DUBLIN CITY UNIVERSITY

School of Electronic Engineering
Control Technology Research Unit

A Comprehensive Study of Robot Control Algorithms

by

Patrick J Gibbs B Eng

A thesis submitted for the degree of
Master of Engineering

Supervisor Dr John Ringwood

September 1991

A Comprehensive Study of Robot Control Algorithms

ABSTRACT

The PUMA 560 Industrial Manipulator is presently controlled using a PID control strategy Robot manipulators are highly coupled, nonlinear mechanical systems designed to perform specific tasks It is the function of any control algorithm to compute the input voltages or torques needed to follow a desired trajectory The PID controller is detuned, so as to cater for variations in system behaviour Thus, the performance of such a control algorithm is poor over the entire operating range of the robot and the need for more complex control strategies is clear

The research presented in this thesis derives a third order comprehensive dynamic model for the three primary robot joints, using the Euler-Lagrange formulation for the equations of motion A simulation package is designed to model this dynamic system Next, a wide range of different techniques are investigated in a simulation environment, to observe their performance on the computer model These control algorithms range from Fixed Parameter techniques to Adaptive strategies and Feedforward routines A set of performance criteria can be used to evaluate these techniques, and the *best* algorithm from each section is chosen Using the results of an identification performed on the robot, each of these control methods is applied to the resulting time varying model The results here are used to determine the optimal control strategy for manipulator use

Also in this thesis, a new hardware structure is designed and implemented This structure is capable of implementing complex control routines with adequately low sample periods The design uses advanced digital signal processors, which can perform arithmetic operations quickly

ACKNOWLEDGEMENTS

The continued help and encouragement from my project supervisor, Dr John Ringwood is greatly appreciated Thanks to all the staff of the Electronic Engineering Dept at Dublin City University Special thanks also to Philip Comerford and Fred Jones for their help and support through the earlier stages of this project Thanks to all the other Postgrads for their friendship which made my time at the university very enjoyable

DECLARATION

I hereby declare that the research herein was completed by the undersigned

Signed: Paddy Guthrie Date: 20-9-91

To my parents as they enter their golden years

CONTENTS

CONTENTS

CHAPTER 1

INTRODUCTION

Many people are fascinated by the operation of mechanical devices, particularly those which mimic human behaviour This fascination seems to have been prevalent throughout the ages Robots have the same fascination, but the control needed for robots is far more extensive than that needed for simple sequencing machines of old Not only will a certain sequence have to be carried out, but its operation must be insensitive to characteristics of the mechanisms In this way the task can be repeatedly performed with the same precision The framework for achieving this aim is provided by the study of automatic control It is the need to reliably and cheaply perform a wide variety of tasks that underlies the use of robots The term robotic control is used to cover not only the control of the mechanisms of the robot, but also the associated sensory systems and other mechanisms needed to carry out these tasks

To define the term ***robot***, the utilitarian definition given by the Robot Institute of America is used *"A robot is a reprogrammable, multifunctional manipulator, designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks"* In order to perform any useful tasks, the robot must interface with its environment, which may comprise of other robots, feeding devices, and most importantly, people Robotics is the study of not only the robot itself, but also the interfaces between it and its surroundings

The past twenty years has seen an increase in the importance of the robot manipulator This increase, for the most part, is due to the pressing need for increased productivity and quality end products. Most manufacturing tasks are performed by special purpose machines designed to perform predetermined functions The inflexibility of such machines has made the computer-controlled manipulator a more attractive and cost effective alternative

Most commercially available industrial robots are widely used in manufacturing and assembly tasks such as simple material handling, spot/arc welding, part assembly, spray painting, loading and unloading numerically based machines, in space and undersea exploration, in prosthetic arm research, and in the handling of dangerous materials such as nuclear or chemical waste

1.1 Robot Control Architectures

Robot control systems, like other large-scale systems, are hierarchial in structure They consist of different levels which perform different tasks The control hierarchy is most often vertical with each upper control level dealing with wider aspects of the overall system behaviour than the lower levels The higher levels in the hierarchy communicate with their next lowest level to transfer any information this levels needs for decision making The most common of these hierarchial structures is a four level one [1], shown in Fig 1 1

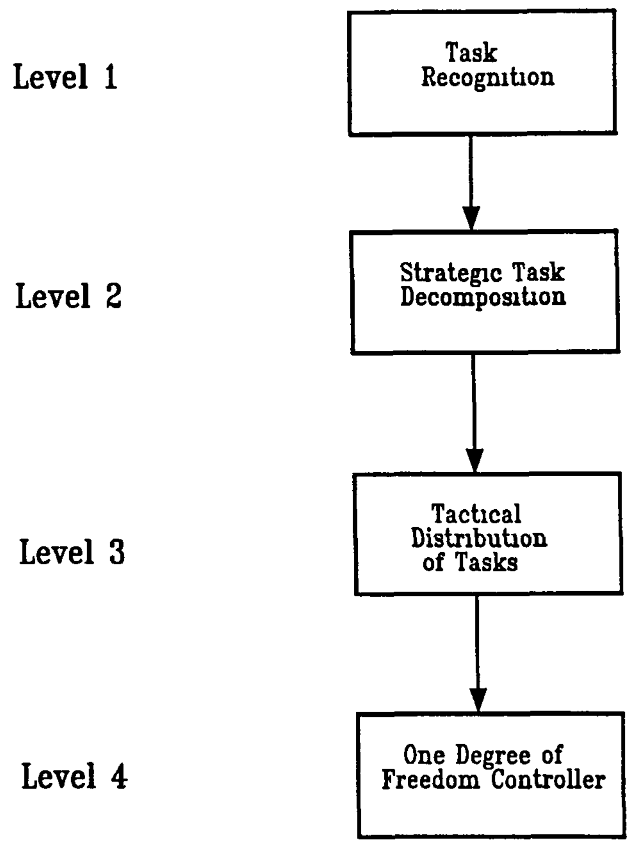


Fig.1.1 The Control Hierarchy Structure

All robots have the two lowest levels, but the upper two levels are specific to second and third generation robots [1] These robots are capable of sensing their work environment and use artificial intelligence means to perform their tasks correctly The two lower levels may be realized in various modes, and it is the size of their capabilities which determine the capabilities of the robot system as a whole

The Unimation control system is an example of a hierarchial control system The upper level consists of the LSI-11/02 microcomputer which serves as a supervisory computer, and the lower level consists of six 6503 Rockwell μPs and the other remaining hardware such as power amplifiers, joint position feedback sensors and a digital-to-analog converter [2] The control algorithm is situated in the lower level The setpoints are downloaded from the upper level Obviously, the performance of the control routine effects the performance of the overall robot system

1.2 The Dynamic Control Problem

In designing a controller for a specific process a model of that process is required The design technique uses this model with design specifications to derive a control equation The main problem in robot control is the complexity of the robot model Robot manipulators belong to a class of large-scale systems which are nonlinear in nature Robots have a large number of special features which makes the control problem difficult

Approximate models are usually used to design the simplest possible control algorithms A linearised system model is frequently used in conjunction with linear control theory to develop linear controllers In general, these approaches assume simplified models to be sufficiently accurate approximations of the actual robot However, this is not always the case, since oversimplification of the model may have occurred Control routines such as Optimal and PID control, for example, are based on the linear, decoupled single-input single-output (SISO) models for each of the three primary joints However, other techniques, such as Computed Torque, which is a multivariable routine, take into account the robot nonlinearities

To achieve robot control at a reasonable price, most robot manufacturers feel it is convenient to apply decentralized control This type of control treats the robot as a set of decoupled subsystems and applies a local controller to each of these subsystems Such a scheme neglects the effects of dynamic coupling among the different degrees of freedom of the manipulator In some cases, the coupling of joints is quite large and the synthesized controller performance may prove unsatisfactory Various methods [3], over the years, have been used to overcome the coupling effects These methods include linear and nonlinear self-tuning or adaptive strategies These controllers hope to overcome the coupling problem by tracking the system nonlinearities and by compensating for their presence in the control design

To implement such strategies, powerful hardware is required Unfortunately robot manufacturers are reluctant to replace existing controller hardware with a faster, more expensive alternative Recent developments, however in VLSI technology provide cost effective solutions to the implementation of such algorithms

1.3 Motivation for Research

This research was undertaken at the Control Technology Research Unit (C T R U) at Dublin City University (D C U) This unit at D C U has in recent years become interested in the area of robotics, and in particular the area of robot control For this reason the C T R U initiated this project, the aims of which were as follows

1 To develop a new controller hardware structure,
2 To develop and simulate suitable robot control routines and
3 To implement these control techniques using the hardware developed in 1 , to control an industrial robot

The C T R U at Dublin City University has a PUMA 560 robot arm It consists of six revolute joints Three relatively large links, which have a likeness to a human torso, upper arm and forearm, determine the end effector position The positions of these three links are changed using revolute joints which are often referred to as the Waist, Shoulder and Elbow joints The three secondary joints are concerned only with the position and orientation of the tool which is attached to the robot

From a control point of view, the most significant problem lies in the positioning of the tool, i e the control of the three primary joints Problems arise from the effects caused by relatively large sizes and masses of these three joints These effects take the form of inertial, centripetal, coriolis and gravitational coupling, and are responsible, in the main part, for the nonlinear nature of the control problem The need for an accurate dynamic model is twofold

1 It provides insight into the control problem and
2 It enables the designer to fully test controllers in simulation before the implementation layer

This thesis develops a fully tested robot simulator package, which models the actual robot dynamics This model was developed in conjunction with Jones [7] The design and simulation of the package is outlined in Chapter 2 Later chapters use this facility, as a control design tool, to simulate the response of various control techniques

1.4 Thesis Contributions

The development of the robot simulator is not a main topic of this thesis However, it was necessary to develop a robot simulation package to investigate the performance of control routines There are two major sections in this thesis

1 The Control Simulation Section and
2 The Hardware Design and Implementation Section

In the control simulation section, a series of control algorithms is developed and tested using the simulation facility These algorithms can be divided into three sections

1 Fixed Gain Algorithms
2 Adaptive Control Techniques and
3 Feedforward Strategies

In Chapter 7, the simulation section is evaluated under a series of performance criteria

The hardware design section is also a major contribution of this thesis The upper and lower levels of the existing Unimation control hardware are replaced with faster options This new design allows for a more flexible environment, where control routines can be easily implemented on the robot The design is user friendly because a personal computer becomes the upper level of the robot structure

1.5 Preview of Thesis

The research in this thesis is organized as follows

Chapter 2 outlines the modelling procedure used to correctly represent the PUMA 560 It then details the computer simulation of this model, and gives a series of open-loop tests which are performed on the model

Chapter 3 deals with two topics, Kinematics and Path Planning The geometric solution to the Forward and Inverse Kinematics problem is outlined Several methods of Path Planning are discussed, in particular the 4-3-4 and the Cubic Spline approaches

Chapter 4 is concerned with the topic of Fixed Parameter Control Algorithms Classical techniques such as PID, Lead-Lag and Optimal Control are investigated here, and their suitability for manipulator control is assessed The newer control method of Predictive Control is also used here, and two versions are tested on the simulator

Chapter 5 deals with Adaptive Control Firstly it details the parameter estimation technique of Recursive Least Squares Adaptive control techniques are designed and implemented These routines range from PID, to MRAC, to Predictive Control and finally to the design of a Self-Tuning Regulator These adaptive strategies should perform better than their fixed parameter versions

Chapter 6 outlines the technique of Feedforward Control, with special reference to Computed Torque Also, the incorporation of feedforward and feedback control is discussed

Chapter 7 evaluates the control routines developed in Chapters 4, 5 and 6 Using a set of performance criteria, these algorithms can be graded and a table of merit formed

Chapter 8 is concerned with the development of a new controller hardware structure It details the shortcomings of the existing controller and deals with the new hardware design necessary for a more flexible environment. To determine which control technique is best suited for manipulator use, the results of an identification performed on the PUMA are used to further evaluate PID, STR and Computed Torque

Chapter 9 summarizes what was achieved m the research It contains the achievements and shortcomings of the project.

CHAPTER 2

THE PUMA 560 DYNAMIC MODEL AND COMPUTER SIMULATOR

This chapter is concerned with the development and computer simulation of a dynamic model for the three primary joints of a PUMA 560 industrial robot Firstly the Euler-Lagrangian formulation of the PUMA 560 equations of motion, is outlined The motor dynamics for the first three links, are incorporated into the manipulator system equations using knowledge of the gearing ratios at each joint and the equivalent circuit model of the motors Modelling the motors as first order systems results in a third order set of differential equations describing the complete PUMA 560 dynamics

The final model has voltage rather than torque as inputs Although this is a comprehensive model for the PUMA 560 robot, no set of equations can ever specify the dynamics of a plant exactly

In order to facilitate computer simulation of the manipulator model, the set of third order differential equations are transformed to matrix form, and a state-space model results This allows for ease and clarity of simulation

Once the model simulator has been developed, several open-loop tests can be performed These simple tests are a quick method to validate the main manipulator dynamics

2.1 Developing a Comprehensive Model for the PUMA 560 Robot

For serially connected open-loop kinematic chains [9], the problem of generating a comprehensive dynamic model remains a challenging one Numerous approaches have been applied to the modelling of robotic manipulators The most commonly used of these is the Euler-Langrangian (E-L) method

The Euler-Lagrangian formulation of the second order differential equations of motion for a manipulator with n degrees of freedom can be written in the following format [4]

$$F_i = \sum_{j=1}^{n} D_{ij}q_i + I_{ai}q_i + \sum_{j=1}^{n}\sum_{k=1}^{n} C_{ijk}q_jq_k + G_i + H_iq_i \tag{2 1}$$

where,

q_i = position of joint i,
F_i = torque acting on joint i,
I_{ai} = actuator inertia of joint i,
D_{ii} = effective coupling of joint i,
D_{ij} = coupling inertia on i joint due to joint j,
C_{ijj} = centripetal force on i due to joint j,
C_{ijk} = coriolis force on joint i due to joints j and k,
G_i = gravity loading of joint i,
H_i = coefficient of friction for joint i

The inertia, centripetal, coriolis and gravity terms have been identified by Bejczy [5] and are defined as follows

$$\begin{aligned} D_{11} = {} & m_1k^2_{1yy} \\ & + m_2(k^2_{2xx}S^2_2 + k^2_{2yy}C^2_2 + a_2{}^2C^2_2 + 2a_2x_2C^2_2) + \\ & m_3[(k^2_{3xx}S^2_{23} + k^2_{3zz}C^2_{23} + d^2_3 + a_2{}^2C^2_2 + a^2_3C^2_{23} \\ & + 2a_2a_3C_2C_{23} + 2x_3(a_2C_2C_{23} + a_3C^2_{23}) + \\ & 2y_3d_3 + 2z_3(a_3C_{23}S_{23} + a_2C_2S_{23})] \end{aligned} \tag{2 2}$$

$$\begin{aligned} D_{12} = {} & m_2a_2z_2S_2 + \\ & m_3[(d_3x_3 + a_3y_3 + a_3d_3)S_{23} + \\ & (a_2y_3 + a_2d_3)S_2 - d_3z_3C_{23}] \end{aligned} \tag{2 3}$$

$$D_{13} = m_3[x_3d_3 + a_3y_3 + a_3d_3)S_{23} - z_3d_3C_{23}] \tag{2.4}$$

$$D_{22} = m_2(k^2{}_{2zz} + a^2{}_2 + 2a_2x_2) + m_3[(2a_2a_3 + 2a_2x_3)C_3 + 2a_2z_2S_3 + k^2{}_{3yy} + a^2{}_2 + a^3{}_3 + 2a_3x_3] \quad (2\ 5)$$

$$D_{23} = m_3[(a_2x_3 + a_2a_3)C_3 + a_2z_3S_3 + 2a_3x_3 + a^2{}_3 + k^2{}_{3yy}] \quad (2\ 6)$$

$$D_{33} = m_3(k^2{}_{3yy} + a^2{}_3 + 2a_3x_3) \quad (2\ 7)$$

$$C_{112} = m_2(k^2{}_{2xx} - k^2{}_{2yy} - a^2{}_2 - 2a_2x_2)C_2S_2 + m_3[k^2{}_{3xx}(C_2S_2 + C_3S_3 - 2S_2S_3S_{23}) + k^2{}_{3zz}(2S_2S_3S_{23} - C_2S_2 - C_3S_3) + x_3(-2a_2C_2S_{23} + 4a_3S_2S_3S_{23} + a_2S_3 - 2a_3C_2S_2 - 2a_3C_3S_3) + z_3(a_2C_2C_{23} - a_2S_2S_{23} + 2a_3C^2{}_{23} - a_3) + a_2a_3S_3 - 2a_2a_3C_2S_{23} - a^2{}_2C_2S_2 + 2a^2{}_3S_2S_3S_{23} - a^2{}_3(C_2S_2 + C_3S_3)] \quad (2\ 8)$$

$$C_{113} = m_3[k^2{}_{3xx}(C_2S_2 + C_3S_3 - 2S_2S_3S_{23}) + k^2{}_{3zz}(2S_2S_3S_{23} - C_2S_2 - S_3C_3) + x_3(4a_3S_2S_3S_{23} - 2a_3C_2S_3 - 2a_3C_3S_3 - a_2C_2S_{23}) + z_3(2a_3C^2{}_{23} + a_2C_2C_{23} - a_3) + 2a^2{}_3S_2S_3S_{23} - a_2a_3C_2S_{23} - a^2{}_3C_2S_2 - a^2{}_3C_3S_3] \quad (2\ 9)$$

$$C_{122} = m_2a_2z_2C_2 + m_3[d_3z_3S_{23} + (d_3x_3 + a_3y_3 + a_3d_3)C_{23}] \quad (2\ 10)$$

$$C_{123} = m_3[d_3z_3S_{23} + (d_3x_3 + a_3y_3 + a_3d_3)C_{23}] \quad (2\ 11)$$

$$C_{133} = m_3[d_3z_3S_{23} + (d_3x_3 + a_3y_3 + a_3d_3)C_{23}] \quad (2\ 12)$$

$$C_{213} = 0 \text{ (because of general PUMA geometry)} \quad (2\ 13)$$

$$C_{223} = m_3[(-a_2x_3 - a_2a_3)S_3 + a_2z_3C_3] \quad (2\ 14)$$

$$C_{233} = m_3[(-a_2x_3 - a_2a_3)S_3 + a_2z_3C_3] \quad (2\ 15)$$

$$G_1 = 0 \text{ (because of general PUMA 560 geometry)} \quad (2\ 16)$$

$$G_2 = m_2g(x_2 + a_2)C_2 - m_3g(x_3C_{23} + z_3S_{23} + a_3C_{23} + a_2C_2) \tag{2 17}$$

$$G_3 = -m_3g(x_3C_{23} + z_3S_{23} + a_3C_{23}) \tag{2 18}$$

The following shorthand notation is used above

$S_i = Sin(q_i)$
$C_i = Cos(q_i)$
$S_{ij} = Sin(q_i + q_j)$
$C_{ij} = Cos(q_i + q_j)$

Using Newton's second law of physics, the following rules apply

$$\begin{aligned} D_{ij} &= D_{ji} \\ C_{ijk} &= C_{ikj} \\ C_{ijk} &= -C_{kji} \text{ for } i,k \geq j \\ C_{iji} &= 0 \text{ for } i \geq j \end{aligned} \tag{2 19}$$

Consequently this gives the following relationships

$$\begin{aligned} & D_{21} = D_{12},\ D_{13} = D_{31},\ D_{32} = D_{23}, \\ & C_{111} = C_{222} = C_{333} = 0,\ C_{121} = C_{112}, \\ & C_{131} = C_{113},\ C_{132} = C_{123},\ C_{221} = C_{212}, \\ & C_{231} = C_{213},\ C_{232} = C_{223},\ C_{321} = C_{312}, \\ & C_{331} = C_{313},\ C_{332} = C_{323},\ C_{211} = -C_{112}, \\ & C_{311} = -C_{113},\ C_{312} = -C_{213},\ C_{322} = -C_{223}, \\ & C_{313} = C_{323} = C_{212} = 0 \end{aligned} \tag{2 20}$$

The quantities x_i, y_i and z_i are the Carthesian coordinates of the centre of mass of joint i referenced to the base of the robot. The quantity m_i is the mass of joint i and k^2_{ixx}, k^2_{iyy} and k^2_{izz} are the radii of gyration for joint i The quantities d_i and a_i are the link twists and the link lengths The values of these geometric and inertial parameters which relate to the three primary joints of the PUMA 560 are listed in Table (2 1) and Table (2 2) These are the estimates obtained by Bejczy [5] He arrived at these values by first taking detailed measurements of all link internal components, then calculating their individual moments of inertia, and later getting the cumulative effect using the Parallel Axis Theorem, Goldstein [6]

Table 2.1 PUMA 560 Inertial Parameters

Link	Centre of Mass			Mass	Radius of Gyration		
i	x_i	y_i	z_i	$g\ s^2/cm$	k^2_{ixx}	k^2_{iyy}	k^2_{izz}
1	0	30 88	3 89	13 21	1816 3	151 93	1811 1
2	-32 89	0	20 38	22 8	595 7	1355 6	1513 6
3	-2 04	-1 37	0 3	5 11	151 48	155 23	20 7

Table 2.2 PUMA 560 Geometric Parameters

a_2(cm)	a_3(cm)	d_2(cm)	d_3(cm)
43 18	1 91	15 05	43 31

From an examination of equation (2 1) one can see that the inputs to this model are joint torques, while the outputs are joint positions, velocities and accelerations The inputs to the PUMA 560 are the actuator inputs needed to drive its DC motors It is therefore necessary to incorporate the actuator dynamics into the overall equations of motion of the robot. The derivation of the manipulator model in this fashion was performed in conjunction with Jones [7] The dc motors used to drive the first three joints of the PUMA 560 are 100Watt permanent magnet direct current servomotors Figure 2 1 shows a simple equivalent circuit model for the permanent magnet dc motor and lists the associated model parameters The model equation can be derived using Kirchoff's voltage law as follows

$$V_i = R_i\, \iota_i + L_i \frac{d\iota_i}{dt} + k_i^e \frac{d\omega_i}{dt} \qquad (2\ 21)$$

The torque produced by a dc motor is proportional to the armature current of the dc motor

$$F_i = k_i^t\, \iota_i \qquad (2\ 22)$$

where F_i is the torque experienced at joint i

The joint position be can related to the motor position by the following equation

$$\omega_i = N_i\, q_i \qquad .(2\ 23)$$

where N_i is the gearing ratio of joint i

Substituting equations (2.22), (2.23) into equation (2.21) gives the following equation for joint voltage

$$V_i = k_i^e N_i \frac{dq_i}{dt} + [R_i F_i + L_i \frac{dF_i}{dt}] / k_i^t \tag{2.24}$$

The quantity F_i is the derivative of the joint torque and is given by

$$F_i = \sum_{j=1}^{3} (D_{ij} q_j + D_{ij} q_j) + I_{ai} q_i$$

$$+ \sum_{j=1}^{3} \sum_{k=1}^{3} (C_{ijk} q_j q_k + C_{ijk} q_j q_k + C_{ijk} q_j q_k)$$

$$+ G_i + H_i q_i \tag{2.25}$$

The total model can then be written as

$$\begin{aligned} V_i = \; & k_i^e N_i q_i + R_i [H_i q_i + G_i \\ & + \sum_{j=1}^{3} D_{ij} q_i + I_{ai} q_i + \sum_{j=1}^{3} \sum_{k=1}^{3} C_{ijk} q_j q_k] / k_i^t \\ & + L_i [G_i + \sum_{j=1}^{3} (D_{ij} q_j + D_{ij} q_j) + I_{ai} q_i \\ & + \sum_{j=1}^{3} \sum_{k=1}^{3} (C_{ijk} q_j q_k + C_{ijk} q_j q_k + C_{ijk} q_j q_k) + H_i q_i] / k_i^t \end{aligned} \tag{2.26}$$

This is the third order model equation for each primary joint of the PUMA 560

2.2 Computer Simulation of the PUMA 560 Robot

The design and computer implementation of the above manipulator model are discussed in this section The model is transformed into a state-space representation, with the highest order terms occurring first The lower order terms are calculated using the Runge-Kutta numerical integration technique

The simulator has three inputs (actuator voltage) and its outputs are joint accelerations, velocities and positions The simulator is designed to aid in the evaluation of possible control algorithms and to decide their suitability for manipulator control

To derive the state-space model, it is necessary to rewrite the fundamental manipulator model equation (2 26) in matrix form The following matrix and vector notation is used in this section, Anderson [8]

$$\mathrm{LMAT} = \mathrm{Diagonal}(\ L_1/k_1^t,\ L_2/k_2^t,\ L_3/k_3^t\ )$$

$$\mathrm{RMAT} = \mathrm{Diagonal}(\ R_1/k_1^t,\ R_2/k_2^t,\ R_3/k_3^t\ )$$

$$\mathrm{HMAT} = \mathrm{Diagonal}(\ H_1,\ H_2,\ H_3\ )$$

$$\mathrm{IMAT} = \mathrm{Diagonal}(\ I_{a1},\ I_{a2},\ I_{a3}\ )$$

$$\mathrm{KMAT} = \mathrm{Diagonal}(\ N_1 k_1^e,\ N_2 k_2^e,\ N_3 k_3^e\ )$$

$$\underset{\sim}{G} = \mathrm{Gravity\ Vector}(\ G_1,\ G_2,\ G_3\ )$$

D = matrix which contains all the effective and coupling inertial terms,
D^1 = matrix which contains the centripetal and coriolis forces experienced by joint 1,
D^2 = matrix which contains the centripetal and coriolis forces experienced by joint 2,
D^3 = matrix which contains the centripetal and coriolis forces experienced by joint 3

Hence equation (2 26) can be rewritten as

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \mathrm{LMAT}\ \left[\ D + \mathrm{IMAT}\ \right] \begin{bmatrix} q_7 \\ q_8 \\ q_9 \end{bmatrix} +$$

$$\Big(\ \mathrm{LMAT}\ D + \mathrm{RMAT}\ \{\ D + \mathrm{IMAT}\ \} + \mathrm{HMAT}\ +$$

$$\mathrm{LMAT}\begin{bmatrix}(q_4, q_5, q_6)\, D^1\\ (q_4, q_5, q_6)\, D^2\\ (q_4, q_5, q_6)\, D^3\end{bmatrix} \Big\} \begin{bmatrix} q_7\\ q_8\\ q_9\end{bmatrix} +$$

$$\Big\{ \mathrm{LMAT}\begin{bmatrix}(q_7, q_8, q_9)\, D^1\\ (q_7, q_8, q_9)\, D^2\\ (q_7, q_8, q_9)\, D^3\end{bmatrix} + \mathrm{RMAT}\begin{bmatrix}(q_4, q_5, q_6)\, D^1\\ (q_4, q_5, q_6)\, D^2\\ (q_4, q_5, q_6)\, D^3\end{bmatrix}$$

$$+ \mathrm{LMAT}\begin{bmatrix}(q_4, q_5, q_6)\, D^1\\ (q_4, q_5, q_6)\, D^2\\ (q_4, q_5, q_6)\, D^3\end{bmatrix} + \mathrm{RMAT\ HMAT} + \mathrm{KMAT}\Big\} \begin{bmatrix} q_4\\ q_5\\ q_6\end{bmatrix}$$

$$+ \mathrm{LMAT}\ \underset{\sim}{G} + \mathrm{RMAT}\ \underset{\sim}{G} \qquad (2\ 27)$$

The following quantities are defined to simplify the model equation

1. $\underline{D} = \mathrm{LMAT}\ [\ D + \mathrm{IMAT}\ ]$

2. $\underline{P}(q) =$

$$\Big\{ \mathrm{LMAT}\ D + \mathrm{RMAT}\ \{\ D + \mathrm{IMAT}\ \} + \mathrm{HMAT}\ +$$

$$\mathrm{LMAT}\begin{bmatrix}(q_4, q_5, q_6)\, D^1\\ (q_4, q_5, q_6)\, D^2\\ (q_4, q_5, q_6)\, D^3\end{bmatrix} \Big\} \begin{bmatrix} q_7\\ q_8\\ q_9\end{bmatrix} +$$

$$\Big\{ \mathrm{LMAT}\begin{bmatrix}(q_7, q_8, q_9)\, D^1\\ (q_7, q_8, q_9)\, D^2\\ (q_7, q_8, q_9)\, D^3\end{bmatrix} + \mathrm{RMAT}\begin{bmatrix}(q_4, q_5, q_6)\, D^1\\ (q_4, q_5, q_6)\, D^2\\ (q_4, q_5, q_6)\, D^3\end{bmatrix}$$

$$+ \mathrm{LMAT}\begin{bmatrix}(q_4, q_5, q_6)\, D^1\\ (q_4, q_5, q_6)\, D^2\\ (q_4, q_5, q_6)\, D^3\end{bmatrix} + \mathrm{RMAT\ HMAT} + \mathrm{KMAT}\Big\} \begin{bmatrix} q_4\\ q_5\\ q_6\end{bmatrix}$$

$$+ \mathrm{LMAT}\ \underset{\sim}{G} + \mathrm{RMAT}\ \underset{\sim}{G}$$

Hence the model equation can be written as

$$\begin{bmatrix} V_1\\ V_2\\ V_3\end{bmatrix} = \underline{D}\begin{bmatrix} q_7\\ q_8\\ q_9\end{bmatrix} + \underline{P}(q) \qquad ..(2.28)$$

Rearranging one gets

$$\Rightarrow \begin{bmatrix} q_7 \\ q_8 \\ q_9 \end{bmatrix} = - \underline{D}^{-1} \underline{P}(q) + \underline{D}^{-1} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (2\ 29)$$

The following relationships apply from the basic laws of physics

$$\begin{array}{lll} q_1 = q_4 & \qquad & q_4 = q_7 \\ q_2 = q_5 & & q_5 = q_8 \\ q_3 = q_6 & & q_6 = q_9 \end{array}$$

Hence the full ninth order comprehensive model for the first three joints of the PUMA 560 can be written as

$$\left[\begin{array}{c} q_1 \\ q_2 \\ q_3 \\ \hline q_4 \\ q_5 \\ q_6 \\ \hline q_7 \\ q_8 \\ q_9 \end{array}\right] = \left[\begin{array}{c} q_4 \\ q_5 \\ q_6 \\ \hline q_7 \\ q_8 \\ q_9 \\ \hline \\ -\underline{D}^{-1}\ \underline{P}(q) \\ \\ \end{array}\right] + \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \hline & & \\ & \underline{D}^{-1} & \\ & & \end{array}\right] \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (2.30)$$

The state vector for the model is $q \in R^9$

$$q = [\ q_1\ q_2\ q_3\ q_4\ q_5\ q_6\ q_7\ q_8\ q_9\ ]^T$$

Note $\underline{P}(q)$ is a vector whose elements are dependent on the vector q and the manipulator parameters,

$$\underline{P}(q) \in R^3.$$

This vector is complex and requires considerable processor time to compute at each interval It is very nonlinear, and the sine and cosine functions are required to calculate the elements of the inertial, centripetal and coriolis matrices Gravity terms are also a nonlinear element

$\underline{D}$ is a matrix whose elements are dependent upon the vector q and the manipulator parameters,

$$\underline{D} \in R^{3 \times 3}$$

This matrix is derived from two static matrices and D, the inertial matrix which is dependent on the state vector

To obtain the joint positions, velocities and accelerations it is necessary to apply some form of numerical integration technique to solve these differential equations From the point of accuracy, rather than speed of simulation, it was decided to use a classical fourth-order Runge-Kutta algorithm to integrate the states in the manipulator model equation An integration interval of 5msecs was chosen and gave sufficient accuracy The next section describes in detail the Runge-Kutta algorithm

The above state-space description of the PUMA 560 robot has actuator voltage as inputs and joint acceleration, velocity and position as states/outputs From the model description one can see that this model is very nonlinear and highly coupled Later chapters in this thesis investigate a wide range of control techniques and evaluate their performance on the manipulator model to assess which are suitable for implementation on an actual robot

2.3 Implementation of 4^{th} Order Runge-Kutta Integration Technique

Numerical integration techniques involve predicting the system states at time k, given the states at time k-1 and the present inputs to the system This section discusses the Runge-Kutta integration technique and why it was chosen in preference to other methods

The simplest numerical integration technique is the Euler Method The Euler Method approximates the curve x = f(t) by a polygon whose slope, at each time t_r is given by the tangent to the curve x = f(t) at t_r. It is a first order method with a truncation error per step of order h^2 Errors occur because the slope of f(t) changes over the interval h A better approximation of the slope, over the interval, will result in a closer estimate of the function.

The fourth order Runge-Kutta method provides a closer approximation of the functions' slope over each interval by taking a weighted sum of the slopes about each point t Truncation errors in Runge-Kutta numerical methods are of the order of h^{n+1} Runge-Kutta algorithms have the following desirable properties

1 the integration is self starting,
2 the step size can easily be changed between iterations,
3 no derivative evaluation is required,
4 algorithms have good stability characteristics,
5 technique can be applied to nonlinear systems

For the nth order equation written as

$x = f(x,t)$

$x_i = f_i(x_1, x_2, \quad x_n, t) \qquad 1 \leqslant i \leqslant n,$

the formula for advancing the solution one step is

$$x_{i,r+1} = x_{i,r} + (k_{i1} + 2(k_{i2} + k_{i3}) + k_{i4})/6$$

where,

$$x_{i,r+1} = x_i(t_{r+1}) = x_i(t_0 + (r+1)h)$$

$$k_{i1} = hf_i(x_{1,r}, x_{2,r}, \quad x_{n,r}, t_r)$$
$$k_{i2} = hf_i(x_{1,r} + 0\ 5k_{11}, \quad x_{n,r} + 0\ 5k_{n1}, t_{r+0\ 5})$$
$$k_{i3} = hf_i(x_{1,r} + 0\ 5k_{12}, \quad x_{n,r} + 0\ 5k_{n2}, t_{r+0\ 5})$$
$$k_{i4} = hf_i(x_{1,r} + 0\ 5k_{13}, \quad x_{n,r} + 0\ 5k_{n3}, t_{r+1})$$

For the nth order system with an external input

$x = f(x,u,t)$

$x_i = f_i(x_1, x_2, \quad x_n, u_i, t) \qquad 1 \leqslant i \leqslant n,$

the Runge–Kutta algorithm must be altered

If the system has an external input then the function must also be differentiated with respect to the input When the input is held constant over the interval then the partial derivatives with respect to the input will be zero giving no cause for adapting the standard formula Thus the input will be treated like a system state and the following solution applies

$$x_i = f_i(x_1, x_2, \quad x_n, u_i, t) \qquad 1 \leqslant i \leqslant n,$$

the formula for advancing the solution one step is

$$x_{i,r+1} = x_{i,r} + (k_{i1} + 2(k_{i2} + k_{i3}) + k_{i4})/6$$

where,

$$x_{i,r+1} = x_i(t_{r+1}) = x_i(t_0 + (r+1)h)$$

$$\begin{aligned}
k_{i1} &= hf_i(x_{1,r}, x_{2,r}, \quad x_{n,r}, u_{i,r}, t_r) \\
k_{i2} &= hf_i(x_{1,r} + 0\ 5k_{11}, \quad x_{n,r} + 0\ 5k_{n1}, u_{i,r+0\ 5}, t_{r+0\ 5}) \\
k_{i3} &= hf_i(x_{1,r} + 0\ 5k_{12}, \quad x_{n,r} + 0\ 5k_{n2}, u_{i,r+0\ 5}, t_{r+0\ 5}) \\
k_{i4} &= hf_i(x_{1,r} + 0\ 5k_{13}, \quad x_{n,r} + 0\ 5k_{n3}, u_{i,r+1}, t_{r+1})
\end{aligned}$$

When simulating the PUMA 560 model on a PC, a integration interval of 5msecs was chosen This step size gives sufficient accuracy and also does not over-burden the processor Larger values of step size are not suitable for simulation purposes because of a considerable reduction in accuracy of the system output at high velocity Also in a simulation environment model accuracy, rather than simulation speed is the priority

2.4 The Simplified Linear Decoupled Joint Models

To design simple linear controllers for the robot, it is usually necessary to have an approximate linear model of the system available, to base the design upon

Taking the torque equation (2 1),

$$F_i = \sum_{j=1}^{n} D_{ij} q_i + I_{ai} q_i + \sum_{j=1}^{n} \sum_{k=1}^{n} C_{ijk} q_j q_k + G_i + H_i q_i$$

if the coupling and gravity terms are ignored then

$$F_i = I_{ai} q_i + H_i q_i \qquad (2\ 31)$$

$$F_i = I_{ai} q_i + H_i q_i \qquad (2\ 32)$$

Substituting equations (2 31) and (2 32) into equation (2 24) gives

$$V_i = L_i I_{ai} q_i / k_i^t + \left(R_i I_{ai} + L_i H_i \right) q_i / k_i^t$$
$$+ \left(k_i^e\ N_i\ k_i^t + R_i H_i \right) q_i\ / k_i^t \qquad (2\ 33)$$

This is a linear model for each of three primary joints of the PUMA 560 robot. It ignores the nonlinear terms which are present in the comprehensive model, so there is no coupling or gravity terms present It can also be represented by the following transfer function

$$\frac{Q(s)}{V(s)} = \frac{b}{s^3 + a_1\ s^2 + a_2\ s}$$

where,

q = joint position,

v = armature voltage,

$$b = k_i^t / (L_i\ I_{ai})$$

$$a_1 = (L_i\ H_i + R_i\ I_{ai}) / (L_i\ I_{ai})$$

$$a_2 = (R_i\ H_i + k_i^e\ N_i\ k_i^t) / (L_i\ I_{ai})$$

Computing the coefficients of the transfer function results in the following three models,

<u>Linear model for Joint 1</u>

$$\frac{Q(s)}{V(s)} = \frac{687.1058}{s^3 + 333\ 46 s^2 + 11219\ 45 s}$$

<u>Linear model for Joint 2</u>

$$\frac{Q(s)}{V(s)} = \frac{225.9552}{s^3 + 333\,47s^2 + 6380\,87s}$$

<u>Linear model for Joint 3</u>

$$\frac{Q(s)}{V(s)} = \frac{915.7552}{s^3 + 333\,58s^2 + 12853\,34s}$$

These model are used extensively in the later control based chapters, when several different control techniques are investigated on the comprehensive model

2.5 Open-Loop Model Performance

The open-loop tests consist of supplying the dynamic model with different sets of constant input voltages to drive the joints These tests show the dominant dynamics of the model and also indicate the level of coupling that exists between joints The effects of gravity can be seen when zero volts is applied to each of the joint motors

<u>Test 1a</u> Apply 10, 0 and 0 volts to joints 1, 2 and 3 respectively (see Fig 2 2a and Fig 2 2b)

<u>1b</u> Apply 0, 10 and 0 volts to joints 1, 2 and 3 respectively (see Fig 2 3a and Fig 2 3b)

<u>1c</u> Apply 0, 0 and 10 volts to joints 1, 2 and 3 respectively (see Fig 2 4a and Fig 2 4b)

This series of tests show the dominant dynamics and coupling between joints

<u>Test 2a</u> Apply 0, 0 and 0 volts to joints 1, 2 and 3 respectively (see Fig 2 5a and Fig 2 5b)

<u>2b</u> Apply V_{1h}, V_{2h} and V_{3h} volts to joints 1, 2 and 3 respectively (see Fig 2 6a and Fig 2 6b)

Test 2a. shows the effect of gravity on each joint and test 2b applies the correct voltage to hold the joints in the zero position These voltages were calculated from the manipulator model, given the robot parameters and the initial states

2.6 Summary

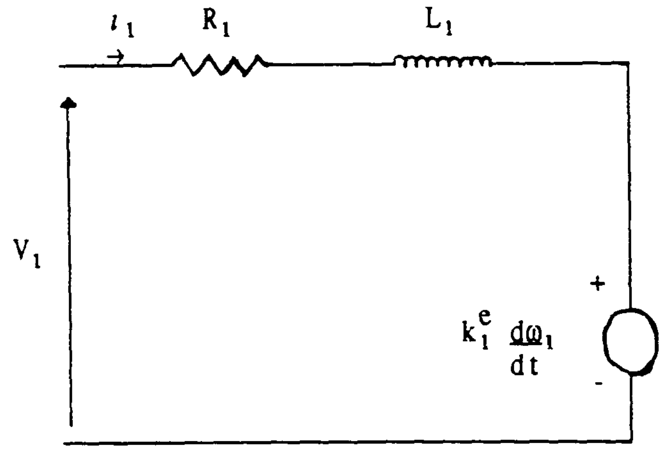
This chapter is concerned with the development and computer simulation of a dynamic model for the three primary joints of a PUMA 560 industrial robot The model combines a second order Euler-Lagrangian formulation of the PUMA 560 equations of motion with the actuator dynamics In the case of the PUMA 560, a first order approximation of the permanent magnet D C motor drive dynamics was chosen, resulting in a set of third order differential equations for the manipulator

The simulation of the robot model is a main topic of this chapter also A description of how to simulate the PUMA 560 in state space format is given Different facilities exist within the simulator package which attempt to make the model a more realistic mirror of the physical system To derive the system state values, an integration technique is required A fourth order Runge–Kutta numerical integration method was selected for the following reasons

1 small truncation error (of the order of h^5),
2 ease of implementation on a digital computer,
3 suitability for systems with piecewise constant input.

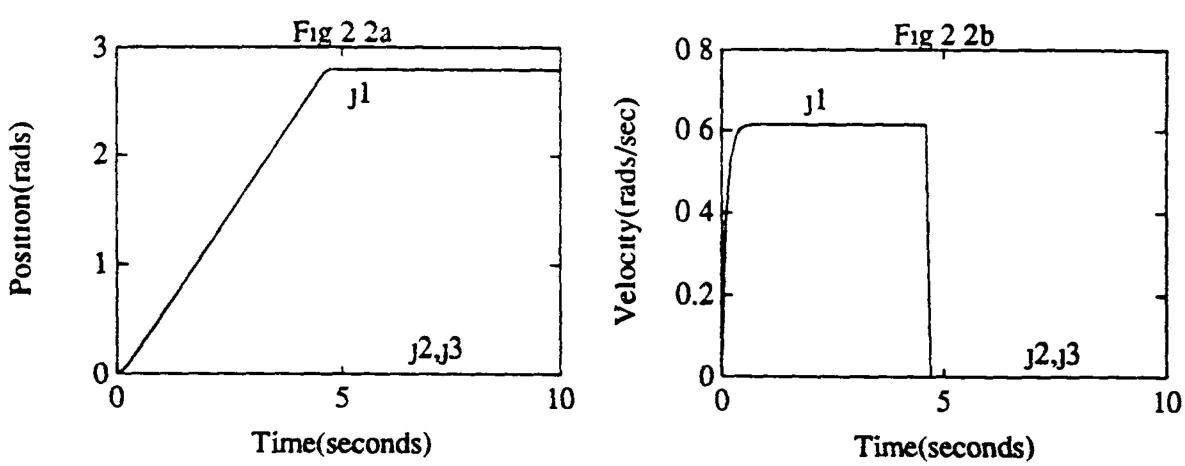
Open-loop tests were performed on the robot model These tests consisted of applying constants voltages to the three main joints to observe the coupling and the main dynamics of each joint, i e integrative action. Also the effect of gravity is shown

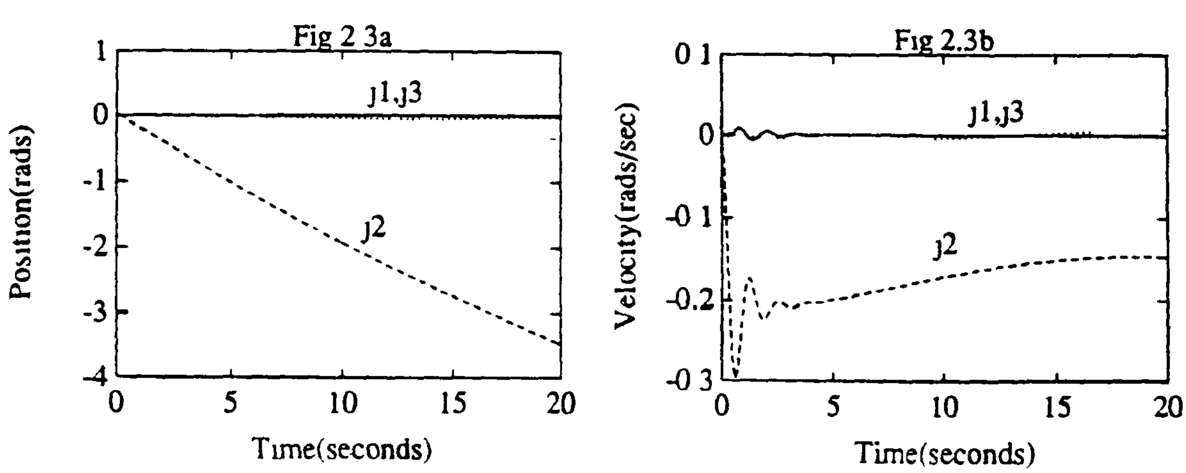
Index to Graphs



ω_1 = motor position
R_1 = armature resistance
L_1 = armature inductance
i_1 = armature current
k_1^t = voltage constant
k_1^e = torque constant
V_1 = armature voltage

Fig. 2.1 *The Equivalent Circuit Model for a Permanent DC Motor*





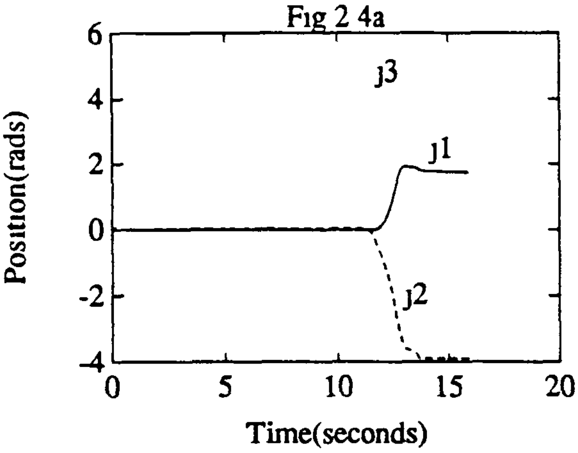


Fig 2 4a

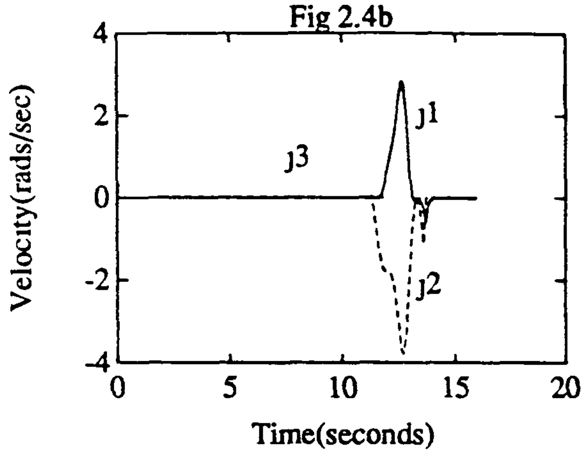


Fig 2.4b

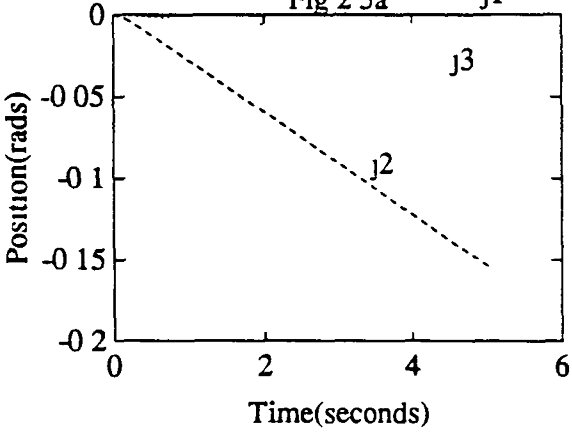


Fig 2 5a

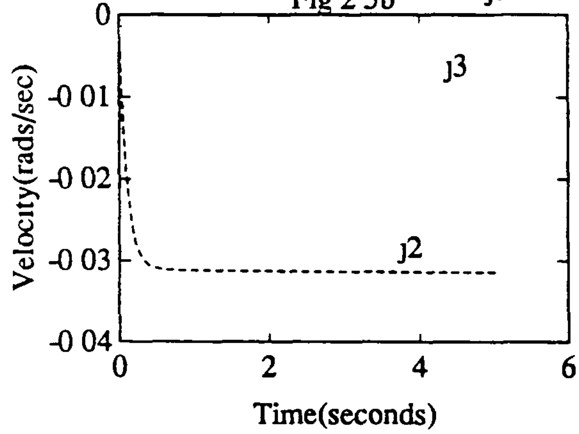


Fig 2 5b

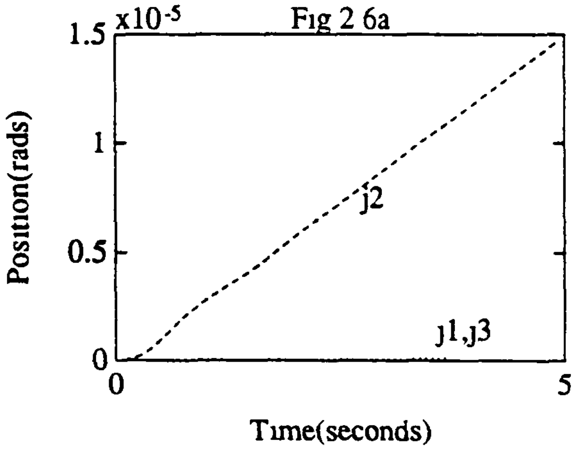


Fig 2 6a

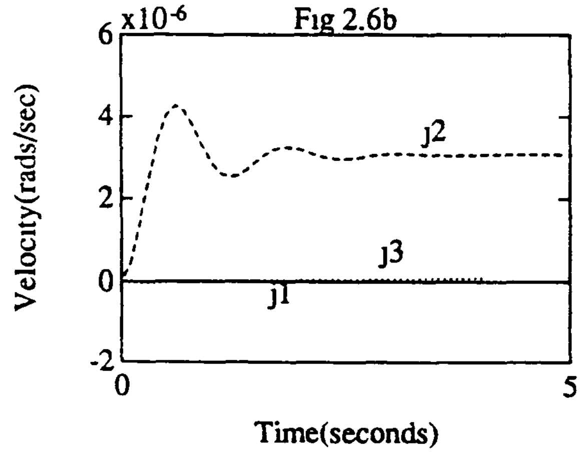


Fig 2.6b

CHAPTER 3

ROBOT ARM KINEMATICS AND MANIPULATOR TRAJECTORY GENERATION

This chapter is concerned with the topics of robot kinematics and the generation of efficient manipulator trajectories These two topics are upper level tasks which are crucial for implementing real-time control Kinematics is concerned with transforming joint angles to determine the end-effector position, and also the inverse transform from hand position to joint angles Path planning is a mathematical technique, which joins the endpoints of a trajectory using polynomial functions of time to interpolate the desired path Several techniques exist for path planning, but only a discussion of the joint-interpolated trajectory method is given

3.1 Kinematics

Robot arm kinematics deals with the analysis of the geometry of motion of a robot arm with respect to a fixed reference coordinate system as a function of time without regard for the forces/moments that cause the motion [10] Thus, it deals with the analytic description of the spatial displacement of the robot as a function of time, in particular the relations between the joint-variable space and the position and orientation of the end-effector of a robot arm This section addresses two fundamental questions of interest in robot kinematics [10]

1 For a given manipulator, given the joint angle vector $q(t) = (q_1(t), q_2'(t), \quad q_n(t))^T$ and the geometric link parameters, where n is the number of degrees of freedom, what is the position and orientation of the end-effector of the manipulator with respect to a reference coordinate system?

2 Given a desired position and orientation of the end-effector of the manipulator and the geometric link parameters with respect to a reference coordinate system, can the manipulator reach the desired prescribed manipulator hand position and orientation? And if it can, how many different manipulator configurations will satisfy the same condition?

