INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17" x 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600
Modelling, simulation, and control of robots

Chatterjee, Trinanjan, M.S.

Rice University, 1989
RICE UNIVERSITY

MODELLING, SIMULATION, AND CONTROL OF ROBOTS

by

TRINANJAN CHATTERJEE

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

APPROVED, THESIS COMMITTEE:

J. Boyd Pearson, Jr., J.S. Abercrombie Professor in Engineering, Chairman

Rui J.P. de Figueiredo
Rui J.P. deFigueiredo, Professor, Electrical and Computer Engineering and Mathematical Sciences

J.B. Cheatham, Jr., Professor, Mechanical Engineering and Material Sciences

Houston, Texas
August, 1988
Abstract

Modelling, Simulation, and Control of Robots

by

Trinanjan Chatterjee

The continuous dynamic model of a robot is represented by nonlinear coupled differential equations. These can be discretized using first or second order Taylor's series approximations. One such model, the Greenspan Discrete Mechanics model appears to be the best for control applications. Global linearization of this model is straightforward and the linearized model is used to design $l^1$ optimal controllers for the robot. The Multi Input Multi Output optimization problem splits up into decoupled Single Input Single Output problems which are individually solved. Optimal systems thus designed are simulated and compared to traditional robot controllers. Simulations confirm the $l^1$ optimality of the controllers. This thesis thus provides a useful new tool for the robot control engineer.
Acknowledgements

I wish to thank my advisor Professor J.B. Pearson for his support and guidance throughout this research.

I would also like to thank Professors Antoulas, deFigueiredo, and Cheatham for being on my thesis committee, and my co-graduate students for many stimulating discussions.
# Table of Contents

Chapter 1. Introduction ................................................................................................. 1  
1.1. Description of a Robot ......................................................................................... 1  
1.2. Areas of Robotics Research .............................................................................. 1  
1.3. Summary of Thesis .............................................................................................. 3  

Chapter 2. Dynamics .................................................................................................. 5  
2.1. Continuous Time Models ................................................................................... 5  
2.2. Discrete Time Models ........................................................................................ 8  
2.3. Dynamics Simulations Using Continuous and Discrete Robot Models .......... 15  
2.4. Parameter Estimation ......................................................................................... 16  

Chapter 3. Linearization Schemes .............................................................................. 23  
3.1. Local and Global Linearization ......................................................................... 23  
3.2. Global Linearization ......................................................................................... 24  

Chapter 4. Traditional Control Systems for Robots .................................................. 33  
4.1. Independent Joint PID Control ......................................................................... 34  
4.2. Computed Torque Control - Continuous Time ............................................... 35  
4.3. Computed Torque Control - Discrete Time ...................................................... 36  
4.4. Simulations of Traditional Controllers ........................................................... 37  

Chapter 5. Results From $l^1$ Optimal Control ...................................................... 75  
5.1. $l^1$ Optimal Control ......................................................................................... 75  
5.2. Framework For Solving MIMO Control Problems ......................................... 79  

Chapter 6. Optimal $l^1$ Controller for a Robot ....................................................... 81  
6.1. Transfer Function for the Linear Model ........................................................... 81  
6.2. The Complete MIMO Robot Model and Optimization ..................................... 82  
6.3. Construction of the Optimal Controller .......................................................... 84  
6.4. Effect of Windowing Functions ......................................................................... 86  
6.5. Effect of Interpolation Points on the Unit Circle ............................................. 87  
6.6. Some Example Designs ................................................................................... 87  
6.7. Generalized Theory ......................................................................................... 92  

Chapter 7. Simulations, Comparisons and Conclusions ......................................... 94  
7.1. Simulations of Optimal Controllers ............................................................... 94  
7.2. Optimality of the Optimal Controller .............................................................. 97
7.3. Conclusions ............................................................................................................. 98
7.4. Topics for Further Research .......................................................................................... 100
Appendix A. Bibliography .................................................................................................. 117
Bibliography .................................................................................................................. 117
CHAPTER 1

Introduction

1.1. Description of a Robot

In general, any system capable of manipulating objects can be termed a robot. Our definition of a robot is more narrow. It is a mechanical system made up of a series of rigid links joined together as a chain. Each joint is associated with one or more degrees of freedom. The degrees of freedom can be rotational or prismatic. The two ends of the robot are the base and the tip. The base is fixed to a support. The tip is free to move in space and usually houses an end-effector to do useful tasks.

1.2. Areas of Robotics Research

Robot Dynamics

A robot moves because of generalized forces being applied at its joints. The generalized force can be either a torque or a force. For rotational joints a torque is applied and for prismatic joints a force is applied.

A robot can be thought of as a black box. The input to the black box is a vector of generalized forces and the outputs are vectors of joint angles or joint positions. The input-output relationship is termed the dynamics of the robot. Mathematically the relation is a set of non-linear coupled differential equations in the joint variables with joint forces as forcing functions. Solving for the dynamics implies solving these
differential equations. In chapter 2, continuous and discrete time dynamic models are presented. Schemes for computing robot response using these models are also presented.

Trajectory and Path Planning

We generally require a robot to perform a task like moving an object from one position to another. This is a high level task. Path planning converts the high level task into a sequence of low level tasks. Low level tasks might involve specifying intermediate points in the path to avoid obstacles. Transforming the low level tasks into joint angles or positions as functions of time is termed trajectory planning.

Control Systems for Robots

The control problem assumes that trajectory planning has already been done. The outputs of the trajectory planner are the desired joint angles or positions as functions of time. These desired values are inputs to the control system. The other inputs called feedback are the actual joint angles and positions as measured by sensors on the robot. The output of the control system is a vector of generalized forces that has to be applied to the robot, if the robot is to move as desired by the trajectory planner.

If the dynamic equations of motion are exactly known, feedback is not required and the control problem reduces to a trivial inverse dynamics solution. In reality, modelling uncertainties are always present. Designing a robust controller that tracks well in spite of uncertainties is a challenging problem. This is particularly so because the robot dynamic equations of motion are nonlinear.
Robotic Vision

An intelligent robot needs to monitor its environment constantly. Hence vision is indispensable to a robot. Optical data is obtained by a battery of cameras. Transforming the raw data into individual images requires a lot of signal processing and number crunching. Interpretation of these images to give a model of the environment involves considerable analysis and uses techniques from artificial intelligence and neural networks.

1.3. Summary of Thesis

Our research concerns designing control systems for robots. We first discuss various dynamic models of the robot. This is done in chapter 2. Methods for linearizing the nonlinear dynamic equations are presented in chapter 3. The simplest kind of control is independent joint PID control or PD control. Almost all industrial and commercial robots use this kind of control. It is computationally very simple but works only when the requirements are elementary. It is described in chapter 4. Other kinds of control use local or global linearization schemes. One such method is the computed torque method of control. It is also briefly described in chapter 4.

We intend to design optimal controllers. We start with a continuous time model of the robot dynamics. We discretize this model and then perform global linearization. The result is a linear discrete time model with the tracking error $e(t)$ as one of the outputs. $e(t)$ is a function of exogenous inputs $\theta_d(t)$, the desired position, and $W(t)$, the modelling error. Our optimal controllers are those that minimize the $l^1$ norm of $\phi(t)$, the impulse response of $\Phi(z)$ the tracking error transfer function. $\Phi(z)$
relates the tracking error to the exogenous inputs. The optimal controllers thus
designed are intended to lead to robot control systems that have small tracking errors
and are robust to modelling errors. Since good tracking and robustness are primary
objectives in the design of any control system our research is well motivated.

The results we have used from $l^1$ control theory are presented in chapter 5.
Chapter 6 develops the theory and the design procedure for the $l^1$ optimal robot con-
troller. Two examples are worked and these explain the design procedure clearly.
Practical robot controllers are designed and tested on the cylindrical robot in chapter
7. Chapter 7 also contains the conclusions from the research and indicates avenues
for further research.

We investigated several robot models and found the Greenspan model to be the
best for our control purposes. It is easy to globally linearize the Greenspan model.
Traditional controllers display good tracking performance for most inputs. We
present the theory and design methodology necessary to design $l^1$ optimal controllers.
These controllers successfully minimize worst case errors and are more robust to
parameter variations. It is preferable to use these optimal controllers over traditional
controllers in situations where the worst case performance of the robot is important.
CHAPTER 2

Dynamics

A good dynamic model is important in designing control systems. A good model is one that is both simple and accurate. Hence the importance of dynamics in this thesis. Apart from helping design control systems, solving the dynamic equations of motion is independently important. It is often inconvenient and sometimes dangerous to use the real robot in experiments. A computer simulation of a robot is then preferred. Fast and accurate computation of the dynamics is the basis of any simulation. The dynamics of a robot with $n$ degrees of freedom can be represented by $n$ coupled, nonlinear, second order differential equations. This is the continuous time model. Various methods of discretization may be used to get a discrete time model. This chapter introduces continuous and discrete time models for robot dynamics.

However good a model might be, it is useless unless the parameters associated with the model can be accurately estimated. Hence a very simple scheme for parameter estimation is presented in this chapter.

2.1. Continuous Time Models

There are two fundamental approaches to deriving the dynamic equations of motion of a robot. They are the Lagrangian and Newton-Euler methods. The first approach uses Lagrange equations while the second uses Newton-Euler equations.
The Lagrange method is simpler and leads to closed form equations that are easy to apply in developing control systems. Newton-Euler equations are more difficult to formulate. On the other hand it is very simple to get a recursive form for the Newton-Euler equations and hence very efficient computation methods can be achieved. The Newton-Euler form is more popular with those writing dynamics simulations while the Lagrange form is more popular with control system designers.

We shall consider only the Lagrange form in this thesis. However it is important to note that the difference between the two forms is superficial. In fact researchers have developed recursive Lagrange equations and closed form Newton-Euler equations. A detailed reference for this section is [1].

**Lagrangian Dynamics**

The Lagrangian of a system $L$ is given by

$$L = K - P$$  \hspace{1cm} \text{(2.1)}

where $K$ is the total kinetic energy and $P$ is the total potential energy of the robot.

The Lagrange equations are

$$\tau_i = \frac{d}{dt} \left[ \frac{\partial L}{\partial \dot{q}_i} \right] - \frac{\partial L}{\partial q_i} \quad i = 1, \ldots, n$$  \hspace{1cm} \text{(2.2)}

where $q_i$ is the joint angle or displacement, and $\tau_i$ is the generalized force for the $i^{th}$ joint.

Equation (2.2) can written in an expanded form using matrix notation as

$$M(\Theta)\ddot{\Theta} + C(\Theta, \dot{\Theta}) = \tau$$  \hspace{1cm} \text{(2.3)}
where $\Theta, \tau, C$ are $n \times 1$ vectors with $\Theta = (q_1, \ldots, q_n)^T$, $\tau = (\tau_1, \ldots, \tau_n)^T$, and $M$ is a $n \times n$ positive definite matrix. Equation (2.3) is a nonlinear coupled differential equation.

**An Example of a Robot**

A simple robot is shown in figure (2.1) on page 18. It has two degrees of freedom. The first degree of freedom, closest to the base, is rotational since the first link is free to rotate around the base. The angle of rotation $\theta$ is the joint variable. The second degree of freedom is prismatic. The joint variable is the radial displacement $r$. The torque applied to the first joint is called $F_\theta$ and the force applied to the second joint is called $F_r$. Hence $\Theta = (\theta \ r)^T$ and $\tau = (F_\theta \ F_r)^T$. The dynamic equations of motion for the above robot can be written as

$$[J + j(r)] \ddot{\theta} + \frac{\partial j}{\partial r} \dot{\theta} \dot{r} = F_\theta(t)$$  \hspace{1cm} (2.4)

$$m \ddot{r} - \frac{1}{2} \frac{\partial j}{\partial r} \dot{\theta}^2 = F_r(t)$$  \hspace{1cm} (2.5)

where $J$ is a constant inertia, $j(r) = mr^2 - m_p R r$ is a variable inertia, $m$ is the mass of the radial link, $m_p$ is the mass of the payload, and $R$ is the maximum extended radius.

Writing (2.4) and (2.5) in the form of (2.3) we get

$$M = \begin{bmatrix} J + j(r) & 0 \\ 0 & m \end{bmatrix}, \quad C = \begin{bmatrix} \frac{\partial j}{\partial r} \dot{\theta} \\ \frac{1}{2} \frac{\partial j}{\partial r} \dot{\theta}^2 \end{bmatrix}$$  \hspace{1cm} (2.6)

The robot described above is a basic industrial robot called the cylindrical robot.
Although relatively simple it displays the nonlinear coupled dynamics that is characteristic of any robot. Actually the entire cylindrical robot can usually be lifted along the vertical axis. Technically this is an additional degree of freedom. However notice that this motion is perpendicular to the other two motions. Hence the dynamics and control of this movement is independent of the other two movements and represents a straightforward control problem that can be independently solved.

