Dublin City University

School of Electronic Engineering.

Control Technology Research Unit.

Thesis Submitted for the Degree of

Master of Engineering.

Modelling, Simulation and Identification of an Industrial Manipulator

by:

Frederick J. Jones BE.

for:

Dr. John Ringwood.

September 1990.

I declare that the research herein was completed by the undersigned

Signed: F. Jones Date: 21st Sept '90

The continued help and encouragement from my project supervisor, Dr. John Ringwood is greatly appreciated. Thanks to all the staff of the Electronic Engineering Department at D.C.U. Thanks also to Patrick Gibbs, Philip Comerford and all the other postgrads for their help and support.

CONTENTS

CONTENTS

CHAPTER 1

Industrial Robots and Their Control

The word robot has it's origins in the Czech word robota, which literally means forced or slave labourer. Webster's dictionary defines a robot as being "an automatic device that performs functions normally ascribed to human beings". By this definition even the humble washing machine is a robot. A more restricted description used by the Robot Institute of America describes a robot as being a "reprogrammable multi-functional manipulator designed to move material parts, tools and specialized devices, through variable programmed motions to perform a variety of tasks". From this definition it is apparent that a robot is a programmable general purpose manipulator with external sensors that can perform a variety of assembly tasks.

An industrial robot, like that in Figure 1.1, can be described as a general purpose manipulator consisting of several rigid links connected by a series of revolute or prismatic joints. One end of this chain is normally attached to a supporting base while the other is attached to to some tool which performs some predetermined task. Mechanically, an industrial robot is composed of an arm and a wrist subassembly which are designed to reach any work piece within its work volume. It's work volume being defined as the area where the robot's arm can deliver the wrist subassembly.

The past 20 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 controlled machines; in space and undersea exploration; in prosthetic arm research; and in the handling of dangerous materials such as nuclear or chemical waste.

The industrial robots used in these tasks are usually simple positioning machines controlled by mini/micro-computers. They execute a given task by playing back prerecorded or pre-programmed sequences of motions. In general, these robots are equipped with little or no sensors for obtaining the information vital to its working environment. Recent research has been directed towards improving the overall performance of the manipulator system. These improvements have largely centred on improving robot control.

1.1 The Robot Control Problem

All the above applications put demands on the design of the robot to be used. The most noticable of these constraints is that the robot must be capable of performing a given task accurately, in real time and at a reasonable operating cost. Since the mechanical design of the industrial robot has varied little in the past two decades, the problem of attaining increased performance is essentially a control engineering problem.

The principle underlying the control of robot systems, like other large-scale systems, is hierarchical in structure. The control hierarchy is most often vertical with each upper control level dealing with wider aspects of overall system behaviour than the lower levels. The higher levels in the hierarchy communicate with their next lowest level to pass along any information this level needs for decision making. The most common of these hierarchical structures has four levels [1] as shown in Figure 1.2. It's levels function as follows:

1) recognition of obstacles in the robot's workspace and the conditions under which a task is to be performed and taking decisions on how the task imposed is to be accomplished.

2) dividing the imposed operation into elementary movements,

3) distributing the elementary movements to each degree of freedom of the robot

4) executing the required elementary movements of each degree of freedom.

All robots have the two lowest levels: a tactical level used to generate the trajectory for each joint and the fourth level that executes these trajectories using actuators incorporated in each degree of freedom. The two upper levels are specific to second and third [1] generation robots. These are robots that are capable of sensing their work environment and use artificial intelligence methods to perform their tasks correctly.

The two lower levels may be realized in various modes, and their ability to implement the motions prescribed by the upper levels determines the make up of the upper levels and the capabilities of the robot system as a whole. It is therefore important to optimize these levels before proceeding with implementing any other level.

Robot manipulators belong to a class of large-scale systems which are nonlinear in nature. This results in their having a large number of special features which makes the robot control problem a difficult one. These features [2] arise from mechano-structural and dynamic considerations. Another characteristic which makes the control problem unique is the robot's variable structure: when a robot is in motion it becomes a closed kinematic chain. This means that the movement of

any one joint will affect the movement of all the remaining joints. All these and other robot characteristics [3] require that the robot be considered as a serarate system class, and should have a controller that takes its unique dynamic characteristics into account.

1.2 The Dynamic Control Problem

The main problem arising in the control task mentioned is to what extent one should take into account the real robot dynamics in control synthesis. It is up to the lower level of the control hierarchy to guarantee the desired system stabiltiy by taking the robot dynamics into account. To do this, it is nearly always necessary to form a dynamic model of the robot in question. Like most large-scale nonlinear systems, robots belong to that class of systems whose models may be set precisely enough [1]. This preciseness allows for the model to be used extensively in control design.

Approximate models are usually used to design the simplist possible control algorithms. A linearized system model is frequently used in conjunction with linear systems theory to develop a linear controller. In general, these approaches assume simplified models which are assumed to be sufficiently accurate approximations of the actual robots. However, this is not always the case, since over simplification of the model may have occured. Obviously, the closer the model is to the actual system the more likely it is that the designed controller will satisfy the system.

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 system 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 sythesised controller performance may prove

unsatisfactory. Various methods [4][5][6][7][8][9] have been used to overcome the coupling effects. These methods involve linear and nonlinear self tuning adaptive controllers. They try to overcome the coupling problem by their ability to accurately track the system nonlinearities and by compensating for their presence in the controller design.

Unfortunately, robot manufacturers are reluctant to implement such control algorithms. This reluctance comes from the fact that in most of the cases the implementation of such control algorithms would require the replacement of the existing controller hardware with a faster more expensive alternative. However, the recent developments in VLSI technologies provide cost effective solutions to the implementation of such algorithms.

1.4 Motivation for this Research

The Control Technology Research Unit (CTRU) at DCU has in recent years become interested in the area of robotics and in particular the area of robotic control. For this reason the CTRU initiated this project the main aims of which were as follows:

a) to develop a fully validated robot model,

b) to develop and implement a new hardware robot controller and,

c) to test this new robot controller.

The Control Techology Research Unit at Dublin City University has a PUMA 560 robot arm which is representative of a large and popular class of industrial manipulator. It consists of six revolute joints, see Figure 1.1. The three primary joints being known as the waist, the shoulder and the elbow. This is because of the similarity of manipulator robot structures to the human arm. The three secondary joints, which make up the wrist subassembly, 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 significent problem lies in the positioning of the tool, i.e., the control of the three primary joints. The problem arises from effects caused by the 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.

Before any new control algorithm can be implemented it is necessary to test it as fully as possible. This means a robot model is required for testing in a simulation environment. The model must must reflect, as closely as possible, the true dynamics of the robot itself. This means that the model must be validated by comparing its performance to that of the actual robot. This comparison will allow the control engineer to tune the model so as to reduce any modelling error which may have occured. If an actual algorithm is not tested and tuned on a validated model, implementation could lead to the discovery of unmodelled force or inaccurate dynamic terms. These modelling errors could cause the control algorithm to become unstable and result in serious damage to the robot.

Once a new algorithm has been tuned and tested on the model, the algorithm must be implemented. To implement the algorithm it is necessary to choose a hardware and software configuration which is capable of implementing the desired control algorithm in real time and with a high degree of numerical precision.