The first question is usually referred to as the *direct kinematics* (or forward) problem, while the second question is the *inverse kinematics* (or arm solution) problem Since the independent variables in a robot arm are the joint variable and a task is usually stated in terms of the reference coordinate frame, the inverse kinematics problem is used more frequently

3 1.1 The Direct Kinematics Problem

The direct kinematics problem can be reduced to finding a transformation matrix that relates the body-attached coordinate frame to a reference coordinate frame Denavit-Hartenberg [11] representation results in a 4x4 homogeneous transformation matrix representing each link's coordinate system at the joint with respect to the previous link's coordinate system Thus, through sequential transformations, the end-effector expressed in the hand coordinates can be transformed and expressed in the base coordinates which make up the inertial frame of this dynamic system [12]

An orthonormal cartesian coordinate system (x_i, y_i, z_i) can be established for each link at its joint axis, where $i = 1,2,$ n (where n = number of degrees of freedom) plus the base coordinate frame For a six-axis PUMA-like robot arm, seven coordinate frames exist, (x_0,y_0,z_0), (x_1,y_1,z_1), (x_6,y_6,z_6)

Every coordinate frame is determined and established on the basis of three rules [10]

1 The z_{i-1} axis lies along the axis of motion of the i^{th} joint
2 The x_i axis is normal to the z_{i-1} axis, and pointing away from it
3 The y_i axis completes the right-handed coordinate system

The Denavit-Hartenberg [11] representation of a rigid link depends on four geometric parameters associated with each link These four parameters completely describe any revolute or prismatic joint These four parameters are defined as follows (see Fig 3 1) :

θ_i is the joint angle from the x_{i-1} axis to the x_i axis about the z_{i-1} axis

d_i is the distance from the origin of the $(i-1)^{th}$ coordinate frame to the intersection of the z_{i-1} axis with the x_i axis along the z_{i-1} axis

a_i is the offset distance from the intersection of the z_{i-1} axis with the x_i axis to the origin of the i^{th} frame along the x_i axis

α_i is the offset angle from the z_{i-1} axis to the z_i axis about the x_i axis

Once the coordinate system has been established for each link, a homogeneous transformation matrix can easily be developed relating the i^{th} coordinate frame to the $(i-1)^{th}$ coordinate frame The homogeneous matrix ${}^0T_i$ which specifies the location of the i^{th} coordinate frame with respect to the base coordinate system is the chain product of successive coordinate transformation matrices of ${}^{i-1}A_i$, and is expressed as

$$ {}^0T_i = \prod_{j=1}^{i} {}^{j-1}A_j \quad \text{for } i = 1,2, \ldots, n $$

$$ = \begin{bmatrix} x_i & y_i & z_i & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.1) $$

where

$[\ x_i,\ y_i,\ z_i\ ]$ = orientation matrix of the i^{th} coordinate system established at link i with respect to the base coordinate system It is the upper left 3x3 partitioned matrix of ${}^0T_i$

p_i = position vector which points from the origin of the base coordinate system to the origin of the i^{th} coordinate system It is the upper 3x1 partitioned matrix of ${}^0T_i$ The general coordinate transformation matrix ${}^{i-1}A_i$ can be written as

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.2) $$

Specifically, for $i = 6$, the T matrix, $T = {}^{0}A_6$, specifies the position and orientation of the endpoint of the manipulator with respect to the base coordinate system This T matrix is often referred to as the Arm Matrix T can be written in the form

$$T = \begin{bmatrix} x_6 & y_6 & z_6 & p_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3 3}$$

where

n = normal vector of the hand

s = sliding vector of the hand

a = approach vector of the hand

p = position vector of the hand It points from the origin of the base coordinate system to the origin of the hand coordinate system, which is usually located at the centre point of the fully closed fingers

The direct kinematics solution of a six-link manipulator is, therefore, simply a matter of calculating $T = {}^{0}A_6$ by chain multiplying the six ${}^{i-1}A_i$ matrices and evaluating each element in the T matrix [10] The direct kinematics solution yields a unique T matrix for a given $q = (q_1, q_2, \; q_6)^T$ and a given set of coordinate systems, where $q_i = \theta_i$ for a rotary joint and $q_i = d_i$ for a prismatic joint

Having obtained all the coordinate transform matrices ${}^{i-1}A_i$ for a robot arm, the next task is to compute T efficiently Let $T = T_1 T_2$ where $T_1 = {}^{0}A_1 \, {}^{1}A_2 \, {}^{2}A_3$ and $T_2 = {}^{3}A_4 \, {}^{4}A_5 \, {}^{5}A_6$

For a PUMA 560 series robot, T_1 is found to be ·

$$T_1 = \begin{bmatrix} C_1C_{23} & -S_1 & C_1S_{23} & a_2C_1C_2 + a_3C_1C_{23} \\ & & & - d_2S_1 \\ S_1C_{23} & C_1 & S_1S_{23} & a_2S_1C_2 + a_3S_1C_{23} \\ & & & + d_2C_1 \\ -S_{23} & 0 & C_{23} & -a_2S_2 - a_3S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3 4}$$

and the T_2 matrix is found to be

$$T_2 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 & d_6C_4S_5 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & d_6S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 & d_6C_5+d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3 5}$$

where $C_{ij} = \cos(\theta_i+\theta_j)$ and $S_{ij} = \sin(\theta_i+\theta_j)$

The arm matrix T for the PUMA robot arm is found to be

$$T = T_1\ T_2$$

$$= \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3 6}$$

where

$$\begin{aligned} n_x &= C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_1(S_4C_5C_6 + C_4S_6) \\ n_y &= S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + C_1(S_4C_5C_6 + C_4S_6) \\ n_z &= -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6 \end{aligned} \tag{3 7}$$

$$\begin{aligned} s_x &= C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] - S_1(-S_4C_5S_6 + C_4C_6) \\ s_y &= S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] + C_1(-S_4C_5S_6 + C_4C_6) \\ s_z &= S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6 \end{aligned} \tag{3 8}$$

$$\begin{aligned} a_x &= C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5 \\ a_y &= S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5 \\ a_z &= -S_{23}C_4S_5 + C_{23}C_5 \end{aligned} \tag{3 9}$$

$$p_x = C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1(d_6S_4S_5 + d_2)$$
$$p_y = S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] + C_1(d_6S_4S_5 + d_2)$$
$$p_z = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} - a_2S_2 \tag{3 10}$$

This is the solution to the direct kinematics problem [10]

3 1.2 The Inverse Kinematics Solution

This section represents a geometric approach to solving the inverse kinematics problem of six-link manipulators with rotary joints [10] An algebraic solution exists for the inverse solution also, [4], [5] and [13], but ambiguity exists in the solution. Based on the link coordinate systems and human arm geometry, various arm configurations of a PUMA-like robot can be identified with the assistance of three configuration indicators (ARM, ELBOW, and WRIST) - two associated with the solution of the first three joints and the other with the last three joints For a six-axis PUMA robot, there are four possible solutions to the first three joints and for each of these four solutions there are two possible solutions to the last three joints These configuration indicators allow one to determine one solution from the eight possible solutions These arm configuration indicators are prespecified by a user for finding the inverse solution

The solution is calculated in two stages First, a position vector pointing from the shoulder to the wrist is derived This is used to derive the solution of each of the first three joints by looking at the projection of the position vector onto the $x_{i-1}y_{i-1}$ plane The last three joints are solved using the calculated joint solution from the first three joints, the orientation submatrices of ${}^0T_i$ and ${}^{i-1}A_i$ (i = 4,5,6), and the projection of the link coordinate frames onto the $x_{i-1}y_{i-1}$ plane [10] From the geometry, one can easily find the arm solution consistently

Arm solution for the first three joints From the kinematics diagram of the PUMA robot arm in Fig 3 1, a position vector p is defined which points from the origin of the shoulder coordinate system (x_0,y_0,z_0) to the point where the last three joint axes intersect as

$$p = p_6 - d_6a = (p_x, p_y, p_z)^T \tag{3 11}$$

which corresponds to the position vector of ${}^0T_4$

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} C_1(a_2C_2 + a_3C_{23} + d_4S_{23}) - d_2S_1 \\ S_1(a_2C_2 + a_3C_{23} + d_4S_{23}) + d_2C_1 \\ d_4C_{23} - a_3S_{23} - a_2S_2 \end{bmatrix} \tag{3 12}$$

Joint 1 solution If the position vector p is projected onto the x_0y_0 plane, the following equations are obtained for solving θ_1

$$\begin{aligned} r &= (p_x^2 + p_y^2 - d_2^2)^{\frac{1}{2}} \\ R &= (p_x^2 + p_y^2)^{\frac{1}{2}} \\ \sin\Phi &= p_y/R \\ \cos\Phi &= p_x/R \\ \sin\theta_1 &= \sin(\Phi - \alpha) \\ \cos\theta_1 &= \cos(\Phi - \alpha) \end{aligned} \tag{3 13}$$

Therefore

$$\begin{aligned} \theta_1 &= \tan^{-1}\left(\frac{\sin\theta_1}{\cos\theta_1}\right) \qquad -\pi \leqslant \theta_1 \leqslant \pi \\ &= \tan^{-1}\left(\frac{-\mathrm{ARM}\ p_y(p_x^2 + p_y^2 - d_2^2)^{\frac{1}{2}} - p_xd_2}{-\mathrm{ARM}\ p_x(p_x^2 + p_y^2 - d_2^2)^{\frac{1}{2}} + p_yd_2}\right) \end{aligned} \tag{3 14}$$

Joint 2 solution The position vector p is projected onto the x_1y_1 plane. Four different arm configurations exist. From table (3 1), θ_2 can be expressed in one equation for different arm and elbow configurations using the ARM and ELBOW indicators as

$$\theta_2 = \alpha + (\mathrm{ARM\ ELBOW})\beta \tag{3 15}$$

Table 3.1 Arm Configurations for joint 2

Arm Configurations	θ_2	ARM	ELBOW	ARM.ELBOW
Left and Above Arm	$\alpha-\beta$	-1	+1	-1
Left and Below Arm	$\alpha+\beta$	-1	-1	+1
Right and Above Arm	$\alpha+\beta$	+1	+1	+1
Right and Below Arm	$\alpha-\beta$	+1	-1	-1

From the arm geometry, one obtains

$$R = (p_x^2 + p_y^2 + p_z^2 - d_2^2)^{\frac{1}{2}}$$
$$r = (p_x^2 + p_y^2 - d_2^2)^{\frac{1}{2}}$$

$$\sin\alpha = -p_z/R$$
$$\cos\alpha = -\mathrm{ARM}\ r/R$$
$$\cos\beta = \frac{a_2^2 + R^2 - (d_4^2 + a_3^2)}{2a_2R}$$

$$\sin\beta = (1 - \cos^2\beta)^{\frac{1}{2}} \qquad (3\ 16)$$

Getting the sine and cosine functions of θ_2

$$\sin\theta_2 = \sin(\alpha + \mathrm{ARM}\ \mathrm{ELBOW}\ \beta)$$
$$\cos\theta_2 = \cos(\alpha + \mathrm{ARM}\ \mathrm{ELBOW}\ \beta) \qquad (3\ 17)$$

Thus, the solution for θ_2 is

$$\theta_2 = \tan^{-1}\left(\frac{\sin\theta_2}{\cos\theta_2}\right) \qquad -\pi \leqslant \theta_2 \leqslant \pi \qquad (3\ 18)$$

Joint 3 solution For joint 3, the position vector p is projected onto the x_2y_2 plane From the arm geometry, the following equations are obtained for θ_3

$$R = (p_x^2 + p_y^2 + p_z^2 - d_2^2)^{\frac{1}{2}}$$

$$\cos\Phi = \frac{a_2^2 + (d_4^2 + a_3^2) - R^2}{2a_2(d_4^2 + a_3^2)^{\frac{1}{2}}}$$

$$\sin\Phi = \mathrm{ARM}\ \mathrm{ELBOW}\ (1 - \cos^2\Phi)^{\frac{1}{2}}$$

$$\sin\beta = \frac{d_4}{(d_4^2 + a_3^2)^{\frac{1}{2}}}$$

$$\cos\beta = \frac{|a_3|}{(d_4^2 + a_3^2)^{\frac{1}{2}}} \qquad (3\ 19)$$

From table (3 2), θ_3 can be expressed in one equation for different arm configurations

$$\theta_3 = \Phi - \beta \qquad (3\ 20)$$

Table 3.2 Arm Configurations for joint 3

Arm Configurations	θ_3	ARM	ELBOW	ARM.ELBOW
Left and Above Arm	Φ-β	-1	+1	-1
Left and Below Arm	Φ-β	-1	-1	+1
Right and Above Arm	Φ-β	+1	+1	+1
Right and Below Arm	Φ-β	+1	-1	-1

Again, obtaining the sine and cosine functions of θ_3 gives

$$\sin\theta_3 = \sin(\Phi - \beta)$$
$$\cos\theta_3 = \cos(\Phi - \beta) \qquad (3\ 21)$$

Thus, the solution for θ_3 is

$$\theta_3 = \tan^{-1}\left[\frac{\sin\theta_3}{\cos\theta_3}\right] \qquad -\pi \leqslant \theta_3 \leqslant \pi$$

$$(3\ 22)$$

Arm solution for the last three joints Knowing the first three joint angles, ${}^0T_3$ which is used extensively in the solution of the last three joints, can be evaluated The solution to the last three joints of a PUMA robot arm can be found by setting these joints to meet the following criteria

1 Set joint 4 such that a rotation about joint 5 will align the axis of motion of joint 6 with the given approach vector

2 Set joint 5 to align the axis of motion of joint 6 with the approach vector

3 Set joint 6 to align the given orientation vector (or sliding vector or y_6) and normal vector

Joint 4 solution Starting with the assumption that the vector cross product (z x a) has a positive sign, define an orientation indicator Ω as

$$\Omega = \begin{cases} 0 & \text{the degenerate case i e } z_3 \text{ is parallel to a} \\ s \cdot y_5 & \text{if } s \cdot y_5 \neq 0 \\ n \cdot y_5 & \text{if } s \cdot y_5 = 0 \end{cases} \qquad (3\ 23)$$

The degenerate case happens when the axes of rotation for joints 4 and 6 are parallel Looking at the projection of the coordinate frame (x_4, y_4, z_4) on the $x_3 y_3$ plane the following results

$$\sin\theta_4 = -M\,(z_4 \cdot x_3)$$
$$\cos\theta_4 = M\,(z_4 \cdot y_3) \qquad (3\ 24)$$

where x_3 and y_3 are the x and y column vectors of ${}^0T_3$, respectively, M = WRIST sign(Ω)

Thus the solution for θ_4 with the orientation and WRIST indicators is

$$\theta_4 = \tan^{-1}\left[\frac{\sin\theta_4}{\cos\theta_4}\right] \qquad -\pi \leqslant \theta_4 \leqslant \pi$$
$$= \tan^{-1}\left[\frac{M\,(C_1 a_y - S_1 a_x)}{M\,(C_1 C_{23} a_x + S_1 C_{23} a_y - S_{23} a_z)}\right]$$

$$(3\ 25)$$

If the degenerate case occurs, any convenient value may be chosen for θ_4 as long as the orientation of the wrist is satisfied

Joint 5 solution To find θ_5, the criterion that aligns the axis of rotation of joint 6 with the approach vector (or a = z_5) is used Looking at the projection of the coordinate frame (x_5, y_5, z_5) on the $x_4 y_4$ plane, it can be shown that the following are true

$$\sin\theta_5 = a \cdot x_4$$
$$\cos\theta_5 = -(a \cdot y_4) \qquad (3\ 26)$$

where x_4 and y_4 are the x and y column vectors of ${}^0T_4$, respectively, and a is the approach vector Thus the solution for θ_5 is

$$\theta_5 = \tan^{-1}\left[\frac{\sin\theta_5}{\cos\theta_5}\right] \qquad -\pi \leqslant \theta_5 \leqslant \pi$$
$$= \tan^{-1}\left[\frac{(C_1C_{23}C_4-S_1S_4)a_x + (S_1C_{23}C_4+C_1S_4)a_y - C_4S_{23}a_z}{C_1S_{23}a_x + S_1S_{23}a_y + C_{23}a_z}\right] \tag{3 27}$$

If θ_5 is approximately zero, then the degenerate case occurs

Joint 6 solution The orientation of the gripper is aligned to ease picking up the object The criterion for doing this is to set s = y_6 Looking at the projection of the hand coordinate frame (n,s,a) on the x_5y_5 plane, it can be shown that the following are true

$$\sin\theta_6 = n\ y_5$$
$$\cos\theta_6 = s\ y_5 \tag{3 28}$$

where y_5 is the column vector of ${}^0T_5$ and n and s are the normal and sliding vectors of ${}^0T_6$, respectively Thus, the solution for θ_6 is

$$\theta_6 = \tan^{-1}\left[\frac{\sin\theta_6}{\cos\theta_6}\right]$$
$$= \tan^{-1}\left[\frac{(-S_1C_4-C_1C_{23}S_4)n_x + (C_1C_4-S_1C_{23}S_4)n_y + S_4S_{23}n_z}{(-S_1C_4-C_1C_{23}S_4)s_x + (C_1C_4-S_1C_{23}S_4)s_y + S_4S_{23}s_z}\right] \tag{3 29}$$

The above derivation of the inverse kinematics solution of a PUMA robot arm is based on the geometric interpretation of the position of the endpoint of link three and the hand (or tool) orientation requirement There is one pitfall m the above derivation for θ_4, θ_5 and θ_6 The criterion for setting the axis of motion of joint five equal to the cross product of z_3 and a may not be valid when $\sin\theta_5$ is approximately zero, which means θ_5 is approximately zero In this case, the manipulator becomes degenerate with both the axis of motion of joints four and six aligned In this state, only the sum of θ_4 and θ_6 is significant.

In summary, there are eight solutions to the inverse kinematics problem of a six-joint PUMA-like robot arm The first three-joints solution (θ_1, θ_2, θ_3) position the arm while the last three-joint solution (θ_4, θ_5, θ_6), provides appropriate orientation for the hand There are four solutions for the first three-joint solutions - two for the right shoulder arm configuration and two for the left shoulder arm configuration

3.2 Planning of Manipulator trajectories

Having already discussed the kinematics of a serial link manipulator, the generation of suitable trajectories is discussed here It is assumed that there are no obstacles in the path which must be traversed (no obstacle constraints) This section focuses attention on the various trajectory planning schemes for obstacle-free motion

Trajectory planning schemes generally interpolate or approximate the desired path by a class of polynomial functions and generate a sequence of time based control setpoints for the control of the manipulator from the initial location to its destination. Path endpoints can be specified either m joint coordinates or in cartesian coordinates However, they are usually specified in cartesian coordinates because it is easier to visualize the correct end-effector configurations m cartesian coordinates than in joint coordmates

Quite frequently, there exists a number of possible trajectories between the two given endpoints For example, one may want to move the manipulator along a straight-line path that connects the endpoints (straight-line trajectory), or to move the manipulator along a smooth, polynomial trajectory that satisfies the position and orientation constraints at both endpoints (joint-interpolated trajectory) In this section only the latter is considered Simple trajectory planning that specifies path constraints is discussed

To servo a manipulator, it is required that its robot arm's configuration at both the initial and final locations must be specified before the motion trajectory is planned In planning a joint-interpolated motion trajectory for a robot arm, Paul [14] showed that the following considerations are of interest

1 When picking up an object, the motion of the hand must be directed away from an object, otherwise the hand may crash into the supporting surface of the object.

2 If the departure velocity (lift-off point) is specified along the normal vector to the surface out from the initial position, and if the hand is required to pass through this position, then an admissible departure motion is attained. If the time to reach this position could be specified, then the speed at which the object is to be lifted can be controlled

3 The same set of lift-off requirements for the arm motion is also true for the set-down point of the final position motion so that the correct approach direction can be obtained and controlled

4 From the above, one can see that there are four positions for each arm motion initial, lift-off, set-down and final (see Fig 3 2)

5 Position Constraints

a Initial position velocity and acceleration are given (normally zero)

b Lift-off position continuous motion for intermediate points

c Set-down position same as lift-off position

d Final position velocity and acceleration are given (normally zero)

6 In addition to these constraints, the extrema of all the joint trajectories must be within the physical and geometric limits

7 Time Considerations

a Initial and final trajectory segments time is based on the rate of approach of the hand to and from the surface and is some fixed constant based on the characteristics of the joint motors

b Intermediate points or midtrajectory segment time is based on the maximum velocity and acceleration of the joints, and the maximum of these times is used (i e the maximum time of the slowest joint is used for normalization)

Based on these considerations, one is concerned with selecting a class of polynomial functions of degree n or less such that the required joint position, velocity and acceleration at these knot points (initial, lift-off, set-down and final position) are satisfied, and the joint position, velocity and acceleration are continuous on the entire trajectory time interval One approach is to specify a seventh-degree polynomial for each joint i,

$$q_i(t) = a_7 t^7 + a_6 t^6 + a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{3 30}$$

where the unknown coefficients a_j can be determined from the known positions and continuity conditions However, the use of such a high-degree polynomial to interpolate the given knot points may not be satisfactory It is difficult to find its extrema and it tends to have extraneous motion [10] An alternative approach is to split the entire joint trajectory into several trajectory segments so that the different interpolating polynomials of a lower degree can be used to interpolate in each trajectory segment. There are different ways a joint trajectory can be split, and each method possesses different properties The most common methods are the following

1. 4-3-4 Trajectory Each joint has the following three trajectory segments the first segment is a fourth-degree polynomial specifying the trajectory from the initial position to the lift-off position The second trajectory segment (or midtrajectory segment) is a third-degree polynomial specifying the trajectory from the lift-off to the set-down position The last trajectory segment is a fourth-degree polynomial specifying the trajectory from the set-down position to the final position

2. 3-5-3 Trajectory Same as the 4-3-4 trajectory, but this uses polynomials of different degrees for each segment a third-degree polynomial for the first segment, a fifth-degree polynomial for the second segment, and a third-degree polynomial for the last segment.

3. 5-Cubic Trajectory Cubic spline functions of third-degree polynomials for five trajectory segments are used

In the next sections, a detailed derivation for generating 4-3-4 and Cubic Spline trajectories is given Note the calculation of a 3-5-3 trajectory is very similar to the 4-3-4 method

3.2.1 Calculation of a 4-3-4 Trajectory

Since N joint trajectories are to be determined in each trajectory segment, it is convenient to introduce a normalized time variable, $t \in [0,1]$, which allows one to treat the equations of each segment for each joint angle in the same way, with time varying from $t = 0$ (initial time for all trajectory segments) to $t = 1$ (final time for all trajectory segments) Let us define the following variables

t normalized time variable, $t \in [0,1]$
τ real time in seconds

τ_i real time at the end of the i^{th} trajectory segment
$t_i = \tau_i - \tau_{i-1}$ real time to travel through the i^{th} segment

$$t = \frac{\tau - \tau_{i-1}}{\tau_i - \tau_{i-1}} \quad , \ \tau \in [\tau_{i-1}, \tau_i] \ , \ t \in [0,1]$$

The trajectory consists of the polynomial sequences, $h_i(t)$, which together form the trajectory for joint j The polynomial equations for each joint variable in each trajectory segment expressed in normalized time are .

$$h_1(t) = a_{14}t^4 + a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \quad (1^{st} \text{ segment}) \qquad (3\ 31)$$

$$h_2(t) = a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20} \quad (2^{nd} \text{ segment}) \qquad (3\ 32)$$

$$h_n(t) = a_{n4}t^4 + a_{n3}t^3 + a_{n2}t^2 + a_{n1}t + a_{n0} \quad (\text{last segment}) \qquad (3\ 33)$$

The subscript of each polynomial equation indicates the segment number, and n indicates the last trajectory segment The unknown coefficient a_{ji} indicates the i^{th} coefficient for the j trajectory segment of a joint segment The boundary conditions that this set of joint trajectory segment polynomials must satisfy are [10]

1 Initial position = θ_0 = $\theta(t_0)$
2 Magnitude of initial velocity = v_0 (normally zero)
3 Magnitude of initial acceleration = a_0 (normally zero)
4 Lift-off = θ_1 = $\theta(t_1)$
5 Continuity in position at t_1 [i e $\theta(t_1^-) = \theta(t_1^+)$]
6 Continuity in velocity at t_1 [i e $v(t_1^-) = v(t_1^+)$]
7 Continuity in acceleration at t_1 [i e $a(t_1^-) = a(t_1^+)$]
8 Set-down position = θ_2 = $\theta(t_2)$
9 Continuity in position at t_2 [i e $\theta(t_2^-) = \theta(t_2^+)$]
10 Continuity in velocity at t_2 [i e $v(t_2^-) = v(t_2^+)$]
11 Continuity in acceleration at t_2 [i e $a(t_2^-) = a(t_2^+)$]
12 Final position = θ_f = $\theta(t_f)$
13 Magnitude of final velocity = v_f (normally zero)
14 Magnitude of final acceleration = a_f (normally zero)

The boundary conditions for the 4-3-4 joint trajectory are shown in Fig 3 3 The first and second derivatives of these polynomial equations with respect to real time τ can be written as

$$v_i(t) = \frac{dh_i(t)}{d\tau} \qquad i = 1,2,n$$

$$= \frac{1}{t_i}\frac{dh_i(t)}{d\tau} = \frac{1}{t_i} h_i(t) \qquad (3\ 34)$$

and

$$a_i(t) = \frac{d^2h_i(t)}{d\tau^2} \qquad i = 1,2,n$$

$$= \frac{1}{t_i{}^2}\frac{d^2h_i(t)}{d\tau^2} = \frac{1}{t_i{}^2}\ddot{h}_i(t) \qquad (3\ 35)$$

For the first trajectory segment, the governing polynomial equation is of fourth degree

$$h_1(t) = a_{14}t^4 + a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \qquad (3\ 36)$$

From equations (3 36) and (3 37), its first two derivatives with respect to real time are

$$v_1(t) = \frac{4a_{14}t^3 + 3a_{13}t^2 + 2a_{12}t + a_{11}}{t_1} \qquad (3\ 37)$$

and

$$a_1(t) = \frac{12a_{14}t^2 + 6a_{13}t + 2a_{12}}{t_1{}^2} \qquad (3\ 38)$$

1 For $t = 0$ (at the initial position of this trajectory segment) Satisfying the boundary conditions at this position leads to

$$a_{10} = \theta_0 \qquad (3\ 39)$$
$$v_0 = a_{11}/t_1 \qquad (3\ 40)$$

which gives

$$a_{11} = v_0 t_1 \qquad (3\ 41)$$

and

$$a_0 = 2a_{12}/t_1{}^2 \qquad (3\ 42)$$

which yields

$$a_{12} = \tfrac{1}{2}a_0 t_1{}^2 \qquad (3\ 43)$$

With these unknowns determined, equation (3 36) can be rewritten as

$$h_1(t) = a_{14}t^4 + a_{13}t^3 + (\tfrac{1}{2}a_0t_1{}^2)t^2 + (v_0t_1)t + \theta_0 \tag{3 44}$$

2 For t = 1 (at the final position of this trajectory segment) At this position, the requirements that the interpolating polynomial must pass through the position exactly is relaxed The only requirement here, is that the velocity and acceleration at this position have to be continuous with the velocity and acceleration, respectively, at the beginning of the next trajectory segment The velocity and acceleration at this position are

$$v_1(1) = \frac{4a_{14} + 3a_{13} + a_{10}t_1{}^2 + v_0t_1}{t_1} \tag{3 45}$$

$$a_1(1) = \frac{12a_{14} + 6a_{13} + a_0t_1{}^2}{t_1{}^2} \tag{3 46}$$

For the second trajectory segment, the governing polynomial equation is of the third degree

$$h_2(t) = a_{23}t^3 + a_{22}t^2 + a_{21}t + a_{20} \tag{3 47}$$

1 For t = 0 (at the lift-off position) Using equations (3 34) and (3 35), the velocity and acceleration at this point are, respectively,

$$a_{20} = \theta_2 \tag{3 48}$$

$$v_1 = a_{21}/t_2 \tag{3 49}$$

which gives,

$$a_{21} = v_1t_2 \tag{3 50}$$

and

$$a_1 = 2a_{22}/t_2{}^2 \tag{3 51}$$

which yields

$$a_{22} = \tfrac{1}{2}a_1t_2{}^2 \tag{3 52}$$

Since the velocity and acceleration at this position must be continuous with the velocity and acceleration at the end of the previous trajectory segment respectively, this gives

$$\frac{h_2(0)}{t_2} = \frac{h_1(1)}{t_1} \qquad (3\ 53)$$

and

$$\frac{h_2(0)}{t_2^2} = \frac{h_1(1)}{t_1^2} \qquad (3\ 54)$$

which, respectively leads to

$$\frac{-a_{21}}{t_2} + \frac{4a_{14}}{t_1} + \frac{3a_{13}}{t_1} + \frac{a_0 t_1^2}{t_1} + \frac{v_0 t_1}{t_1} = 0 \qquad (3\ 55)$$

and

$$\frac{-2a_{22}}{t_2^2} + \frac{12a_{14}}{t_1^2} + \frac{6a_{13}}{t_1^2} + \frac{a_0 t_1^2}{t_1^2} = 0 \qquad (3\ 56)$$

2 For $t = 1$ (at the set-down position) Again the velocity and acceleration at this position must be continuous with the velocity and acceleration at the beginning of the next trajectory segment The velocity and acceleration at this position are obtained, respectively, as

$$h_2(1) = a_{23} + a_{22} + a_{21} + a_{20} \qquad (3\ 57)$$

$$v_2(1) = \frac{3a_{23} + 2a_{22} + a_{21}}{t_2} \qquad (3\ 58)$$

and

$$a_2(1) = \frac{6a_{23} + 2a_{22}}{t_2^2} \qquad (3\ 59)$$

For the last trajectory segment, the governing polynomial equation is of fourth degree

$$h_n(t) = a_{n4}t^4 + a_{n3}t^3 + a_{n2}t^2 + a_{n1}t + a_{n0} \qquad (3\ 60)$$

Substituting $\underline{t} = t-1$ into t in the above equation, the normalized time t has been shifted from $t \in [0,1]$ to $\underline{t} \in [-1,0]$. Then equation (3.60) becomes

$$h_n(t) = a_{n4}\underline{t}^4 + a_{n3}\underline{t}^3 + a_{n2}\underline{t}^2 + a_{n1}\underline{t} + a_{n0} \qquad (3.61)$$

Using equations (3.34) and (3.35), its first and second order derivatives with respect to real time are

$$v_n(\underline{t}) = \frac{4a_{n4}\underline{t}^3 + 3a_{n3}\underline{t}^2 + 2a_{n2}\underline{t} + a_{n1}}{t_n} \qquad (3.62)$$

and

$$a_n(\underline{t}) = \frac{12a_{n4}\underline{t}^2 + 6a_{n3}\underline{t} + 2a_{n2}}{t_n^2} \qquad (3.63)$$

1. For $\underline{t} = 0$ (at the final position of this position segment). Satisfying the boundary conditions at this final position of the trajectory, one obtains

$$h_n(0) = \theta_f \qquad (3.64)$$

$$v_f = a_{n1}/t_n \qquad (3.65)$$

which gives

$$a_{n1} = v_f t_n \qquad (3.66)$$

and

$$a_f = 2a_{n2}/t_n^2 \qquad (3.67)$$

which yields

$$a_{n2} = \tfrac{1}{2}a_f t_n^2 \qquad (3.68)$$

2. For $\underline{t} = -1$ (at the starting position of this trajectory segment). Satisfying the boundary conditions at this position, one has, at the set-down position .

$$hn(-1) = a_{n4} - a_{n3} + \tfrac{1}{2}a_f t_n^2 - v_f t_n + \theta_f = \theta_2(1) \qquad (3.69)$$

and

$$\frac{h_n(-1)}{t_n} = \frac{-4a_{n4} + 3a_{n3} - a_f t_n^2 + v_f t_n}{t_n} \tag{3 70}$$

and

$$\frac{h_n(-1)}{t_n^2} = \frac{12a_{n4} - 6a_{n3} + a_f t_n^2}{t_n^2} \tag{3 71}$$

The velocity and acceleration continuity conditions at this set-down point lead to the following equations

$$\frac{4a_{n4} - 3a_{n3} + a_f t_n^2 - v_f t_n}{t_n} + \frac{3a_{23}}{t_2} + \frac{2a_{22}}{t_2} + \frac{a_{21}}{t_2} = 0 \tag{3 72}$$

and

$$\frac{-12a_{n4} + 6a_{n3} - a_f t_n^2}{t_n^2} + \frac{6a_{23}}{t_2^2} + \frac{2a_{22}}{t_2^2} = 0 \tag{3 73}$$

The difference of joint angles between successive trajectory segments can be found to be

$$\delta_1 = \theta_1 - \theta_0 = h_1(1) - h_1(0) = a_{14} + a_{13} + \tfrac{1}{2}a_0 t_1^2 + v_0 t_1 \tag{3 74}$$

$$\delta_2 = \theta_2 - \theta_1 = h_2(1) - h_2(0) = a_{23} + a_{22} + a_{21} \tag{3 75}$$

$$\delta_n = \theta_f - \theta_2 = h_n(0) - h_n(-1) = -a_{n4} + a_{n3} - \tfrac{1}{2}a_f t_n^2 + v_f t_n \tag{3 76}$$

All the unknown coefficients of the trajectory polynomial equations can be determined simultaneously solving equations (3 55), (3 56), (3 72), (3 73), (3 74), (3 75), and (3 76) Rewriting these equations in matrix vector notation, one obtains

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{3 77}$$

where

$$y = \left(\delta_1 - \tfrac{1}{2}a_0 t_1^2 - v_0 t_1,\ -a_0 t_1 - v_0,\ -a_0,\ \delta_2,\ -a_f t_n + v_f,\ a_f,\ \delta_n + \tfrac{1}{2}a_f t_n^2 - v_f t_n \right)^T \tag{3 78}$$

$$C = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3/t_1 & 4/t_1 & -1/t_2 & 0 & 0 & 0 & 0 \\ 6/t_1^2 & 12/t_1^2 & 0 & -2/t_2^2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1/t_2 & 2/t_2 & 3/t_2 & -3/t_n & 4/t_n \\ 0 & 0 & 0 & 2/t_2^2 & 6/t_2^2 & 6/t_n^2 & -12/t_n^2 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{3 79}$$

and

$$x = \left(a_{13},\ a_{14},\ a_{21},\ a_{22},\ a_{23},\ a_{n3},\ a_{n4} \right)^T \tag{3 80}$$

Solution to the problem is given by

$$x = C^{-1} y \tag{3 81}$$

The structure of the matrix **C** makes it easy to compute the unknown coefficients and the inverse value of **C** always exists if the time intervals t_i, $i = 1, 2, n$ are positive values Solving equation (3 81), all the coefficients for the polynomial equations for the joint trajectory segments for joint j, are obtained

The calculation of a 3-5-3 trajectory is very similar to this solution, and it is trivial to discuss it further

3.2.2 The Cubic Spline technique

The interpolation of a given function by a set of cubic polynomials, preserving continuity in the first and second derivatives at the interpolation points is known as cubic spline functions. The degree of approximation and smoothness that can be achieved is relatively good In general, a spline curve is a polynomial of degree k with continuity of derivative of order k-1, at the interpolation points Cubic splines offer several advantages First it is the lowest degree polynomial function that allows continuity in velocity and acceleration Secondly, low-degree polynomials reduce the effort of computations and the possibility of numerical instabilities [10]

The general equation of five-cubic polynomials for each joint trajectory segment is

$$h_j(t) = a_{j3}t^3 + a_{j2}t^2 + a_{j1}t + a_0 \qquad j = 1,2,3,4,n \tag{3 82}$$

with $\tau_{j-1} \leqslant \tau \leqslant \tau_j$ and $t \in [0,1]$ The unknown coefficient a_{ji} indicates the i^{th} coefficient for joint j trajectory segment and n indicates the last trajectory segment

In using five-cubic polynomial interpolation, one needs to have five trajectory segments and six interpolation points However, from the previous discussion, only four positions for interpolation exist. Thus, two extra interpolation points must be selected to provide enough boundary conditions for solving the unknown coefficients in the polynomial sequences These two extra knot points are chosen between lift-off and set-down positions It is not necessary to know these locations exactly The boundary conditions for a five-cubic joint trajectory are shown in Fig 3 4

The first and second derivatives of the polynomials with respect to real time are given by

$$v_j(t) = \frac{h_j(t)}{t_j} = \frac{3a_{j3}t^2 + 2a_{j2}t + a_{j1}}{t_j} \tag{3 83}$$

and

$$a_j(t) = \frac{h_j(t)}{{t_j}^2} = \frac{6a_{j3}t + 3a_{j2}}{{t_j}^2} \tag{3 84}$$

where t_j is the real time required to travel through the j^{th} trajectory segment. Given the positions, velocities and accelerations at the initial and final positions, the polynomial equations for the initial and final trajectory segments [$h_1(t)$ and $h_n(t)$] are completely determined Once these polynomial equations are calculated, $h_2(t)$, $h_3(t)$ and $h_4(t)$ can be determined using the position constraints and continuity conditions

Because the derivation of the solution to the Cubic Spline trajectory is similar to the 4-3-4 technique, a detailed discussion of the derivation of this technique is not included, only the solution to the problem is given here. The coefficients for five polynomial segments are found using the boundary conditions, position constraints and continuity conditions

For the first trajectory segment, the governing polynomial equation is

$$h_1(t) = a_{13}t^3 + a_{12}t^2 + a_{11}t + a_{10} \tag{3 85}$$

where

$$a_{10} = \theta_0$$
$$a_{11} = v_0 t_1$$
$$a_{12} = \tfrac{1}{2}a_0 t_1^2$$
$$a_{13} = \delta_1 - v_0 t_1 - \tfrac{1}{2}a_0 t_1^2 \quad \text{and } \delta_1 = \theta_1 - \theta_{1-1} \tag{3 86}$$

For the last trajectory segment, the solution for the unknown coefficients is

$$a_{n0} = \theta_4$$
$$a_{n1} = 3\delta_n - 2v_f t_n + \tfrac{1}{2}a_f t_n^2$$
$$a_{n2} = -3\delta_n + 3v_f t_n - a_f t_n^2$$
$$a_{n3} = \delta_n - v_f t_n + \tfrac{1}{2}a_f t_n^2 \quad \text{where } \delta_n = \theta_f - \theta_4 \tag{3 87}$$

Using the solution for the first and last trajectory segments, the solution to the remaining three segments can be found

The solution to these segments is given by the following .

$$a_{20} = \theta_1 \tag{3 88a}$$
$$a_{21} = v_1 t_2 \tag{3 88b}$$
$$a_{22} = \tfrac{1}{2}a_1 t_2^2 \tag{3 88c}$$

$$a_{30} = \theta_2 \tag{3 89a}$$
$$a_{31} = v_2 t_3 \tag{3 89b}$$
$$a_{32} = \tfrac{1}{2}a_2 t_3^2 \tag{3 89c}$$

$$a_{40} = \theta_3 \tag{3 90a}$$
$$a_{41} = v_3 t_4 \tag{3 90b}$$
$$a_{42} = \tfrac{1}{2}a_3 t_4^2 \tag{3 90c}$$

The a_{13} coefficient is calculated as follows [10] :

$$a_{23} = t_2^2 x_1 / D \tag{3 91a}$$
$$a_{33} = t_3^2 x_2 / D \tag{3 91b}$$
$$a_{43} = t_4^2 x_3 / D \tag{3 91c}$$

with

$$x_1 = k_1(u-t_2) + k_2(t_4^2-d) - k_3[(u-t_4)d + t_4^2(t_4-t_2)] \quad (3\ 92a)$$

$$x_2 = -k_1(u+t_3) + k_2(c-t_4^2) + k_3[(u-t_4)c + t_4^2(u-t_2)] \quad (3\ 92b)$$

$$x_3 = k_1(u-t_4) + k_2(d-c) + k_3[(t_4-t_2)c - d(u-t_2)] \quad (3\ 92c)$$

$$D = u(u-t_2)(u-t_4) \quad (3\ 93)$$

$$u = t_2 + t_3 + t_4 \quad (3\ 94)$$

$$k_1 = \theta_4 - \theta_1 - v_1 u - \tfrac{1}{2}a_1 u^2 \quad (3\ 95a)$$

$$k_2 = \frac{v_4 - v_1 - a_1 u - \tfrac{1}{2}(a_4 - a_1)u}{3} \quad (3\ 95b)$$

$$k_3 = \frac{a_4 - a_1}{6} \quad (3\ 95c)$$

$$c = 3u^2 - 3ut_2 + t_2^2 \quad (3\ 96)$$

$$d = 3t_4^2 + 3t_3t_4 + t_3^2 \quad (3\ 97)$$

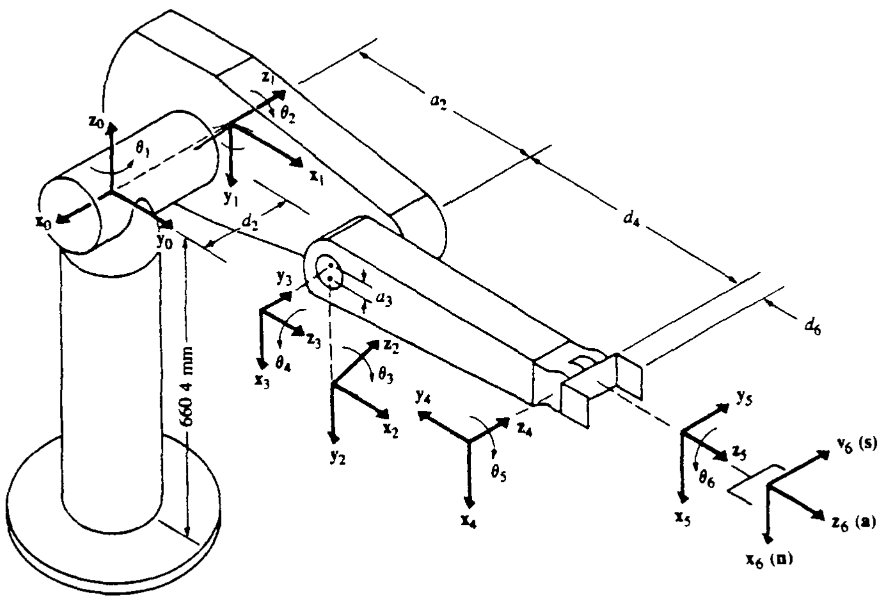
Five-cubic polynomial equations can be uniquely determined to satisfy all the position constraints and continuity conditions given the initial, the lift-off, the set-down, and the final positions, as well as the time to travel each trajectory t_i

3.3 Summary

Both direct and indirect kinematics are discussed in this chapter The parameters of robot arm links and joints are defined and a 4x4 homogeneous transformation matrix is introduced to describe the location of a link with respect to a fixed coordinate frame The forward kinematic equations for a six-axis PUMA like robot are derived

The inverse kinematics problem is solved using a geometric approach, with the assistance of three arm configuration indicators (ARM, ELBOW and WRIST) The validity of the forward and inverse kinematics solution was verified by computer simulation The geometric approach, with appropriate modification and adjustment, can be generalized to other simple industrial robots with rotary joints.

The generation of efficient trajectories for manipulator control is discussed in detail also Joint-interpolated trajectories are discussed with special emphasis on the 4-3-4 and the Cubic Spline techniques Software programs were developed to implement the solution to these schemes, so that path generation could be achieved quickly and efficiently



PUMA robot arm link coordinate parameters					
Joint i	θ_i	α_i	a_i	d_i	Joint range
1	90	−90	0	0	−160 to +160
2	0	0	431 8 mm	149 09 mm	−225 to 45
3	90	90	−20 32 mm	0	−45 to 225
4	0	−90	0	433 07 mm	−110 to 170
5	0	90	0	0	−100 to 100
6	0	0	0	56 25 mm	−266 to 266

Fig. 3.1 *Establishing link coordinate systems for a PUMA robot*

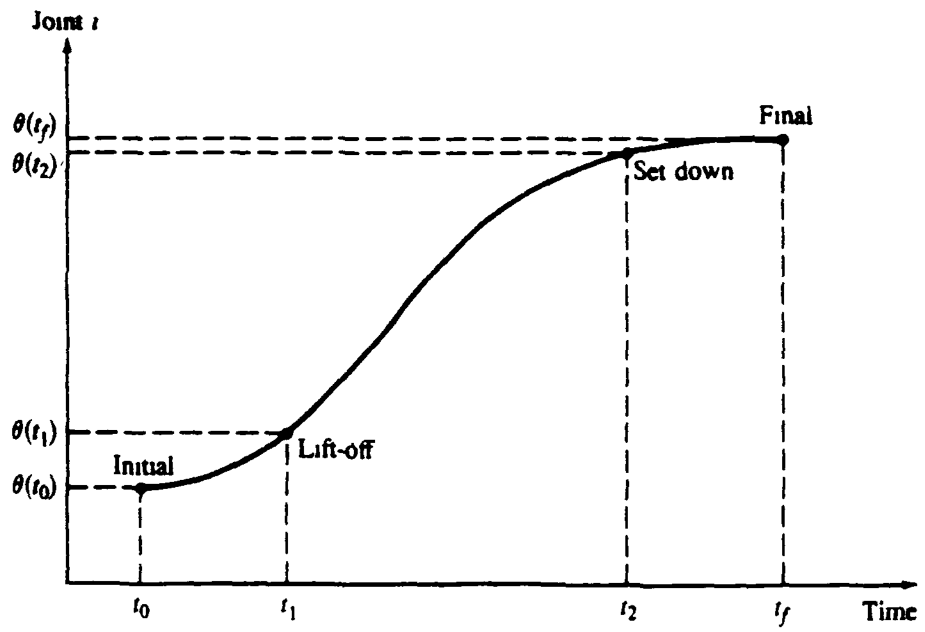


Fig. 3.2 *Position conditions for a joint trajectory*

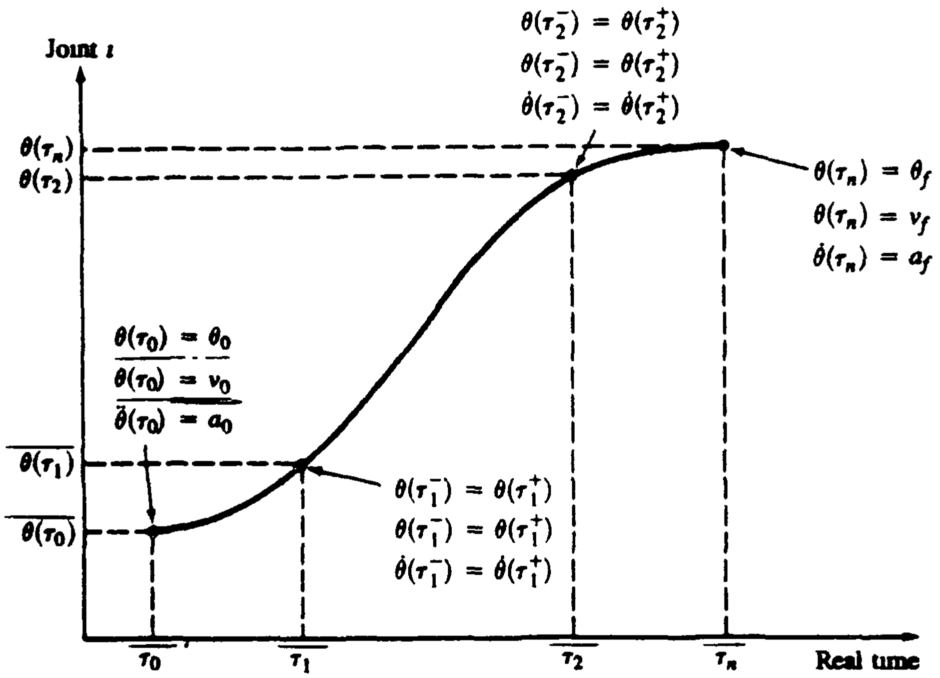


Fig. 3.3 *Boundary conditions for a 4-3-4 joint trajectory*

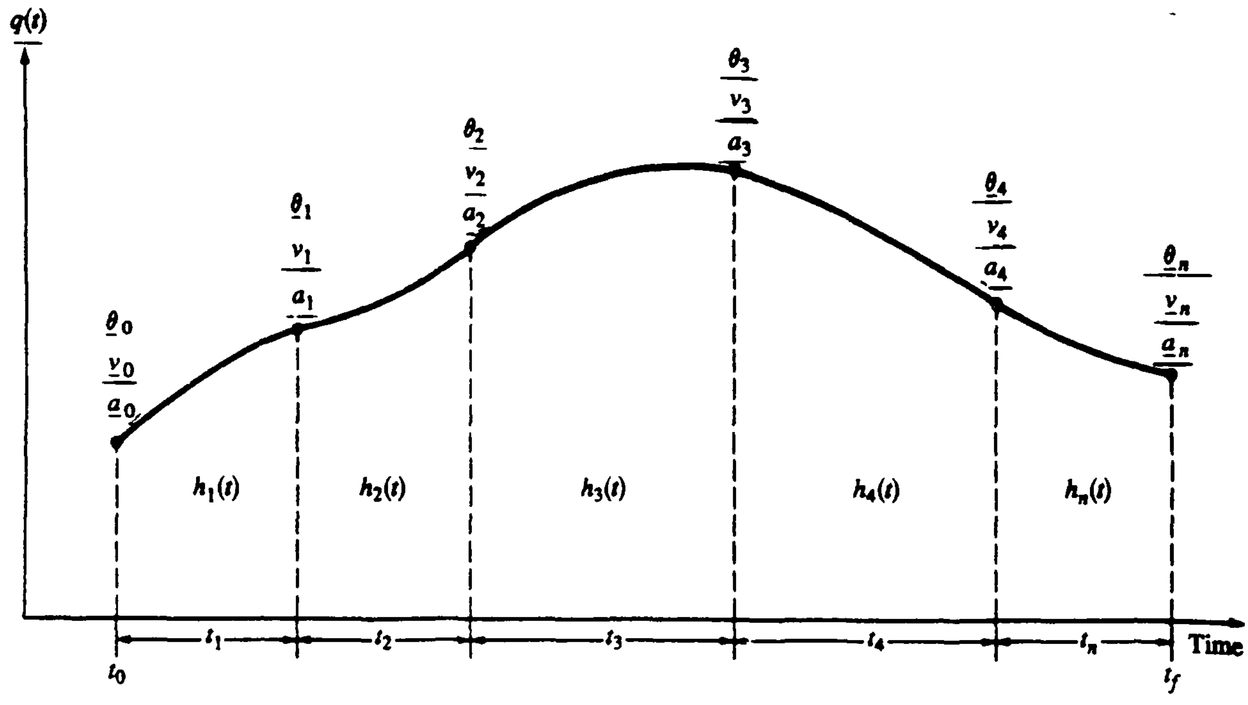


Fig. 3.4 *Boundary conditions for a 5-cubic joint trajectory*

CHAPTER 4

FIXED PARAMETER LINEAR CONTROL TECHNIQUES

In this chapter several linear control techniques are investigated Each technique is applied to the nonlinear manipulator model developed in Chapter 2 and evaluated according to its performance in a simulation environment. The techniques presented here were chosen as a suitable representation of the control methods available in this area Their suitability for manipulator control is determined here, and the most suitable routine is chosen from a set of performance criteria

The results here are also influential in later chapters The choice of adaptive control algorithms is determined partially by the performance of their fixed parameter versions This chapter also gives an insight into the difficulty of robot control due to the high degree of nonlinearity present in the system and shows why complex control algorithms are required for high precision accuracy in the control action

The existing Unimation system implements a PID control strategy The control gains are detuned to give a stable performance over the full operating range of the robot

4.1 Digital PID Control Techniques

For many control applications, it is sufficient to use a standard PID-controller In this section, different ways to implement digital PID-controllers are discussed, together with some operational aspects In the continuous time domain, the equation for a PID controller is [8]

$$u(t) = K_g \left[e(t) + \frac{1}{T_i} \int_0^t e(t)\, dt + T_d \frac{de(t)}{dt} \right] \qquad (4\ 1)$$

where,

K_g = gain factor,

T_i = integral coefficient,

T_d = differential coefficient

The above equation can be written in the complex frequency domain as

$$G(S) = K_g \left[1 + T_d\, s + \frac{1}{T_i\, s} \right] \qquad (4\ 2)$$

This type of PID-controller is called a *positive form* because the total output is calculated from the corresponding control equation. If the change in the control signal, $\Delta u(k)$, is computed instead, then this type of controller is called a *velocity*, or *incremental form* One drawback of the incremental algorithm is that it cannot operate in P- or PD-mode [15]

There are many ways to change the structure of the textbook PID-controller Fig 4 1 shows the different PID-structures, which can be used in both continuous and discrete time The structure in Fig 4 1b has the advantage that the controller does not give a large control signal at step changes in the reference signal This is the structure of the controller seen most often in the literature The 'set-point-on-I-only' controller m Fig 4 1c, is less commonly seen. The filter for the derivative part can be used in different ways. It is also possible to filter all three parts of the controller or only the proportional and the derivative parts The latter will attenuate high-frequency measurement noise [15]

The different structures in Fig 4 1 can be described using a common form as (see Fig 4 2)

$$R(z)U(z) = T(z)U_c(z) - S(z)Y(z) \qquad (4\ 3)$$

where the interpretation of the polynomials T and S depends on the structure All three polynomials are of second order and

$$R(z) = (z + \delta)(z - 1) \qquad (4\ 4)$$

in all cases From Fig 4 2, the closed-loop system is given by

$$Y(z) = \frac{BT}{AR + BS} U_c(z) + \frac{AR}{AR + BS} W(z) \qquad (4\ 5)$$

The closed-loop poles can be made the same for all structures This means that all four controllers can be tuned such that the closed-loop systems get exactly the same pulse-transfer operator from the process disturbance to the output However, the polynomial T will depend on the form of the controller, and T will introduce two zeros in the pulse-transfer operator from u_c to y The values of the zeros will depend on the form of the controller and the polynomials R and S For the forms in Fig 4 1a, Fig 4 1b and Fig 4 1d, T will have two nonzero zeros, while the form in Fig 4 1c gives one zero at the origin and one that is nonzero It is also possible to get a polynomial T with two zeros at the origin This can be achieved using the structure in Fig 4 1c This structure is advantageous if the method for tuning the parameters in the controller is based on pole placement [15]

Tuning Rules The discrete-time PID-controllers have the advantage that they look and behave as continuous PID-controllers when the sampling interval is short. Thus there is no educational problem if a controller is redesigned into digital form, so the same heuristic rules for tuning a PID-controller can be used Zeiger and Nicholas [61] gave two methods for tuning the transient response method and the ultimate-sensitivity method The transient response method uses the steepest slope, R, and the delay time, L, from the unit-step response of the open-loop system The parameters are then obtained from table (4 1)

Table 4.1 Controller Parameters using the Transient Response Method

	K_d	T_i	T_d
P	1/RL		
PI	0 9/RL	3L	
PID	1 2/RL	2L	½L

In the ultimate sensitivity method, a P-controller is used first to control the system The gain of the controller, K_{max}, and the period time T_p, when the closed-loop system is on the stability boundary are measured The parameters of the controller are then obtained from table (4 2)

Table 4.2 Controller Parameters using the ultimate-sensitivity Method

	K_d	T_i	T_d
P	$\frac{1}{2}K_{max}$		
PI	$0.45K_{max}$	$T_p/1.2$	
PID	$0.6K_{max}$	$T_p/2$	$T_p/8$

The tuning rules above should only be used as a first approximation. The final tuning usually has to be done manually. There are also several other methods for tuning digital PID-controllers. Some involve a compensation for the length of the sampling interval, others use a pole placement technique for determining the controller parameters [16].

4.1.1 A PD Control Algorithm

In this section, a Proportional and Differential Controller is discussed. No integrator is present in the control action, but because an integrator is contained in the robot dynamics, PD-only may prove sufficient. Although a full PID controller will improve the static accuracy but it can often make the overall closed-loop system less stable.

The design here is not based on the Zeiger-Nicholas Method of tuning, but on pole-placement and static accuracy requirements [16].

4.1.1.1 Controller Derivation

In the continuous time domain, the transfer function for a PD controller is given by

$$G_c(s) = K_p + K_d\, s \qquad (4.6)$$

where,

K_p = proportional gain,

K_d = differential gain

Transforming directly to the discrete domain gives

$$G_c(z) = K_p + K_d\, \frac{(z-1)}{h\, z} \qquad (4.7)$$

where h is the sampling interval.

Hence

$$\Rightarrow \quad \frac{U(z)}{Y(z)} = \frac{(K_p\,h + K_d)z - K_d}{h\,z} \tag{4 8}$$

Hence, the controller equation is given by

$$u(k) = \frac{(K_p\,h + K_d)}{h}\,e(k) - \frac{K_d}{h}\,e(k-1) \tag{4 9}$$

Equation (4 9) expresses the present control input in terms of the present and past error signals Since the robot has integrative action, the control input decays to zero when the system output reaches its desired position The steady state error attains a low value to drive the system Hence PD control is only suitable for systems with an integrator in their dynamics, otherwise large steady state errors will result

4 1.1.2 Simulation Results

Recalling the simplified linear models for the primary joints from Chapter 2, pole placement design is based on these models Sampling these models using the Zero Order Hold Method [17] with a sampling interval of five milliseconds gives the following transfer functions ·

$$G_1(z) = \frac{9\ 774\mathrm{x}10^{-6}(\ z^2 + 2\ 74394z + 0\ 4369\ )}{(\ z - 1\ )(\ z - 0\ 2282\ )(\ z - 0\ 827\ )}$$

$$G_2(z) = \frac{3\ 231\mathrm{x}10^{-6}(\ z^2 + 2\ 7618z + 0\ 4376\ )}{(\ z - 1\ )(\ z - 0\ 2089)(\ z - 0\ 90312\ )}$$

$$G_3(z) = \frac{1\ 3\mathrm{x}10^{-5}(\ z^2 + 2\ 7376z + 0\ 4365\ )}{(\ z - 1\ )(\ z - 0\ 2356\ )(\ z - 0\ 8\ )}$$

The design is performed on joint 1 to demonstrate the technique The results of the design are shown for joints 2 and 3

The transfer function for a PD controller is given by

$$G_c(z) = \frac{(\ K_p\,h + K_d\ )[\ z - K_d/(K_p\,h + K_d)\ ]}{h\,z} \tag{4 10}$$

Cancelling the pole at 0 827, gives

$$K_{d_1}/(K_{p_1}\,h + K_{d_1}) = 0\;2282$$

$$\Rightarrow\quad K_{p_1}/K_{d_1} = 41\;84$$

To determine control gains, another design specification is required This part of the design is based on the static accuracy requirements The velocity error constant is defined as

$$K_V = \frac{1}{h}\lim_{z\to 1}(z-1)G_{O\,1}(z)$$

$$= \left.\frac{1\;9548\mathrm{x}10^{-3}(z^2 + 2\;74394z + 0\;4369)(K_{p_1}\,h + K_{d_1})}{z\,(z - 0\;2282)}\right|_{z=1}$$

$$= 2\;1178\,(K_{p_1}\,h + K_{d_1})$$

By specifying a value for K_V, K_{p_1} and K_{d_1} can be calculated uniquely The ratio of K_{p_1} to K_{d_1} is specified by the open-loop pole, which is to be cancelled The exact values depend on the velocity error specification but trial and error is required to attain the desired response

For joints 2 and 3 the following ratios for the gains are obtained

$$K_{p_2}/K_{d_2} = 35\;3 \qquad K_{p_3}/K_{d_3} = 49\;8$$

which cancel poles at 0 90312 and 0 8006 respectively Using the following set of gains

$$K_{p_1} = 24 \qquad K_{p_2} = 24 \qquad K_{p_3} = 24$$
$$K_{d_1} = 0\;57 \qquad K_{d_2} = 0\;679 \qquad K_{d_3} = 0\;48$$

results in the control action seen in Fig 4 3a and Fig 4 3b These graphs show that the settling time is long Therefore the proportional gains can be increased further (i e. increase the velocity error constant)

$$K_{p_1} = 48 \qquad K_{p_2} = 60 \qquad K_{p_3} = 48$$
$$K_{d_1} = 1\;14 \qquad K_{d_2} = 1\;7 \qquad K_{d_3} = 0\;96$$

These gains give a fast, overdamped response, with a low static error (see Fig 4 4a and Fig 4 4b) When reference signal is a Cubic Spline Tracjectory the PD controller performs poorly This is due to the fact that for a type-1 system (one integrator in the open-loop dynamics) the steady state approaches zero for a step input A steady state error exists when a type-1 system tries to track a higher order reference input. The gains are adjusted to the following values to reduce this steady state error

$K_{p_1} = 84 \qquad K_{p_2} = 84 \qquad K_{p_3} = 84$

$K_{d_1} = 1\ 995 \qquad K_{d_2} = 2\ 38 \qquad K_{d_3} = 1\ 68$

and a sufficient performance is attained (see Fig 4 5a, Fig 4 5b and Fig 4 5c) The peak error for each joint is

$e_{pk_1} = 0\ 12$

$e_{pk_2} = 0\ 19$

$e_{pk_3} = 0\ 12$

Next an integrator is added to the closed-loop system to see if the performance will improve further

4 1.2 A PID Control Algorithm

In this section, the performance of a classical PID-control algorithm on the manipulator model is investigated Firstly the controller equation is derived The three gains are in the forward loop, as shown in Fig 4 1a These control gains are tuned firstly using the Zeiger Nicholas Ultimate Sensitivity Method, and later tuned manually

4.1.2.1 Controller Derivation

The digital form of equation (4 1) is

$$u(k) = K_g \left[e(k) + \frac{h}{T_i} \sum_{i=0}^{k-1} e(i-1) + \frac{T_d}{h} \{ e(k)-e(k-1)\} \right] \qquad (4\ 11)$$

where h is the sampling interval

Transforming into a recursive equation, equation (4 11) becomes

$$u(k+1) = K_g\, e(k) + \left[\, u(k) - K_g\left(1 + \frac{T_d}{h}\right) e(k) + \frac{K_g\, h}{T_1}\, e(k) + \frac{K_g\, T_d}{h}\, e(k-1) \,\right] - \frac{K_g\, T_d}{h}\, e(k)$$

$$= u(k) + K_g\left(1 + \frac{T_d}{h}\right) e(k+1) + K_g\left(\frac{h}{T_1} - \frac{2T_d}{h} - 1\right) e(k) + \frac{K_g\, T_d}{h}\, e(k-1) \qquad (4\ 12)$$

$$\Rightarrow \quad G(z) = \frac{U(z)}{E(z)} = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}} \qquad (4\ 13)$$

and $u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2)$ (4 14)

where

$$q_0 = K_g\left[\, 1 + \frac{T_d}{h} \,\right]$$

$$q_1 = K_g\left[\, \frac{h}{T_1} - \frac{2T_d}{h} - 1 \,\right]$$

$$q_2 = K_g \frac{T_d}{h} \qquad (4\ 15)$$

This is the relationship between the discrete and analog coefficients

4 1.2.2 Simulation Results

A proportional controller is placed on the robot model and K_{max} (the value of proportional gain which causes the closed-loop system to oscillate) is found to have the following values for joints 1, 2 and 3

$K_{max\,1}$ = 2,000
$K_{max\,2}$ = 2,500
$K_{max\,3}$ = 2,000

T_p (the period of oscillation) is measured at the following values

$T_{p1} = 0\ 45$
$T_{p2} = 0\ 55$
$T_{p3} = 0\ 3$

These values of T_{pi} and K_{maxi} yield the following control parameter gains

$K_{g1} = 1{,}200$ $K_{g2} = 1{,}500$ $K_{g3} = 1{,}200$
$T_{i1} = 0\ 225$ $T_{i2} = 0\ 275$ $T_{i3} = 0\ 15$
$T_{d1} = 0\ 05625$ $T_{d2} = 0\ 06875$ $T_{d3} = 0\ 0375$

The control which results using these gains is shown in Fig 4 6a and Fig 4 6b The results show the closed-loop response is fast and underdamped Also the control inputs are initially excessively large The closed-loop oscillation is undesirable so in fine tuning the algorithm, the proportional gains are decreased slightly The *best gains* were found to be

$K_{g1} = 800$ $K_{g2} = 1{,}000$ $K_{g3} = 800$
$T_{i1} = 0\ 225$ $T_{i2} = 0\ 275$ $T_{i3} = 0\ 15$
$T_{d1} = 0\ 05625$ $T_{d2} = 0\ 06875$ $T_{d3} = 0\ 0375$

The results using these control gains are shown in Fig 4 7a and Fig 4 7b The response is fast and overdamped Because of the I-part of the controller there is approximately zero steady state error

$e_{ss1} = 2\ 5\mathrm{x}10^{-1\ 4}$
$e_{ss2} = 1\ 05\mathrm{x}10^{-1\ 2}$
$e_{ss3} = 2\ 24\mathrm{x}10^{-1\ 3}$

Also when the reference signal is a Cubic-Spline trajectory, the algorithm can track this trajectory with a high degree of accuracy (see Fig 4 8a, Fig 4 8b and Fig 4 8c) The peak error is acceptably low for all the three joints

$e_{pk1} = 0\ 006$
$e_{pk2} = 0\ 011$
$e_{pk3} = 0\ 008$

The performance of PID greatly outmatches the PD version, mainly due to the integrator, which eliminates the steady state error

4 1.3 Conclusion on PID-controllers

PID configurations are easy to implement. Depending on the configuration, two or three gains require tuning The control equation is a simple difference equation The properties of PID control techniques can be summarized as follows

1 The design is relatively simple, and can be based on several methods of tuning
2 The control technique is easy to implement, it requires only the calculation of a simple difference equation
3 The algorithm is suitable for use on manipulator-type robots and in fact, is presently one of the most commonly used algorithms in industry
4 The full PID network greatly improves the static accuracy over the PD configuration

The addition of an integrator into the closed-loop system can induce instability but it considerably improves the accuracy of tracking

4.2 Frequency Compensators

In this section the design of three types of compensator using frequency analysis methods, is discussed The three compensators are

1 Lead compensator
2 Lag compensator
3 Lag-Lead compensator

The design satisfies specifications such as phase margin, error constant and bandwidth requirements

In general, there are two situations in which compensation is required In the first case, the system is absolutely unstable and the compensation is required to stabilize it as well as to achieve a specified performance In the second case, the system is stable but the compensation is required to obtain the desired performance If the system is type-1 (one pole at zero) or type-0 (no poles at zero), stable operation is always possible if the gain is sufficiently reduced and any of the three compensators, lag, lead and lag-lead may be used to obtain the desired performance. For type-2 systems or higher, lead compensation is required because only the lead compensator increases the margin of stability [17] Lag compensation also increases the margin of stability but at the expense of bandwidth Some systems cannot be stabilised using lead but most systems can be stabilised using lag

In the previous section the controller was designed based on the discrete model for each joint However bode plot design using the pulse transfer function is complicated In order to circumvent this difficulty and to use Bode design techniques, the following transformation is used [18]

$$\omega = \frac{2}{T}\frac{(z-1)}{(z+1)} \tag{4 16}$$

Procedure

$$G(s) \rightarrow \text{z o h} \rightarrow G(z) \rightarrow \underset{\text{transformation}}{\text{bilinear}} \rightarrow G(\omega)$$

For joint 1

$$G_1(s) = \frac{687\ 106}{s^3 + 333\ 4685s^2 + 11219\ 46s}$$

$$G_1(z) = \frac{9\ 774\text{x}10^{-6}(\ z^2 + 2\ 74394z + 0\ 4369\ )}{(\ z - 1\ )(\ z - 0\ 2282\ )(\ z - 0\ 827\ )}$$

$$G_1(\omega) = \frac{2\ 864\text{x}10^{-6}(\ \omega^2 - 344\ 645\omega - 511810\ 7\ )(\omega - 400\ )}{\omega\ (\ \omega + 251\ 356\ )(\ \omega + 37\ 854\ )}$$

Looking at the Bode plots of G(s) and G(ω), one can see that these plots are approximately the same Therefore using Bode design methods on G(ω), the different compensators can be designed

4.2.1 Lead Compensation

Phase Lead Compensation using Bode Plots proceeds by adjusting the system error constant to the desired value The phase margin (PM) of the uncompensated system is then checked If it is found unsatisfactory then the lead compensation technique is applied to meet the specified PM

In lead compensation the following are the effects of introducing this compensation technique

1 the crossover frequency is increased

2 the high frequency end of the Log v Mag. plot is raised up by a db gain of $20Log_{10}(1/a)$

The transfer function for a general phase lead network is given by

$$G_c(s) = \frac{1 + a\tau.s}{1 + \tau s} \qquad a > 1 \ , \ \tau > 0 \tag{4 17}$$

4.2.1.1 Phase Lead Design Procedure

The basic design procedure is as follows [17]

1 Determine the open-loop gain K to satisfy the specified error constant

2 Use this value of K, draw the bode plot of the uncompensated system and determine the phase margin of the uncompensated system

3 Determine the phase lead required using the relation

$$\Phi_L = \Phi_S - \Phi_1 + \epsilon \tag{4 18}$$

where,

Φ_S = specified phase margin

Φ_1 = phase margin of fixed part of the system

ϵ = safety margin

4 Let $\Phi_m = \Phi_L$

then

$$a = \frac{1 + \sin\Phi_m}{1 - \sin\Phi_m} \tag{4 19}$$

If $\Phi_m > 60^o$, then it is better to use two identical lead networks, each with 30^o phase margin, to achieve the required specifications

5 Calculate the gain $10Log_{10}a$ provided by the network at ω_m Next locate the frequency at which the uncompensated system has a gain of $-10Log_{10}a$ This is the new crossover frequency $\omega_c = \omega_m$

6 The upper corner frequency

$$\omega_{corner} = \frac{1}{\tau} = \frac{1}{\omega_c \surd a} \qquad ..(4\ 20)$$

4.2.1.2 Simulation Results

The design is performed for joint 1 to demonstrate this technique The results of the design are given for joints 2 and 3 Two different sets of specifications are used, resulting in two sets of different controllers

Specifications :

Joint 1

a $K_V = 50$, $PM = 45^0$

b $Kv = 25$, $PM = 60^0$

Joint 2

a $K_V = 30$, $PM = 45^0$

b $Kv = 15$, $PM = 60^0$

Joint 3

a $K_V = 60$, $PM = 45^0$

b $Kv = 20$, $PM = 60^0$

Design :

1 For a type-1 system, the steady state error for a unit ramp input is

$$e_{ss} = 1/K_V \tag{4 21}$$

Also

$$e_{ss} = \lim_{s \to 0} s \frac{1}{1 + K\,G(s)} \frac{1}{s^2}$$

$$= 16\ 32/K$$

$$\Rightarrow \quad K = 16\ 32 K_V \simeq 800$$

2 Using this value of K, the bode plot is drawn (see Fig 4 9a). The PM is 35^0 and ω_c (crossover frequency) = 35 3 rads/sec

3 $\Phi_L = 45^0 - 35^0 + 5^0 = 15^0$ where a 5^0 safety margin is included

4 Let $\Phi_m = \Phi_L$, then

$$a = \frac{1 + \sin\Phi_m}{1 - \sin\Phi_m} = 1\ 698$$

5 The gain at $\omega_m = 10\mathrm{Log}_{10}a = 2\ 3$db Thus the new crossover frequency ω_c, is the frequency where the gain is -2 3db From Fig 4 9a, $\omega_c = 42\ 62$ rads/sec

6 $\tau = \omega c \sqrt{a} = 0\ 018$

Thus the transfer function for the lead compensator is

$$K(\omega) = 800 \quad \frac{1 + 0.0305\omega}{1 + 0\ 018\omega}$$

Looking at Fig 4 9b the PM $\simeq 45^0$ for K = 800 Thus the specifications have be met.

