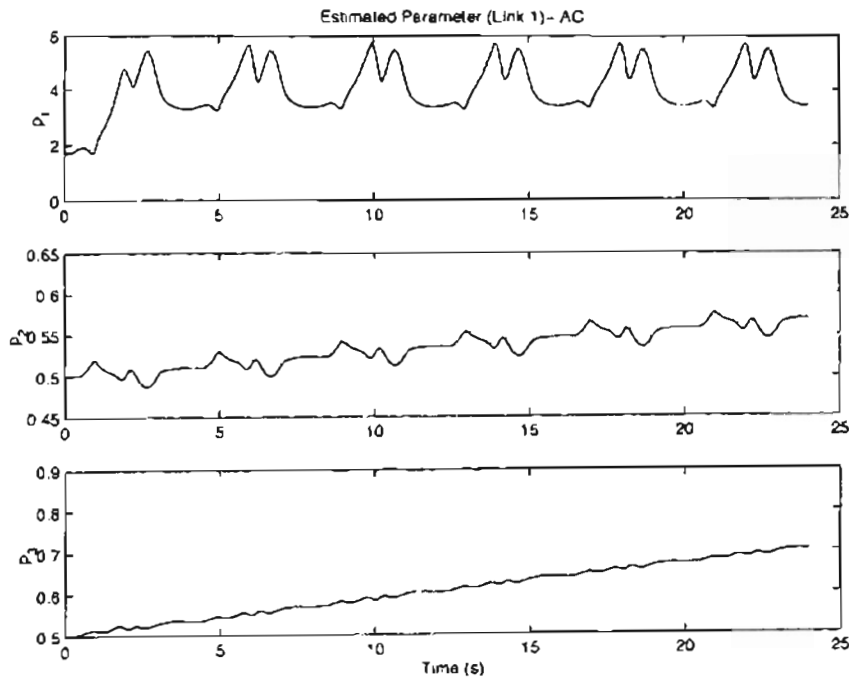


Figure 5.12: Adaptive control experimental result: estimated parameters ( $\hat{p}_1, \hat{p}_2, \hat{p}_3$ ) (circle trajectory)

www.ijerap.in

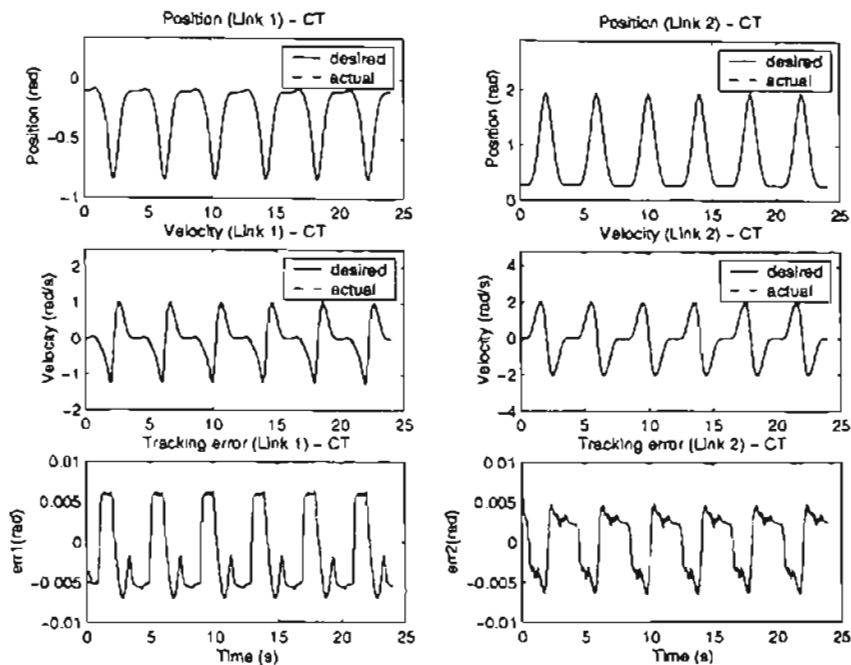


Figure 5.13: Computed torque control experimental result: velocity and tracking error (circle trajectory)

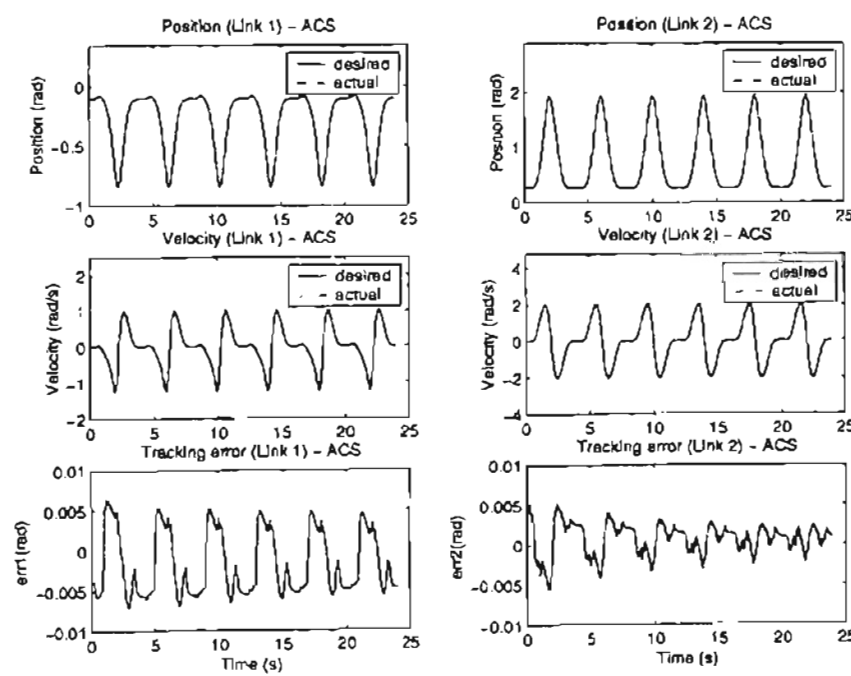


Figure 5.14: Adaptive control + static friction compensation experimental result: velocity and tracking error (circle trajectory)

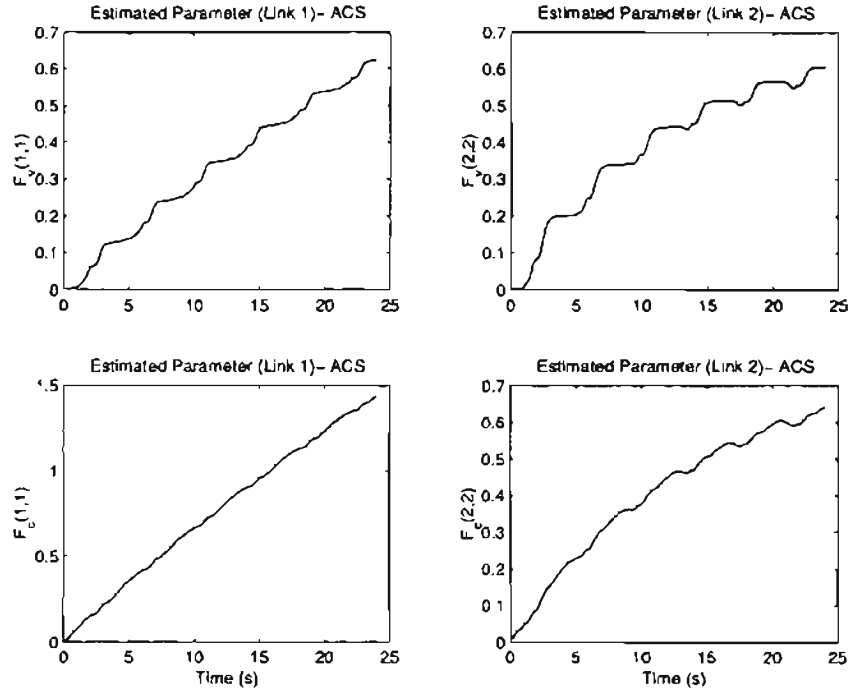


Figure 5.15: Adaptive control + static friction compensation experimental result: estimated parameters ( $\hat{F}_v(1, 1)$ ,  $\hat{F}_v(2, 2)$ ,  $\hat{F}_c(1, 1)$ ,  $\hat{F}_c(2, 2)$ ) (circle trajectory)

### 5.5.5 Adaptive Control with Dynamic Friction Compensation

The experimental results for ACD are shown in Figure 5.16 to 5.20. The tracking error shown in Figure 5.16. Figure 5.17 shows the estimated parameters  $\hat{p}_1$ ,  $\hat{p}_2$  and  $\hat{p}_3$ . Figure 5.18 and 5.19 show the estimated friction coefficients in LuGre friction model for link 1 and link 2, respectively. The estimated internal state  $z$  is shown in Figure 5.20. The same tracking performance pattern with ACS is observed from Figure 5.16, but the tracking errors decrease much faster than in ACS control. This illustrates that the friction compensation using a dynamic friction model is better than using a static friction model.

## 5.6 Comparison of Experimental Results

Figure 5.21 to 5.24 present the comparative results of five different controllers to track a circle trajectory. Figure 5.21 and Figure 5.27 show the tracking errors for both links under

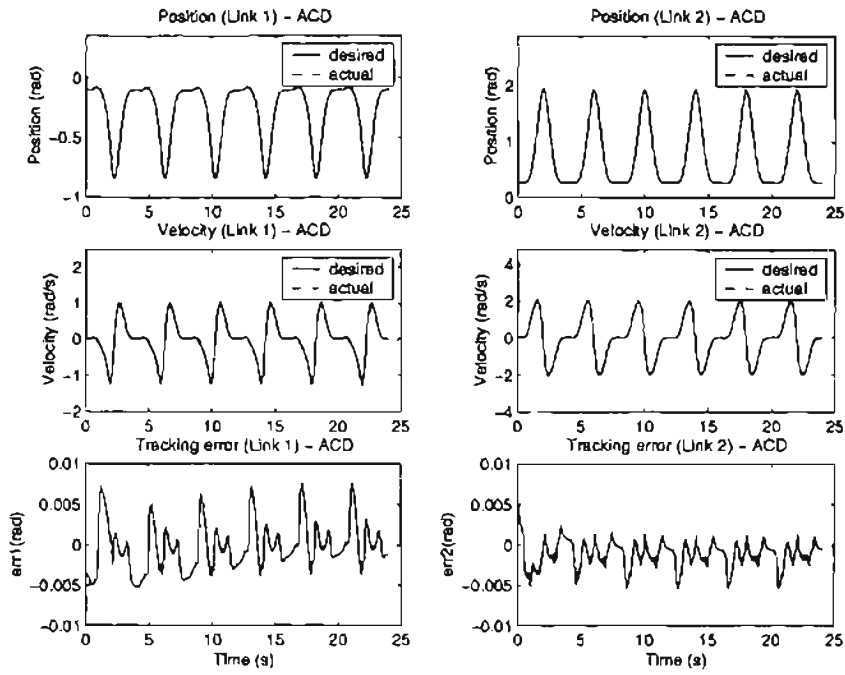


Figure 5.16: Adaptive control with dynamic friction compensation experimental result: velocity and tracking error (circle trajectory)

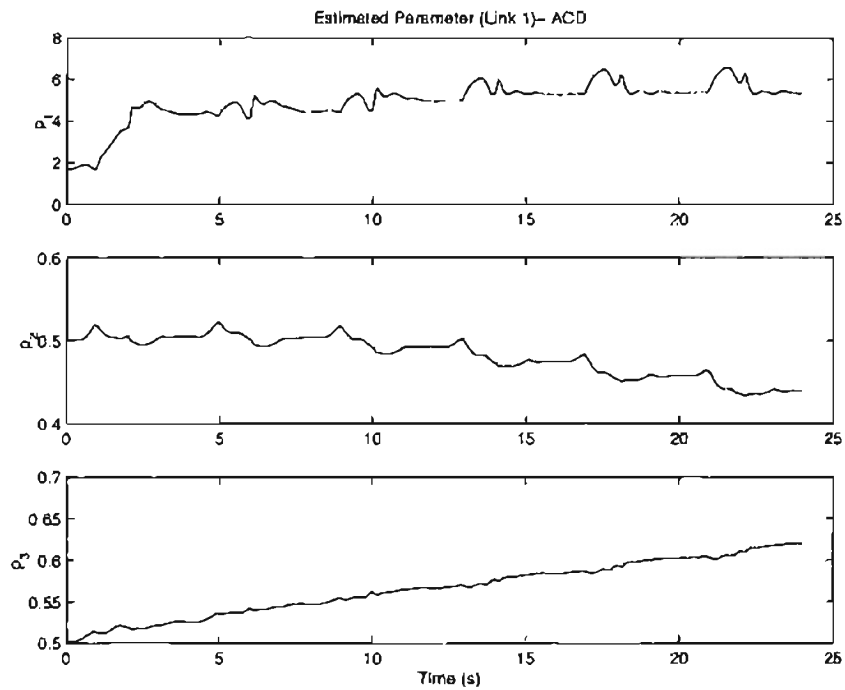


Figure 5.17: Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{p}_1, \hat{p}_2, \hat{p}_3$ ) (circle trajectory)

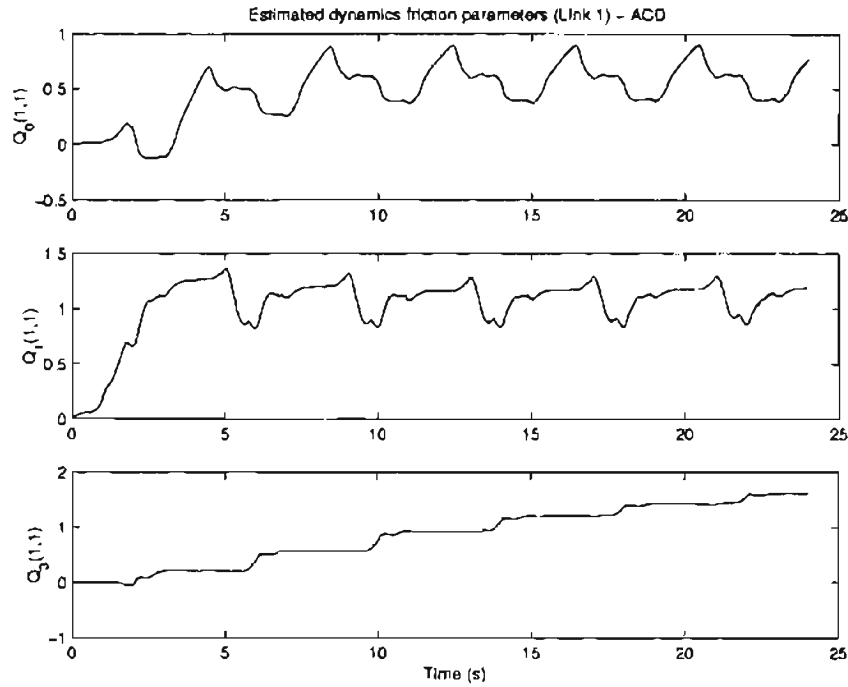


Figure 5.18: Adaptive control with dynamic friction compensation experimental result: estimated parameters  $(\hat{Q}_0(1, 1), \hat{Q}_1(1, 1), \hat{Q}_3(1, 1))$  (circle trajectory)

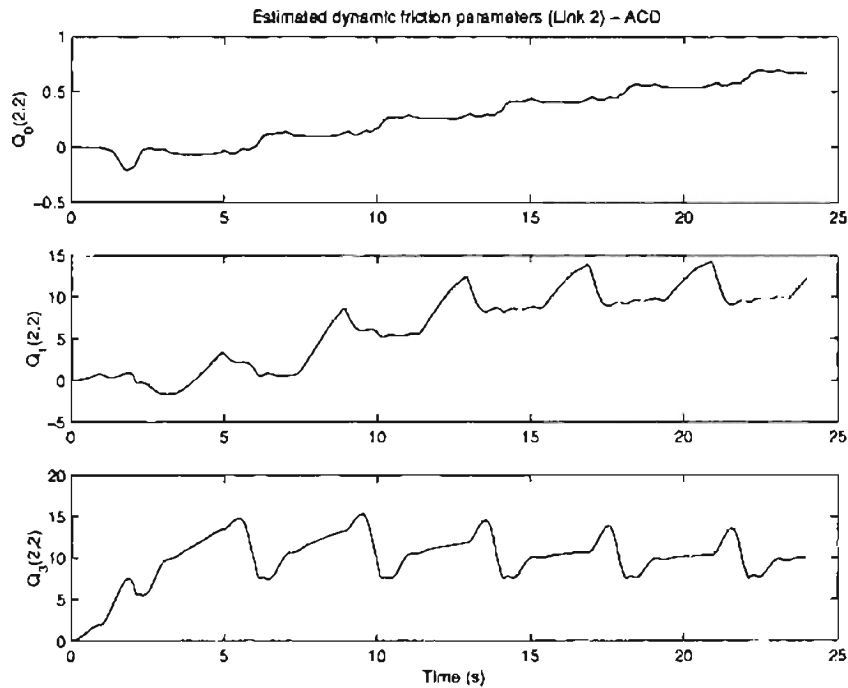


Figure 5.19: Adaptive control with dynamic friction compensation experimental result: estimated parameters  $(\hat{Q}_0(2, 2), \hat{Q}_1(2, 2), \hat{Q}_3(2, 2))$  (circle trajectory)