1.5 Thesis Contributions

This thesis developes a fully tested robot controller design suite which allows for the design, testing and implementation of robot control algorithms. The contributions of this thesis to the area of robot research are as follows:

a) The thesis developes a comprehensive model for the three primary joints of the PUMA 560 industrial robot based on the work of Anderson [14]. The model developed takes the form of a set of third order differential equations which describe the dynamic behavior of the robot. These equations are then used to simulate and test the validity the model developed.

b) A control hardware specification is developed for robotic manipulators. This specification is then implemented using a personal computer and three special purpose digital signal processor boards. An interface between the PUMA 560 and the new control hardware is designed and implemented which incorporates all the safety features of the existing interface. In addition to these features and a vast increase in computational performance, the new interface offers a greater degree of flexibliity due to the ability of adding extra sensors.

c) The thesis discusses the kinematic analysis of the three primary joints of the PUMA 560 robot with a view to path planning and setpoint generation. It focuses on the the 4x4 homogeneouss transformation matrices between ajacent manipulator links, the kinematic equations and the inverse kinematic solution for the PUMA 560 robot.

d) A general time series model is developed for the motion of the primary joints of the PUMA 560 for use in self-tuning/adaptive controllers. Various linear and nonlinear least squares-based identification techniques are implemented to determine

the parameters of this model. The identified models are then compared to assess how accurately these model represent the dynamic characteristics of the robot.

1.6 Preview of Thesis

The research in this thesis is organized as follows:

Chapter 2 outlines the modelling procedure used to model the PUMA 560. It then details the simulation of the new model and outlines how the model has been validated.

Chapter 3 is concerned with the development of a new controller hardware structure. It details the shortcomings of the existing controller and draws up the ideal controller specification and details the hardware chosen to implement the specification.

Chapter 4 deals with the hardware design and implementation necessary to interface the new controller hardware to the PUMA 560 robot.

Chapter 5 examines the computational considerations for the robot control hardware,

Chapter 6 gives a solution to the inverse kinematic problem for the primary joints of the PUMA 560 robot.

Chapter 7 discusses the parameterization of the robot model using linear and nonlinear parameter identification techniques.

Conclusions are given in Chapter 8.

FIGURE 1.1 THE PUMA 560 INDUSTRIAL MANIPULATOR

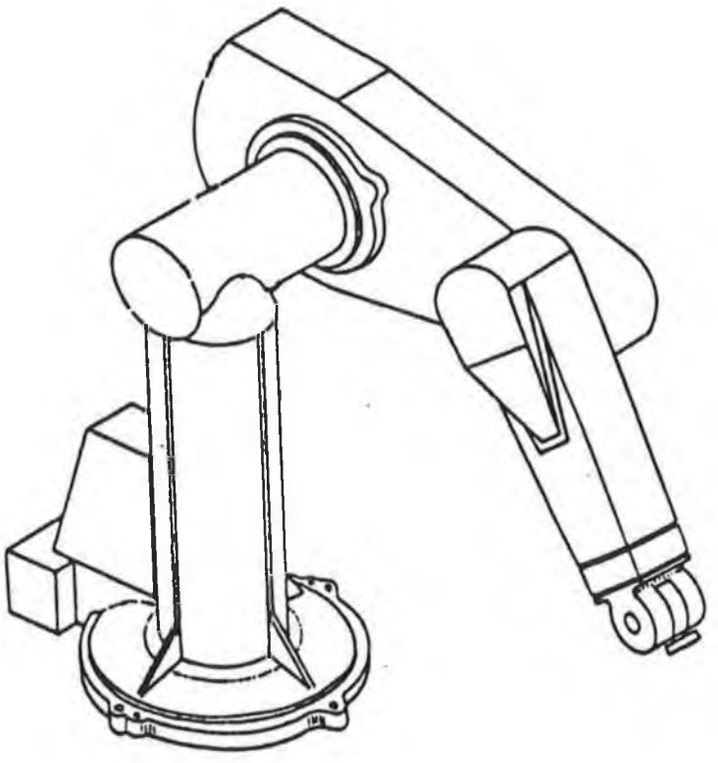
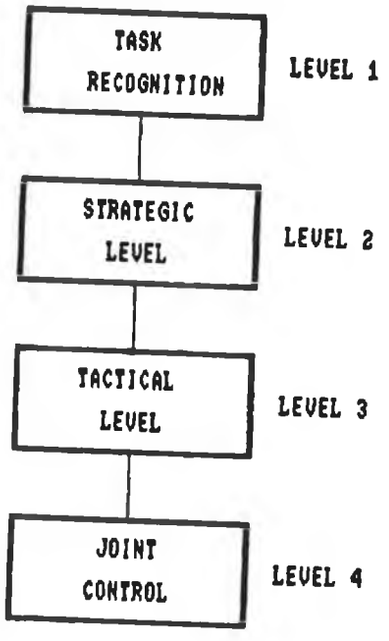


FIGURE 1.2 THE CONTROL HIERARCHY FOR MANIPULATORS



CHAPTER 2

Dynamic Modelling and Simulation of the PUMA 560

This chapter is concerned with the development and computer simulation of a dynamic model for the three primary joints of PUMA 560 industrial robot. The model outlined here is based on a second-order Euler-Lagrangian formulation of the PUMA 560 equations of motion [10]. This model is expanded to include the effect of the joint actuators, producing a third-order model for this particular robot.

In order to facilitate controller appraisal, the model is simulated on a digital computer. The simulation uses a classical 4th order Runge-Kutta technique to solve the third order differential equations present in the model. The simulator has, as it's inputs, the motor voltages necessary to drive the primary robot joints. The outputs from the model are in the form of positions, velocities and accelerations of these joints.

The model simulator is then tuned and put through a number of time response tests. The aim of these tests is to see if the simulator exhibits dynamic characteristics similar to those of the actual robot.

2.1 Dynamic Modelling of the PUMA 560

The first step in the design of a robot controller usually entails the dynamic modelling of the physical system to be controlled. For serially connected open-loop kinematic chains [10], like robots, the problem of generating a comprehensive dynamic model remains a challanging one. In the past 20 years, numerous approaches [11][12][13] for the modelling of kinematic chains have been applied to themodelling of robotic manipulators. The most commonly used of these approaches is the Euler–Lagrangian (E-L) method.

The Euler-Lagrange method dates back to the mid-sixties [12] and has been applied to numerous robots, including the PUMA 560 [13] [14]. The derivation of a dynamic model based on this method is simple to understand because it uses a systematic approach to derive the model equations. The resulting equations of motion, excluding the dynamics of the control device, gear friction, and backlash are a set of second order, coupled, nonlinear, differential equations. Each equation contains a number of torque terms classified into four groups: inertial torque due to the links, reaction torques generated by joint accelerations, velocity generated reaction torques and torques generated due to gravity effects. From a control point-of-view, it is desirable to obtain a model which is a set of closed-form differential equations. This allows all the reaction forces represented in the model to be monitored for the purpose of designing a controller. The Euler-Lagrange method is the most suitable method for this monitoring because of its systematic approach in deriving the torque equations. For this reason the Euler-Lagrange method was chosen.

The Euler-Lagrange equations of motion for the three primary joints of the PUMA 560 robot [13] can be written in the following format:

$$F_i = \sum_{j=1}^{3} D_{ij}\ddot{q}_i + N_i Ia_i \ddot{q}_i + \sum_{j=1}^{3}\sum_{k=1}^{3} C_{ijk}\dot{q}_j\dot{q}_k + G_i \qquad (2.1)$$

where,

$q_i, \dot{q}_i, \ddot{q}_i$ = position, velocity and acceleration of joint i

F_i = torque acting on joint i,

Ia_i = 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,

N_i = gearing ratio of joint i.