To obtain the compensator pulse transfer function, substitute

$$\omega = \frac{2}{T} \quad \frac{(z-1)}{(z+1)}$$

$$\Rightarrow \quad K(z) = 1256 \frac{(z - 0.848)}{(z - 0\ 762)}$$

which yields the following controller difference equation for joint 1 using specification a

$$u_c(k) = 1256 \left[e(k) - 0\ 848e(k-1) \right] + 0\ 762u_c(k-1)$$

For joint 2

$$u_c(k) = 1147 \left[e(k) - 0\ 91e(k-1) \right] + 0\ 87u_c(k-1)$$

For joint 3

$$u_c(k) = 1085\ 27 \left[e(k) - 0\ 94e(k-1) \right] + 0\ 922u_c(k-1)$$

Fig 4 10a, Fig 4 10b and Fig 4 10c show the closed-loop response of the manipulator with the lead compensators above The response has good accuracy characteristics The steady state error for each joint is

$e_{ss_1} = 1\ 63\text{x}10^{-11}$
$e_{ss_2} = 1\ 107\text{x}10^{-3}$
$e_{ss_3} = 4\ 8\text{x}10^{-5}$

and the peak error is

$e_{pk_1} = 0\ 024$
$e_{pk_2} = 0\ 042$
$e_{pk_3} = 0\ 024$

Repeating the design procedure using specification b, the controller difference equation for joint 1 is

$$u_c(k) = 573\ 8\ [\ e(k) - 0\ 9e(k-1)\ ] + 0\ 86u_c(k-1)$$

For joint 2

$$u_c(k) = 542\ 6\ [\ e(k) - 0\ 943e(k-1)\ ] + 0\ 922u_c(k-1)$$

For joint 3

$$u_c(k) = 333\ 5\ [\ e(k) - 0\ 906e(k-1)\ ] + 0\ 822u_c(k-1)$$

Fig 4 11a, Fig 4 11b and Fig 4 11c show the closed-loop response of the manipulator with the lead compensators above The response is not as good as before The steady state error is larger for each joint

$e_{ss_1} = 2\ 393\text{x}10^{-11}$
$e_{ss_2} = 2\ 218\text{x}10^{-3}$
$e_{ss_3} = 5\ 01\text{x}10^{-5}$

and the peak error is also larger in magnitude

$e_{pk_1} = 0\ 06$
$e_{pk_2} = 0\ 1$
$e_{pk_3} = 0\ 03$

The controllers designed with the b specifications have lower values for velocity error constant K_V, and therefore have larger errors in the velocity profiles

4.2.2 Lag Compensation

A Phase-Lag network acts like a low-pass filter, attenuating high frequencies The phase lag normally occurs at the geometric mean of the corner frequencies It must be recognized that any phase-lag is undesirable at the crossover frequency of the compensated system Therefore, it is the attenuation characteristic of the network which is exploited for compensation purposes [17]

The transfer function for a general phase lag network is given by

$$G_C(s) = \frac{1 + a\tau.s}{1 + \tau s} \qquad a < 1 \ , \ \tau > 0 \tag{4 22}$$

4.2.2.1 Phase Lag Design Procedure

The basic design procedure is as follows [17]

1 Determine the open-loop gain necessary to satisfy the specified error constant
2 Find the frequency ω_{c2} at which the uncompensated system makes a phase margin contribution of

$$\Phi_2 = \Phi_S + \epsilon \tag{4 23}$$

where Φ_2 is measured above the -180° line Allow for ϵ = 5° to 15° for phase lag contribution by the network at ω_{c2}
3 Measure the gain of the uncompensated system at ω_{c2} and equate $20Log_{10}a$ to -gain at ω_{c2} Hence find a Now the magnitude at ω_{c2} = 0db
4 Place the upper corner frequency $1/a\tau$ one octave to one decade below ω_{c2},

$$\text{i e } \quad \omega_2 = \frac{1}{a\tau} = \frac{\omega_{c2}}{2} \text{ or } \frac{\omega_{c2}}{10} \tag{4 24}$$

5 Redraw the Bode Plot and check the specifications

4.2.2.2 Simulation Results

Again the design is performed for joint 1 to demonstrate this technique The results of the design are given for joints 2 and 3 Two different sets of specifications are used, resulting in two sets of different controllers

***Specifications* :**

Joint 1

a $K_V = 50$, PM = 45^0

b $K_V = 25$, PM = 60^0

Joint 2

a $K_V = 30$, PM = 45^0

b $K_V = 15$, PM = 60^0

Joint 3

a $K_V = 60$, PM = 45^0

b $K_V = 20$, PM = 60^0

***Design* :**

1 For the K_V value, K = 800

2 Using this value of K, the bode plot is drawn (see Fig 4 9a) The PM is 45^0 and ω_c (crossover frequency) = 27 47 rads/sec

3 Include a safety margin of 5^0, therefore a PM = 50^0 is required The new crossover frequency ω_{c2} = 23 8 rads/sec At this frequency the gain = 4 754db

$$\Rightarrow \quad 20\mathrm{Log}_{10}a = -4\ 754 \quad \text{and } a = 0\ 5785$$

4 $a\tau = \omega_c/10$ and $\tau = 0\ 726$

Thus the transfer function for the lead compensator is

$$K(\omega) = 800 \quad \frac{1 + 0.42\omega}{1 + 0\ 726\omega}$$

Looking at Fig 4 12 the PM $\simeq 45^0$ for K = 800 Thus the specifications have be met.

To obtain the compensator pulse transfer function, substitute

$$\omega = \frac{2}{T}\,\frac{(z-1)}{(z+1)}$$

$$\Rightarrow \quad K(z) = 464\,\frac{(z - 0.98)}{(z - 0\ 99)}$$

which yields the following controller difference equation for joint 1 using specification a

$$u_c(k) = 464\ [\ e(k) - 0\,98e(k-1)\ ] + 0\,99u_c(k-1)$$

For joint 2

$$u_c(k) = 620\,5\ [\ e(k) - 0\,991e(k-1)\ ] + 0\,993u_c(k-1)$$

For joint 3

$$u_c(k) = 394\ [\ e(k) - 0\,986e(k-1)\ ] + 0\,993u_c(k-1)$$

Fig 4 13a, Fig 4 13b and Fig 4 13c show the closed-loop response of the manipulator with the lag compensators above The response is not as good as the lead controller designed with the same specifications The steady state error is

$$e_{ss_1} = 3\,8\times10^{-7}$$
$$e_{ss_2} = 1\,08\times10^{-3}$$
$$e_{ss_3} = 5\,3\times10^{-5}$$

and the peak error is

$$e_{pk_1} = 0\,03$$
$$e_{pk_2} = 0\,06$$
$$e_{pk_3} = 0\,03$$

Repeating the design procedure using specification b, the controller difference equation for joint 1 is

$$u_c(k) = 312\,9\ [\ e(k) - 0\,991e(k-1)\ ] + 0\,993u_c(k-1)$$

For joint 2

$$u_c(k) = 324\,5\ [\ e(k) - 0\,994e(k-1)\ ] + 0\,995u_c(k-1)$$

For joint 3

$$u_c(k) = 208 \; [\; e(k) - 0\,992e(k-1) \;] + 0\,993u_c(k-1)$$

Fig 4 14a, Fig 4 14b and Fig 4 14c show the closed-loop response of the manipulator with the lag compensators above The response is not as good as the previous lag compensator The steady state error is larger for each joint

$$e_{ss1} = 2\,33x10^{-5}$$
$$e_{ss2} = 2\,04x10^{-3}$$
$$e_{ss3} = 2\,334x10^{-4}$$

and the peak error is also larger in magnitude

$$e_{pk1} = 0\,06$$
$$e_{pk2} = 0\,12$$
$$e_{pk3} = 0\,12$$

The controllers designed with the b specifications have lower values for velocity error constant K_V, and therefore have larger errors in the velocity profiles

4.2.3 The Lag-Lead Compensator

For large specified error constant and moderately large bandwidth, it may not be possible to meet the specifications through either lead or lag compensation In such situations lag-lead compensation is employed where the lag section supplies part of the phase margin specification and the lead section supplies the rest of the phase margin and the desired bandwidth [17]

The transfer function for a general phase lag network is given by

$$G_C(s) = \underbrace{\frac{(1 + a\tau_1 s)}{(1 + \tau_1 s)}}_{\textbf{Lead}} \; \underbrace{\frac{(1 + b\tau_2 s)}{(1 + \tau_2 s)}}_{\textbf{Lag}} \qquad a > 1 \quad b < 1 \quad \tau_1, \tau_2 > 0 \quad ab = 1 \qquad (4\ 25)$$

4.2.3.1 Phase Lag-Lead Design Procedure

The basic design procedure is as follows [17]

1 Check the phase margin and bandwidth of the uncompensated system with the specifications If bw < specified value try lead compensation, but if bw > specified value try lag compensation provided the uncompensated system is not absolutely unstable

2 If lag compensator design results in too low a bandwidth, then a lag-lead network is required in order to have a faster time response A lag-lead compensator is essentially a band-pass filter

3 Design the lag section to provide some of the phase margin requirements in the usual fashion

4 Once the lag compensator has been designed τ_2 and b are assigned values

5 Because ab = 1, the value for a is already calculated Hence τ_1 is the only parameter to be chosen in the lead section design

4.2.3.2 Simulation Results

Again the design is performed for joint 1 to demonstrate this technique The results of the design are given for joints 2 and 3

Specifications :

<u>Joint 1</u>
$K_V = 50$, PM = $45^0 \rightarrow 60^0$

<u>Joint 2</u>
$K_V = 30$, PM = $45^0 \rightarrow 60^0$

<u>Joint 3</u>
$K_V = 60$, PM = 45^0

Design :

a. Lag

1. For the K_V value, K = 800

2 Using this value of K, the bode plot is drawn (see Fig 4 9a) The PM is 35^0 and thus $10^0 \rightarrow 25^0$ additional phase required

3 With the lag network, 10^0 phase (15^0 for safety) is attained This occurs at ω_{c_1} = 23 56 rads/sec The gain at ω_{c_1} = 4 9db Thus b = 0 568 and τ_2 = 0 747 The transfer function for the lag section is

$$G_{lag}(\omega) = \frac{1 + 0.424\omega}{1 + 0\ 747\omega}$$

b. Lead

1 The value for a is fixed from the lag section and is given by a = 1/b = 1 76

2 The maximun lead provided by the lead section is

$$\Phi_m = \sin^{-1}\left[\frac{a - 1}{a + 1}\right]$$

$$= 16^0$$

3 Gain of phase lead at ω_{c_2} (the eventual crossover frequency) = $-10\mathrm{Log}_{10}a$ = -2 455db

$\Rightarrow \omega_{c_2}$ = 29 3 rads/sec

and $\tau_1 = \omega_{c_2}/\sqrt{a}$ = 0 0257 The transfer function for the lead compensator is

$$G_{lead}(\omega) = \frac{1 + 0.0452\omega}{1 + 0\ 0257\omega}$$

and the total lag-lead controller is

$$G_{lag\text{-}lead}(\omega) = 800\ \frac{(1 + 0.424\omega)}{(1 + 0\ 747\omega)}\ \frac{(1 + 0.0452\omega)}{(1 + 0\ 0257\omega)}$$

Fig 4 15 shows the bode plot of the compensated system The PM = 57^0 and ω_c = 28 9 rads/sec Therefore the specifications have been fulfilled

To obtain the compensator pulse transfer function, substitute

$$\omega = \frac{2}{T}\ \frac{(z-1)}{(z+1)}$$

$$\Rightarrow \quad G_{lag\text{-}lead}(z) = 770\ \frac{(z - 0.988)}{(z - 0\ 993)}\ \frac{(z - 0.895)}{(z - 0\ 822)}$$

This results in the following controller equations for joint 1

$$u_C(k) = 770\ [\ e(k) - 1\ 883e(k-1) + 0\ 884e(k-2)\ ] + 1\ 815u_C(k-1) - 0\ 816u_C(k-2)$$

For Joint 2

$$u_C(k) = 786\ 3\ [\ e(k) - 1\ 992e(k-1) + 0\ 922e(k-2)\ ] + 1\ 885u_C(k-1) - 0\ 885u_C(k-2)$$

For Joint 3

$$u_C(k) = 780\ 5\ [\ e(k) - 1\ 816e(k-1) + 0\ 82e(k-2)\ ] + 1\ 773u_C(k-1) - 0\ 776u_C(k-2)$$

Fig 4 16a, Fig 4 16b and Fig 4 16c show the manipulator model closed-loop response over a specified trajectory using the lag-lead compensator The result is encouraging The static accuracy is

$$e_{ss_1} = 1\ 164\text{x}10^{-8}$$
$$e_{ss_2} = 1\ 64\text{x}10^{-3}$$
$$e_{ss_3} = 5\ 09\text{x}10^{-5}$$

and the peak error

$$e_{pk_1} = 0\ 035$$
$$e_{pk_2} = 0\ 075$$
$$e_{pk_3} = 0\ 03$$

The specification for joint 3 is different to joint 1 and joint 2 It was found that too much lag causes overshoot and oscillation in the response of joint 3

4.2.4 Conclusion on Lag-Lead Performance

Lead compensation results in an increased bandwidth and faster speed of response For high order systems and systems with large error constants, large leads are required for compensation, resulting in excessively large bandwidth, which is undesirable from a noise transmission point of view For such a system, lag compensation is preferred, provided the uncompensated system is not absolutely unstable

Lag compensation results in a reduction of the crossover frequency Thus the lag compensator reduces the system bandwidth (crossover being a rough measure of bandwidth) and the additional attenuation of high frequencies improves the s/n (signal to noise) ratio A drawback of reduced bandwidth is that the rise time t_r is increased, since $t_r \propto 1/bw$

To overcome the problems of Lead-only compensation and Lag-only compensation, Lag-Lead compensation is employed for high order systems and for systems with large error constants Since a full lag compensator will reduce the bandwidth excessively, the lag-section of the lag-lead compensator must be designed so as to provide partial compensation only There is only one variable parameter for the lead-section after the lag-section is designed

These frequency domain compensators are similar in form to PID configurations PD control is similar to Lead compensation, PI control and Lag compensation are similar and finally, full PID control is similar to Lag–Lead compensation. Comparing the results of Lead, Lag and Lag-Lead, a Lead controller gives marginally the best response of the three Lag compensation is not desirable for manipulator use, because of the increased rise time effect.

4.3 Optimal Control

Optimal Control is well suited to the tracking or regulator control problem, since optimal control can increase the speed of systems while also reducing oscillatory behaviour

To design an optimal controller for some process, a scalar valued cost function J(u,e,...), that realistically quantifies all the process factors of importance, is formed Once one knows the required information about the plant, state equations or transfer function and the cost function J, it is the role of optimal control to determine a control sequence which will achieve the control objectives and simultaneously minimize the cost function [19]

Optimal performance is defined with respect to some specification. The quality or goodness of a system is represented by selectmg a suitable cost function An optimal controller is then obtained by minimizing the selected performance index The most frequently employed cost functions are based on error or functions of error, or control energy or functions of control energy

4.3.1 Properties of Optimal Control

1 The optimal control problem is solved for a particular plant, producing a dedicated controller
2 The solution to the optimal control problem is designed with respect to a specific input.
3 It greatly improves the time response
4 Controllers are the same order as the plant.
5 Stability is guaranteed

The design of optimal controller can be based on

1 Frequency domain analysis,
2 State Space (time domain) solution, or
3 The Transfer Function approach

When designing a controller certain specifications must be met. The quadratic cost function J is of the following form

$$\int_0^{\infty} [\ e^2(t)\ q + u^2(t)\ r\ ]\ dt \qquad (4\ 26)$$

where q and r are positive constants called weighting factors If q is large and r is small, more weight is imposed on the error, hence the controller is designed for the tracking problem If however r is much larger than q, then the controller is designed for power conservation [19]

4.3.2 Application to Robotics

Little work has been done in the area of optimal controllers for robotic manipulators, due mainly to extremely heavy real-time computational load incurred when attempting to determine control parameters Also mechanical constraints place severe physical limitations on manipulator speed [20]

Some interest has centered on the time optimal control of certain robots that do not need to perform coordinated motion In this instance the individual joints are moved sequentially in time, and what is sought is the sequence of separate joint motions so that the overall motion is minimized

As the computational power of robot controllers increases, optimal control will become more feasible However it will also be necessary for novel mechanical structures and materials to be developed that permit high-performance manipulators to withstand the extreme stress encountered while optimal control is being executed

4.3.3 Controller Derivation

The objective here is to derive an expression for a Z domain optimal controller The design of digital time controllers is very similar to the continuous control design technique The design is initiated in the continuous domain and then transformed to the discrete domain The solution to optimal open loop control is found, and then the optimal output feedback solution is obtained [19]

The performance criterion to be minimized is defined as follows

$$\int_0^{\infty} [\ e^2(t)\ q + u^2(t)\ r\ ]\ dt$$

Q is the error weighting matrix,
R is the control weighting matrix,
Output $Y(s) = W(s)\,U(s)$,
Error $E(s) = r(s) - Y(s)$,
Control input $U(s) = C_0(s)\,E(s)$,
Plant Transfer Function = $W(s)$,

The gradient function is defined as

$$g = \tfrac{1}{2}\,\frac{dJ}{dU} \qquad (4\ 27)$$

Replacing the error e, in the cost function by r - W U, where W(s) is the system transfer function This yields

$$g = \{\ W^*\ Q\ W + R\ \}\ U - W^*\ Q\ r \qquad (4\ 28)$$

and for optimality in the s domain, the transformed optimal gradient function g(s) is analytic in the closed left half plane, i e g(t) = 0

The transformed gradient may be obtained as

$$g(s) = \{ W^T(-s)\ Q\ W(-s) + R \}\ U(s) - W^T(-s)\ Q\ r(s) \tag{4 29}$$

To manipulate the frequency domain gradient optimality condition, the operator decomposition known as *spectral factorisation* is required The matrix $\{W^T(-s)\,Q\,W(s) + R\}$ can be spectrally factored as

$$Y^T(-s)\ Y(s) = W^T(-s)\ Q\ W(s) + R \tag{4 30}$$

and also,

$$\begin{aligned} &Y^T(-s)^{-1}\ g(s) + \{ Y^T(-s)^{-1}\ W^T(-s)\ Q\ r(s) \}_- \\ &\qquad = Y(s)\ u(s) - \{ Y^T(-s)^{-1}\ W^T(-s)\ Q\ r(s) \}_+ \end{aligned} \tag{4 31}$$

where $\{\ \}_+$ implies the enclosed function is analytic in the closed right half plane, and $\{\ \}_-$ implies an analytic function in the left half plane

Transforming to the discrete domain equations (4 30) and (4 31) become respectively

$$U(z) = Y^{-1}(z)\ \{ Y^T(z^{-1})^{-1}\ W^T(z^{-1})\ z^{k_0}\ Q\ r(z) \} \tag{4 32}$$

$$Y^T(z^{-1})\ Y(z) = W^T(z^{-1})\ Q\ W(z) + R \tag{4 33}$$

Define P(z), where

$$P(z)\ r(z) = \{ Y^T\ (z^{-1})^{-1}\ W^T\ (z^{-1})\ z^{ko}\ Q\ r(z) \}_+ \tag{4 34}$$

$$P(z)\ r(z) = Y(z)\ U(z) \tag{4 35}$$

and $Y^{-1}(z).P(z)$ is the optimal open-loop controller matrix The optimal closed-loop matrix is found to be

$$K(z) = Y^{-1}(z)\ P(z)\ [\ I - z^{k_0}\ W(z)\ Y^{-1}(z)\ P(z)\ ]^{-1} \tag{4 36}$$

Let the plant transfer function

$$W(z) = \delta(z)/\sigma(z), \tag{4.37}$$

then,

$$Y(z^{-1})\, Y(z) = W^T(z^{-1})\, Q\, W(z) + R$$

$$= \frac{d(z^{-1}).d(z)}{\sigma(z^{-1})\ \sigma(z)} \tag{4 38}$$

Optimisation is done with respect to a step input

$$\Rightarrow \quad r(z) = \frac{z}{z-1}$$

Let $F_0(z)$ (the optimal open loop controller) be

$$F_0(z) = Y^{-1}(z)\ P(z) \tag{4 39}$$

this gives

$$F_0(z)\ \frac{z}{z-1} = Y(z)^{-1} \left\{ \frac{\sigma(z^{-1}).\delta(z^{-1})}{d(z^{-1})\ \sigma(z^{-1})}\ z^{k_0}\ Q\ \frac{z}{z-1} \right\}_+ \tag{4 40}$$

The only part which is analytic is the DC part (z=1)

$$F_0(z) = Y(z)^{-1}\ \frac{\delta(1)}{d(1)}\ \{z^{k_0}\}_+\ Q \tag{4 41}$$

The required closed-loop controller is given from (4 36) as

$$K(z) = \{ F_0(z)^{-1} - z^{-k_0}\, W(z)\}^{-1} \tag{4 42}$$

$$K(z) = \sigma(z) \left\{ \frac{d(z).d(1)}{\delta(1)}\ Q^{-1} - z^{-k_0}\ \delta(z) \right\}^{-1} \tag{4 43}$$

For the scalar case

$$d(z^{-1})\ d(z) = \delta(z^{-1})\ \delta(z)\ q + \sigma(z^{-1})\ \sigma(z)\ r \tag{4 44}$$

$$d(1) = \{ \delta(1)^2.q + \sigma(1)^2\ r \}^{\frac{1}{2}} \tag{4 45}$$

The final controller equation is as follows

$$K(z) = \frac{\sigma(z).\delta(1).q}{d(z)\ d(1)\ -\ z^{-k_0}\ \delta(z)\ \delta(1)\ q} \qquad (4\ 46)$$

4.3.4 Simulation Results

Refer to Chapter 2 for the linear decoupled models for the three primary joints The following controllers are designed for joint 1 using the q and r values shown

1 q = 100 , r = 0 0001

$$K_{11}(z) = 445\ 86\ \frac{z^3 - 2.0552z^2 + 1.2439z - 0.1887}{z^3 - 1\ 4589z^2 + 0\ 5395z - 0\ 0702}$$

2 q = 100 , r = 0 001

$$K_{12}(z) = 273\ 645\ \frac{z^3 - 2.0552z^2 + 1.2439z - 0.1887}{z^3 - 1\ 8073z^2 + 0\ 9497z - 0\ 1385}$$

3 q = 1000 , r = 0 001

$$K_{13}(z) = 692\ 493\ \frac{z^3 - 2.0552z^2 + 1.2439z - 0.1887}{z^3 - 1\ 5945z^2 + 0\ 6964z - 0\ 0102}$$

Tests are carried out on the robot simulator using these controllers to find which give the best results When the best controller is found, the same q and r values are used to design joint 2 and 3 controllers The value of r is chosen in proportion to the sampling interval, otherwise the closed-loop pole polynomial is very similar to the open-loop equation

The velocities of joints 2 and 3 are set to sero to keep the joints locked at position zero The controllers $K_{11}(z)$, $K_{12}(z)$ and $K_{13}(z)$ are placed on the manipulator model Supplying a constant setpoint to the control loop, the controllers are evaluated on their performance, using response time, overshoot and steady state error as performance criteria The control inputs are bounded between ±40 This restricts the q/r value.

Using $K_{11}(z)$, (see Fig.4.17a and Fig 4 17b) the response is slow and a large steady state error exists Using $K_{12}(z)$, (see Fig 4 18a and Fig 4 18b) the response has improved only slightly from before However $K_{13}(z)$ (q=1000, r=0 001), (see Fig 4 19a and Fig 4 19b) gives the fastest response with the smallest steady state error Hence this controller is chosen to be the most suitable controller for joint 1, giving the best responses with allowable control inputs

Using these results, q = 1,000 and r = 0 001 are the chosen parameter values

$$K_{21}(z) = 836\ 32\ \frac{z^3 - 2.1121z^2 + 1.300z - 0.1887}{z^3 - 1\ 8027z^2 + 0\ 936z - 0\ 0702}$$

$$K_{31}(z) = 696\ 65\ \frac{z^3 - 2.0362z^2 + 1.225z - 0.1886}{z^3 - 1\ 5837z^2 + 0\ 686z - 0\ 1022}$$

However $K_{21}(z)$ does not give suitable results Its gain is reduced to 695 (the proportional gain used for the other joints) to prevent oscillatory behaviour

Now these *best controllers* are applied to track an input trajectory All three joints are moved through a considerable portion of their range Fig 4 20a, Fig 4 20b and Fig 4 20c show the control voltage inputs, joint positions and tracking error respectively Investigation shows the peak error to be as follows

joint 1 = 0 2 rads
joint 2 = 0 12 rads
joint 3 = 0 1 rads

Therefore one can conclude that optimum control does not perform as well as PID or even PD control techniques

4.3.5 Conclusion on Optimal Control

An optimal control system is a system whose design optimizes (minimizes or maximizes) the value of a function chosen as the performance index It differs from the ideal case in that the former is the best attainable in the presence of physical constraints whereas the latter may well be an unattainable goal It is desirable that the criteria for optimal performance originate not from a mathematical but from an application point of view In general, however, the choice of a performance index involves a compromise between a meaningful evaluation of system performance and a tractable mathematical problem [21]

The solution of an optimal control problem is to determine the optimal control sequence u(k) within the class of allowable control inputs This input u(k) depends on

1 Nature of the performance index,
2 Nature of the constraints,
3 Initial state or initial output,
4 Desired state or desired output

In the design method used here, the weighting function

$$\int_0^\infty [\ e^2(t)\ q + u^2(t)\ r\ ]\ dt$$

is minimised The constants q and r are chosen depending on which type of control is required The resulting controller minimizes this function

The robot is a very complex model, highly coupled and nonlinear The optimal controllers designed above are based on three linear decoupled models for the three primary joints This means a substantial approximation is made before the optimal solution is applied, and this in fact defeats the point of finding the optimal solution

Comparing the optimal control approach to other control methods, its tracking performance is poor compared to PID control A more complicated approach is needed here, nonlinear optimal control is required which is more complex but should improve the closed-loop performance substantially to justify its use

4.4 Predictive Control Methods

The concept of predictive control was introduced by Richalet [23] in the late seventies Predictive controllers are based on a *prediction* of the *future* behaviour of the process to be controlled These predictions are based on a model of the process that is assumed to be available For this reason predictive controllers are sometimes denoted internal model controllers Not only simple processes (e g first or second order without time delay) but also difficult processes (e g processes with a long time delay, non-minimum phase and unstable processes) can be controlled by predictive controllers without the designer having to take much special precautions Moreover, in contrast with other control methods, predictive controllers have shown themselves to be remarkably robust with respect to model mismatch Further, it is claimed [23] that predictive controllers are easy to tune, even by people who are not control engineers

The predictive control concept is not restricted to linear single-input, single-output (SISO) processes, but can also be applied to linear multi-input, multi-output (MIMO) processes and to nonlinear SISO processes

Various algorithms exist at the present moment but four basic principles are fundamental to the control concept in each

1 The Internal Model
2 The Reference Trajectory
3 Algorithmic Control
4 The Self Compensator

The reference trajectory is the method used to connect the actual process state to the desired dynamic setpoint. A Reference Trajectory is initiated from the process output that will tend towards the setpoint C_p according to a desired dynamic path, over a prediction horizon The nature of the reference trajectory is open, but usually chosen as

$$S_r(i) = \alpha^i \; S_0(k) + (1-\alpha^i) \; C_p \qquad i = 1,2, \quad H \tag{4 47}$$

a first order curve with decaying error between itself and the set-point

S_r = reference trajectory output,
S_0 = measured control variable,
S = model output,
α and H are the tuning parameters (H is the prediction horizon)

The match between the Reference Trajectory and the predicted process output is to be looked for, mainly for controllability reasons, on a particular future horizon called the Coincidence Horizon

Any mismatch between plant and internal model will result in an error or offset from the setpoint. A compensation technique compensates for mismatch and corrective action is taken Also a disturbance may be present at the output and the compensation technique allows the process output to return to the setpoint. In any real life situation an exact model of a plant is not practical and there will always be some mismatch present The purpose of the control action is to keep the output at a set value, and so to nullify the effect of the mismatch or disturbance The speed of the error compensation depends on the tuning parameters [25].

4.4.1 Full State Feedback (Adapted Monoreg Algorithm)

The Monoreg algorithm uses a convolution internal model but is adapted here to a State-Space representation. This algorithm is derived in the same fashion but with a State-Space Model. Certain assumptions are made during the derivation of this algorithm which inhibit the performance slightly. The results section determines the suitability of this algorithm for manipulator control applications.

4.4.1.1 Algorithmic Derivation

The Monoreg control algorithm [24] is obtained by expressing the coincidence, at the end of the prediction horizon, between the desired increment of the system output through the reference trajectory and that of the model output, i.e.

$$S_r(H) - S_0(k) = S(k+H) - S(k) \tag{4.48}$$

where,

S_0 = measured control variable,
S_r = reference trajectory output,
S = model output,
H = prediction horizon,
k = present sampling instant.

The general State-Space description of a system can be written as follows

$$\begin{aligned} x(k+1) &= A\ x(k) + B\ u(k) \\ y(k) &= C\ x(k) \end{aligned} \tag{4.49}$$

The general solution is

$$y(k) = C\left[\, A^k\ x(0) + \sum_{i=0}^{k-1} A^{k-i+1}\ B\ u(i) \,\right] \tag{4.50}$$

Consider the following arbitrary state trajectory as k increases (see Fig 4 21)

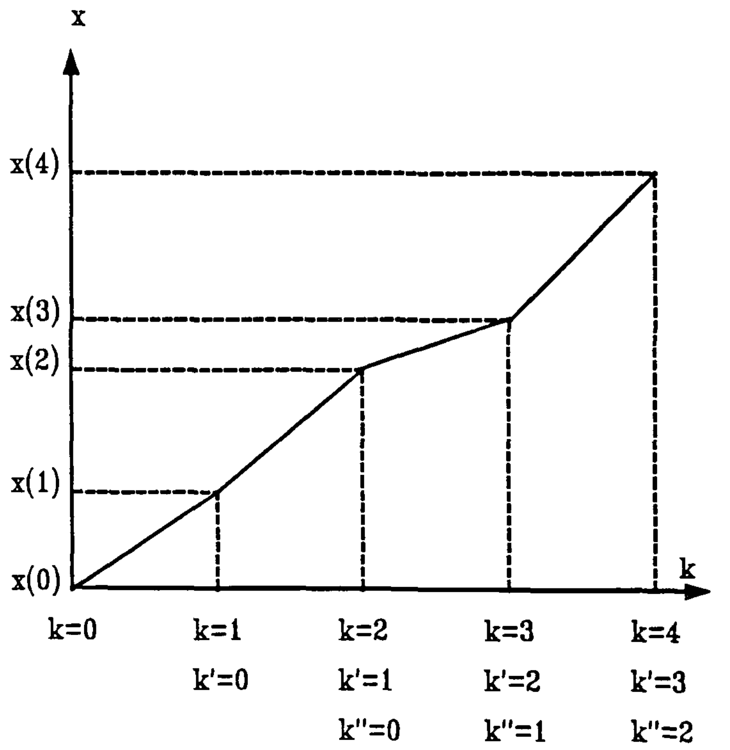


Fig 4.21 An Arbitrary State Trajectory

Now

$$
\begin{aligned}
y(1) &= C\ \{ A^1\ x(0) + A^0\ B\ u(0) \} \\
&= C\ \{ A\ x(0) + B\ u(0) \}
\end{aligned}
\tag{4 51}
$$

Defining a new initial condition

$$x'(0) = x(1) \text{ with } k' = 0$$

$$y(k) = C\ [\ A^{k'}\ x'(0) + \sum_{i=0}^{k'-1} A^{k'-1+i}\ B\ u(i)\ ] \tag{4 52}$$

$$y(2) = C\ \{ A\ x(1) + B\ u(1) \} \quad \text{with } k = 2,\ k' = 1$$

$$x'(0) = x(1)$$

The general expression for y(k) with all the states being measurable is

$$y(k) = C \{ A\, x(k-1) + B\, u(k-1) \} \qquad (4\ 53)$$

At time t+H (end of the prediction horizon)

$$y(k+H) = C \left[A^{k+H} x(0) + \sum_{i=0}^{k+H-1} A^{k+H-i+i} B\, u(i) \right] \qquad (4\ 54)$$

e g k=1

$$y(H+1) = C \left[A^{H+i} x(0) + \sum_{i=0}^{H+1-1} A^{H+i-i+i} B\, u(i) \right] \qquad (4\ 55)$$

$$\Rightarrow \quad y(H+1) = C \left[A^{H} x(1) + \sum_{i=0}^{H-1} A^{H-i+i} B\, u(i+1) \right] \qquad (4\ 56)$$

$$\Rightarrow \quad y(H+2) = C \left[A^{H} x(2) + \sum_{i=0}^{H-1} A^{H-i+i} B\, u(i+2) \right] \qquad (4\ 57)$$

In general

$$y(H+k) = C \left[A^{H} x(k) + \sum_{i=0}^{H-1} A^{H-i+i} B\, u(i+k) \right] \qquad (4\ 58)$$

Looking at the summation terms

$$\sum_{i=0}^{H-1} A^{H-i+i} B\, u(i+k) = A^{H-i} B\, u(k) + A^{H-2} B\, u(k+1) + \quad + B\, u(k+H-1)$$

Our control strategy is to assume that the mv remains constant over the prediction horizon i e

$$u(k) = u(k+1) = u(k+2) = \quad = u(k+H-1) \qquad (4\ 59)$$

This later leads to restrictions on the performance of the algorithm with respect to disturbances on the output

$$\sum_{i=0}^{H-1} A^{H-i+i} B\, u(i+k) = \{ B + AB + \quad + A^{H-1} B \}\, u(k)$$

Equation (4 48) can be written in state-space terminology

$$S_r(H) - S_0(k) = Y(k+H) - Y(k)$$

$$S_r(H) - S_0(k) = (1-\alpha^H)\{ C_p - S_0(k)\}$$

$$y(k+H) - y(k) = C\{ A^H x(k) \} + C\{ B + A B + \quad + A^{H-1} B \}\, u(k) - C\, x(k)$$

(4 60)

Let $P = C\{ B + A B + \quad + A^{H-1} B \}$
= scalar for the SISO case (4 61)

$$\Rightarrow \quad (1-\alpha^H)\,(C_p - S_0(k)) = C\,(A^H x(k)) + P\, u(k) - C\, x(k) \tag{4 62}$$

Thus the manipulated variable is calculated by

$$u(k) = \frac{(1-\alpha^H)\,(C_p - S_0(k)) - C\,(A^H - I)\, x(k)}{P} \tag{4 63}$$

Two tuning parameters must be chosen, α and H Tuning is done in the **Time Domain.**

$$\alpha^* = \exp(-T/\tau) \qquad 0 < \alpha < 1$$

where τ is the system time constant and T is the sampling interval For a fast response with high initial control inputs use

$$\alpha < \alpha^*$$

But for a slow response with low initial control inputs choose

$$\alpha > \alpha^*$$

The equation for u(k) is quite simple, since $C(A^H - I)$ and P are constants and can be evaluated off-line a priori

4.4 1.2 Properties

1 The assumption that the setpoint is constant over the prediction horizon restricts its application to regulatory control

2 The robustness follows from trying to drive the output to the setpoint at the end of the prediction horizon and not at the next sampling instant.

3 The Principal of Receding Horizon is used At the sampling instant k a reference trajectory is initialized from the process output to the setpoint at the end of the prediction horizon Using this reference trajectory the controller output is calculated But at the next sampling instant k+1 the whole procedure is repeated, with a new reference trajectory being initialized The prediction horizon is continually receding into the future [24]

4.4 1.3 Simulation Results

To demonstrate the effect α has on the closed-loop response, different values are used keeping H constant The test is done with constant setpoints With α_i = 0 7 convergence takes approximately 1 2 seconds (see Fig 4 22a and Fig 4 22b) but with α_i = 0 3 the response is faster and higher initial are applied to the process (see Fig 4 23a and Fig 4 23b) The optimal tuning parameters are chosen from these tests to be α_i = 0 7 H_1 = 10

Using these values for the tuning parameters, each of the primary joints is controlled over a specified trajectory (see Fig 4 24a, Fig 4 24b and Fig 4 24c). The controller performs with good accuracy, giving a peak error for each joint of

j_1 = 0 03 rads

j_2 = 0 03 rads

j_3 = 0 05 rads

4.4.2 Output Feedback Control

This method incorporates predictive control and a mathematical technique called System Inversion The two methods are used together because it is possible to generate exact inverse models for nonlinear systems Hence the control of nonlinear systems is possible whenever the inverse model can be generated uniquely [22]. Using this control technique, no local or global linearisation transformation is necessary for nonlinear control The internal on-line model of the plant in this technique is an inverse model, generated quite easily from the approximate linear models for each of the robots primary joints

4.4.2.1 Algorithmic Derivation

The control equation is calculated for a general third order model Consider the following transfer function [22]

$$G(z) = \frac{a_1 z^2 + a_2 z + a_3}{z^3 + b_1 z^2 + b_2 z + b_3}$$

$$G(z) = \frac{Y(z)}{U(z)}$$

Calculate the inverse model by cross multiplying and taking the inverse Z transform, i e converting the transfer function to a difference equation. For the general case above

$$a_1 u(k+2) + a_2 u(k+1) + a_3 u(k) = y(k+3) + b_1 y(k+2) + b_2 y(k+1) + b_3 y(k) \tag{4 64}$$

Isolate the u(k+2) term

$$\Rightarrow \quad u(k+2) = \{ - a_2 u(k+1) - a_3 u(k) + y(k+3) + b_1 y(k+2) + b_2 y(k+1) + b_3 y(k)\}/a_1 \tag{4 65}$$

If the following assumption is made

$$u(k+2) = u(k+1) = u(k) \tag{4 66}$$

Then equation becomes

$$\Rightarrow \quad u(k) = - \{ y(k+3) + b_1 y(k+2) + b_2 y(k+1) + b_3 y(k)\}/(a_1+a_2+a_3) \tag{4 67}$$

If u(k-1) = u(k) then .

$$\Rightarrow \quad u(k) = \{ y(k+4) + b_1 y(k+3) + b_2 y(k+2) + b_3 y(k+1)\}/(a_1+a_2+a_3) \tag{4 68}$$

so there is no dependence of u(k) on y(k), which means that the algorithm u(k) is derived only from points on the reference trajectory Also since four inputs (past and present) have been equated together then H = 4

The variable y(k) is the joint position at time k A reference trajectory based on this value of y(k) is initiated to generate the outputs necessary to calculate the control input. The reference trajectory takes the form of a first order curve

$$y_r(k+i) = \alpha^i\ y(k) + (1-\alpha^i)\ C_p \text{ where } C_p = \text{setpoint} \qquad (4\ 69)$$

The control algorithm presented above is open-loop and no compensation takes place in the presence of model mismatch or a disturbance on the output. Two types of compensation techniques are possible The first type assumes that the error over the prediction horizon is constant, and the other type tries to fit a first order polynomial to the future error based on past measurements, using the method of Least Squares The predicted error is then added to each point on the reference trajectory which adjusts the control input to compensate for mismatch or disturbances It is better to use the second method of compensation because it can overcome severe mismatch, due to the structured form of the future error

4.4.2.2 Properties

1 The assumption that the setpoint is constant over the prediction horizon restricts its application to regulatory control but the error compensation technique helps reduce the error when tracking varying setpoints

2 The robustness follows from trying to drive the output to the setpoint at the end of the prediction horizon and not at the next sampling instant.

3 The Principal of Receding Horizon applies also

4 This algorithm is not computationally complex The internal model used is an *inverse model*, a simple linear equation The most computationally complex part of the algorithm is in computing the coefficients of the first order error polynomial

4.4.2.3 Simulation Results

Based on the simplified model for the PUMA 560 the following inverse models result for joint one, two and three respectively

$$u_1(k) = 24471\ 69\{\ y_1(k+4) - 2\ 0552y_1(k+3) + 1\ 2439y_1(k+2) - 0\ 1887y_1(k+1)\ \}$$

$$u_2(k) = 73701\ 39\{\ y_2(k+4) - 2\ 1121 y_2(k+3) + 1\ 3003 y_2(k+2) - 0\ 1887 y_2(k+1)\ \}$$

$$u_3(k) = 18428\ 47\{\ y_3(k+4) - 2.0352 y_3(k+3) + 1\ 2234 y_3(k+2) - 0\ 1884 y_3(k+1)\ \}$$

To find the optimal parameters, firstly multi-joint control with constant setpoints is tried For the models derived above the prediction horizon has a value of four This can be extended if one so desires The value of α has to be chosen to provide a sufficiently fast response without having too severe control inputs

With α_1 equal to 0 95, the response is fast, there is little overshoot and the static error is low (see Fig 4 25a and Fig 4 25b) Reducing α_1 to 0 85, undesirable results are achieved, i e a large static error is present (see Fig 4 26a and Fig 4 26b) The value of α should be close to unity since the sampling frequency is 200Hz. Trying to drive the close-loop system too fast, results m unsatisfactory results since the inverse models are only linear approximations to the actual system. The *best* tuning parameters are chosen to be $\alpha_1 = 0\ 95$, $H = 4$ Tracking a path using these parameters results in a large peak error (see Fig 4 27a, Fig 4 27b and Fig 4 27c)

j1 = 0 2 rads
j2 = 0 3 rads
j3 = 0 25 rads

4.4.3 Conclusion

The revised Monoreg Predictive Control Algorithm performs well in the simulation experiments Although, this algorithm is only for regulatory control and one of its assumptions that the control input remains constant over the prediction horizon, it still performs well when asked to track a specified path This is due to the fact that the trajectory is sampled at intervals of 5msecs, and in that time the setpoint does not change very much

This algorithm performs better than PD or Optimal Control Computationally it requires some off-line calculation before the algorithm is initiated The on-line computation is not very demanding on processor time, therefore this algorithm is suitable for manipulator control

The second algorithm is not computationally complex and it is easy to derive the control law provided a model of the system exists However the performance of this algorithm on the PUMA 560 model is poor, with a large tracking error and poor static accuracy compared to the previous algorithms This algorithm is not suitable for high precision manipulator tasks

4.5 Summary

Several fixed parameter control algorithms are presented in this chapter These algorithms range from the classical controllers, like PID and Optimal Control, to the modern control technique of Predictive Control Frequency Domain Compensators are also discussed The performance of each of these algorithms is investigated on the robot simulator to determine the most suitable controller for manipulator-type robots Several criteria are used to pick the *best* algorithm and this is discussed in a later chapter

The tuning of such algorithms is as diverse as the algorithms themselves PID can be tuned using the Zeiger-Nicholas rules or using a Pole-Placement scheme Lag-Lead configurations are tuned using the Bode design technique. Optimal Control optimizes (minimizes or maximizes) a cost function based on the system parameters for given values of q and r Finally, Predictive Control is tuned in the Time Domain Two parameters determine the closed-loop response and a few simple rules are used to determine their values

From the results presented in this chapter, full PID compensation performs better than the other techniques It has an extremely low static error due to the integrator in its action Also the peak error values recorded when tracking a specified trajectory are the lowest in magnitude of the controllers presented here The Adapted Monoreg Algorithm is a close second place Its simplicity is its advantage Lead Compensation also performs well but the optimal control technique does not seem suited for manipulator control The assumptions before solving for the optimal controller are the downfall of this method However variations of nonlinear optimal control are currently under investigation and the results could prove encouraging. The second Predictive Control method is not suitable for use in this area The linear models do not specify the joint dynamics sufficiently and the algorithm suffers from this inaccuracy

Fixed parameter algorithms suffer from several disadvantages due to their lack of flexibility Adaptive controllers, which are discussed in Chapter 5, can overcome some of these problems by continually updating the control gains Using the results from an identification the controller parameters can be derived These gains are continuously adapted to cater for varying conditions Because the robot is highly nonlinear, its parameters varying widely over its operating range Thus it is better to varying the controller gains also Linear and nonlinear adaptive routines exist but only the linear techniques are investigated

Index to Graphs

PD Control Results

PID Control Results

Lead Control Results

- Fig 4 9a *Bode Plot of Gain K G(s)*

- Fig 4 9a *Bode Plot of Phase K G(s)*

- Fig 4 9b *Bode Plot of Gain K(s) G(s)*

- Fig 4 9b *Bode Plot of Phase K(s) G(s)*

- Fig 4 10a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory Demands*

- Fig 4 10b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory Demands*

- Fig 4 10c *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

- Fig 4 11a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory Demands*

- Fig 4 11b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory Demands*

- Fig 4 11c *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

Lag Control Results

- Fig 4 12 *Bode Plot of Gain K(s) G(s)*

- Fig 4 12 *Bode Plot of Phase K(s) G(s)*

- Fig 4 13a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory Demands*

- Fig 4 13b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory Demands*

□ Fig 4 13c *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

□ Fig 4 14a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory Demands*

□ Fig 4 14b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory Demands*

□ Fig 4 14c *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

Lag-Lead Control Results

□ Fig 4 15 *Bode Plot of Gain K(s) G(s)*

□ Fig 4 15 *Bode Plot of Phase K(s) G(s)*

□ Fig 4 16a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory Demands*

□ Fig 4 16b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory Demands*

□ Fig 4 16c *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

Optimal Control Results

□ Fig 4 17a *Plot of Control Inputs versus Time for Constant Setpoint Demands*

□ Fig 4 17b *Plot of Joint Positions versus Time for Constant Setpoint Demands*

□ Fig 4 18a *Plot of Control Inputs versus Time for Constant Setpoint Demands*

□ Fig 4.18b *Plot of Joint Positions versus Time for Constant Setpoint Demands.*

□ Fig 4 19a *Plot of Control Inputs versus Time for Constant Setpoint Demands*

Predictive Control Results (Method 1)

Predictive Control Results (Method 2)

- Fig 4 26b *Plot of Joint Positions versus Time for Constant Setpoint Demands*

- Fig 4 27a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory Demands*

- Fig 4 27b *Plot of Joint Positions versus Time for Cubic Spline Trajectory Demands*

- Fig 4 27c *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

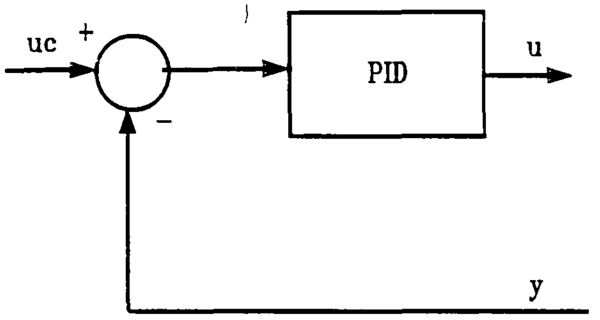


Fig.4.1a Textbook Controller

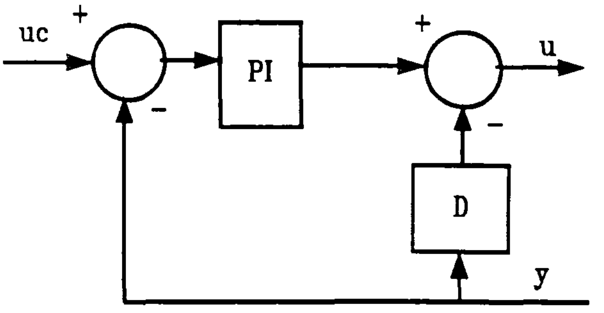


Fig.4.1b Derivative of Output Controller

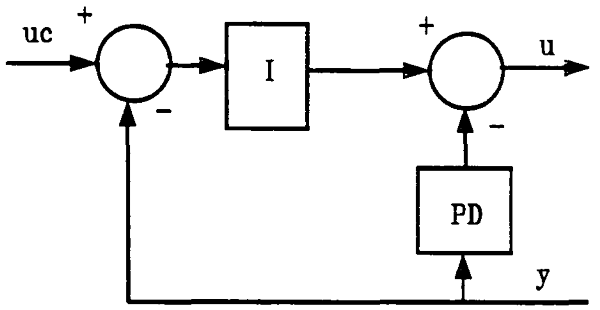


Fig.4.1c Set Point on I only controller

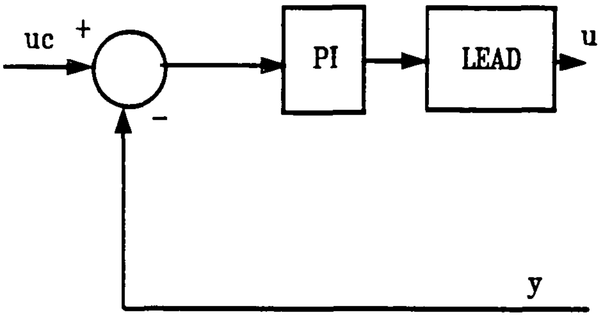


Fig.4.1d PI followed by Lead Network

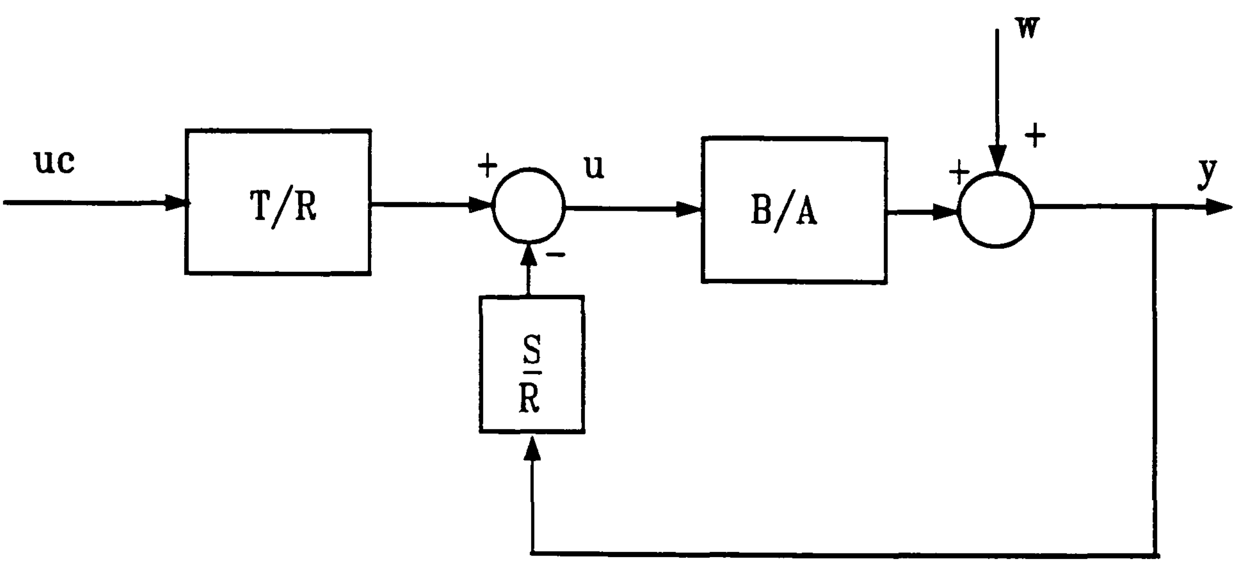


Fig.4.2 A Common General Form for the PID controller based on Pole Placement design

Fig 4 3a

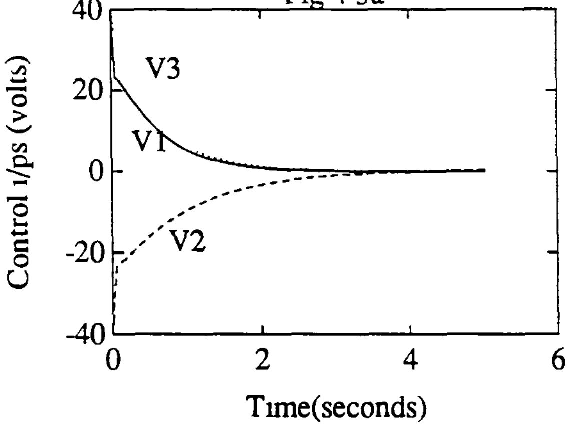


Fig 4 3b

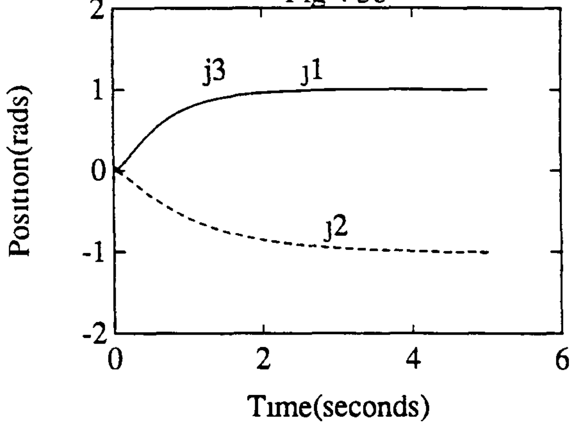


Fig 4 4a

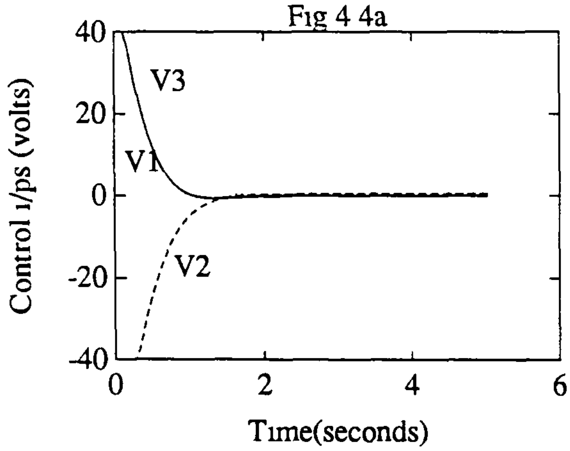


Fig 4 4b

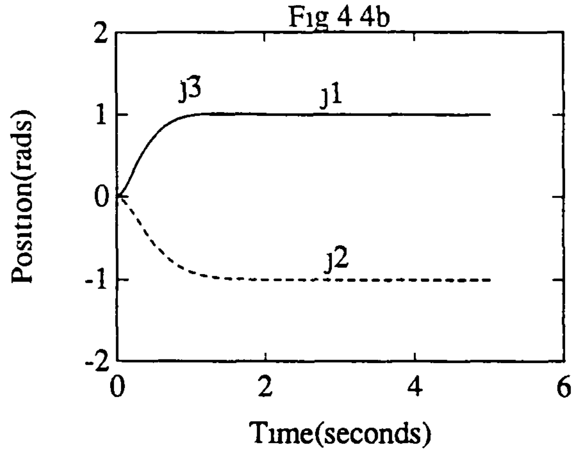


Fig 4 5a

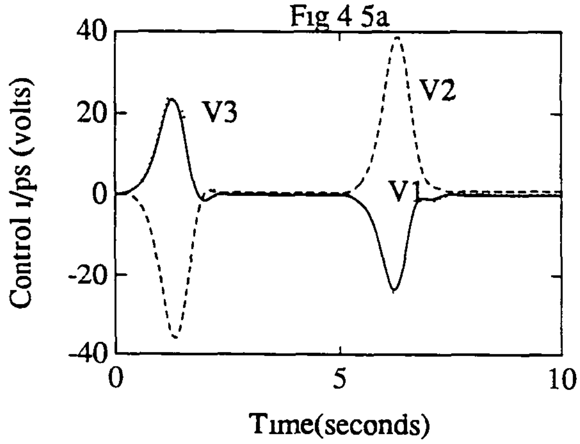


Fig 4 5b

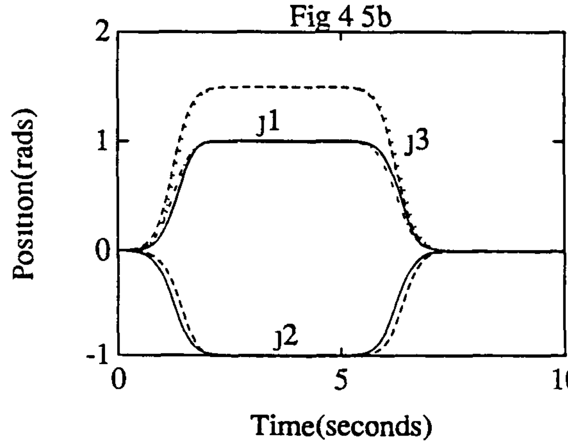
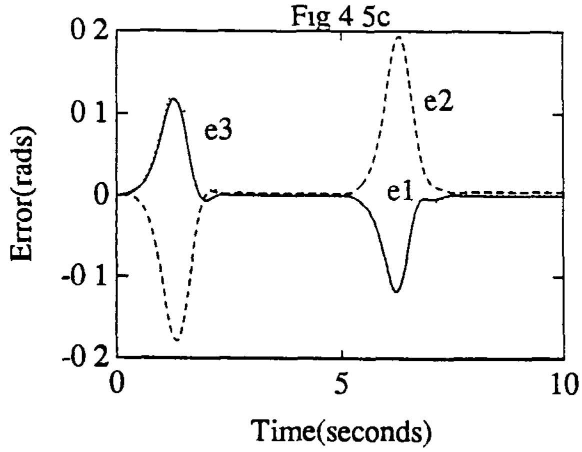