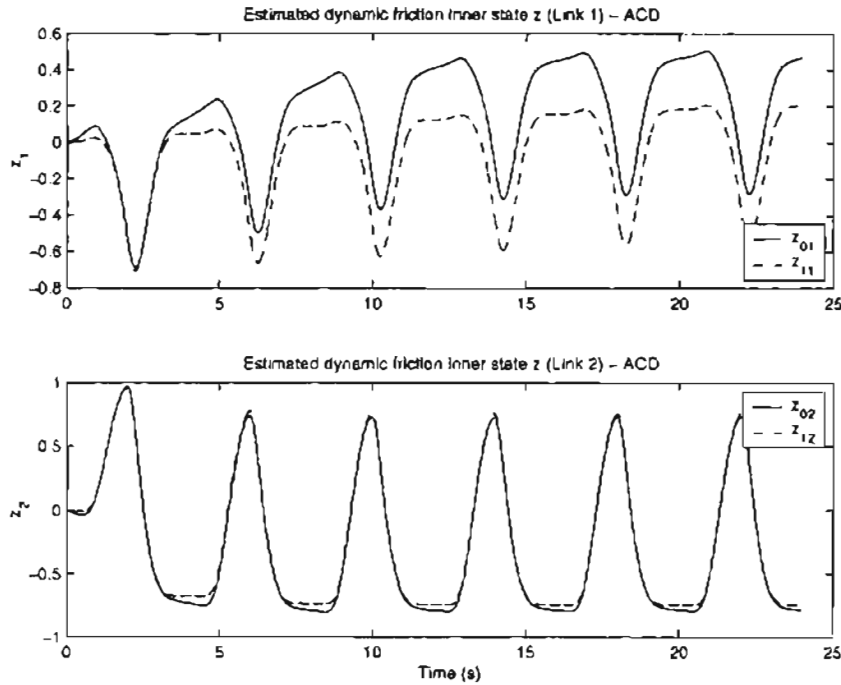


Figure 5.20: Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{z}_{01}$ ,  $\hat{z}_{11}$ ,  $\hat{z}_{02}$ ,  $\hat{z}_{12}$ ) (circle trajectory)

four controllers. Figure 5.23 and Figure 5.24 give the joint errors of the first two cycles of each controller, which are put side-by-side to compare the relative performance of different controllers. The 2 norm of the tracking error is also shown. It can be observed from the tracking error that the adaptive controller with dynamic friction compensation(ACD) is considerably superior in performance to the other three controllers.

To further test the performance of ACD, another set of experiments, which use a sinusoid as a desired trajectory are conducted. The desired position, velocity and acceleration trajectories are shown in Figure 5.25. Figure 5.26 to 5.33 show the experimental results of the four controllers.

Figure 5.26 and 5.27 show the tracking error for link 1 and link 2, respectively. It can be seen that the tracking error using ACD is about 5 to 10 times smaller than the other three controllers. Also, observe that the controller ACS performs better than the controllers CT and AC. Notice that the tracking error for the three controllers, AC, CT, and ACS, is

high when the link is going through the zero velocity region; change in friction torque is significant in this region. Although similar tendency can be observed for the ACS controller but dynamic friction compensation seems to significantly account for the friction changes and hence has much lower tracking errors.

The estimates of the static friction parameters are shown in Figure 5.28. Estimates of the dynamic friction parameters for link 1 and link 2 are shown in Figure 5.29, Figure 5.30 and Figure 5.31, respectively. Observe that all the estimated friction parameters tend to converge to a constant value as more number of cycles are implemented. The estimated internal state  $\hat{z}$  tries to track the angular position.

Figure 5.32 shows the position tracking error and the 2-norm of the position tracking error for link 1. The 2-norm shown is for all six cycles of data. Similarly, Figure 5.33 shows the position tracking error and the 2-norm of the position tracking error for link 2. Notice that for each link the 2-norm of the tracking error with the ACD controller is considerably smaller than the other controllers.

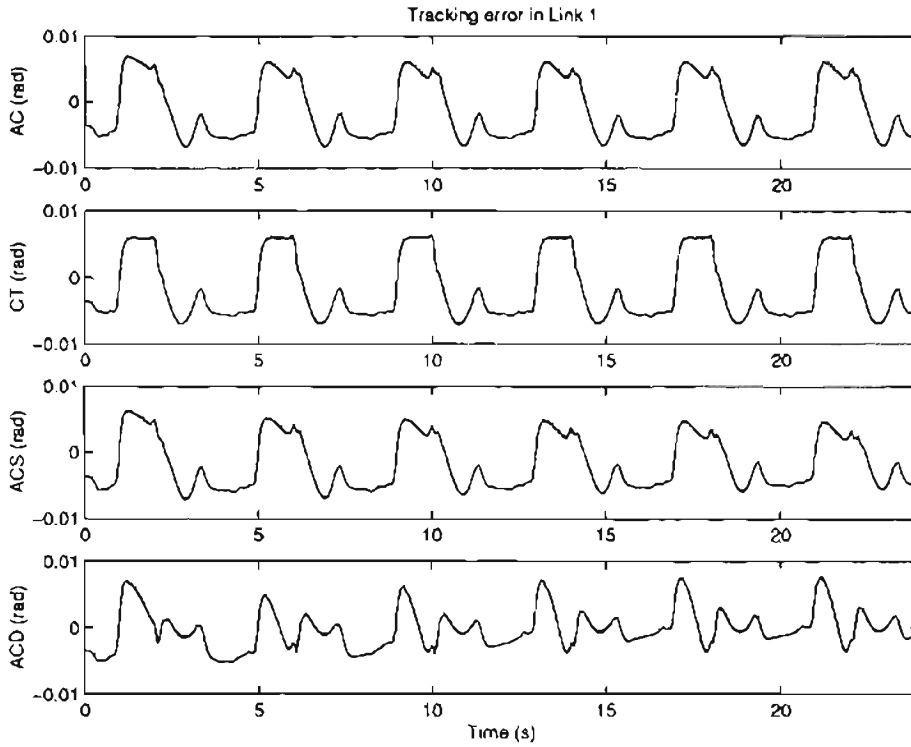


Figure 5.21: Comparison of tracking errors for link 1 (circle trajectory)

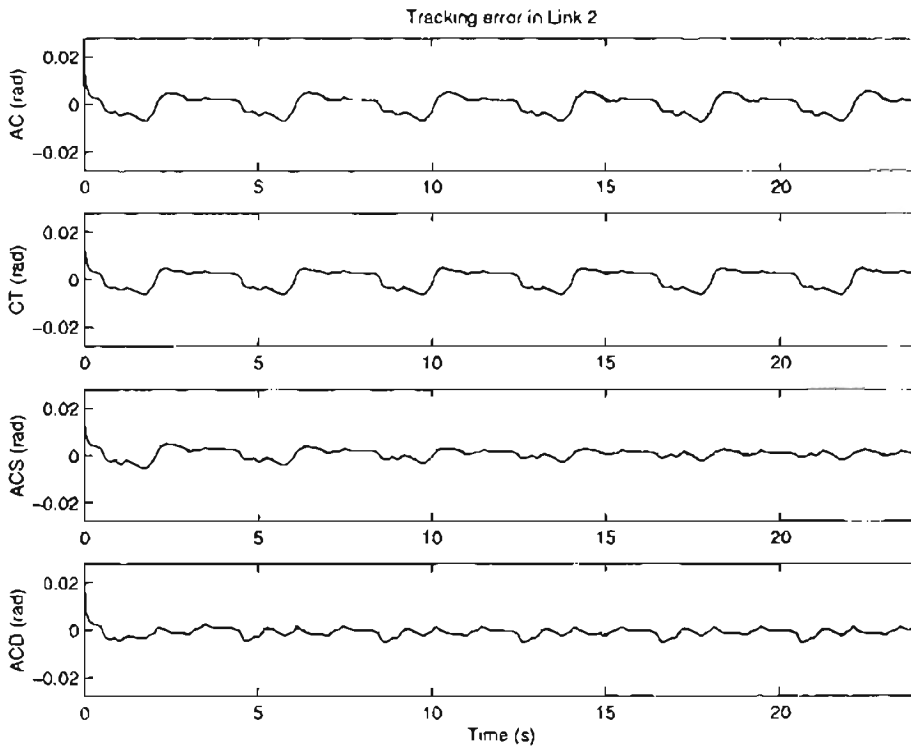


Figure 5.22: Comparison of tracking errors for link 2 (circle trajectory)

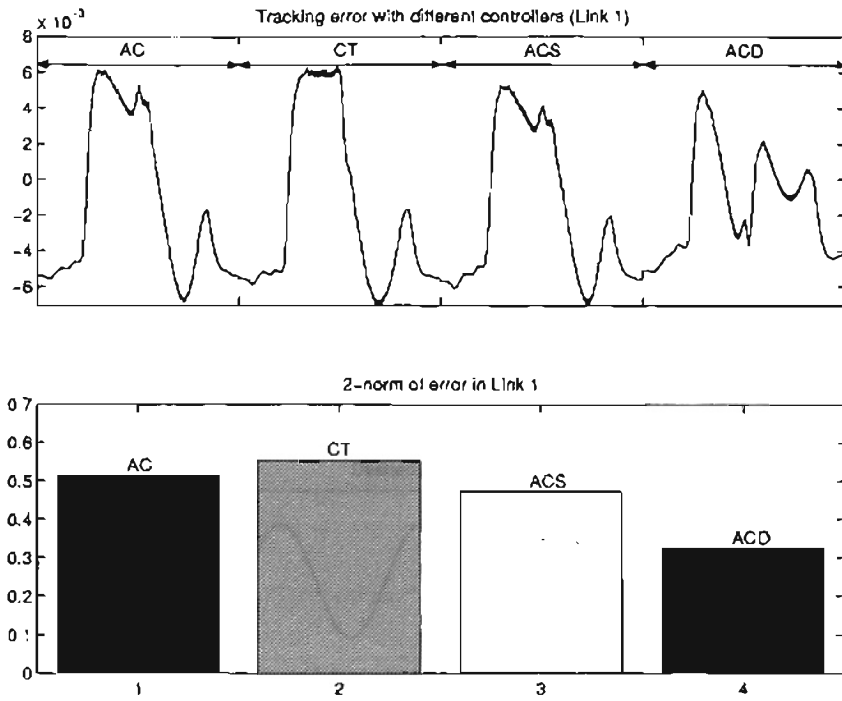


Figure 5.23: Tracking error and its 2-norm for link 1 (circle trajectory)

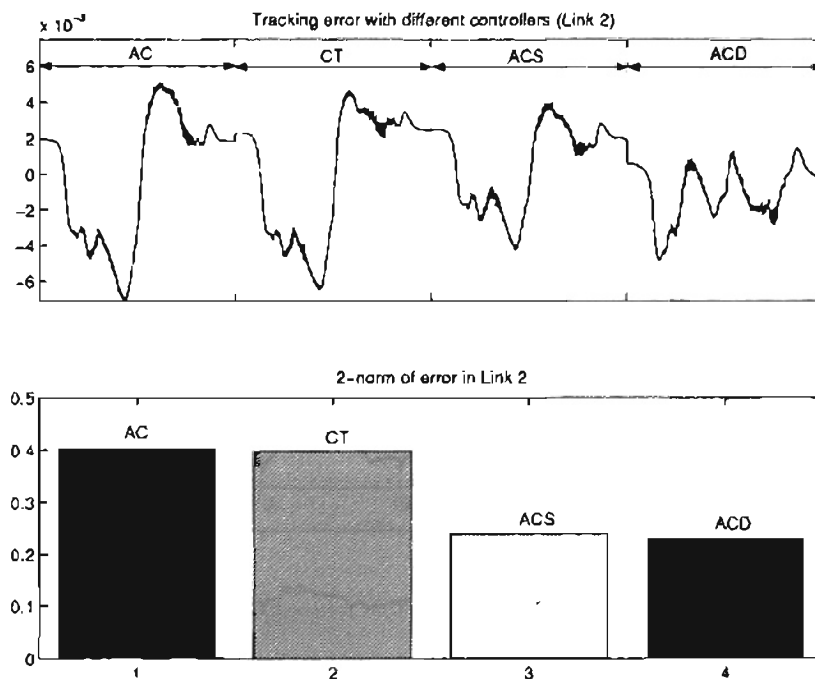


Figure 5.24: Tracking error and its 2-norm for link 2 (circle trajectory)

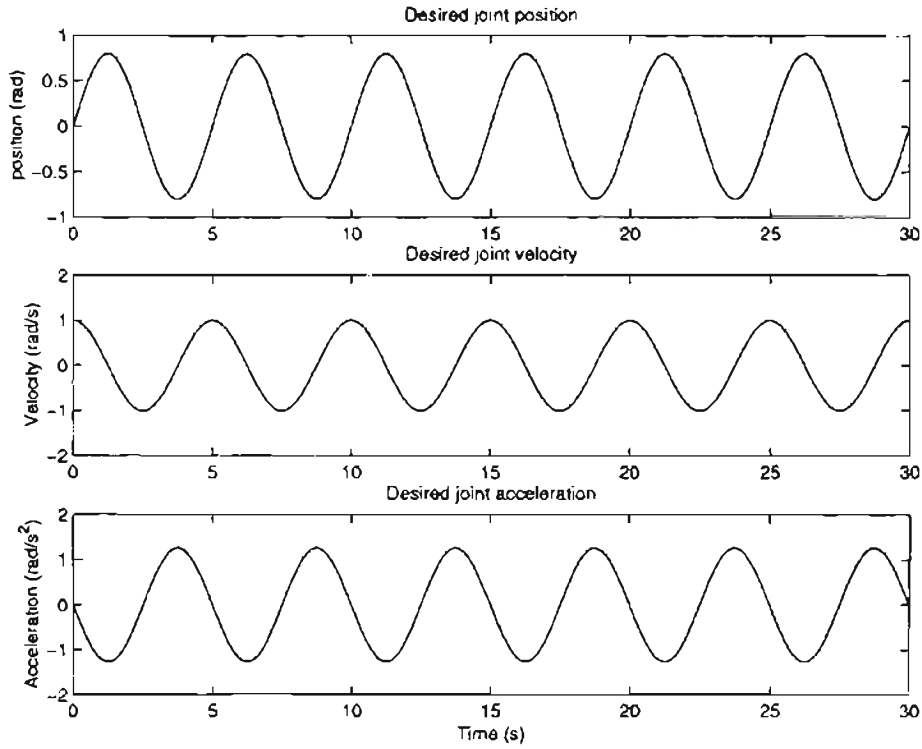


Figure 5.25: Desired trajectories for link 1 and 2 (Sine trajectory)

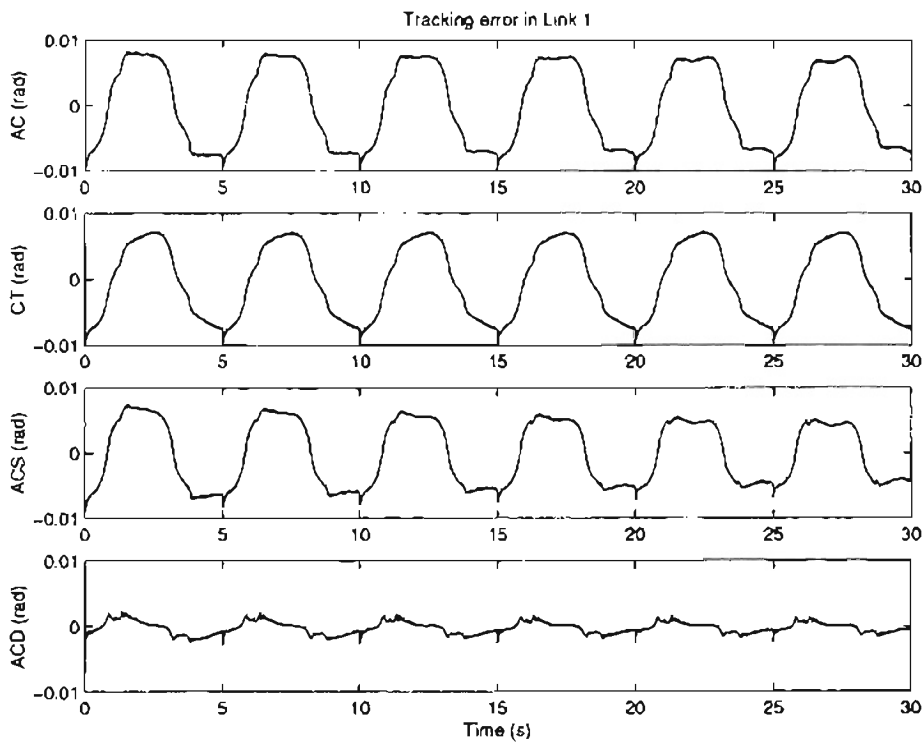


Figure 5.26: Comparison of tracking errors for link 1 (Sine trajectory)