The cylindrical robot shall be used in simulations and as an example in the rest of the thesis. An accurate simulation of the cylindrical robot dynamics can be done if we use a good numerical differential equations solving algorithm. The algorithm we shall use is the Fehlberg fourth-fifth order Runge-Kutta method as programmed in Fortran in the routine RKF45 by Watts and Shampine at Sandia Laboratories. Dynamic simulations using the continuous model are presented in section (2.3).

2.2. Discrete Time Models

In a continuous time world our main computational tool is the digital computer. In order to have computer implementable algorithms it is necessary to use a discrete time model of the robot dynamics. It is not possible in general to get exact discrete time equivalents of nonlinear continuous time systems. A certain amount of approximation is almost always necessary. To get smaller modelling errors we could use more accurate discrete time models. The idea of one model being more accurate than another requires a little explanation. Let us consider two models with the same sampling time $T$. The model that can predict the trajectory of the robot more accurately than the other is called the more accurate model. Reducing the sampling time $T$ for a
particular model will in general also lead to more accurate trajectories. However in real time applications like robot control there is a practical limitation as to how small $T$ can be. All the computations at each interval have to be done within the interval itself. Usually the more accurate a model is, the more is the computation required at every sampling interval. Hence a less accurate, but simpler model with a smaller $T$ could effectively predict trajectories with less error. Getting small errors is therefore a tradeoff between small sampling times and complex models. In this section two kinds of discrete models will be presented. The first kind are straightforward first and second order algorithms. The second kind of discretization developed in [2], is an inherently discrete time model that aims to conserve energy and momentum.

First and Second Order Algorithms

Equation (2.3) can be written as

$$\ddot{\Theta} = M(\Theta)^{-1}(\tau - C(\Theta, \dot{\Theta}))$$  \hspace{1cm} (2.7)

Defining

$$\dot{\Theta} = \Omega$$  \hspace{1cm} (2.8)

we can write (2.7) as

$$\dot{\Omega} = F(\Theta, \Omega, \tau)$$  \hspace{1cm} (2.9)

where $F$ is given by equation (2.7).

Assuming a sampling time of $T$ any first order differential equation like $\dot{x} = \nu$
can be approximated by a first order Taylor series as $x(k+1) = x(k) + T\nu(k)$. A second
order Taylor series approximation called the trapezoidal rule gives
\[ x(k+1) = x(k) + \frac{T}{2} [v(k+1) + v(k)]. \]

Using a first order approximation for both (2.8) and (2.9) we get
\[ \Theta(k+1) = \Theta(k) + T \Omega(k) \] (2.10)
\[ \Omega(k+1) = \Omega(k) + T F(\Theta(k), \Omega(k), \tau(k)) \] (2.11)

This is called the Forward Euler model.

Using a trapezoidal approximation for (2.8) and a first order approximation for (2.9) we get
\[ \Theta(k+1) = \Theta(k) + \frac{T}{2} [\Omega(k+1) + \Omega(k)] \] (2.12)
\[ \Omega(k+1) = \Omega(k) + T F(\Theta(k), \Omega(k), \tau(k)) \] (2.13)

This is called the Greenspan Discrete Mechanics model. For a frictionless constant inertia system this model conserves energy and momentum.

Let us compute the Forward Euler model and the Greenspan model for the cylindrical robot. Writing the dynamic equations of the robot in the form of (2.8) and (2.9) we get
\[ \dot{\theta} = \omega \] (2.14)
\[ \dot{r} = v \] (2.15)
\[ \dot{\omega} = \frac{F_\theta(t) - \frac{\partial}{\partial r} \dot{\theta} \dot{r}}{J + j(r)} \]  
(2.16)

\[ \dot{v} = \frac{F_r(t) + \frac{1}{2} \frac{\partial}{\partial r} \dot{\theta}^2}{m} \]  
(2.17)

We can now write the Forward Euler model for the cylindrical robot as below

\[ \theta(k+1) = \theta(k) + T \omega(k) \]  
(2.18)

\[ r(k+1) = r(k) + T \nu(k) \]  
(2.19)

\[ \omega(k+1) = \omega(k) + T \left[ \frac{F_\theta(k) - \frac{\partial}{\partial r} \dot{\theta}(k) \dot{r}(k)}{J + j(k)} \right] \]  
(2.20)

\[ \nu(k+1) = \nu(k) + T \left[ \frac{F_r(k) + \frac{1}{2} \frac{\partial}{\partial r} \dot{\theta}(k)^2}{m} \right] \]  
(2.21)

Equations (2.18) and (2.19) are called smoothing formulas since they relate the position and velocity on a trajectory. The only difference between the Forward Euler model and the Greenspan model is the smoothing formulas. The Greenspan model has a second order smoothing formula instead of a first order one. The smoothing formulas for the Greenspan model are given below

\[ \theta(k+1) = \theta(k) + \frac{T}{2} [\omega(k+1) + \omega(k)] \]  
(2.22)

\[ r(k+1) = r(k) + \frac{T}{2} [\nu(k+1) + \nu(k)] \]  
(2.23)
It is important to note that the Forward Euler model is an explicit model whereas the Greenspan model is an implicit model. In an explicit model the values of joint variables at time $k+1$ are functions only of variables that are known at time $k$. Solving for the dynamics in a explicit model is therefore simply a matter of plugging in the values at time $k$ to get the values at time $k+1$. Although the Greenspan model is implicit it can be converted into an explicit model by substituting (2.13) into (2.12). Apart from the ease of solving explicit equations, we shall see in chapter 3 that explicit models are easier to globally linearize than implicit models.

Dynamics simulations using Forward Euler and the Greenspan model are displayed in section (2.3).

**Neuman-Tourassis Model**

Instead of considering the general case we shall only investigate the cylindrical robot as an example. The Neuman-Tourassis model ensures conservation of energy and momentum, in the absence of friction, in a variable inertia system like the robot [2]. This is something that the first and second order methods do not necessarily do. The Neuman-Tourassis model uses trapezoidal approximation for the smoothing formula. The model appears below

$$\theta(k+1) = \theta(k) + \frac{T}{2} [\omega(k+1) + \omega(k)] \tag{2.24}$$

$$r(k+1) = r(k) + \frac{T}{2} [\nu(k+1) + \nu(k)] \tag{2.25}$$
\[ (J + j(k+1))\omega(k) - (J + j(k))\omega(k) = TF_\theta(k) \quad (2.26) \]

\[ m[v(k+1)-v(k)] - \frac{T}{2} \left[ \frac{j(k+1) - j(k)}{r(k+1) - r(k)} \right] \omega(k)\omega(k+1) = TF_r(k) \quad (2.27) \]

It is debatable as to whether conserving energy and momentum leads to more accurate trajectory computation. Even if it does lead to more accurate trajectories, the model is definitely more complicated than the previous ones.

**Non-Uniqueness of the Neuman-Tourassis Model**

There is a very serious objection to the Neuman-Tourassis model. To understand this notice that the model is clearly implicit. In fact equations (2.24) - (2.27) are a set of nonlinear, simultaneous, algebraic equations. Given the values of \( \theta, r, \omega, F_r, F_\theta, \) and \( v \) at time \( k, \) and the parameters of the robot, these equations tell us what \( \theta, r, \omega, \) and \( v \) will be at time \( k+1. \) Any valid model should give a unique value for the variables at time \( k+1. \) The variables at time \( k+1 \) are solutions of the system of equations. However there is no reason to believe that the equations have only one real solution. A little analysis will clarify the situation. Algebraic manipulation of equations (2.24)-(2.27) gives us the set of equations

\[ \left\{ a_3v^2(k+1) + a_2v(k+1) + a_1 \right\} \omega(k+1) + a_0 = TF_\theta(k) \quad (2.28) \]

\[ b_3v(k+1) + \left\{ b_2v(k+1) + b_1 \right\} \omega(k+1) - b_0 = TF_r(k) \quad (2.29) \]

where
\[
a_3 = \frac{mT^2}{4} \quad a_2 = mT(r(k) + \frac{T}{2}v(k)) - \frac{m_pRT}{2}
\]

\[
a_1 = J + m(r(k) + \frac{T}{2}v(k))^2 - m_pR(r(k) + \frac{T}{2}v(k)) \quad a_0 = (J + \mu(r(k)))\omega(k)
\]

and

\[
b_3 = m \quad b_2 = -\frac{T^2}{4}m\omega(k)
\]

\[
b_1 = -\frac{T}{2}m(r(k) + v(k)\frac{T}{2})\omega(k) - \frac{T}{2}m_pR\omega(k) - \frac{T}{2}mr(k)\omega(k) \quad b_0 = -mv(k)
\]

Equations (2.28) and (2.29) are third order algebraic equations. They have three roots. Either all the roots are real or one is real and the other two are complex conjugates. In most of the simulations that we have done there has been only one real solution. However Mustafa Khammash at Rice University has found a counter example to this trend [3]. It is given below.

Consider the parameters \( J=1.0, m=1.0, m_p=0.1, R=10.0, T=1.0 \) and the variables at time \( k \) are \( \theta(k)=0.0, \ r(k)=0.0, \ \omega(k)=4.0, \ v(k)=1.69546, \ F_r(k)=4.40761 \) and \( F_\theta(k)=6.390642 \). There are three sets of values of \( \theta(k+1), \ r(k+1), \ \omega(k+1) \) and \( v(k+1) \) that satisfy the the Neuman-Tourassiss model as given by equations (2.24)-(2.27). They are

\[
\theta(k+1)=2.1893 \quad r(k+1)=5.6662 \quad \omega(k+1)=.37867 \quad v(k+1)=9.6371 \quad (2.30)
\]

\[
\theta(k+1)=8.3804 \quad r(k+1)=.75348 \quad \omega(k+1)=12.761 \quad v(k+1)=-.18849 \quad (2.30)
\]
\[ \theta(k+1) = 3.0752 \quad r(k+1) = -1.5204 \quad \omega(k+1) = 2.1503 \quad v(k+1) = -4.7363 \quad (2.30) \]

Given the three solutions as above there is no way to decide which solution to choose. This is clearly unacceptable. The Neuman-Tourassis model therefore suffers from a serious lack of credibility in dynamic simulation and control applications.

To be fair to the Neuman-Tourassis model however I must mention that it took considerable effort to find the counter example. Non unique trajectories did not show up in the normal simulations. The trajectory predicted by the Neuman-Tourassis model for the cylindrical robot matches the trajectory as predicted by differential equation solving routines like RKF45. It might therefore be possible that the equations are locally unique. Locally unique means that if we use a small sampling time and search for a real solution (values of variables at time \( k+1 \)) close to the previous solution (values of variables at time \( k \)) we would get trajectories that are unique and match actual trajectories as predicted by RKF45.

In this thesis we display the results of dynamic simulation using the Neuman-Tourassis model. In Chapter 3 we describe a scheme for global linearization of the model. But owing to its proven non-uniqueness we shall not use it to design control systems.

2.3. Dynamics Simulations Using Continuous and Discrete Robot Models

Computer simulations of the dynamics is done using the continuous time model, Forward Euler model, the Greenspan model and the Neuman-Tourassis model. The results are shown in figures (2.2), (2.3), (2.4) and (2.5) respectively. The figures are
on pages 19 through 22. The same generalized forces are used as an input to all the models and hence the results may be compared. Looking at the simulation figures it is difficult to differentiate between the responses. At this stage a closer look at the differences in the the predicted responses of the different models would perhaps not lead to any definite conclusions. This is because the open loop robot responses are unstable and it is unfair to conclude that one model is more accurate than another based on unstable responses. Therefore should such a comparison be necessary, we shall wait until we have stable closed loop systems before we compare the models.

2.4. Parameter Estimation

Consider the equations of a robot as in equation (2.3). It has been shown in [4] that it is possible to write this equation in the form

\[ W(\Theta, \dot{\Theta}, \ddot{\Theta})P = \tau \]  

(2.28)

where \( P \) is a constant vector of unknown parameters. Assuming that it is possible to measure \( \Theta, \dot{\Theta}, \ddot{\Theta}, W \) and \( \tau \) are numerically known at every measurement. The only unknown is the vector of parameters \( P \). Every measurement leads to a set of linear simultaneous equations in \( P \). Every new measurement leads to a new set of equations. The resultant system is of the form

\[ AP = B \]  

(2.29)

where \( A \) and \( B \) are known. \( P \) can now be found by solving the linear algebraic system (2.29) in a least square fashion.
The above method while very simple and elegant suffers from the drawback that velocities and accelerations have to be measured. The success of the method therefore depends on how accurately we are able to measure these. If velocity or acceleration cannot be accurately measured then more complicated parameter estimation schemes have to be used.