The inertia, centripetal, coroilis and gravity terms have been identified by Bejczy [13] and are defined as follows:

$$D_{11} = m_1 k^2_{1yy} + m_2(k^2_{2xx} S^2_2 + k^2_{2yy} C^2_2 + a_2{}^2 C^2_2 + 2a_2 x_2 C^2_2) + m_3[(k^2_{3xx} S^2_{23} + k^2_{3zz} C^2_{23} + d^2_3 + a_2 C^2_2 + a^2_3 C^2_{23}) + 2a_2 a_3 C_2 C_{23} + 2x_3(a_2 C_2 C_{23} + a_3 C^2_{23}) + 2y_3 d_3 + 2z_3(a_3 C_{23} S_{23} + a_2 C_2 S_{23})] \quad (2.2)$$

$$D_{12} = m_2 a_2 z_2 S_2 + m_3[(d_2 x_3 + a_3 y_3 + a_2 d_3) S_{23} + (a_2 y_3 + a_2 d_3) S_2 - d_3 z_3 C_{23}] \quad (2.3)$$

$$D_{13} = m_3[(x_3 d_3 + a_3 y_3 + a_3 d_3) S_{23} - z_3 d_3 C_{23}] \quad (2.4)$$

$$D_{22} = m_2(k^2_{2zz} + a^2_2 + 2a_2 z_2) + m_3[(2a_2 a_3 + 2a_2 x_3) C_3 + 2a_2 z_2 S_3 + k^2_{3yy} + a^2_2 + a^3_3 + 2a_3 x_3] \quad (2.5)$$

$$D_{23} = m_3[(a_2 x_3 + a_2 a_3) C_3 + a_3 z_3 S_3 + 2a_3 x_3 + a^2_3 + k^2_{3yy}] \quad (2.6)$$

$$D_{33} = m_3(k^2_{3yy} + a^2_3 + a_3 x_3) \quad (2.7)$$

$$C_{112} = m_2(k^2_{3xx} - k^2_{2yy} - a^2_2 - 2a_2 x_2) C_2 S_2 + m_3[k^2_{3xx} C_2 S_2 + C_3 S_3 + 2S_2 S_3 S_{23}) + k^2_{3zz}(2S_2 S_3 S_{23} - C_2 S_2 - C_3 S_3) + x_3(-2a_2 S_2 S_{23} + 4a_3 S_2 S_3 S_{23} + a_2 S_3 - 2a_3 C_2 S_2 - 2a_3 C_3 S_3) + z_3(a_2 C_2 C_{23} - a_2 S_2 S_{23} + 2a_3 C^2_{23} - a_3) + \quad (2.8)$$

$$a_2a_3S_3 - 2a_2a_3C_2S_{23} - a^2{}_2C_2S_2 +$$
$$2a^2{}_3S_2S_3S_{23} - a^2{}_3(C_2S_2 + C_3S_3)]$$

$$C_{113} = m_3[k^2{}_{3xx}(C_2S_2 + C_3S_3 - 2S_2S_3S_{23}) + \quad (2.10)$$
$$k^2{}_{3zz}(2S_2S_3S_{23} - C_2S_2 - S_3C_3) +$$
$$x_3(a_3S_2S_3S_{23} - 2a_3C_2S_3 - 2a_3C_3S_3 -$$
$$a_2C_2S_{23}) + z_3(2a_3C^2{}_{23} + a_2C_2C_{23} - a_2) +$$
$$2a^2{}_3S_2S_3S_{23} - a_2a_3C_2S_{23} - a^2{}_3C_2S_2 - a^2{}_3C_3S_3]$$

$$C_{122} = m_2a_2z_2C_2 + \quad (2.11)$$
$$m_3[d_3z_3S_{23} + (d_3x_3 + a_3y_3 + a_3d_3)C_{23}]$$

$$C_{123} = m_3[d_3z_3S_{23} + (d_3x_3 + a_3y_3 + a_3d_3)C_{23}] \quad (2.12)$$

$$C_{133} = m_3[d_3z_3S_{23} + (d_3x_3 + a_3y_3 + a_3d_3)C_{23}] \quad (2.13)$$

$$C_{213} = 0 \text{ (because of PUMA geometry)} \quad (2.14)$$

$$C_{223} = m_3[(-a_2x_3 - a_2x_3 - a_2a_3)S_3 + a_2x_3C_3] \quad (2.15)$$

$$C_{233} = m_3[(-a_2x_3 - a_2x_3 - a_2a_3)S_3 + a_2x_3C_3] \quad (2.16)$$

$$G_1 = 0 \text{ (because of the PUMA geometry)} \quad (2.17)$$

$$G_2 = -m_2g(x_2 + a_2)C_2 - \quad (2.18)$$
$$m_3g(x_3C_{23} + z_3S_{23} + a_3C_{23} + a_2C_2)$$

$$G_3 = -m_2g(x_3C_{23} + z_3S_{23} + a_3C_{23}) \quad (2.19)$$

where $S_i = \sin(\Theta_i)$, $S_{ij} = \mathrm{Sin}(\Theta_i+\Theta_j)$, $C_i = \mathrm{Cos}(\Theta_i)$ and $C_{ij} = \mathrm{Cos}(\Theta_i+\Theta_j)$. Using Newton's second law of physics the following rules apply:

$$
\begin{aligned}
D_{ij} &= D_{ji} \\
D_{ijk} &= D_{ikj} \\
D_{ijk} &= -D_{kji} \text{ for } i, k \geq j \\
D_{iji} &= 0 \text{ for } i \geq j
\end{aligned}
\tag{2.20}
$$

Consequentially this gives the following relationships:

$$
\begin{aligned}
&D_{21} = D_{12},\ D_{13}=D_{31},\ D_{32}=D_{23}, \\
&C_{111}= C_{222} =C_{333} = 0,\ C_{121}= C_{112}, \\
&C_{131}= C_{113},\ C_{132} = C_{123},\ C_{221} = C_{212}, \\
&C_{231}= C_{213},\ C_{232} = C_{223},\ C_{321} = C_{312}, \\
&C_{331}= C_{313},\ C_{332} = C_{323},\ C_{211} = -C_{112}, \\
&C_{311}= -C_{113},\ C_{312} = -C_{213},\ C_{322} = -C_{223}, \\
&C_{313}= C_{313} = C_{212}= 0
\end{aligned}
\tag{2.21}
$$

The quantities x_i, y_i, z_i are the Cartesian coordinates of the centre of mass of joint i referenced to the base of the robot. The quantity m_i is the the mass of joint i. The values of 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 of the primary joints, see Figure 7.1. 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.

Table 2.1 PUMA 560 Inertial Parameters

link	centre of mass			mass	radius of gyration (cm)		
i	x_i(cm)	y_i(cm)	z_i(cm)	g.s²/cm	k^2_{ixx}	k^2_{iyy}	k^2_{izz}
1	0	30.88	3.89	13.21	1816.3	151.93	1811.13
2	-32.89	0.0	20.38	22.8	595.7	1355.64	1513.63
3	-2.04	-1.37	0.3	5.117	151.48	155.23	20.68

Table 2.2 PUMA 560 Geometric Parameters

a_2(cm)	a_2(cm)	d_2(cm)	d_3(cm)
43.18	1.91	15.05	43.31

This representation differs from the Euler-Lagrange representation given in [14], due to the multiplication of the actuator inertia terms by the gearing ratios of the joints. The justification for this inclusion lies in the fact that the motor inertia is reflected through the joint's gear train. This causes the inertia at joint i to become N_i times greater.