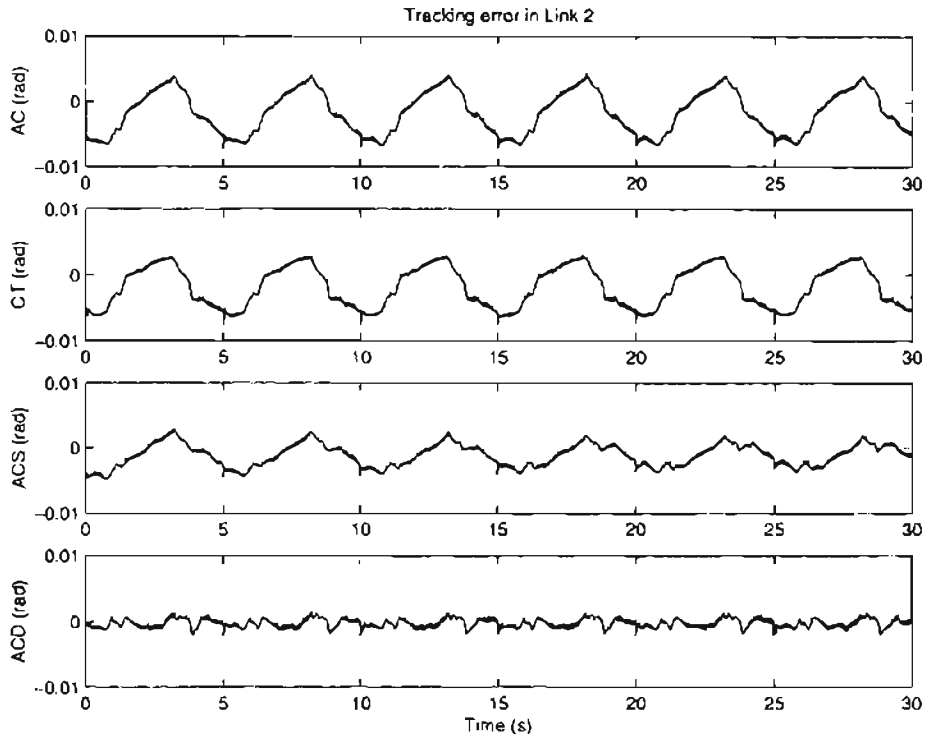


Figure 5.27: Comparison of tracking errors for link 2 (Sine trajectory)

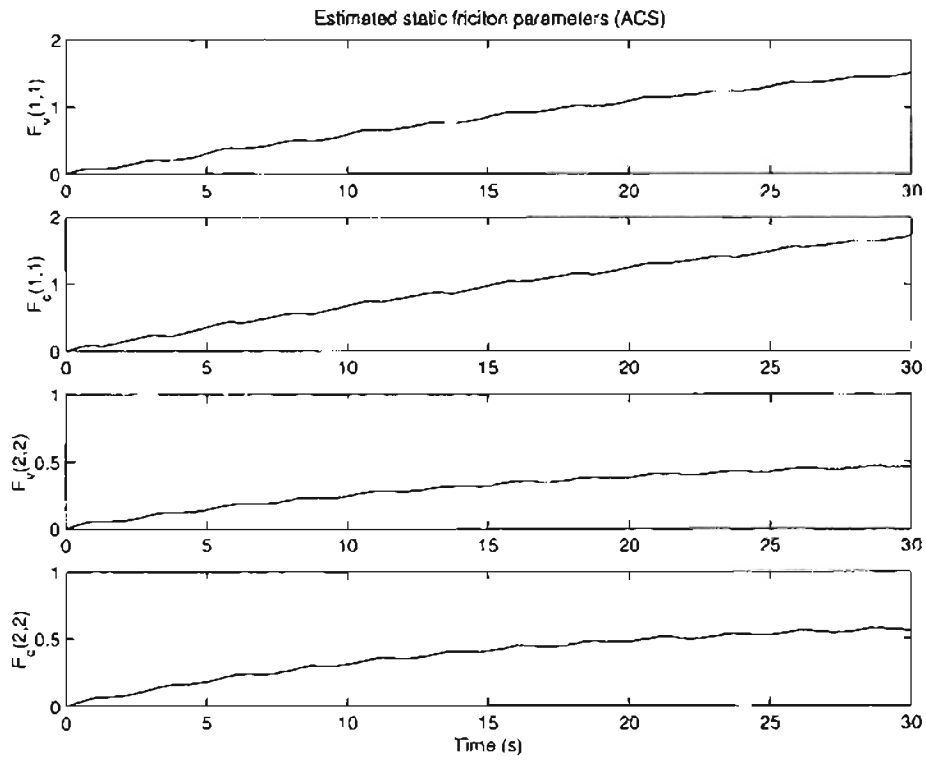


Figure 5.28: Estimated static friction parameters (Sine trajectory)

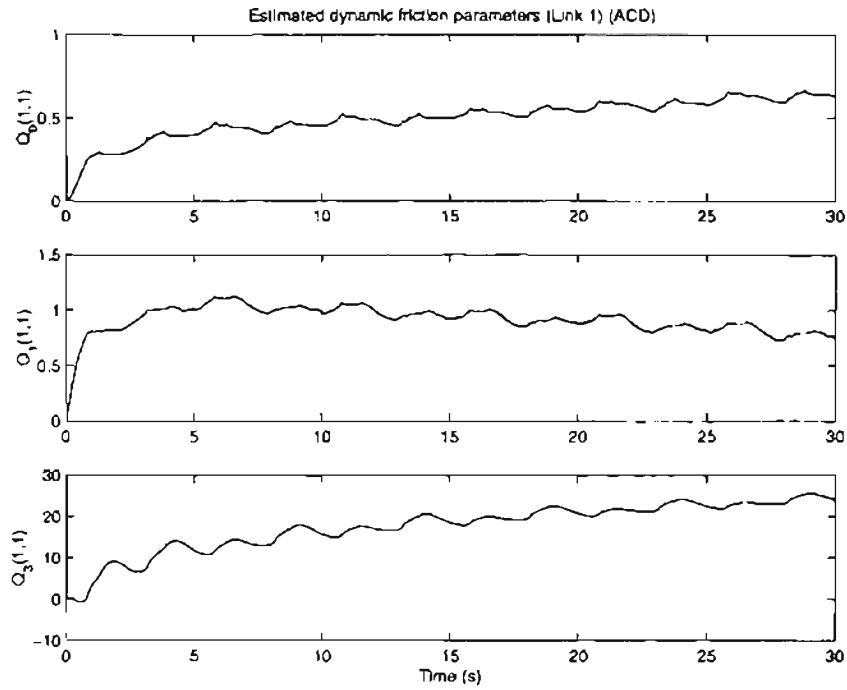


Figure 5.29: Estimated dynamic friction parameters ( $\hat{Q}_0, \hat{Q}_1, \hat{Q}_3$ ) for link 1 (Sine trajectory)

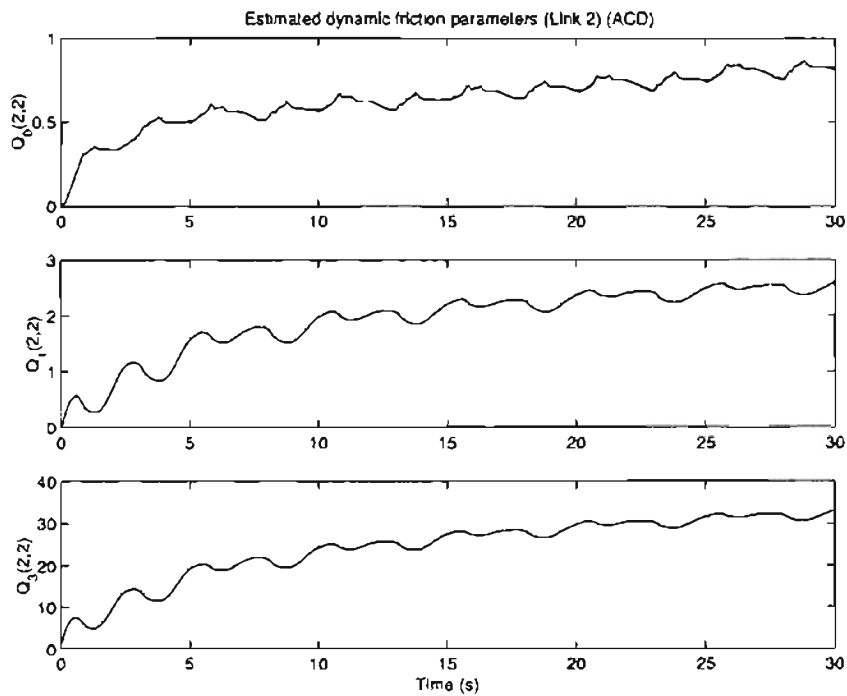


Figure 5.30: Estimated dynamic friction parameters ( $\hat{Q}_0, \hat{Q}_1, \hat{Q}_3$ ) for link 2 (Sine trajectory)

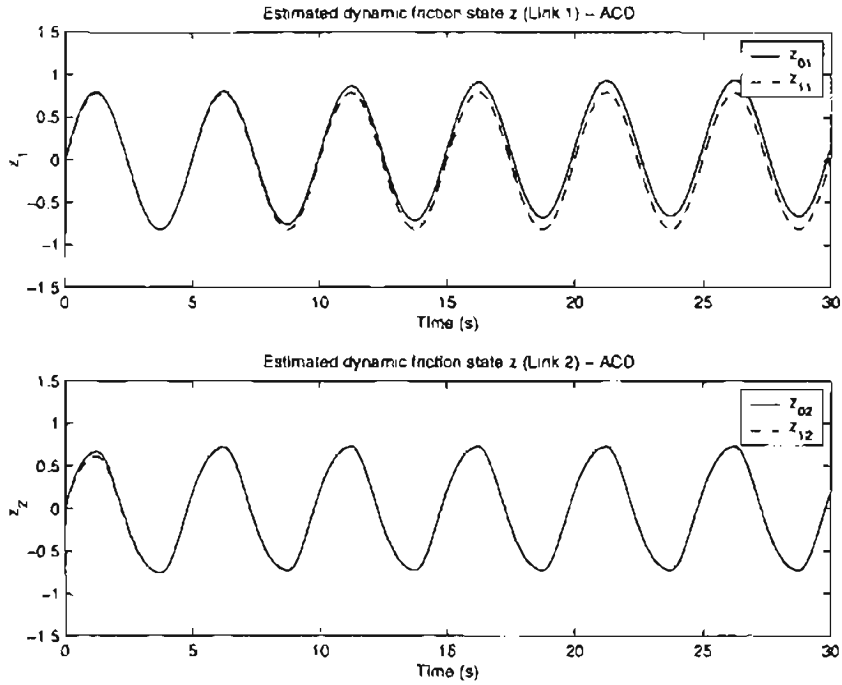


Figure 5.31: Adaptive control with dynamic friction compensation experimental result: estimated parameters ( $\hat{z}_{01}$ ,  $\hat{z}_{11}$ ,  $\hat{z}_{02}$ ,  $\hat{z}_{12}$ ) (sine trajectory)

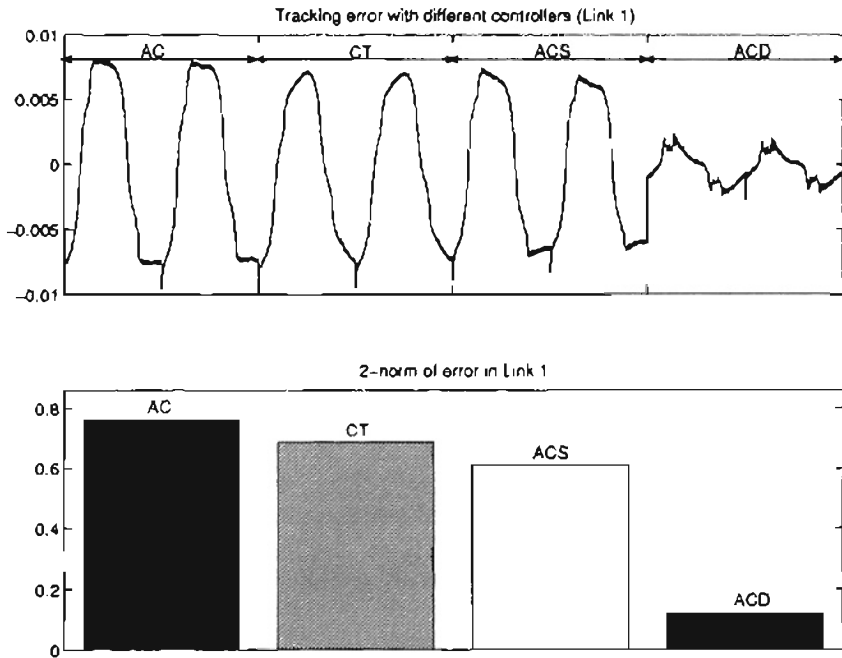


Figure 5.32: Tracking error and its 2-norm for link 1(Sine trajectory)



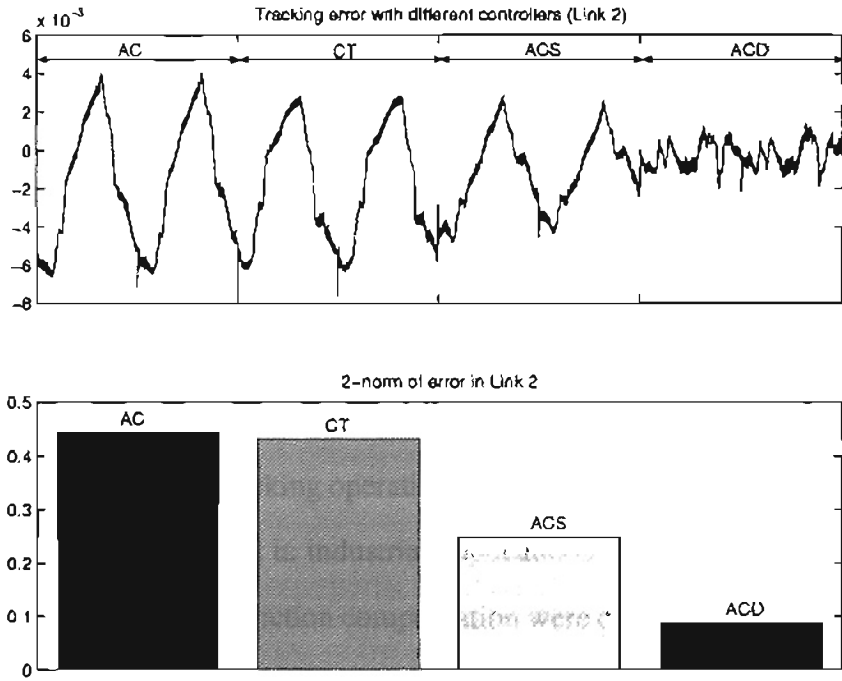


Figure 5.33: Tracking error and its 2-norm for link 2 (Sine trajectory)

## CHAPTER 6

### Summary and Future Work

#### 6.1 Summary

Tracking performance of mechanical systems with friction was studied in this thesis. As a special case, a robot is selected as the real plant. Modeling, control design, investigation of friction model for mechanical systems were considered. The dynamic model of robots performing trajectory tracking operations was formulated in an efficient and realistic manner that can be readily used in industrial applications. Based on this dynamic model, robust control algorithms with friction compensation were developed for robots performing high accuracy tracking operations. The control algorithms that were developed in this thesis are robust to uncertain parameters of the robot and uncertain parameters of the friction model. An important aspect that is studied in this thesis is to reduce the trajectory tracking error for mechanical systems. The stability of the robot in the presence of friction and uncertain parameters has been proved. In addition to the theoretical contributions towards this problem, some real-time software improvement has been done on the open-architecture experimental platform. Extensive experiments were conducted on a two-link manipulator, and experimental results verify the feasibility of proposed control designs with friction compensation.

Five types of control designs are considered in the thesis. They are: (1) PD control (PD); (2) computed torque control (CT); (3) model based adaptive control (AC); (4) model based adaptive control with static friction compensation (ACS); and (5) model based adaptive control with dynamic friction compensation (ACD). Control law of each control is given in the thesis. Stability of the two adaptive controllers with friction compensation

(ACS and ACD) that were proposed was shown.