In this chapter we investigated different dynamic models for a robot. In the next chapter we shall present schemes for globally linearizing these models.
THE CYLINDRICAL ROBOT

FIGURE (2.1)
Dynamic Response of the Cylindrical Robot - Figure (2.2)

Computed Using RKF45
Dynamic Response of the Cylindrical Robot - Figure (2.3)

Computed Using the Discrete Forward Euler Model

Torque Applied

Force Applied

Angular Displacement

Radial Displacement

Angular Acceleration

Radial Acceleration
Dynamic Response of the Cylindrical Robot - Figure (2.4)

Computed Using the Greenspan Model

- Torque Applied
- Force Applied
- Angular Displacement
- Radial Displacement
- Angular Acceleration
- Radial Acceleration
Dynamic Response of the Cylindrical Robot - Figure (2.5)

Computed Using the Neuman-Tourassis Model

Torque Applied

Force Applied

Angular Displacement

Radial Displacement

Angular Acceleration

Radial Acceleration
CHAPTER 3

Linearization Schemes

Linear control systems are much better understood than nonlinear control systems. In fact there are very few techniques that can be applied to all classes of nonlinear systems. However generalized techniques exist which can be applied to the control of any linear system. A lot of nonlinear control therefore aims at transforming the nonlinear system into a linear system, if possible. This is the topic of discussion in this chapter.

3.1. Local and Global Linearization

The classical technique used involves linearizing a nonlinear system around a certain operating point. A Jacobian is computed at the operating point and this is used as the system matrix of an equivalent linear system with state variables measured as deviations of the original state variables from the operating point. A new Jacobian and a new linear system is associated with every new operating point. Also the closer one is to the operating point the closer the behaviour of the linear system is to that of the nonlinear system. The above linearization scheme is valid only in a local region and hence is called a local linearization scheme. Obviously a linearization scheme that holds at any point in the workspace of the robot would be preferred. Such a scheme is called a global linearization scheme. We have used it in our control system. Global linearization is described in the next section.
3.2. Global Linearization

As mentioned earlier global linearization is exact and valid almost everywhere. However not all nonlinear systems can be globally linearized. Fortunately the continuous time model of a robot can be globally linearized. Discretization may destroy the global linearizability of a continuous time robot model. Hence an important consideration in choosing a discrete time model is whether global linearizability is lost or not. We shall consider global linearization of the continuous time and discrete time robot models.

Continuous Time Models

Consider equation (2.3) repeated below as

\[ M(\Theta)\ddot{\Theta} + C(\Theta,\dot{\Theta}) = \tau \]  \hspace{1cm}(3.1)

Assume that the torque \( \tau \) we feed to the robot is given by

\[ \tau = M(\Theta)U + C(\Theta,\dot{\Theta}) \]  \hspace{1cm}(3.2)

This is nonlinear feedback with \( U \) as any arbitrary vector of input signals. The resultant system can be represented as

\[ \ddot{\Theta} = U \]  \hspace{1cm}(3.3)

This is a linear system. To control \( \Theta \), the output of the nonlinear system, all we need to control is this linear system and this can be done using conventional linear control theory. However this is an oversimplification. The reason is that we do not know what \( M \) and \( C \) exactly are. In fact all we really have are some estimates \( \hat{M} \) of \( M \) and
\( \dot{C} \) of \( C \). Therefore the actual nonlinear feedback is not given by equation (3.2) but by

\[
\tau = \dot{M}(\Theta) U + \dot{C}(\Theta, \dot{\Theta})
\]  

(3.4)

Substituting (3.4) in (3.1) we get

\[
\ddot{\Theta} = U - \dot{M}^{-1}(\ddot{M}\dot{\Theta} + \dot{C})
\]  

(3.5)

where \( \dot{M} = M - \hat{M} \) and \( \dot{C} = C - \hat{C} \) and the dependencies of each of them on \( \Theta \) and its derivatives is not shown but implicitly understood. We now have equation (3.5) instead of (3.3) as before. The additional term is a kind of modelling error. We call it \( W \) and it has the form

\[
W(\Theta, \dot{\Theta}, \ddot{\Theta}) = -\dot{M}^{-1}(\ddot{M}\dot{\Theta} + \dot{C})
\]  

(3.6)

So equation (3.5) can be written as

\[
\ddot{\Theta} = U + W
\]  

(3.7)

\( W \) is an additional input to the linear plant due to modelling error. Note that if \( M = \hat{M} \) and \( C = \hat{C} \) then \( W = 0 \) and there is no modelling error and hence no additional input.

**Discrete Time Models**

The Forward Euler model and the Greenspan model, defined in chapter 2, can always be linearized. This can be understood if we consider equations (2.11) or (2.13). They are repeated below as

\[
\Omega(k+1) = \Omega(k) + TF(\Theta(k), \Omega(k), \tau(k))
\]  

(3.8)

If it is possible to choose \( \tau(k) \) such that
\[ TF(\Theta(k), \Omega(k), \tau(k)) = U(k) \]  

(3.9)

then we get the linear system

\[ \Omega(k+1) = \Omega(k) + U(k) \]  

(3.10)

To understand that it is always possible to choose \( \tau(k) \) as required by equation (3.9) we have to study the structure of \( F \). Using (2.7) and (2.8) we find \( F = M^{-1}(\tau - C) \). Equation (3.9) requires that \( F = U/T \). Choosing

\[ \tau(k) = M(\Theta(k))U(k)/T + C(\Theta(k), \hat{\Theta}(k)) \]  

(3.11)

would give the linear system as in equation (3.10). However as in the continuous case we only have estimates of \( M \) and \( C \). Therefore we actually get the linear system

\[ \Omega(k+1) = \Omega(k) + U(k) + W(k) \]  

(3.12)

with \( W(k) \) given by

\[ W(k) = -T\hat{M}^{-1}(k)[\hat{M}(k)/T(\Omega(k+1) - \Omega(k)) + \hat{C}(k)] \]  

(3.13)

At this point it is useful to introduce errors due to discretization also. The Greenspan and Forward Euler models as specified by equation (3.8) are not exact like the continuous time dynamic model. Whenever we use a non zero sampling time \( T \) discretization errors always appear. A correct formulation of equation of equation (3.8) is therefore

\[ \Omega(k+1) = \Omega(k) + TF(\Theta(k), \Omega(k), \tau(k)) + D_e(k) \]

where \( D_e(k) \) is the discretization error. Consequently \( W(k) \) as defined by equation
(3.12) is now given by

\[ W(k) = -T\hat{M}^{-1}(k)[\hat{M}(k)/T(\Omega(k+1) - \Omega(k)) + \hat{C}(k)] + \hat{M}^{-1}MD_e(k) \]

Note that due to the presence of the discretization error in \( W(k) \) it is non zero even if all the parameters are known exactly. This is an important difference from the continuous time case.

Any linear system can be written in the state variable form as

\[ X(k+1) = AX(k) + B\bar{U}(k) \quad Y(k) = CX(k) \quad (3.14) \]

In the robotics problem

\[ X(k) = \begin{bmatrix} \Theta(k) \\ \Omega(k) \end{bmatrix} \quad \bar{U}(k) = \begin{bmatrix} 0 \\ U(k) + W(k) \end{bmatrix} \quad Y(k) = \begin{bmatrix} \Theta(k) \end{bmatrix} \quad (3.15) \]

For a \( n \) degree of freedom robot \( A \) and \( B \) depend on the discrete model used but \( C = [0 \ I_n] \) for any model. \( I_n \) is the \( n^{th} \) order identity matrix. \( A \) and \( B \) for the linearized Forward Euler model can be written using equations (2.10) and (3.12) as

\[ A = \begin{bmatrix} I_n & TI_n \\ 0 & I_n \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ I_n \end{bmatrix} \quad (3.16) \]

\( A \) and \( B \) for the linearized Greenspan model can be written using equations (2.12) and (3.12) as

\[ A = \begin{bmatrix} I_n & TI_n \\ 0 & I_n \end{bmatrix} \quad B = \begin{bmatrix} \frac{T}{2}I_n \\ I_n \end{bmatrix} \quad (3.17) \]

A very interesting and useful characteristic of \( A \) and \( B \), both in equation (3.16) and
(3.17) is that they represent $n$ decoupled systems, one for each joint. The form of each of these decoupled systems is identical to the form of the entire system. For example the system corresponding to equation (3.17) is equivalent to $n$ independent systems of the form

$$\begin{bmatrix} \theta(k+1) \\ \omega(k+1) \end{bmatrix} = A \begin{bmatrix} \theta(k) \\ \omega(k) \end{bmatrix} + B \begin{bmatrix} 0 \\ u(k) + w(k) \end{bmatrix}$$  \hspace{1cm} (3.18)$$

where

$$A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} T \\ 2 \\ 1 \end{bmatrix}$$  \hspace{1cm} (3.19)$$

Hence building $n$ independent controllers to control $n$ systems of the form of equation (3.18) is equivalent to controlling the linear system corresponding to equation (3.17) representing the entire robot.

It is evident from the discussion above that discretization using Forward Euler model and the Greenspan model maintains the global linearizability of the continuous time model. An inspection of the implicit Neuman-Tourassis model will show that there is no straightforward way to globally linearize this model. However under certain restrictive assumptions a global linearization can be done. One method to do global linearization will next be presented briefly.

Consider the Neuman-Tourassis model as shown in equations (2.24)-(2.27).

$$\theta(k+1) = \theta(k) + \frac{T}{2} [\omega(k+1) + \omega(k)]$$  \hspace{1cm} (2.24)$$
\[ r(k+1) = r(k) + \frac{T}{2} [v(k+1) + v(k)] \quad (2.25) \]

\[ [J + j(k+1)] \omega(k+1) - [J + j(k)] \omega(k) = TF_\theta(k) \quad (2.26) \]

\[ m[v(k+1)-v(k)] - \frac{T}{2} \left[ \frac{j(k+1) - j(k)}{r(k+1) - r(k)} \right] \omega(k) \omega(k+1) = TF_r(k) \quad (2.27) \]

Using equations (2.24) and (2.25) we can eliminate the dependencies of equations (2.26) and (2.27) on \( \theta(k+1) \) and \( r(k+1) \). The only unknowns at time \( k \) in the left hand side of equations (2.26) and (2.27) are \( \omega(k+1) \) and \( v(k+1) \). Hence equations (2.26) and (2.27) can be written as

\[ \Phi_\theta(\omega(k+1), v(k+1)) = TF_\theta(k) \quad (3.20) \]

\[ \Phi_r(\omega(k+1), v(k+1)) = TF_r(k) \quad (3.21) \]

In this case, for simplicity, we want the linear systems to look like

\[ \omega(k+1) = u_\theta(k) \quad v(k+1) = u_r(k) \quad (3.22) \]

If we choose

\[ TF_\theta(k) = \Phi_\theta(u_\theta(k), u_r(k)) \quad (3.23) \]

\[ TF_r(k) = \Phi_r(u_\theta(k), u_r(k)) \quad (3.24) \]

we would get the system

\[ \Phi_\theta(\omega(k+1), v(k+1)) = \Phi_\theta(u_\theta(k), u_r(k)) \quad (3.25) \]

\[ \Phi_r(\omega(k+1), v(k+1)) = \Phi_r(u_\theta(k), u_r(k)) \quad (3.26) \]
Given $u_\theta(k)$ and $u_\psi(k)$, $\omega(k+1) = u_\theta(k)$, $\nu(k+1) = u_\psi(k)$ is clearly a solution of equations (3.25) and (3.26). If this solution is unique then we have successfully linearized the Neuman-Tourassis model. If we had nonunique solutions then given a certain $u_\theta(k)$ and $u_\psi(k)$ there has to be more than one set of $\omega(k+1)$ and $\nu(k+1)$ that satisfies equations (3.25) and (3.26). This is equivalent to saying that given a certain $F_\theta(k)$ and $F_\psi(k)$, there is more than one set of $\omega(k+1)$ and $\nu(k+1)$ that satisfies equations (3.20) and (3.21). If this is true then at a particular sampling instant there is more than one solution possible using the Neuman-Tourassis model. The possibility of nonuniqueness is therefore inherent in the model itself and is not introduced by the global linearization scheme specified above. Hence if the model itself leads to unique trajectories then the global linearization above is valid.