If one examines the model in equation (2.1) it can be seen that the inputs to this model are the joint torques while the outputs are the positions, velocities and accelerations of joints. The inputs to the PUMA 560 are the actuator inputs needed to drive its DC motors. As a result, the model is not complete without the inclusion of the motor dynamics in the overall equations of motion of the robot. The dc motors used to drive the first three joints of the PUMA 560 are 100 Watt permanent magnet direct current servomotors. Figure 2.2 shows a simple equivelent 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 :

$$V_i = R_i i_i + L_i \frac{di_i}{dt} + k_i{}^e \frac{d\omega_i}{dt} \qquad (2.22)$$

The torque produced by a dc motor is propotional to the armature current in Figure 6.2 for the dc motor as follows:

$$F_i = k^t{}_i i_i \qquad (2.23)$$

Rearranging this term gives an equation for the armature current as:

$$i_i = \frac{F_i}{k^t{}_i} \qquad (2.24)$$

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

$$q_i = N_i \omega_i \qquad (2.25)$$

Back substituting equations (2.24) and (2.25) into equation (2.2) gives the following equation for joint voltage:

$$V_i = \frac{R_i F_i}{k_i{}^t} + L_i \frac{d}{dt}\left[\frac{F_i}{k_i{}^t}\right] + k_i N_i \frac{dq_i}{dt} \qquad (2.26)$$

This is the new model equations for the PUMA 560. From this it can be seen that the new equation contains a term which is the derivative of the joint torque F. Since the torque term F in equation (2.1) is a second order equation (i.e. contains joint acceleration terms), by including the derivative of the joint torque the model becomes a third order model. The model can be rearranged as follows:

$$V_i = \frac{R_i F_i + L_i \dot{F}_i}{k_i{}^t} + k_i{}^e N_i \dot{q}_i \qquad (2.27)$$

The quantity $\dot{F}_i$ is given by:

$$\dot{F}_i = \sum_{j=1}^{3} (\dot{D}_{ij}\ddot{q}_j + D_{ij}\dddot{q}_j) + N_i Ia_i \dddot{q}_i \qquad (2.28)$$

$$+ \sum_{j=1}^{3}\sum_{k=1}^{3} (\dot{C}_{ijk}\dot{q}_j\dot{q}_k + C_{ijk}\ddot{q}_j\dot{q}_k + C_{ijk}\dot{q}_j\ddot{q}_k) + \dot{G}_i$$

The full model can then be written by substituting equations (2.1) and (2.28) into equation (2.27). This substitution gives::

$$V_i = k_i e N_i \dot{q}_i \qquad (2.29)$$

$$+ \frac{R_i}{k_i} t \left[\sum_{j=1}^{3} D_{ij}\ddot{q}_i + N_i Ia_i \ddot{q}_i + \sum_{j=1}^{3}\sum_{k=1}^{3} C_{ijk}\dot{q}_j\dot{q}_k + G_i \right]$$

$$+ \frac{L_i}{k_i} t \left[\sum_{j=1}^{3} (\dot{D}_{ij}\ddot{q}_j + D_{ij}\dddot{q}_j) + N_i Ia_i q_i \right.$$

$$\left. + \sum_{j=1}^{3}\sum_{k=1}^{3} (\dot{C}_{ijk}\dot{q}_j\dot{q}_k + C_{ijk}\ddot{q}_j\dot{q}_k + C_{ijk}\dot{q}_j\ddot{q}_k) + \dot{G}_i \right]$$

This is the third order model equation for joint i of the PUMA 560 robot.

2.2 Computer Simulation of the PUMA 560 Robot Arm

The design and computer implementation of the manipulator simulator are described in this section. The simulator is designed, from a control engineering point-of-view, to model the arm's dynamics. The inputs to the simulator are the joint constraints, the initial joint positions, velocities, accelerations and the voltages to drive the motors as a function of time. The outputs of the simulator are the positions, velocities and accelerations of the joints as a function of time.

A state-space representation similar to those used in [13] and [14] was used to program the model. This meant that the model given in equation (2.29) had to be rearranged as to isolate the highest order terms (3rd order terms). To achieve this isolation of terms equation (2.27) the derivative of the joint torque term in

this equation was rearranged as follows:

$$\dot{F}_i = f_i + A_i \tag{2.30}$$

where,
$$A_i = \sum_{j=1}^{3} D_{ij}\ddot{q}_j + Ia_i\dot{\ddot{q}}_i \tag{2.31}$$

and f_i contains the remaining terms of F_i in equation (2.27). Back substituting equation (2.30) into the model of equation (2.27) gives a model of the form:

$$V_i = \frac{R_iF_i + L_i(f_i + A_i)}{k_i^t} + k_i^e N_i \dot{q}_i \tag{2.32}$$

Rearranging this equation to isolate the A_i term:

$$\frac{k_i e}{L_i}\left[V_i - \frac{(R_iF_i + L_if_i)}{k_i^t} - k_i^e N_i q_i \right] = A_i \tag{2.33}$$

So far the model that has been developed has been for one joint only. To represent the three joints in the model it is convenient to use some form of vector and matrix notation. By changing to vector and matix notations the model can be written as:

$$L^{-1}K_e[V - (K_e)^{-1}(RF + L.f) - K_e.N.q] = A \tag{2.34}$$

where the diagonal matrices appearing in equation (2.34) are as follows:

Joint gearing ratio matrix $N = \text{Diagonal}[N_1, N_2, N_3]$,

Moter back emf matrix $K_e = \text{Diagonal}[k_1^e, k_2^e, k_3^e]$,

Torque constant matrix $K_t = \text{Diagonal}[k_1^t, k_2^t, k_3^t]$,

Armature resistance matrix $R = \text{Diagonal}[R_1, R_2, R_3]$,

Armature inductance matrix $L = \text{Diagonal}[L_1, L_2, L_3]$,

and the vectors appearing in equation (2.34) are as follows

Input voltage vector $V = [V_1, V_2, V_3]$,

Joint torque vector $F = [F_1, F_2, F_3]$,

joint velocity vector $\dot{q} = [\dot{q}_1, \dot{q}_2, \dot{q}_3]$,.

The vector A can be rewritten as follows:

$$A = (D + NI)\dddot{q} \tag{2.35}$$

where D is the matrix of self and coupling inertias of the robot joints (ie. element D[i,j] of the D matrix contains the inertial term D_{ij} in equation (2.1), and the matrix I is a diagonal matrix with the reflected rotor inertias of each joint making up its diagonal elements. The vector q contains the third derivatives of the joint position necessary for the state-space representation. By back substituting equation (2.35) into (2.33) and multiplying both sides of the new equation by the matrix inverse quantity $(D + NI)^{-1}$, the third derivatives can be isolated.

The state-space model can then be written [13] using the notation:

$$\begin{aligned} X_1 &= q_1 \quad X_2 = q_2 \quad X_3 = q_3 \\ X_4 &= \dot{q}_1 \quad X_5 = \dot{q}_2 \quad X_6 = \dot{q}_3 \\ X_7 &= \dot{q}_1 \quad X_8 = \dot{q}_2 \quad X_9 = \dot{q}_3 \end{aligned} \tag{2.36}$$

to represent the model the positions, velocities and accelarations of all the joints. These states can then be written in the format:

$$\begin{aligned} \dot{X}_1 &= X_4 \\ \dot{X}_2 &= X_5 \\ \dot{X}_3 &= X_6 \\ \dot{X}_4 &= X_7 \\ \dot{X}_5 &= X_8 \\ \dot{X}_6 &= X_9 \end{aligned} \tag{2.37}$$

and

$$\begin{bmatrix} \dot{X}_7 \\ \dot{X}_8 \\ \dot{X}_9 \end{bmatrix} = [\, D + NI \,]^{-1} . A$$

To obtain the joint positions, velocities and accelerations it was necessary to apply some from of numerical intergration technique to solve these differential equations. Since simulator accuracy, rather than speed, was the goal it was decided to use a classical fourth-order Runge-Kutta algorithm to integrate the states in equation (2.37). Bejczy [13] points out that this method is probably the most widely used in engineering applications and lists the reasons for its popularity as follows:

1) it is self starting,
2) the integration step size can be changed easily, and
3) it displays good stability characteristics

For the state space description in equation (2.37) the Runge-Kutta integration across the k^{th} interval of state X_i is defined as:

$$X_i(t_{k+1}) = X_i(t_k) + [\, K_{1i} + 2K_{2i} + 2K_{2i} + K_{4i} \,] \qquad (2.38)$$

where h_k is the integration interval for the kth sampled interval. The Runge-Kutta coefficients are defined as follows:

$$K_{1i} = h_k f(X(t_k) \qquad (2.39)$$

$$K_{2i} = h_k f(X(t_k) + K_{1i}/2)$$

$$K_{3i} = h_k f(X(t_k) + K_{2i}/2)$$

$$K_{4i} = h_k f(X(t_k) + K_{3i}/2)$$

where $f(.)$ denotes the quantities on the right hand side of the state equations in equation (2.37).