PD control is the most commonly used controller. It is simple and robust. For applications demanding high accuracy tracking performance, PD controller cannot achieve desired result in the presence of dynamic uncertainties and friction.

Computed torque approach is well known in robotic applications. It consists of a forward compensation part and a PD part. To use the computed torque approach, the dynamics of the robot should be known, which sometimes is not possible.

Model based adaptive control is prevalent in robotics literature. Usually, smooth Cartesian motions are planned a priori and the inverse kinematics is used to compute the desired joint position, velocity, acceleration. This kind of controller is advantageous since it follows the desired joint trajectories corresponding to the planned Cartesian motion, and it does not require assumption or simplification such as local linearization, time invariance, or decoupled-dynamics. The other important advantage is it does not need the measurement of the joint acceleration or inversion of the estimated inertia matrix.

The adaptive controller used in the experiment enjoys essentially the same level of robustness to measurement noise and unmodeled dynamics as the PD controller, yet achieves higher tracking accuracy. The parameter estimation should be stopped when the estimated parameters reach an upper bound. Stopping the adaptation process does not affect the stability of the closed-loop system.

Friction is a complicated phenomenon, which exhibits coulomb friction, viscous friction and nonlinearity at low velocity. The classic friction models cannot sufficiently describe all the dynamic effects of friction, such as pre-sliding, the friction lag and Stribeck effect. These phenomena all occur in the low velocity and pre-sliding region. Compared to the Dahl model and the Bliman-Sorine model, the LuGre friction model includes more dynamic effects of friction.

To decrease the friction effect, friction compensation is incorporated into the adaptive control design. Two types of friction models were used in the control design. They are

the static friction model, which considers Coulomb and viscous friction, and the LuGre dynamic friction model. The internal state (bristle displacement  $z$ ) and three of the six parameters in LuGre model were estimated.

To compare the performance of the proposed control designs in this thesis, experiments were conducted on a two-link manipulator. The sampling and control period was chosen to be 2 milli-second. All the five controllers (PD, CT, AC, ACS and ACD) were implemented in real-time and the control algorithms were programmed in an efficient way such that the tracking error, joint velocity, the estimated parameters and the control torque data were obtained for off-line analysis. To consistently compare controllers, the control gains that are common to all the controllers were set to the same value for each controller implementation. The initial values of the estimated parameters in the controllers were chosen as follows: (1) the initial values of the robot dynamic parameters  $p_1$ ,  $p_2$  and  $p_3$  in AC, ACS and ACD were chosen as half of their nominal values; (2) the initial values of the Coulomb and the viscous friction coefficients in ACS were chosen to be zero; (3) the initial values of the estimate of the internal state  $z$ , the rest stiffness coefficient  $\beta_0$ , the velocity dependent damping coefficient  $\beta_1$ , and the viscous friction coefficient  $\beta_2$  in ACD were all chosen to be zero.

To illustrate the performance of the proposed controllers to various types of trajectories, in this thesis two different types of desired trajectories were considered; circle trajectory in Cartesian space and sinusoidal trajectory in joint space. Experimental data for six cycles of the trajectory was collected for each controller for comparison. The following observations can be made based on the experimental data:

- CT, AC, ACS and ACD all achieved better tracking performance than PD control.
- AC controller achieved about the same tracking error as CT controller, but AC does not require accurate knowledge of parameters of the dynamic model.
- ACS and ACD gave smaller tracking error than the other three controllers. This indicates that friction compensation plays an important role in improving the tracking

performance. Furthermore, it can be observed from the tracking error that the adaptive controller with dynamic friction compensation(ACD) is considerably superior in performance to the other four controllers. For the sinusoidal trajectory, where the robot motion is in the low velocity region and crosses the zero velocity point frequently, ACD reduced the tracking error significantly. The tracking error using ACD is about half of that of ACS and about one-fifth of the other three controllers.

## 6.2 Future Work

In this work, control design with friction compensation was investigated for robotic applications. Future work should focus on some potential improvements to the proposed control algorithms and potential applications in other areas.

Model based adaptive control with dynamic friction compensation with application to robot motion control was the focus in this thesis. However, friction in all mechanical systems is known to exhibit similar behavior. Besides the robotic applications, the control schemes developed in this thesis can be applied to many industrial applications such as web handling systems, CNC machines, and excavators [23]. In web handling systems, the lateral position of a web is usually controlled by steering guides. A steering guide consists of a DC motor and some mechanical structures. Friction affects the lateral position accuracy and should be compensated. Another examples is CNC machining. It is well known that CNC machining system usually consists of multi-axis machines and cutting tools. High accuracy positioning cannot be achieved if the friction effects are not considered. Excavators are typical manipulator-like structures with four degrees of freedom. The identification of friction parameters and the control design with friction compensation for excavator arms can be conducted using the methods outlined in this thesis.

The LuGre friction model is characterized by six parameters ( $\beta_0, \beta_1, \beta_2, F_c, F_s$  and  $\omega_s$ ). Currently, the identification of the parameters of the LuGre friction model is conducted via several procedures. In each procedure, the LuGre friction model is simplified by either dif-

ferent velocity regions or nonlinear performance in consideration of the friction behavior. These experiments for identifying the friction parameters are off-line, time-consuming, and maybe impossible for some systems. Hence, the identification of the friction parameters in LuGre friction model in one experiment which can be readily implemented is a challenging task for the future. This task includes designing the experiment and the identification algorithm for estimating all the friction parameters together.

From the adaptive control point of view, the control algorithm may avoid the requirement of the actual parameters in the LuGre friction model. In the model based adaptive control with dynamic friction compensation proposed in this thesis, three parameters ( $F_c$ ,  $F_s$  and  $\omega_s$ ) were identified off-line. Future work should focus on designing an adaptive controller with dynamic friction compensation with all six unknown friction parameters. Hence designing a control algorithm with estimation of all the friction parameters on-line will also be a challenging task for the future.

## BIBLIOGRAPHY

- [1] Y.D. Song, "Adaptive Motion Tracking Control of Robot Manipulators - Non-regressor based Approach," *Proc. of IEEE International Conference on Robotics and Automation, San Diego*. Vol. 4, pp 3008 – 3031, May, 1994
- [2] J.J.E. Slotine and W. Li, "Adaptive Manipulator Control: A Case Study," *IEEE Trans. on Automatic Control*, Vol. AC-33, No. 11, pp. 995 – 1003, November, 1988.
- [3] J.J.E. Slotine and W. Li, "On the Adaptive Control of Robot manipulators," *Proc. ASME Winter Annual Meeting*, pp. 55 – 56, December, 1986.
- [4] Y. Stepanenko and C.Y. Su, "Adaptive Motion Control of Rigid-Link Electrically-Driven Robot Manipulators," *Proc. of IEEE International Conference on Robotics and Automation, San Diego, USA*, Vol. 1, pp. 630 – 635, May, 1994.
- [5] J.B. Pomet and L. Praly, "Adaptive Nonlinear Regulation: Estimation from the Lyapunov Equation," *IEEE trans. Automation Control*, Vol. 37, pp. 729 – 740, 1992.
- [6] P. Tomei, "Robust Adaptive Friction Compensation for Tracking Control of Robots," *Proc. of the 1999 IEEE International Conference on Control Application, Hawi, USA*, pp. 875 – 879, August, 1999.
- [7] R. Colbaugh and H. Seraji, "Adaptive Tracking Control of Manipulators: Theory and Experiments," *Proc. of IEEE International Conference on Robotics and Automation, San Digeo, USA*, Vol. 4, pp. 2992 – 2999, May, 1994.

- [8] G. Niemeyer and J.J.E. Slotin, "Performance in Adaptive Manipulator Control," *Proc. of the 27th Conference on Decision and Control, Austin, USA*, pp. 1585 – 1591, December, 1988.
- [9] J.J. Alsina and N.S. Gethlot, "Adaptive Controller of Robotic Manipulators: a Modular Approach," *Proc. of the 34th Conference on Decision and Control, New Orleans, USA*, pp. 57 – 62, 1999.
- [10] M. Gafvert, "Comparisons of Two Dynamic Friction Models," *Proc. of the 1997 IEEE international Conference on Control Applications, Hartford, CT*, pp. 386 – 391, 1997.
- [11] C.C. Wit, H. Olsson, K.J. Astrom, and P. Lischinsky, "A New Model for Control of Systems with Friction," *IEEE Trans. on Automatic Control*, Vol. 40, No. 3, pp. 419 – 425, March, 1995.
- [12] C.C. Wit, "Comments on A New Model for Control of Systems with Friction," *IEEE Trans. on Automatic Control*, Vol. 43, No. 8, pp. 1189 – 1190, August, 1998.
- [13] C.C. Wit and P. Lischinsky, "Adaptive Friction Compensation with Partially Known Dynamic Friction Model," *International Journal of Adaptive Control and Signal Processing*, 11:65–80, 1997.
- [14] N.E. Leonard and P.S. Krishnaprasad, "Adaptive Friction Compensation for Bidirectional Low-velocity Position Tracking," *Proc. of the 31st IEEE Conference on Decision and Control*, Vol. 1, pp. 267 – 273, 1992.
- [15] S. Cong, "Two Adaptive Friction Compensations for DC Servomotors," *Proc. of the IEEE International Conference on Industrial Technology*, pp. 113 – 117, 1996.
- [16] Y. Tan and I. Kanellakopoulos, "Adaptive Nonlinear Friction Compensation with Parametric Uncertainties," *Proc. of American Control Conference, San Diego, California*, pp. 2511 – 2515, June, 1999.



- [17] P.R. Pagilla and B. Yu, "A Stable Transition Controller for Constrained Robots," *IEEE Transactions on Mechatronics*, Vol. 6, No. 1, pp. 65 – 74, March, 2001.
- [18] J.J.E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, NJ, 1991.
- [19] S.S. Neo and M.J. Er, "Adaptive fuzzy control of robot manipulators Control Applications," *Proceedings of the 4th IEEE Conference on Control Applications*, pp. 724 – 729, 1995
- [20] M. Jansen and R. Eckmiller, "Globally Stable Neural Robot Control Capable Of Payload Adaptation," *Proc. of 1993 International Joint Conference on Neural Networks, Nagoya, Japan*, Vol. 1, pp. 639 – 642, 1993.
- [21] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronics systems," *Proc. of the IEEE Conference on Decision and Control Including The Symposium on Adaptive Processes*, pp. 1064 – 1069, 1984.
- [22] S. Arimoto, "Robustness of learning control for robot manipulators," *Proc of the IEEE Conference on Robotics and Automation*, Vol. 3, pp. 1528 – 1533, May, 1990.
- [23] S. Tafazoli, P.D. Lawrence, and S.E. Salcudean, "Identification of Inertial and Friction Parameters for Excavator Arms," *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 5, pp. 966 – 971, October, 1999.

## APPENDIX A

### Manipulator Application Software

#### A.1 Servo.c

```
#include <cntrl.h>
#include <math.h>

#define PTS 2503
#define MAX_TAU1 200
#define MAX_TAU2 20

#define VVFF_lamap2 15.0
#define VVFF_lamai2 0.01
#define VVFF_Fv2 25.0
#define VVFF_beta32 0
#define VVFF_beta02 0
#define VVFF_beta12 0
#define VVFF_gamma02 5.0
#define VVFF_gamma12 5.0
#define VVFF_gamma32 5.0
#define VVFF_z02 0.0
#define VVFF_z12 0.0

#define VVFF_lamap1 15.0
#define VVFF_lamai1 0.01
#define VVFF_Fv1 100.0
#define VVFF_beta31 0
#define VVFF_beta01 0
#define VVFF_beta11 0
#define VVFF_gamma01 5.0
#define VVFF_gamma11 5.0
#define VVFF_gamma31 5.0
```

```

#define VVFF_z01 0.0
#define VVFF_z11 0.0

#define VVFF_p1 1.7
#define VVFF_p2 0.5
#define VVFF_p3 0.5
#define VVFF_f1 0.0
#define VVFF_f2 0.0
#define VVFF_g1 0.0
#define VVFF_g2 0.0
#define VVFF_gAmmap1 1.0
#define VVFF_gAmmap2 .2
#define VVFF_gAmmap3 .2
#define VVFF_gAmmaf1 1
#define VVFF_gAmmaf2 1
#define VVFF_gAmmag1 1
#define VVFF_gAmmag2 1

float VF_tau2max=19;
float VF_tau1max=121;
float VF_csaveflag=99;

float VF_abs_err1=0;
float VF_abs_err2=0;
float abs_err1=0;
float abs_err2=0;

float tau2max=19;
float tau1max=121;

float c2r=0.00004091,t2c_1=8.3551,t2c_2=51.75;
float pi=3.1415927;
float v2c=204.8;
float Ts=0.002;
float alpha=0.0;
float l1=14.5*0.0254,l2=9.5*0.0254;

```

```

float c2=0,s2=0,c12=0,s12=0;

int iSave=0;
int iDisplay=0;

float tau1=0,tau2=0;

float TF_output[0x8000];
float TF_q1[PTS];
float TF_q2[PTS];
float TF_dq1[PTS];
float TF_dq2[PTS];
float TF_ddq1[PTS];
float TF_ddq2[PTS];
float p1=3.4, p2=0.4, p3=0.3;
float ft_data[3];

float kp1=85*24,kp2=18*4;
float kd1=1.9*24,kd2=1.9*4;
float ki1=0, ki2=0;
float VF_kp1=85*24,VF_kp2=18*4;
float VF_kd1=1.9*24,VF_kd2=1.9*4;

float VF_u1=0, VF_u2=0;
float VF_q1=0, VF_q2=0;
float VF_display=0;
float VF_controlflag=10.0; /*PD*/
float VF_controlflag1=0;
float VF_csaveflag1=0;
int controlflag=0;

float VF_lamap2=15, VF_lamai2=0;
float VF_lamap1=20, VF_lamai1=0;
float VF_Fv1=VVFF_Fv1;
float VF_Fv2=VVFF_Fv2;

```

```

float VF_beta32=0, VF_beta02=0, VF_beta12=0;
float VF_gamma02=5, VF_gamma12=5, VF_gamma32=5;
float Fc2=0.7805, Fs2=1.6;
float inv_vs2=(1.0/0.096)*(1.0/0.096);
float VF_z02=0, VF_z12=0, varphi2=0;

float VF_beta31=0, VF_beta01=0, VF_beta11=0;
float VF_gamma01=5, VF_gamma11=5, VF_gamma31=5;
float Fc1=1.857, Fs1=11,
float inv_vs1=(1.0/0.14)*(1.0/0.14);
float VF_z01=0, VF_z11=0, varphi1=0;

float oldp1=0, oldp2=0, oldp3=0, oldf1=0;
float oldf2=0, oldg1=0, oldg2=0;
float VF_p1=1.7, VF_p2=0.5, VF_p3=0.5;
float VF_f1=0.0, VF_f2=0.0;
float VF_g1=0.0, VF_g2=0.0;
float VF_gAmmap1=1, VF_gAmmap2=.2, VF_gAmmap3=.2;
float VF_gAmmaf1=0, VF_gAmmaf2=0;
float VF_gAmmag1=0, VF_gAmmag2=0;

int VI_PTS=PTS; /* # of pts in a circle*/
int pts=0;
float VF_CYCNUM=0.0; /*# of circle*/

float serr1=0, serr2=0, rate_fv=1.0;
float err1=0, err2=0, olderr1=0, olderr2=0;
float derr1=0, derr2=0;
float ev1=0, ev2=0;

int i=0, j=0; /* counters or index */
unsigned long k=0; /* index for saving data*/
float r1=0, r2=0;
float qq1=0, qq2=0;

float q1d, q2d, dq1d, dq2d, ddq1d, ddq2d;

```

```

float dq1r=0,dq2r=0,ddq1r=0,ddq2r=0,oldq1=0;
float oldq2=0,dq1=0,dq2=0,olddq1=0,olddq2=0,ddq1=0,ddq2=0;

float tmp1=0,tmp2=0,tmp3=0;
float Yp11=0,Yp12=0,Yp13=0,Yp21=0,Yp22=0,Yp23=0;

init_control()
{
    i=0;
    k=0;
    VI_PTS=1300;
    Ts=0.002;
    r1=0;
    r2=0;
    User_Data[0]=0;
    VF_CYCNUM=1;
    serr1=0;
    serr2=0;
}

control()
{
    qq1=(float)pos1;qq2=(float)pos2;

    kp1=VF_kp1;
    kp2=VF_kp2;
    kd1=VF_kd1;
    kd2=VF_kd2;
    controlflag=(int)VF_controlflag;
    pts=VI_PTS;

    tau2max=VF_tau2max;
    tau1max=VF_tau1max;

/* inactive the robot */