The important question therefore is whether the Neuman-Tourassis model leads to unique trajectories. The authors dont shed any light on the subject in [2]. We have seen in Chapter 2 that it is in fact possible to get non unique trajectories. However in a local sense as explained in Chapter 2 one might get unique trajectories. Should the last statement be true then it should be possible to linearize the Neuman-Tourassis model using the scheme presented above. A simulation shows that we do get a linear model in this way. We use the nonlinear feedback as specified by equations (3.23) and (3.24) on a Neuman-Tourassis model of the cylindrical robot. Studying the plots in figure (3.1) on page 32, of $\omega(k+1) - u_\theta(k)$ and $\nu(k+1) - u_\psi(k)$, we find that they are zero within limits of computational error. Hence we do get a linear model. However there is no guarantee that the scheme works in all cases and hence we shall not use
the Neuman-Tourassis model in our control systems.

In this chapter we used global linearization to arrive at a linear model for the robot. This linear model will be the basis of some of the traditional robot controllers which we shall present in the next chapter. It will also be the basis for designing our optimal controller in chapter 6.
Linearization of the Neuman-Tourassis Model - Figure (3.1)
CHAPTER 4

Traditional Control Systems for Robots

Most industrial robots built in the last two decades have a rudimentary control system called the independent joint Positional Derivative (PD) control. It was found that this worked satisfactorily in practice for point to point control. It was not until recently that the stability of the scheme was proven theoretically [5]. In most cases point to point control was enough. However as more advanced robots are being built it is necessary not just to control the final position but to control the entire trajectory of motion. Implementing a trajectory using point to point control could be done by specifying a large number of intermediate points on the trajectory and then moving from one intermediate point to another. This is not only just an approximation but there is no control over how long it would take to traverse the trajectory. Hence the need for better control systems. Using global linearization and then applying PD control to the linear system is another approach. It is called the computed torque method of control [6]. It can be implemented using either a continuous model or a discrete model. The independent joint PD and computed torque control systems are described in this chapter. Other control schemes suggested have been adaptive schemes [4] and the so called sliding mode control [6]. These are complicated and are not described. It should be pointed out that all commercial robots use the independent joint PD control and most of the activity in designing better robot control sys-
tems is still in the research stage.

4.1. Independent Joint PID Control

This scheme of control treats the robot as if it were made of $n$ independent joints that have no interaction with each other. Further it ignores the dynamic structure of the robot entirely. That the scheme works at all appears miraculous at first glance. The control scheme is simple and consists of putting a PID controller at each joint. Most practical systems have only a PD controller and this is what we describe and simulate here. For the $i^{th}$ joint we have

$$\tau_i = k_p e_i + k_v \dot{e}_i$$  \hspace{1cm} (4.1)

and

$$e_i = \theta_{d,i} - \theta_i$$  \hspace{1cm} (4.2)

$\tau_i$ is the generalized force, $\theta_i$ is the joint variable, $\theta_{d,i}$ is the desired position, and $e_i$ is the position error, for the $i^{th}$ joint. If the above scheme is used for point to point control then $\dot{e}_i = -\dot{\theta}_i$. A careful study of the structure of Lagrangian dynamics is required to prove that the above scheme works. In [5] and [6] it has been proven that if we compensate for gravitational and frictional forces and choose positive $k_p$'s and $k_v$'s, the independent joint PD does in fact achieve stable point to point control. This only means that the robot will eventually get to the desired position. However to get better responses we have to choose the $k_p$'s and $k_v$'s properly. The process of choosing the right values of $k_p$ and $k_v$ for each joint is done by a trial and error process called tuning. Choosing $k_v^2 = 4k_p$ gives us a critically damped controller. Note however that
because we ignore the dynamics in independent joint control, the system itself is not necessarily critically damped. The speed of response can be increased by increasing the undamped natural frequency of the controller. The undamped natural frequency is equal to $\sqrt{k_p}$. The problem with increasing $k_p$ and $k_v$ is that this also increases the generalized force $\tau$. Since there are constraints on how large $\tau$ can be the tuning process is a trade off between having reasonable torques and getting quick responses. The above behaviour can be noticed in the simulations of the independent joint PD control at the end of the chapter.

4.2. Computed Torque Control - Continuous Time

Global linearization of the nonlinear robot leads to equation (3.7) shown below as

$$\ddot{\Theta} = U + W$$  \hspace{1cm} (4.3)

Let us choose

$$U = \ddot{\Theta}_d + K_v \dot{E} + K_p E$$  \hspace{1cm} (4.4)

where $K's$ are constant positive diagonal matrices, $E$ and $\dot{E}$ are the position and velocity error vectors, and $\ddot{\Theta}_d$ is the vector of desired accelerations. Combining equation (4.3) and (4.4) we have

$$\ddot{E} + K_v \dot{E} + K_p E = -W$$  \hspace{1cm} (4.5)

which represents $n$ stable second order differential equations in the joint errors $e_i$. As long as the $W$ is zero the tracking error goes asymptotically to zero. In the presence
of a persistent but small modelling error input $W$, which we hope is the case if we have a good model, the tracking error is small.

4.3. Computed Torque Control - Discrete Time

We need to consider only the Greenspan model since the Forward Euler model differs from it only in the smoothing formulas. The scheme of global linearization is the same in both cases. Since the Greenspan model uses a second order smoothing formula and the Forward Euler model uses a first order smoothing formula the Greenspan model is likely to be more accurate.

After we have performed global linearization on the Greenspan discrete time model we get equation (3.12). This is a vector equation. A single element of this vector can be written as

$$\omega(k+1) = \omega(k) + u(k) + w(k)$$

Let us choose a linear control

$$u(k) = \omega_d(k+1) - \omega(k) + k_v \dot{e}(k) + \frac{2}{T} k_p e(k)$$

where $e(k) = \theta_d(k) - \theta(k)$. Combining the above with a second order trapezoidal smoothing formula for the error $\dot{e}(k+1) + \dot{e}(k) = \frac{2}{T} [e(k+1) - e(k)]$ we get the error equation

$$e(k+2) + [k_v + k_p - 1] e(k+1) + [k_p - k_v] e(k) = \frac{T}{2} [w(k+1) + w(k)]$$

Again if we choose $k_v$ and $k_p$ such that the roots of the characteristic equation
$$z^2 + [k_v + k_p - 1]z + [k_p - k_v] = 0$$

lie inside the unit circle, we have a stable difference equation in $e(k)$. Note that even if there is no modelling error $w(k)$ is not zero. This has been explained in chapter 3 and is because the discrete models are generally not exact and discretization errors almost always exist. Hence $e(k)$ is not zero even though the parameters of the model are known perfectly. This is an important difference between the discrete time case and the continuous time case.

4.4. Simulations of Traditional Controllers

An ideal continuous time controller can usually only be implemented on an analog computer hardwired into the robot. In most cases the robot control laws are too complicated to be implemented on an analog computer and a digital computer is necessary. In real time control with a digital computer a forced discretization of the continous signals using sample and hold is necessary. In non real time simulations it is possible to integrate the controller into the dynamic structure of the robot and hence emulate the performance of an ideal continuous time controller. This is what we have done in some of our simulations of the continuous time controllers.

Simulations of traditional controllers are found in figures (4.1) - (4.34). These are on pages 41 through 74.

A simulation of the independent joint PD control on the cylindrical robot is shown in figures (4.1) - (4.6). $k_p$ and $k_v$ are chosen to give critical damping. In figures (4.1) - (4.6) we try to track a step input. It has been proven [5] that the con-
controller is stable in this case. By changing the magnitude of \( k_p \) one can change the natural frequency of the controller and hence the response time of the robot. Note that although the controller is critically damped the system as a whole is not. This is because the controller ignores the dynamics of the robot. Studying the graphs we notice an overshoot of the error for the critically damped controller. As expected increasing the gains increases the speed of response but also increases the forces required. Hence there is a tradeoff between speed of response and low torques.

In figure (4.4) - (4.6) the performance of the independent joint PD controller in tracking a time varying signal is shown. The signal to be tracked is combination of sine waves of frequencies between 0.1 and 2.0 Hz. This is going to be our standard desired signal to be tracked. We find that choosing a high enough gain allows us to track the time varying signal reasonably well. This is remarkable considering that no knowledge of the dynamics of the robot need be known in designing the controller. This time we notice that increasing gains not only speeds up the response but also decreases the magnitude of the error. As before the forces and torques required increase with controller gains.

A simulation of the computed torque continuous time control on the cylindrical robot is shown in figure (4.7) - (4.19). \( K_p \) and \( K_v \) are chosen to give critical damping in all cases. The signal to be tracked is a combination of sine waves as before. As mentioned before an ideal continuous time controller is implemented without any sampling. However practical realtime continuous controllers implemented on the digital computer must use some kind of sampling. Both the above have been simu-
lated. Figures (4.7) - (4.9) show the step response of the continuous time computed torque PD controller with no sampling. Unlike the independent joint PD controller the use of global linearization in the computed torque controller ensures that the system as a whole is critically damped. There is no overshoot in the error. Figures (4.10) - (4.12) show the tracking performance of the continuous time computed torque PD controller with no sampling. Again the tradeoff between high torques and low errors is clear. Figures (4.13) - (4.15) show the effect of using sampling on controller performance as shown in figures (4.10) - (4.12). Using a small sampling time $T=0.001$ does not change the performance significantly. Only the torques required are increased. With a larger sampling time $T=0.01$ the beginnings of instability are clear. With $T=0.1$ the closed loop system is unstable and the errors increase without bound.

Robustness of the controller with no sampling is studied in figures (4.16) - (4.18). It appears to be very robust since stability is maintained in spite of large ($400\%$) parameter errors. Figure (4.19) shows that using sampling significantly decreases the robustness.

A simulation of the computed torque discrete time control on the cylindrical robot is shown in figures (4.20) - (4.34). $k_p$ and $k_v$ are chosen to give critical damping. Figures (4.20) -(4.25) show step responses. Figures (4.26) - (4.31) show tracking performance. Different combinations of $k_p$ and $k_v$ are tried. The choice $k_p=0.405$ and $k_v=3.95$ appears to give a quick response, low maximum error and no oscillations. Using a smaller sampling time $T$ decreases the response time but increases the torques required. It is important to notice that the oscillations noticed in the continu-
ous time controller with $T=0.01$ are not present in the discrete time controller with the same sampling time. Hence it appears advantageous to design controllers in the discrete domain than to design in the continuous domain and then use sample and hold. The robustness of this controller to parameter variations is shown in figures (4.32) - (4.34). It is a lot less robust than the continuous time computed torque PD controller with no sampling. Sampling appears to affect the robustness of controllers.

The independent joint PD controller is very robust because it does not use any information about the robot.

In this chapter we investigated some traditional control systems. In the next chapter we shall provide the theoretical framework for designing $l^1$ optimal systems.
Step Response of Independent Joint PD Controller - No Sampling - kp=1/kv=2 - Fig (4.1)
Step Response of Independent Joint PD Controller - No Sampling - kp=100/kv=50 - Fig (4.2)
Step Response of Independent Joint PD Controller - No Sampling - \( kp=10000/kv=200 \) - Fig (4.3)
Independent Joint PD Controller - Tracking - No Sampling - kp=100/\nu=20 - Fig (4.4)
Independent Joint PD Controller - Tracking - No Sampling - kp=10000/kv=200 - Fig (4.5)
Independent Joint PD Controller - Tracking - No Sampling - kp=1000000/kv=2000 - Fig (4.6)
Computed Torque PD Controller - Step response - No Sampling - $kp=1$, $kv=2$ - Fig (4.7)
Computed Torque PD Controller - Step response - No Sampling - kp=100/kv=20 - Fig (4.5)

Desired Angular Displacement

Desired Radial Displacement

Angular Displacement Error

Radial Displacement Error

Torque Required

Force Required
Computed Torque PD Controller - Step response - No Sampling - kp=10000/kv=200 - Fig (4.9)
Computed Torque PD Controller - Tracking - No Sampling - \( kp=1/kv=2 \) - Fig (4.10)
Computed Torque PD Controller - Tracking - No Sampling - kp=100/kv=20 - Fig (4.11)

- Desired Angular Displacement
- Desired Radial Displacement
- Angular Displacement Error
- Radial Displacement Error
- Torque Required
- Force Required
Computed Torque PD Controller - Tracking - No Sampling - kp=10000/kv=200 - Fig (4.12)
Computed Torque PD Controller - Tracking - Sampling T=.001 - kp=10000/kv=200 - Fig (4.13)

- Desired Angular Displacement
- Desired Radial Displacement
- Angular Displacement Error
- Radial Displacement Error
- Torque Required
- Force Required
Computed Torque PD Controller - Tracking - Sampling $T=0.01$ - $kp=10000/kv=200$ - Fig (4.14)