Fig 4 5c



SIMULATION RESULTS
USING A PD CONTROLLER

Fig 4 6a

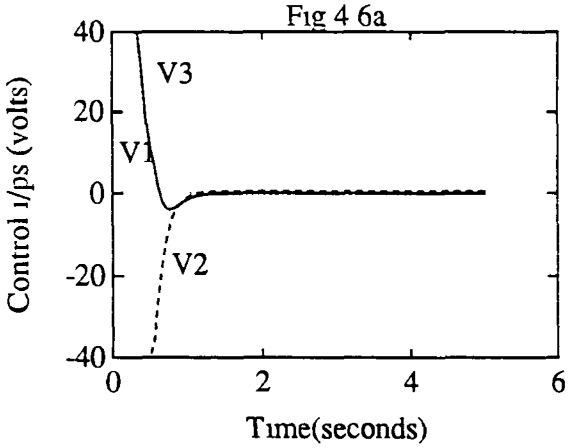


Fig 4 6b

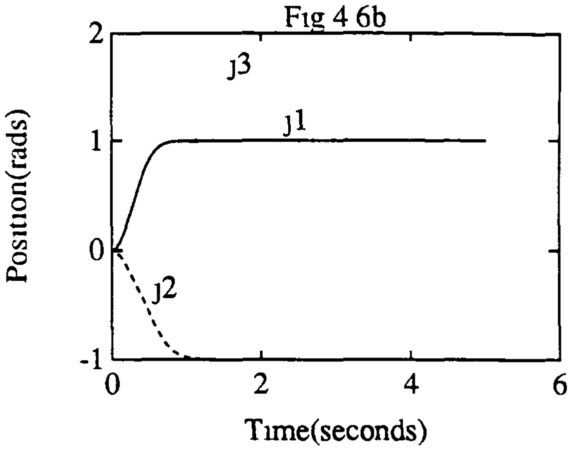


Fig 4 7a

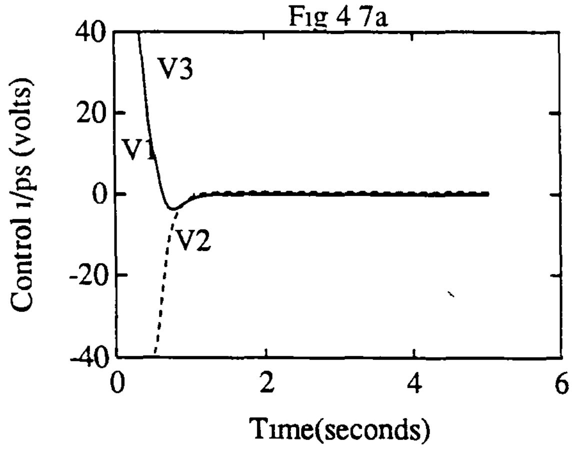


Fig 4 7b

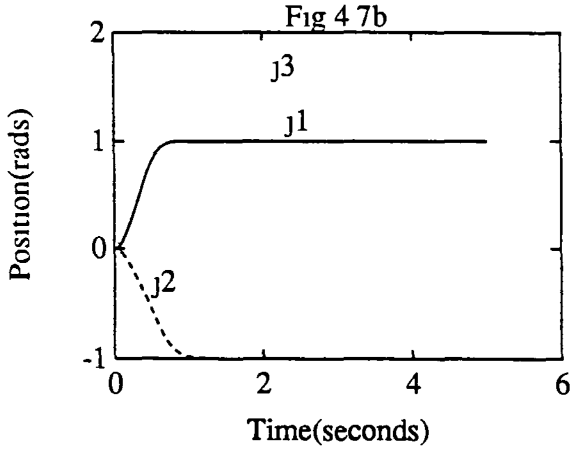


Fig 4 8a

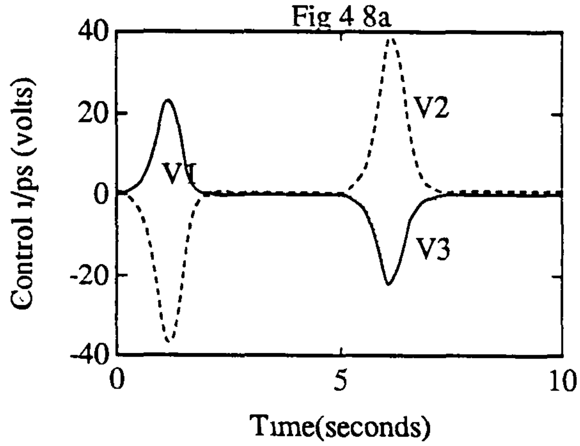


Fig 4 8b

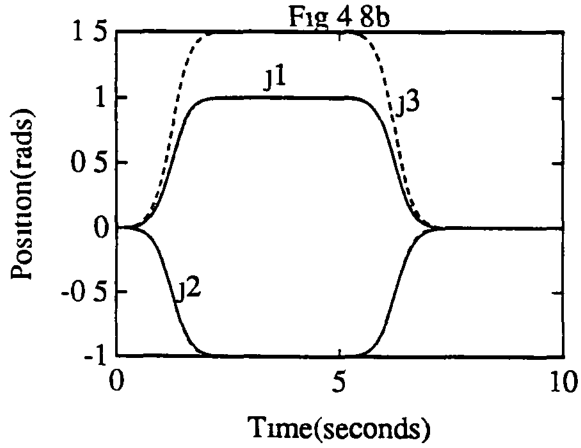
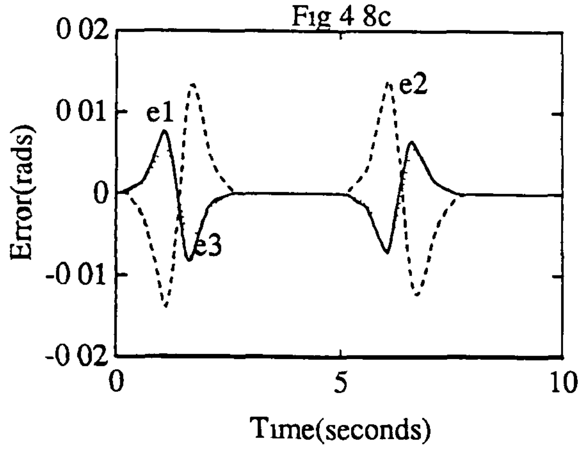


Fig 4 8c



SIMULATION RESULTS
USING A PID CONTROLLER

Fig 4 9a Bode Plot of K G(s)

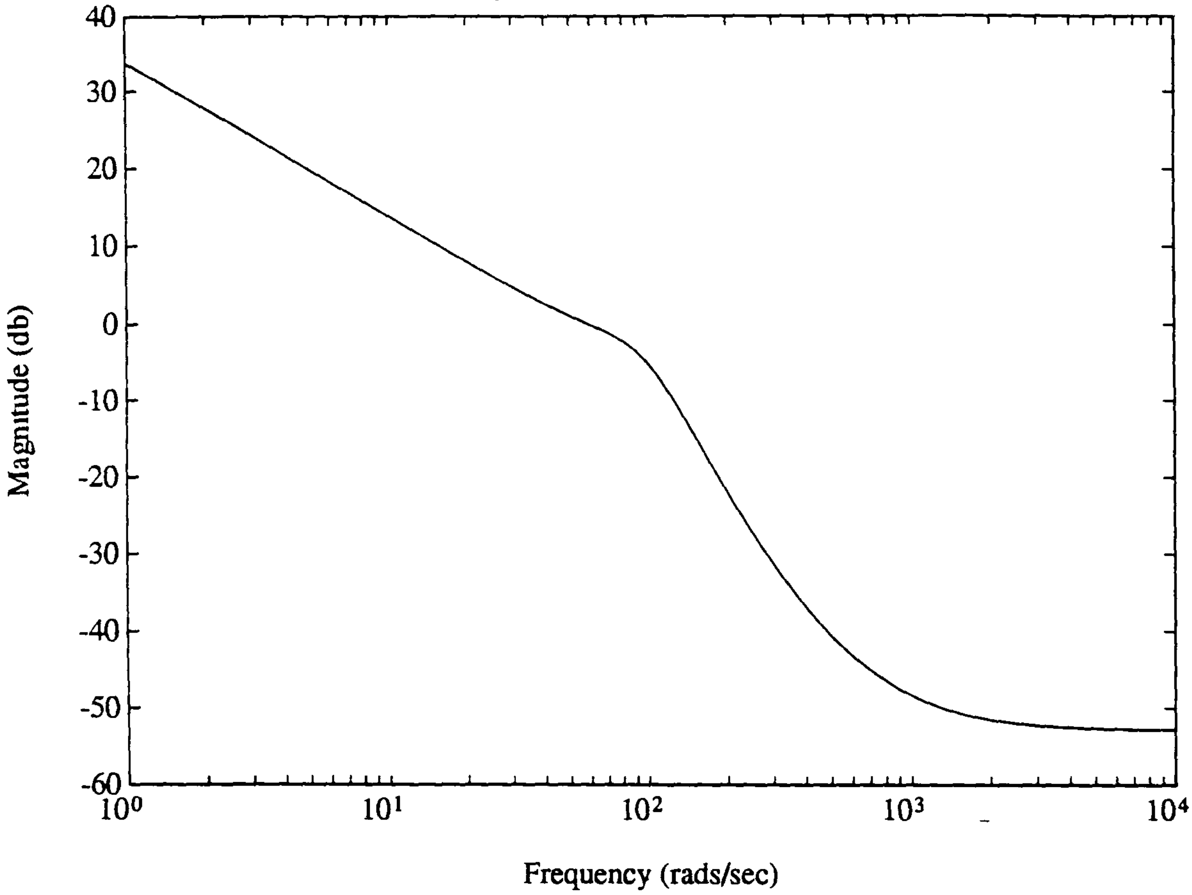


Fig 4 9a Bode Plot of K G(s)

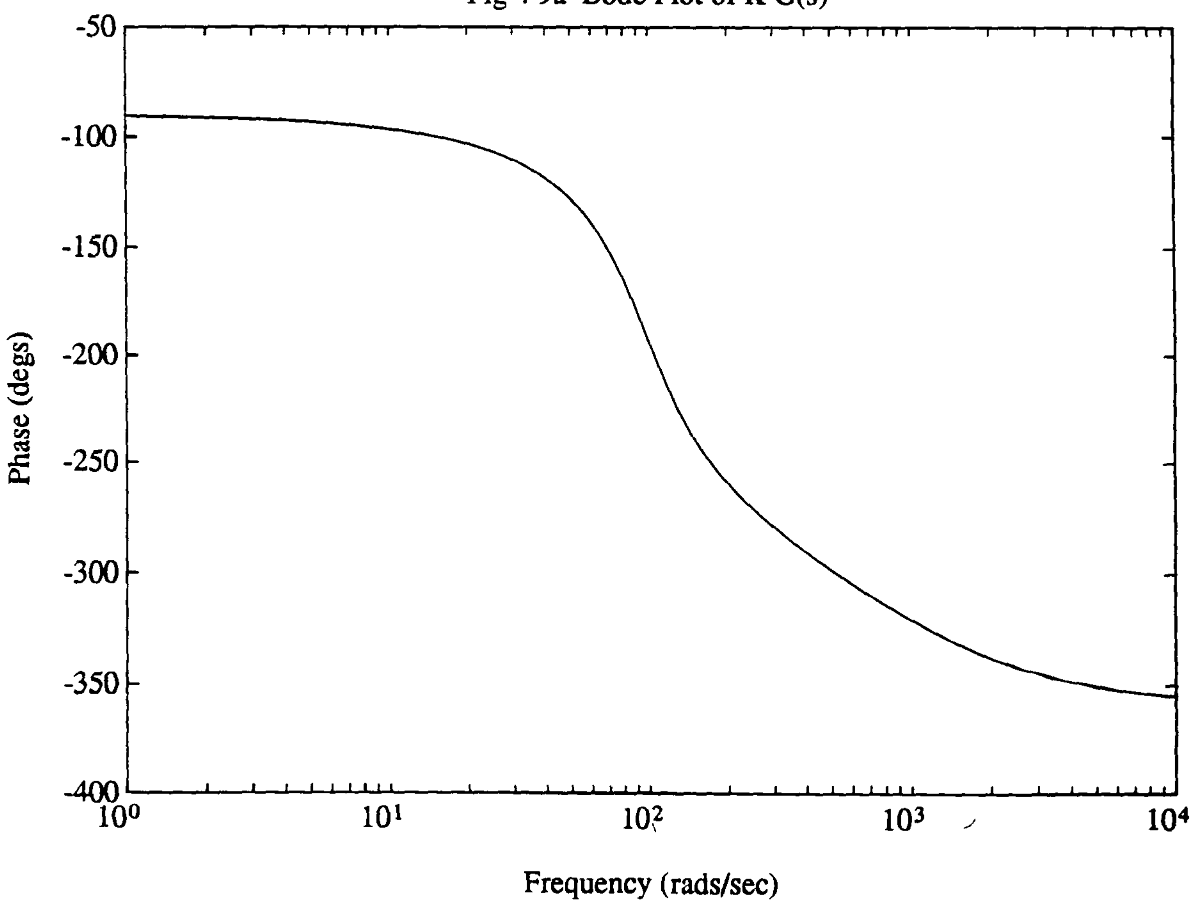


Fig 4 9b Bode Plot of K(s) G(s)

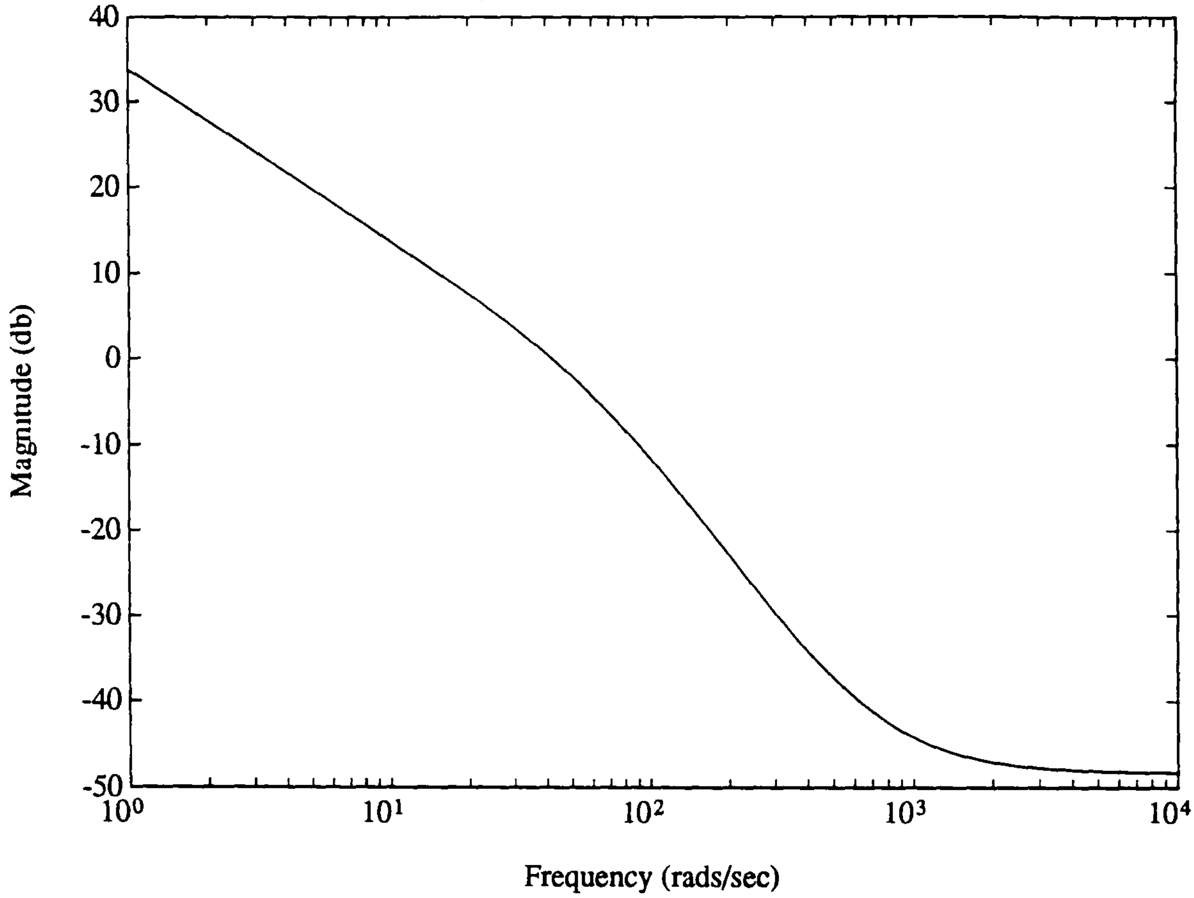
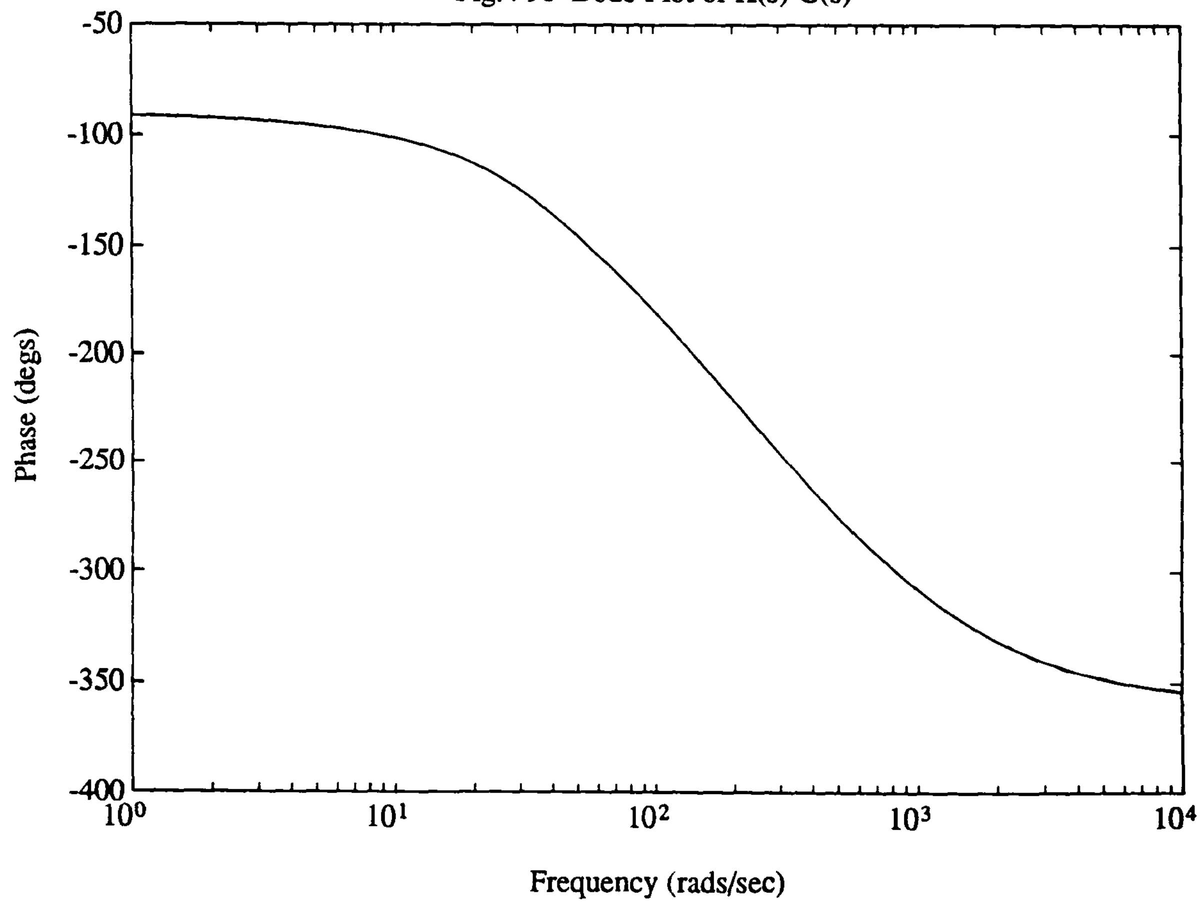
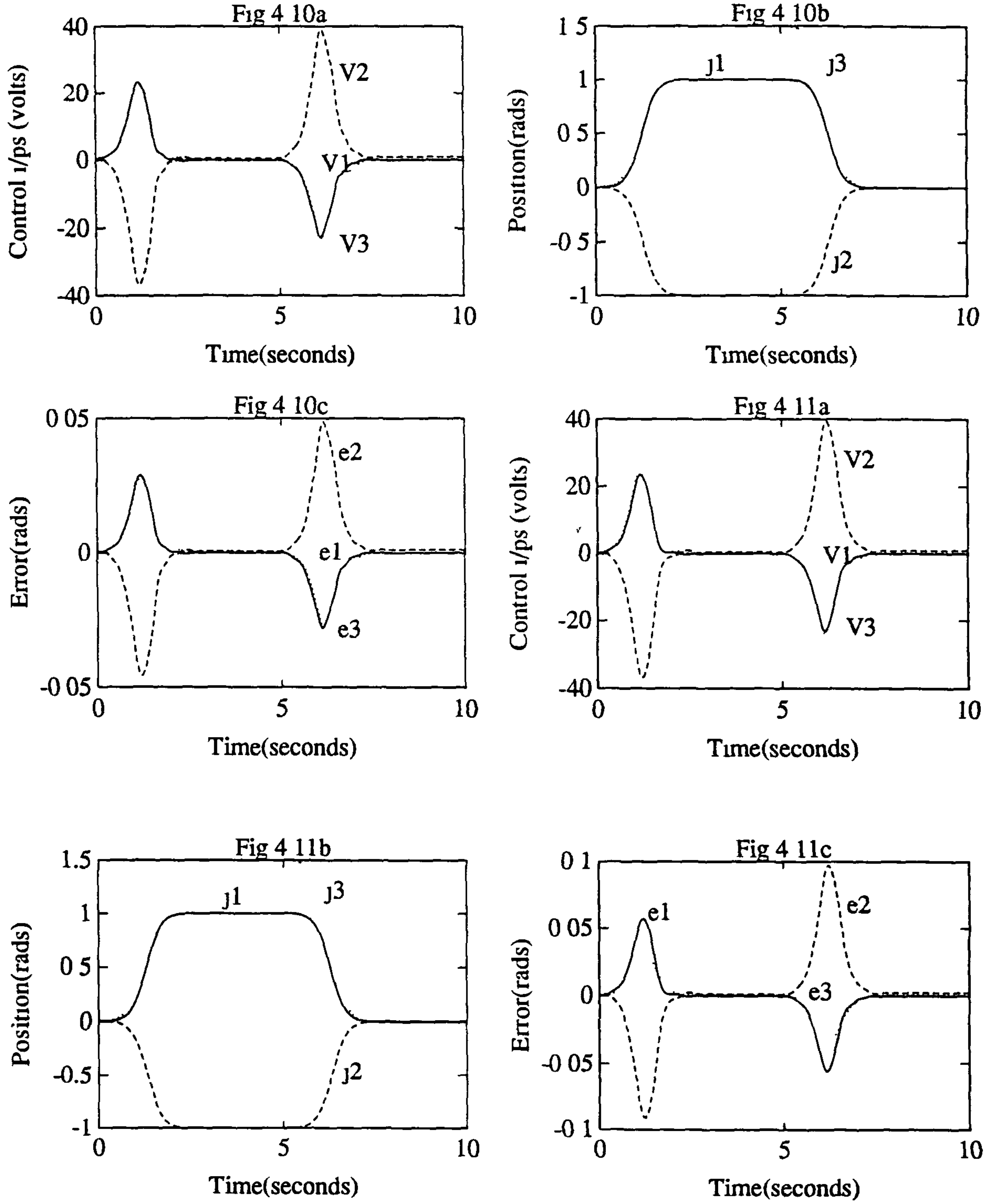


Fig.4 9b Bode Plot of K(s) G(s)





SIMULATION RESULTS USING LEAD COMPENSATION

Fig 4 12 Bode Plot of K(s) G(s)

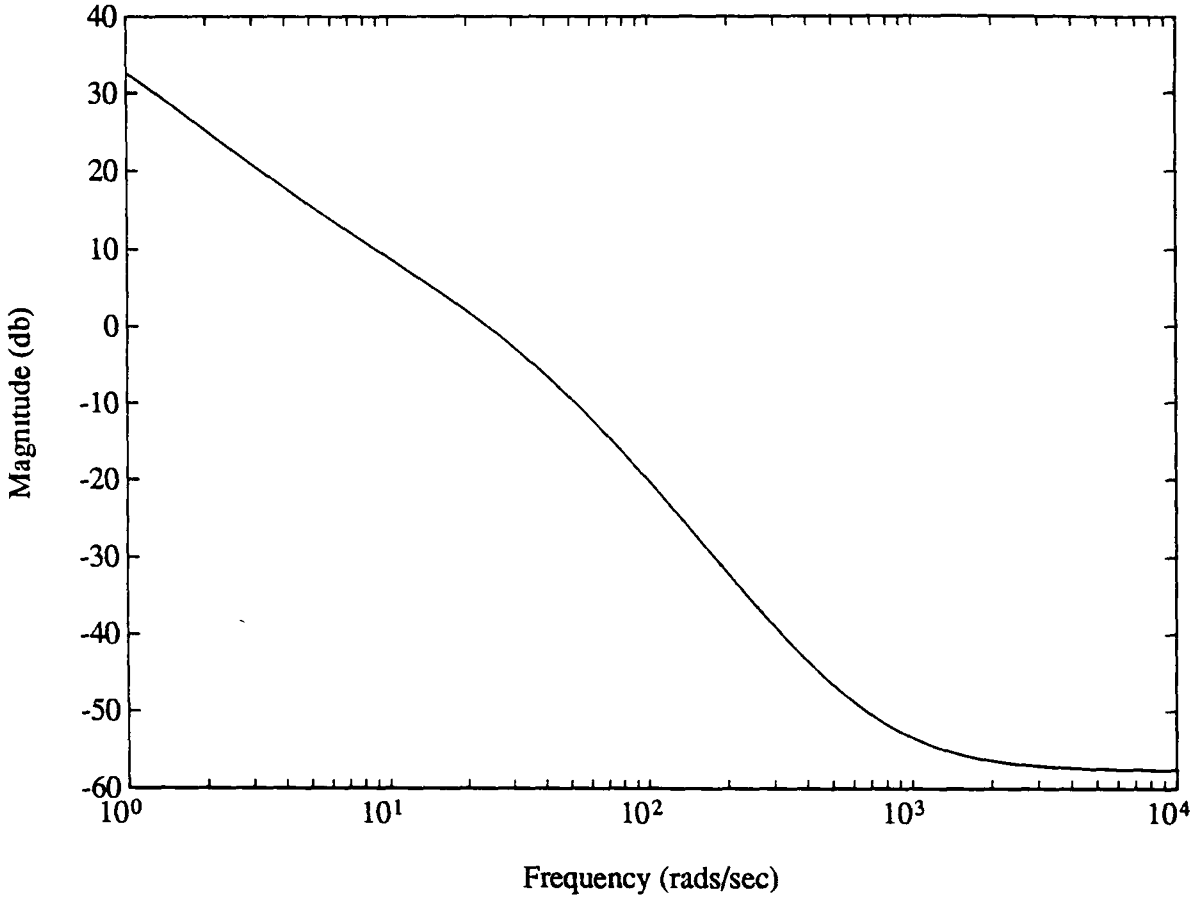
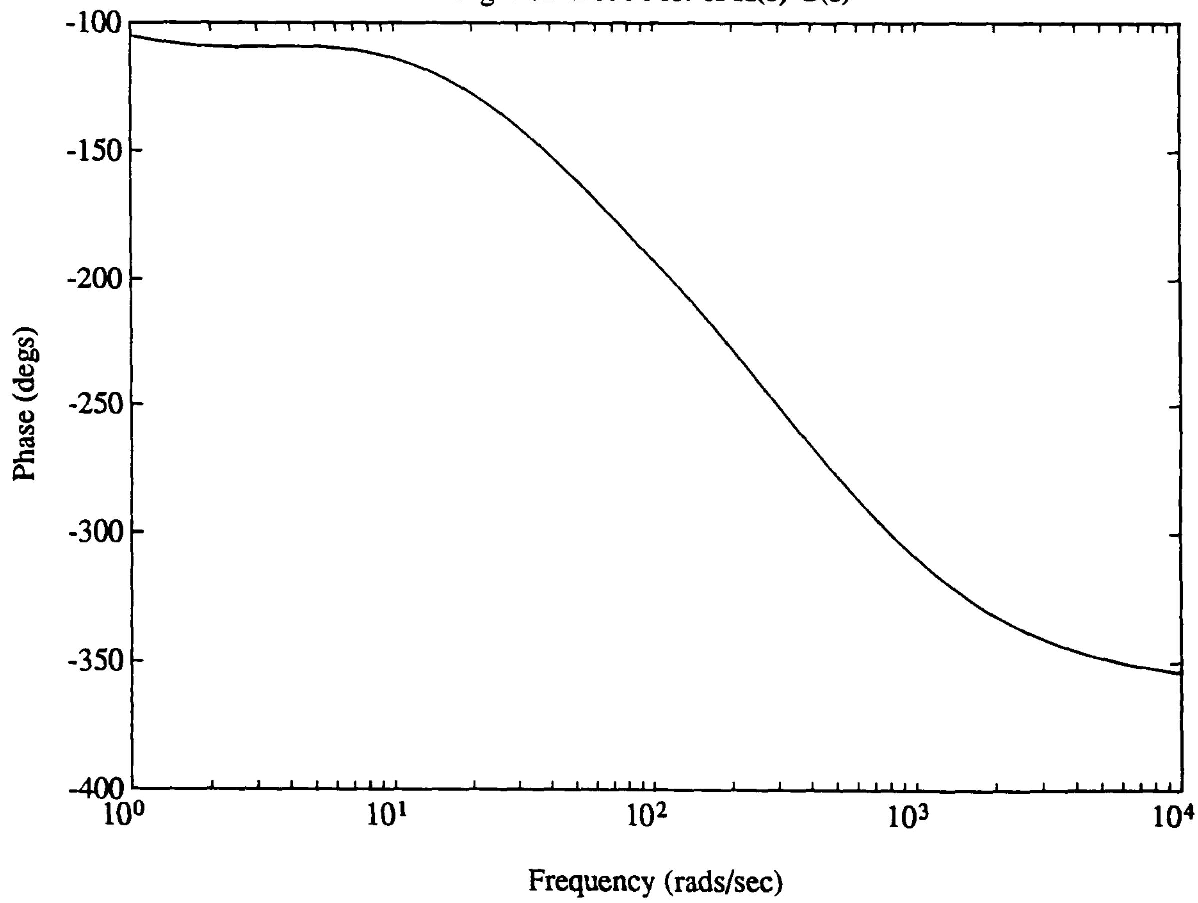
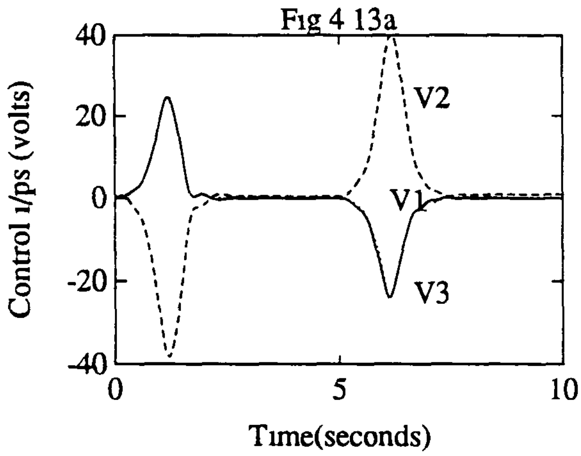
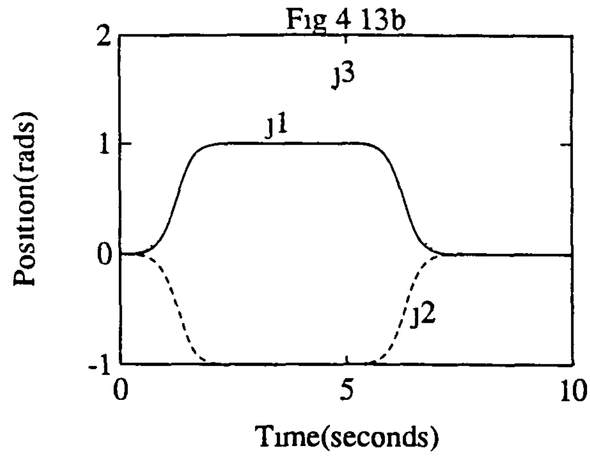
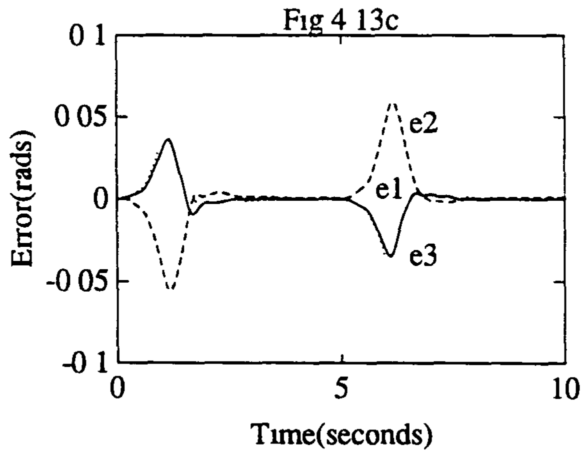


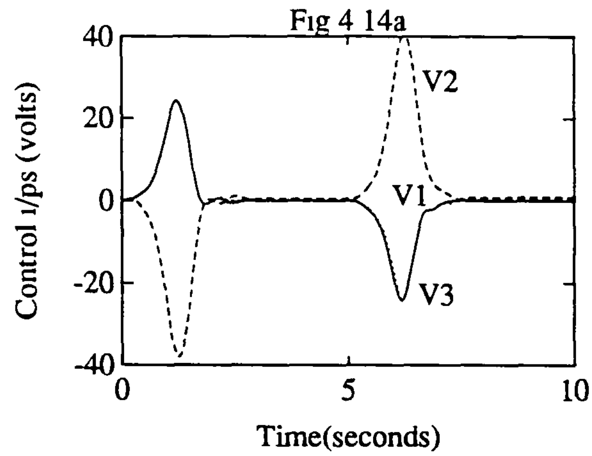
Fig 4 12 Bode Plot of K(s) G(s)

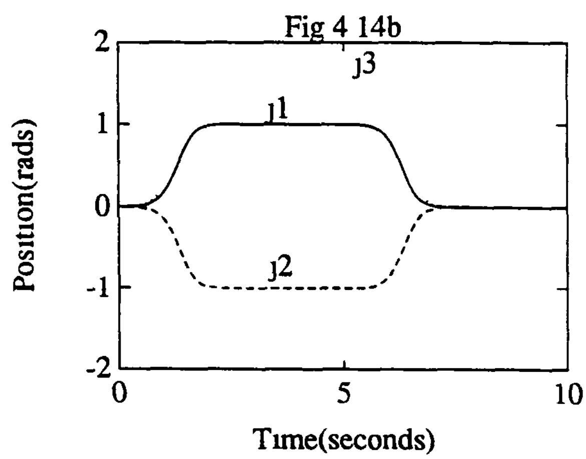


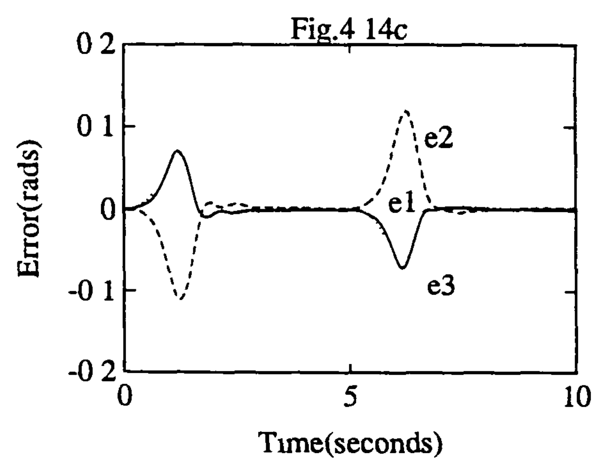












SIMULATION RESULTS USING LAG COMPENSATION

Fig 4 15 Bode Plot of K(s) G(s)

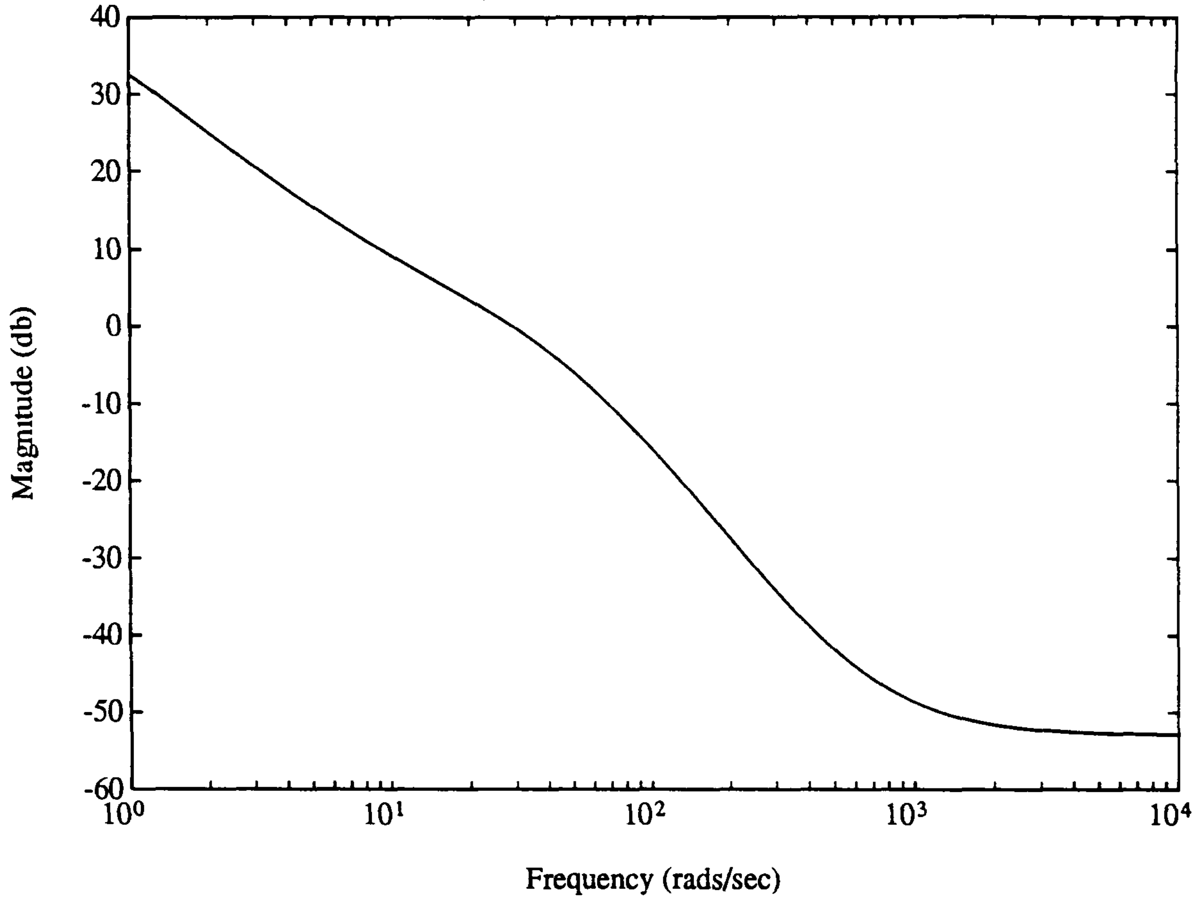
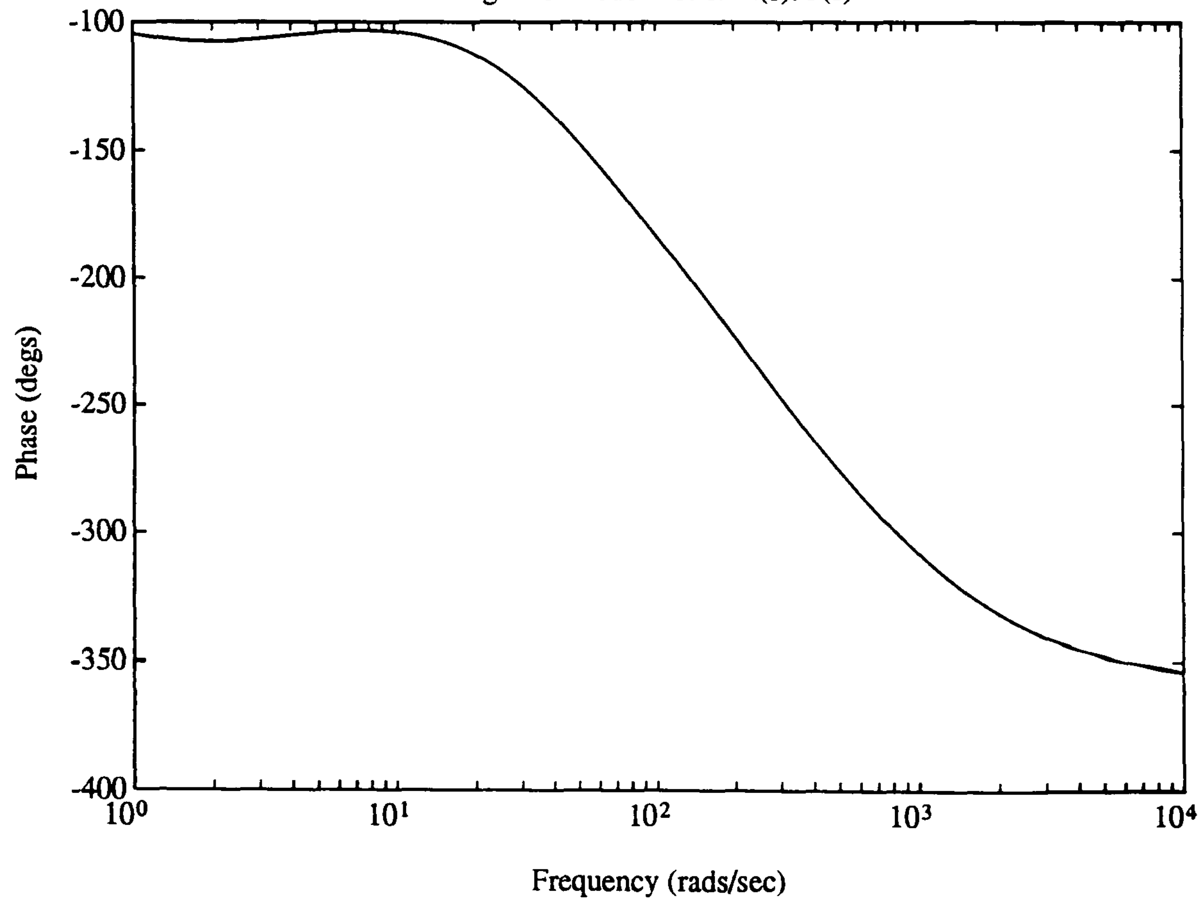
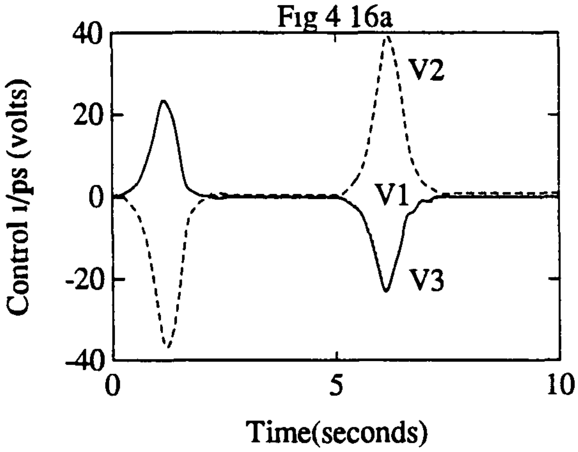
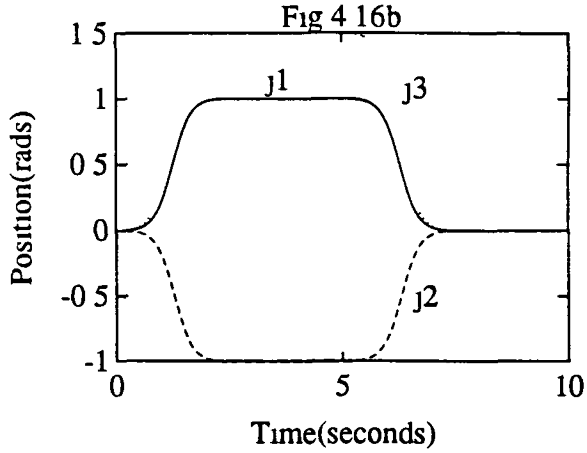
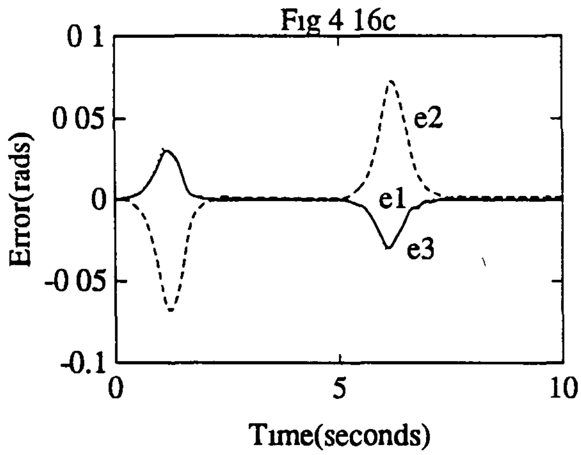


Fig.4 15 Bode Plot of K(s).G(s)

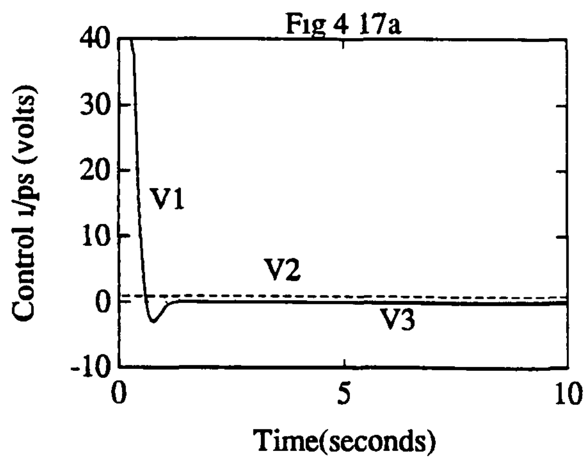








SIMULATION RESULTS USING
LAG-LEAD COMPENSATION



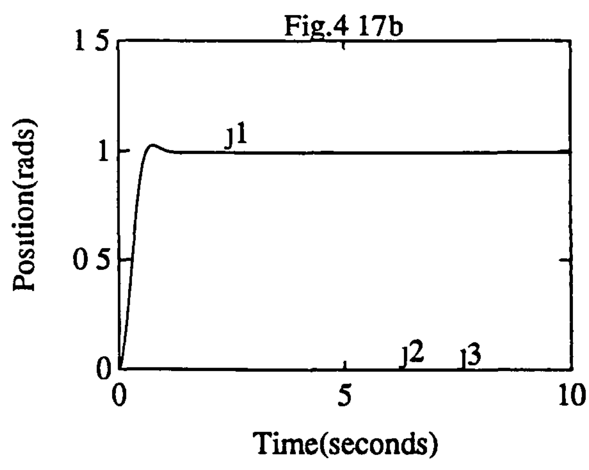


Fig 4 18a

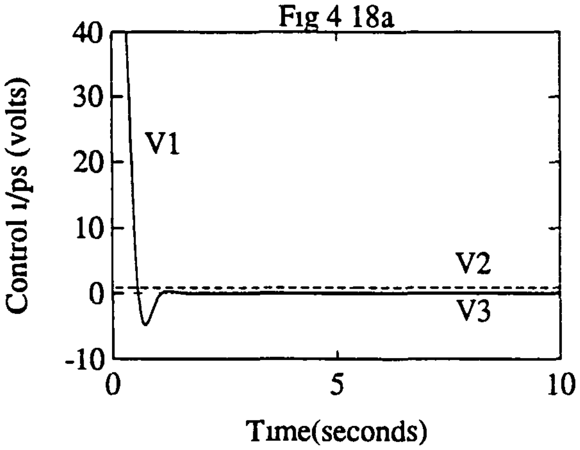


Fig 4 18b

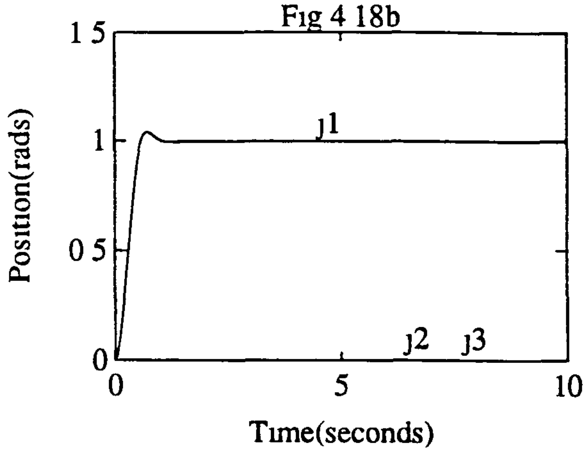


Fig 4 19a

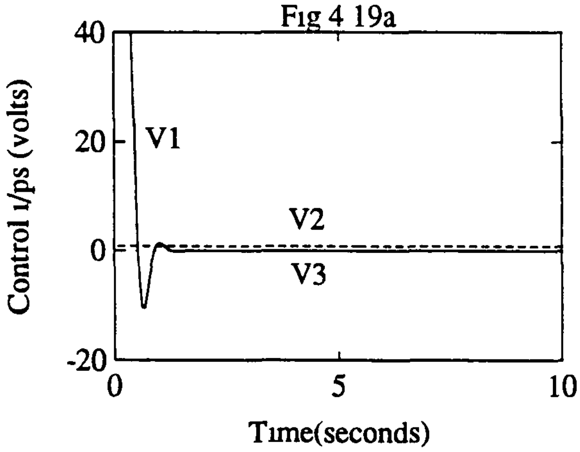


Fig 4 19b

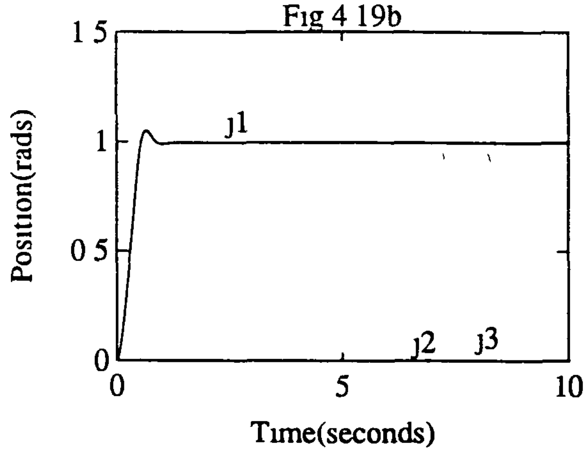


Fig 4 20a

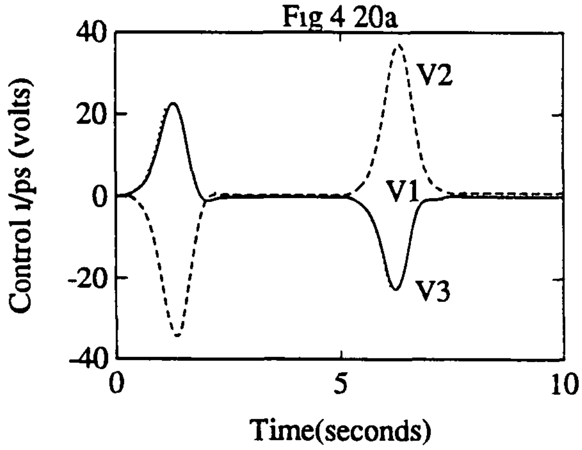


Fig 4 20b

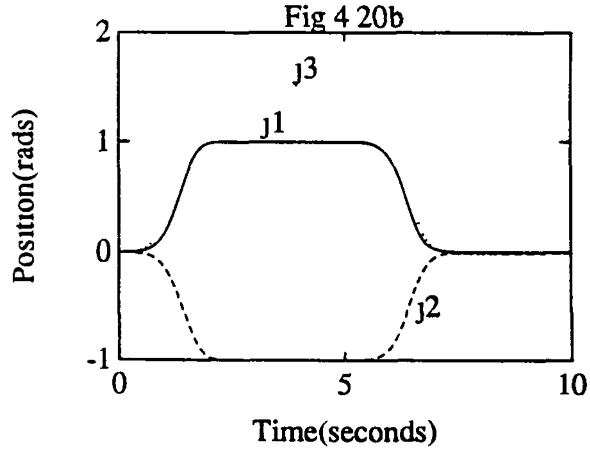
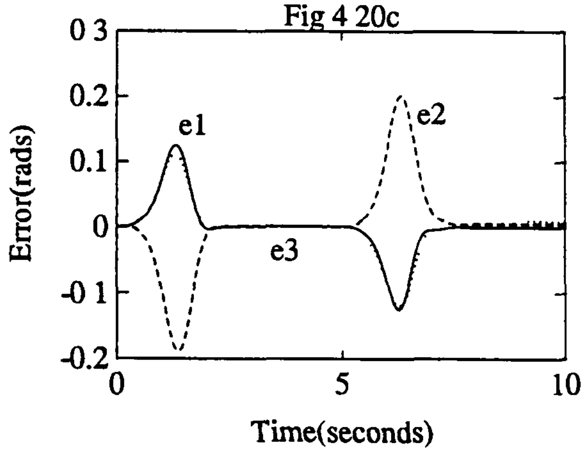


Fig 4 20c



SIMULATION RESULTS
USING OPTIMAL CONTROL

Fig 4 22a

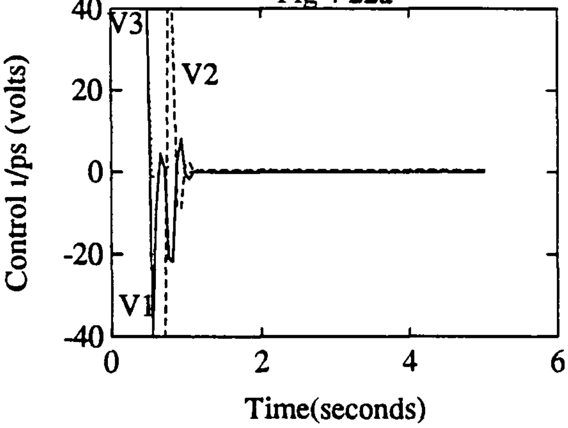


Fig 4 22b

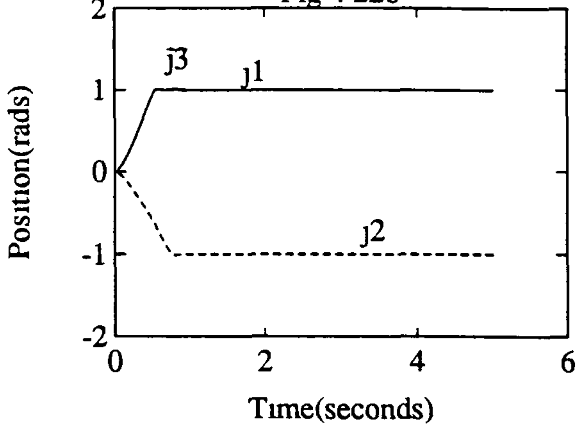


Fig 4 23a

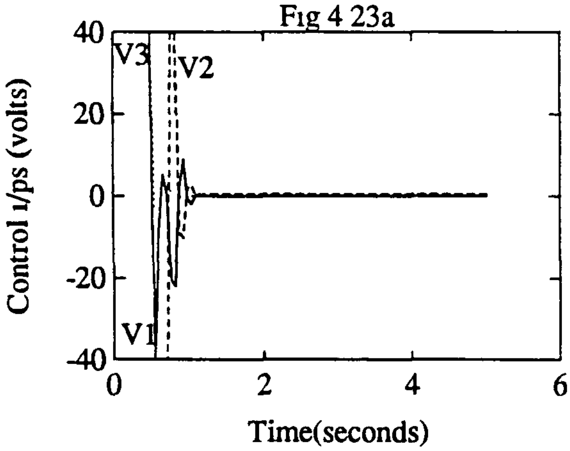


Fig 4 23b

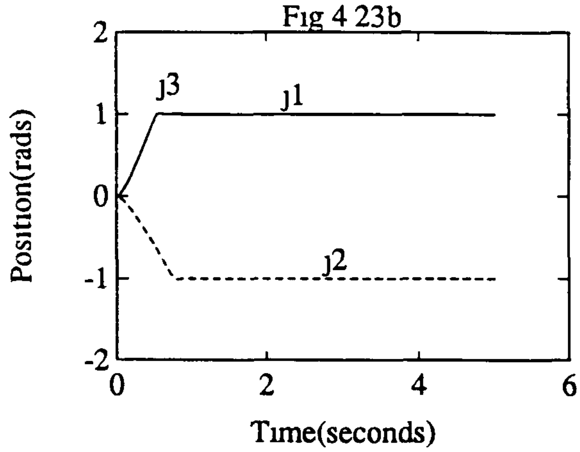


Fig 4 24a

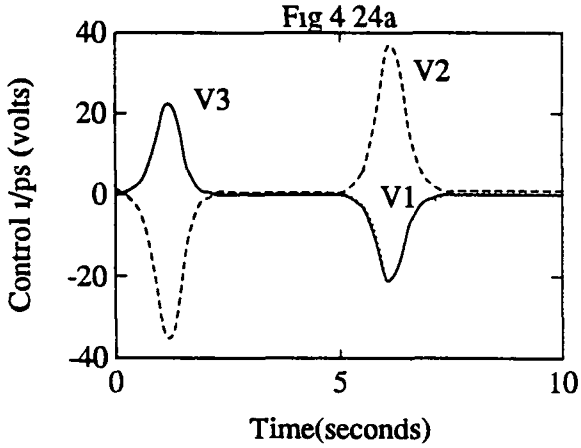


Fig 4 24b

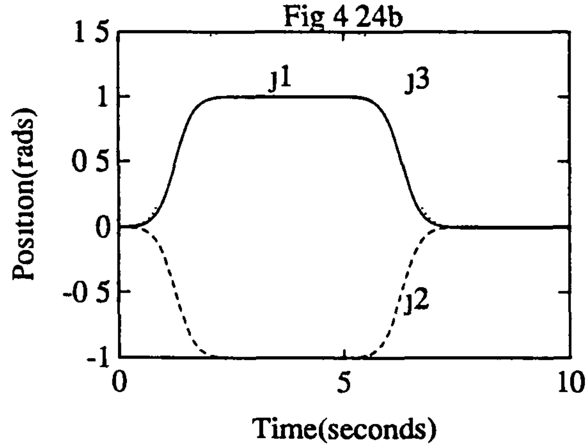
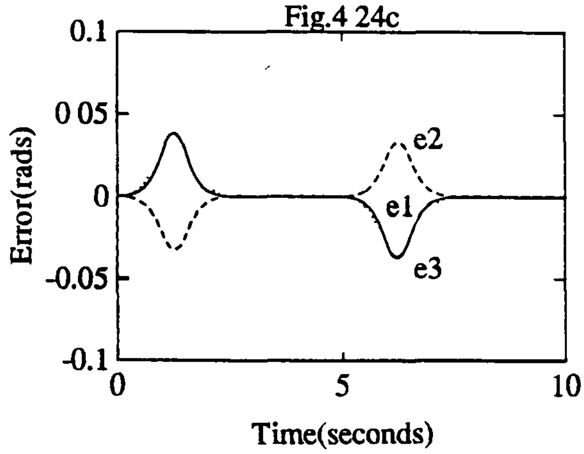


Fig.4 24c



SIMULATION RESULTS USING FULL STATE FEEDBACK PREDICTIVE CONTROL

Fig 4 25a

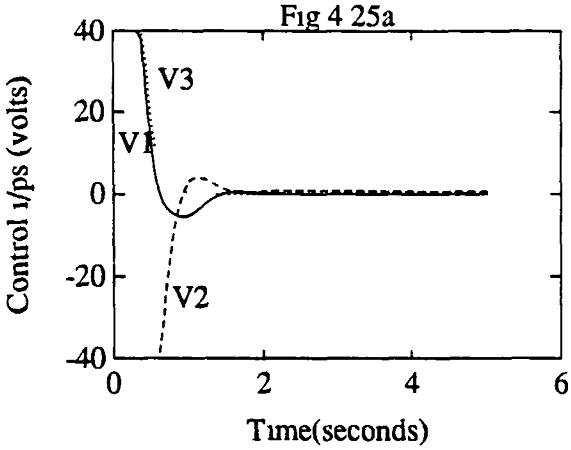


Fig 4 25b

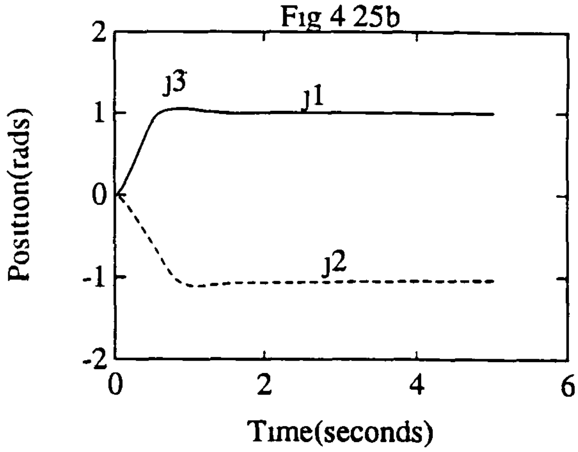


Fig 4 26a

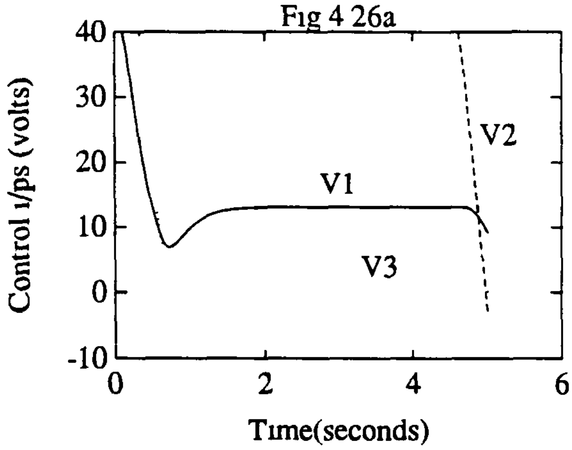


Fig 4 26b

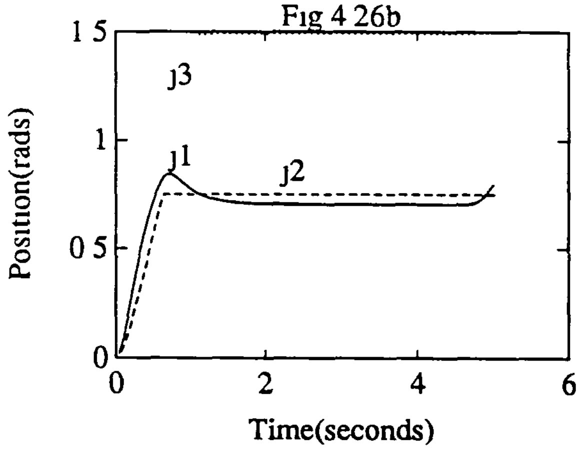


Fig 4 27a

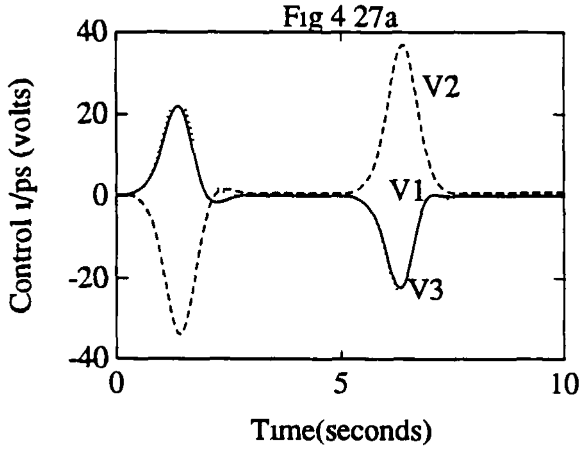


Fig 4 27b

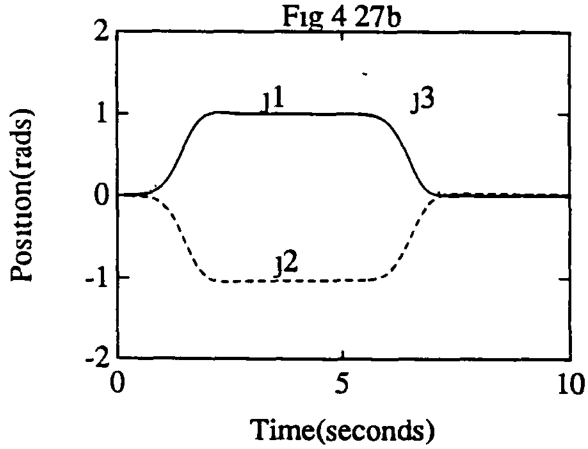
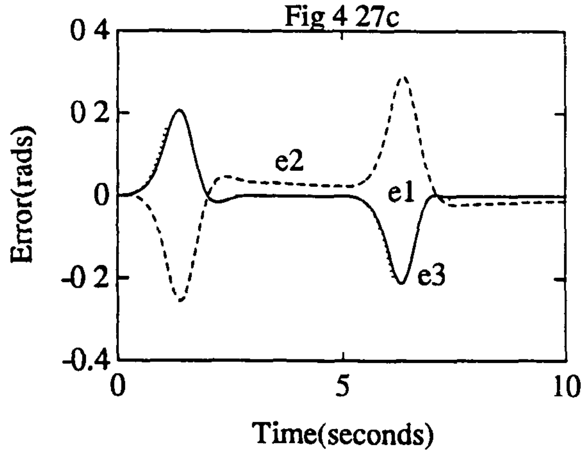


Fig 4 27c



SIMULATION RESULTS USING PREDICTIVE CONTROLLER NUMBER2

CHAPTER 5

ADAPTIVE CONTROL STRATEGIES

In the last chapter, it is assumed that all the robot joints could be represented by linear models that are fixed with time In fact, this is true for virtually every manipulator controller currently being manufactured A number of robot controllers do permit the user to specify the load before performing any move operations. Parameters that give the best (compromise) performance are then downloaded from a table located in memory This is a form of adaptive control, called Gain-Scheduling and has been used for years in the field of missile guidance [62]

To improve the performance of the robot, the area of adaptive control is investigated Dynamically compensating for inertial load variation, by constantly adjusting servo parameters, results in improved operation characteristics, but safeguards must be taken In this chapter, some of the fixed gain controllers, of Chapter 4 are transformed to adaptive routines to observe the improvements, if any Single-loop adaptive control schemes are examined here. This type of adaptive controller may also compensate (to some degree) for interaction between joints Before adaptive control is performed, a suitable parameter identification routine is required. Many routines are available, and these are discussed with reference to the robot Adaptive Control can be divided into two classifications Explicit control is where the plant parameters are calculated in the recursive identification, in contrast to the implicit type where the identification produces the controller gains Thus, the control design step has been avoided (see Fig 5 1)

Nonlinear control theory is a topic of continued investigation. It is by no means an area of general theories, and nonlinear control methods are very specific in their applications. Adaptive control can be used to control nonlinear processes and can be applied to a wide variety of applications.