The state space model of equation (2.38) was implemented using the Runge-Kutta algorithm just described to perform the numerical integtation. The flow chart in Figure 2.3 shows the operating sequence of the simulator program used. From this flow chart it can be seen that the simulator inputs are the joint voltages, while the outputs are the joint positions velocities and accelerations. It can also be seen that the joint positions, velocities and acceleration are checked at every sampling interval to see if they are within the operating limits of the PUMA 560 robot. In order to increase the speed of the simulator program the constant coefficients in the inertial, gravitational, centripetal and coriolis terms were calculated only once at the beginning of the simulation program.

To choose the integration interval for the Runge Kutta integration, the simulator was tested over a wide range of integration intervals. The tests involved calculating the voltage necessary for holding the joints of the model stationary. Ideally, when these voltages are applied to the simulator the joints would show no movement. So, by measuring any movement which might occur in the simulator joints it is possible to get a measure of how accurate the robot simulator was at a particular value of integration interval. Integration intervals of 1 to 5 millisecond were found to produce joint position movements of approximately 10^{-12} radians after a period of 20 seconds. Integration intervals of 10 milliseconds and higher were found to produce position errors of approximately 10^{-3} for the same time period. Integration intervals less than 1 milliseconds showed no improvement simulator accuracy over those measured for the range 1 to 5 milliseconds. For these reasons an integration interval range of 1 to 5 millivolts was chosen.

2.3 Evaluation of the Model's Performance

The model validation undertaken for the PUMA 560 model can be divided into two main parts:

1) tuning of the model terms, and
2) an evaluation of the tuned model's dynamic performance.

The following sections detail these parts of the validation.

2.3.1 Tuning of the Robot Model

To validate any model it is necessay to validate the terms which make up the model equations. Unfortunately, it is not always possible to validate all terms in the model equations. This is due to the fact that to validate a model term it must be possible to isolate that term in both the robot model and in the physical system itself and compare the two terms. In the case of the PUMA 560 it is possible to isolate very few of the force terms which make up the robot's dynamic equations. This is due to the complex nature of the dynamic and the absence of force sensors on the PUMA 560. The best one can hope for is to be able to validate the physically measurable terms of the model. The remaining model terms must then be tuned by using any relevant information about the robot's geometric and inertial parameters which can be obtained by expermimental means. Anderson [14] outlines experimental procedues for evaluation of the PUMA 560's masses, radii of gyration and reflecter motor inertia. The results obtained by [14] are shown in Table 2.3. From these it can be seen that all the experimental parameters obtained by Anderson take into account the presence of the robot's hand by modelling it as an extension of the third link. This is necessary if the model of the three primary joints is going to reflect the physical makeup of the robot. These parameters were used in the robot simulator instead of those identified by [13] in Table 2.1.

Table 2.3 Model Link Parameters

link	mi(kg)	$k^2_{ixx}(m^2)$	$k^2_{iyy}(m^2)$	$k^2_{izz}(m2)$
1	12.69	0.1802	0.1800	0.0141
2	22.37	0.0526	0.0691	0.0031
3	5.01	0.051	0.0691	0.0150

In equation (2.29), gravity can be seen as one of the terms that effect the joint positions, velocities and accelerations. The tuning of the gravity forces present in the PUMA 560 can be achieved without the use of external sensors. From the the model equation (2.29), it can be seen that if the robot joints are all stationary the model for joint i reduces to:

$$V_i = R_i F_i / k^t_i \qquad (2.40)$$

This reduction in model terms is due to the fact that all the remaining model terms which are functions of velocity and acceleration are zero when the joints are stationary. This model can be reduced even further by removing the velocity and acceleration terms of of the joint torque term, F_i. The model in equation (2.40) then becomes:

$$V_i = R_i G_i / kt_i \qquad (2.41)$$

where G_i is the gravity torque of joint i. This means that the the gravity effects on each joint can be calculated by reading the joint voltages at a stationary position. Gravity does not effect joint 1 of the PUMA 560 because its movement is an a horizonal plane. Therefore, to tune the gravity terms of the robot simulator, it was decided to record the input voltages of joint 2 and joint 3 of the PUMA 560 with all the PUMA 560 joints stationary. These voltages were recorded at 5° (0.087 rad) intervals and compared at each interval to the values derived by using the model simulator. At each interval the gravity effect on the robot was obtained using equation (2.41) and the phase difference between the model gravity terms

and the robot's gravity was recorded. The results seemed to show that the amplitude of the robot's gravity terms was on average about 15% lower than that of the model for joints 2 and 3 and that there was an average phase difference of +/-5.5° in the case of joint 2 and +/-3.5° in joint 3. Because of the phase difference variations, it was unfeasible to tune the phase difference. The amplitude difference was tuned by subtracting an offset from the gravity equation used in the model. This offset was chosen to be 15% of the calculated gravity terms. Figure 2.4 and Figure 2.5 show an example of the model's gravity parameters after adjustment and the measured joint gravity terms for joint 2 and joint 3 obtained from the robot. From these it can be seen that the amplitude in both cases shows a small error while the phase difference seems to to vary with joint position.

2.3.2 Evaluation of the Model's Performance

To evaluate the performance the robot model it was decided to carry out a number of model tests using the simulator. These tests were carefully chosen to see if certain characteristics known to be present in the PUMA 560 actually appeared in the simulated model. The following is a description of the performance results.

Test1: this involved the examination of the effects of coupling on joint 2 and joint 3 of the model simulator due to movement of joint 1. All three joint were given an initial position of 0 radians in their respective joint ranges. A step voltage of 10 volts was then applied to the simulator input of joint 1, while joint 2 and 3 were given the voltage inputs they required to stay at angles of 0 radians.
The position and velocity curves for this test are shown in Figure 2.6 and Figure 2.7. From these it can be seen that the step response of joint 1 is characteristic of step response of a motor with a constant inertial load. This type of response is to be expected since with joint 2 and 3 stationary the inertial load on actual robot's joint would remain constant. The movement of joint 2 and 3 in this test was

found to be of the order of magnitude of 0.05° proving that the model coupling effects on joint 2 and joint 3 due to the movement of joint 1 is quite small.

Test 2: the function of this test was to see if the model reflects the coupling effect joint 2 has on joint 1 in the actual robot. For this test, all the joints of the simulator were positioned at their 0 radian joint angles. The input voltage used to drive joint 1 was 10 volts, while joint 2 was given a step input of -5 volts and joint three was given a voltage to keep it stationary. The position and velocity results from this test, see Figures 2.8 and 2.9, seem to show the coupling effect of joint 2 on joint 1 is larger than when joint 2 is not moving. This can be seen by comparing the velocity profile obtained for joint 1 in this test with the velocity profile of joint 1 obtained when joint 2 was stationary, see Figure 2.7. This can be explained in terms of the actual robot by considering the changes that occur in the coupling centripetal and inertia torques of joint 1 due to changes in the velocity of joint 2.

Test 3: the purpose of this test is to examine the coupling effects between joint 1 and joint 3 of the model simulator. For this test joint 1 and joint 3 were given inputs of 10 and 5 volts respectively while joint 2 was given a voltage to hold it in a constant position. The resultant positions and velocities of the three simulator joints are shown in Figures 2.10 and 2.11. The results show that joint 1 reached its joint limit first and its velocity was zeroed. This was followed by joint 3 reaching its limit and the its velocity was reset. The effect of joint 3 suddenly stopping caused joint 2 to move to its negative position limit. This is consistent with the jarring of joints which occurs in the actual robot when joint 2 reaches its limit stop at a high velocity. This seem to reflect the strong coupling between joints 3 and 2 which is present in the robot.