- Desired Angular Displacement
- Desired Radial Displacement
- Angular Displacement Error
- Radial Displacement Error
- Torque Required
- Force Required
Computed Torque PD Controller - Tracking - Sampling T=1 - kp=10000/kv=200 - Fig (4.15)
Computed Torque PD Controller - Robustness Varying $J$ - True Value $J=10$ - No Sampling - $kp=10000/kv=200$ - Fig (4.16)
Computed Torque PD Controller - Robustness Varying M - True Value M=7 - No Sampling - kp=10000/kv=200 - Fig (4.17)
Computed Torque FD Controller - Robustness Varying $M_p$. True Value $M_p=2$. No Sampling. $kp=10000/kv=200$. Fig (4.18)
Computed Torque PD Controller - Robustness Varying J - True Value J=10 - Sampling T=.01 - kp=10000/kv=200 - Fig (4.14)
Discrete Computed Torque PD Controller - Step Response - $T=0.1, k_p=1, k_v=0.75$ - Fig (4.20)

- Desired Angular Displacement

- Desired Radial Displacement

- Angular Displacement Error

- Radial Displacement Error

- Torque Required

- Force Required
Discrete Computed Torque PD Controller - Step Response - $T_0 = 0.01/\text{kp} = 0.405/\text{kv} = 0.395$ - Fig (4.21)
Discrete Computed Torque PD Controller - Step Response - T=.01/kp=.005/kv=-.005 - Fig (4.22)

- Desired Angular Displacement
- Desired Radial Displacement
- Angular Displacement Error
- Radial Displacement Error
- Torque Required
- Force Required
Discrete Computed Torque PD Controller - Step Response - $T=0.01$, $kp=405$, $kv=395$ - Fig (4.24)
Discrete Computed Torque PD Controller - Step Response - T=0.01/KP=0.05/KV=0.05 - Fig (4.25)
Discrete Computed Torque PD Controller - Tracking - T=0.01/\(k_p=1.125/\lambda=0.875\) - Fig (4.26)
Discrete Computed Torque PD Controller - Tracking - $T=.01/k_p=.405/k_v=.395$ - Fig (4.27)
Discrete Computed Torque PD Controller - Tracking - T=.01/\text{k}_{p}=0.05/\text{k}_{v}=.805 - Fig (4.25)
Discrete Computed Torque PD Controller - Tracking - $T=.001/\text{kp}=1.125/\text{kv}=875$ - Fig (4.29)
Discrete Computed Torque PD Controller - Tracking - $T=0.001/\text{bp}=0.405/\text{kv}=0.295$ - Fig (4.30)
Discrete Computed Torque PD Controller - Tracking - $T=0.01$, $kp = 0.005$, $kv = 0.005$ - Fig (4.31)
Discrete Computed Torque PD Controller - Robustness Varying J - True Value of J=10 - T=.01/kp=.405/kv=.395 - Fig. (4.32)
Discrete Computed Torque PD Controller - Robustness Varying $M$ - True Value of $M=7$ - $T=.01$/kp=.405/kv=.395 - Fig. (4.33)
Discrete Computed Torque PD Controller - Robustness Varying $M$ - True Value of $M = 2$, $T = 0.1/k_p = 0.485/k_v = 0.395$ - Fig. (4.34)
CHAPTER 5

Results From $l^1$ Optimal Control

$l^1$ optimal control theory was recently developed at Rice University by M.A. Dahleh and J.B. Pearson and is a promising addition to control systems design. This chapter is not intended to be a review of the $l^1$ theory. Its intention is to state some of the results that we have used in this thesis. An extension of the theory is also developed. A complete treatment of $l^1$ theory can be found in [7, 8]. We consider only Single Input Single Output (SISO) systems. Later we shall see that this is all that we need to consider.

In this chapter we also set up a general framework for solving Multi Input Multi Output (MIMO) type control problems. We shall solve the robot $l^1$ optimal control problem using this framework.

5.1. $l^1$ Optimal Control

A discrete linear system is uniquely identified by a sequence $h$ which represents its impulse response. The $z$ transform of the impulse response is defined by

$$h(z) = \sum_{i=0}^{\infty} h_i z^i. \quad (5.1)$$

Note that by this definition a delay is represented by $z$ and not by $z^{-1}$ as is more common. $h(z)$ can be thought of as an operator that maps input sequences onto output sequences. $h(z)$ is considered BIBO stable if it maps all bounded sequences
(belonging to $L^\infty$) onto other bounded sequences. $h(z)$ is BIBO stable iff $\hat{h}$ is an $l^1$ sequence. We define the $A$ norm of $h(z)$ as being

$$\| h \|_A = \sup_{y \in B^*} \| \hat{h}^* y \|_{\infty} = \| \hat{h} \|_1.$$  \hspace{1cm} (5.2)

If $h(z)$ were the closed loop error transfer function of a digital system a good way to design a controller $c(z)$ would be to minimize $\| h \|_A$. This is a minimax design and is precisely what $l^1$ optimization does. $c(z)$ must also ensure system stability. To achieve these dual goals we first parametrize all stabilizing compensators in terms of a parameter $q(z)$ and then choose $q(z)$ to minimize $\| h \|_A$. The YJBK method [9] parametrizes all stabilizing compensators of a closed loop system in terms of a single stable parameter $q(z)$ called the Youla parameter. What is remarkable is that the closed loop transfer function is linear in $q(z)$.

If $p(z)$ is the plant transfer function then the closed loop transfer functions are of the type $(1 + pc)^{-1}$, $pc(1 + pc)^{-1}$, or $p(1 + pc)^{-1}$ depending on what the actual configuration is. All the above closed loop transfer functions can be parametrized in the form $(h - gq)$ where $h$ and $g$ are BIBO stable and determined by the plant $p$ and the configuration. $q$ is the Youla parameter and can be any BIBO stable rational function. Hence we have to choose a stable $q$ that minimizes $\| h - gq \|_A$. It is convenient to define $gq = k$. We now have to choose $k$ to minimize $\| h - k \|_A$. $k$ must not only be stable but it must have the same non minimum phase zeroes as $g$. If we did not impose this constraint then the non minimum phase zeroes of $g$ that did not appear in $k$ would have to be cancelled by unstable poles of $q$. This would make $q$ unstable and it could no longer be a Youla parameter. If $a_i$ for $i = 1, \ldots, n$ are the distinct real
non minimum phase zeros (inside the unit circle) of $g(z)$ then we have the constraint
$k \in S$ where

$$S = \left\{ k | \hat{k} \in l^1 \text{ and } <\hat{k}, \overline{a_i}> = 0 \quad i = 1, \ldots, n \right\} \quad (5.3)$$

$\overline{a}$ is the sequence $(1, a, a^2, \cdots)$ and $<x, y>$ represents a linear functional given by the
dot product of the two sequences $x$ and $y$. We now state the result

$$\inf_{k \in S} \| h - k \|_A = \left\{ \max_{\alpha} \left[ \sum_{i=1}^{n} \alpha_i h(a_i) \right] \text{ s.t. } -1 \leq \sum_{i=1}^{n} \alpha_i a_i^j \leq 1 \quad j = 0, 1, 2, \cdots \right\} \quad (5.4)$$

The inequalities are called the constraints. There are an infinite number of constraints. It would be very difficult to solve a problem with infinite constraints. However it has been proven that once a certain number of these constraints are satisfied all the rest will also be satisfied and can hence be ignored. There is no simple formula to calculate this number. The usual way to solve the problem is to start with a certain number of constraints and get a solution. We add additional constraints and get a new solution. This process is repeated until the solution obtained does not change.

This is the optimal solution. Solving equation (5.4) leads to a set of $\alpha_i$ and a norm $\mu$ corresponding to the optimal solution. These are used to generate the optimal solution $\phi = h - k$.

A useful property of the $l^1$ optimization process can now be stated. Multiplying $\phi(z) = h(z) - g(z)q(z)$ by functions of the type $kz^n$ has no effect on the optimization process. While $k$ being a constant just scales the norm and does not affect the optimal $q(z)$, the effect of $z^n$ can be seen by considering the resulting function $\overline{\phi}(z) = z^n \phi(z)$.
Notice that \( \| \phi \|_A = \| \tilde{\phi} \|_A \). Hence \( z^n \) does not affect the optimization either. This result is used in reverse and we cancel out the common constants and powers of \( z \) before optimizing.

Unfortunately in the robot problem the non minimum phase zeroes are repeated. The results have to be extended to account for the repeated zero case. Let \( g(z) \) have a \( m^{th} \) order zero at \( z = a_1 \). \( k(z) \) must also have a \( m^{th} \) order zero at \( z = a_1 \). This implies that \( k(a_1) = 0, k^1(a_1) = 0, \ldots, k^{m-1}(a_1) = 0 \), where \( k^i = \frac{d^i k}{dz^i} \). Hence \( S \) in equation (5.3) needs to be redefined as

\[
S = \left\{ k | \hat{k} \in l^1 \text{ and } \langle \hat{k}, \bar{a}_i^j \rangle = 0 \quad \text{for} \quad i = 1, \ldots, m \right\} \quad \left\{ \langle \hat{k}, \bar{a}_i^j \rangle = 0 \quad \text{for} \quad i = m+1, \ldots, n \right\}
\]  

(5.5)

where the \( k^{th} \) element of \( \bar{a}_i^j \) is \( P_k^{j-1} a_1^{k-i+1} \) and

\[
P_j^i = \begin{cases} 0 & \text{if } j<i \\ \frac{j!}{(j-i)!} & \text{if } j\geq i \end{cases}
\]

Conditions on the derivative of \( k \) lead to conditions on the derivatives of \( h \). The result stated in equation (5.4) has to be restated as

\[
\inf_{k \in S} \| h - k \|_A = \begin{cases} \max_{i=1}^{m} \alpha_i h^{i-1}(a_1) + \sum_{i=m+1}^{n} \alpha_i h(a_i) \\ \exists \ -1 \leq \sum_{i=1}^{m} \alpha_i P_j^{i-1} a_1^{j-i+1} + \sum_{i=m+1}^{n} \alpha_i a_i^j \leq 1 \quad j=0,1,2,\ldots \end{cases}
\]

(5.6)
The above optimization problems can be solved by using standard linear programming techniques with some modifications. There are Fortran programs we have written that automatically set up the problem up and solve it. One program solves the $l^1$ optimization for the distinct zero case while another solves the repeated zero case. We input the non minimum phase zeros called the interpolation points, the values of $h(z)$ evaluated at these interpolation points, and the number of constraints that should be considered. The output of the programs is the coefficients of the transfer function $\phi(z)$ which is the optimal value of $h(z) - k(z)$. We calculate $q(z)$ using

$$q(z) = \frac{(h(z) - \phi(z))}{g(z)}. \quad (5.7)$$

A good check is to make sure that $q(z)$ thus obtained is stable.

5.2. Framework For Solving MIMO Control Problems

A general MIMO plant can be represented by the equation

$$\begin{pmatrix} y \\ z \end{pmatrix} = \begin{bmatrix} -P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{pmatrix} u \\ e \end{pmatrix} \quad (5.8)$$

where $y$ is the measured output vector, $z$ is the regulated output vector, $u$ is the control input vector, $e$ is the exogenous input vector, and $P_{ij}$'s are rational transfer function matrices. The objective is to design a feedback compensator $C(z)$, where $u = Cy$, such that the closed loop system is internally stable. In our problem we also want the effect of $e$ on $z$ to be minimized.

Left and right coprime factorizations of $P_{11}$ are given by
\[ P_{11} = A^{-1}B = B_1A_1^{-1} \] (5.9)

If we assume the system to be admissible [10] then all controllers that stabilize the system can be parametrized as

\[ C = (Y + A_1Q)(X - B_1Q)^{-1} \] (5.10)

where \( Q \), the Youla parameter, is any stable matrix and \( X \) and \( Y \) satisfy the Bezout identity

\[
\begin{pmatrix}
A & B \\
-B_1 & X
\end{pmatrix}
\begin{pmatrix}
X & -B_1 \\
Y & A_1
\end{pmatrix}
= 
\begin{pmatrix}
I & 0 \\
0 & I
\end{pmatrix}
\] (5.11)

The above equations give us the transfer function between the regulated output \( z \) and the exogenous input \( e \) as

\[ z = (P_{21}A_1QP_{12} + P_{21}YAP_{12} + P_{22})e = \Phi e \] (5.12)

In the robot problem we choose \( Q \) to minimize \( \| \Phi \|_A \) and then calculate \( C \) from equation (5.10).

The theoretical framework developed for general MIMO systems and \( l^1 \) optimization in this chapter will be used in the next chapter to provide the theory and design procedure for \( l^1 \) optimal control systems for robots.
CHAPTER 6

Optimal $l^1$ Controller for a Robot

In this chapter we start with the linearized model of the robot and develop the theory to design a $l^1$ optimal controller for it. Consider equations (3.18) and (3.19). $N$ such equations represent the Greenspan model for a $n$ degree of freedom robot. As has already been stated in chapter 3 each of these linearized equations can be controlled independently. The design procedure is developed for one such decoupled controller. Using this procedure we design the $n$ independent controllers. Although we use the linearized Greenspan model the theory holds for any linearized robot model. Actually the theory is general enough to design a $l^1$ optimal controller for any nonlinear system linearized using global linearization.