Hence, the area of adaptive control is concerned with the study and design of controllers and regulators that adjust to the varying properties of the controlled process AGC (automatic gain control) in radio, which adjusts the receiver gain so that its output level is relatively constant over a wide range of input signal amplitudes, is an example of such an adaptive system

5.1 Identification Techniques

The Method of Least Squares is the most commonly known identification algorithm Two different types of identification are possible

1 Off-line or Batch identification
2 On-line or Recursive identification.

There are two advantages of recursive identification over off-line identification One is that the decision of what model structure to use has to be made a priori, before starting the recursive identfcation procedure In the off-line situation different types of models can be tried out. The second advantage is that, with few exceptions, recursive methods do not give as good an accuracy of the models as off-line methods However, during adaptive control, it is necessary to infer the model at the same time as the data is collected The model is then updated at each sample instant when some new data becomes available [26]

Least Squares is the method of identfcation used here. It is a flexible routine and it is easy to change the number of identified parameters A parameter vector θ is estimated from the measurements of y(t) This estimate is chosen by minimizing what is left unexplained by the model, i e the equation error e(t) Minimization is done with respect to θ Variations on the basic Least Squares algorithm exist. A model is put on the error in Extended Recursive Least Squares. This helps to reduce any bias that may exist in the presence of non-white noise. These extensions are explained in the following sections.

5.1.1 Recursive Least Squares (RLS)

When optimal control theory has been applied to the construction of robot controllers, a common simplification of the above model is to assume that the coupling terms, due to the other joints, can be neglected [27][28] By assuming this, and by assuming that the PUMA 560 system parameters are slowly time-varying with negligible measurement noise, it is possible to apply the simplest form of RLS to the identification of this robot's parameters. This model can be written as

$$y(k) = A(q^{-1})y(k-1) + B(q^{-1})u(k-1) + e(k) \qquad (5.1)$$

If the parameter vector θ and the regressor information vector Φ are defined as

$$\theta^T = (a_1, \ldots, a_n, b_1, \ldots, b_n) \qquad (5.2)$$

and

$$\Phi^T = [\, y(k-1), \ldots, y(k-n),\ u(k-1), \ldots, u(k-n)\,] \qquad (5.3)$$

then the model can be written as

$$y(k) = \theta^T \Phi(k-1) + e(k) \qquad (5.4)$$

The parameter estimation problem is to find the estimates of the unknown parameters which minimize the loss function

$$E(\theta_i) = \frac{1}{m+1} \sum_{i=1}^{m} [e_i(k)]^2 \qquad (5.5)$$

where $e_i(t)$ is the prediction error in the parameters of joint i, and m is the number of parameters being estimated. The principle underlying Least Squares is that by minimizing the prediction error it is possible to minimize what is unexplained in the model. The solution to the Least Squares problem is furnished by the following recursive equations [26]

$$\theta_i(k) = \theta_i(k-1) + P(k)\Phi(k-1)\,[\, y_i(k) - \theta^T{}_i(k-1)\Phi(k-1)] \qquad (5.6)$$

$$P(k) = \frac{1}{\mu}\left[P(k-1) - \frac{P(k-1)\Phi(k-1)\Phi^T(k-1)P(k-1)}{\mu + \Phi^T(k-1)P(k-1)\Phi(k-1)} \right] \qquad (5.7)$$

where P is the covariance matrix (2nx2n) of the estimation errors and μ is what is known as the forgetting factor. The P matrix is a positive definite measure of the estimation error and its elements tend to decrease as time increases. It is therefore necessary to initialize the elements of this matrix to some large value, to ensure that its elements do not tend to zero too rapidly. If this occurs equation (5.6) reduces to

$$\theta_1(k) = \theta_1(k-1) \tag{5.8}$$

and the estimated values become constant before they have converged to a value close to or equal to the true model parameters. An initial value [29] of 1000 on the diagonal elements of the P matrix should prevent this problem occurring. Once the estimates have reached their true value, the P matrix elements tend to zero. As a result, any parameter which drifts with time in the system will only be tracked until the P elements become zero. To overcome this, [29] suggests the use of a forgetting factor (μ). This factor can be used to account for an exponential decay of past data in tracking a slow drift in the system parameters. It works by dividing the elements of the P matrix by a value less than 1. This prevents the elements of P becoming zero. The value of μ is generally in the region of 0.95 to 1.0. A value of μ equal to 0.95 results in an estimation method which is capable of tracking time variance in the system parameters but which fails to converge totally to its true value. To obtain a tradeoff between good estimates and time variance monitoring, [29] suggests the use of an exponential forgetting factor which tends towards a value of 1 as time tends to infinity. The forgetting factor chosen for this application is given by

$$\mu(t) = 0.95\mu(t-1) + 0.95 \tag{5.9}$$

with $\mu(0)$ equal to zero.

5.1.2 Modified Recursive Least Squares (MRLS)

This method is based on the least squares model just described. This more comprehensive autoregressive model can be written as

$$y(k) = A(q^{-1})y(k-1) + B(q^{-1})u(k-1) + h + e(k) \tag{5.10}$$

where h is a forcing term intended to include the nonlinear effects of torque-dependent terms. In this case, the parameter estimates and the regressors can be written in the following vector format

$$\theta^T = (a_1, \quad , a_n, b_1, \quad , b_n, h_1) \tag{5.11}$$

and,

$$\Phi^T = [\ y(k-1),\quad ,y(k-n),\ u(k-1),\quad ,u(k-n),1] \tag{5 12}$$

The autoregressive model can be written as

$$y(k) = \theta^T\ \Phi(k-1) + e(t) \tag{5 13}$$

This is the format required, and it is possible to apply the loss function of equation (5 5) for minimizing the prediction error This results in the parameters being identified by equations (5 6) and (5 7) To ensure that this estimation method has the same ability as the RLS algorithm to track time varying parameters the same forgetting factor scheme is used

5.1.3 Extended Least Squares

This method attempts to estimate a model for the noise present in any system, as well as the system model itself This model can be written in time series form as follows

$$y(k) = A(q^{-1})y(k-1) + B(q^{-1})u(k-1) + C(q^{-1})e(k) + d(k) \tag{5 14}$$

where $C(q^{-1})$ is the polynomial containing the parameters of the noise model and d(k) is called the loaded disturbance variable In this case, the parameter estimates and the regressors can be written in the following vector format .

$$\theta^T = (a_1,\quad ,a_n,b_1,\quad ,b_n,c_1,\quad ,c_n) \tag{5 15}$$

and,

$$\Phi^T = [\ y(k-1),\quad ,y(k-n),\ u(k-1),\quad ,u(k-n),e(k),\quad ,e(k-n+1)] \tag{5 16}$$

The autoregressive model can be written as

$$y(k) = \theta^T\ \Phi(k-1) + e(t) \tag{5 17}$$

This means that equations (5 6) and (5 7) can be used to update the parameter estimates of the model Once again the same variable forgetting factor is used to track parameter variations

5.1 4 Nonlinear Least Squares

This method attempts to estimate a model for the residual as a combination of linear and nonlinear functions It does this by formulating the autoregressive model [30] as follows

$$y(k) = A(q^{-1})y(k-1) + B(q^{-1})u(k-1) + C(q^{-1})e(k) + N(k) \tag{5 18}$$

where $C(q^{-1})$ is the polynomial containing the parameters of the noise model and $N(k)$ is a nonlinear polynomial defined by

$$N(k) = n_1 u^2(k-1) + n_2 u^3(k-1) \tag{5 19}$$

In this case, the parameter estimates and the regressors can be written in the following vector format

$$\theta^T = (a_1, \quad ,a_n, b_1, \quad ,b_n, c_1, \quad ,c_n, n_1, n_2) \tag{5.20}$$

and

$$\Phi^T = [\ y(k-1), \quad ,y(k-n),\ u(k-1), \quad ,u(k-1), e(k), \quad ,e(k-n+1), \\ u^2(k-1), u^3(k-1)\,] \tag{5 21}$$

The autoregressive model can again be written as

$$y(k) = \theta^T\ \Phi(k-1) + e(t) \tag{5.22}$$

5.1.5 Results

Using the basic RLS algorithm, the parameters of each of the joints can be identified A second order model is identified for each joint. The following results show the performance, which proves to be satisfactory in a control environment. A pseudo-random binary sequence (prbs) is used as input to each of the joints to stimulate sufficiently the dynamics of the model. Fig 5.2a shows the numerator parameters, while Fig 5.2b shows the denominator coefficients. These results are obtained with $P_0 = 10{,}000$ and $\mu = 0.9$. The results shows that the identifier is in adaptive mode, i.e. the parameters converge to their values quickly but never actually settle at a constant level. In Fig 5.3a and Fig 5.3b, $\mu = 0.95$ and a slower response is obtained, where there is less oscillation by the parameters about their true values.

The other identification techniques are not investigated in this project. Jones [7], was involved in the analysis of this area. He concluded that the nonlinear identification technique results in a loss function lower in magnitude to the other methods. Hence, this method gives the best parameter estimates. This is because the model identifies a linear and a nonlinear part. Extended Least Squares also proves to be a good identification tool.

5.1.6 Conclusion

Although Recursive Least Squares is the simplest of the algorithms, it is suitable for robotic applications. Using input/ouput data, a second order model can be identified where four parameters are calculated. These parameters are used to calculate the adaptive controller gains. The choices of forgetting factor and P_0 influence the identification performance. The routine can be tuned for fast or slow parameter convergence, or can be tuned to deal with highly time varying parameters.

5.2 Adaptive PID Controllers

In an adaptive PID controller, the control parameters are obtained using an identifier and a control design technique. The design technique is based on pole-placement in both algorithms presented here. Full PID, as observed before, should perform better than just PD control.

5.2.1 An Adaptive PD Control Algorithm

The algorithm presented here is derived from Chapter 4. Pole-Zero cancellation is employed [31]. A second order model is identified for the input/output data. The identification results in four model parameter estimates. No additional prbs input is required to ensure successful results

5.2.1.1 Controller Derivation

From Chapter 4, the transfer function for a PD controller is

$$K(z) = (K_p h + K_d) \frac{[z - K_d/(K_p h + K_d)]}{h z} \tag{5 23}$$

The identified model is

$$\frac{Y(z)}{U(z)} = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0} = \frac{b_1 z + b_0}{(z - p_1)(z - p_2)} \tag{5 24}$$

where $p_1 \approx 1$ and

$$p_2 = \frac{-a_1 + (a_1{}^2 - 4a_0)^{\frac{1}{2}}}{2} \tag{5 25}$$

Cancelling p_2 gives

$$\frac{K_d}{K_p h + K_d} = p_2 \tag{5 26}$$

and

$$\frac{K_d}{K_p} = \frac{p_2 h}{1 - p_2} \tag{5 27}$$

An extra design requirement is needed to determine the control gains uniquely One can use either phase margin or an error specification as the extra design criterion. Proceeding as in Chapter 4, specify K_v to find K_p and K_d From the forward transfer function, K_v is found to be

$$K_v = \frac{(b_1 + b_0).(K_p.h + K_d)}{h} \tag{5 28}$$

and thus

$$K_d = p_2 \frac{K_v h}{b_1 + b_0} \qquad (5.29)$$

5.2.1.2 Simulation Results

If $K_v = 1/5$, for constant setpoints, the response is shown in Fig 5.4 The response is good, and the static error is low,

$$e_{ss_1} \approx 0$$
$$e_{ss_2} = 2.1 \times 10^{-4}$$
$$e_{ss_3} = 2 \times 10^{-5}$$

This is a considerable improvement on the fixed gain PD controller Fig 5.5 shows the response to a variable reference input The peak error for each joint is

$$e_{pk_1} = 0.06$$
$$e_{pk_2} = 0.06$$
$$e_{pk_3} = 0.08$$

and again these figures are lower than in the fixed parameter case. Hence the adaptive algorithm is a more efficient control algorithm

5.2.2 Full Adaptive PID Control

The fixed parameter PID algorithm in Chapter 4 is tuned using the Zeiger-Nicholas Ultimate Sensitivity Method Here, a pole placement technique is used to compute the controller parameters [32] In this algorithm a training, or learning period, is used, in which the robot parameters are identified, so that good initial estimates are obtained when control commences

5.2.2.1 Controller Derivation

Consider a single input single output, discrete, time invariant second order model for each robot joint

$$A(z^{-1})Y(z) = z^{-1}B(z^{-1})U(z) \qquad (5.30)$$

where

$A(z^{-1}) = 1 + a_1z^{-1} + a_2z^{-2}$ and

$B(z^{-1}) = b_0z^{-1} + b_1z^{-2} \quad b_0 \neq 0$

Consider the following PID structure, given in velocity form

$$S(z)\ U(z) = R(z)\ E(z) \tag{5 31}$$

where

$$S = (1 - z^{-1})\ (1 + s_1z^{-1}) \tag{5 32}$$

$$R = 1 + r_0 + r_1z^{-1} + r_2z^{-2} \tag{5 33}$$

The error e(k) is given by

$$e(k) = u_m(k) - y(k) \tag{5 34}$$

where $u_m(k)$ is the setpoint sequence

Hence the closed-loop system is .

$$(AS + z^{-1}BR)\ Y(z) = z^{-1}BR\ U_m(z) \tag{5 35}$$

The PID has four parameters, and it is possible to select these parameters to fix the closed loop poles To position the closed-loop poles, S and R must satisfy

$$AS + z^{-1}BR = C \tag{5 36}$$

where

$$C = 1 + c_1z^{-1} + c_2z^{-2} + c_3z^{-3} + c_4z^{-4} \tag{5 37}$$

C is the closed-loop pole polynomial chosen by the designer Four simultaneous equations result, and solving these give r_0, r_1, r_2 and s_1

The solution to the design problem is as follows

$$s_1 = \text{num/den} \tag{5.38}$$

$$\text{num} = (c_2 - b_1 - a_1 + a_0) - (c_1 + 1 - a_0 - b_0)\ \frac{b_1}{b_0} - \frac{b_0}{b_1}\ (a_1 + c_3 - b_0c_4/c_1) \tag{5 39}$$

$$\text{den} = -\frac{b_1}{b_0} - \frac{b_0}{b_1}(a_1 - a_0) - \frac{b_0^2}{b_1^2} a_1 + (a_0 - 1) \qquad (5\ 40)$$

$$r_0 = \frac{(c_1+1) - (a_0+b_0+s_1)}{b_0} \qquad (5\ 41)$$

$$r_1 = \frac{c_3 + a_1 - b_0\ (a_1 s_1 + c_4)/b_1 + (a_0 - a_1)\ s_1}{b_1} \qquad (5\ 42)$$

$$r_2 = \frac{c_4 + a_1\ s_1}{b_1} \qquad (5\ 43)$$

5.2.2.2 Simulation Results

The forgetting factor (μ) for each joint is set to 0 995 and P_0 = 1,000 The identification is tuned so that large variations in the controller gains do not occur Large parameter variation causes problems when full PID control is used, due to the sensitivity of the closed-loop system Fig 5 6 shows the simulation results when the above identification tuning is used The C pole polynomial is chosen to have two stable poles Thus c_3 and c_4 are zero The results show that the adaptive algorithm does not perform as well as its fixed parameter counterpart, but this algorithm is still suitable for high precision manipulator tasks The peak error for each joint is

e_{pk1} = 0 006
e_{pk2} = 0 006
e_{pk3} = 0 02

and the static error is very low for each joint.

Using only a self-tuning PID (i e the identification is turned off after the learning period) gives the results in Fig 5 7 The control here is similar to fixed gain PID control

5.2.3 Conclusion

The adaptive PD controller is a significant improvement on its fixed parameter counterpart.The static and peak error values have lower magnitude, therefore justification for the extra algorithmic complexity exists. The control design section is the same for both algorithms.

The adaptive PID performs well, but it is felt that there is no justification for an adaptive PID algorithm based on the results found here Maybe for widely varying loads and high speed movement along specified trajectories, the adaptive algorithm could be suitable, but for industrial use, the extra complexity is not justified Operation over a wide range of conditions (especially load variations encountered in Pick and Place tasks) should show improved response

5.3 Model Reference Adaptive Control (MRAC)

The Model Reference Adaptive System (MRAS) is one of the main approaches to adaptive control The desired performance is expressed in terms of a reference which gives the desired response to a command signal The system also has an ordinary feedback loop composed of the process and the regulator The error is the difference between the outputs of the system and reference model. The regulator has parameters that are changed based on the error There are thus two loops; an outer loop which adjusts the parameters in the inner loop, and an inner loop which provides the ordinary control feedback (see Fig 5 8)

There are essentially three basic approaches to the analysis and design of a MRAS,

1 The Gradient Approach
2. Passivity Theory
3 Lyapunov Functions.

The gradient method is used here It is important to note that the gradient approach will not always result in a stable closed-loop system, but the design is simpler than the other methods mentioned This observation inspired the application of stability theory Lyapunov's stability theory and the Passivity theory have been used to modify the adaptation mechanism

5.3.1 MRAC : The Concept

For a system with adjustable parameters, the model reference adaptive method gives a general approach for adjusting the parameters so that the closed-loop transfer function will be close to the prescribed model. This is called the Model–Following problem. One important question is - how small the error can be made? This depends on the model, the system and the command signal If it is possible to make the error equal to zero for all command signals, then Perfect Model–Following is achieved Optimization methods are natural tools in MRAS design Perfect model-following can only achieved in idealised situations

5.3.2 Controller Derivation

This method is based on the Independent Joint Control Method of Sensitivity Analysis [33] (see Fig 5 9) The manipulator dynamic equation can be written as

$$a_{pi} q_{pi} + b_{pi} q_{pi} + q_{pi} = r_i(t) \qquad (5\ 44)$$

where a_{pi} and b_{pi} are functions of changing coefficients with the operation environments of the system The reference model is given by

$$a_{mi} q_{mi} + b_{mi} q_{mi} + q_{mi} = r_i(t) \qquad (5\ 45)$$

The sensitivity approach is based on adjusting the parameters a_{pi} and b_{pi} in order to minimize a quadratic (or objective) function of the generalized output error,

$$e(t) = q_m(t) - q_p(t) \qquad (5\ 46)$$

Consider the following error function

$$f(e) = \tfrac{1}{2}.\int_0^t (d_0 e + d_1 e + d_2 e\ )^2.dt \qquad (5\ 47)$$

where d_i, $i = 0,1,2$ are the weighting factors Let the parameters a_{pi} and b_{pi} be adjusted in order to minimize this integral This is realized by making small variations in a_{pi} and b_{pi} such that

$$a_{pi}(e,t) = -\alpha_i \ \frac{\partial}{\partial t} \left[\frac{\partial f(e)}{\partial a_{pi}} \right] \qquad (5\ 48)$$

$$b_{pi}(e,t) = -\beta_i \ \frac{\partial}{\partial t} \left[\frac{\partial f(e)}{\partial b_{pi}} \right] \qquad .(5\ 49)$$

Substituting (5 47) into (5 48) and (5 49) using (5 46) gives

$$a_{p1}(e,t) = \alpha_1 \; (d_0 e_1 + d_1 e_1 + d_2 e_1) \; [\; d_0 \frac{\partial q_{p1}}{\partial a_{p1}} + d_1 \frac{\partial q_{p1}}{\partial a_{p1}} + d_2 \frac{\partial q_{p1}}{\partial a_{p1}} \;] \qquad (5\ 50)$$

$$b_{p1}(e,t) = \beta_1 \; (d_0 e_1 + d_1 e_1 + d_2 e_1) \; [\; d_0 \frac{\partial q_{p1}}{\partial b_{p1}} + d_1 \frac{\partial q_{p1}}{\partial b_{p1}} + d_2 \frac{\partial q_{p1}}{\partial b_{p1}} \;] \qquad (5\ 51)$$

where

$$\frac{\partial q_{p1}}{\partial a_{p1}} \quad \text{and} \quad \frac{\partial q_{p1}}{\partial b_{p1}}$$

are the sensitivity functions of the adjustable system with respect to a_{p1} and b_{p1} respectively, and α_1 and β_1 are positive constants known as the adaptation gains.

For slow adaptation (i e a_{p1} and b_{p1} change with slow rate), the adaptation mechanisms reduce to

$$a_{p1}(e,t) = \alpha_1 \; (d_0 e_1 + d_1 e_1 + d_2 e_1) \; [\; d_0 \, u_1 + d_1 \, u_1 + d_2 . u_1 \;] \qquad (5.52)$$

$$b_{p1}(e,t) = \beta_1 \; (d_0 e_1 + d_1 e_1 + d_2 e_1) \; [\; d_0 \, w_1 + d_1 \, w_1 + d_2 \, w_1 \;] \qquad (5\ 53)$$

where

$$u_1 = \frac{\partial q_{p1}}{\partial a_{p1}} \quad \text{and} \quad w_1 = \frac{\partial q_{p1}}{\partial b_{p1}}$$

and the above assumption (slow adaptive rate) results in the following differential equations

$$a_{p1} u_1 + b_{p1} u_1 + u_1 = -q_{p1} \qquad (5.54)$$

$$a_{p1}w_1 + b_{p1}w_1 + w_1 = -q_{p1} \qquad (5\ 55)$$

The rates of adjustment of the control gains can be calculated from (5 56) and (5 57)

$$K_{p1}(e,t) = -a_{p1}(e,t)\ \frac{K_{p1}(e,t)}{a_{p1}(e,t)} \qquad (5\ 56)$$

$$K_{d1}(e,t) = b_{p1}(e,t)\ K_{p1}(e,t) - \frac{a_{p1}(e,t)\ K_{d1}(e,t)}{a_{p1}(e,t)} \qquad (5\ 57)$$

These equations, (5 56) and (5 57), cannot be solved for the sensitivity functions, u_1 and w_1 because the coefficients $a_{p1}(e,t)$ and $b_{p1}(e,t)$ are not available, since they are functions of the unknown coefficients of the controlled plant and the adjustable controller gains Two further assumptions are needed

1 The parametric distances ($\Phi_{a1} = a_{m1} - a_{p1}$, $\Phi_{b1} = b_{m1} - b_{p1}$) are very small, $a_{m1} \approx a_{p1}(e,t)$, $b_{m1} \approx b_{p1}(e,t)$

2 The output generalized error ($e = q_m - q_p$) is small.

Introducing these two supplementary assumptions into equations (5 54) and (5 55) yields a set of differential equations of the form

$$a_{m1}u_1 + b_{m1}u_1 + u_1 = -q_{m1} \qquad (5\ 58)$$

$$a_{m1}w_1 + b_{m1}w_1 + w_1 = -q_{m1} \qquad (5\ 59)$$

and the rates of the adjustment of the control gains are

$$K_{p1}(e,t) = -a_{p1}(e,t)\ \frac{K_{p1}(e,t)}{a_{m1}} \qquad (5\ 60)$$

$$K_{d1}(e,t) = b_{pi}(e,t).K_{p1}(e,t) - \frac{\dot{a}_{p1}(e,t)\ K_{d1}(e,t)}{a_{m1}} \qquad (5\ 61)$$

Hyperstability and Positivity Concept : Popov's hyperstability theory is used to determine the coefficients of the adaptation gain. To formulate the hyperstability problem the generalized error equation has to be derived Subtracting the manipulator dynamic equation (5 44) from the reference model (5 45) results in the generalized error equation given by

$$(a_{mi}\ p^2 + b_{mi}\ p + 1)\ e_i = -w_i \qquad (5\ 62)$$

where $p = d/dt$

and $w_i = (a_{pi} - a_{mi})\ q_{pi} + (b_{pi} - b_{mi})\ q_{pi}$

A linear compensator is introduced to process the generalized state error

$v = D\,e$ where $D = [\ d_0 I + d_1 I\ ]$

The adaptation algorithm can be written as .

$$a_{pi}(v,t) = -\alpha_i\ v_i\ q_{pi} \qquad (5\ 63)$$

$$b_{pi}(v,t) = -\beta_i\ v_i\ q_{pi} \qquad (5\ 64)$$

where $\alpha_i, \beta_i > 0$

From the decoupled equations, the control equation is given by

$$u_i(t) = K_{pi}(t)\ [\ r_i - q_{pi}\ ] - K_{di}(t)\ q_{pi} \qquad (5\ 65)$$

Hence the adaptation mechanism can be written as

$$a_{pi}(v,t) = -\alpha_i\ v_i\ u_i(t) = -\alpha_i\ v_i\ [\ K_{pi}(t)\ (\ r_i - q_{pi}\ ) - K_{di}(t)\ q_{pi}\ ] \qquad (5\ 66)$$

$$b_{pi}(v,t) = -\beta_i\ v_i\ q_{pi} \qquad (5\ 67)$$

The rates of adjustment of the control gains are given by equations (5 60) and (5 61)

$$K_{p1}(e,t) = -a_{p1}(e,t)\,\frac{K_{p1}(e,t)}{a_{m1}}$$

$$K_{d1}(e,t) = b_{p1}(e,t)\,K_{p1}(e,t) - \frac{a_{p1}(e,t)\,K_{d1}(e,t)}{a_{m1}}$$

5.3.3 Results

The optimal initial values of the controller gains are

$K_{p_1} = 500$ $K_{p_2} = 500$ $K_{p_3} = 600$

$K_{d_1} = 30$ $K_{d_2} = 60$ $K_{d_3} = 10$

The position reference model for each joint is

$$\frac{Y(s)}{U(s)} = \frac{200}{(s+10)(s+20)}$$

and for velocity

$$\frac{Y(s)}{U(s)} = \frac{200s}{(s+10)(s+20)}$$

This model has unity dc gain and a fast response to command inputs.

Test 1 :

$\alpha_1,\beta_1 = 1\text{x}10^{-5}$

$d_0 = 1$

$d_1 = 10$

$d_2 = 0$

Fig.5.10 shows the results of this test The setpoints are constant. The result is good with a small steady state error for each joint .

$e_{ss_1} \approx 0$

$e_{ss_2} = 7\,2\text{x}10^{-4}$

$e_{ss_3} = 6x10^{-5}$

Test 2 :

Using the same controller tuning, a variable trajectory is used as the reference input. Fig 5 11 shows the response The peak error for each joint is low

$e_{pk_1} = 0\ 1$
$e_{pk_2} = 0\ 2$
$e_{pk_3} = 0\ 08$

Test 3 :

Use $\alpha_1,\beta_1 = 1x10^{-5}$ The reference models are changed to

$$\frac{Y(s)}{U(s)} = \frac{600}{(s+30)(s+20)}$$

and

$$\frac{Y(s)}{U(s)} = \frac{600s}{(s+30)(s+20)}$$

Again, a variable trajectory is used as the reference input. Fig.5 12 shows the response. The peak error for each joint is low

$e_{pk_1} = 0\ 1$
$e_{pk_2} = 0\ 15$
$e_{pk_3} = 0\ 07$

This is a good response, but not the best result achieved to date

5.3.4 Conclusion on MRAC

This version of MRAC is based on the MIT rule, known more specifically as MRAC - the Hyperstability Method. It is basically an adaptive PD controller with certain properties. More complicated versions are available which produce better results, i e. The Sensitivity Approach.

One has to ask the question whether one can justify the extra computation required to update the controller gains The constants d_0, d_1 and d_2 are quoted from [33]

The control parameters change when the position is varying When the position reaches a constant value, the parameters stabilize Maximum parameter variations occur when the joint follows the specified path to a new setpoint Larger values of alpha and beta result in larger changes in the parameters

5 4 The Self-Tuning Regulator (STR)

Using pole placement an Adaptive Regulator can be designed The design tries to fit the closed-loop system to a specified reference system by correct choice of the controller parameters The process involves solving a Diophantine equation The order of the controller parameters is also specified by design regulations [34]

Two different types of controller are possible, *explicit* and *implicit* types. The results in the end of the chapter compare the performances of both types. The controller structure is very similar to a PID controller in the design stage (see Fig 5 13)

5.4.1 The Explicit Method

Explicit adaptive control incorporates a design stage after the identification stage is complete. The identification produces estimates for the plant parameters, and the control design stage transforms these plant parameters using whatever design technique is chosen by the designer

5.4.1.1 Controller Derivation

From Fig 5 13 the control equation is given by

$$R(z)U(z) = T(z)U_C(z) - S(z)Y(z) \qquad (5.68)$$

and the closed loop system transfer function is

$$\frac{B_m}{A_m} = \frac{BT}{AR + BS} \qquad (5\ 69)$$

where B_m and A_m are the specified closed-loop polynomials, determined by the designer R(z) is assumed to be monic Equation (5 69) can be solved for the three unknowns giving,

$$B_m = BT \qquad (5\ 70)$$

$$A_m = AR + BS \qquad (5\ 71)$$

Equation (5 71) is a ***Diophantine equation.***

The polynomial B can be divided into two polynomials B^- contains the unstable plant zeros, and B^+ contains the stable zeros

$$\Rightarrow \quad B = B^- B^+ \qquad (5\ 72)$$

Unstable zeros cannot be cancelled.

The solution to the design problem is given as follows

$$T = B_m^* \qquad (5\ 73)$$

$$A_m = AR^* + B^-S \qquad (5\ 74)$$

$$\deg S = \deg A - 1 \qquad (5\ 75)$$

$$\deg R = \deg A_m - \deg A \qquad (5\ 76)$$

$$\deg A_m - \deg B_m \geqslant \deg A - \deg B \qquad (5\ 77)$$

The general procedure is as follows [34]

1 Select A_m and B_m subject to equation (5 77)
2 $B = B^-B^+$ and $B_m = B^-B_m$
3 Solve $AR^* + B^-S = A_m$
4 Find $R = B^+R^*$ and $T = B_m^*$
5 The control law is derived as

$$RU = TU_c - SY \qquad (5\ 78)$$

Applying this solution to the robot, results in the following controller The robot is identified as a second order model

$$\frac{Y(z)}{U(z)} = \frac{K\ (z - b)}{(z - c)\ (z - a)} \tag{5 79}$$

Choosing the reference model $H_m(z)$ as

$$\frac{Y(z)}{U_c(z)} = \frac{z(1 + p_1 + p_2)}{z^2 + p_1 z + p_2} \tag{5 80}$$

gives unity dc gain

$$B = B^+ B^- = (z - b)\ K \tag{5 81}$$

and

$$B_m^* = \frac{B_m}{K} = \frac{z\ (1 + p_1 + p_2)}{K} \tag{5 82}$$

Also,

$$\text{deg}S = \text{deg}A - 1 = 1$$

$$\text{deg}R^* = \text{deg}A_m - \text{deg}A = 0$$

The Diophantine equation,

$$AR^* + B^-S = A_m$$

reduces to

$$(z - c)\ (z - a)\ r_0 + K\ (s_0\ z + s_1) = z^2 + p_1 z + p_2 \tag{5 83}$$

Comparing coefficients gives the control parameter solutions :

$$r_0 = 1 \quad , \quad s_0 = \frac{p_1 + a + c}{K}$$

$$s_1 = \frac{p_2 - a\,c}{K} \quad , \quad R = B^+ R^* = z - b$$

$$T = B_m^* = \frac{z\,(1 + p_1 + p_2)}{K}$$

The controller difference equation is

$$u(k+1) = b\,u(k) + (1 + p_1 + p_2)\,u_c(k+1) - s_0 y(k+1) - s_1 y(k) \tag{5 84}$$

The basic algorithmic procedure is as follows

1 Estimate K, a, b and c

The identification is configured as follows

$$[\,y(k+2)\,] = [\,-y(k+1) \quad -y(k) \quad u(k+1) \quad u(k)\,]\ \theta^T$$

where

$$\theta^T = [\ -(a+c) \quad a\,c \quad K \quad -K\,b\ ]$$

2 Determine B^+, B^- and B_m^*

3 Update R, S and T

4 Compute the present control input u(k+1)

5.4.1.2 Simulation Results

The reference model parameters are assigned the following values

p_1 = -1 4

p_2 = 0 49

giving the following transfer function

$$H_{m1}(z) = \frac{z(0\ 09)}{(z - 0\ 7)^2}$$

Fig 5 14 shows the response to a varying reference input. The peak error is low for each joint

$$e_{pk_1} = 0\ 027$$
$$e_{pk_2} = 0\ 027$$
$$e_{pk_3} = 0\ 043$$

If a faster reference model is used

$$H_{m_2}(z) = \frac{z(0\ 36)}{(z - 0\ 4)^2}$$

the peak error for each joint is reduced to

$$e_{pk_1} = 0\ 006$$
$$e_{pk_2} = 0\ 008$$
$$e_{pk_3} = 0\ 012$$

which is extremely low (see Fig 5 15 for this result)

5.4.2 An Implicit STR

The idea in this section is to rewrite the process model in such a way that the control design step is no longer needed, i.e. the identification now estimates the controller parameters not the process parameters now By a proper choice of model structure, the regulator parameters are updated directly and the design calculations are thus eliminated. Implicit can also be called a *direct* method because the parameters of the regulator are updated directly

5.4.2.1 Controller Derivation

Recall equation (5 71), the Diophantine equation,

$$A_m = AR + BS$$

also $T = B_m{}^*$

$$AR^*Y + B^-SY = A_mY \qquad (5\ 85)$$

Since AY = BU, then

$$BR^{*}U + B^{-}SY = A_mY \qquad (5.86)$$

Equation (5.86) can be used as an identification model if, and only if, $B^{-}=1$. If $B^{-}=1$ then $BR^{*} = R$ and

$$RU + SY = A_mY \qquad (5.87)$$

Also $T = B_m$

Applying this to the robot

$$R = r_0 z + r_1$$
$$S = s_0 z + s_1$$
$$A_m = z^2 + p_1 z + p_2$$
$$T = B_m = z(1 + p_1 + p_2) = z t_0$$

From equation (5.87)

$$r_0\, u(k+1) + r_1\, u(k) + s_0\, y(k+1) + s_1\, y(k) = y(k+2) + p_1\, y(k+1) + p_2\, y(k) \qquad (5.88)$$

Equation (5.88) can be used to identify the controller parameters

$$y = \Phi\,\theta^T$$

$$y = [\; y(k+2) + p_1\, y(k+1) + p_2\, y(k)\,]$$

$$\Phi = [\, y(k+1) \quad y(k) \quad u(k+1) \quad u(k)\,]$$

and

$$\theta^T = [\, s_0 \quad s_1 \quad r_0 \quad r_1\,]$$

and

$$t_0 = 1 + p_1 + p_2$$

The control law is

$$u(k+1) = [-r_1\ u(k) + (1+p_1+p_2)\ u_C(k+1) - s_0\ y(k+1) - s_1\ y(k)]/r_0 \quad (5\ 89)$$

The implicit control algorithm is as follows

1 Update θ (the controller parameters)
2 Update the control input u(k+1)

5 4.2.2 Simulation Results

Use $H_{m1}(z)$ as the reference model Fig 5 16 shows the results with a varying reference input. The peak error for each joint is

$e_{pk1} = 0\ 1$
$e_{pk2} = 0\ 1$
$e_{pk3} = 0\ 013$

If $H_{m2}(z)$ is used, joint 2 does not follow the specified path. The results here are good, but not as good as the explicit algorithm

5.4.3 Conclusion

From the results obtained, the explicit algorithm behaves in a more robust fashion It can control the robot joints even when the reference model is made extremely fast.

The parameters identified in both cases are different. The explicit algorithm identifies the process model, but the implicit type identifies the controller parameters Hence implicit is simpler in nature

The reference model can be adjusted so as to reduce the tracking error One model gave an error as low as the full PID controller Since this algorithm is adaptive one would expect better results in the case of varying payloads.

5.5 Adaptive Predictive Control

In Chapter 4, Predictive Control shows some encouraging results. It is hoped that by introducing an identification routine, thus implementing an adaptive algorithm, the performances of the two previous algorithms improve sufficiently to justify the use of adaptive strategies

Predictive Control can be implemented as a Gain-Scheduled Algorithm by varying the tuning parameters, and the gains, over a range of operating points Note that this type of adaptive algorithm contains no parameter estimation technique Also, to implement *full* adaptive control, the parameter estimates are entered in the existing control routine to determine the controller gains In these sections various adaptive forms are investigated, these routines are explicit adaptive control algorithms, i e a control design stage is not redundant

5.5.1 The Adaptive Monoreg Algorithm

Incorporating Recursive Least Squares into the Monoreg routine introduces adaptability into the function The internal model used is a state-space model in observable form In this way, the second order model has two states, one of which is joint position The identified model can be written in observable form simply by entering the correct term directly into the matrices.

5.5.1.1 Controller Derivation

The robot is identified as a second order model in transfer function form ·

$$H_1(z) = \frac{b_1\ z + b_0}{z^2 + a_1\ z + a_0}$$

The internal model is

$$A = \begin{bmatrix} -a_1 & 1 \\ -a_0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_0 \end{bmatrix}$$

$$C = [\ 1 \quad 0\ ]$$

These model values are entered into the on-line internal model and the control equation from Chapter 4 is used, i e the manipulated variable is calculated by

$$u(k) = \frac{(1-\alpha^H)\ (C_p - S_0(k)) - C\ (A^H - I)\ x(k)}{P} \qquad (5\ 90)$$

The term $(A^H\text{-}I)$ is no longer computed off-line before the algorithm is initiated, but must be computed at each sample instant due to A changing This increases the computational burden

5.5 1.2 Simulation Results

Using α_1=0 85 and H_1,H_2=10 and H_3 = 5, the response in Fig 5 17 is achieved The peak error for each joint is low

e_{pk1} = 0 016
e_{pk2} = 0 014
e_{pk3} = 0 022

and the algorithm performs well. However reducing α_1 to 0.1 gives a further improvement. Fig 5 18 shows the response The peak error for each joint is reduced to

e_{pk1} = 0 01
e_{pk2} = 0 01
e_{pk3} = 0 018

but some oscillation is present in the voltage signal, and may be undesirable depending on the application

5.5.2 A Gain-Scheduled Predictive Controller

The Monoreg Control Algorithm can be implemented as a Gain-Scheduled Algorithm The tuning parameters α and H are varied over the operating range of the robot arms. Since the parameters of the robot vary widely as they transgress a specified trajectory, gain-scheduling is employed so as to retune the compensator to the varying parameters.

5.5.2.1 The Concept of Gain-Scheduling

For the robot to react to fast changes in the reference signal, it is proposed to choose alpha as a small value in its allowable range, and to have a short horizon value Then the term $(1-\alpha^H)$ makes a large contribution to the control action As the robot reaches the end of the specified trajectory, the tuning parameters are set for slow movement of the joints i e the setdown point has been reached

Table 5 1 shows the variation in the peak error of joint 1 when different tuning parameters are used

Table 5.1 A Gain-Scheduled Test

α	H	Peak Error	
0 9	3	0 052	
0 9	20	0 08	Slowest
0 1	20	0 072	
0 1	10	0 038	
0 1	3	0 01	Fastest

The fastest parameters give the lowest error values, and the slowest parameters give the largest values in peak error Table 5 2 shows the gain-scheduled tuning parameters, and how they are varied over the reference signal. The fastest parameters are used in the beginning to ensure a low peak error for each joint, and these parameters are continually changed to slow the response as the desired position is reached, for low static error

Table 5.2 The Tuning Parameter Variations

% Trajectory Time	α	H
First 32%	0 1	3
Next 18%	0 25	5
Next 14%	0 4	7
Next 16%	0 6	10
Next 16%	0 75	15
Final 4%	0 9	20

5.5.2.2 Simulation Results

Using the parameter variation according to Table 5 2, the results are shown in Fig 5 19 The peak error for each joint is

$e_{pk_1} = 0\ 01$
$e_{pk_2} = 0\ 01$
$e_{pk_3} = 0\ 018$

This is an improvement on the simple controller in Chapter 4

5.5.3 Adaptive Output Feedback Control

The second Predictive Control method can also be transformed into an adaptive routine The internal model is derived directly from a transfer function representation of the model The fixed parameter model did not prove suitable for manipulator use Investigation is now performed in this section to determine whether the adaptive version is suitable

5.5.3.1 Controller Derivation

The controller equation is as before

$$u(k) = \{ y(k+4) + b_1\ y(k+3) + b_2\ y(k+2) + b_3\ y(k+1) \}/(a_1+a_2+a_3) \qquad (5\ 91)$$

The parameters from the identification can be entered directly into this equation The algorithm is very simple, even though it is adaptive

5.5.3.2 Simulation Results

From Chapter 4, one found that a value for α_1=0 97, or greater, is required, otherwise undesirable results are obtained Fig 5 20 shows the response The peak error for each joint is

$e_{pk_1} = 0\ 51$
$e_{pk_2} = 0.53$
$e_{pk_3} = 0\ 22$

This is far from optimal performance If α_1 is reduced, a large static error results on joint 2

5.5.4 Conclusion

The adaptive Monoreg Algorithm is a more complex algorithm than its fixed parameter version in Chapter 4 The identification algorithm increases the complexity of the routine Also, the term $(A^H\text{-}I)$ must be computed at each sample interval because the A matrix is identified at each sample interval, and its value is constantly changing If a long horizon is used, the algorithm takes a considerable amount of processor time for the matrix manipulation The performance of this algorithm is better than the fixed case The peak errors have been reduced from 0 3, 0 3 and 0 5 to 0 01, 0 01 and 0 08 for joints 1, 2 and 3 respectively This is indeed a significant reduction in error, thus justification exists for use of the more complicated algorithm

One reason for using a Gain-Scheduled controller is that sudden changes in desired setpoint can be accommodated by a highly sensitive controller, which can take fast compensating action. When the controller returns the process output to the setpoint, low gains can be switched-in for safe operation

The adaptive output feedback controller does not exhibit the desired features of a high precision control algorithm Therefore, it is not very suitable for manipulator use.

5.6 Summary

This chapter investigates a wide range of adaptive control algorithms. The area of Adaptive Control is a huge area of research, so a literature survey was required to determine the areas of interest The algorithms chosen here deemed to be a fair representation of the facilities in this area of control

Adaptive Digital Controllers are an obvious choice The performance of adaptive PD control is encouraging This routine outperforms its fixed parameter counterpart. Full adaptive PID control does not justify its complexity, based on the results here Possibly, in a varying payload situation, the adaptive routine might excel.

Model Reference Adaptive Control is another major section of adaptive control The method used here is based on the Hyperstability Approach where two feedback gains are updated using an adaptation mechanism derived from the error and its derivatives This type of MRAC does not use an identification algorithm The results here also prove good, even though the controller is basically a PD controller

A pole placement algorithm (STR) is also used here Two types of this algorithm are possible - Explicit and Implicit versions The algorithm performs better when used in Explicit form The design procedure requires the solution of a Diophantine Equation The algorithm has a feedback and a feedforward section.

Finally, Adaptive Predictive Control is employed A Gain-Scheduled routine is used to control the robot simulator It performs with a high degree of accuracy Two other methods are also used An Adaptive Monoreg Algorithm and an Adaptive Output Feedback Algorithm from Chapter 4 are investigated The latter is deemed not suitable for use here The Monoreg algorithm is suitable, and is one of the best algorithms to date, but the STR (explicit type) performs with the greatest degree of accuracy in all the tests, and is chosen as the number one adaptive algorithm.

Index to Graphs

Identification Results

Adaptive PD Control Results

Adaptive PID Control Results

- Fig 5 6b *Plot of Identified Numerator Parameters versus Time*

- Fig 5 6c *Plot of Identified Denominator Parameters versus Time*

- Fig 5 6d *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

- Fig 5 7a *Plot of Joint Positions versus Time for Cubic Spline Tracjectory*

- Fig 5 7b *Plot of Identified Numerator Parameters versus Time*

- Fig 5 7c *Plot of Identified Denominator Parameters versus Time*

- Fig 5 7d *Plot of Joint Positional Errors versus Time for Cubic Spline Tracjectory Demands*

Model Reference Control Results

- Fig 5 10a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory*

- Fig 5 10b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory*

- Fig 5 10c *Plot of Identified Proportional Gains versus Time*

- Fig 5 10d *Plot of Identified Derivative Gains versus Time*

- Fig 5 11a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory*

- Fig 5 11b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory*

- Fig 5 11c *Plot of Identified Proportional Gains versus Time*

- Fig 5 11d *Plot of Identified Derivative Gains versus Time*

- Fig.5.12a *Plot of Control Inputs versus Time for Cubic Spline Tracjectory*

- Fig 5.12b *Plot of Joint Positions versus Time for Cubic Spline Tracjectory*

- Fig 5 12c *Plot of Identified Proportional Gains versus Time*

Adaptive Predictive Control Results (Gain-Scheduled Algorithm)

Adaptive Predictive Control Results (Output-Feedback Algorithm)

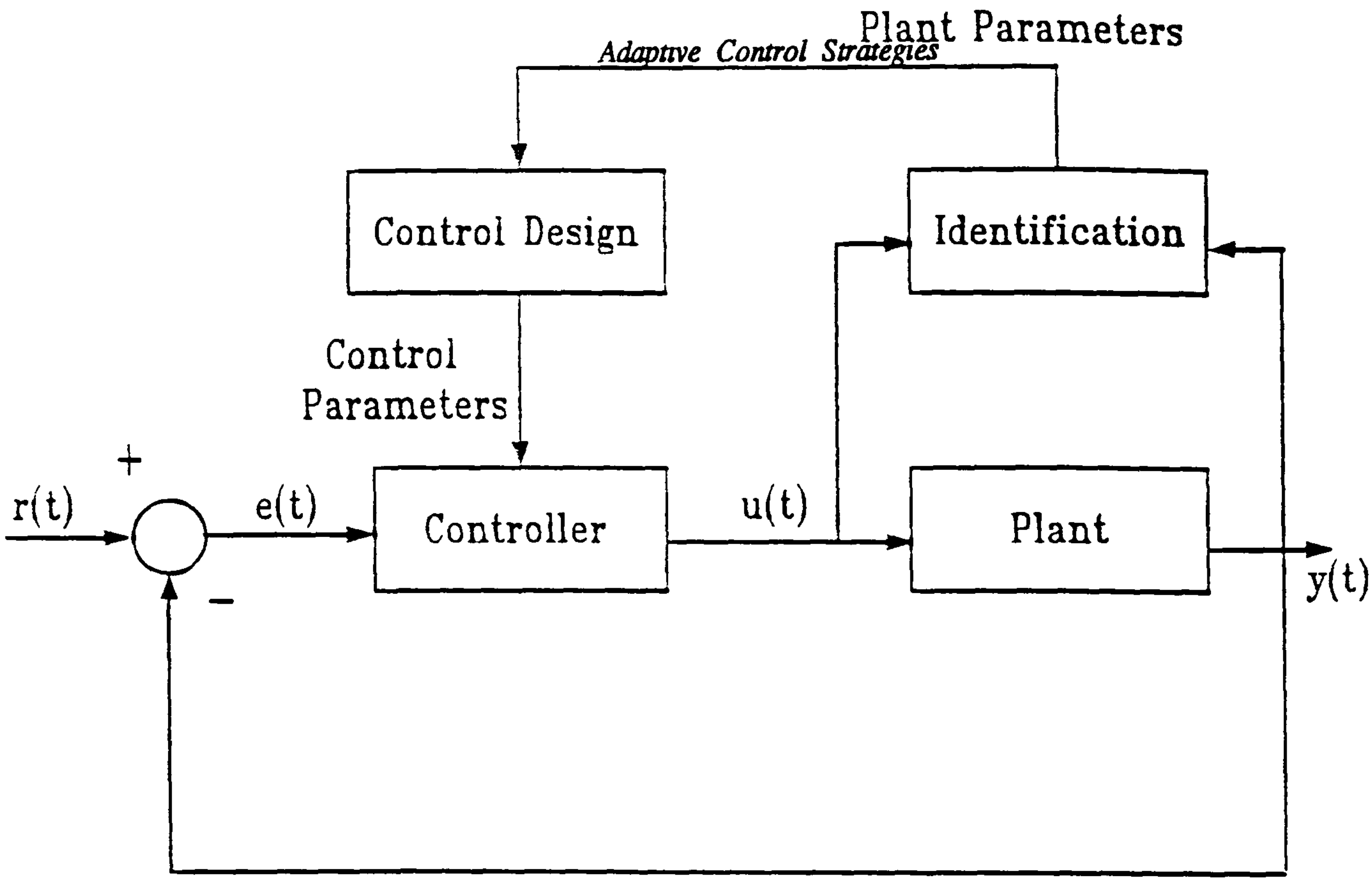
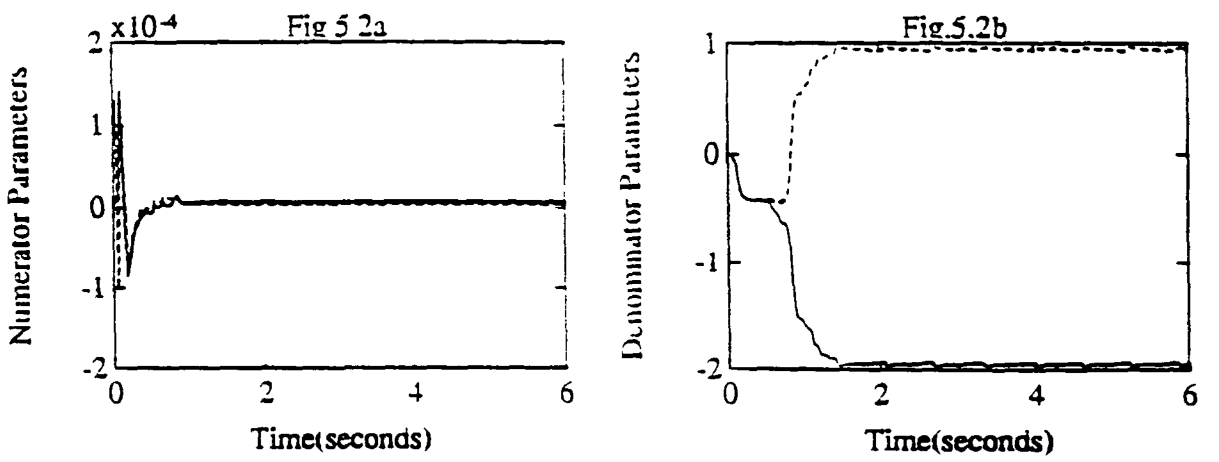
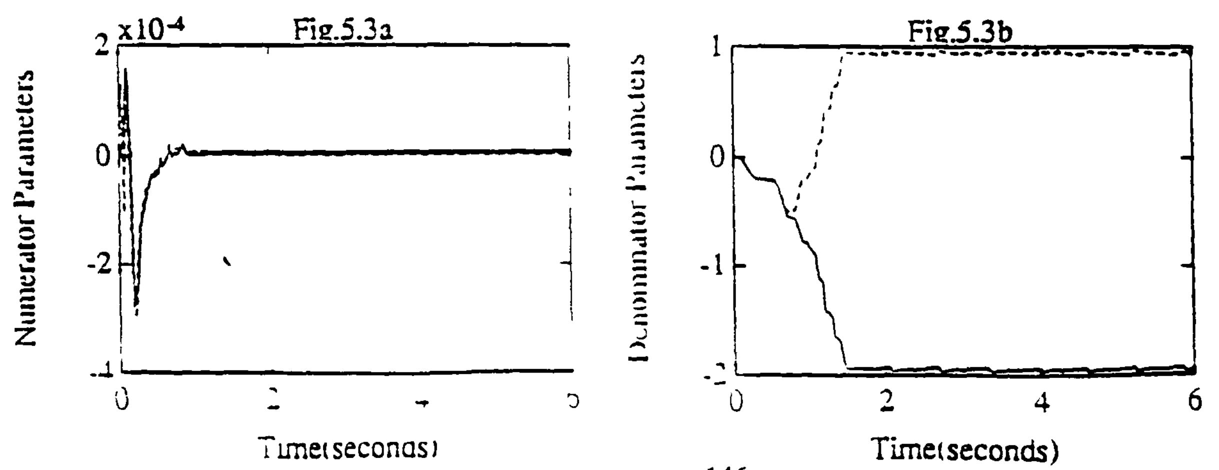


Fig.5.1 The Strategy of Adaptive Control





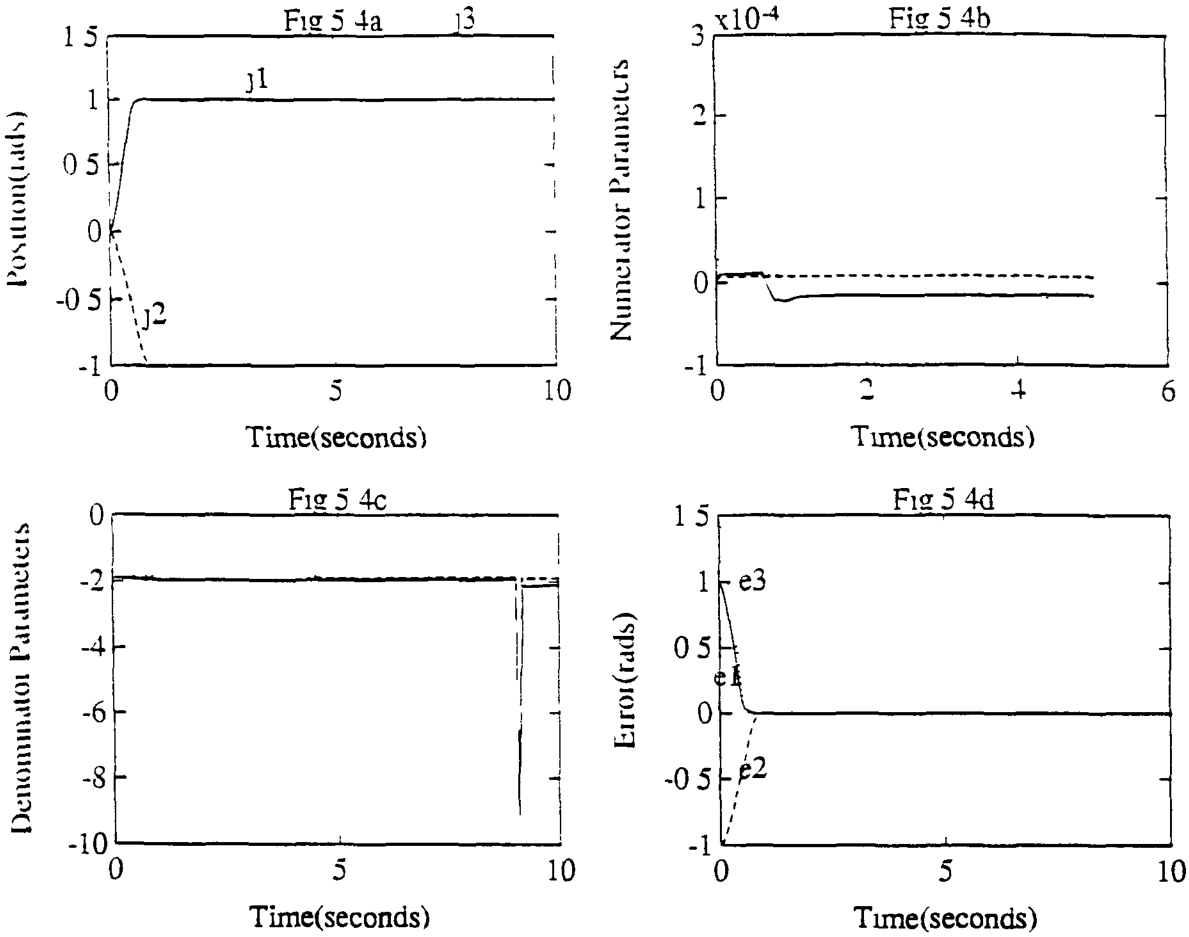


Fig 5 4a
Position(rads)
Time(seconds)
j1
j2
j3
Fig 5 4b
x10-4
Numerator Parameters
Time(seconds)
Fig 5 4c
Denominator Parameters
Time(seconds)
Fig 5 4d
Error(rads)
Time(seconds)
e1
e2
e3