Test 4: This test consists of moving all three joints of the simulator simultaneously to examine the coupling effects between all three joints. The input voltages for the joint 1, 2 and three were 10, -5 and 5 volts respectively. The simulator position and velocity results for this test are shown in Figure 2.12 and Figure 2.13. From these results it can be seen joint 1 reaches its positive position limit, followed by joint 3. Finally, joint 2 reaches its position limit after approximately 13 seconds. It is interesting to note that in Test 2, when joint 3 was held stationary, joint 2 reached its limit much faster. This decrease in joint speed is again consistent with the increased coupling effects that are present between joints 2 and 3 of the actual robot with both joints moving.

Test 5: the function of this test is to examine the model performance to see the effects of coupling due to joint 2. For this test joints 1 and 2 of the simulator was given an input voltage to hold them at a constant angle, while joint 2 was given a -5 volt input. The resultant simulator position and velocity curve for the three joints are shown in Figure 2.14 and Figure 2.15 respectively. From these it can be seen that the coupling effect on joint 1 due to joint 2's movement is smaller than the coupling effect on joint 3 due to joint 2. These result seem to reflect the coupling effects known to be present because of joint 2's movement in the actual robot.

Test 6: the purpose of this test is to examine the coupling effects of joint 3 on joint 1 of the model simulator. For this test joints 2 and 3 are given voltage inputs of -5 volts and +5 volts respectively. The simulator position and velocity results for this test are shown in Figure 2.16 and Figure 2.17. From these it can be seen that when joints 2 and 3 are moving the effect on the position of joint 1 is minimal. However, when both joints reach their position limits simultaneously, joint 1 can be seen to move from its stationary position. Such a small movement is consistent with a limit crash experienced for the actual robot with both joints reaching their limits instantaneously.

Test 7: this purpose of this test is to examine the coupling effect joint 3 of the model has on the movement of joints 1 and 2. The simulator input voltage used for joint 3 was 5 volts, while the input to the other two joints was the voltage required to keep these joints stationary. The positions and velocities for these tests can be seen in Figures 2.18 and 2.19. From these it can be seen that when joint 3 is driven to a limit stop, it has the effect of causing joint 2 to move off to its negative limit, while joint 1 moved to a position of constant displacement. This type of response when compared with the joint response of Test 2 can be seen to show that the coupling effects of joint 3 on joint 2 are considerably less than the the coupling effects of joint 2 on joint 3. This is consistent with the coupling effects known to exist between joint 2 and joint 3 on the actual robot.

The results, overall, seemed to show that the movement and coupling exhibited by the model can be explained in terms of the actual robot. The coupling experienced in the model was found to vary between the different joints. Joint 1 was found to have the smallest effects on the other 2 joints. This can be explained by the fact that joint 1 of the PUMA 560 moves in a horizontal plane while the other 2 joints movement is in a vertical plane. The strong coupling experienced by joint 2 due to joint 3 can be explained by the fact that in the actual robot the movement of joint 3 causes considerable increases in the inertial and gravitational effect changes in joint 2. The effect of joint 2 on joint 3 movement can be explained in terms of the real robot by the fact that the main coupling effect on joint 3 of the robot is the effect caused by the change in its gravity torque terms.

2.4 Summary

This chapter has been concerned with the development, simulation, tuning and performance evaluation of a robot model for the three primary joints of the PUMA 560. The model developed was a third order one based on the second order dynamic equations of motion for the PUMA 560. The chapter then detailed how the model was simulated based on a state–space representation. The simulation itself involves the use of a Runge-Kutta integration method to solve the differential equations present. Once the model was simulated, the model parameters were tuned to include the effect of the mass and inertia caused by the robot's hand. Finally, the robot model was put through a number of tests to see if it reflected the coupling effects known to be present in the actual robot. In all the tests carried out, the simulated model was found to exhibit performance characteristics which could be explained in terms of the real robot's performance.

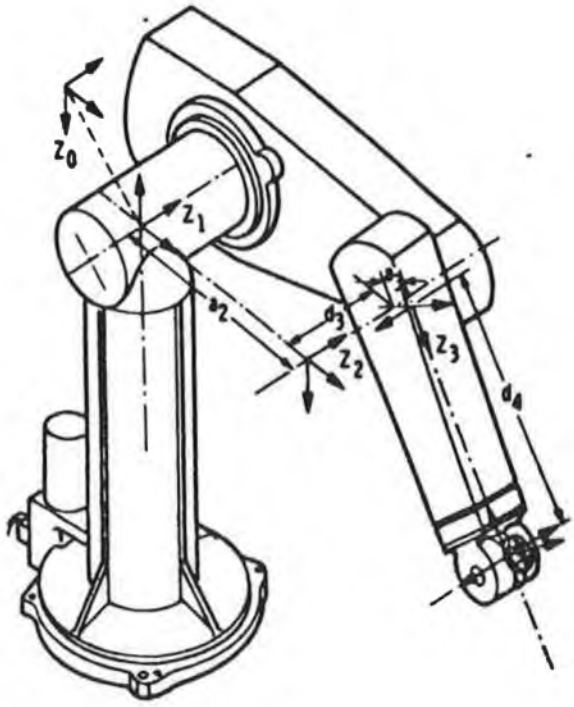
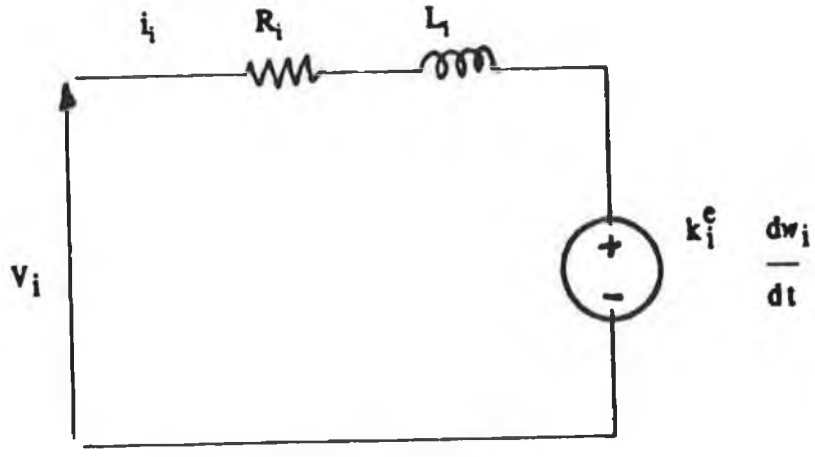