```

```

if(controlflag==100)
{
    i=0;
    k=0;
    u1=0;
    u2=0;
    ref1=pos1;
    ref2=pos2;
    Host_Trigger=0.0;
    goto m1;
}
else if(controlflag==98)
{
    i=0;
    k=0;
    u1=0;
    u2=0;
    ref1=pos1;
    ref2=pos2;
    Host_Trigger=0.0;
    goto m1;
}
else if(controlflag==97)
{
    i=0;
    k=0;
    u1=0;
    u2=0;
    ref1=0;
    ref2=0;
    Host_Trigger=0.0;
    goto m1;
}
else if(controlflag==99)
{
    VF_lamap2=VVFF_lamap2;

```

VF\_lamai2=VVFF\_lamai2;  
VF\_Fv2=VVFF\_Fv2;  
VF\_beta32=VVFF\_beta32;  
VF\_beta02=VVFF\_beta02;  
VF\_beta12=VVFF\_beta12;  
VF\_gamma02=VVFF\_gamma02;  
VF\_gamma12=VVFF\_gamma12;  
VF\_gamma32=VVFF\_gamma32;  
VF\_z02=VVFF\_z02;  
VF\_z12=VVFF\_z12;

VF\_lamap1=VVFF\_lamap1;  
VF\_lamai1=VVFF\_lamai1;  
VF\_Fv1=VVFF\_Fv1;  
VF\_beta31=VVFF\_beta31;  
VF\_beta01=VVFF\_beta01;  
VF\_beta11=VVFF\_beta11;  
VF\_gamma01=VVFF\_gamma01;  
VF\_gamma11=VVFF\_gamma11;  
VF\_gamma31=VVFF\_gamma31;  
VF\_z01=VVFF\_z01;  
VF\_z11=VVFF\_z11;

VF\_p1=VVFF\_p1;  
VF\_p2=VVFF\_p2;  
VF\_p3=VVFF\_p3;  
VF\_f1=VVFF\_f1;  
VF\_f2=VVFF\_f2;  
VF\_g1=VVFF\_g1;  
VF\_g2=VVFF\_g2;  
VF\_gAmmap1=VVFF\_gAmmap1;  
VF\_gAmmap2=VVFF\_gAmmap2;  
VF\_gAmmap3=VVFF\_gAmmap3;  
VF\_gAmmaf1=VVFF\_gAmmaf1;  
VF\_gAmmaf2=VVFF\_gAmmaf2;  
VF\_gAmmag1=VVFF\_gAmmag1;



```

VF_gAmmag2=VVFF_gAmmag2;

VF_controlflag=0;
goto m1;

}
/* home the robot */
else if(controlflag==7)
{

olderr1=err1; olderr2=err2;

err1=c2r*qq1-VF_q1;
err2=c2r*qq2-VF_q2;

derr1=(err1-olderr1)/Ts;
derr2=(err2-olderr2)/Ts;

serr1+=err1*Ts;serr2+=err2*Ts;

dq1=c2r*(qq1-oldq1)/Ts;
dq2=c2r*(qq2-oldq2)/Ts;

ddq1=c2r*(dq1-olddq1)/Ts;
ddq2=c2r*(dq2-olddq2)/Ts;

oldq1=qq1;oldq2=qq2;
olddq1=dq1;olddq2=dq2;

tmp1=(kp1*err1+kd1*derr1+ki1*serr1);
tmp2=(kp2*err2+kd2*derr2+ki2*serr2);

if(tmp2>tau2max)tmp2=tau2max;
else if (tmp2<-tau2max)tmp2=-tau2max;

if(tmp1>tau1max)tmp1=tau1max;

```

```

else if (tmp1<-taulmax)tmp1=-taulmax;

u1=- (int) (tmp1*t2c_1);
u2=- (int) (tmp2*t2c_2);
goto m1;

}

if(Host_Trigger==1.0)
{
  if(i<pts)
  {
    q1d=TF_q1[i];
    q2d=TF_q2[i];
    dq1d=TF_dq1[i];
    dq2d=TF_dq2[i];
    ddq1d=TF_ddq1[i];
    ddq2d=TF_ddq2[i];

    olderr1=err1; olderr2=err2;
    err1=c2r*qq1-q1d;
    err2=c2r*qq2-q2d;

    derr1=(err1-olderr1)/Ts;
    derr2=(err2-olderr2)/Ts;

    serr1+=err1*Ts; serr2+=err2*Ts;

abs_err1=(err1>0)?abs_err1+err1:abs_err1-err1;
abs_err2=(err2>0)?abs_err2+err2:abs_err2-err2;

    dq1=c2r*(qq1-oldq1)/Ts;
    dq2=c2r*(qq2-oldq2)/Ts;

    oldq1=qq1;oldq2=qq2;

```

```

rate_fv=1.0;
c2=cos(qq2*c2r);s2=sin(qq2*c2r);
tmp2=100.0;

if (controlflag==10.0)
{
    /*pd_part();*/
    tau1=-(kp1*err1+kd1*derr1);
    tau2=-(kp2*err2+kd2*derr2);
}
else if(controlflag==1.0)
{
    Common_part();
    adaptive_part();

    tau1=(Yp11*VF_p1+Yp12*VF_p2
          +Yp13*VF_p3)-VF_Fv1*rate_fv*ev1;

    if(dq1r>0)tau1+=VF_f1*dq1r+VF_g1;
    else
        tau1+=VF_f1*dq1r-VF_g1;

    tau2=(Yp22*VF_p2+Yp23*VF_p3)
          -VF_Fv2*rate_fv*ev2;
    if(dq2r>0)tau2+=VF_f2*dq2r+VF_g2;
    else
        tau2+=VF_f2*dq2r-VF_g2;
}
else if(controlflag==4.0)
{
    Common_part();
    adaptive_part();

    tau1=(Yp11*VF_p1+Yp12*VF_p2
          +Yp13*VF_p3)-VF_Fv1*rate_fv*ev1;
    tau2=(Yp22*VF_p2+Yp23*VF_p3)
          -VF_Fv2*rate_fv*ev2;
}

```

```

}
else if(controlflag==2.0)
{
    Common_part();
    LuGre_part();
    adaptive_part();
    tau1=(Yp11*VF_p1+Yp12*VF_p2+Yp13*VF_p3)
        -VF_Fv1*rate_fv*ev1;
    tau1=tau1+VF_beta31*dq1+VF_beta01*VF_z01
        -varphi1*VF_beta11*VF_z11;
    tau2=(Yp22*VF_p2+Yp23*VF_p3)
        -VF_Fv2*rate_fv*ev2;
    tau2=tau2+VF_beta32*dq2+VF_beta02*VF_z02
        -varphi2*VF_beta12*VF_z12;
}
else if(controlflag==8.0)
{
    Common_part();
    LuGre_part();
    tau2=Yp22*p2+Yp23*p3-VF_Fv2*ev2+VF_beta32*dq2
        +VF_beta02*VF_z02-varphi2*VF_beta12*VF_z12;
    tau1=ddq1r*p1+ddq2r*p2+Yp13*p3 - VF_Fv1*ev1
        +VF_beta31*dq1+VF_beta01*VF_z01
        -varphi1*VF_beta11*VF_z11;
}
else if(controlflag==3.0)
{
    Common_part();
    tau2=Yp22*p2+Yp23*p3 -VF_Fv2*ev2;
    tau1=ddq1r*p1+ddq2r*p2+Yp13*p3 -VF_Fv1*ev1;
}
else
{
    tau1=0.0;
    tau2=0.0;
}
}

```

```

tau1=(tau1>taulmax)?taulmax:tau1;
    tau1=(tau1<-taulmax)?-taulmax:tau1;
tau2=(tau2>tau2max)?tau2max:tau2;
tau2=(tau2<-tau2max)?-tau2max:tau2;

u1=(int) (tau1*t2c_1);
u2=(int) (tau2*t2c_2);

if (i==VI_PTS-1)
{
    r1=qq1;
    r2=qq2;
    VF_CYCNUM--;

    VF_abs_err1=abs_err1;
    VF_abs_err2=abs_err2;
    abs_err1=0;
    abs_err2=0;

    iSave=(int)VF_csaveflag;

    if (iSave==100)
    {
        iSave=(int)VF_controlflag1;
        VF_controlflag=iSave;
        k=0;
        VF_CYCNUM=7;
        iSave=(int)VF_csaveflag1;
        VF_csaveflag=iSave;
    }

    if (VF_CYCNUM>0) i=-1;
else
    i=VI_PTS+10;

```

```

    }

    i++;

    if(k<(unsigned long)0x07ff0)savedata();
}
else
{
    olderr1=err1;
    olderr2=err2;
    err1=c2r*(float)(qq1-r1);
    err2=c2r*(float)(qq2-r2);
    derr1=(err1-olderr1)/Ts;
    derr2=(err2-olderr2)/Ts;
    serr1+=err1*Ts;serr2+=err2*Ts;
    oldq1=qq1;oldq2=qq2;

    tmp1=(kp1*err1+kd1*derr1+ki1*serr1);
    tmp2=(kp2*err2+kd2*derr2+ki2*serr2);

    tmp1=(tmp1>taulmax)?taulmax:tmp1;
    tmp1=(tmp1<-taulmax)?-taulmax:tmp1;
    tmp2=(tmp2>tau2max)?tau2max:tmp2;
    tmp2=(tmp2<-tau2max)?-tau2max:tmp2;

    u1=-(int)(tmp1*t2c_1);
    u2=-(int)(tmp2*t2c_2);
}
}
else /*if(Host_Trigger==0.0)*/
{
    i=0;k=0;

    olderr1=err1;
    olderr2=err2;
    err1=c2r*(float)(qq1-ref1);

```

```

err2=c2r*(float)(qq2-ref2);
derr1=(err1-olderr1)/Ts;
derr2=(err2-olderr2)/Ts;
serr1+=err1*Ts;serr2+=err2*Ts;
oldq1=qq1;oldq2=qq2;

tmp1=(kp1*err1+kd1*derr1+ki1*serr1);
tmp2=(kp2*err2+kd2*derr2+ki2*serr2);

tmp1=(tmp1>taulmax)?taulmax:tmp1;
tmp1=(tmp1<-taulmax)?-taulmax:tmp1;
tmp2=(tmp2>tau2max)?tau2max:tmp2;
tmp2=(tmp2<-tau2max)?-tau2max:tmp2;

u1=-(int)(tmp1*t2c_1);
u2=-(int)(tmp2*t2c_2);
}
m1:
display();
}

savedata()
{
/*int iSave;
iSave=(int)VF_csaveflag;*/

switch(iSave)
{
case 0:
TF_output[k]=err1;      k++;
TF_output[k]=err2;      k++;
TF_output[k]=dq1;       k++;
TF_output[k]=dq2;       k++;
TF_output[k]=u1;        k++;
TF_output[k]=u2;        k++;

```

```

    TF_output [k]=VF_z01;      k++;
    TF_output [k]=VF_z02;      k++;
    TF_output [k]=VF_beta02;   k++;
    TF_output [k]=VF_beta12;   k++;
    TF_output [k]=VF_beta32;   k++;
    TF_output [k]=VF_beta01;   k++;
    TF_output [k]=VF_beta11;   k++;
    TF_output [k]=VF_beta31;   k++;
    break;
case 1:
    TF_output [k]=dq1;         k++;
    TF_output [k]=dq2;         k++;
    break;
case 2:
    TF_output [k]=VF_beta02;   k++;
    TF_output [k]=VF_beta12;   k++;
    TF_output [k]=VF_beta32;   k++;
    break;
case 3:
    TF_output [k]=ddq1;        k++;
    TF_output [k]=ddq2;        k++;
    break;
case 4:
    TF_output [k]=VF_beta01;   k++;
    TF_output [k]=VF_beta11;   k++;
    TF_output [k]=VF_beta31;   k++;
    break;
case 5:
    TF_output [k]=err1;        k++;
    TF_output [k]=err2;        k++;
    break;
case 6:
    TF_output [k]=err1;        k++;
    TF_output [k]=err2;        k++;
    TF_output [k]=dq1;         k++;
    TF_output [k]=dq2;         k++;

```



```

    TF_output [k]=u1;           k++;
    TF_output [k]=u2;           k++;
    TF_output [k]=VF_p1;       k++;
    TF_output [k]=VF_p2;       k++;
    TF_output [k]=VF_p3;       k++;
    TF_output [k]=VF_f1;       k++;
    TF_output [k]=VF_f2;       k++;
    TF_output [k]=VF_g1;       k++;
    TF_output [k]=VF_g2;       k++;
    break;
case 7:
    TF_output [k]=VF_p1;       k++;
    TF_output [k]=VF_p2;       k++;
    TF_output [k]=VF_p3;       k++;
    break;
case 8:
    TF_output [k]=VF_f1;       k++;
    TF_output [k]=VF_f2;       k++;
    break;
case 9:
    TF_output [k]=VF_g1;       k++;
    TF_output [k]=VF_g2;       k++;
    break;
case 10:
    TF_output [k]=VF_f1;       k++;
    TF_output [k]=VF_g1;       k++;
    break;
case 11:
    TF_output [k]=VF_f2;       k++;
    TF_output [k]=VF_g2;       k++;
    break;
case 12:
    TF_output [k]=VF_beta02;    k++;
    TF_output [k]=VF_beta12;    k++;
    break;
case 13:

```

```

        TF_output[k]=VF_beta32;          k++;
        TF_output[k]=VF_beta01;          k++;
        break;
    case 14:
        TF_output[k]=VF_beta11;          k++;
        TF_output[k]=VF_beta31;          k++;
        break;
    case 15:
        TF_output[k]=VF_p1;              k++;
        TF_output[k]=VF_p2;              k++;
        break;
    case 16:
        TF_output[k]=VF_p2;              k++;
        TF_output[k]=VF_p3;              k++;
        break;
    case 17:
        TF_output[k]=VF_z01;              k++;
        TF_output[k]=VF_z11;              k++;
        break;
    case 18:
        TF_output[k]=VF_z02;              k++;
        TF_output[k]=VF_z12;              k++;
        break;
    default:
        break;
}
}

```

```
display()
```

```

{
    /*int iDisplay;*/
    iDisplay=(int)VF_display;

    switch (iDisplay)
    {
        case 0:
            User1=(float)err1*50;

```

```
        User2=(float)err2*50;
        break;
case 1:
    User1=u1/t2c_1;
    User2=u2/t2c_2;
    break;
case 2:
    User1=VF_z02;
    User2=VF_z12;
    break;
case 3:
    User1=VF_z01;
    User2=VF_z11;
    break;
case 4:
    User1=VF_beta02;
    User2=VF_beta12;
    break;
case 5:
    User1=VF_beta02;
    User2=VF_beta32;
    break;
case 6:
    User1=VF_beta12;
    User2=VF_beta32;
    break;
case 7:
    User1=VF_beta01;
    User2=VF_beta11;
    break;
case 8:
    User1=VF_beta01;
    User2=VF_beta31;
    break;
case 9:
    User1=VF_beta11;
```

```
        User2=VF_beta31;
        break;
case 10:
    User1=VF_p1;
    User2=VF_p2;
    break;
case 11:
    User1=VF_p2;
    User2=VF_p3;
    break;
case 12:
    User1=VF_p1;
    User2=VF_p3;
    break;
case 13:
    User1=VF_f1;
    User2=VF_f2;
    break;
case 14:
    User1=VF_g1;
    User2=VF_g2;
    break;
case 15:
    User1=VF_f1;
    User2=VF_g1;
    break;
case 16:
    User1=VF_f2;
    User2=VF_g2;
    break;
case 17:
    User1=q1d;
    User2=q2d;
    break;
case 18:
    User1=dq1d;
```

```

        User2=dq2d;
        break;
    case 19:
        User1=ddq1d;
        User2=ddq2d;
        break;
    default:
        break;
    }
}

pd_part()
{
    tau1=-(kp1*err1+kd1*derr1);
    tau2=-(kp2*err2+kd2*derr2);
}

pd_control()
{
    u1=-(int)((kp1*err1+kd1*derr1)*t2c_1);
    u2=-(int)((kp2*err2+kd2*derr2)*t2c_2);
}

Common_part()
{
    dq1r=dq1d-rate_fv*(VF_lamap1*err1+VF_lamai1*serr1);
    dq2r=dq2d-rate_fv*(VF_lamap2*err2+VF_lamai2*serr2);

    ddq1r=ddq1d-rate_fv*(VF_lamap1*derr1+VF_lamai1*err1);
    ddq2r=ddq2d-rate_fv*(VF_lamap2*derr2+VF_lamai2*err2);

    ev1=dq1-dq1r;
    ev2=dq2-dq2r;

    Yp13=c2*(2.0*ddq1r+ddq2r)-(dq2*dq1r+(dq1+dq2)*dq2r)*s2;
    Yp11=ddq1r; Yp12=ddq2r;
}