6.1. Transfer Function for the Linear Model

Equations (3.18) and (3.19) are repeated below

$$
\begin{bmatrix}
\theta(k+1) \\
\omega(k+1)
\end{bmatrix}
= A \begin{bmatrix}
\theta(k) \\
\omega(k)
\end{bmatrix}
+ B \begin{bmatrix}
0 \\
u(k) + w(k)
\end{bmatrix}
$$

where

$$
A = \begin{bmatrix}
1 & T \\
0 & 1
\end{bmatrix} \quad B = \begin{bmatrix}
T \\
\frac{T}{2} \\
1
\end{bmatrix} \quad C = \begin{bmatrix}
1 & 0
\end{bmatrix}
$$

The plant transfer function $P(z)$ is given by
\[ P(z) = \frac{\theta(z)}{u(z) + w(z)} = C(zI - A)^{-1}B = \frac{T}{2} \frac{(z+1)}{(z-1)^2} \]

where \( z^{-1} \) represents a delay. Since our \( l^1 \) theory has been developed using \( z \) as a delay let us transform the above equation using \( z = z^{-1} \) to the \( z \) delay form

\[ P(z) = \frac{T}{2} \frac{z(z+1)}{(z-1)^2} \quad (6.3) \]

6.2. The Complete MIMO Robot Model and Optimization

The robot has to be written in the general MIMO framework as given by equation (5.8)

\[
\begin{bmatrix}
y \\
z
\end{bmatrix} = \begin{bmatrix}
-P_{11} & P_{12} \\
P_{21} & P_{22}
\end{bmatrix}
\begin{bmatrix}
u \\
e
\end{bmatrix}
\quad (5.8)
\]

where \( y \) is the measured output vector, \( z \) is the regulated output vector, \( u \) is the control input vector, \( e \) is the exogenous input vector, and \( P_{ij} \)'s are rational transfer function matrices. The robot equations in this framework are

\[
\begin{align*}
y &= \begin{bmatrix}
\theta_d(z) \\
\theta(z)
\end{bmatrix} \\
e &= \begin{bmatrix}
\theta_d(z) \\
w(z)
\end{bmatrix} \\
z &= \begin{bmatrix}
\theta_d(z) - \theta(z)
\end{bmatrix} \\
u &= \begin{bmatrix}
u(z)
\end{bmatrix}
\quad (6.4)
\end{align*}
\]

The \( P_{ij} \)'s are as follows

\[
\begin{align*}
P_{11} &= \begin{bmatrix}
0 \\
-P(z)
\end{bmatrix} \\
P_{12} &= \begin{bmatrix}
1 & 0 \\
0 & P(z)
\end{bmatrix} \\
P_{21} &= \begin{bmatrix}
-P(z)
\end{bmatrix} \\
P_{22} &= \begin{bmatrix}
1 & -P(z)
\end{bmatrix}
\quad (6.5)
\end{align*}
\]

where \( P(z) \) is as in equation (6.3). We now obtain left and right coprime factorizations of \( P_{11} \) as given by equation (5.9). The required \( A, B, A_1 \) and \( B_1 \) are given by
\[ A(z) = \begin{bmatrix} 1 & 0 \\ 0 & (z-1)^2 \end{bmatrix} \quad B(z) = \begin{bmatrix} 0 \\ \frac{T}{2} z(z+1) \end{bmatrix} \]

\[ A_1(z) = (z-1)^2 \quad B_1(z) = B(z) \]

Next we obtain \(X, Y, X_1\), and \(Y_1\) to satisfy the generalized Bezout identity as given by equation (5.11). They are as follows

\[ X(z) = \begin{bmatrix} 1 & 0 \\ 0 & \frac{(3z+4)}{4} \end{bmatrix} \quad Y(z) = \begin{bmatrix} 0 \\ \frac{T}{2} \frac{(3z-5)}{4} \end{bmatrix} \]

\[ X_1(z) = \frac{(3z+4)}{4} \quad Y_1(z) = Y(z) \]

Equations (6.6) and (6.7) have been obtained using programs written at Rice University by Bor-Chin Chang [11]. Using equation (5.12) we can write

\[ \Phi = P_{22} + P_{21}YA_P_{12} + P_{21}A_1QAP_{12} \]

where \(Q\), the Youla parameter, is made up of two scalar parameters \(Q = [Q_1, Q_2]\). \(\Phi\) is a 1\(\times\)2 matrix and can be written as

\[ \Phi = \begin{bmatrix} \Phi_1 & \Phi_2 \end{bmatrix} \]

Using equation (6.9) we can write down \(\Phi_1\) and \(\Phi_2\) as

\[ \Phi_1 = 1 - \left[ \frac{T}{2} z(z+1) \right] Q_1(z) \]
\[
\Phi_2 = \frac{-T}{2} \frac{z(z+1)(3z+4)}{4} - \left[ \frac{T^2}{4} z^2(z+1)^2 \right] Q_2(z)
\]  

(6.12)

A very important fact to notice is that \( \Phi_1 \) is a function of \( Q_1 \) only and \( \Phi_2 \) is a function of \( Q_2 \) only. Further consider \( \phi \) the impulse response of \( \Phi \). A property of the \( l^1 \) norm is that the matrix norm of a row vector is equal to the sum of the scalar norms of the individual elements. Hence

\[
\| \Phi \|_1 = \| \Phi_1 \|_1 + \| \Phi_2 \|_1 \quad \text{and} \quad \| \Phi \|_A = \| \Phi_1 \|_A + \| \Phi_2 \|_A
\]

(6.13)

Minimising \( \| \Phi \|_A \) is equivalent to minimising \( \| \Phi_1 \|_A \) and \( \| \Phi_2 \|_A \) independently. This along with the fact that \( \Phi_1 \) is a function of \( Q_1 \) only and \( \Phi_2 \) is a function of \( Q_2 \) only means that we have effectively split the MIMO problem into two independent SISO problems. This is of fundamental importance in this thesis. Firstly solving a MIMO \( l^1 \) problem is a lot more complicated than solving a SISO problem. Secondly \( Q_1 \) and \( Q_2 \) can be chosen independently. This means that there is no tradeoff involved between optimizing \( \phi_1 \) and optimizing \( \phi_2 \).

As far as the optimization is concerned it is almost as if equations (6.11) and (6.12) represent two different SISO systems. They are each of the form required by the results in chapter 5 and can be solved by the SISO programs written.

6.3. Construction of the Optimal Controller

Optimization of \( \Phi_1 \) and \( \Phi_2 \) lead to certain \( Q_1 \) and \( Q_2 \) corresponding to the optimal solution. These are used to construct the optimal compensator \( C(z) \) which is a \( 1 \times 2 \) matrix given by \( C = [C_1 \quad C_2] \) such that
\[ u = C y = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} \theta_d \\ \theta \end{bmatrix} \] (6.14)

\[ C_2 = \frac{\frac{2}{T} (3z-5) + 4(z-1)^2 Q_2(z)}{(3z+4) + 4 \frac{T}{2} z(z+1) C_2(z)} \] (6.15)

\[ C_1 = Q_1(z) \left[ (z-1)^2 - \frac{T}{2} z(z+1) C_2(z) \right] \] (6.16)

Since \( Q_1(z) \) and \( Q_2(z) \) are polynomials we can write

\[ C_1(z) = \frac{N_1(z)}{D(z)} \qquad C_2(z) = \frac{N_2(z)}{D(z)} \] (6.17)

where \( N_1(z) \), \( N_2(z) \) and \( D(z) \) are polynomials in \( z \). Let

\[ N_1(z) = \sum_{i=0}^{i=m} a_i z^i \quad N_2(z) = \sum_{i=0}^{i=n} b_i z^i \] (6.18)

\[ D(z) = \sum_{i=0}^{i=p} c_i z^i \] (6.19)

Since

\[ u(z) = C_1(z) \theta_d(z) + C_2(z) \theta(z) \] (6.20)

the control input \( u(n) \) is given by the difference equation

\[ u(n) = \frac{1}{c_0} \left\{ \sum_{i=0}^{i=m} a_i \theta_d(n-i) + \sum_{i=0}^{i=n} b_i \theta(n-i) - \sum_{i=1}^{i=p} c_i u(n-i) \right\} \] (6.21)
6.4. Effect of Windowing Functions

A window is a stable transfer function with a certain frequency response. Passing a signal through a window is called windowing. Low pass filters and high pass filters are examples of windows. We have so far not considered any windowing on the exogenous inputs. Systems designed without windows assume that nothing is known about the frequency content of the exogenous inputs and hence the input is equally likely to have any frequency. Optimal systems thus designed are not biased towards a particular frequency range. However very often we have some idea about the frequency content of the exogenous input. For instance it is unlikely that the robot will be required to track frequencies higher than 3 Hertz. We can use a low pass filter with a cut off frequency of 3 Hertz as a windowing function for $\theta_d(z)$. The optimal system designed would now penalize the effect of low frequencies much more than high frequencies. The weighting at different frequencies is exactly the frequency response of the window. Any frequency content information of the input should be used in the form of windows and should lead to better controllers.

The effect of windows on the optimization procedure is very straightforward. An input window results in multiplying the transfer function by the transfer function of the window $W$. Instead of optimizing $\|H - GQ\|$ we now optimize $\|\tilde{H} - \tilde{G}Q\|$ where $\tilde{H} = WH$ and $\tilde{G} = WG$.

To be able to compare results we shall always normalize windows to have a $A$ norm of 1.
6.5. Effect of Interpolation Points on the Unit Circle

If there are zeroes on the unit circle, it is necessary to interpolate at these points too. Although the results of equations (5.4) and (5.6) still hold at the boundary (on the unit circle) we can no longer guarantee that a finite number of constraints are required. This is because the magnitude of $a_j^j$ decreases with increasing $j$ only if the magnitude of the non minimum phase zero, $a_p$, is strictly less than one and not if it is equal to one. Since we have zeroes on the unit circle in the linearized robot model we have to contend with this problem.

A solution involves a scaling transformation of the $z$ plane. Using a change of variables of the type $w=az$ where $a<1$ the zeroes on the unit circle in the $z$ domain are now inside the unit circle in the $w$ domain. We perform the optimization in the $w$ domain, get the solution and then transform it back into the $z$ domain. This method also leads to a suboptimal solution. The closer $a$ is to 1 the closer is the suboptimal solution to the optimal one. We shall use the second method to solve the robot control problem.

6.6. Some Example Designs

In this section we shall solve some example problems using the theory developed in this chapter.

Example 1

The simplest design problem is to try and optimize $\Phi_1$ and $\Phi_2$ as given by equations (6.11) and (6.12) without using any windowing functions.
Consider

\[ \Phi_1 = 1 - \left[ \frac{T}{2}z(z+1) \right]Q_1(z) \]  
(6.22)

Use the transformation \( z = \frac{w}{0.9} \) and \( T = 0.01 \) to get

\[ \Phi_1(w) = 1 - \frac{0.01}{2 \times 0.9^2}w(w+.9)Q_1(w) \]  
(6.23)

The interpolation points are \( w = 0 \) and \( -0.9 \) and the values of \( H \) at these points are \( H(0) = 1 \) and \( H(-0.9) = 1 \). These are fed into L1opt.f to give the optimal \( \Phi(z) = 1 \). Hence the optimal \( Q_1(w) = 0.0 \). Substituting \( w = 0.9z \) we get \( Q_1(z) = 0.0 \).