FIGURE 2.1 GEOMETRIC PARAMETERS OF THE PUMA 560

FIGURE 2.2 A TYPICAL DC SERVO MOTOR



w_i = motor position

R_i = armature resistance

L_i = armature inductance

i_i = armature current

k_i^e = voltage constant

k_i^t = torque constant

v_i = armature voltage

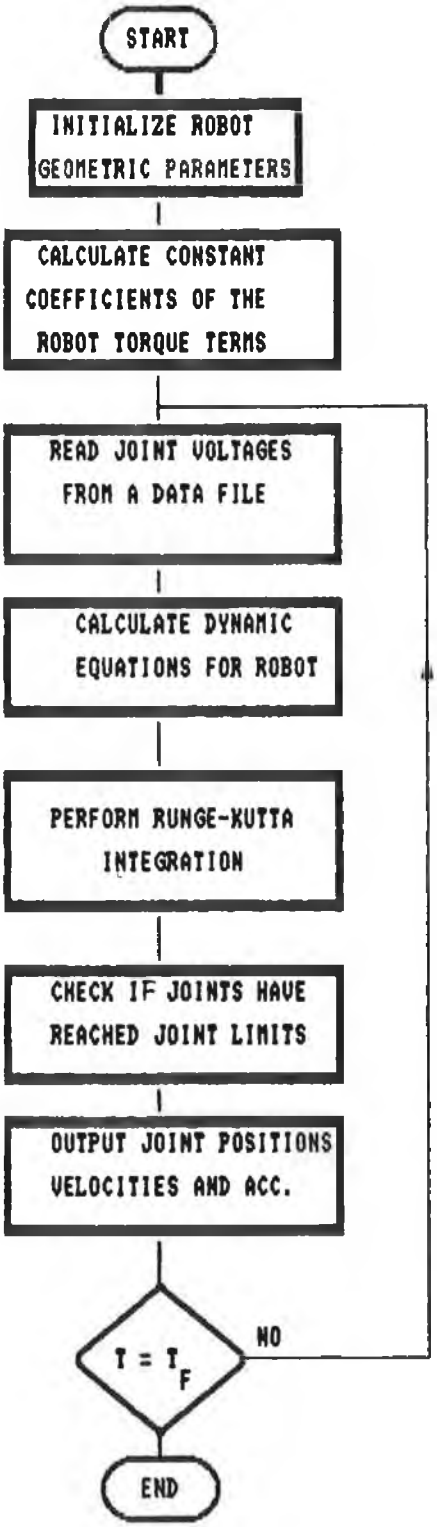


FIGURE 2.3 FLOW CHART OF THE ROBOT SIMULATOR

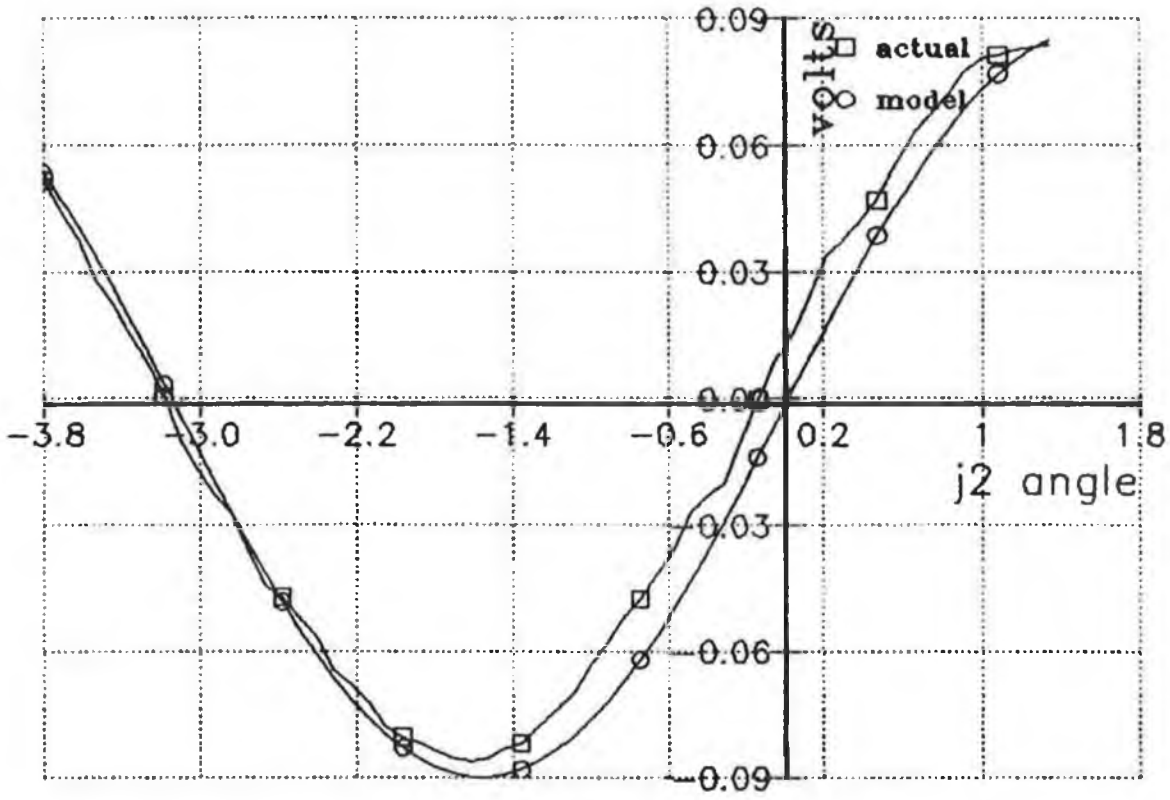
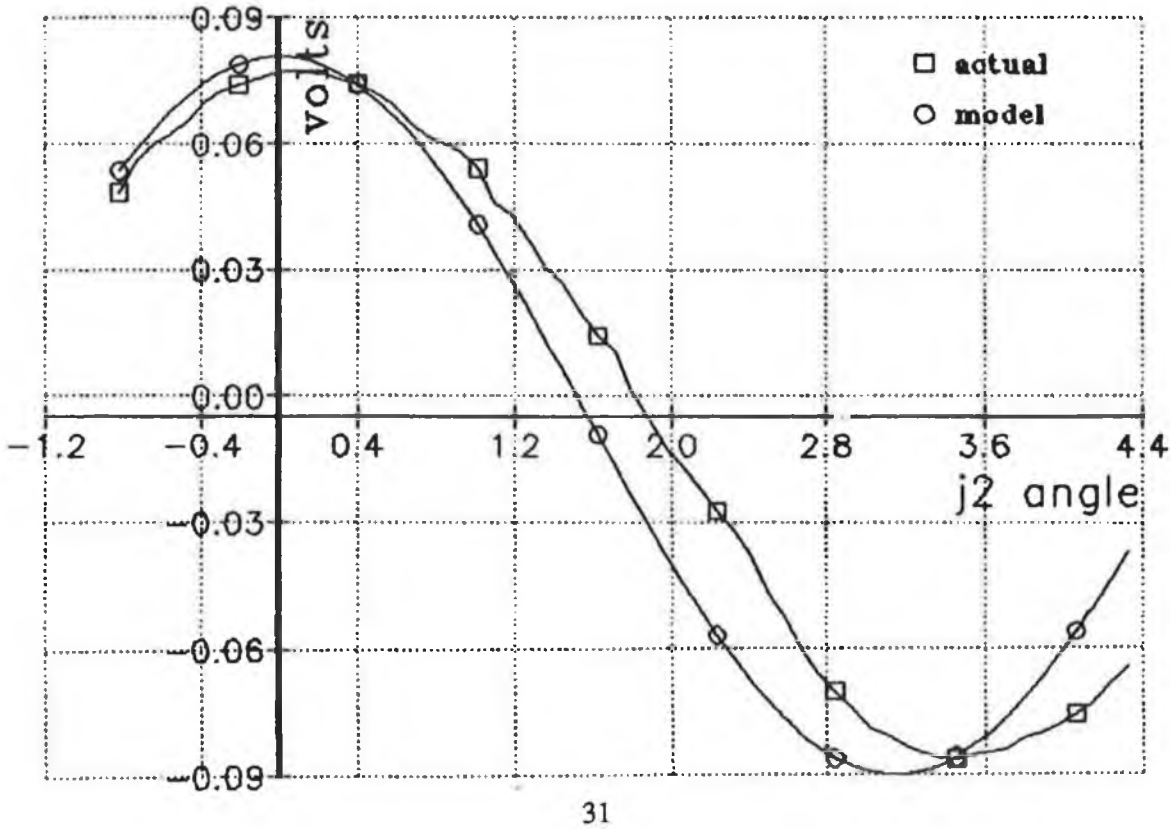