```

```

    Yp21=0;   Yp22=ddq1r+ddq2r;   Yp23=c2*ddq1r+s2*dq1*dq1r;

}

LuGre_part ()
{
    varphi1=fabs(dq1)/(Fc1+(Fs1-Fc1)*exp(-dq1*dq1*inv_vs1));
    VF_z01=VF_z01+(dq1-varphi1*VF_z01-ev1)*Ts;
    VF_z11=VF_z11+(dq1-varphi1*(VF_z11-ev1))*Ts;
    VF_beta01=VF_beta01-(VF_gamma01*ev1*VF_z01)*Ts;
    VF_beta11=VF_beta11+(VF_gamma11*ev1*varphi1*VF_z11)*Ts;
    VF_beta31=VF_beta31-(VF_gamma31*ev1*dq1r)*Ts;

    varphi2=fabs(dq2)/(Fc2+(Fs2-Fc2)*exp(-dq2*dq2*inv_vs2));
    VF_z02=VF_z02+(dq2-varphi2*VF_z02-ev2)*Ts;
    VF_z12=VF_z12+(dq2-varphi2*(VF_z12-ev2))*Ts;
    VF_beta02=VF_beta02-(VF_gamma02*ev2*VF_z02)*Ts;
    VF_beta12=VF_beta12+(VF_gamma12*ev2*varphi2*VF_z12)*Ts;
    VF_beta32=VF_beta32-(VF_gamma32*ev2*dq2r)*Ts;

}

adaptive_part ()
{
    tmp3=VF_gAmmap1*Yp11*ev1*Ts;
    VF_p1=VF_p1-tmp3;

    tmp3=VF_gAmmap2*(Yp12*ev1+Yp22*ev2)*Ts;
    VF_p2=VF_p2-tmp3;

    tmp3=VF_gAmmap3*(Yp13*ev1+Yp23*ev2)*Ts;
    VF_p3=VF_p3-tmp3;

    tmp3=VF_gAmmaf1*dq1r*ev1*Ts;
    VF_f1=VF_f1-tmp3;
}

```

```

tmp3=VF_gAmmaf2*dq2r*ev2*Ts;
VF_f2=VF_f2-tmp3;

if(dq1r>0)tmp3=VF_gAmmag1*ev1*Ts;
else
    tmp3=-VF_gAmmag1*ev1*Ts;
    VF_g1=VF_g1-tmp3;

if(dq2r>0)tmp3=VF_gAmmag2*ev2*Ts;
else
    tmp3=-VF_gAmmag2*ev2*Ts;

VF_g2=VF_g2-tmp3;
)

```

## A.2 Rpl.c

```

#include "RPL.H"
#include "math.h"
#include "address.c"
#include "conio.h"

#define CYC_NUM 1
#define PI2 6.2831854
#define P5000 2501

extern void far PromptString(char far*,char far *,char far*);

char cNL=0x20;
float ft_data[6];
int ft_ready=0;
    float FREQ1=0.5, AMP1=0.6;
    float FREQ2=0.5, AMP2=0.6;

float c2r=0.00004091,t2c_1=8.3551,t2c_2=51.75;
float pi=3.1415927;

```

```

float v2c=204.8;    /*volts to D/A coun conv.*/

int nPts=2000;    /* 2000;*/
int nStayPts=00; /* stay at one point for nStayPts*Ts time */

int CYC_NUM_forsave=0;
float l1=14.5*.0254; /*=14.5 inch    length of base arm*/
float l2=9.5*.0254; /*=9.5; inch    length of the elbow*/

/* check it after compiling the corresponding servo program */
unsigned long addr_ft=0x00030126; /* address of ft_data */
unsigned long li,lj;
float c2n=8,c2nm=8;    counts to Newton and Newton Meter */

float uu1=0,uu2=0;
char *str1="";

int ptnum;
/*float aa1[300],aa2[300],aa3[300];*/

float VFcontrolflag=0;
float tf_input;
float tmp;
float tmp1,tmp2;

float rr=5*.0254;
float xx0=19*.0254, yy0=0;

float ftmp[80];

float VFu1=0,VFu2=0,VFfreq=0,VFexp=0;
double lasttheta1,lasttheta2;

#include <dos.h>

```



```

#define FALSE          0
#define TRUE           1

#define Ts 0.002

unsigned addr_reg      = 0x280;
unsigned data_reg     = 0x282;
unsigned status_reg   = 0x284;
unsigned irq_num      = 7;
unsigned data_available = FALSE;
unsigned interrupt_mode = FALSE;

int send_command(unsigned *bptr, int numwords)
{
    unsigned dword, command, retcode, loop ;

    command = *bptr;
    outpw(addr_reg,0);
    for(;numwords;numwords--)
    {
        dword = *bptr ;
        outpw(data_reg,dword) ;
        bptr ++ ;
    }
    outpw(addr_reg,0x3FE);
    inpw(data_reg);

    outpw(addr_reg,0x3FE) ;
    outpw(data_reg,command);
    outpw(addr_reg,0x3ff) ;
    outpw(data_reg,command) ;

    retcode = TRUE ;
    for(loop=TRUE;loop;)
    {

```

```

        dword = inpw(status_reg);

        if((dword & 0x0010)==0)
            {
                outpw(addr_reg,0x3FE) ;
                retcode = inpw(data_reg) ;

                if(retcode & 0x00ff) loop=FALSE;
            }
        }
    return (retcode);
}

void receive_command(unsigned *bptr, int numword)
{
    unsigned dword;

    outpw(addr_reg,0);
    for(;numword;numword--)
        {
            dword = inpw(data_reg);
            *bptr = dword ;
            bptr ++ ;
        }
}

struct
{
    unsigned          cmd_rc;
    unsigned          mode;
    unsigned          matrix[96];
    unsigned          sbias[8];
    signed            Mm[8];
    signed            Bm[8];
    signed            Mb[8];
}

```

```

signed            Bb[8];
unsigned          number_of_gages;
unsigned          counts_per_force;
unsigned          counts_per_torque;
unsigned          max_force;
unsigned          max_torque;
unsigned          maxtemperature;
unsigned          min_temperature;
unsigned          units;
unsigned          serial_number;
} sensor_data;

int read_sensor_info(void)
{
    sensor_data.cmd_rc = 0x0100 ;
    sensor_data.mode   = 0x0000 ;
    if(!send_command(&sensor_data.cmd_rc,2)) return(FALSE);
    receive_command(&sensor_data.cmd_rc,sizeof(sensor_data));
    c2n=sensor_data.counts_per_force*0.1;
    c2nm=sensor_data.counts_per_torque*0.1;
    return (TRUE);
}

struct
{
    unsigned      cmd_rc;
    unsigned      mode;
    unsigned      Data_C_flag;
    unsigned      Data_Mask;
    unsigned      Enable_Inter;
    unsigned      Sample_Rate;
    unsigned      Card_Model;
    unsigned      Ucode;
    unsigned      serial_number;
} card_data;

```

```

int change_card_data(void)
{
    card_data.cmd_rc = 0x0200 ;
    card_data.mode   = 0x0000 ;
    if(!send_command(&card_data.cmd_rc,2)) return(FALSE);
    receive_command(&card_data.cmd_rc,sizeof(card_data)) ;
    card_data.Sample_Rate=250;
    card_data.Data_C_flag
    =(card_data.Data_C_flag&0xffff8)|0x0003; /* set filter */
    card_data.cmd_rc=0x0200;
    card_data.mode=0x0001;
    if(!send_command(&card_data.cmd_rc,sizeof(card_data)))
        return(FALSE);
    return (TRUE);
}

```

```

struct
{
    unsigned      cmd_rc;
    unsigned      mode;
    signed        Bias_Array[8];
} bias_array;

```

```

int change_bias_array(void)
{
    bias_array.cmd_rc = 0x0300 ;
    bias_array.mode   = 0x0001 ;
    if(!send_command(&bias_array.cmd_rc,2)) return(FALSE);
    return (TRUE);
}

```

```

int read_ft_poll(void)
{

```

```

int    loop, i;
int    dword ;
int    force_counts[3], torque_counts[3];

outpw(addr_reg,0x100);
dword = inpw(data_reg);
if(!dword) return(0);

dword = inpw(data_reg);
if(dword)
{
    outpw(addr_reg,0x101);
    outpw(data_reg,0x00);
}

outpw(addr_reg,0x0101);
outpw(data_reg,0x00);
for(i=0;i<3;i++) force_counts[i] = inpw(data_reg);
for(i=0;i<3;i++) torque_counts[i]= inpw(data_reg);

outpw(addr_reg,0x100);
outpw(data_reg,0x00);

for(i=0;i<3;i++)
{
    ft_data[i]=force_counts[i]/c2n;
    ft_data[i+3]=torque_counts[i]/c2nm;
}

return(1);
}

```

```

Upload_Table1(char *astr)
{
    FILE *fn, *fn1;
    float tf1,tf2,tf3,tf4,tf5,tf6,tf7,tf8,tf9;

```

```

float tf10,tf11,tf12,tf13,tf14,tf15,tf16;
int i,j;

fn=fopen("c:\\robot\\zhu\\aaaref.c","w");

if(*astr==0)fn1=fopen("c:\\robot\\zhu\\aaact.c","w");
else
    fn1=fopen(astr,"w");

printf("%c[save to: %s] %c",13,astr,cNL);
printf("%c[# of data: %g]          %c",13,
        (float)CYC_NUM_forsave*(nPts+nStayPts),cNL);

if((fn1!=NULL)&&(fn!=NULL))
{
    lj=addr_TF_output;

    for(li=0;li<(unsigned long)6*(nPts-1);li++)
    {
        /*lj=li*(unsigned long)13;*/
        if(li>0x7fff)break;
        tmp1=GetFloat(addr_VF_csaveflag,DUAL);
        if(tmp1==0.0)
        {
            tf1=GetFloat((unsigned long)lj,DUAL);
            lj+=(unsigned long)1;/*zdelay(); err1*/
            tf2=GetFloat((unsigned long)lj,DUAL);
            lj+=(unsigned long)1;/*zdelay(); err2*/
            tf3=GetFloat((unsigned long)lj,DUAL);
            lj+=(unsigned long)1;/*zdelay(); dq1*/
            tf4=GetFloat((unsigned long)lj,DUAL);
            lj+=(unsigned long)1;/*zdelay(); dq2*/
            tf5=GetFloat((unsigned long)lj,DUAL);
            lj+=(unsigned long)1;/*zdelay(); u1*/
            tf6=GetFloat((unsigned long)lj,DUAL);
            lj+=(unsigned long)1;/*zdelay(); u2*/
            tf7=GetFloat((unsigned long)lj,DUAL);
            lj+=(unsigned long)1;/*zdelay(); z01*/

```

```

tf8=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); z02*/
tf9=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); beta02*/
tf10=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); beta12*/
tf11=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); beta32*/
tf12=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); beta01*/
tf13=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); beta11*/
if(tmp1==0.0)
(
tf14=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); beta31*/
fprintf(fn1,"%g %g %g %g %g %g %g %g %g %g %g %g
%g %g %g\n", (float)li*Ts,tf1,tf2,tf3,tf4,tf5,tf6,
tf7/t2c_1,tf8/t2c_2,tf9,tf10,tf11,tf12,tf13,tf14);
)
else
(
fprintf(fn1,"%g %g %g %g %g %g %g %g %g %g %g %g
%g %g\n", (float)li*Ts,tf1,tf2,tf3,tf4,tf5,tf6,
tf7/t2c_1,tf8/t2c_2,tf9,tf10,tf11,tf12,tf13);
)
}
else if((tmp1==1.0)|| (tmp1==3.0)|| (tmp1==5.0)||
((tmp1>7.0)&&(tmp1<18.1)) )
(
tf1=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); err2*/
tf2=GetFloat((unsigned long)lj,DUAL);
lj+=(unsigned long)1;/*zdelay(); dq2*/
fprintf(fn1,"%g %g %g\n", (float)li*Ts,tf1,tf2);
)
else if((tmp1==2.0)|| (tmp1==4.0)|| (tmp1==7.0))

```

```

        {
            tf1=GetFloat((unsigned long)lj, DUAL);
            lj+=(unsigned long)1;
            tf2=GetFloat((unsigned long)lj, DUAL);
            lj+=(unsigned long)1;
            tf3=GetFloat((unsigned long)lj, DUAL);
            lj+=(unsigned long)1;
            fprintf(fn1, "%g %g %g %g\n", (float)li*Ts, tf1, tf2, tf3);
        }
    }

    for(i=0; i<(nPts+nStayPts); i++)
    {
        tf1=GetFloat(addr_TF_q1+i, DUAL);
        tf2=GetFloat(addr_TF_q2+i, DUAL);
        tf3=GetFloat(addr_TF_dq1+i, DUAL);
        tf4=GetFloat(addr_TF_dq2+i, DUAL);
        tf5=GetFloat(addr_TF_ddq1+i, DUAL);
        tf6=GetFloat(addr_TF_ddq2+i, DUAL);
        fprintf(fn, "%g %g %g %g %g %g %g\n",
            (float)i*Ts, tf1, tf2, tf3, tf4, tf5, tf6);
    }
}

fclose(fn);
fclose(fn1);
}

set_current_ref()
{
    ref1=pos1;
    ref2=pos2;
}

```



```

set_display_variable()
{
    /* Define which to be plotted in real time */
    draw_base=0;
    draw_elbow=0;
    draw_base_ref=0;
    draw_elbow_ref=0;
    draw_err1=0;
    draw_err2=0;
    draw_base_vel=1;
    draw_elbow_vel=1;
    draw_user1=1;
    draw_user2=0;
}

set_display_range
(double xmin,double xmax,double ymin, double ymax)
{
    x_min=xmin;
    x_max=xmax;
    y_min=ymin;
    y_max=ymax;
}

delay(int i)
{
    while(i)
    {
        i=i-1;
        i=i/1;
        i=i/1;
    }
}

int zWait_For(seconds)

```

```

double seconds;
{
    float aa,bb,cc;
    float aal,bb1,cc1;
    double final_time;
    char oldcom = 0;

    setpoints(1);
    if(Simulation == 1)init_simu(param_set1, cntrl_var1);

    _disable();
    final_time = timer + seconds;
    _enable();
    for(;;)
    {
        _disable();
        if(Simulation == 1) Do_Sim();
        art();

        $command = char_in();

        aal=GetFloat(addr_VF_p1,DUAL);
        bb1=GetFloat(addr_VF_p2,DUAL);
        cc1=GetFloat(addr_VF_p3,DUAL);
        aa=GetFloat(addr_VF_abs_err1,DUAL);
        bb=GetFloat(addr_VF_abs_err2,DUAL);
        cc=GetFloat(addr_VF_CYCNUM,DUAL);

        if(seconds<0.0)printf("%c %g %d[p1:% 6.4f p2:
        % 6.4f p3:% 6.4f e1:% 6.4f e2:%6.4f] %c",13,cc,
        $command,aal,bb1,cc1,aa,bb,cNL);

        if($command == 91) /* left windows key to exit*/
        { PutFloat(addr_VF_controlflag, DUAL, (float)100);
          Slave_Trigger(0.0);
          refl=pos1;ref2=pos2;

```

```

        bye();
    }
else if($command==92) /*stay at current point*/
{
    PutFloat(addr_VF_controlflag,DUAL,(float)98);
    Slave_Trigger(0.0);
    ref1=pos1;ref2=pos2;
    bye();
}
else if($command==93) /*goto zero point*/
{
    ref1=0;ref2=0;
    PutFloat(addr_VF_controlflag,DUAL,(float)97);
    Slave_Trigger(0.0);
    ref1=0;ref2=0;
    bye();
}
else if($command==88) /*F12 save data*/
{
/*          aa=GetFloat(addr_VF_csaveflag,DUAL);
    delay(10);
    PutFloat(addr_VF_controlflag1,DUAL,(float)(aa));
    delay(10);*/
    PutFloat(addr_VF_csaveflag,DUAL,(float)(100));
}
else if(($command>58)&&($command<69))
{
    PutFloat(addr_VF_controlflag,DUAL,
        (float)($command-58));
}

if(seconds==7.0) /*&&VFcontrolflag==7.0)*/
{
    ref1=pos1;ref2=pos2;
    if($command==92)goto zwait1;
}

```

```

        if((fabs(tmp2)<0.04)
            &&(fabs(tmp1)<0.05 ))goto zwait1;
    }

    if($command)oldcom = $command;
    if{(timer >= final_time)&&(seconds>=0.0))break;

    if{(seconds== -1.0)&&(cc<=0))break;

    _enable();
}
zwait1:
    _enable();
    if(oldcom) $command = oldcom;
    else return(1);
}

```

```

go_point()
{
    /* back to (VF_q1, VF_q2) */
    PutFloat(addr_VF_controlflag,DUAL,7);
    zWait_For(-7.0);
    zWait_For(1.0);
    set_current_ref();
}