Take equation (6.12), substitute \( z = \frac{w}{0.9} \)

and \( T = 0.01 \), cancel constants and common factors, to get

\[ \Phi_2(w) = \frac{(w+.9)(3w+3.6)}{4} + \frac{0.01}{2 \times 0.9}w(w+.9)^2Q_2(w) \]  
(6.23)

There is a double interpolation point at \( w = 0.9 \) and the other interpolation point is at \( w = 0 \). The values of \( H \) and its derivatives are \( H(-0.9) = 0.0 \), \( H^1(-0.9) = 0.225 \) and \( H(0) = 0.81 \). Running L1optrep.f we get the optimal \( \Phi(w) \) as

\[ \Phi(w) = 0.81 + 0.95625w - 0.199165w^{13} \]  
(6.24)

Substituting equation (6.24) in (6.23) we get \( Q_2(w) \) and using \( w = 0.9z \) we get

\[ Q_2(z) = -35.9 \left\{ \frac{3.84 - 3.49z + 3.14z^2 - 2.79z^3 + 2.44z^4}{-2.09z^5 + 1.74z^6 - 1.40z^7 + 1.05z^8 - 0.70z^9 + 0.35z^{10}} \right\} \]  
(6.25)

We use the values of \( Q_1(z) \) and \( Q_2(z) \) in equation (6.15) and equation (6.16) to get
\(C_1(z)\) and \(C_2(z)\)

\[
C_2(z) = \begin{cases}
-1550 + 2200z - 2000z^2 + 1800z^3 \\
-1600z^4 + 1400z^5 - 1200z^6 + 1000z^7 \\
-800z^8 + 600z^9 - 400z^{10} + 200z^{11} - 50z^{12}
\end{cases}
\]

\[
= \begin{cases}
4 + 0.25z - 0.25z^2 + 0.25z^3 \\
-0.25z^4 + 0.25z^5 - 0.25z^6 + 0.25z^7 \\
-0.25z^8 + 0.25z^9 - 0.25z^{10} + 0.25z^{11} - 0.25z^{12}
\end{cases}
\]

\(C_1(z) = 0\)

Notice that \(C_1(z)\) is a zero compensator. Using this would isolate \(\theta_d\) from the rest of the control system. This is clearly not desirable. Unfortunately without using windowing this is the best that we can do. Some understanding of this can be had if we notice that a step input would pass unchanged if there was no filtering window. The initial error to a step would be one, and hence this is the best performance index that can be achieved. In the next example we shall use windowing functions.

**Example 2**

Let us use a windowing function on \(\Phi_1\) as

\[
W_1(z) = \frac{1}{K} \left( \frac{1}{1 - \alpha z} \right)^n
\]

where \(K\) is chosen such that \(\|W_1(z)\|_A = 1\). Using a positive value of \(\alpha\) gives us an \(n^{th}\) order low pass filter. Hence
\[ \Phi_1(z) = \left[ \frac{1}{K(1-\alpha z)} \right]^n - \frac{T}{2} z(x+1) \left[ \frac{1}{K(1-\alpha z)} \right]^n Q_1(z) \]  
(6.27)

Take equation (6.27), substitute \( z = \frac{w}{a} \) and \( T = 0.01 \) to get

\[ \Phi_1(w) = \frac{a^n}{K(a-\alpha w)^n} - \frac{0.005a^{n-2}w(w+a)}{K(a-\alpha w)^n} Q_1(w) \]  
(6.28)

The interpolation points are \( w = 0 \) and \( w = -\alpha \) and the values of \( H \) at these points are \( H(0) = \frac{1}{K} \) and \( H(-\alpha) = \frac{1}{K(1+\alpha)^n} \). Let us choose \( \alpha = 0.9 \), \( \alpha = 0.5 \), and \( n = 2 \). This gives us \( K = 4.0 \), \( H(0) = 1/4 \) and \( H(-0.9) = 0.444444/4 \). These are fed into L1opt.f to get the optimal \( \Phi_1(w) \) as

\[ \Phi_1(w) = 1/4 + 0.1728/4w \]  
(6.29)

with a performance index of 0.40432. Substituting this in equation (6.28) and using \( w = 0.9z \) we get

\[ Q_1(z) = 200 \{ 0.444444 - 0.13889z \} \]  
(6.30)

The windowing function that we shall use on \( \Phi_2 \) is

\[ W_2(z) = \left[ \frac{1}{K(1+\beta z)} \right] \]  
(6.30)

where \( K \) is chosen such that \( \| W_2(z) \|_A = 1/K(1-|\beta|) = 1 \). Using a positive value of \( \beta \) gives us a first order high pass filter. Notice that transforming a low pass filter to a high pass filter or vice versa is just a matter of changing signs of \( \alpha \) or \( \beta \). Using the windowing function we get
\[ \Phi_2(z) = \frac{T}{2} \frac{z(z+1)(3z+4)}{4K(1+\beta z)} - \frac{T^2}{4} \frac{z^2(z+1)^2}{K(1+\beta z)} Q_2(z) \]  

(6.31)

Take equation (6.31), substitute \( \frac{w}{a} \) to get

\[ \Phi_2(w) = \frac{T}{8a^2 K} \frac{(w+a)(3w+4a)}{(a+\beta w)} - \frac{T^2}{4a^3 K} \frac{w(w+a)^2}{(a+\beta w)} Q_2(w) \]  

(6.32)

There is a double interpolation point at \( w=-a \) and the other interpolation point is at \( w=0 \). The values of \( H \) and their derivatives are \( H(-a)=0.0, H'(-a)=\frac{T}{8a^2 K} \frac{1}{1-\beta} \) and \( H(0)=\frac{T}{8a^2 K} 4a \). Let us use \( \beta=.9, a=.9 \) and \( T=.01 \). Hence \( K=10.0 \). Running \texttt{L1optrep.f} we get the optimal \( \Phi(w) \) as

\[ \Phi_2(w) = 10^{-4} \left\{ -5.55552 + 3.429333w^2 - 3.81037w^3 \right\} \]  

(6.33)

Hence

\[ \Phi_2(w) = 10^{-4} \left\{ -5.55552 + 2.777778w^2 - 2.777778w^3 \right\} \]

and \( \| \Phi_2(z) \| = 1.1111\times10^{-3} \). Substituting equation (6.33) in equation (6.32) and using \( w=.9z \) we get

\[ Q_2(z) = 100 \left\{ -1.7 + .9z \right\} \]  

(6.34)

We use the values of \( Q_1(z) \) and \( Q_2(z) \) in equation (6.15) and equation (6.16) to get \( C_1(z) \) and \( C_2(z) \)
\[ C_2(z) = \frac{-1680 + 2320z - 1400z^2 + 360z^3}{\left\{ 4 - .4z - 1.6z^2 + 1.8z^3 \right\}} \]  \hspace{1cm} (6.35)

and

\[ C_1(z) = 800 \frac{\{.44444 - .13889z\}}{\left\{ 4 - .4z - 1.6z^2 + 1.8z^3 \right\}} \]  \hspace{1cm} (6.36)

Using windowing functions leads to more useful compensators. The design procedure used in the examples will be followed in the next chapter to design controllers to be used in the simulations.

6.7. Generalized Theory

We shall show that the decoupling principle that was crucial in developing the theory in this chapter, holds not only for the linearized Greenspan robot model as given by equation (6.3) but for any arbitrary linearized plant \( P(z) \) as given by

\[ P(z) = \frac{P_a(z)}{P_d(z)} \]  \hspace{1cm} (6.37)

Equations (6.4) and (6.5) still hold and are repeated as

\[
\begin{align*}
   y &= \begin{bmatrix} \theta_d(z) \\ \theta(z) \end{bmatrix} \\
   e &= \begin{bmatrix} \theta_d(z) \\ w(z) \end{bmatrix} \\
   z &= \begin{bmatrix} \theta_d(z) - \theta(z) \end{bmatrix} \\
   u &= \begin{bmatrix} u(z) \end{bmatrix} \\
\end{align*}
\]  \hspace{1cm} (6.4)

\[
\begin{align*}
   P_{11} &= \begin{bmatrix} 0 & 0 \\ -P(z) & 0 \end{bmatrix} \\
   P_{12} &= \begin{bmatrix} 1 & 0 \\ 0 & P(z) \end{bmatrix} \\
   P_{21} &= \begin{bmatrix} -P(z) \end{bmatrix} \\
   P_{22} &= \begin{bmatrix} 1 & -P(z) \end{bmatrix} \\
\end{align*}
\]  \hspace{1cm} (6.5)
Consider $x(z)$ and $y(z)$ that satisfy the scalar Bezout identity

$$P_d(z)x(z) - P_n(z)y(z) = 1$$  \hspace{1cm} (6.38)

The required $A, B, A_1$ and $B_1$ are given by

$$A(z) = \begin{bmatrix} 1 & 0 \\ 0 & P_d(z) \end{bmatrix} \hspace{1cm} B(z) = \begin{bmatrix} 0 & -P_n(z) \end{bmatrix}$$  \hspace{1cm} (6.39)

$$A_1(z) = \begin{bmatrix} P_d(z) \end{bmatrix} \hspace{1cm} B_1(z) = B(z)$$

$X, Y, X_1$, and $Y_1$ are chosen to satisfy the generalized Bezout identity as given by equation (5.11). They are as follows

$$X(z) = \begin{bmatrix} 1 & 0 \\ 0 & x(z) \end{bmatrix} \hspace{1cm} Y(z) = \begin{bmatrix} 0 & y(z) \end{bmatrix}$$  \hspace{1cm} (6.40)

$$X_1(z) = \begin{bmatrix} x(z) \end{bmatrix} \hspace{1cm} Y_1(z) = Y(z)$$

Using equation (5.12) we can write

$$\Phi_1(z) = 1 + P_n(z)Q_1(z)$$  \hspace{1cm} (6.41)

$$\Phi_2(z) = -P_n(z)x(z) + P_n(z)^2Q_2(z)$$  \hspace{1cm} (6.42)

From equations (6.41) and (6.42) we see that the decoupling principle holds for a general plant in the same framework as the robot Greenspan model.

The theory developed in this chapter will be used in the next chapter to design and simulate a $l^1$ optimal controller for the cylindrical robot. The next chapter will also contain comparisons and conclusions of the research.
CHAPTER 7

Simulations, Comparisons and Conclusions

In this chapter we shall design $l^1$ optimal controllers using the procedure detailed in chapter 6. We shall simulate control of a cylindrical robot as in chapter 4. The scheme for these simulations is explained in figure (7.1) on page 104. These will then be compared to the results obtained using traditional control systems. The optimality of the $l^1$ optimal controllers will be explained and clearly shown in simulations. The rest of the chapter will contain the conclusions drawn from this research and possible directions of further research.

7.1. Simulations of Optimal Controllers

The design procedure for designing $l^1$ optimal controllers has been explained in chapter 6. The important thing in this chapter is to choose the right windowing functions. As mentioned before a good way to choose them is to let $W_1(z)$ represent the frequency content of $\theta_d(z)$ and $W_2(z)$ represent the frequency content of $w(z)$.

Since $\theta_d(z)$ is usually a low frequency signal we choose $W_1(z)$ to be a low pass filter as

$$W_1(z) = \frac{1}{K} \left( \frac{1}{1-\alpha z} \right)^n$$  (7.1)

where $\|W_1(z)\|_A = 1$. Let us use $\alpha = .8$ and $n = 5$. This gives us an optimal
\[ \Phi_1(z) = 10^{-4} (3.21 + 3.04z) \] with a performance index \[ \| \Phi_1(z) \|_A = 6.25 \times 10^{-4}. \] The Youla parameter is given by

\[ Q_1(z) = 94.60 \times \left\{ 6.4543 - 11.976z + 9.9858z^2 - 4.064z^3 + .6561z^4 \right\} \tag{7.2} \]

The frequency content of \( w(z) \) is uncertain. Although in most of our simulations we have found it to be predominantly high frequency, there have been significant components at lower frequencies in certain cases. Hence we use only a first order high pass filter of the form

\[ W_2(z) = \left[ \frac{1}{K(1 + \beta z)} \right] \tag{7.3} \]

where \[ \| W_2(z) \|_A = 1/\sqrt{1 - |\beta|} = 1. \] Let us choose \( \beta = .75. \) In our simulations we have seen that choosing a different \( \beta \) does not significantly affect performance. The above choice leads to an optimal \( \Phi_2(z) \) given by

\[ \Phi_2(z) = 10^{-3} \{ -1.3889 - 1.3889z \} \tag{7.4} \]

with a performance index of \( 2.77778 \times 10^{-3}. \) The Youla parameter is \( Q_2(z) = 0.0. \) Hence

\[ C_2(z) = \frac{200(3z-5)}{(3z+4)} \tag{7.5} \]

\[ C_1(z) = \frac{Q_1(z) \times 4}{(3z+4)} \tag{7.6} \]

Simulations of the above optimal controller controlling the cylindrical robot is shown in figures (7.2) - (7.4) on pages 105 - 107. Figure (7.2) shows the step response and
figure (7.3) shows tracking performance. The desired trajectory is a combination of low frequency sine waves as usual. Figure (7.4) shows the robustness of the controller to parameter variations.

Increasing the order of the windowing function used we could further decrease $\|\Phi_1(z)\|_A$, and thus improve performance for the particular class of inputs. However this would not only increase the order of the controller but also place a stronger restriction on the kind of inputs we could track. Simulations show that choosing a different high pass filter for $W_2(z)$ does not significantly affect the performance of the controller.