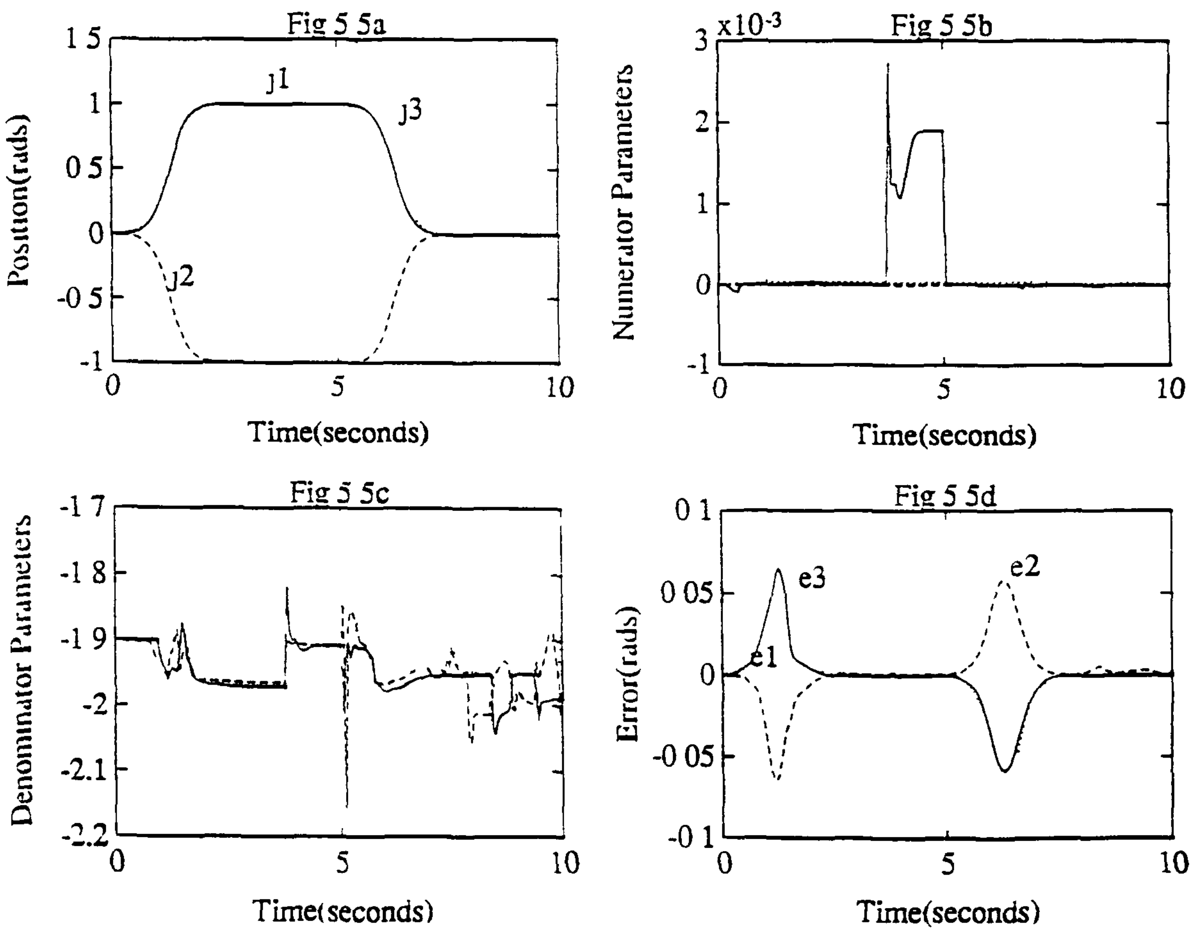


Fig 5 5a
Position(rads)
Time(seconds)
j1
j2
j3
Fig 5 5b
x10-3
Numerator Parameters
Time(seconds)
Fig 5 5c
Denominator Parameters
Time(seconds)
Fig 5 5d
Error(rads)
Time(seconds)
e1
e2
e3

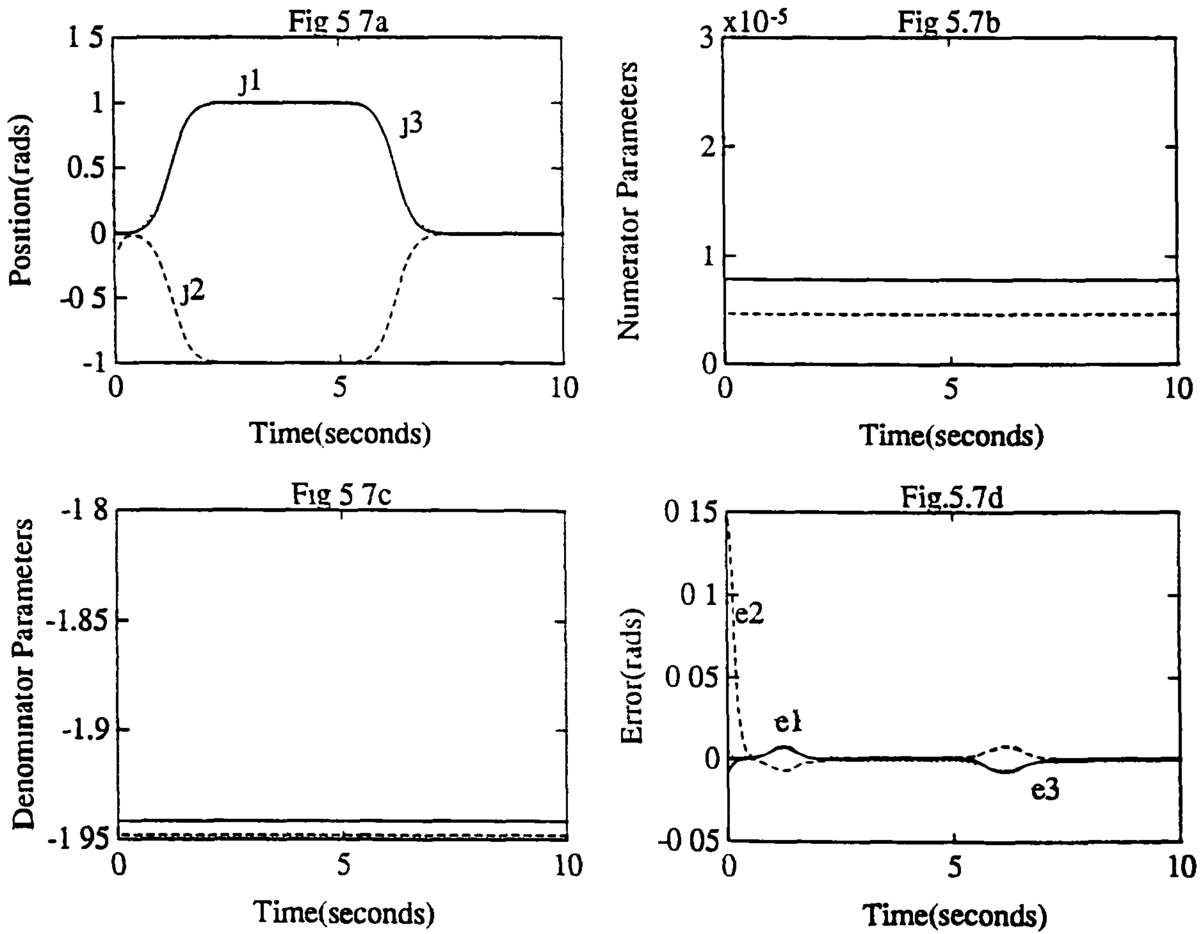


Fig 5 7a
Position(rads)
1 5
1
0.5
0
-0 5
-1
0
5
10
j1
j2
j3
Time(seconds)
Fig 5.7b
x10-5
Numerator Parameters
3
2
1
0
0
5
10
Time(seconds)
Fig 5 7c
Denominator Parameters
-1 8
-1.85
-1.9
-1 95
0
5
10
Time(seconds)
Fig.5.7d
Error(rads)
0 15
0 1
0 05
0
-0 05
0
5
10
e1
e2
e3
Time(seconds)

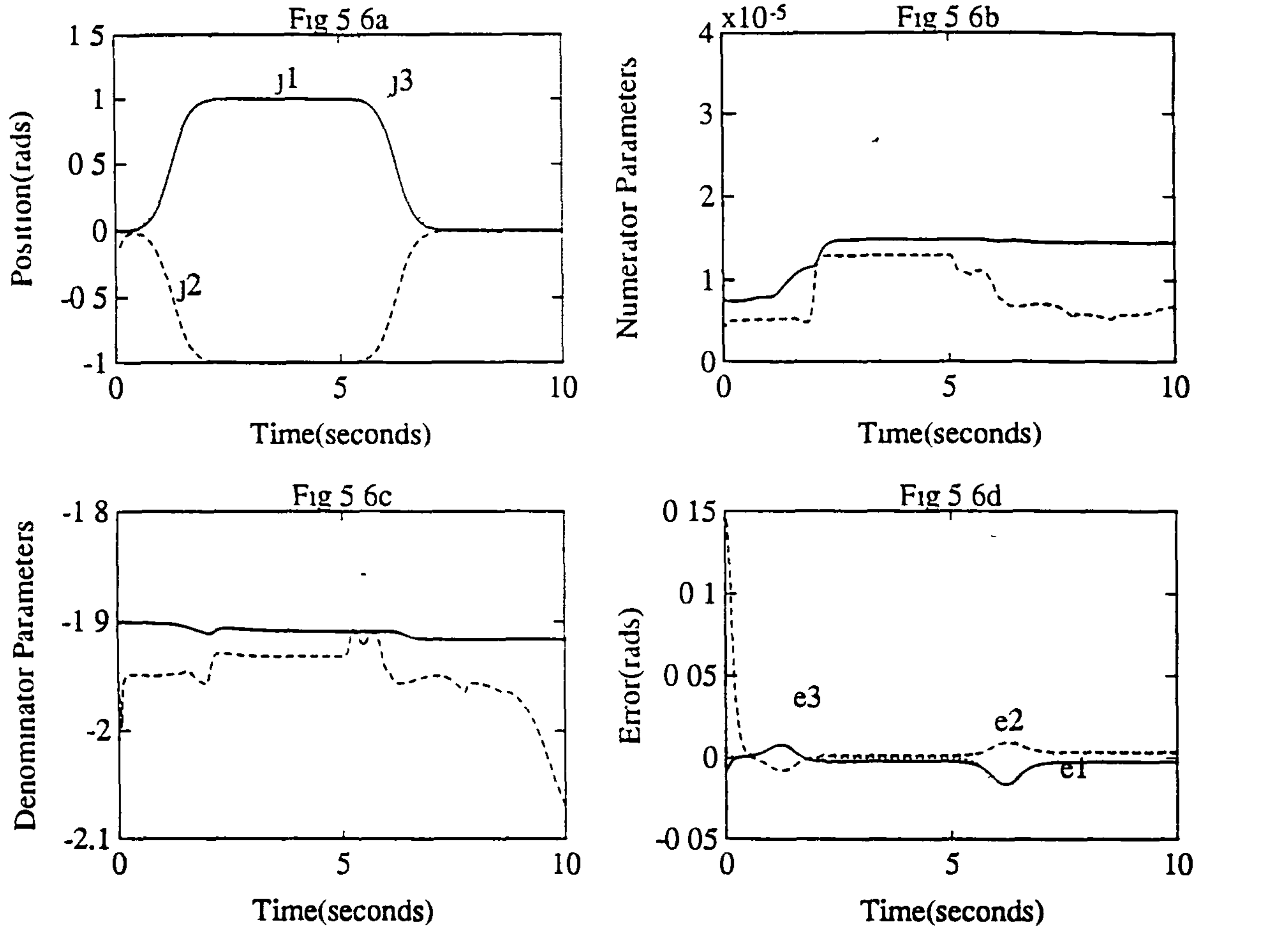


Fig 5 6a
Position(rads)
1 5
1
0 5
0
-0 5
-1
j1
j2
j3
0
5
10
Time(seconds)
Fig 5 6b
x10^-5
Numerator Parameters
4
3
2
1
0
0
5
10
Time(seconds)
Fig 5 6c
Denominator Parameters
-1 8
-1 9
-2
-2.1
0
5
10
Time(seconds)
Fig 5 6d
Error(rads)
0 15
0 1
0 05
0
-0 05
e1
e2
e3
0
5
10
Time(seconds)

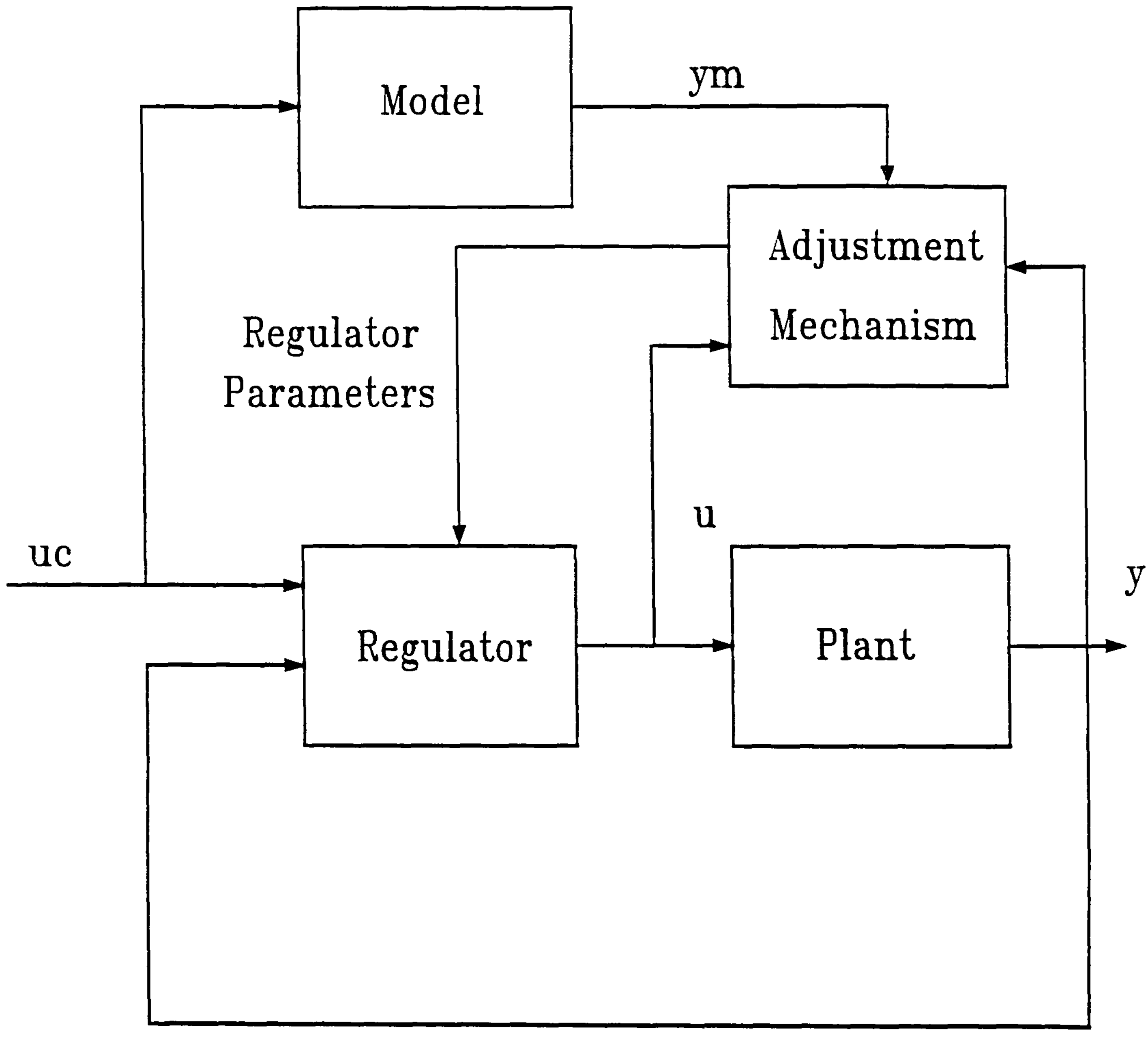


Fig.5.8 The Model Reference Control Strategy

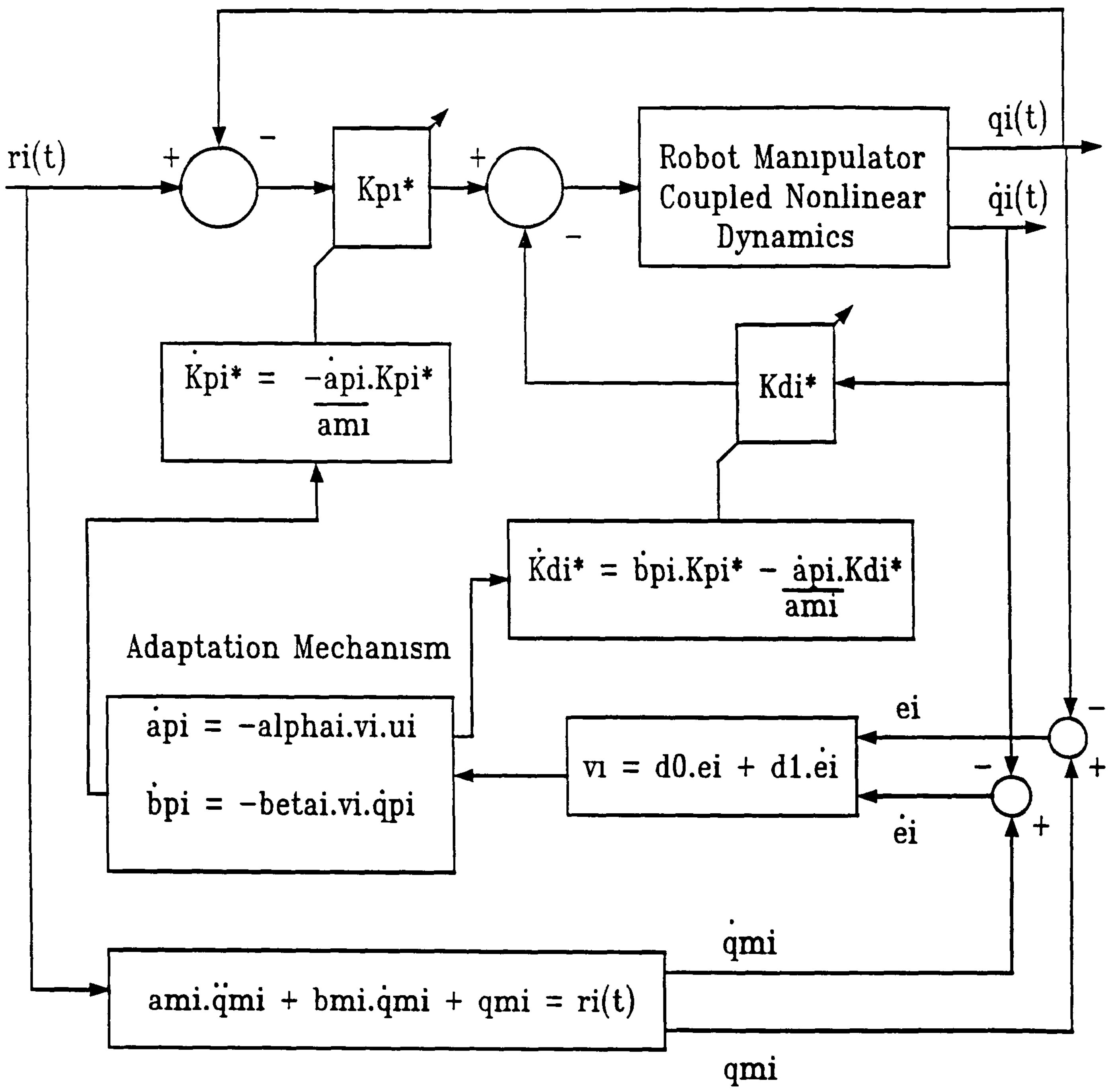


Fig.5.9 MRAC - The Hyperstability Approach

Fig 5 10a

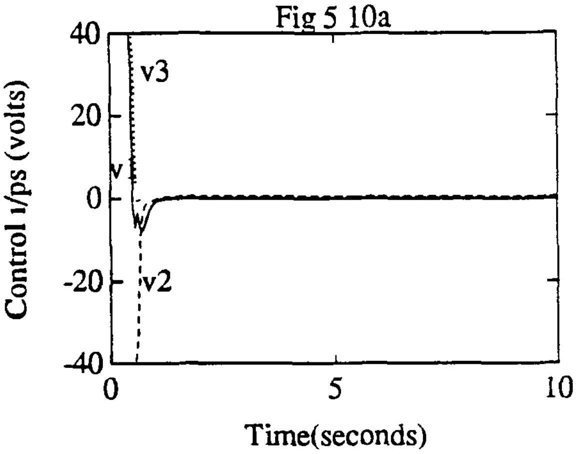


Fig 5 10b

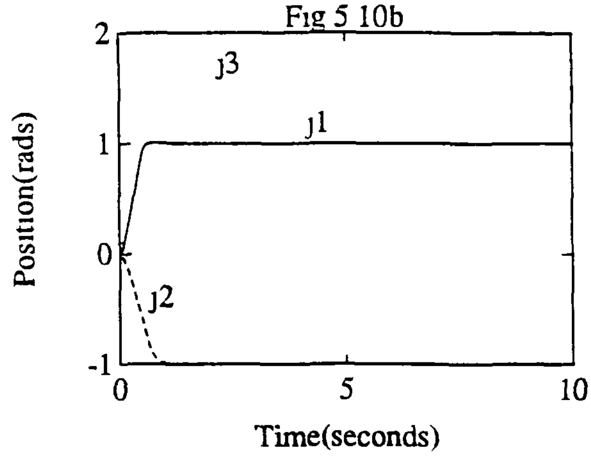


Fig 5 10c

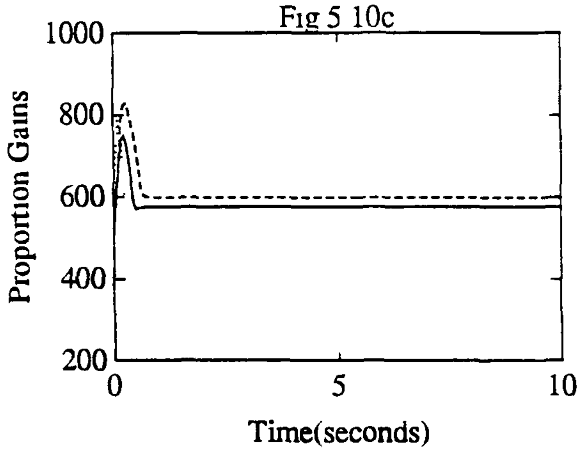


Fig 5 10d

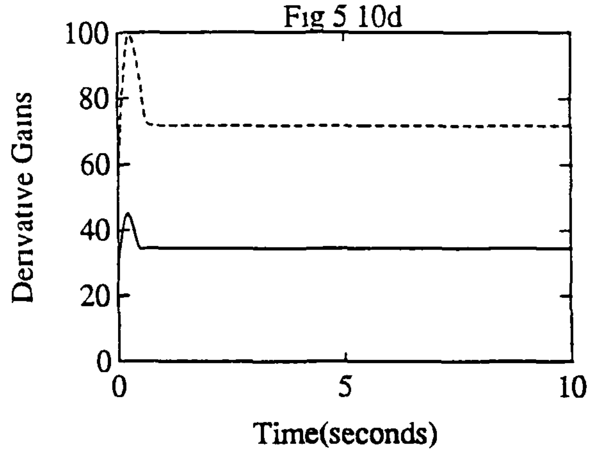


Fig 5.11a

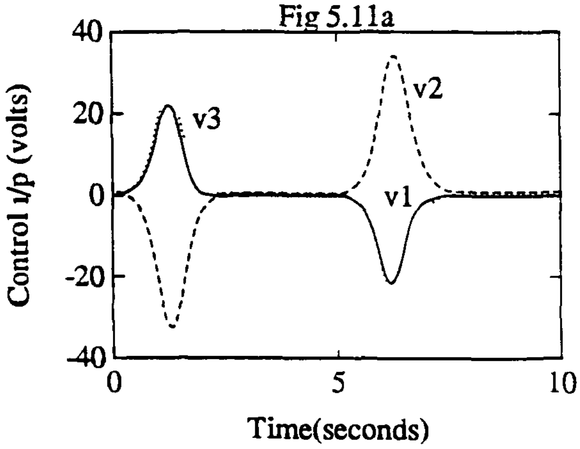


Fig.5 11b

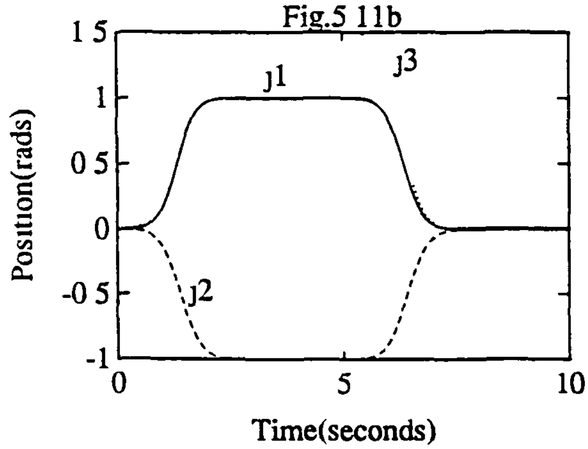


Fig 5 11c

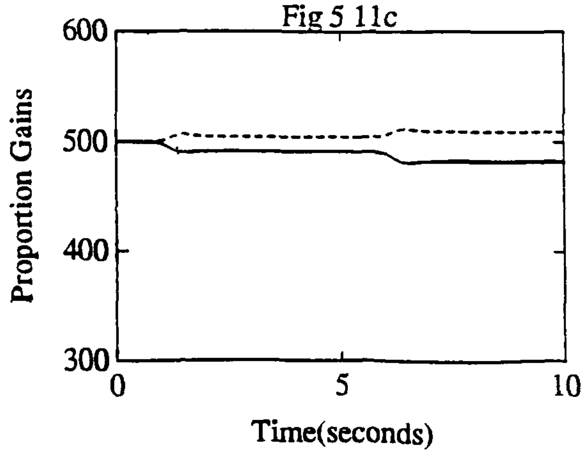
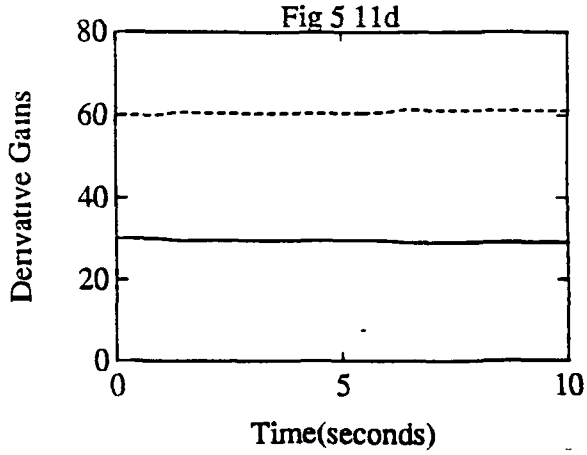
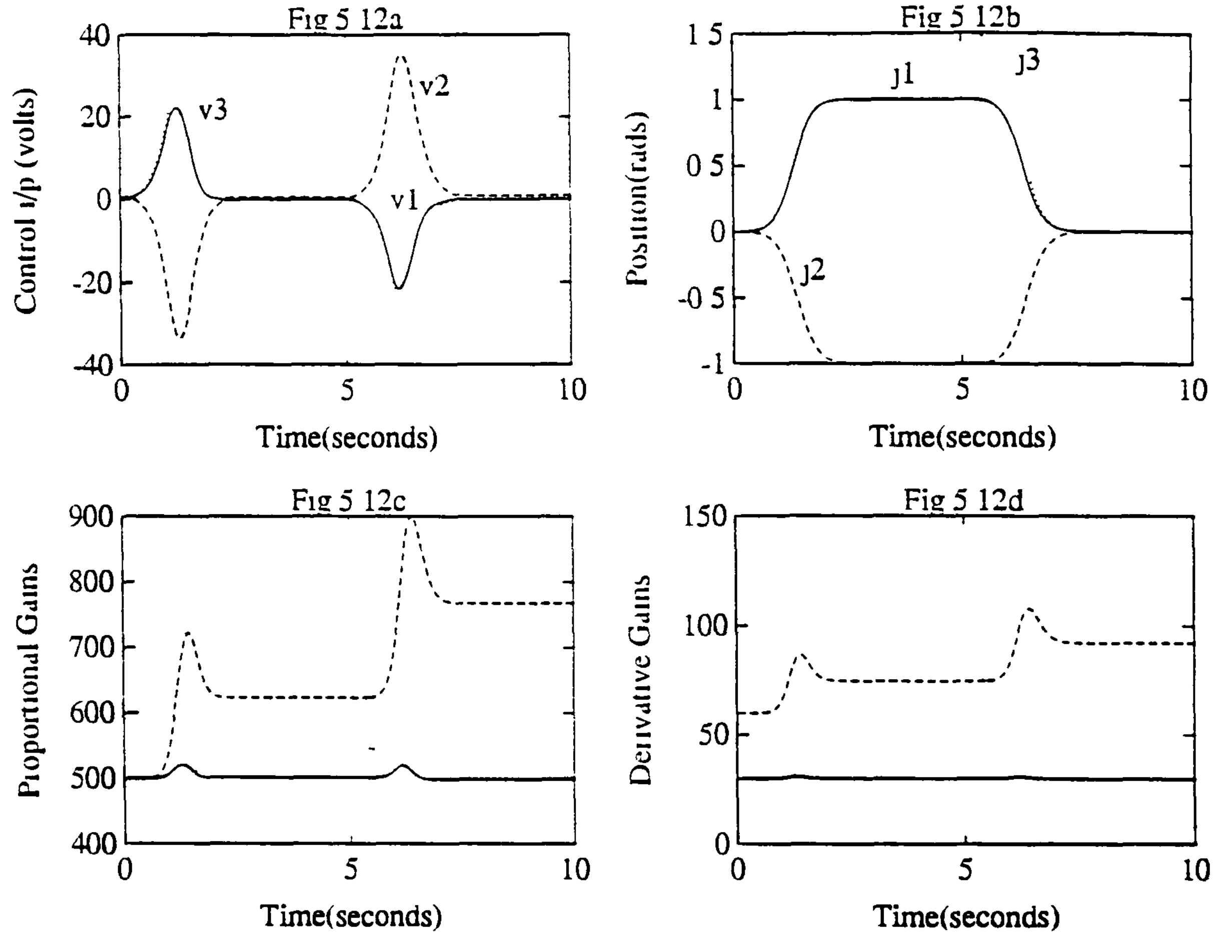


Fig 5 11d





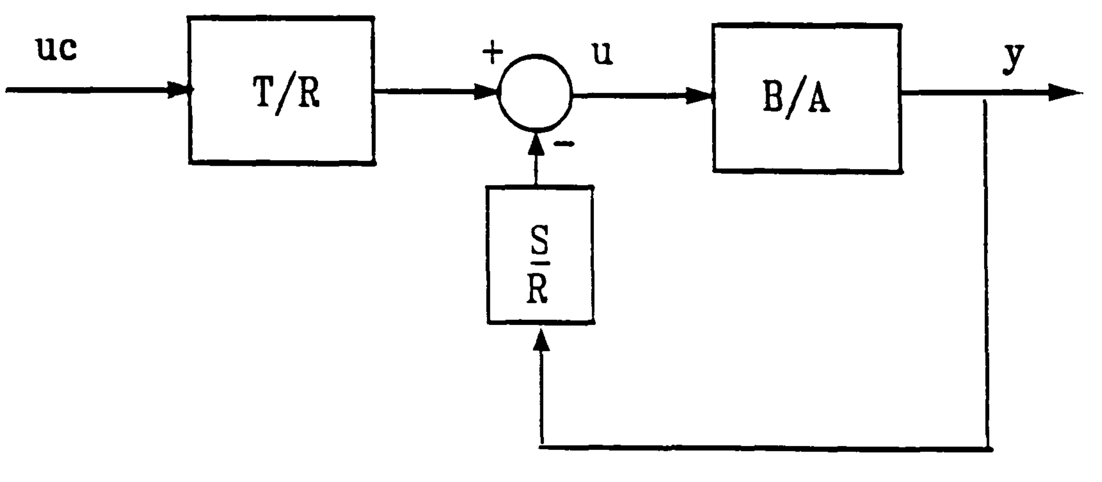


Fig.5.13 A General Pole Placement Scheme

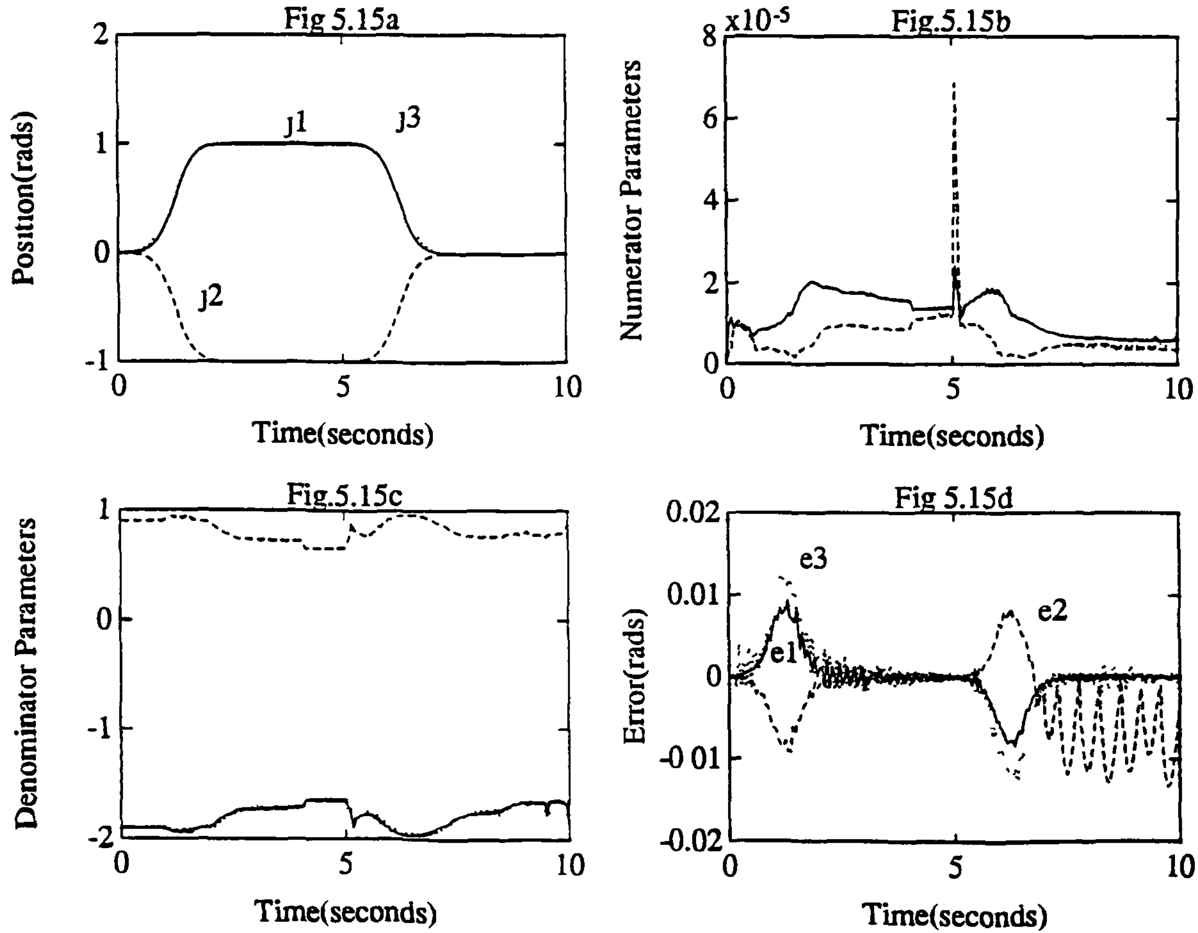


Fig 5.15a
Position(rads)
Time(seconds)
j1
j2
j3
Fig.5.15b
Numerator Parameters
x10-5
Time(seconds)
Fig.5.15c
Denominator Parameters
Time(seconds)
Fig 5.15d
Error(rads)
Time(seconds)
e1
e2
e3

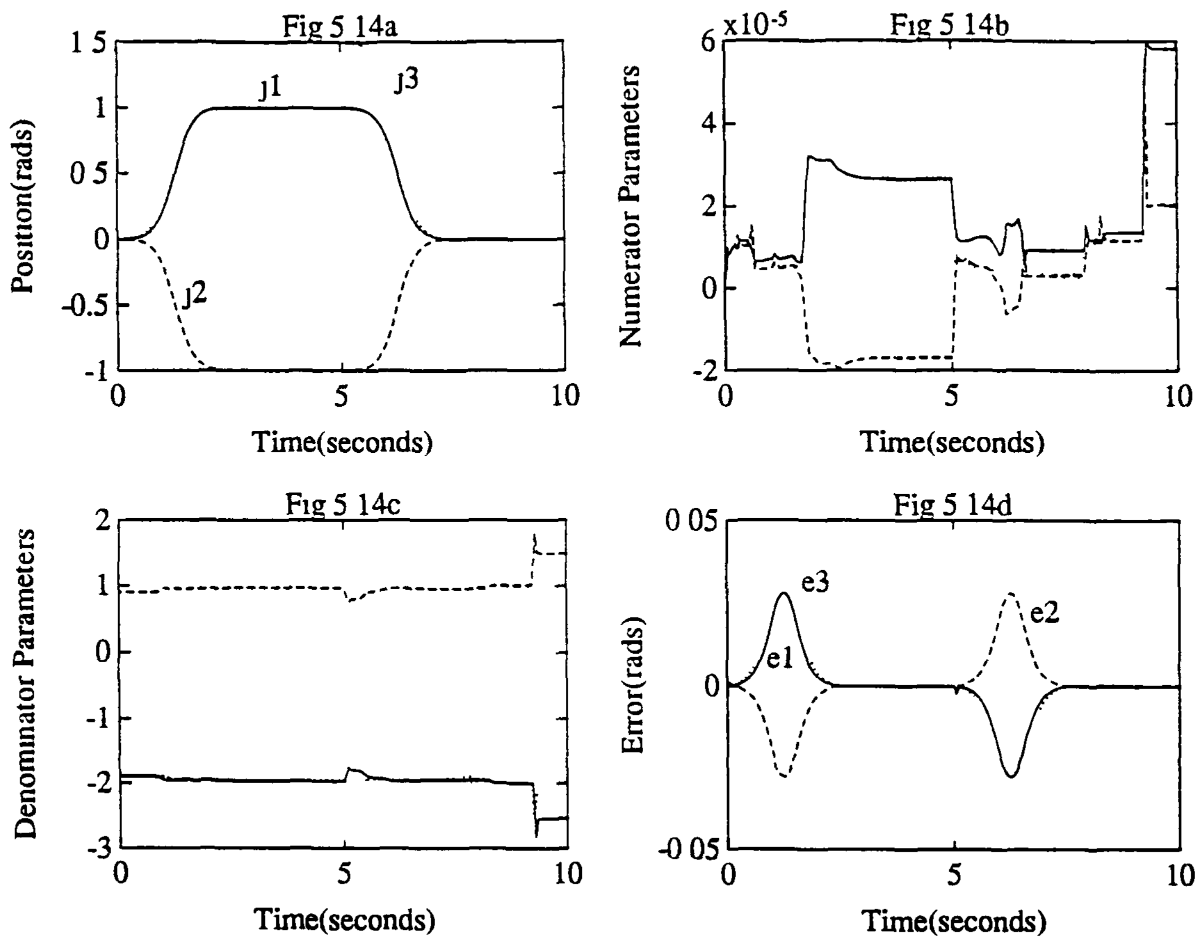


Fig 5 14a
Position(rads)
Time(seconds)
j1
j2
j3
Fig 5 14b
Numerator Parameters
x10-5
Time(seconds)
Fig 5 14c
Denominator Parameters
Time(seconds)
Fig 5 14d
Error(rads)
Time(seconds)
e1
e2
e3

Fig 5 17a

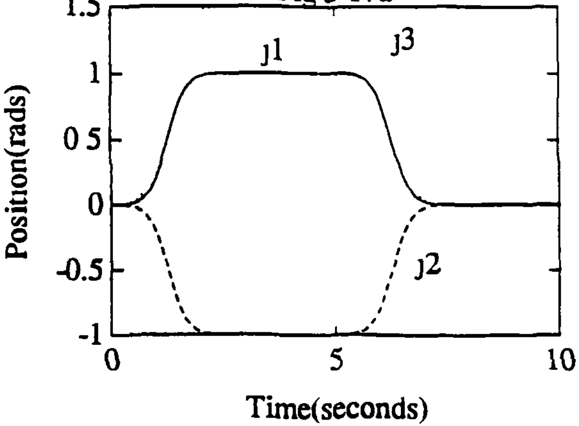
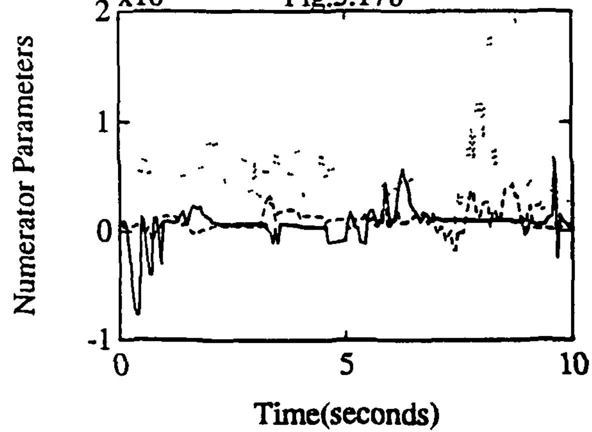


Fig.5.17b



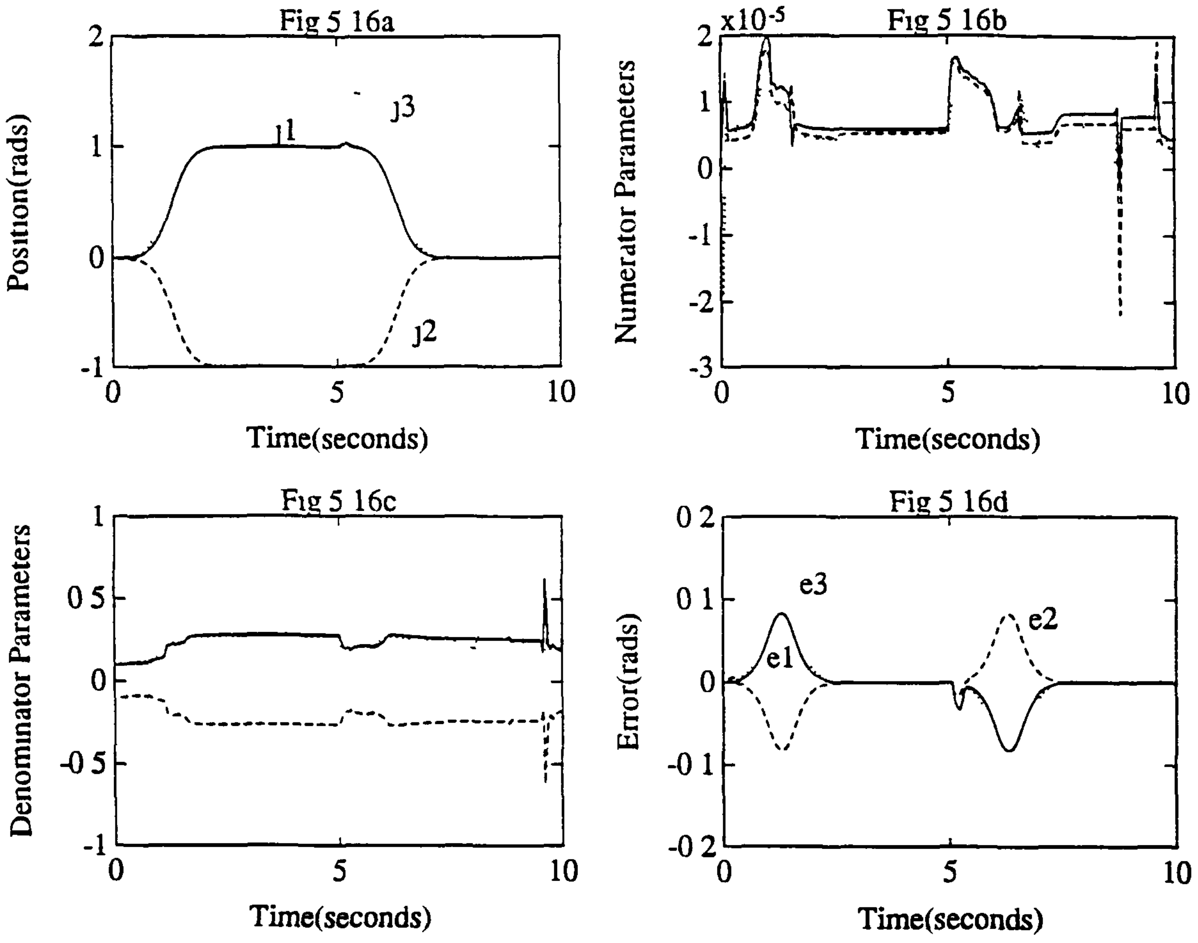


Fig 5 16a
Position(rads)
Time(seconds)
j1
j2
j3
Fig 5 16b
x10-5
Numerator Parameters
Time(seconds)
Fig 5 16c
Denominator Parameters
Time(seconds)
Fig 5 16d
Error(rads)
Time(seconds)
e1
e2
e3

Fig 5 18c

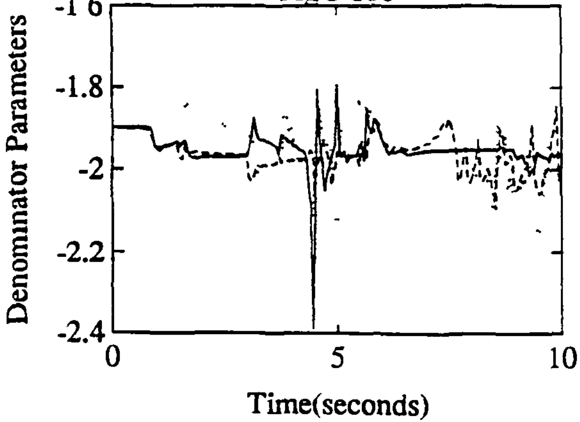


Fig 5.18d

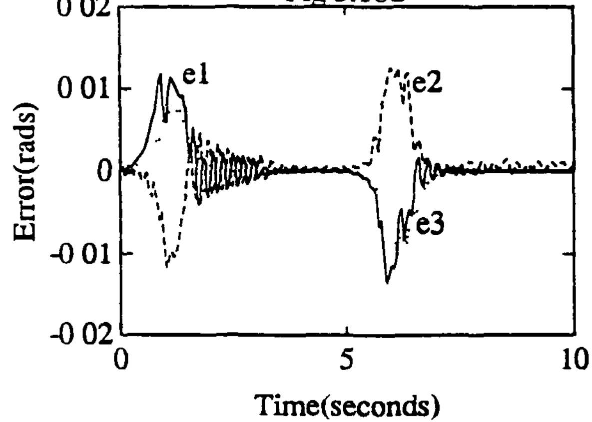
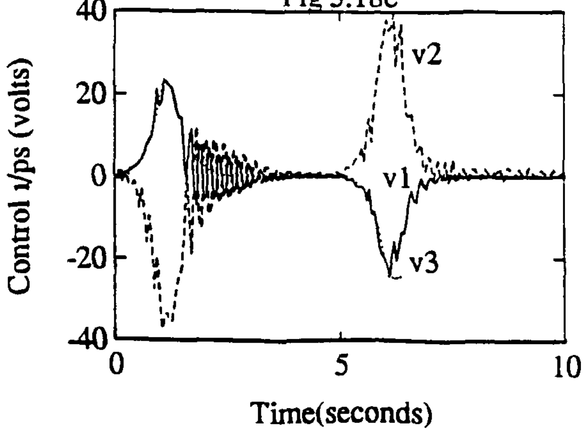


Fig 5.18e



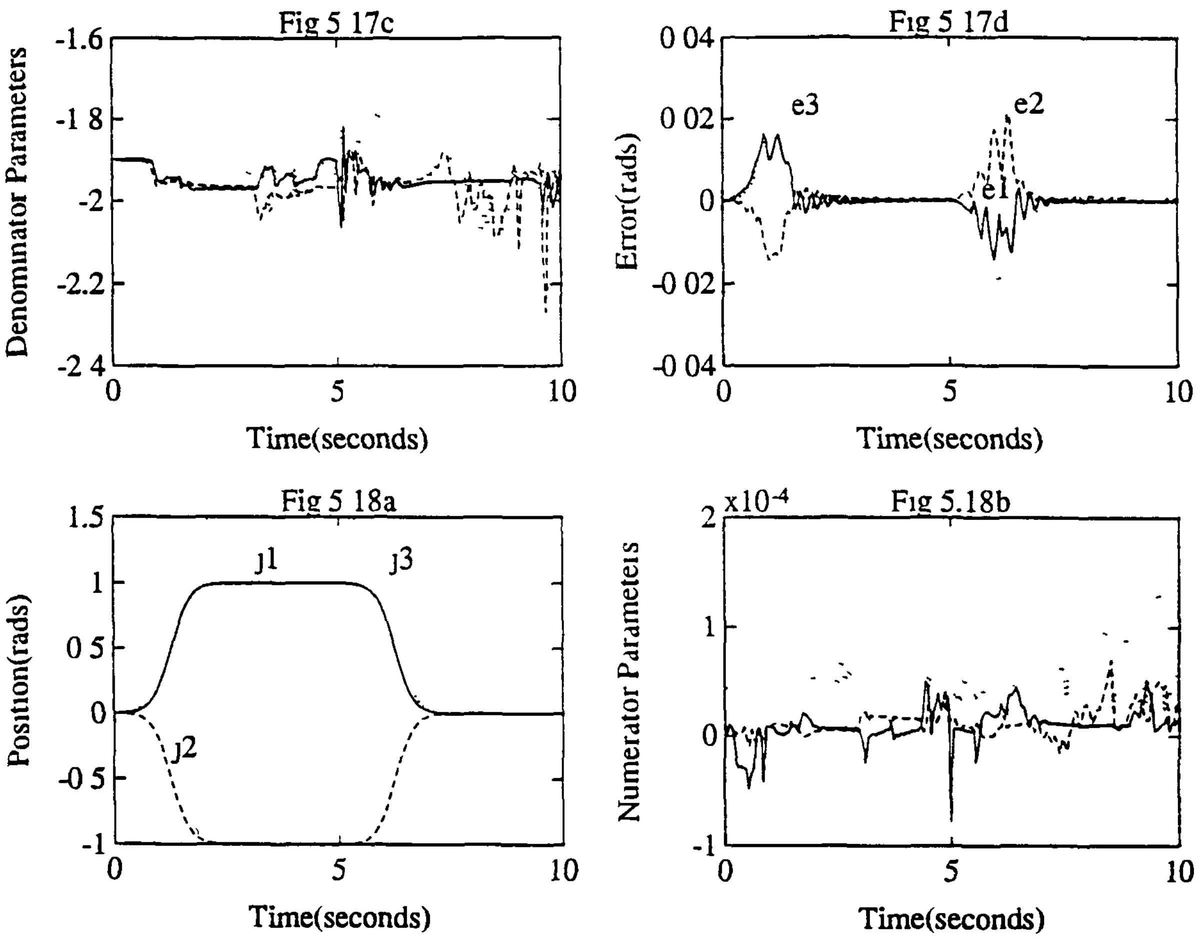


Fig 5 17c
-1.6
-1 8
-2
-2.2
-2 4
0
5
10
Denominator Parameters
Time(seconds)
Fig 5 17d
0 04
0 02
0
-0 02
-0 04
e3
e2
e1
Error(rads)
Time(seconds)
Fig 5 18a
1.5
1
0 5
-0 5
-1
j1
j3
j2
Position(rads)
Time(seconds)
Fig 5.18b
x10-4
2
Numerator Parameters
Time(seconds)

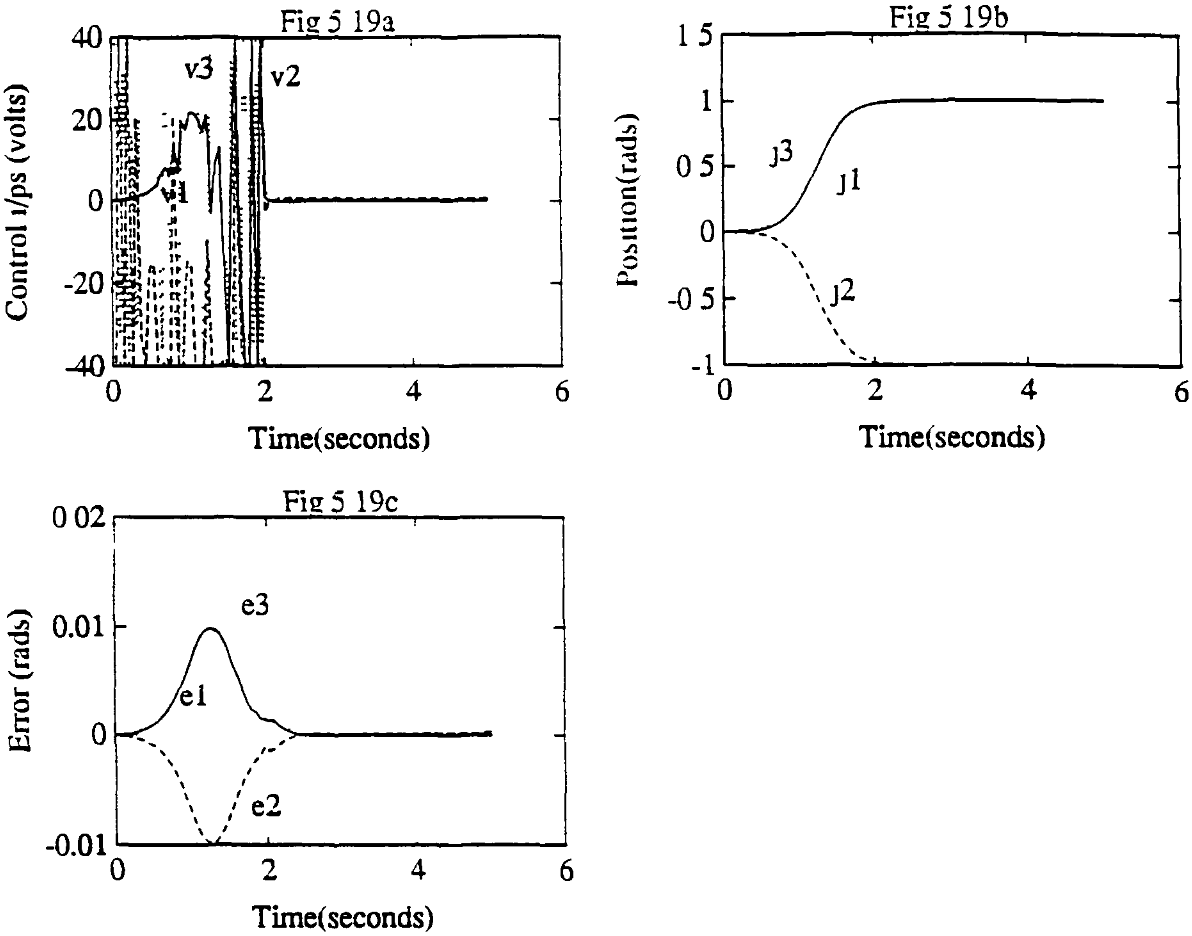


Fig 5 19a
Control i/ps (volts)
v3
v2
v1
Time(seconds)
Fig 5 19b
Position(rads)
j3
j1
j2
Time(seconds)
Fig 5 19c
Error (rads)
e3
e1
e2
Time(seconds)

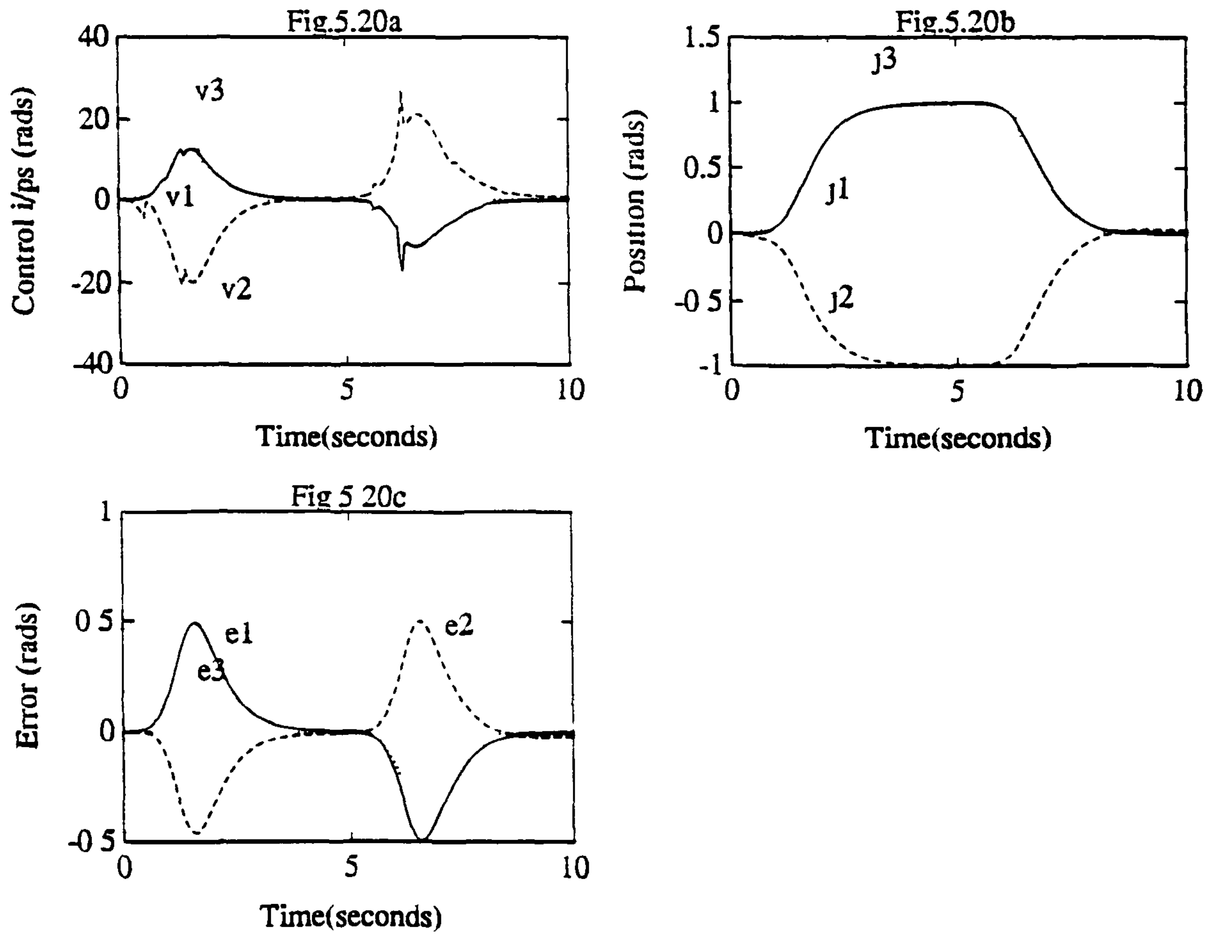


Fig.5.20a
Control i/ps (rads)
v3
v1
v2
Time(seconds)
Fig.5.20b
Position (rads)
j3
j1
j2
Time(seconds)
Fig 5 20c
Error (rads)
e1
e3
e2
Time(seconds)

CHAPTER 6

COMPUTED TORQUE AND FEEDFORWARD CONTROL ALGORITHMS

This chapter is concerned with the investigation of Feedforward Control Algorithms, and in particular Computed Torque This control technique uses an inverse model of the robot dynamics to compute the control inputs Feedforward Controllers have several disadvantages and to improve the performance of Computed Torque, an adaptive PD controller is added to the control loop to improve the static accuracy of the control action Before the control question is addressed, the topic of Feedforward Control is discussed Later in the chapter, the simulation results are presented for Computed Torque, with and without the feedback PD loop

6.1 Properties of Feedforward Controllers

Feedforward Control is also known as Model-Based Control Model-Based Control is a scheme in which a computer model of the controlled process is used to calculate control commands Model-Based schemes can be implemented on powerful digital computers, which are capable of implementing these schemes in real-time Many of feedforward control methods use Inverse Dynamics, e g Computed Torque, Decoupling Torque and Resolved Acceleration Control Computed Torque was the first to be proposed and it was influential in other schemes [38]

Feedforward algorithms are very sensitive to unmodelled dynamics which may result from modelling inaccuracies or dynamic load variations This results in a static error in the output However feedforward control has the advantage of improved transient response over feedback control To achieve a fast transient response and low static error, feedback and feedforward controllers are often combined The control signal from each is added to obtain the total control input (see Fig 6 1) The feedback control command is Δv, the extra control signal which is required to reduce the static error (if any)

6 2 The Computed Torque Method

The Computed Torque method is an alternative approach for manipulator control Its uses an inverse model of the system and dynamically evaluates the torque (or voltage) required by each servo to track a desired trajectory Computed Torque algorithms have the advantage of feedforward controllers, i e fast transient response

Computed Torque is a *Multivariable Control Technique*, not like any of the methods used to date No assumptions are required in the derivation of the control algorithm, unlike the linear control methods where the design is based on the simplified single-joint models Load variation can be accounted for, as long as the load variation is known The next section details the control equation, and it can be seen that there is a large computational burden imposed by this compensator

6.2 1 Controller Derivation

Recalling from Chapter 2, the comprehensive dynamic model of the PUMA 560 robot

$$\begin{bmatrix} y_7 \\ y_8 \\ y_9 \end{bmatrix} = -\underline{D}^{-1}\,\underline{P}(y) + \underline{D}^{-1}\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} \qquad (6\ 1)$$

where V_1, V_2, V_3 are the voltage inputs and,
y_7, y_8, y_9 are the joint accelerations

Rearranging, this gives

$$\Rightarrow \quad \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \underline{P}(y) + \underline{D}^{-1}\begin{bmatrix} y_7 \\ y_8 \\ y_9 \end{bmatrix} \qquad (6\ 2)$$

This is the control equation The desired reference position is not an input to the control equation However, the desired rate of change of the joint accelerations,

$$y_7^*, \; y_8^*, \; y_9^*$$

which are inputs to equation (6 2), are computed using the reference input as shown below

$$y_4^* = \frac{y_1(k+1)^* - y_1(k)^*}{h} \qquad (6\ 3)$$

$$y_7^* = \frac{y_4(k+1)^* - y_4(k)^*}{h} \qquad (6\ 4)$$

$$y_7^* = \frac{y_7(k+1)^* - y_7(k)^*}{h} \qquad (6\ 5)$$

Similarly for y_8 and y_9 Having computed the desired rate of acceleration, the control equation can be computed

$$\Rightarrow \quad \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \underline{P}(y^*) + \underline{D}^{-1} \begin{bmatrix} y_7^* \\ y_8^* \\ y_9^* \end{bmatrix} \qquad (6\ 6)$$

This control equation requires the computation of the robot's inverse dynamics, which involves a considerable number of multiplications and additions [35] This is a computationally complex routine

So in summary, this scheme uses nonlinear feedback to decouple the manipulator The control torque (or voltage) is computed by the inverse dynamics from equation (6 6), using the commanded acceleration y_i^*, i=7,8,9, instead of the measured acceleration y_i, i=7,8,9, where * indicates the desired values of the associated variable [36]

6.2.2. Adding an Adaptive Feedback Layer

To improve the static accuracy of the feedforward controller, a PD controller is added in the feedback loop An adaptive algorithm is chosen due to its superior performance over its fixed gain counterpart Although the static accuracy of the feedforward controller is very good, in this simulation environment there are no unmodelled dynamics to introduce error The error that is present is due to the technique for estimating the rate of change of acceleration In practice the inverse manipulator model used might not contain all the robot's dynamic elements, and it is reasonable to assume it doesn't, and therefore larger static errors will result Hence feedback control must be employed [37]

Fig 6 1 shows the control loop used here [35] The feedforward section however, is not adaptive, only the feedback section The function of the feedback section is to reduce the static error, the feedforward section will ensure a low peak error, due to its fast transient response The outputs of the two controllers are added together to form the control input The identification uses this input and the position outputs to derive the plant estimates The Adaptive PD algorithm from Chapter 5 is used, where a pole-placement design method transforms, the plant estimates to controller gains The feedback control signal is derived from the present and past errors

6.2.3 Simulation Results

Using only a feedforward compensator results in the control action shown in Fig 6 2 The peak error for each joint is

$$e_{pk_1} = 5.5\times10^{-4}$$
$$e_{pk_2} = 1\times10^{-3}$$
$$e_{pk_3} = 3\times10^{-4}$$

These are the lowest values achieved, thus the computational complexity is justified The static error is

$$e_{ss_1} = 4.9\times10^{-5}$$
$$e_{ss_2} = 1.5\times10^{-4}$$
$$e_{ss_3} = 1.2\times10^{-5}$$

and these values are acceptable The feedforward algorithm thus demonstrates its superior transient response over the feedback schemes, exhibiting such a low tracking error

Adding an Adaptive PD controller to the control loop results in the control action shown in Fig 6 3 The static error for each joint is approximately zero, i e

$e_{ss_1} = 6x10^{-11}$
$e_{ss_2} = 5x10^{-8}$
$e_{ss_3} = 1x10^{-10}$

The forgetting factor (μ) is set to 0 99, and P_0 is set at 1,000 The PD parameters are tuned to give a velocity error constant (K_V) of 0 4 Hence the identification is tuned to track slow variance in the manipulator parameters, so the estimated parameters do not vary as widely as in Chapter 5 The only job of the feedback compensator is to reduce the static error, it does not have to track the parameter variations exactly, the feedforward compensator has a comprehensive inverse model of the robot and is able to account for the variations in the manipulator dynamics If K_V is increased to 0 6, this results in a slightly smaller peak error than before, but there is an increase in the static error increases (see Fig 6 4) .