```

```

read_traj_from_file()
{
    float a1,a2,a3,a4,a5,a6;
    int i;
    FILE *fn;
    _disable();
    fn=fopen("c:\\\\robot\\zhu\\traj.tra","r");
}

```

```

for (i=0; i<P5000;i++)
{
    if(!feof(fn))
    {
        fscanf(fn,"%g%g%g%g%g%g",&a1,&a2,&a3,&a4,&a5,&a6);
        PutFloat(addr_TF_q1+i,DUAL,a1);
        PutFloat(addr_TF_q2+i,DUAL,a2);
        PutFloat(addr_TF_dq1+i,DUAL,a3);
        PutFloat(addr_TF_dq2+i,DUAL,a4);
        PutFloat(addr_TF_ddq1+i,DUAL,a5);
        PutFloat(addr_TF_ddq2+i,DUAL,a6);
    }
}
fclose(fn);
_enable();
}

```

```
read_data_from_file()
```

```

{ FILE *fn;
_disable();
fn=fopen("c:\\robot\\zhu\\aaa.c","r");
fscanf(fn,"%f%f%f%f%f%f%&s",&AMP1,&FREQ1,
&AMP2,&FREQ2,&tmp1,&tmp2,str1);
printf("%c[Amp1:%g Freq1:%g Amp2:%g
Freq2:%g ctr:%g save:%g saveto:%s] %c",13,
AMP1,FREQ1,AMP2,FREQ2,tmp1,tmp2,str1,cNL);
fclose(fn);
_enable();
PutFloat(addr_VF_controlflag1,DUAL,tmp1);
PutFloat(addr_VF_csaveflag1,DUAL,tmp2);
}

```

```
Download_sin()
```

```

{ FILE *fn;
int i;
float tt;

```

```

float a1=0,a2=0,aldot=0,a2dot=0,a12dot=0,a22dot=0;

zWait_for(2.0);

if(FREQ1<FREQ2)tmp1=FREQ1;
else
    tmp1=FREQ2;
nPts=1.0/(tmp1*Ts);

_disable();
fn=fopen("c:\\robot\\zhu\\traj.tra","w");
/*for(i=0;i<nPts+nStayPts;i++)*/
for(i=0;i<P5000;i++)
/*    for(i=0;i<5000;i++)*/
{
    tt=(float)i*Ts;
    if(i<nPts)
    {
        a1=AMP1*sin(PI2*FREQ1*tt);
        a2=AMP2*sin(PI2*FREQ2*tt);
        aldot=PI2*FREQ1*AMP1*cos(PI2*FREQ1*tt);
        a2dot=PI2*FREQ2*AMP2*cos(PI2*FREQ2*tt);
        a12dot=-PI2*FREQ1*AMP1*PI2*FREQ1*sin(PI2*FREQ1*tt);
        a22dot=-PI2*FREQ2*AMP2*PI2*FREQ2*sin(PI2*FREQ2*tt);
    }
    fprintf(fn,"%g %g %g %g %g %g \n",
        a1,a2,aldot,a2dot,a12dot,a22dot);
}
fclose(fn);
_enable();
}

read_parameter()
{
    char *stra1="";
    char *stra2="";

```

```

unsigned long li1,li2;
FILE *fn;
_disable();

fn=fopen("c:\\robot\\zhu\\aaact.c","w");
fclose(fn);

fn=fopen("c:\\robot\\zhu\\addresss.c","r");
while(!feof(fn))
{
    fscanf(fn,"%s",stral);
    if(strchr(stral,'_'))strcpy(stra2,stral);
    if(strstr(stral,"="))
    {
        fseek(fn,3,SEEK_CUR);
        fscanf(fn,"%04x",&li1);
        fscanf(fn,"%04x",&li2);

        li1=(li1<<16)+(li2&0xffff);

        fscanf(fn,"%s",stral);

        if(!feof(fn))
        {
            fscanf(fn,"%s",stral);
            if(!strstr(stral,"unsigned"))
            {
                tmp1=atof(stral);
                printf("%c [%s=%g    ]%c",13,stra2,tmp1,cNL);
                PutFloat(li1,DUAL,tmp1);
                zWait_for(0.1);
            }
        }
    }
}
fclose(fn);

```

```

_enable();

}

test()
{
    Slave_Trigger(0.0);
    Torque_Mode();

    PutFloat(addr_VF_controlflag,DUAL,(float)99);
    zWait_for(0.1);

    go_point();

    read_parameter();
    read_data_from_file();

/* download_sin();*/

    read_traj_from_file();

    PutInt(addr_VI_PTS,DUAL,nPts+nStayPts);

    tmp=GetFloat(addr_VF_CYCNUM,DUAL);
    if(tmp<1)tmp=CYC_NUM;
    CYC_NUM_forsave=(int)tmp;
    PutFloat(addr_VF_CYCNUM,DUAL,tmp);

    Slave_Trigger(1.0);
    zWait_For(-1.0);
    set_current_ref();

    Slave_Trigger(0.0); Wait_For(.5);
}

```



```
RPL()
{

    float stepa;
    float x,y;
    float theta;

    int i;
    int j;

    /* Define which to be plotted in real time */

    /* initiation */
    Sample_Time((double)Ts);
    test();
    _disable();
    Upload_Table1(str1);
    _enable();

}
```

## APPENDIX B

### MATLAB Programs

#### B.1 Pddiff.m

```
function QDOT=pddiff(t,p) global l1 global l2 global tf global r
xc yc global phic dphic ddp hic global num1 global tempdata global
ttt global kp kd global p1 p2 p3 global pp1 pp2 pp3

q=[p(1);p(2)];      %angle
dq=[p(3);p(4)];    %angular velocities

if t>tf/2
    p1=pp1;
    p2=pp2;
    p3=pp3;
end

q1=p(1);q2=p(2);dq1=p(3);dq2=p(4);

phi=phic*[1;t;t^2;t^3;t^4;t^5;t^6;t^7];
dphi=dphic*[1;2*t;3*t^2;4*t^3;5*t^4;6*t^5;7*t^6];
ddphi=ddp hic*[2;6*t;12*t^2;20*t^3;30*t^4;42*t^5];

x=xc+r*cos(phi); y=yc+r*sin(phi); dx=-r*sin(phi)*dphi;
dy=r*cos(phi)*dphi; ddx=-r*sin(phi)*ddphi-r*cos(phi)*dphi*dphi;
ddy=r*cos(phi)*ddphi-r*sin(phi)*dphi*dphi;

%plot(t,x,'b',t,y,'r',t,dx,'g',t,dy,'b',t,ddx,'r',t,ddy,'g');

%% plot the desired trajectory
%subplot(2,1,1);
%plot(x,y,'--');
```

```
d=(x^2+y^2-l1^2-l2^2)/(2*l1*l2); q2d=acos(d); q1d=atan(y/x) atan(
(l2*sin(q2d))/(l1+l2*cos(q2d)) );
```

```
qd=[q1d;q2d];
```

```
c1=cos(q1d);c2=cos(q2d);c12=cos(q1d+q2d)
;s1=sin(q1d);s2=sin(q2d);s12=sin(q1d+q2d);
```

```
j1=[l2*c12 l2*s12]; j2=[-l1*c1-l2*c12 -l1*s1-l2*s12];
```

```
xv=[dx;dy];
```

```
jinv=[j1;j2]/(l1*l2*s2);
```

```
dqd=jinv*xv; dq1d=dqd(1); dq2d=dqd(2);
```

```
ddq1d=(l1*c2*dq2d*(c12*dx+s12*dy)-l1*s2*(-s12*dx*(dq1d+dq2d)
+c12*dx+c12*dy*(q1d+q2d)+s12*ddy))/(l1^2*s2^2);
```

```
ddq2d=(l1*l2*c2*dq2d*((l1*c1+l2*c12)*dx+(l1*s1+l2*s12)*dy)
-l1*l2*s2*((l1*c1+l2*c12)*ddx+(-l1*s1*dq1d
-l2*s12*(dq1d+dq2d))*dx+(l1*s1+l2*s12)*ddy
+(l1*c1*dq1d+l2*c12*(dq1d+dq2d))*dy )/((l1*l2*s2)^2);
```

```
ddqd=[ddq1d;ddq2d];
```

```
%%%%%%%%%% m and c matrix
```

```
m=[p1+2*p3*cos(q2) p2+p3*cos(q2);p2+p3*cos(q2) p2];
```

```
c=[-p3*dq2*sin(q2) -p3*(dq1+dq2)*sin(q2); p3*dq1*sin(q2) 0];
```

```
%%%%%%%%%% Control
```

```
e=qd-q;de=dqd-dq; u=kp*e+kd*de;
```

```
%%%%%%%%%% state space equation
```

```
qdot=dq; dqdot=inv(m)*(u-c*dq);
```

```

%if ttt~=t
    ttt=t;
    num1=num1+1;
    tempdata(num1,1)=t;
    tempdata(num1,2)=dqdot(1);tempdata(num1,3)=dqdot(2);
    tempdata(num1,4)=u(1);tempdata(num1,5)=u(2);
%end

```

```

%%% plot the angular error
%subplot(2,1,2);plot(t,-q(1)+q1d,'-',t,-q(2)+q2d,'--')
%%%%%%%%% function return
QDOT=[qdot;dqdot];

```

## B.2 Pdcontrolload.m

```

global l1 global l2 global tf global r xc yc global phic dphic
ddphic global num1 global tempdata global ttt global kp kd global
p1 p2 p3 global pp1 pp2 pp3

```

```

ttt=-1; %used for synchronize the time in diff equ.
l1=14.5*.0254; %.38;
l2=9.5*.0254; %.24;
r=4*.0254;xc=19*.0254;yc=0; p1=3.4;p2=.4;p3=.3;
mp=50;pp1=p1+mp*(l1^2+l2^2);pp2=p2+mp*l2^2;
pp3=p3+mp*l1*l2;%payload =50(N)
kp=[1200 0;0 600]; kd=[100 0;0 30]; tf=4;

```

```

l1=.38; l2=.24; r=.1;xc=0.48;yc=0;

```

```

num1=0;

```

```

A=[1 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0;
    0 0 2 0 0 0 0 0;
    0 0 0 6 0 0 0 0;

```

```

1 tf tf^2 tf^3 tf^4 tf^5 tf^6 tf^7;
0 1 2*tf 3*tf^2 4*tf^3 5*tf^4 6*tf^5 7*tf^6;
0 0 2 6*tf 12*tf^2 20*tf^3 30*tf^4 42*tf^5;
0 0 0 6 24*tf 60*tf^2 120*tf^3 210*tf^4];
phi0=[0;0;0;0;2*pi;0;0;0]; a=inv(A)*phi0; phic=a' dphic=[phic(2)
phic(3) phic(4) phic(5) phic(6) phic(7) phic(8)] ddp hic=[phic(3)
phic(4) phic(5) phic(6) phic(7) phic(8)]

x=xc+r;y=yc; d=(x^2+y^2-l1^2-l2^2)/(2*l1*l2); q2=acos(d);
q1=atan(y/x)-atan( (l2*sin(q2))/(l1+l2*cos(q2)) ); x0=[q1;q2;0;0];
%x0=[q1+.0;q2+.1;0;0] %position uncertainty

T=0:0.002:tf; options = odeset('RelTol',1e-3);
[t,x]=ode23('pddiff',T,x0,options);

close all
%Output equations
q1=x(:,1); q2=x(:,2); q1dot=x(:,3); q2dot=x(:,4);

% tempdata(num1,1)=t;
% tempdata(num1,2)=dqdot(1);tempdata(num1,3)=dqdot(2);
% tempdata(num1,4)=u(1);tempdata(num1,5)=u(2);
tt=tempdata(:,1);

trajref; %get desired trajectory

xx=l1*cos(q1)+l2*cos(q1+q2); yy=l1*sin(q1)+l2*sin(q1+q2);

%%% Plot the payload and angular error
%set(fig,'Position',[0 60 800 660]);
subplot(2,1,1); plot([0;tf/2;tf/2+.001;tf],[0;0;mp;mp]); axis([0 4
0 mp]); grid on; title('Payload (PD control)'); ylabel('Payload Mp
(N)'); subplot(2,1,2);plot(t,q1-qd(:,1),'-',t,q2-qd(:,2),'--');
grid on;xlabel('t (sec)'); ylabel('q_1-q_1_d, q_2-q_2_d (rad)');
title('Angular Error(PD control)'); legend('Err1','Err2',0); print

```

```
c:\zhu\robotics\payload_err_pd.eps -dps
```

### B.3 Adptvpayload1.m

```
global l1 global l2 global p1 p2 p3 global tf global r xc yc
global gama fv tau global phic dphic ddp hic global num1 global
tempdata global f1 f2 g1 g2 fg tau fg global pp1 pp2 pp3

f1=.05; f2=.05; g1=.01; g2=.01; fg=[f1;f2;g1;g2];
%fg=[0;0;0;0];
taufg=1.0*eye(4);

l1=.38; l2=.24; p1=3.4;p2=.4;p3=.3;
%mp=50;pp1=p1;pp2=p2;pp3=p3; % payload =50 (N)
tf=4; r=.1;xc=0.48;yc=0; gama=eye(2)*100; fv=200*eye(2);
tau=1.0*eye(3); num1=0;
mp=50;pp1=p1+mp*(l1^2+l2^2);pp2=p2+mp*l2^2;
pp3=p3+mp*l1*l2; % payload =50 (N)

A=[1 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0;
    0 0 2 0 0 0 0 0;
    0 0 0 6 0 0 0 0;
    1 tf tf^2 tf^3 tf^4 tf^5 tf^6 tf^7;
    0 1 2*tf 3*tf^2 4*tf^3 5*tf^4 6*tf^5 7*tf^6;
    0 0 2 6*tf 12*tf^2 20*tf^3 30*tf^4 42*tf^5;
    0 0 0 6 24*tf 60*tf^2 120*tf^3 210*tf^4];
phi0=[0;0;0;0;2*pi;0;0;0]; a=inv(A)*phi0; phic=a' dphic=[phic(2)
phic(3) phic(4) phic(5) phic(6) phic(7) phic(8)] ddp hic=[phic(3)
phic(4) phic(5) phic(6) phic(7) phic(8)]

close all;figure;subplot(2,1,1);hold on;subplot(2,1,2);hold on;

xr0=[xc+r;yc]; dxr0=[0;0];
```

```

x=xc+r;y=yc; d=(x^2+y^2-l1^2-l2^2)/(2*l1*l2); q2=acos(d);
q1=atan(y/x)-atan( (l2*sin(q2))/(l1+l2*cos(q2)) );
x0=[q1;q2;dxr0;p1/2;p2/2;p3/2;0;0;0;0];
%x0=[q1;q2;dxr0;0;0;0;0;0;0;0]

T=0:0.004:tf; options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4
1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-4]);
[t,x]=ode23('adptvf1diff',T,x0,options);

%Output equations
q1=x(:,1); q2=x(:,2); qldot=x(:,3); q2dot=x(:,4);

xx=l1*cos(q1)+l2*cos(q1+q2); yy=l1*sin(q1)+l2*sin(q1+q2);
%dfgsdfg

%%% Plot the payload and angular error
subplot(2,1,1); plot([0;tf/2;tf/2+.001;tf],[0;0;mp;mp]); axis([0 4
0 mp]); grid on; title('Payload (Adaptive control)');
ylabel('Payload Mp (N)');
%subplot(2,1,2);plot(t,q1-qd(:,1),'-',t,q2-qd(:,2),'--');
grid on;xlabel('t (sec)'); ylabel('q_1-q_1_d, q_2-q_2_d (rad)');
title('Angular Error(Adaptive control)'); legend('Err1','Err2',0);
print c:\zhu\robotics\payload_err_adptv.eps -dps

```

#### B.4 Adptvf1.m

```

global l1 global l2 global p1 p2 p3 global tf global r xc yc
global gama fv tau global phic dphic ddphic global num1 global
tempdata global f1 f2 g1 g2 fg taufg global pp1 pp2 pp3

f1=.05; f2=.05; g1=.01; g2=.01; fg=[f1;f2;g1;g2];
%fg=[0;0;0;0];
taufg=1.0*eye(4);

l1=.38; l2=.24; p1=3.4;p2=.4;p3=.3;