The above figures are to be compared with performance of traditional controllers. However the PD controllers all need to use the derivative of the desired signal $\dot{\theta}_d$. This information is often not available to the controller and the optimal controller does not use this information. Hence for purposes of comparison we shall simulate in this chapter, the discrete time computed torque PD controller without using $\dot{\theta}_d$. These are shown in figures (7.5) - (7.8) on pages 108 - 111. Figures (7.5) - (7.7) show the tracking performance of the controllers for different combinations of $k_p$ and $k_r$. Figures (7.8) shows the robustness, to parameter variations, of the controller with $k_p = .405$ and $k_r = .395$.

We find that tracking performance of the PD controllers are significantly affected by not using the derivative of the desired signal. The steady state tracking error of the optimal controller is now smaller than those of the computed torque PD controllers. Further the optimal controller is more robust in that it is stable over wider
ranges of parameter variation.

7.2. Optimality of the Optimal Controller

It is not clear from the comparisons in the previous section what the optimality of the $l^1$ optimal controller is. To see the optimality let us concentrate on the tracking performance alone, since verification is easier in this case. Consider the closed loop transfer function of any of the linearized controllers. Considering only tracking performance we have

$$\Phi(z) = \frac{e(z)}{d(z)} \quad \theta_d(z) = W_1(z) d(z)$$

(7.7)

where $e$ is the tracking error, $\theta_d$ is the signal to be tracked, $d(z)$ is any $l^\infty$ signal such that $\|d\|_\infty \leq 1$ and $W_1(z)$ is the window which represents the information that we have about $\theta_d$. Since all the windows we use are normalized, $\theta_d$ is also a $l^\infty$ signal like $d$ with $\|\theta_d\|_\infty = 1$. However it falls into a subset of all $l^\infty$ signals such that it is the output of a stable filter $W_1$. For instance if we want to track only low frequency signals then we would choose $W_1$ to be a low pass filter.

Coming back to optimality, for each $\Phi$ there exists a $d$ such that we get the worst case error with $\|e\|_\infty = \|\phi\|_1$. The system with the smallest worst case error is the $l^1$ optimal system. To demonstrate optimality we have to find the input for each system that leads to the worst output and use this input in the simulation. The optimal system should have a error with the smallest $l^\infty$ norm for this input. The question then is to find the input that leads to the worst error. If the closed loop impulse response of a system is the sequence $(a_0, a_1, \ldots, a_n)$, this input is given by any sequence that has
\((\text{sgn}[a_n], \text{sgn}[a_{n-1}], \ldots, \text{sgn}[a_0])\) as a subsequence. The impulse response \(\phi\) can either be theoretically or experimentally determined.

Consider \(W_1 = \frac{1}{5(1-0.8z)}\). The optimal controller has \(Q_1(z) = 71.1144\) and \(\Phi_1(z) = 0.2 + 0.088893z\) with a norm of 0.28889. The response of this controller to an input which leads to the worst error is shown in figure (7.9) on page 112. As expected the error has a \(l^\infty\) norm of about 0.29. The response of a discrete time computed torque PD controller to an input that leads to its worst error is shown in figure (7.10) on page 113. The error has a \(l^\infty\) norm of about 0.55 which is more than the optimal norm of 0.29. To confirm our results we use the input, that produced the error norm of 0.55 in the PD controller, on our optimal controller. The response of the optimal controller is shown in figure (7.11) on page 114. The \(l^\infty\) norm of the error is about 0.29 which confirms our theoretical result. Notice that the input in figure (7.11) is also a worst case input for the optimal controller. Figures (7.12) on page 115 and (7.13) on page 116 show the corresponding results for the independent joint PD controller. Again we notice that the optimal controller has a smaller \(l^1\) norm for the worst case error.

This section and the accompanying simulations demonstrate the \(l^1\) optimal controller as one that minimises the worst possible error of a system.

7.3. Conclusions

We investigated several discrete time dynamic models of the robot. The Green-span model appears to be the best choice for control applications. While it is more accu-
rate than the first order Euler model, controllers designed using it are no more complicated than those using first order Euler models. Hence the Greenspan model is preferred to the first order Euler model. A possible rival to the Greenspan model was the Neuman-Tourassis model. However as we have shown it displays non unique solutions and this would make it suspect in any control applications.

Global linearization of the Greenspan model of the robot is straightforward and simple. Hence research that is being done in developing generalized global linearization schemes for non linear systems appears irrelevant to our robot model.

For most ordinary trajectories the traditional control systems appear to track pretty well. In fact the responses of some of the computed torque PD controllers as shown by the simulations in chapter 4 have smaller steady state errors than the responses in the simulations of the optimal controller as shown in chapter 7. Why then should anyone want to build an optimal controller. Firstly the controllers in chapter 4 require knowledge of the derivative of the desired trajectory. When deprived of this knowledge the responses of the traditional controllers worsen considerably and the optimal controller performs better. The second reason is much more important. The $l^1$ optimal controllers have been designed to give the best worst case performance, and not the best performance for all inputs. As shown in this chapter, whatever the input to be tracked, the error of the optimal controller is always below a certain optimal value. The traditional controllers on the other hand have much larger errors for certain inputs. Hence if we really care about worst case performance then $l^1$ controllers for robots is preferable to traditional controllers. This would be so if
the robot being controlled was in a high risk environment like a nuclear power plant or a space ship. Thirdly our optimal controller minimises the effect of the modelling error input \( w(k) \). Our simulations show that the optimal controllers are more robust to parameter variations than the comparable discrete time computed torque PD controllers.

In this thesis we have provided the framework and design procedure for building \( l^1 \) optimal controllers for robots. Our simulations have shown that these controllers are indeed optimal and perform as designed. Coupled with the motivation to minimise worst case tracking errors, this thesis should provide useful new tools for the robot control engineer.

7.4. Topics for Further Research

Alternate Boundary Interpolation

When we had zeros on the unit circle as interpolation points, we used a linear scaling transformation to get finite length solutions. These are sub optimal solutions. An alternate approach is to consider only a finite number of constraints without using any transformations of the \( z \) plane. A good choice is to stop when additional constraints do not change the norm \( \mu \) appreciably. A solution \( \phi \) thus constructed is also sub optimal but can be made as close to the optimal solution as we desire. It would be interesting to compare the solutions using this alternate approach to the solutions we already have.

Measurement and Prediction of the Modelling Error
The modelling error $W(k)$ as given by equation (3.13) at first glance appears to be impossible to determine. But consider equation (3.12). At time $t=k+1$ we can measure $\Omega(k+1)$, $\Omega(K)$, and $U(k)$. Hence we know what $W(k)$ was. We can record $W(k)$ as a function of time during the motion of a robot. This profile can be used to derive useful frequency characteristics of $W(k)$ which in turn can be used to design window functions for the optimal controller. Unfortunately we have no way of knowing what $W(k)$ is at time $t=k$. Had we known $W(k)$ at $t=k$ we could modify the input $U(k)$ by subtracting $W(k)$. The new input $\tilde{U}(k) = U(k) - W(k)$ would lead to an ideal linearized system as given by equation (3.10). Not knowing what $W(k)$ is we can only use an estimate $\hat{W}(k)$. Using $\tilde{U}(k) = U(k) - \hat{W}(k)$ and substituting it in equation (3.12) we get

$$\Omega(k+1) = \Omega(k) + U(k) + W(k) - \hat{W}(k)$$  
(7.8)

We now have an effective modelling error $Y(k)$ given by

$$Y(k) = W(k) - \hat{W}(k)$$  
(7.8)

A good way to form an estimate $\hat{W}(k)$ is to use a linear predictor based on the $n$ previous values of $W$ which are known.

$$\hat{W}(k) = a_1 W(k-1) + a_2 W(k-2) + \cdots + a_n W(k-n)$$  
(7.9)

There are two interesting ways to choose the coefficients $a_i$'s. The first is to use a least square type estimator. This attempts to choose the coefficients such that $\|W(k) - \hat{W}(k)\|_2$ is minimized. An alternate method is to choose the coefficients such that $Y(k)$ is a filtered version of $X(k)$. To understand this let us substitute equa-
tion (7.9) in equation (7.8). We get

$$Y(k) = W(k) - \left[ a_1W(k-1) + a_2W(k-2) + \cdots + a_nW(k-n) \right] \quad (7.10)$$

Taking the $Z$ transform we get

$$\frac{Y(z)}{X(z)} = 1 - a_1z - a_2z^2 - \cdots - a_nz^n \quad (7.11)$$

Equation (7.11) represents a $n^{th}$ order Finite Impulse Response (FIR) filter. It is easy to see that by using $Y(k-i)$'s in our estimate $\hat{W}(k)$ we can get a $n^{th}$ order Infinite Impulse Response (IIR) filter. This filtering action could be utilized to our advantage. If for instance we can build better controllers to attenuate the effect of exogenous input $W(k)$ at high frequencies than at low frequencies a good choice is to use a high pass filter in equation (7.11). This would ensure that the effective exogenous input $Y(k)$ was primarily high frequency. Alternately knowing that $W(k)$ was high frequency we could choose a low pass filter to decrease the overall amplitude of $Y(k)$. The above discussion is of course dependent on the assumption that $W(k)$ itself is not appreciably changed by the filtering process. Unfortunately in our simulations we found that the above assumption is not true. In trying to use low pass filters to attenuate $Y(k)$ we seemed to make $W(k)$ itself unstable. We were unable to investigate the filtering effect further but we believe that further research is required to experimentally validate this theoretically powerful idea.

Controlling Torques Using The Optimal Controller
We notice from our simulations that decreasing errors often leads to high torques. This is clearly undesirable and a good extension to this thesis would be to add a term for minimizing torques to the optimization criteria. However the difficulty with this is that the decoupling into separate SISO problems, that is characteristic of our solution, would no longer be valid, and a MIMO problem would have to be solved.
BLOCK DIAGRAM FOR SIMULATIONS
FIGURE (7.1)
Optimal Controller - $W_1=1/(1-0.2s)^{**5}$, $W_2=1/(1+0.75s)$ - Step response

Figure (7.3)

- Desired Angular Displacement
- Desired Radial Displacement
- Angular Displacement Error
- Radial Displacement Error
- Torque Required
- Force Required
Optimal Controller - $W_1 = 1/(1-8s)^2$, $W_2 = 1/(1+75s)$ - Tracking

Figure (7.3)
Optimal Controller - Robustness Varying $J$ - Figure (7.4)

True Value of $J=10.0$ - $W_1=1/(1-0.8z)^{**5}, W_2=1/(1+0.75z)$

Angle Tracking Error $/J=0.0$

Radius Tracking Error $/J=0.0$

Angle Tracking Error $/J=0.5$

Radius Tracking Error $/J=0.5$

Angle Tracking Error $/J=17.0$

Radius Tracking Error $/J=17.0$

Angle Tracking Error $/J=17.5$

Radius Tracking Error $/J=17.5$
Discrete Computed Torque PD Controller - T=0.01/kp=1.125/kv=.875

Figure (7.5)

Desired Angular Displacement

Desired Radial Displacement

Angular Displacement Error

Radial Displacement Error

Torque Required

Force Required
DiscreteComputed Torque PD Controller - $T=.01/kp=.405/kv=.395$

**Figure (7.6)**

- **Desired Angular Displacement**
- **Desired Radial Displacement**
- **Angular Displacement Error**
- **Radial Displacement Error**
- **Torque Required**
- **Force Required**
Discrete Computed Torque PD Controller - $T=0.01/k_p=0.005/k_v=-0.805$

Figure (7.7)

- Desired Angular Displacement
- Desired Radial Displacement
- Angular Displacement Error
- Radial Displacement Error
- Torque Required
- Force Required
Discrete Computed Torque PD Controller - Robustness Varying J - Figure (7.8)

True Value of J=10.0 - T=.01/kp=.405/kv=.395

Angle Tracking Error /J=0.0

Radius Tracking Error /J=0.0

Angle Tracking Error /J=0.5

Radius Tracking Error /J=0.5

Angle Tracking Error /J=15.0

Radius Tracking Error /J=15.0

Angle Tracking Error /J=15.5

Radius Tracking Error /J=15.5
Optimal Controller - $W1 = 1/(J-\xi^2)$ - Worst Output - Figure (7.9)
Discrete Computed Torque PD Controller - kp=.405/kr=.395 - Worst Output - \( W_1=1/(1-8\epsilon) \) - Figure (7.19)
Optimal Controller - W1=1/(1-\beta) - Response To Same Input As For Computed Torque - Figure (7.11)
Independent Joint PD Controller - $kp=10000/kv=200$ - Worst Output - $W1=1/(1-s)$ - Figure (7.12)
Optimal Controller - W1=1/(1+.8z) - Response To Same Input As For Independent Joint - Figure 7.13
APPENDIX A

Bibliography