$e_{ss_1} = 3\ 2x10^{-8}$
$e_{ss_2} = 2x10^{-5}$
$e_{ss_3} = 1\ 9x10^{-5}$

The results in Fig 6 3 are the best obtained in this chapter

6.2.4 The Effect of Model Mismatch

Feedforward controllers are very sensitive to inaccuracies in the modelled dynamics Here the effect of model mismatch is investigated The contribution of gravity is reduced by a factor of two in the internal model This creates a sizeable mismatch between the control model and the process model

Fig 6 5 shows the result of this modelling inaccuracy when using the computed torque control technique A large static error of 0 11rads results on joint 2 This error is proportional to the degree of model mismatch However, when feedback is employed in conjunction with computed torque, the effect of this mismatch is negligible (see Fig 6 6) These results confirm that the most efficient and robust controller is the computed torque algorithm with a feedback loop

6.3 Summary

This chapter is concerned with the topic of feedforward control and especially, Computed Torque Feedforward control has several advantages over Feedback systems The most important is the improved transient response effect This merits no explanation when one considers the method of feedforward compensation, no error measurement is required to calculate the control input However, there are also drawbacks when using feedforward compensation Feedback compensation is very sensitive to modelling inaccuracies, which result in static inaccuracies in the controlled variable

From the above discussion, it was decided to use both types of compensators, Feedforward and Feedback The feedforward section ensures a fast transient response, i e low peak error, and the feedback section reduces the static error The incorporation of the two techniques gives the best results achieved in this thesis Fig 6 1 shows a block diagram of the control loop Only the feedback section contains an adaptive layer, the feedforward controller is fixed For future improvements, a nonlinear adaptive identifier could be added to the loop, thus having an adaptive feedforward section also

In the results obtained here, no modelling inaccuracies are present An exact inverse model of the robot simulator dynamics is possible In practice, however the inverse model will not contain all the robot dynamics, and the control results will not be as good as the above simulation performance But despite this fact, this algorithm (with the adaptive PD section) is considerably better than any other algorithm used It outperforms all the algorithms in the areas of static accuracy and peak error This controller is more robust and can compensate for model mismatch Also it is the most complex of all, but the complexity is justified due to its superior performance State of the art processors are able to implement these control schemes with suitable sample periods

Index to Graphs

o *Fig 6 6a* Plot of Joint Positions versus Time for Cubic Spline Trajectory Demands

o *Fig 6 6b* Plot of Joint Positional Error versus Time for Cubic Spline Trajectory Demands

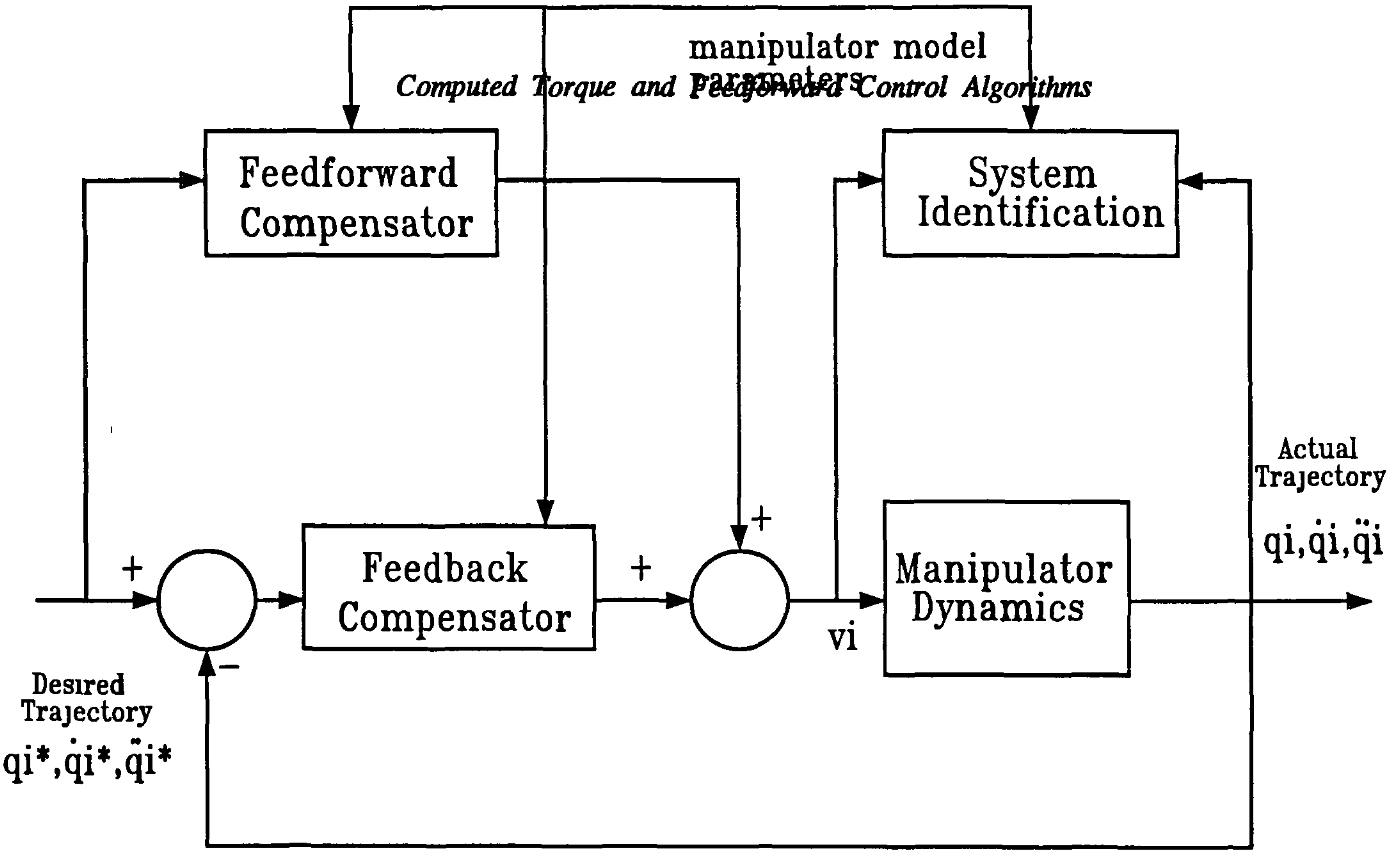


Fig.6.1 General Manipulator Dynamic Controller

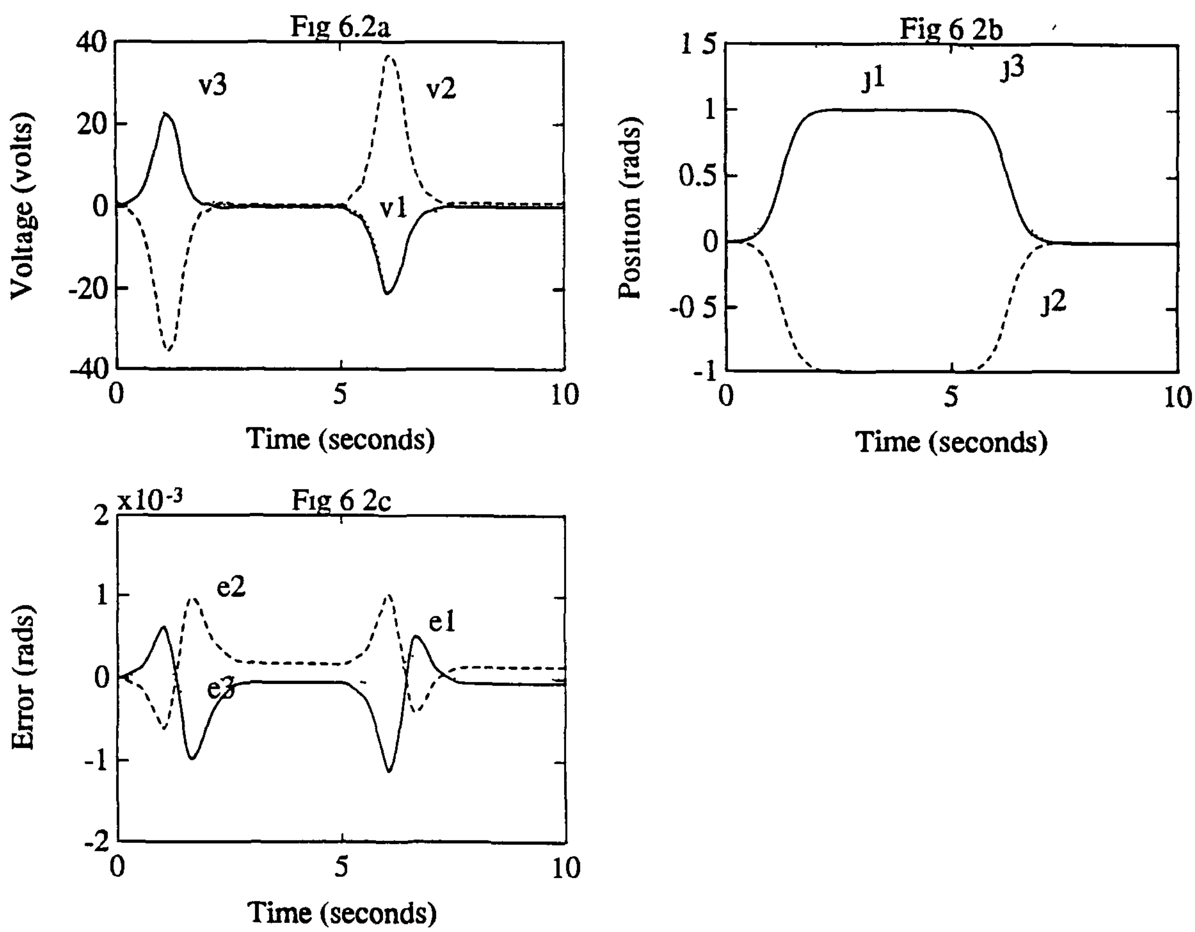


Fig 6 3a

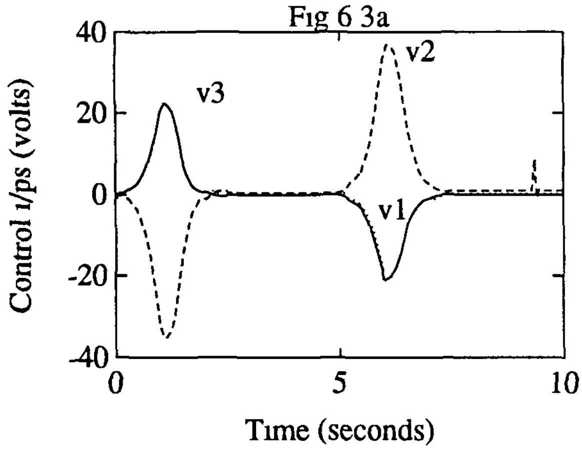


Fig 6 3b

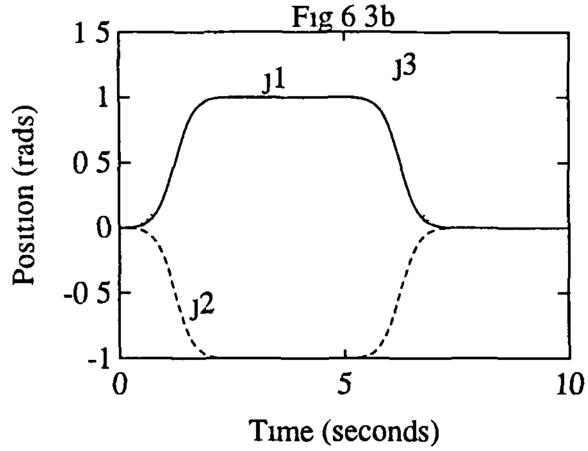


Fig 6 3c

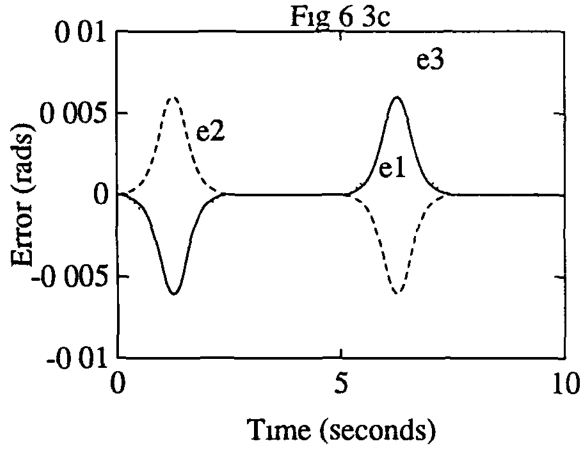


Fig 6 4a

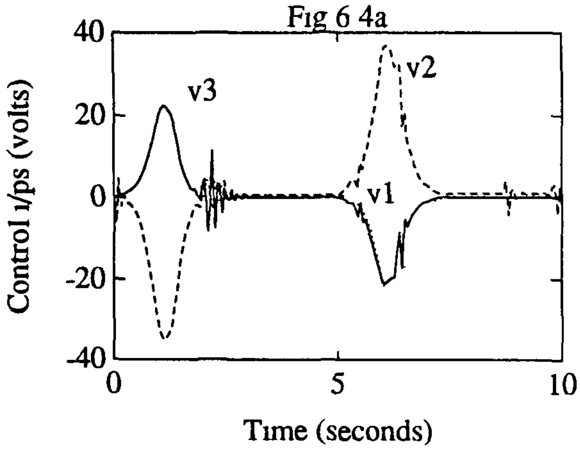


Fig 6 4b

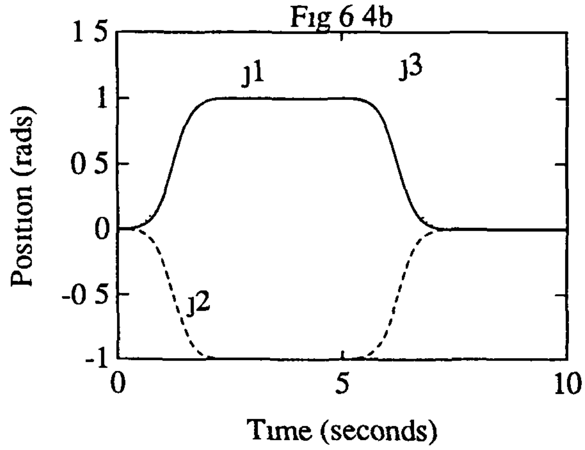


Fig 6 4c

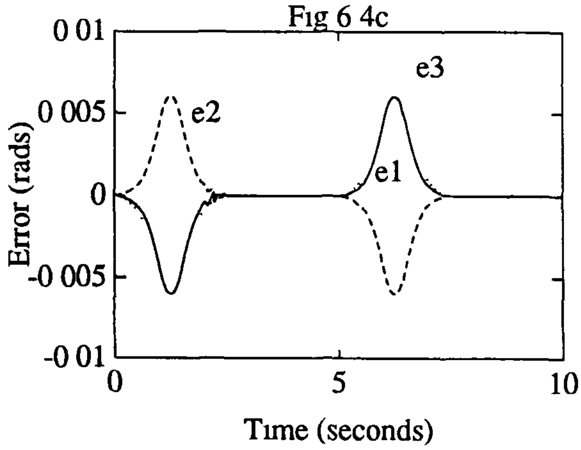


Fig 6 5a

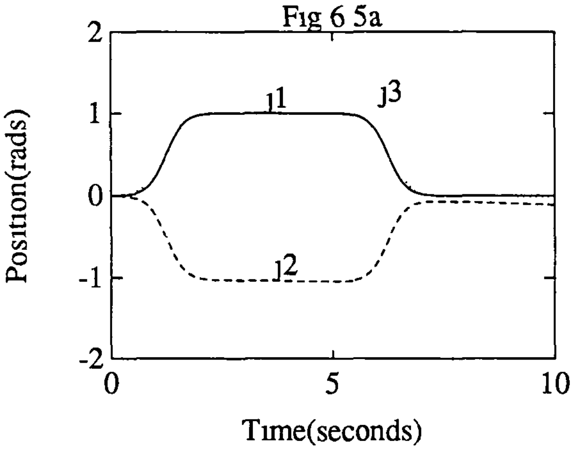


Fig 6 5b

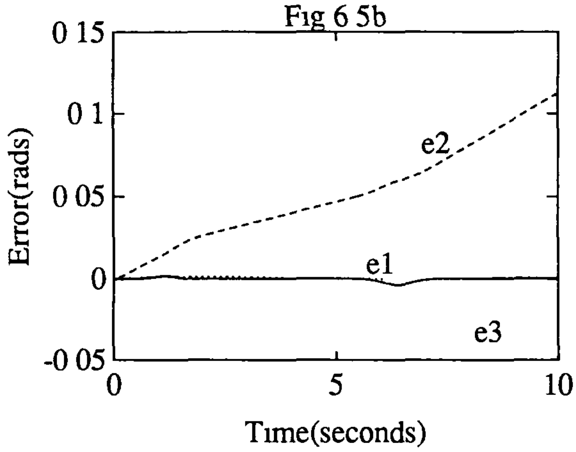


Fig 6 6a

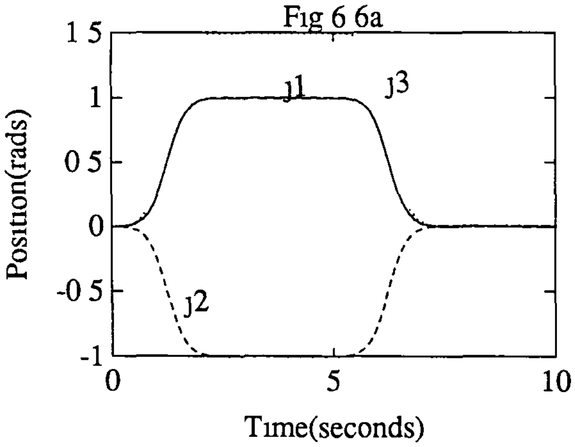
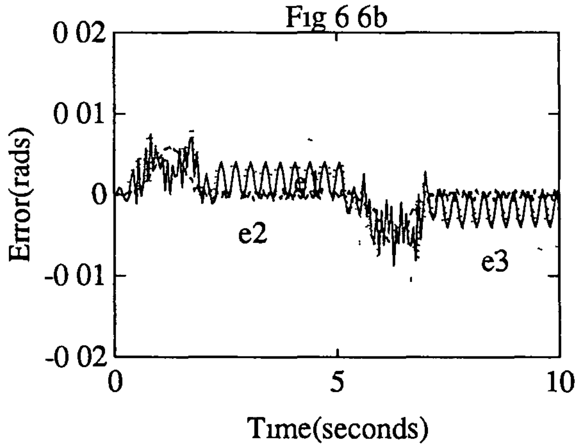


Fig 6 6b



CHAPTER 7

CRITICAL EVALUATION OF THE SIMULATION RESULTS

In this chapter, the simulation results from Chapters 4, 5 and 6 are evaluated according to a set of performance criteria. The results of this evaluation are for later investigation in Chapter 8. A wide range of digital control techniques is presented in these earlier chapters, so the conclusions here incorporate the performance of most of the suitable control techniques available.

The performance criteria are chosen with complete performance in mind, i.e. from the design stage to the implementation stage. The control algorithms are assessed thoroughly in this chapter, and an order of merit table is formed.

Also included here are the results of each algorithm when a varying payload is introduced. The graphs show the error after the payload is increased. A peak error and a static error results, and the algorithms are rated according to how low these errors are in magnitude. The total error introduced by varying the payload, is calculated by adding the error at each sampling interval and multiplying the answer by the product of the time interval and the sampling interval. This gives a measure of the integral of the error curve and hence one can evaluate the performance from this information.

7.1 The Performance Criteria

To determine which control algorithms are the most suitable for manipulator control, a set of performance criteria is required. A points scheme is devised, where each algorithm receives a number according to its performance in each of the categories. The algorithm with the total highest number of points is deemed to be the most suitable algorithm for robotic control, but the applications or tasks of the robot also influence the decision as to what algorithm to use The algorithms can then be listed in their order of merit. Some of the performance criteria are weighted, i e a good performance in one section is worth more points than the same performance in another section There are six specifications with which to judge the algorithms, and these are as follows

1 Design Complexity
2 Computational Complexity
3 Transient Response
4 Static Accuracy
5 Varying Payload Test
6 Robustness

Numbers 3, 4 and 5 are the important sections Design complexity is not really that important, from an academic point of view A very complex design procedure means more effort prior to the implementation of the algorithm, and hence more work for the designer A complex design procedure is not really a downside to any algorithm The computational complexity can be ignored if sufficiently powerful hardware is available, so little weight is given to this section. The varying load test consists of adding an extra mass to joint 3 to simulate the result of picking up a load At t=3, the load is added, and the cubic spline trajectory is continued, to observe the extra tracking error introduced by this test. The robustness of each routine is measured as the amount of variation that is permitted in the tuning parameters or control gains before the closed-loop system becomes unstable Each performance index has three grades associated with it. These grades are listed below

Design Complexity
1 Very complex
2 Medium
3 Relatively Simple

Computational Complexity
1 High
2 Medium
3 Low

Transient Response
1 good - peak error $< 10^{-2}$
2 o k - $10^{-2} <$ peak error $< 10^{-1}$
3 poor - peak error $> 10^{-1}$

Static Accuracy
1 good - $error_1 < 10^{-9}$
$error_{2,3} < 10^{-5}$
2 medium - $10^{-9} < error_1 < 10^{-5}$
$10^{-5} < error_{2,3} < 10^{-3}$
3: poor - $error_1 > 10^{-5}$
$error_{2,3} > 10^{-3}$

Varying Load Test

1 good - error < 0 35

2 o k - 0 35 < error < 1 75

3 poor - 1 75 < error < 3 5

4 bad - error > 3 5

Robustness

1 very robust

2 o k

3 poor

The static accuracy is divided into two specification sections, one for joint 1 and the other for joints 2 and 3 Since there is no gravity acting on joint 1, there is always lower static error than for any of the other joints

Table 7 1 is a points key to the Performance Table (see Table 7 2) It allocates a value for the performance of each algorithm under the grades 1-4,

	Grade			
Index	1	2	3	4
Design Complexity	1	3	5	
Computational Complexity	1	3	5	
Transient Response	7	4	1 *	
Static Accuracy	7	4	1 *	
Varying Payload Test	7	4	1	0 *
Robustness	5	3	1	

Table 7.1 Points Key

where * indicates a weighted index Using table 7 1, the next section proceeds to evaluate each control technique individually

7.2 Evaluation of the Control Algorithms

Starting with PID techniques, the fixed parameter and adaptive technique (both PD and PID) are relatively easily designed, only two or three gains to evaluate (all grade 3) The adaptive versions are more computationally complex than their fixed parameter versions (grade 3 for fixed parameter and grade 2 for the adaptive case) The peak error for a fixed PD algorithm fits into grade 3, but the adaptive algorithms and the fixed gain PID give a sufficiently low error to merit grade 2 However, for static accuracy, the fixed PD performs poorly, only grade 3 Its adaptive version is a grade 2 in this category, but full PID (fixed and adaptive) is the best in this respect (grade 1), the integrator greatly reduces the static error The total error introduced by varying the payload is as follows for the four routines

Fixed

PD	PID
e_{total_1} = 0 829	e_{total_1} = 0 066
e_{total_2} = 6 7	e_{total_2} = 0 148
e_{total_3} = 0 721	e_{total_3} = 0 0434
(grade 4)	(grade 1)

Adaptive

PD	PID
e_{total_1} = 0 0868	e_{total_1} = 0 107
e_{total_2} = 0 735	e_{total_2} = 0 285
e_{total_3} = 0 11	e_{total_3} = 0 35
(grade 2)	(grade 1)

The robustness of these PD and PID algorithms is grade 2 The controller gains can be changed without instability occurring immediately PID is slightly less robust than PD because of the integrator in the closed-loop The adaptive algorithms are also grade 2, as long as constraints are placed on the parameter estimates

The frequency domain compensators are discussed as a single unit, except in the categories of transient response, static accuracy and the varying payload test These compensators have a fairly complex design procedure (grade 2) but the control equation is a simple difference equation, i e computational complexity is grade 3 All three give grade 2 and 3 in their transient response and static accuracy, respectively However, a lead compensator is preferred over the lag. Large lags increase the rise time The total error introduced by the varying load is .

LEAD	LAG	LAG–LEAD
e_{total_1} = 0 347	e_{total_1} = 0 287	e_{total_1} = 0 182
e_{total_2} = 3 35	e_{total_2} = 1 77	e_{total_2} = 2 45
e_{total_3} = 0 189	e_{total_3} = 0 273	e_{total_3} = 0 199
(grade 3)	(grade 3)	(grade 3)

These type of compensators are of medium robustness (grade 2), the specifications for the design have a wide allowable range

The optimal design procedure is complex (grade 1). It is difficult to calculate the controller transfer function by the minimization of a cost function. However, the control equation is only a difference equation, very simple to implement and the computational complexity is grade 3 The performance is poor, grade 3 in each of the error sections The error introduced by the varying load is

OPTIMAL

e_{total_1} = 0 84
e_{total_2} = 7 36
e_{total_3} = 1 149
(grade 4)

The choice of the specification weights, r and q, is wide Hence the algorithm is very robust (grade 1), but the performance is poor

The Monoreg Predictive Control Algorithm (pred1 in table 7 2) is of medium complexity in its design stage (grade 2), but the output feedback method (pred2) has a very simple design procedure (grade 3) The adaptive routines have the same design complexity as their fixed parameter counterparts In the Monoreg routine, matrix manipulation takes place In the fixed algorithm, A^H is calculated off-line once, prior to the control algorithm, hence grade 3 for the fixed Monoreg routine However, for the adaptive algorithms, adaptive Monoreg and the gain-scheduled routine, this constant must be evaluated at each sampling instant, as well as the parameter estimation taking place Hence, these algorithms are of medium complexity (grade 2) The other predictive controller (both fixed and adaptive) is very simple to implement, both are grade 3 This algorithm (both fixed and adaptive) performs poorly in the error sections - grade 3 in both transient response and static error performance The Monoreg routine (both fixed and adaptive) and the gain-scheduled routine perform as grade 2 in these sections The error introduced by the varying load is .

Fixed

PRED1	PRED2
e_{total_1} = 0 212	e_{total_1} = 1 582
e_{total_2} = 0 317	e_{total_2} = 7 36
e_{total_3} = 0 329	e_{total_3} = 1 566
(grade 1)	(grade 4)

Adaptive

PRED1	PRED2	PRED3
e_{total_1} = 0 444	e_{total_1} = 0.125	e_{total_1} = 31 6
e_{total_2} = 0 81	e_{total_2} = 0 35	e_{total_2} = 11 3
e_{total_3} = 0 702	e_{total_3} = 0 103	e_{total_3} = 3 9
(grade 2)	(grade 1)	(grade 4)

The output feedback routine (both fixed and adaptive) is not very robust (grade 3), the value of α must remain above 0 97, or instability occurs The other routine is robust and almost all values of the tuning parameters give closed-loop stability

Model Reference Adaptive Control has a very complicated design procedure (grade 1) and the control loop requires the solution of two differential equations Hence the computational complexity is grade 2 The peak error is comparatively large (grade 3), but the static accuracy is grade 2 The error introduced by the varying load is

MRAC

$e_{total_1} = 0\ 653$

$e_{total_2} = 1\ 771$

$e_{total_3} = 0\ 43$

(grade 2)

The adaptation constants can be widely varied giving stable closed-loop results, hence this is a very robust algorithm

The Self Tuning Regulator design requires the solution of a diophantine equation and the evaluation of five controller parameters This merits a grade 2 design complexity Also, the computational complexity is grade 2 The Explicit version (STR1) performs to grade 2 in both peak error and static error requirements The Implicit algorithm (STR2) also merits grade 2 for static accuracy, but only grade 3 for its transient response The error introduced by the varying load is

STR1	STR2
$e_{total_1} = 0\ 0576$	$e_{total_1} = 0\ 287$
$e_{total_2} = 0\ 209$	$e_{total_2} = 2.86$
$e_{total_3} = 0\ 1136$	$e_{total_3} = 0\ 441$
(grade 1)	(grade 3)

The implicit routine is not very flexible If a faster reference model is used, undesirable results are obtained The explicit version is fairly robust (grade 2), and faster reference models do not cause instability

Computed Torque is the only feedforward control technique investigated in this thesis Its design and computational complexity are grade 2 The performance of both algorithms is grade 1 in the transient response section The controller with adaptive feedback loop merits a grade 1 for static accuracy, but the other method merits only grade 2 The error introduced by the varying load is

COM1	COM2
$e_{total_1} = 0\ 0149$	$e_{total_1} = 0\ 0413$
$e_{total_2} = 0\ 213$	$e_{total_2} = 0\ 0497$
$e_{total_3} = 0\ 0189$	$e_{total_3} = 0\ 0836$
(grade 1)	(grade 1)

Both are very robust algorithms (grade 1), the simple Euler approximations for the acceleration derivatives give very low peak error values

Table 7 2 shows the grade achieved by each algorithm in the various categories This number is transferred using table 7 1, to a performance number These are then added to evaluate the total performance of the algorithms Those on equal points are further graded by the use of an extra number in brackets beside the points awarded Computed Torque, from the evaluation process, is deemed to be the most suitable algorithm for manipulator use However, some simpler algorithms, such as the Self Tuning Regulator (Explicit version) and the fixed gain PID, are not far behind in their performance, and offer competitive alternatives Fixed gain PID is the most desirable routine in the first section of control routines, and the STR (Explicit version) is the most efficient in section 2

7.3 Choosing the *Best* Algorithm

Not all robot controllers are capable of implementing the Computed Torque control technique with adequate sampling periods due to limitations in the hardware being used The Unimation Control hardware, for example, employs six Rockwell microprocessors (µPs) [39]. These µPs are not sufficiently powerful to implement the more complex control routines with low sampling periods (5msecs) that are required when controlling industrial robots If the control hardware is powerful enough, the first choice of algorithm would be Computed Torque with an adaptive PD feedback layer Algorithms such as the Monoreg Predictive controller (both fixed parameter and adaptive versions) and the Self-Tuning Regulator offer competitive options The application, or daily tasks of the robot, is also a key factor when deciding which routine to use If only simple tasks, such as spray painting, are performed by the robot, then a simple PD or PID routine is sufficient. On the other hand, if high accuracy is required when the robot is performing high speed PICK and PLACE operations, then a more complex algorithm is required. Some manufacturers do not agree with the use of complex schemes, saying the extra cost does not sufficiently improve the performance of the manipulator However, from the simulation results in this project, the extra processor burden dramatically improves the response speed, and the accuracy of the manipulator is also increased The static error in some routines can be made very low So, in choosing a suitable control routine, the available hardware, the tasks to be performed by the manipulator, and the performance specifications have to be considered

When implementing these controllers, a sampling period of 1→5msecs is required due to the complexity of the robot model Larger sampling periods introduce uncertainty and the controller performance is degraded

7.4 Summary

This chapter reviews and assesses the simulation results of Chapters 4, 5 and 6 In these chapters three different types of control algorithms are investigated Using different performance indices, the algorithms are evaluated and compared Design, performance and implementation, are used to assess these algorithms

The results of this evaluation are shown in Table 7 2 The most effective routine is the Computed Torque method with a PD feedback loop However, this is a very complex routine and requires powerful hardware for its implementation Competitive options to this routine include Predictive Control and the Self Tuning Regulator Simple fixed gain PID performs surprisingly well and proves to be a leading control method One disappointing routine is the MRAC method. For the complexity involved, it does not justify the solution of controller differential equations from the performance achieved

Index to Graphs

Results of the Varying Load Test

- ☐ Fig 7 15 *Plot of Joint Positional Error versus Time using the Adaptive Monoreg controller*

- ☐ Fig 7 16 *Plot of Joint Positional Error versus Time using the Adaptive Output-Feedback controller*

- ☐ Fig 7 17 *Plot of Joint Positional Error versus Time using the Computed Torque controller*

- ☐ Fig 7 18 *Plot of Joint Positional Error versus Time using the Computed Torque controller with an Adaptive feedback PD-loop*

Performance Criteria

Control Algorithms	Design Complexity	Computational Complexity	Transient Response	Static Accuracy	Varying Payload Results	Robustness	Total Points for Performance
Section 1							
PD	3	3	3	3	4	2	15
PID	3	3	2	1	1	2	31
Lead	2	3	2	3	3	2	17 (1)
Lag	2	3	2	3	3	2	17 (3)
Lag-Lead	2	3	2	3	3	2	17 (2)
Optimal	1	3	3	3	4	1	13 (1)
Pred1	2	3	2	2	1	1	28
Pred2	3	3	3	3	4	3	13 (3)
Section 2							
PD	3	2	2	2	2	2	23 (2)
PID	3	2	2	1	1	2	29 (2)
MRAC	1	2	3	2	2	1	18
STR1	2	2	2	2	1	2	24
STR2	2	2	3	2	3	3	13 (2)
Pred1	2	2	2	2	2	1	23 (1)
Pred2	2	2	2	2	1	1	26
Pred3	3	3	3	3	4	3	13 (4)
Section 3							
Com-Tor1	2	2	1	2	1	1	29 (1)
Com-Tor2	2	2	1	1	1	1	32

Table 7.2 Performance Table

Fig 7 1

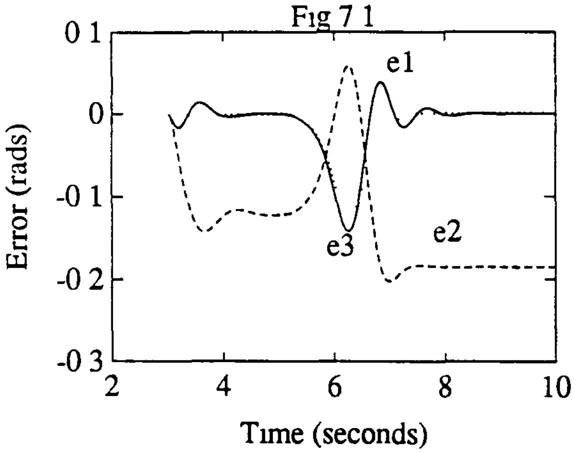


Fig 7 2

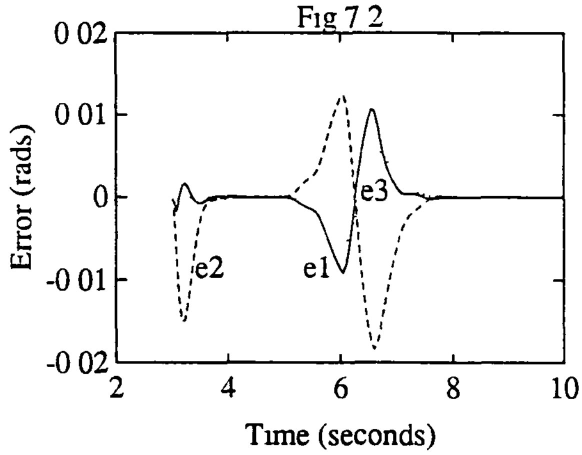


Fig 7 3

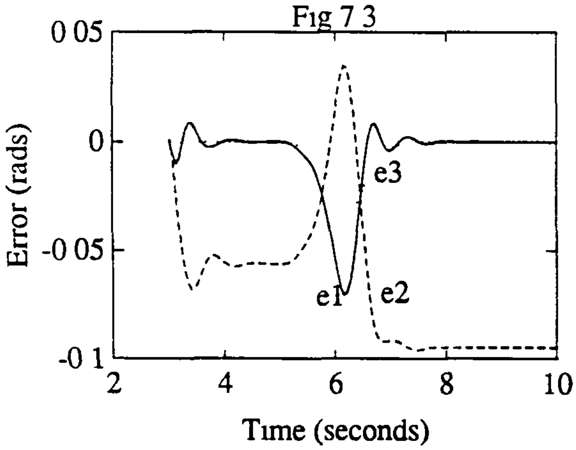


Fig 7 4

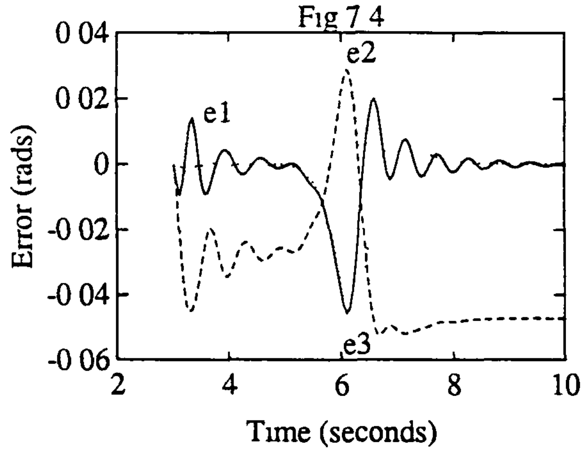


Fig 7 5

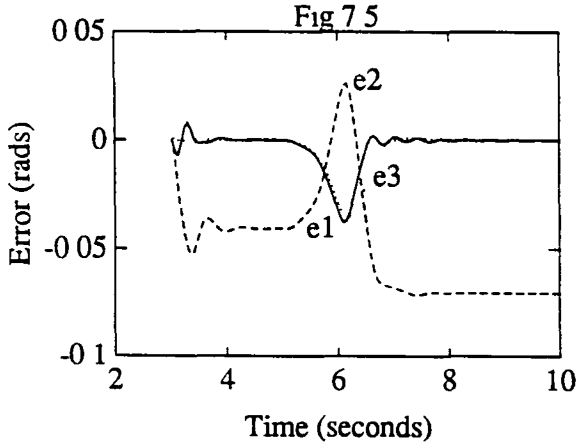


Fig 7 6

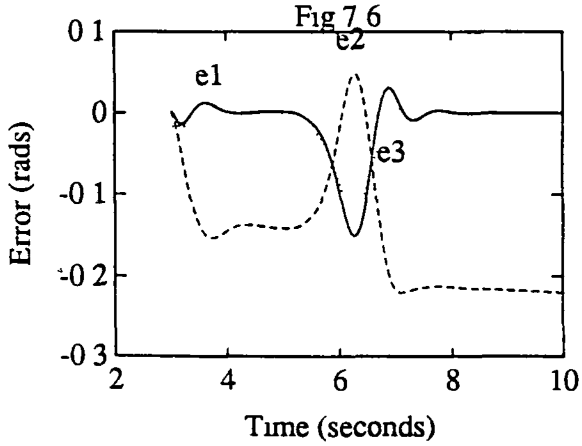


Fig 7 7

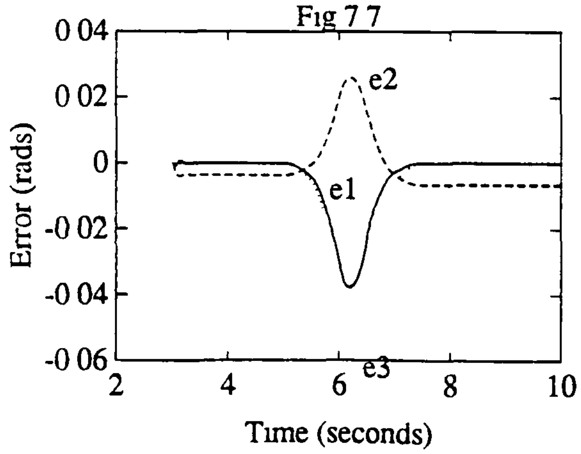


Fig 7 8

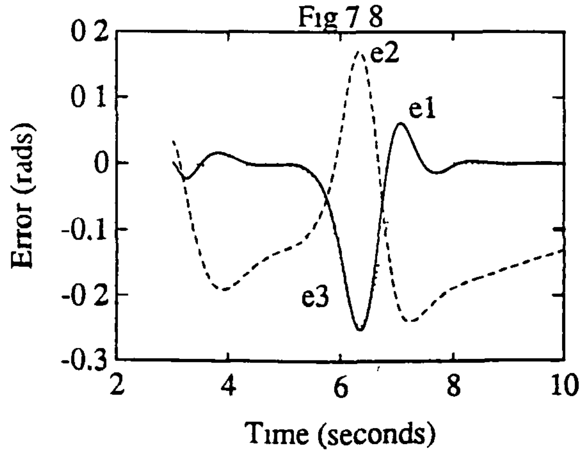


Fig 7 9

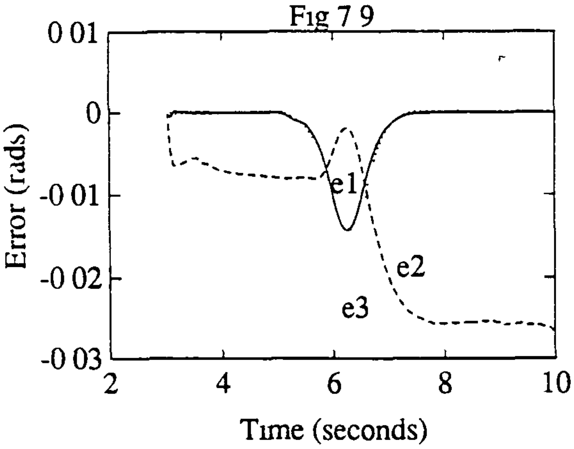


Fig 7 10

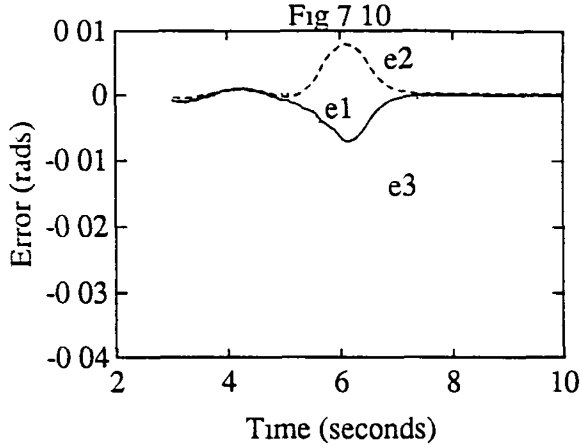


Fig 7 11

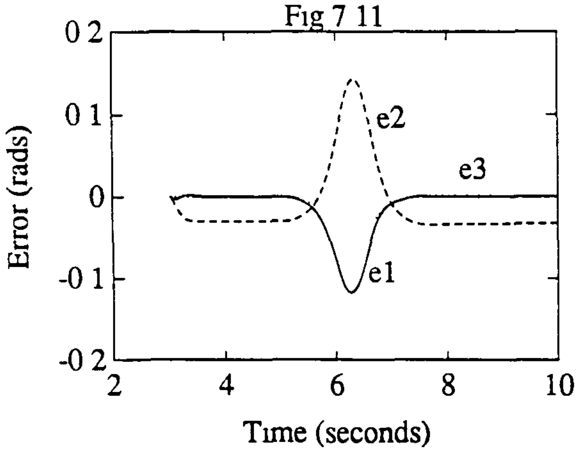


Fig 7 12

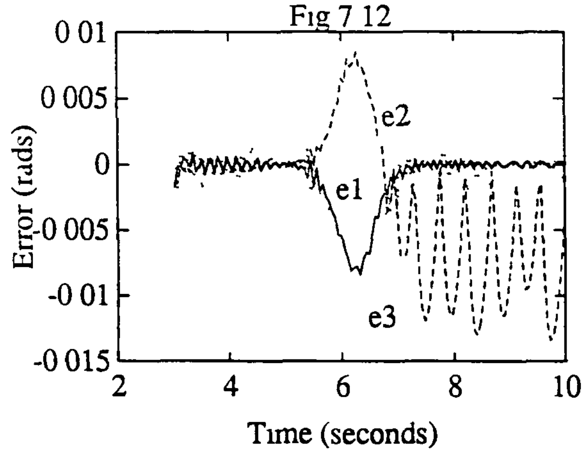


Fig 7 13

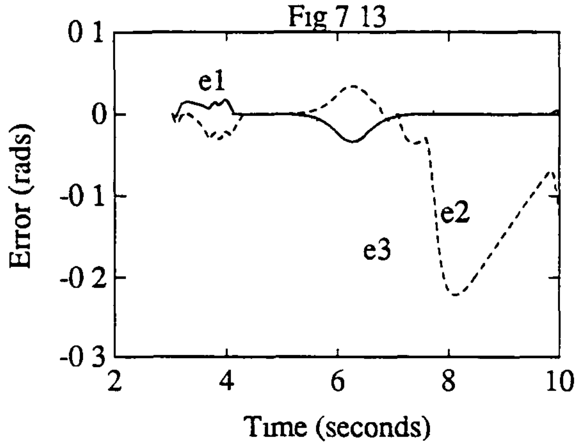


Fig 7 14

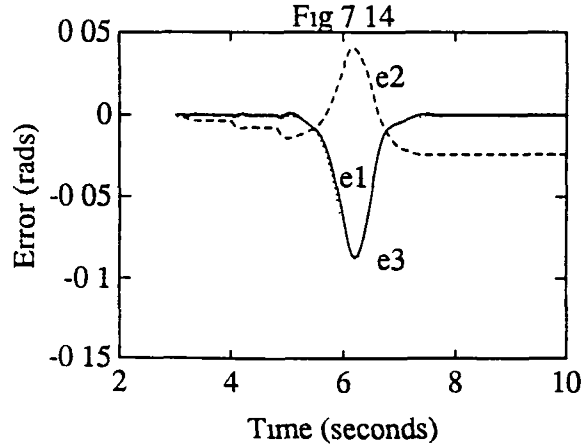


Fig 7 15

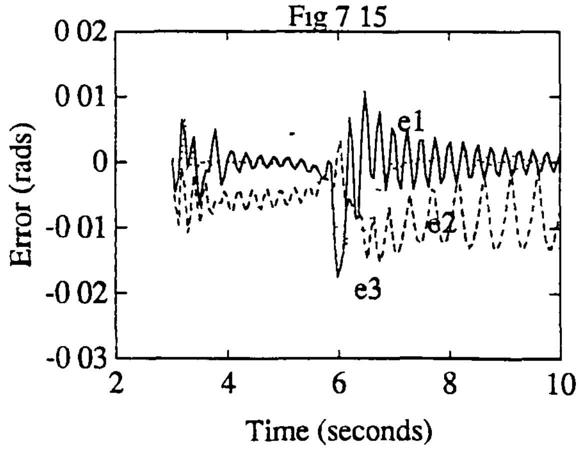
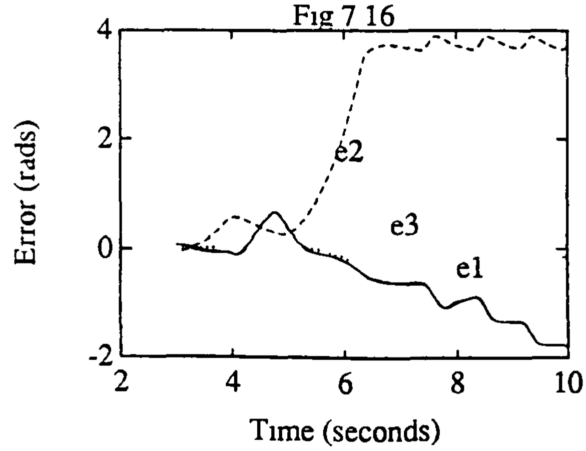


Fig 7 16



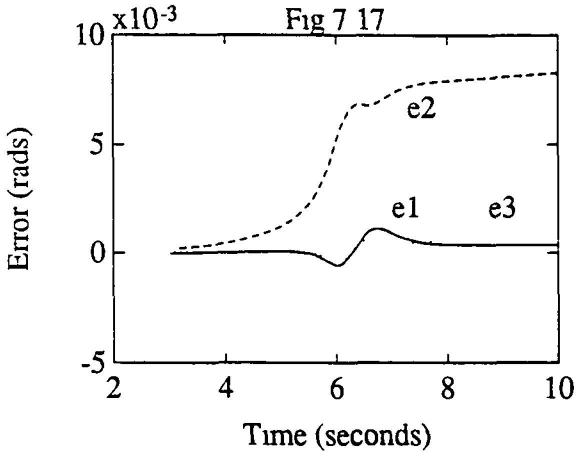


Fig 7 17
x10-3
10
5
0
-5
2
4
6
8
10
Error (rads)
Time (seconds)
e2
e1
e3

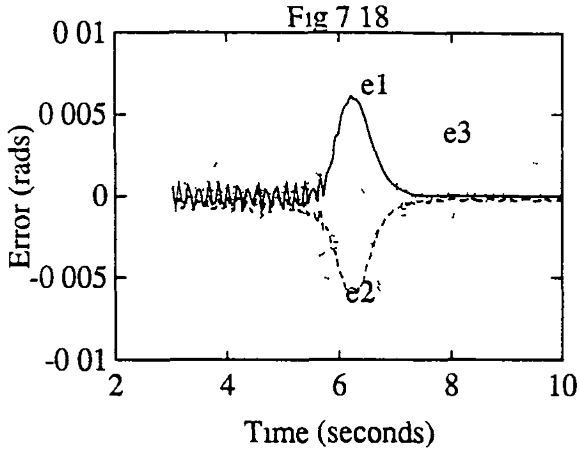


Fig 7 18
0 01
0 005
0
-0 005
-0 01
2
4
6
8
10
Error (rads)
Time (seconds)
e1
e2
e3

CHAPTER 8

HARDWARE SYSTEM DESIGN AND IMPLEMENTATION

Commercial robot systems are generally restricted in terms of modifications to hardware and software for real time control This may be acceptable in workspaces where the repetition of a limited sequence of motions is all that is required In both flexible manufacturing and robotic research environments, however, the primary considerations are ease of modification, adaptability and programmability These three characteristics are essential in order to manufacture a new product for the evaluation of a new sensor system or robot control algorithm [7]

Most commercial robots, like the PUMA 560, are sold with a dedicated programming language, which runs on a dedicated hardware configuration. As a result, the characteristics mentioned above are not present in the PUMA 560 This necessitates the design of a new, more flexible, controller for this robot Before designing a new controller, it is essential to point out the shortcomings in the existing controller to make sure these shortcomings do not reappear in the new controller

In the case of the PUMA 560 industrial robot, a limited form of task-space control is provided by VAL2 (Victor's Assembly Language) [40] VAL combines the features of an operating system and a programming language with the aim of allowing the user to teach new paths and to control the robot in a variety of tasks As an operating system, VAL provides the necessary input/output to control the robot, retrieve data from the floppy disk and to interact with the user via the terminal or a teaching pendant. Despite the relative ease of use and its capabilities, the VAL-based system is seriously lacking [11] in terms of flexibility and expandability, and is devoid of the ability to implementing powerful real-time task space control This can be contributed to the following reasons

1 VAL was written specifically for a PUMA-type manipulator using only if-then commands, like those found in the BASIC language
2 The operating system has only an interpreter, and has no compiler.
3 The VAL software is currently stored in Eproms, which does not enable the user to examine and modify the software
4 Inverse kinematics and path planning software is not user accessible, hence new trajectories cannot be planned off-line

Several suggestions have been made to allow for large program creation, two possible alternatives are outlined in Unimation [41] However in order to gain more flexibility and the ability to program in a high level language, it is necessary to break away from VAL completely

The Unimation control hardware [39] consists of an LSI-11/02 and six Rockwell 6503 microprocessors each with a digital-to-analog converter (DAC), a current amplifier and some joint position feedback sensors The hardware is hierarchically arranged The upper level of the system hierarchy consists of the LSI-11/02 microcomputer which serves as a supervisory computer, while the lower level of the hierarchy consists of the 6503 Rockwell μPs and the remaining hardware just mentioned

The LSI–11/02, or upper level, performs two functions

1 On-line user interaction and subtask scheduling of the user's VAL commands and
2 Subtask coordination of the six 6503 microprocessors to carry out the command On-line interaction with the user includes parsing, interpreting and decoding VAL commands, as well as monitoring possible error messages

The lower level of the hardware hierarchy consists of six digital servo boards, an analog servo board and six power amplifiers The six 6503 μPs, residing on the digital servo boards with their EPROM and digital-to-analog converter (DAC), are an integral part of the joint controller They communicate with the LSI-11/02 computer through a specially designed interface board that routes set-point information to each joint controller

This PUMA 560 hardware suffers from some limitations These have been described by Goldenberg [41]

1 Both levels of the controller hierarchy contain only fixed point processors
2 The existing memory in both levels is inadequate to support large programs.
3 The instruction speed of the Rockwell 6503 μP and the LSI 11/02 are inadequate

to implement computationally complex control algorithms, and finally,
4 It is impossible to add additional sensors to the robot, such as vision and tactile sensors, without a complete redesign of the lower level

From this list of limitations it can be seen that if a more flexible hardware control structure is required, capable of implementing complex real time control, then the existing Unimation controller hardware must be replaced with a more flexible alternative

8.1 The New Control Hardware Structure

The PUMA 560, because of its two distinct hardware levels, offers what is known as a decentralized control structure Such structures have been widely accepted [42] by the robotics industry due to ease of implementation and tolerance of failure The main advantage of such a structure is that it allows for easier implementation of the control layers discussed in Chapter 1 For this reason, it was decided that the new hardware structure should be mainly decentralized, with the possibility of implementing mutivariable control Together with this structure, the new control structure offers the following

1 Floating point processors to perform mathematical calculations with high precision and at high enough speed for real-time control
2 Interfacing hardware which is compatible with the existing Unimation hardware
3 Software that can be written in a single high-level language
4 A memory capacity suitable for large program storage
5 An ability to implement multivariable control
6 The ability to provide real time path planning
7 The ability to connect sensory devices through serial, parallel or bus interfaces

Finally, on top of all these requirements, the new control structure is economically viable, and therefore is a realistic alternative to the existing control structure as far as the robot manufacturer is concerned

Numerous implementations of the control structure's upper level, including [43], [44] and [45], have replaced the existing upper level computer with various other machines [46]. More recent implementations such as the TUNIS [44] and SIERA [45] have replaced the existing upper level with powerful personal computers (PCs) Both of these systems are capable of offering the capabilities just mentioned above but at a fraction of the cost For this reason it was decided to use a PC to implement the new upper level [46]

The personal computer chosen was an Intel-based 80386 PC [47] The features which governed the choice of this PC included the presence of

1 A 32-bit architecture (data and addressing)
2 A clock speed of 20MHz
3 The ability to add a floating-point coprocessor (80387)
4 1 megabyte of RAM
5 An 80 megabyte hard disk and
6 Seven parallel expansion slots

From this list of features, it can seen that the new upper level offers a development and storage environment suitable for large program generation. It also offers a fast execution speed for such programs, even if they contain floating-point calculations The expansion slots offer the ability to add extra memory and the ability to interface with the new lower level

To replace the lower level of the controller architecture, it was again necessary to choose a processor with high speed floating-point capabilities A solution which has become more popular in recent years is to use advanced signal processors (ASPs) to implement this level The reasons for their rise in popularity include the reduction in operation and development time which they offer, and recent advances in VLSI technologies have meant cheaper ASP chips [48]

8.1.1 The ASP Card Features

It was decided to use an ASP configuration to implement the lower level of the controller because of the reasons above The ASP chosen for this level was the NEC µPD77230 [49] The µPD77230 can execute arithmetic operations with 32-bit, floating point data (8 bits for exponent and 24 bits for mantissa) or 24-bit, fixed-point data at 150ns per instruction Its internal circuitry comprises a multiplier (32 x 32 bits), an ALU (55 bits), an instruction ROM (1K by 32 bits) and one pair of data RAM pointers (512 words by 32 bit each) The processor itself can be used in either of two modes: *master* or *slave* For this application three PC compatible boards, operating in master mode, were purchased from LSI [50] By operating in master mode, the processor's instruction area occupies 8K words by 32 bits of memory In addition, it allows for 3-stage pipelining and provides a dedicated data bus for internal RAM, a multiplier and an ALU Such an arrangement makes the processor suitable to process algorithms in which a few operations (such as addition of terms) occur repeatedly [51] These are the type of operations that occur in the more complex control algorithms such as the computed torque method [52] In [52] it was found that a single µPD77230 was capable of achieving throughput rates of 1,350 setpoints per second and by utilizing the pipelining nature fully it was found that this algorithm could achieve a throughput of 2,220 setpoints per second These figures produce controller sampling of 0 740ms and 0 450ms respectively These sampling rates are much faster than the existing controller which implements a much simpler PD control algorithm These timing statistics mean that a µPD77230-based lower level is well capable of implementing real-time control algorithms for robotic control

8.1.2 The Analog I/O Card

The analog boards used, supplied by LSI [55], each support 4 analog input channels, two analog output channels and a sample rate timer All of these channels have 12-bit resolution The four analog input channels have a fast conversion time of 5µs, while the two output DACs have a settling time of 3µs One of the input channels present is used for reading the feedback potentiometer, while one of the output channels is used to drive the motor amplifier The reason why there are more I/O channels than necessary is to make the controller more flexible - other sensors such as vision or tactile sensors can be attached to any joint at a later stage if required

The sample rate timer on this board consists of a 16-bit reloadable up-counter which is clocked by an 8MHz clock. This timer, upon completion of a sample period, has the ability to interrupt both the upper and lower levels of the controller hardware In the case of the PUMA 560, it must be possible to generate these at intervals of between 125ns and 30ms These are well within the range of the sampling periods necessary for real-time control of the PUMA 560

8.1.3 Interfacing The New Control Hardware To The PUMA 560 Unimation System

The µPD77230 processor board has a range of 14 input/output (I/O) parallel expansion ports Each of these ports uses 16 bit wide data. The main interfacing problem was that the µPD77230 board has to have access to both the encoder counter outputs and the analog board. Here the design here involved the use of 74623 [56] octal bus transceiver chips to allow bidirectional data transfer between the interface boards and the lower level of the control hardware The control lines for determining the data transfer direction over the new interface are derived by decoding the 14 I/O address lines as shown in Table 8 2

In addition to the I/O ports the µPD77230 board has a number of digital I/O lines which are used to complete the interface These lines consist of two output lines and two input lines One of the output lines, FLAGOUT, is used to generate the BRAKE RELEASE ENABLE SIGNAL, while the other, BIT OUT, is used to generate the ARM RESET signal for the reset circuit. The input line, BIT IN, is used to monitor the ARM STATUS line to see if an index has occurred

8.2 Design of the New Interface Card

This section details how the specifications described above are used in the design of the new controller interface From the above specification, it can be seen that the interface circuitry is a collection of the following subsystems

1 An encoder counter circuit
2 An encoder reset circuit.
3 An analog input subsystem with a sample rate generator and
4 The interface with the new lower level hardware

The control hardware designed and implemented in this project, (see Fig 8 1), consists of three basic elements - the host computer, the processor boards and some special purpose interface hardware The function of the digital computer is to implement the upper levels of the control hierarchy presented in Chapter 1, while the processor boards present implement the lowest level of that hierarchy The function of the interface hardware is to provide a link between the digital hardware of the new controller and the analog inputs and outputs necessary to control the PUMA 560 industrial robot.