```

```

mp=50;pp1=p1;pp2=p2;pp3=p3; % payload =50(N)
tf=4; r=.1;xc=0.48;yc=0; gama=eye(2)*100; fv=200*eye(2);
tau=1.0*eye(3); num1=0;
%mp=10;pp1=p1+mp*(l1^2+l2^2);pp2=p2+mp*l2^2;
pp3=p3+mp*l1*l2; % payload =4(N)

A=[1 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0;
    0 0 2 0 0 0 0 0;
    0 0 0 6 0 0 0 0;
    1 tf tf^2 tf^3 tf^4 tf^5 tf^6 tf^7;
    0 1 2*tf 3*tf^2 4*tf^3 5*tf^4 6*tf^5 7*tf^6;
    0 0 2 6*tf 12*tf^2 20*tf^3 30*tf^4 42*tf^5;
    0 0 0 6 24*tf 60*tf^2 120*tf^3 210*tf^4];
phi0=[0;0;0;0;2*pi;0;0;0]; a=inv(A)*phi0; phic=a' dphic=[phic(2)
phic(3) phic(4) phic(5) phic(6) phic(7) phic(8)] ddphic=[phic(3)
phic(4) phic(5) phic(6) phic(7) phic(8)]

close all; fig=figure set(fig,'Position',[0 60 800 660]);
subplot(2,1,1);hold on;subplot(2,1,2);hold on;

xr0=[xc+r;yc]; dxr0=[0;0];

x=xc+r;y=yc; d=(x^2+y^2-l1^2-l2^2)/(2*l1*l2); q2=acos(d);
q1=atan(y/x)-atan( (l2*sin(q2))/(l1+l2*cos(q2)) );
x0=[q1;q2;dxr0;p1/2;p2/2;p3/2;0;0;0;0];
%x0=[q1;q2;dxr0;0;0;0;0;0;0;0]

T=0:0.004:tf; options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4
1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-4]);
[t,x]=ode23('adptvflldiff',T,x0,options);

```



```

%fhfdghdfh
%Output equations
q1=x(:,1); q2=x(:,2); q1dot=x(:,3); q2dot=x(:,4);

xx=l1*cos(q1)+l2*cos(q1+q2); yy=l1*sin(q1)+l2*sin(q1+q2);
%dfgsdfg

%plot the actual trajectory;
subplot(2,1,1);
plot(tempdata(:,1),tempdata(:,2),'-',xx,yy,'--');zoom on; grid
on;legend('Desired','Generated',0);title('Circle Trajectory
(Adaptive control)'); xlabel('x (m)');ylabel('y (m)');

%plot the angular errors
subplot(2,1,2);
%plot(t,tempdata(:,3)-q1,'-',t,tempdata(:,4)-q2,'--')
grid on;legend('Error (q_1_d-q_1)','Error
(q_2_d-q_2)',0);title('Angular Error Signals'); xlabel('t
(sec)');ylabel('q_1_d-q_1, q_2_d-q_2 (rad)'); print
c:\zhu\robotics\pos_err_adptv.eps -dps

%figure

%subplot(2,1,1);plot(t,q1,'-',t,q2,'--');zoom on;
%grid on;legend('q_1','q_2');title('Angule (Adaptive control)');
%xlabel('t (sec)');ylabel('q_1, q_2 (rad)');

%subplot(2,1,2);
%plot(t,q1dot,'-',t,q2dot,'--');
%zoom on;
%grid on;legend('q_1dot','q_2dot');
title('Angular Velocities (Adaptive control)');
%xlabel('t (sec)');ylabel('q_1dot, q_2dot (rad/s)');

```

```

figure plot(t,x(:,5),'-',t,x(:,6),'--',t,x(:,7),'.-');zoom on;

%plot(t,x(:,5)+p1,'-',t,x(:,6)+p2,'--',t,x(:,7)+p3,'.-');zoom on;
grid on;legend('p1hat','p2hat','p3hat',0);title('Estimated
Parameters (Adaptive Control)'); xlabel('t (sec)');ylabel('p1hat,
p2hat, p3hat'); print c:\zhu\robotics\phat_adptv.eps -dps

figure
plot(t,x(:,8),'-',t,x(:,9),'--',t,x(:,10),'.-',t,x(:,11),'.'');
zoom on;

%plot(t,x(:,5)+p1,'-',t,x(:,6)+p2,'--',t,x(:,7)+p3,'.-');zoom on;
grid on;legend('f1hat','f2hat','g1hat','g2hat',0);title('Estimated
Parameters (Adaptive Control)'); xlabel('t (sec)');ylabel('f1hat,
f2hat, g1hat, g2hat'); print c:\zhu\robotics\fg_hat_adptv.eps -dps

```

## B.5 Lugre.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Friction compensation using LuGre friction model %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global l1 global l2
global p1p2 p3
global tf
global r xc yc
global gama fv tau
global phic dphic ddp hic
global num1

global sigma01 sigma02
global sigma11 sigma12
global sigma21 sigma22

global Fs Fc vs

```

```

global r01 r02 r11 r12 r31 r32
global kp kd ki serr

kp=[1200 0;0 600]; kd=[100 0;0 30]; serr=0;

r01=5; r02=5; r11=5; r12=5; r31=5; r32=5;

Fs=1.5; Fc=1;

sigma01=10^5; sigma02=10^5; sigmall=sqrt(10^5);
sigma12=sqrt(10^5);
sigma21=0.4; sigma22=0.4; vs=1/(0.001*0.001);

l1=.38; l2=.24; p1=3.4;p2=.4;p3=.3;
mp=50;pp1=p1;pp2=p2;pp3=p3; % payload =50(N)
tf=4; r=.1;xc=0.48;yc=0; gama=eye(2)*100; ki=[0 0;0 0]

fv=200*eye(2); tau=1.0*eye(3); num1=0;

A=[1 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0;
    0 0 2 0 0 0 0 0;
    0 0 0 6 0 0 0 0;
    1 tf tf^2 tf^3 tf^4 tf^5 tf^6 tf^7;
    0 1 2*tf 3*tf^2 4*tf^3 5*tf^4 6*tf^5 7*tf^6;
    0 0 2 6*tf 12*tf^2 20*tf^3 30*tf^4 42*tf^5;
    0 0 0 6 24*tf 60*tf^2 120*tf^3 210*tf^4];
phi0=[0;0;0;0;2*pi;0;0;0];
a=inv(A)*phi0;
phic=a'
dphic=[phic(2) phic(3)phic(4) phic(5) phic(6) phic(7) phic(8)]
ddphic=[phic(3) phic(4) phic(5) phic(6) phic(7) phic(8)]

xr0=[xc+r;yc]; dxr0=[0;0];

```

```

x=xc+r;y=yc; d=(x^2+y^2-l1^2-l2^2)/(2*l1*l2);
q2=acos(d); q1=atan(y/x)-atan((l2*sin(q2))/(l1+l2*cos(q2)));
x0=[q1;q2;dxr0;0;0;0;0;0;0;0;0;0;0;0]; pack;
T=0:0.004:tf; options = odeset('RelTol',1e-6,'AbsTol',1e-6);
[t,x]=ode23('lugrediff',T,x0,options);

%%% Plot the result %%%
trajref;
%plot the angular position;
fig=figure set(fig,'Position',[0 60 800 660]);
subplot(3,1,1);plot(t,x(:,1),'-',t,x(:,2),'--');zoom on; grid
on;legend('q_1','q_2');
title('Angular position (Adaptive control-LuGre model
compensation)');
xlabel('t (sec)');
ylabel('q_1, q_2 (rad)');

subplot(3,1,2); plot(t,x(:,3),'-',t,x(:,4),'--'); zoom on;
grid on;
legend('q_1dot','q_2dot',0);
title('Angular velocity (Adaptive
control-LuGre model compensation)');
xlabel('t (sec)');
ylabel('q_1dot, q_2dot (rad/s)');

%plot the angular errors
subplot(3,1,3);
plot(t,qd(:,1)-x(:,1),'-',t,qd(:,2)-x(:,2),'--')
grid on;
legend('Error (q_1_d-q_1)','Error (q_2_d-q_2)',0);
title('Angular Error Signals');
xlabel('t (sec)');ylabel('q_1_d-q_1, q_2_d-q_2 (rad)');

print c:\zhu\robotics\pos_err_adptvlugre.eps -dps

```

```

fig=figure set(fig,'Position',[0 60 800 660]);
subplot(3,1,1);
F=zeros(length(t),2);
    for ntspan=1:length(t),
        varphi1=abs(x(ntspan,3))*sigma01/(Fc+(Fs-Fc)
            *exp(-x(ntspan,3)*x(ntspan,3)*vs));    % vs==1/vs^2
        varphi2=abs(x(ntspan,4))*sigma01/(Fc+(Fs-Fc)
            *exp(-x(ntspan,4)*x(ntspan,4)*vs));    % vs==1/vs^2
        F1=[(sigma11+sigma21)*x(ntspan,3)+(sigma01-varphi1*sigma11)
            *x(ntspan,15) (sigma12+sigma22)*x(ntspan,4)
            +(sigma02-varphi2*sigma12)*x(ntspan,16)];
        F(ntspan,:)=F1;
    end
plot(t,F(:,1),'-',t,F(:,2),'--'); zoom on;
grid on;legend('F_1','F_2',0);
title('Froction torque (Adaptive control-LuGre
model compensation)');
xlabel('t (sec)');
ylabel('F_1, F_2 (N*m)');

subplot(3,1,2);
plot(t,x(:,11),'-',t,x(:,13),'--',t,x(:,15),'.-'); zoom on;
grid on;legend('z_0_1','z_1_1','z_1',0);
title('Internal friction state &
estimates for base joint(Adaptive
control-LuGre model compensation)');
xlabel('t (sec)');
ylabel('z_0_1, z_1_1, z_1');

subplot(3,1,3); plot(t,x(:,12),'-',t,x(:,14),'--',t,x(:,16),'.-');
zoom on; grid on;
legend('z_0_2','z_1_2','z_2',0);
title('Internal friction state & estimates for arm
(Adaptive control-LuGre model compensation)');
xlabel('t(sec)'); ylabel('z_0_2, z_1_2, z_2');
print c:\zhu\robotics\force_z_adptvluGre.eps -dps

```

```

fig=figure set(fig,'Position',[0 60 800 660]);

subplot(3,1,1); plot(t,x(:,5),'-',t,x(:,6),'--'); zoom on; grid
on;legend('\beta_0_1','\beta_0_2',0);
title('Parameter estimate
      (Adaptive control-LuGre model compensation)');
xlabel('t (sec)');
ylabel('\beta_0_1, \beta_0_2');

subplot(3,1,2); plot(t,x(:,7),'-',t,x(:,8),'--'); zoom on; grid
on;legend('\beta_1_1','\beta_1_2',0);
xlabel('t (sec)');
ylabel('\beta_1_1, \beta_1_2');

subplot(3,1,3); plot(t,x(:,9),'-',t,x(:,10),'--'); zoom on; grid
on;legend('\beta_3_1','\beta_3_2',0);
  xlabel('t (sec)');
  ylabel('\beta_3_1,\beta_3_2');

print c:\zhu\robotics\estimate_adptvlugre.eps -dps

```

## B.6 Lugrediff.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Integration fuction for LuGre.m           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function QDOT=adptvdiff(t,p)
global l1
global l2
global p1 p2 p3
global tf
global r xc yc
global gama fv tau
global phic dphic ddp hic

```

```

global num1
global sigma01 sigma02
global sigma11 sigma12
global sigma21 sigma22
global Fs Fc vs
global r01 r02 r11 r12 r31 r32
global kp kd ki
global serr

num1=num1+1;
x1=[p(1);p(2)];      %angle
v1=[p(3);p(4)];      %angular velocities
beta01hat=p(5);
beta02hat=p(6);
beta11hat=p(7);
beta12hat=p(8);
beta31hat=p(9);
beta32hat=p(10);
z0hat=[p(11);p(12)];
z1hat=[p(13);p(14)];
z=[p(15);p(16)];

if t>tf
    t1=t-tf;
else
    t1=t;
end

phi=phic*[1;t1;t1^2;t1^3;t1^4;t1^5;t1^6;t1^7];
dphi=dphic*[1;2*t1;3*t1^2;4*t1^3;5*t1^4;6*t1^5;7*t1^6];
ddphi=ddphic*[2;6*t1;12*t1^2;20*t1^3;30*t1^4;42*t1^5];

x=xc+r*cos(phi);
y=yc+r*sin(phi);

```

```

dx=-r*sin(phi)*dphi;
dy=r*cos(phi)*dphi;
ddx=-r*sin(phi)*ddphi-r*cos(phi)*dphi*dphi;
ddy=r*cos(phi)*ddphi-r*sin(phi)*dphi*dphi;

d=(x^2+y^2-l1^2-l2^2)/(2*l1*l2);
q2=acos(d);
q1=atan(y/x)-atan( (l2*sin(q2))/(l1+l2*cos(q2)) );

q=[q1;q2];
c1=cos(q1);c2=cos(q2);c12=cos(q1+q2);s1=sin(q1);
s2=sin(q2);s12=sin(q1+q2);

j1=[l2*c12 l2*s12];
j2=[-l1*c1-l2*c12 -l1*s1-l2*s12];

xv=[dx;dy];

jinv=[j1;j2]/(l1*l2*s2);

dq=jinv*xv;
dq1=dq(1); dq2=dq(2);

ddq1=(l1*c2*dq2*(c12*dx+s12*dy)-l1*s2*(-s12*dx*(dq1+dq2)
+c12*ddx+c12*dy*(q1+q2)+s12*ddy))/(l1^2*s2^2);
ddq2=( l1*l2*c2*dq2*((l1*c1+l2*c12)*dx+(l1*s1+l2*s12)*dy)
-l1*l2*s2*((l1*c1+l2*c12)*ddx+(-l1*s1*dq1-l2*s12*(dq1+dq2))*dx
+(l1*s1+l2*s12)*ddy
+(l1*c1*dq1+l2*c12*(dq1+dq2))*dy )/((l1*l2*s2)^2);
ddq=[ddq1;ddq2];

%%%%%%%%%%%% m and c matrix
m=[p1+2*p3*cos(x1(2)) p2+p3*cos(x1(2));p2+p3*cos(x1(2)) p2];
c=[-p3*v1(2)*sin(x1(2))
-p3*(v1(1)+v1(2))*sin(x1(2)); p3*v1(1)*sin(x1(2)) 0];

```



```

%%%%%%%% Control
e=(x1-q);de=(v1-dq);serr=serr+e*.004;
ev=de+gama*e+ki*serr;
dqr=dq-gama*e;
ddqr=ddq-gama*de;
ev1=ev(1);
ev2=ev(2);

varphi1=abs(v1(1))*sigma01/(Fc+(Fs-Fc)*exp(-v1(1)*v1(1)*vs));
varphi2=abs(v1(2))*sigma02/(Fc+(Fs-Fc)*exp(-v1(2)*v1(2)*vs));
varphi=[varphi1 0; 0 varphi2];

u=-fv*ev+[beta31hat*v1(1); beta32hat*v1(2)]+ [beta01hat*z0hat(1);
        beta02hat*z0hat(2)]-[varphi1*beta11hat*z1hat(1);
        varphi2*beta12hat*z1hat(2)]+m*ddqr+c*dqr;
F=[(sigma11+sigma21)*v1(1)+(sigma01-varphi1*sigma11)
        *z(1);(sigma12+sigma22)*v1(2)+(sigma02-varphi2*sigma12)*z(2)];

%%%%%%%% state space equation
x1dot=v1;
v1dot=inv(m)*(u-c*v1-F);
beta01hatdot=-r01*ev1*z0hat(1);
beta02hatdot=-r02*ev2*z0hat(2);
beta11hatdot=r11*ev1*varphi1*z1hat(1);
beta12hatdot=r12*ev2*varphi2*z1hat(2);
beta31hatdot=-r31*ev1*v1(1);
beta32hatdot=-r32*ev2*v1(2);
z0hatdot=v1-varphi*z0hat-ev;
z1hatdot=v1-varphi*(z1hat-ev);
zdot=v1-varphi*z;

QDOT=[x1dot; v1dot; beta01hatdot; beta02hatdot; beta11hatdot;
        beta12hatdot; beta31hatdot; beta32hatdot;
        z0hatdot; z1hatdot; zdot];

```

VITA

Yongliang Zhu

Candidate for the Degree of

Master of Science

Thesis: ADAPTIVE CONTROL OF MECHANICAL SYSTEMS WITH STATIC AND DYNAMIC FRICTION COMPENSATION

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Anhui, P.R.China , on January 15, 1967, the son of Zhixiong Zhu and Guangzao Fang.

Education: Received the B.S. degree from Zhejiang University, Zhejiang, P.R.China in 1988, in Electrical Engineering; Received the M.S. degree from University of Science and Technology of China, Hefei, Anhui, P.R.China in 1991, in Automation; Completed the requirements for the Master of Science degree with a major in Mechanical Engineering and a minor in Dynamic Systems and Control Engineering at Oklahoma State University in December, 2001.

Experience: Research Assistant at Oklahoma State University from 1999 to present; Senior Engineer at Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, P.R.China, from July 1998 to August 1999; Engineer at Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, P.R.China, from July 1994 to July 1998; Assistant Engineer at Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, P.R.China, from July 1991 to July 1994.