The control of a PUMA 560 arm is achieved through the control of the joint d c motors The inputs necessary to control the PUMA 560 [53] are the input voltages used to drive the motors and the voltage signal necessary to apply motor brakes. The robot outputs necessary for control are the outputs of the potentiometer and incremental encoders, which are position feedback measurement devices

The incremental encoders located in the joints of the PUMA 560 each produce three signals for measuring the joint position of the robot - an A channel, a B channel and an Index channel The A and B channels, see Fig 8 2, each produce a squarewave output, with one channel leading the other by 90° By counting the state changes (0→1 or 1→0) of both channels, the magnitude of a joint movement relative to some initial joint position, can be determined It is also possible to know the direction of movement by observing which channel is leading and which is lagging

The Index channel, in conjunction with the position potentiometer, is used to find the initial position The index channel produces a pulse on every motor rotation An Index pulse is produced at regular intervals and each of the intervals is some multiple of the number of degrees in one motor revolution. The potentiometer is used to determine which multiple The position potentiometer used is coupled to the motor shaft, through a gear train, so that the angle read by the position potentiometer corresponds directly to the joint angle The potentiometer is prone to inaccuracy, and this is why it cannot be used on its own to determine absolute position The inaccuracy, however, in the potentiometer reading is much less than ±1/2 of a motor revolution So if the potentiometer is read at an index pulse, the absolute position can be interpreted to be the nearest multiple of motor revolutions to the potentiometer value read

The initialization of the joint angle measurement for the PUMA 560 can, therefore, be achieved by using the feedback sensors in the following manner

1 The joint motor is rotated until an index is found
2 The motor is then halted.
3 The potentiometer voltage is read, converted to degrees and stored
4 The decoded relative positions of the A and B channels are set to zero

Any subsequent movement of the joint will cause an increase or decrease in the decoded values of the A and B channels. This decrease or increase, when converted to degrees, can be added to the stored potentiometer value to produce an accurate joint position

Having outlined the steps necessary to determine the joint position, the next step is to describe in more detail the design which was required to implement these steps The required design comprises of four main areas

1. Reading the incremental encoders
2 Reading the potentiometers

3 Driving the DC motors
4 Applying the motor brakes

These basic design requirements are discussed in the following subsections

8.2.1 The Incremental Encoder Counter System

The optical encoders are directly attached to the motor shaft, and, because of the gear coupling, they rotate several times when the joint is driven through its full motion This gives a precise measurement of relative motion [54] The A and B channels determine both the amount, and the direction, of the rotation in discrete steps The index channel produces a short pulse on each motor revolution (360°), which can be used by the system, in conjunction with the position potentiometer value, to determine absolute position

The A and B channels detect the relative motion of the joints The direction of rotation (clockwise or anti-clockwise) can be determined by observing the state transitions on these two channels These transitions can be interpreted to perform three operations

1 Increment joint position (A leads B)
2 Decrement joint position (B leads A), and
3 Remain at same position (no state changes)

Almost all the PUMA 560 joints [54], with the exception of joint 2 which has 800 state changes per revolution, produce 1000 state changes per motor revolution Since the motor rotates between 40 and 60 times (again joint dependent) during a full joint rotation, 40,000 to 60,000 state transitions occur in that joint rotation Any counter circuit used to keep track of these transitions should be able to hold the maximum number of transitions that are likely to occur For this reason 16-bit counters (maximum count 65526) are sufficient to keep track of the PUMA 560's joint movements

The PUMA 560 position potentiometers are incorporated into the joint motors and are connected between +5 volts and ground Rotating the potentiometer through 360° produces a proportional voltage output of between 0 and +5 volts The potentiometers themselves have been geared to rotate less than 360° during a complete joint rotation In some cases the full movement of a joint could be as little as 200° and, as a result, the change in the potentiometer voltage would be about 2 78 volts. Since on average 60 index pulses are produced over the entire joint sweep, then the potentiometer voltage must be measured to an absolute accuracy of 1/60th of 2 78 volts (0 046 volts) per motor revolution

A 16-bit up-down counter, consisting of four 4-bit cascaded counters, is used to count the number of encoder state changes The counters in question have four controls - a count up/down, an enable input, a clock input and a load input The truth tables for these signals can be found in [56] This counter uses a 1MHz clock which is generated on the new interface card by a 1MHz crystal This value of clock frequency was chosen because it is much greater than the maximum frequency of the encoder state changes

The enable and up-down signals of the counter are derived from the A and B channel signals of the encoders The counter is incremented or decremented when the encoder goes through a state change These state changes are asynchronous and must be synchronized by the decoding logic The basic idea of the scheme is presented here and illustrated in Fig 8 3 From Fig 8 3 it can be seen that the encoder signals A and B are both fed through 2-stage shift registers clocked by the 1MHz clock The outputs of the first stage (A', B') are synchronized versions of the A and B inputs, since they are clocked by the 1MHz clock signal Similarly, the outputs of the second stage (A'', B'') are synchronized versions of A' and B' It is useful to think of the first stage outputs (A', B') as the *present* states and the outputs of the second stage (A'', B'') as the *previous* state Together the four states, A', B', A'' and B'', make up 16 (2^4) possible state combinations which can be decoded to determine which direction the count must go - up or down Table 8 1 shows all the possible combinations of these states and the decoded command signals for the counter. Rather than use logic gates to directly implement the decoder it was decided to use an EPROM. This EPROM has the A states and the B states, and the counter reset line as its address inputs The outputs are the decoded command signals for the counters generated from Table 8 1.

Table 8.1 Counter Control Commands

EPROM ADDRESS	PROM O/P (COUNTER I/PS) ENT	D/U	LOAD	OPERATION
0	1	1	1	NOP
1	0	1	1	DEC
2	0	0	1	INC
3	1	1	1	NOP.
4	0	0	1	INC
5	1	1	1	NOP
6	1	1	1	NOP
7	0	1	1	DEC
8	0	1	1	DEC
9	1	1	1	NOP
10	1	1	1	NOP
11	0	0	1	INC
12	1	1	1	NOP
13	0	0	1	INC
14	0	1	1	DEC.
15	1	1	1	NOP
16 → 31	0	0	0	CLEAR

8.2.2 The Control Output Signal

The drive current and voltage needed to drive a DC motor is entirely motor dependent. It is therefore not necessary to design power amplifiers for the system, since satisfactory ones already exist Instead it was considered practical to use the existing ones and to concentrate on the hardware necessary to drive the amplifiers In the case of the PUMA 560, the existing power amplifiers [2] can be conveniently used because they were designed explicitly with this robot in mind Using these amplifiers simplifies the external connections to the arm's joint motors In addition, the Unimation power amplifier unit contains a Miscellaneous Functions Unit [2] (MFU), which provides useful safeguards that can be monitored to prevent damage to the arm These safeguards include the ability to monitor the amplifier's input current and temperature to see if they are operating within the values specified for that amplifier manufacturer

The PUMA 560 power amplifiers are controlled by analog voltages These voltages can be generated by digital to analog converters (DACs) Two basic specifications must be considered in the choice of DAC - voltage swing and resolution The PUMA 560 power amplifiers require a voltage input swing of 10 volts to -10 volts Selection of resolution is more difficult Typical digital servo systems use 8 or 10-bit DACs - the Unimation uses 10-bit. It was decided to increase this to 12-bit for this project This increase in the resolution means that the new drive signal is four times more accurate than the original one

8.2.3 The Data Direction Control System

To solve the communication in the lower level, four tristate octal transceivers are employed The transceiver allows data to flow in both directions by correctly setting the two control input lines (G_{ab} and G_{ba}) Enabling G_{ab} (=1) and disabling G_{ba} (=1 active low) allows data to pass from A to B Setting both these values low, allows data to pass from B to A The chip can also be set to a high impedance state, where no link exists between A and B

To allow the digital signal processor to communicate between the new interface card and the 4 channel analog card, two sets of transceivers are placed on the counter outputs and on the 16bit bus from the analog card The outputs of both these sets of transceivers are connected together using pull-up resistors If a READ or WRITE is performed using one set, then the other set is set to high impedance. Table 8 2 shows the settings of the control signals required to perform the desired operations Fig 8 5 shows a schematic diagram of the circuit used to achieve this data control

TABLE 8.2 Data Direction Control

	I/O PORT ADD.				Transceiver Control Signals			
Operation	A_3	A_2	A_1	A_0	G_{ab1}	G_{ba1}	G_{ab2}	G_{ba2}
NOP	0	0	X	X	0	1	1	1
READ COUNTERS	0	1	X	X	1	1	0	1
READ ANALOG	1	0	X	X	0	1	1	1
WRITE ANALOG	1	1	X	X	0	1	0	0

where X don't care

8.2.4 System Timing

The PUMA 560 brake is used to lock each joint in position when the motor power is turned off This prevents the joints from collapsing when no power is present to hold them in position It is impossible to individually apply or release the brakes of the PUMA 560 This is due to the fact that the brakes of each joint motor [2] are wired together The MFU mentioned above contains the circuitry needed to apply or release the brake This circuitry can be controlled by setting or resetting a digital input of the MFU known as BRAKE RELEASE ENABLE

The joint interface circuitry must not only accommodate the joint motor signals but it must also provide the upper and lower hardware levels of the new controller with additional functions to allow complete system integration. The single most important of these functions is system timing

Implementation of a digital controller requires some means of regulating a sampling interval A hardware timer is used to interrupt the CPU The hardware timer can take the form of a programmable up-counter This counter should be free-running from an NHz clock giving a clock period of 1/N seconds The sample period can therefore be set in terms of an integral number of clock cycles, each clock cycle adding 1/N seconds A program that requires this sampling interval can then be written as an interrupt service routine Then, if a timer interrupt occurs, the CPU will be interrupted and the program can commence

The hardware scheme of Fig 8 4 is used to monitor the index pulse for initialization Each new counter reset circuit has two flipflops and a NAND gate The circuit is asynchronously *armed* or enabled via an ARM RESET signal Once armed, the next index pulse occurrence generates a single reset pulse, which is sent to the associated counter circuit. When the reset pulse is issued, the circuit disarms itself so that further occurrences of the index pulse will not reset the counters The ARMED STATUS signal can be monitored by the system software to see if the index has occurred

8.3 Software Considerations for the New Control Structure

The purpose of this section is to provide an insight into the computational aspects of the new PUMA 560 control structure. The new hardware configuration is a hierarchical, multi-processor system, and as a result it requires a considerable amount of inter-processor communication to perform its robot control function. Fortunately, since the two levels in the new PUMA 560 controller are "off-the-shelf" items, existing software tools can be used to achieve the designed inter-processor communication desired

This type of robot control hardware, with a personal computer as the upper hardware level, allows for easier implementation in both industrial and educational environments This is due to the general familiarity with the personal computer operating system and hardware By using a commercially available operating system with the robot control hardware, one can speed up the development process and the learning curve of potential users, since features such as file management, batch file generation and on-line debugging tools are available

The software tools used for the new controller consist of a Microsoft C language compiler,and an NEC µPD77230 monitor [57] with linker, assembler and object converter facilities The choice of this C compiler was dictated by the fact that the µPD77230 processors can use a Microsoft C compatible compiler for program development The µPD77230 C compiler is used to convert C language programs into µPD77230 assembly language programs This assembly language can then be converted to hexidecimal format using the object converter In this format, the programs can be downloaded into the µPD77230 memory space and then executed The downloading and execution can be achieved by using either the monitor or C drivers specifically written for this purpose, or by using the monitor which comes with the board

The computational elements of the new control structure involve a wide range of applications, including the roles of the operating system and programming language just discussed In addition to these roles, the processors of the new system are used to drive the joint servos and to interface with external position sensors The following sections are concerned with the functionality of the computational elements of the new controller under the headings of interface, communication, and calculation.

One role of the computational elements of the new control hardware is to provide communication, i e exchange of information between and among components In the case of the new control structure, these components are the upper and lower hardware levels This communication involves downloading position setpoints to the µPD77230 boards from the personal computer The µPD77230 boards are mapped to the input/output addressing area of the personal computer The address map of each µPD77230 board takes up 8 addresses in the personal computer input/output area The function of the control register is to enable or disable the processor and any interrupts to the personal computer, and the status register is used to monitor the operation of the µPD77230

The calculation functionality of the new hardware can be defined in terms the speed at which the basic operations such as add, subtract, divide and multiply can be performed on fixed and floating point data For the personal computer the fixed-point operations were found to take 3 clock cycles to execute (i e 150ns) Double precision floating point additions were found to take 10μs, and multiplications took approximately 32μs each

In the lower level, computational functionality involves the μPD77230 board's ability to perform floating and fixed point addition, subtraction, division and multiplication For fixed point data these calculations were found to take 1 instruction cycle or 150ns, [57] In the floating-point case, addition and subtraction each take 5 instruction cycles, and multiplication takes 6 instruction cycles This means that the lower level is capable of performing thousands of additions and multiplications per millisecond The advantage can be seen more clearly if one examines the algorithms developed in [58],[59] and [60] These algorithms are among some of the most computationally complex available, yet preliminary calculations suggest that these algorithms can be implemented in real-time using the μPD77230 boards In the case of [58] and [59] these calculations show that both algorithms could implemented in times less than 0 5ms, while [60] could be implemented in a time less than 0 8ms The same algorithms, if implemented on the existing Rockwell 6503μPs, would require that the sampling interval be increased by a factor of 10 Such high sampling intervals are unsuitable for real-time control

8.4 Identifying the Robot Parameters

The new hardware system is used to capture the control commands and joint positions The control commands from the amplifier are read using the four channel analog card These commands are stored on the dsp card, in memory, but are later echoed back to a file on the pc The joint positions are read using the new interface card The counter circuit determines the movement of the joints and sends this information to the digital signal processor The captured input/output data is shown in Fig 8 6 and Fig 8 7 respectively

8.4 1 Identification Results

Using the input/output data captured, it is possible to identify a model for this data using Recursive Least Squares A second order model is estimated, where four parameters are determined for each joint, i e the model takes the form

$$G_{es}(z) = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$$

A variable forgetting factor (μ) is used to allow for both fast initial convergence and small oscillation of the final parameters. Initially, μ is set at 0.8, and increases exponentially to 0.995 at the end of the test. P_0 is set at 1,000. The results of the identification are shown in Fig 8.8 through to Fig 8.13. These results here can be used to validate the simulation model developed in Chapter 2.

Looking at Fig 8.8, a pole at 1.0 is found to exist, and the other pole is at 0.9. These findings are very similar to the results obtained from the simulation model, where the model consists of an integrator and another pole close 0.9. The poles of the system do not vary hugely with changes in joint positions. The zeros of the plant widely vary with changing position, so the results in Fig 8.9 can be compared with the results from before, when a different reference signal was used. However, the zeros from the simulation model and those from the actual robot are close in magnitude. Similarly, the other joints' parameters are found to be close to the results in Chapter 5. Using these parameters, a time varying second order model can be constructed to simulate the dynamics of the robot joints.

8.5 Simulated Control of the Identified System

From the evaluation (in Chapter 7) of the simulation control section, one control routine is chosen as the best controller in each category. It is the routine which outperforms the other algorithms. The three algorithms chosen are PID, the Self-Tuning Regulator and Computed Torque (with an adaptive feedback layer). To investigate which of these algorithms is suitable for control of the actual robot, each of these three algorithms is used to control the time varying, second order model, derived from the above section. These control results are a strong indication of the *optimal* manipulator control technique.

8.5.1 Parameter Algorithm - PID Control

The PID controller does not perform very well in this test. The initial parameter estimates cause the controller to give an undesirable initial response (see Fig 8.14). When the joints track the specified path, there is a noticeable static error for joint 2.

8.5.2 Adaptive Control Algorithm - Self-Tuning Regulator

The STR performs very well - no undesired initial behaviour is experienced (see Fig 8 15) The control parameters are derived from the parameter estimates There is a little oscillation present, but the static error is very low

8.5.3 Feedforward Control Algorithm - Computed Torque with Feedback

This algorithm does not perform to expectations The response is similar to the PID results (see Fig 8 16) However, the response of joint 2 is improved The initial variation in the joint angles is present.

8.5.4 Conclusion

It is clear that the STR performs best in this test This routine takes the parameter estimates and transforms them to controller gains The PID is the least efficient here The Computed Torque method is second best to STR However, when Computed Torque is used in an actual implementation, the scenario is different. The results from the identification test are not required, the algorithm requires no parameter estimates The identification results are only suitable for adaptive routine use in this scenario However, if the inverse dynamic model of the robot is not precise enough, then undesirable results may be obtained if Computed Torque is used. The STR is the most flexible, no internal model is used, and the tuning is easily changed In conclusion, the STR method gives the most desirable results

8.6 Summary

This chapter shows how the new control hardware is designed and interfaced to the existing Unimation System The new interface is similar to the existing Unimation because it uses the Unimation power amplifiers and MFU It also adds a degree of flexibility to the new control hardware which is not found m the Unimation interface The flexibility it provides lies m the increased input/output capabilities and in the provided accuracy that it provides over the existing input channels It also provides a flexible sample rate timer which is capable of producing sample rates in a range suitable for real time control

This chapter also investigates the identification of the robot parameters Using data captured from the robot and the RLS identification technique, the robot parameters can be estimated These identification results are used to simulate the control of the actual robot system Conclusions are made as to which algorithm is the most suited for manipulator control, based on the results found in this chapter

Index to Graphs

Input/Output Data

Identification Results

Control Results

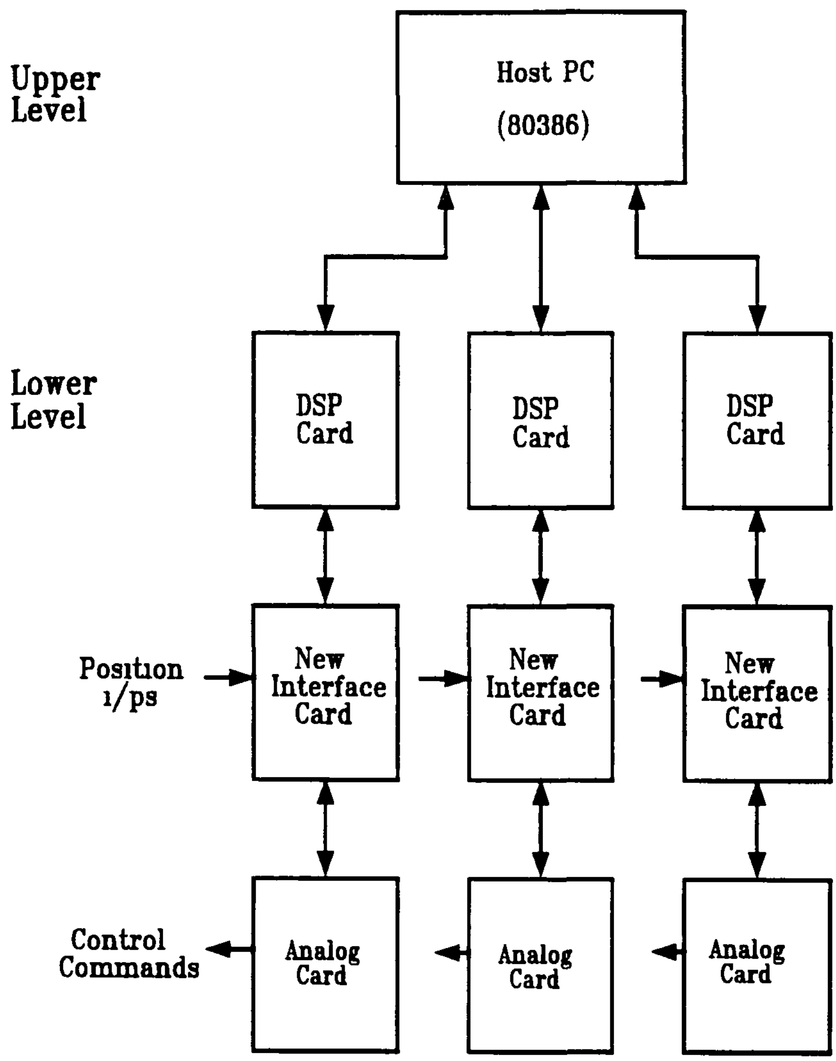


Fig.8.1 The New Hardware Structure

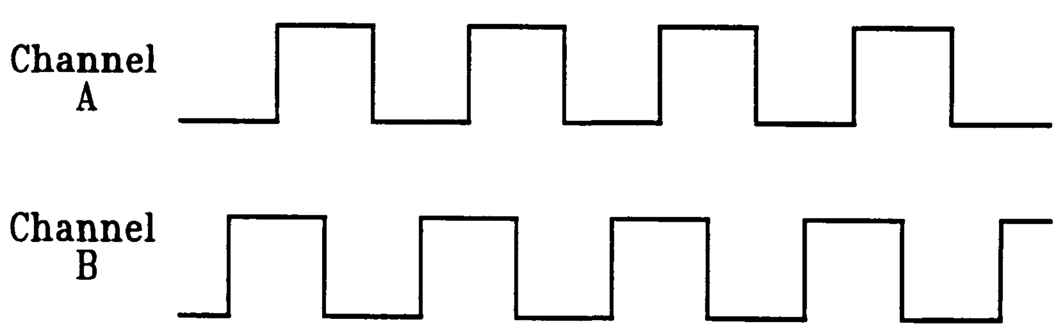


Fig.8.2 Incremental Encoder Pulses

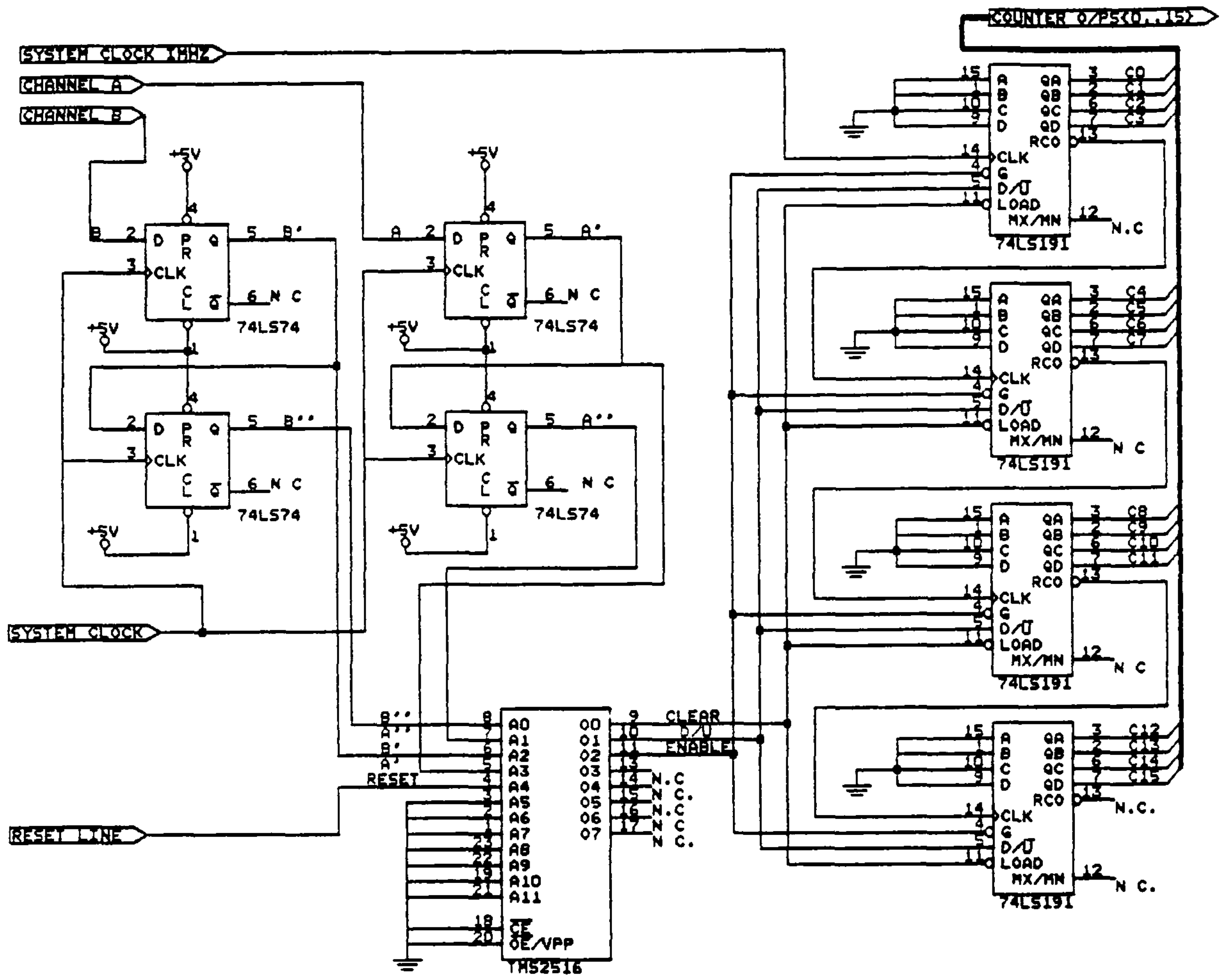


Fig.8.3 *Counter Circuit for determining Joint Position*

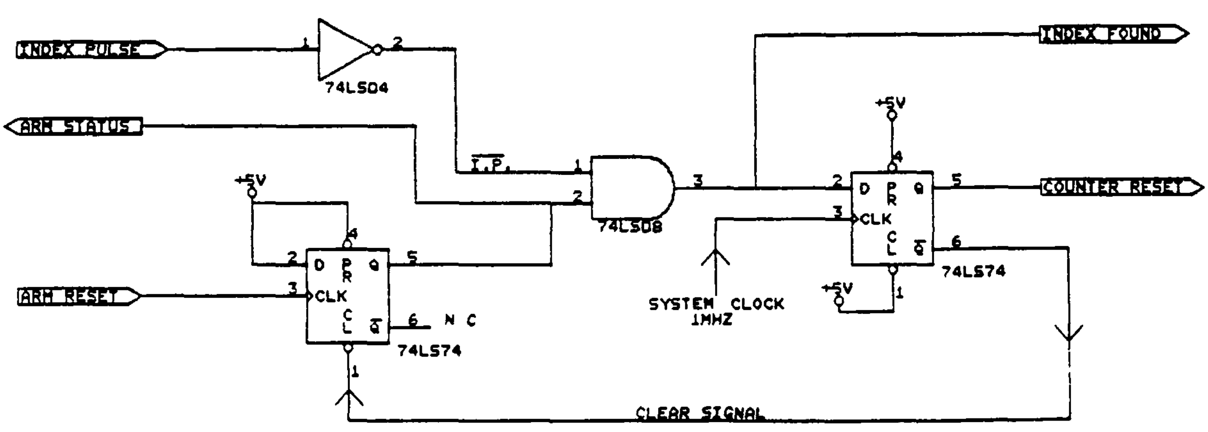


Fig.8.4 *Reset Circuit*

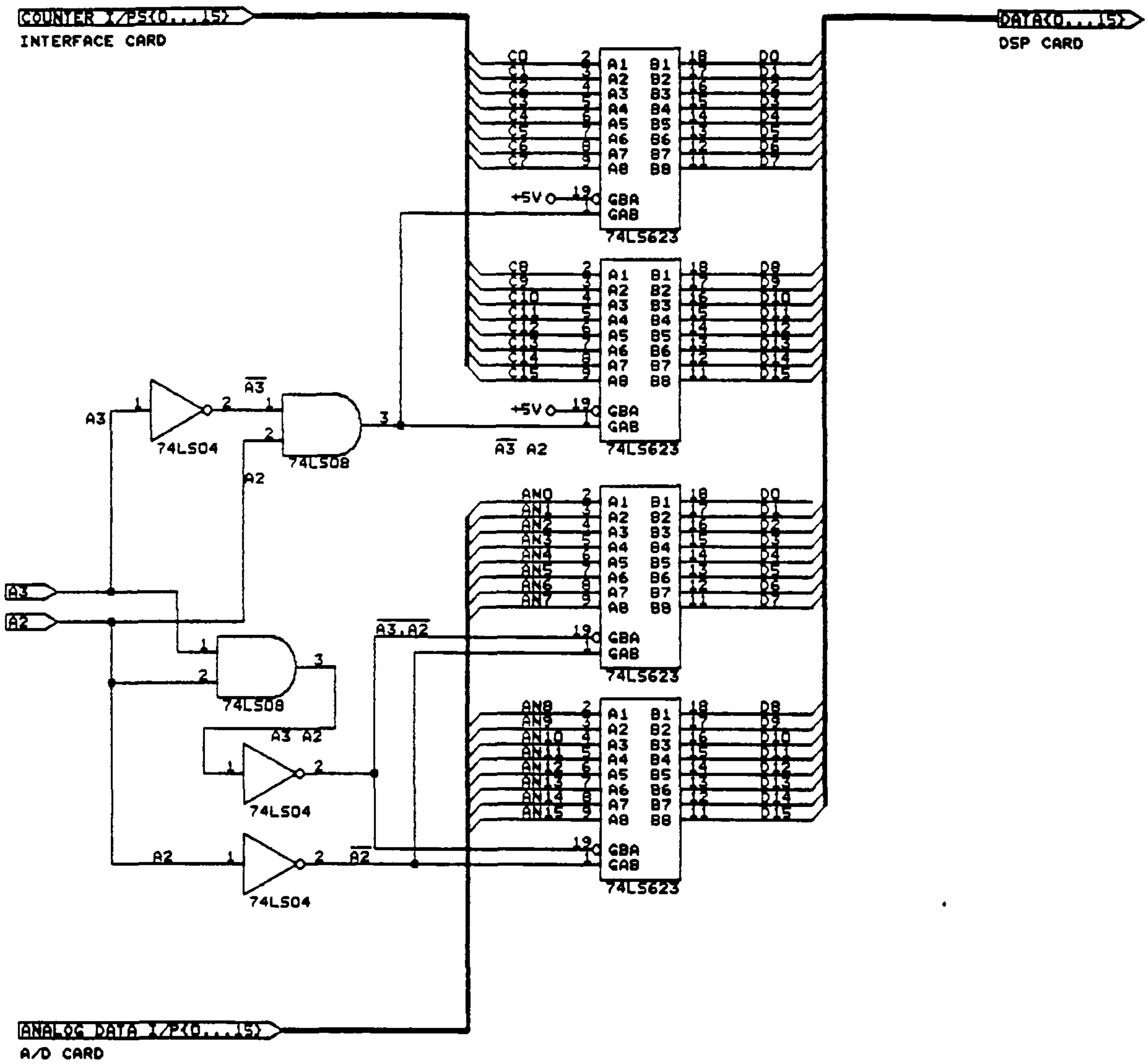


Fig.8.5 *Data Direction Control Circuit*

Fig 8 10

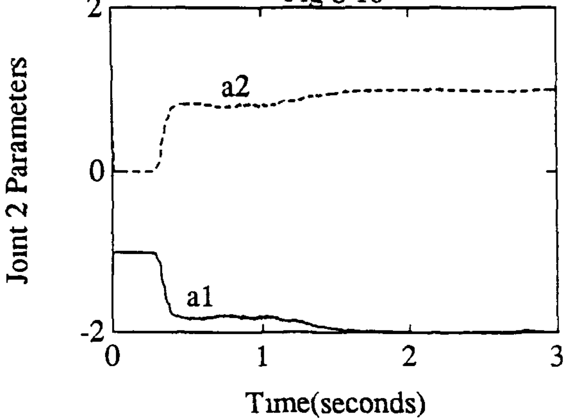


Fig 8 11

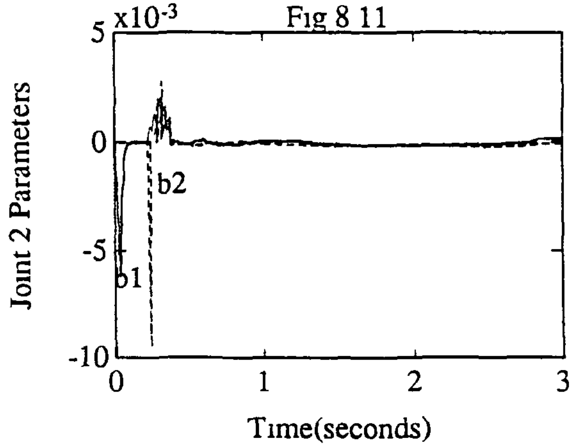


Fig 8 12

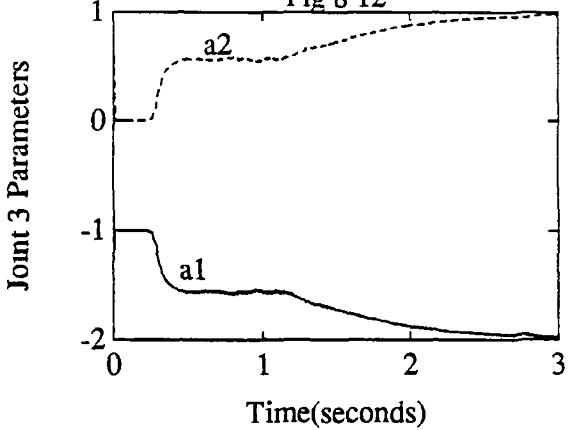


Fig.8 13

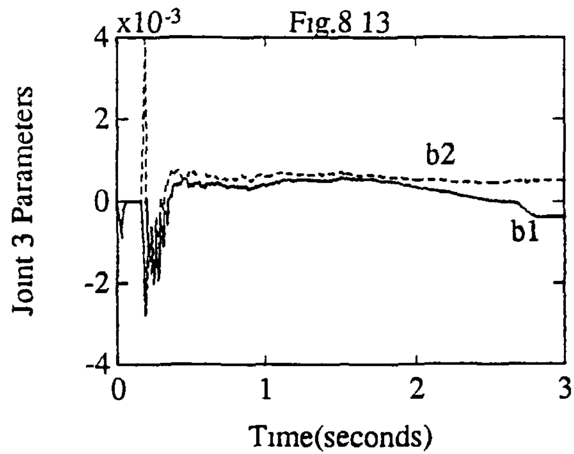


Fig 8 6

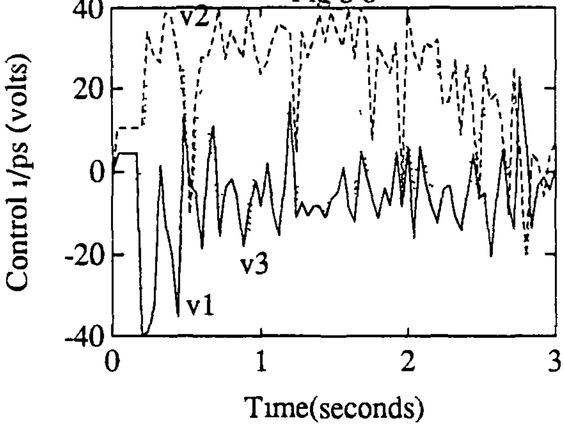


Fig 8 7

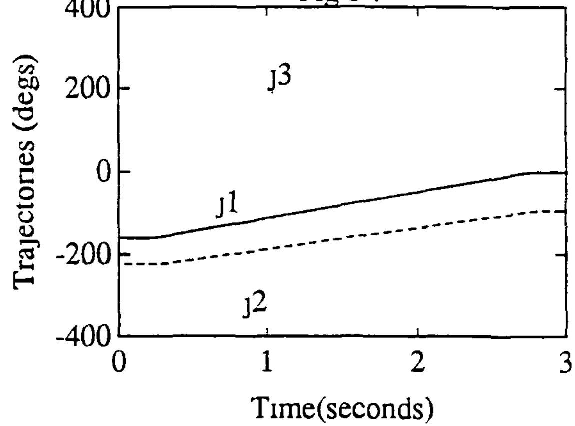


Fig 8 8

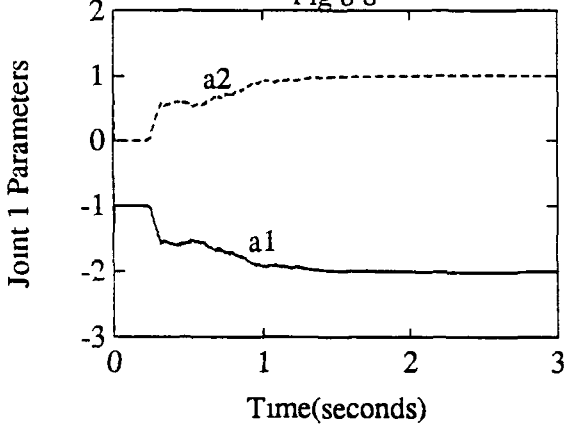


Fig 8 9

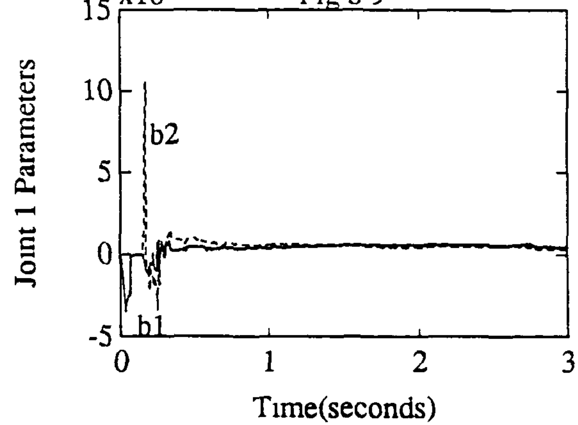


Fig 8 14a

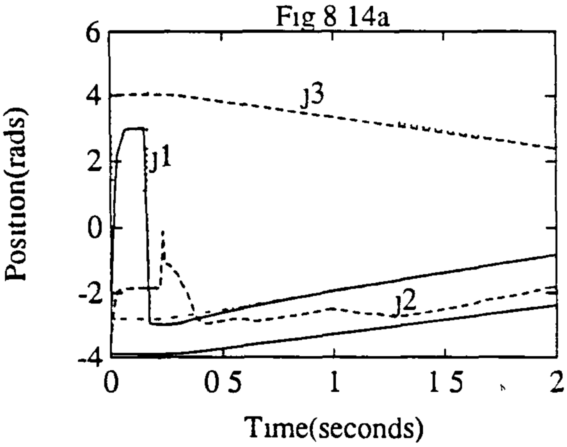


Fig 8 14b

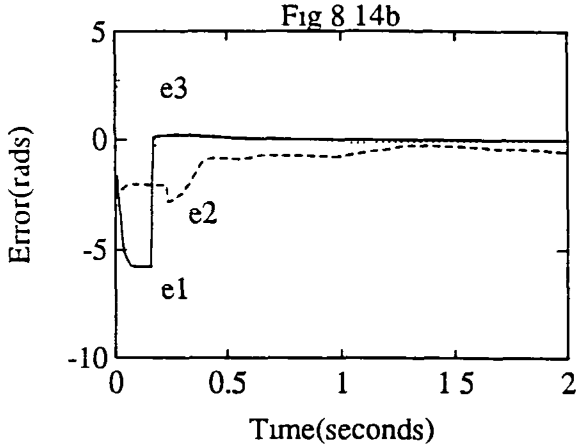


Fig 8 15a

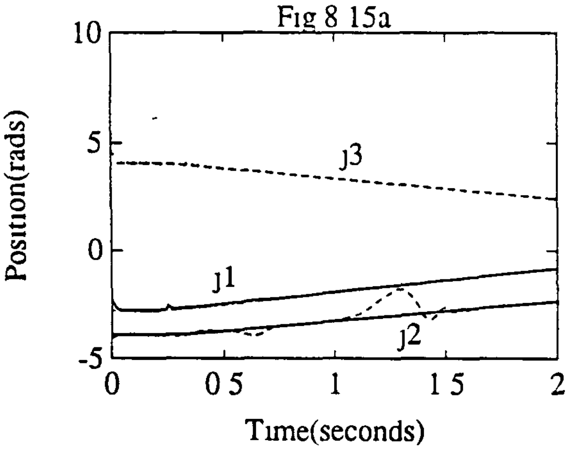


Fig 8 15b

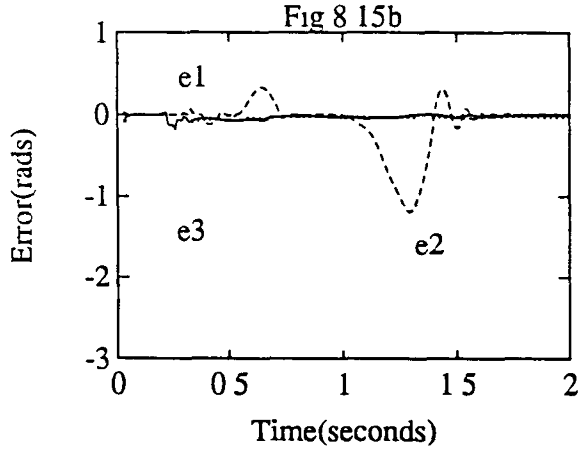


Fig 8 16a

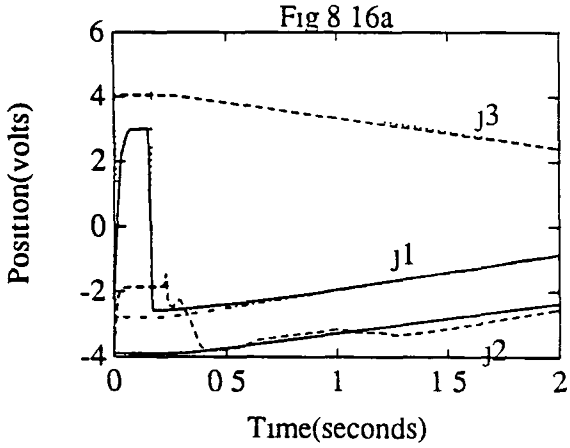
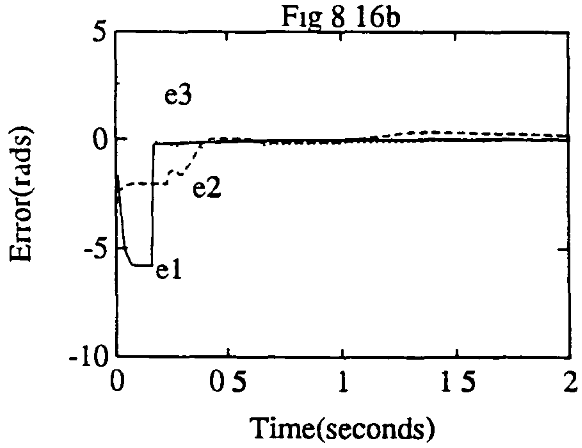


Fig 8 16b



CHAPTER 9

CONCLUSIONS

This thesis can be broken down into three subsections The first subsection is concerned with the topics in Chapters 2 and 3, where the dynamics for the three primary joints are explained and a simulation package is designed to implement these dynamic equations In Chapter 3, the forward and inverse solutions to the kinematics problem are detailed, along with several techniques for trajectory generation These two chapters serve as an introduction to the background work, which is used at a later stage in the thesis

The second section of this thesis is concerned with the area of robot control Three main types of control techniques are used The performance of these algorithms is simulated in a robot environment using the simulation package designed in Chapter 2 Evaluation of their performance is based on several performance criteria

The final section of the thesis details the hardware side of this project It also includes the results of an identification performed on a PUMA 560, using this hardware system to capture the input/output data The design is aimed at producing a flexible working environment, where new control techniques can be readily investigated on the robot.

9.1 What was achieved

The aim of this project was to perform an investigation of a wide range of control techniques, suitable for manipulator control, and to simulate their performance using the robot model From the simulation results, the best suited algorithms can be chosen for real time implementation on the PUMA 560 robot The successes of this research are in the areas of robot modelling, hardware design and the analysis of an extensive range of control algorithms

A complete dynamic model has been developed for the three primary joints of the PUMA 560 industrial manipulator The Euler-Lagrange formulation models the manipulator as a set of second order differential equations Incorporating the actuator dynamics into these equations results in a third order model with voltage inputs and position, velocity and acceleration outputs Simulation is performed using the Runge-Kutta numerical integration technique to solve these differential equations

A wide range of control algorithms has been investigated, from the classical techniques of PID and Optimal Control to the newer methods of Predictive Control Adaptive and Feedforward strategies are also of interest An evaluation of these algorithms is performed to grade the algorithms according to their performance

The complete design and implementation of a hierarchial control structure, using special purpose processors for the control of the three primary joints of a PUMA 560 has been presented in this thesis Using a personal computer as a host machine, with attached digital signal processor boards, the old hardware of the Unimation system can be replaced with this new arrangement The digital signal processors are very powerful and are capable of implementing complex control algorithms such as Computed Torque, for example These DSP boards form the new lower level of the controller's hierarchy, of which the 80386-based personal computer forms the upper level

Also, the solution to the forward and inverse Kinematics problem is given, along with several techniques for Path Planning These serve as introductory material for the reader

9.2 What was not achieved

Real time control of the robot was not performed, only simulated control of the identified model was achieved However, this strongly indicates which of the control routines is most suitable for manipulator use Because real time control was not performed, small modifications may be necessary to the overall system

9.3 Summary

This project achieved considerable ground in the area of robotic research Topics such as Robot Dynamics, Kinematics, Path Planning, Robot Control, Identification techniques, and Hardware Design for robot systems are discussed in this thesis A suitable selection of control algorithms exist, and the hardware system, which is capable of implementing these in real time, is now available at D C U Engineering School This project has reached nearly all its goals The simulation side of the project is very comprehensive, spanning a wide range of control methods Future work into robotics at this University should be aimed at the implementation of the techniques conceived in this research

REFERENCES

REFERENCES

[1] Vukobratovic, M & Stokic, D, "*Control of Robot Manipulation Robots Theory and Application*", Springer-Verlag Berlin, 1982

[2] Unimation, "*PUMA 500 Mk2 Electrical and Mechanical Drawings*", July 1985

[3] Craig, J J, "*Adaptive Control of Mechanical Manipulators*", The International Journal of Robot Research, Volume 6, No 2, Summer 1987

[4] Hollerbach J M, "*A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity*", IEEE Transactions on Systems, Man and Cybernetics, Vol SMC-10, No 11, Nov 1980

[5] Bejczy A K, "*Nonlinear Feedback Control of the PUMA 560 Robot Arm by Computer*", Procs of the 24th Conference on Decision and Control, Ft. Lauderdale, Fl, Dec 1985

[6] Goldstein H, "*Classical Mechanics 2nd edition*", Addison–Wesley Publishing Company, Student Series, 1980

[7] Jones F, "*Modelling, Simulation and Identification of an Industrial Manipulator*", D C U, M Eng Thesis, 1990

[8] Anderson G P, "*Modelling, Simulation of a PUMA 560 Manipulator for Control System Appraisal*", N I H E Dublin, M Eng Thesis, 1988.

[9] Lee C S G & Lee B H, "*An Efficient Formulation of Robot Arm Dynamics for Control Analysis and Manipulator Design*", Tech Report TSD-TR8-82, Center for Robotics and Integrated Manufacturing, University of Michigan 1987

[10] Fu K S, "*Robotics Control, Sensing, Vision and Intelligence*", McGraw-Hill International Editions 1987

[11] Denavit J, & Hartenberg R, "*A Kinematic Notation for Lower Pair Mechanisms based on Matrices*", Journal of Applied Mechanics, Vol.77, pp215-221, 1955

[12] Taylor P M, "*Robotic Control*", MacMillan Education 1990

[13] Armstrong W M , "*Recursive Solutions to the Equations of Motion of an H-Link Manipulator*", Procs of the Fifth World Congress, Theory of Machines, Mechanisms, Vol 2, July 1979

[14] Paul, R P , "*Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm*", Memo AIM-177, Stanford Artifical Intelligence Laboratory, Palo Alto, California, 1972

[15] Astrom, Karl J & Wittenmark, Bjorn, "*Computer Controlled Systems*", Prentice-Hall 1984

[16] Kim, Jong-Hwan & Choi, Keh-Kun, "*Design of Direct Pole Placement PID Self–Tuners*", IEEE Transactions on Industrial Electronics, Vol IE-34, No 3, August 1987

[17] Nagrath, I J & Gopal, M , "*Control Systems Engineering*", Wiley, 2nd Edition 1982

[18] Chen, Yilong, "*Replacing a PID Controller by a Lag-Lead Compensator for a Robot - A Frequency Response Approach*", IEEE Transactions on Robotics and Automation, Vol 5, No 2, April 1989

[19] Ringwood, J V & Grimble, M J , "*An Optimal Output Feedback Solution to the Strip Shape Multivariable Control Problem*", IASTED Conf on Applied Control and Identification, Copenhagen, Denmark, June 1983

[20] Klafter, D & Chmielewski, T A & Negin, Michael, "*Robotic Engineering - An Integrated Approach*", Prentice-Hall International Editions 1989

[21] Leigh, J R , "*Applied Digital Control*", Prentice-Hall International Editions 1989

[22] Gibbs, P , "*Nonlinear Predictive Control Using System Inversion*", B Eng Thesis, D C U , 1989

[23] Richalet, J, Abu el Ata-Doss, S , Estival, J L & Karam, M Abi, "*Model Based Predictive Control - Part I A qualitative description*", Automatica, Nov. 1989.

[24] McKeown, N , "*Adaptation and Implementation of Monoreg Predictive Control Strategy using State-Space Control Theory*", B Eng. Thesis, D C U , 1989

[25] De Keyser, R M C , Van de Velde, G A & Dumortier, F A G , "*A Comparative Study of Self-Adaptive Long Range Predictive Control Methods*", Automatica, Vol 24, 1988

[26] Ljung, Lennart & Soderstrom, Torsten, "*Theory and Practice of Recursive Identification*", MIT Press 1983

[27] Lieninger, G G, "*Self-Tuning Adaptive Control of Manipulators*", Advanced Software in Robotics, Elsevier Science Publishers b v (North-Holland), 1984

[28] Dubowsky, S et al, "*Application of Model Reference Adaptive Control to Robotic Manipulators*", Journal of Dynamic Systems Measurement and Control", Volume 101, 1979

[29] Bejczy, A K , "*Robot Arm Dynamics and their Control*", Technical Memo 33-669, Jet Propulsion Laboratory 1985

[30] Lee, C S G & Lee, B H , "*An Efficient Formulation of Robot Arm Dynamics for Control Analysis and Manipulator Design*", Tech. Report TSD-TR8-82, Center for Robotics and Integrated Manufacturing , University of Michigan 1989

[31] Ortega, R & Kelly, R , "*PID Self-Tuners Some Theoretical and Practical Aspects*", IEEE Transactions on Industrial Electronics, Vol IE-31, No 4, November 1984

[32] Kim, Jong-Hwan & Choi, Keh-Kun, "*Design of Direct Pole–Placement PID Self-Tuners*", IEEE Transactions on Industrial Electronics, Vol IE-34, No 3, August 1987

[33] Asare, H R & Wilson, D G , "*Evaluation of Three Model Reference Adaptive Control Algorithms for Robotic Manipulators*", IEEE Transactions on Robotics and Automation, Vol 3, April 1987

[34] Astrom, K J , & Wittenmark, B , "*Self-Tuning Controllers based on Pole-Zero Placement*", IEE Proceedings, Vol 127, May 1980

[35] Ringwood, J V , "*Control Strategies for Robotic Manipulators*", Proceedings of the IMC-6 Conference on Advanced Manufacturing Technology, Dublin City University, September 1989

[36] Kholsa, P K , "*Real Time Implementation and Evaluation of Computed-Torque Scheme*", IEEE Transactions on Robotics and Automation, Vol 5, 1989

[37] Leahy, M B , "*Industrial Manipulator Control with Feedforward Dynamic Compensation*", Proceedings of the 27th IEEE Conference on Decision and Control, Austin Tx, December 1988

[38] Uchyama, Masaru, "*Control of Robot Arms*", JSME International Journal Series, March 1989

[39] Unimation (Europe) Ltd , "*PUMA 560 Mk2 Robot System Technical Manual*", Sept 1985

[40] Unimation Inc , "*Programming Manual User's Guide to Val2 398T1*", Version 1 1, August 1987

[41] Melidy, A & Goldenberg, A A , "*Operation of the PUMA 560 without VAL*", Robotics Proceedings Robots 9, 1985

[42] Paul, R , "*Robot Manipulators*", MIT Press, 1981

[43] Unimation Inc , "*Breaking away from VAL*", Danbury Connecticut, 1982

[44] Kananzides, P , Wasti, H & Wolovich, W A , "*A Multiprocessor System for Real Time Robot Control Design and Application*", Proc IEEE International Conference on Robotics and Automation 1987

[45] Penny, D , "*Control of the PUMA Robot without VAL*", University of Toronto, RAL Tech Report, April 1985

[46] Gibbs, P , Jones, F & Ringwood, J V , "*A Flexible Electronic Controller for a Manipulator-Type Robot*", FAIM Joint International Conference, University of Limerick 1991

[47] Olivetti, "*M380 User's Guide*", 1989

[48] Guruasavaraj, K H , "*Implementation of a Self-Tuning Controller using Digital Signal Processing Chips*", IEEE Control Systems Magazine, June 1989

[49] NEC, "*Dıgıtal Sıgnal Processors - Product Descrıptıon*", 1989

[50] Loughborough Sound Images Ltd, "*LSI 77230 PC Processor Board Hardware Manual*", Versıon 2, Sept 1988

[51] Kabuka, M & Escoto, R, "*Robot Arm Controller*", IEEE Mıcro, Feb 1989

[52] Khosla, P K, & Kanada, T, "*Experımental Evaluatıon of Feedforward Compensatıon and Computed-Torque Control Schemes*", Proceedıngs of the Amerıcan Control Conference, Seatle WA, June 1987

[53] Bıhn, D G & Steve Hsıa T C, "*Unıversal Sıx Joınt Robot Controller*", IEEE Control Systems Magazıne, February 1988

[54] Bıhn, D G, "*A Unıversal Sıx Joınt Robot Controller*", M S Thesıs, Department of Electrıcal Engıneerıng, Unıversıty of Calıfornıa, Davıs, Calıfornıa 1986

[55] LSI Ltd, "*4 Channel Analog Interface Card User's Manual*", Versıon 2 1, November 1988

[56] Texas Instruments, "*The TTL Data Book for Desıgn Engıneers*", 1988

[57] NEC, "*Dıgıtal Sıgnal Processor Development Tools*", 1989

[58] Astrom, S J, "*LQG Self-Tuners*", IFAC Adaptıve Systems, San Francısco, 1983

[59] Grımble, M, "*Implıcıt and Explıcıt LQG Self-Tunıng Controllers*", Automatıca, Vol 20, No 5, 1984

[60] Lelıc, M A & Wellstand, "*A Generalızed Pole-Placement Self-Tunıng Controller - An Applıcatıon to Manıpulator Control*", Control Systems Centre Report, No 658, UMIST, Manchester, August 1986

[61] Zeıgler, J G & Nıcholls, N B, "*Optımum Settıngs for Automatıc Controllers*", Trans ASME, Vol 65, 1943

[62] Ellıot, J R, "*NASA's F-8 Adaptıve Flıght Control Program*", IEEE Conf on Decısıon and Control, Heuston Texas, 1975