FIGURE 2.4 GRAVITY TUNING FOR JOINT 2 (VOLTS) V ANGLE (RAD)

FIGURE 2.5 GRAVITY TUNING FOR JOINT 3 (VOLTS) V ANGLE (RAD)



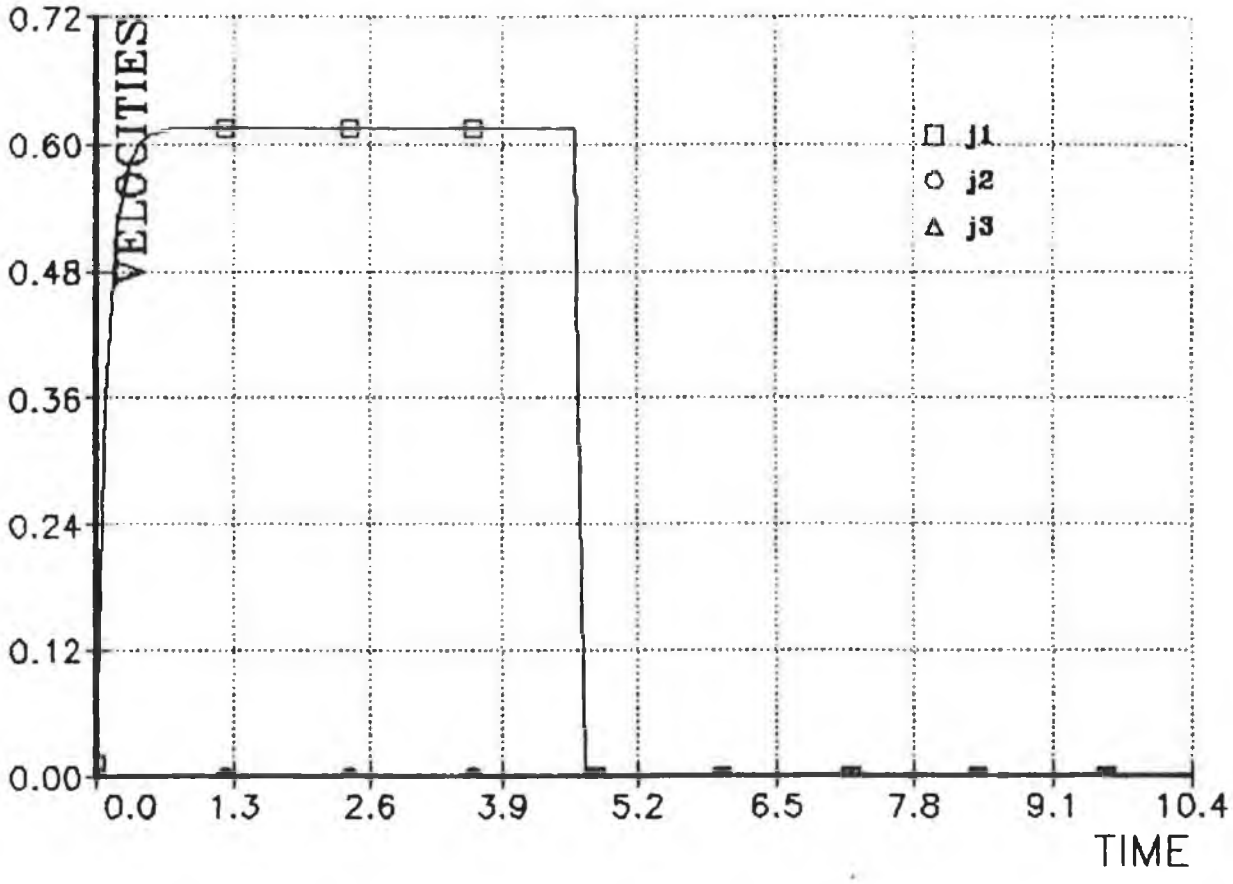
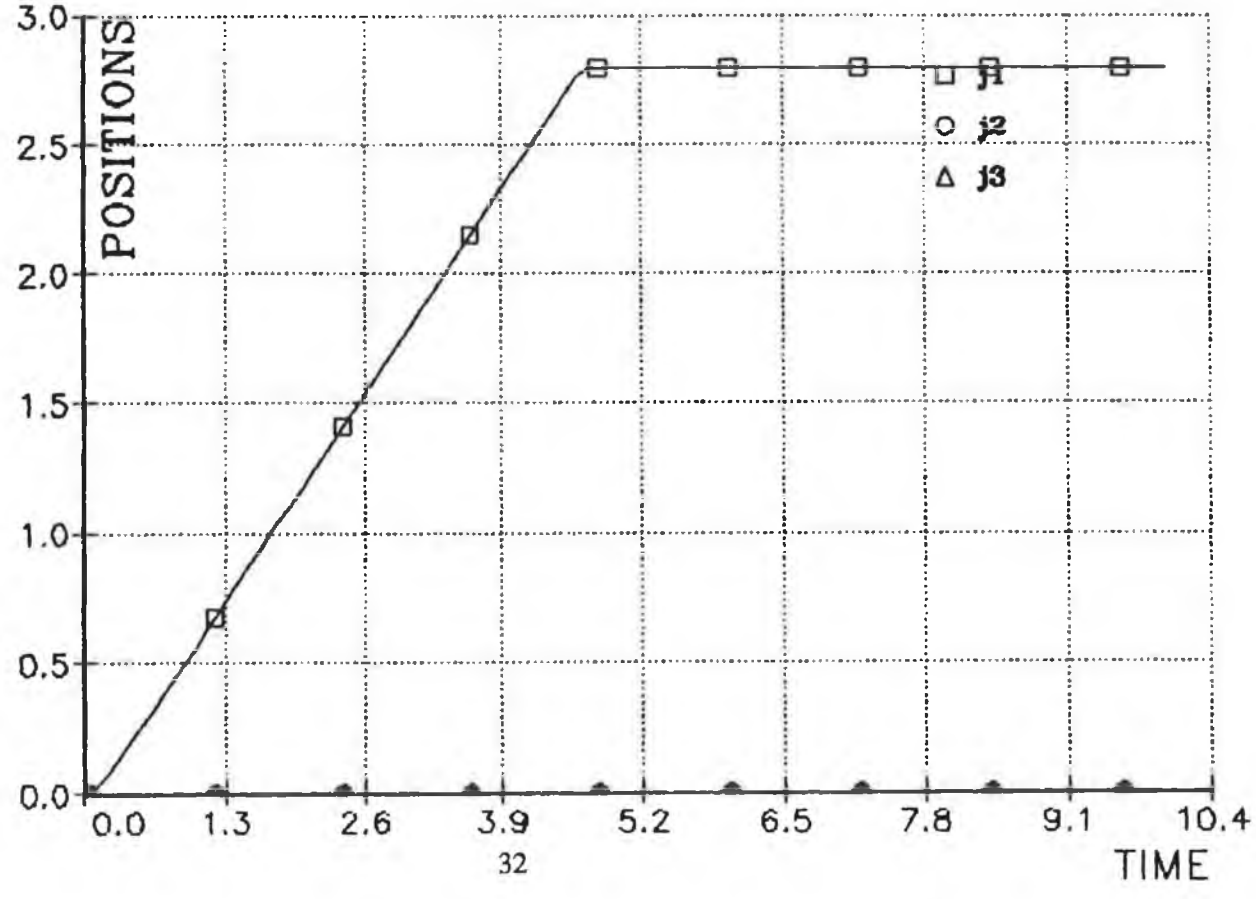


FIGURE 2.6 TEST 1 JOINT VELOCITIES (RAD/SEC) v TIME (SEC)

FIGURE 2.7 TEST 1 JOINT POSITIONS (RAD) v TIME (SEC)



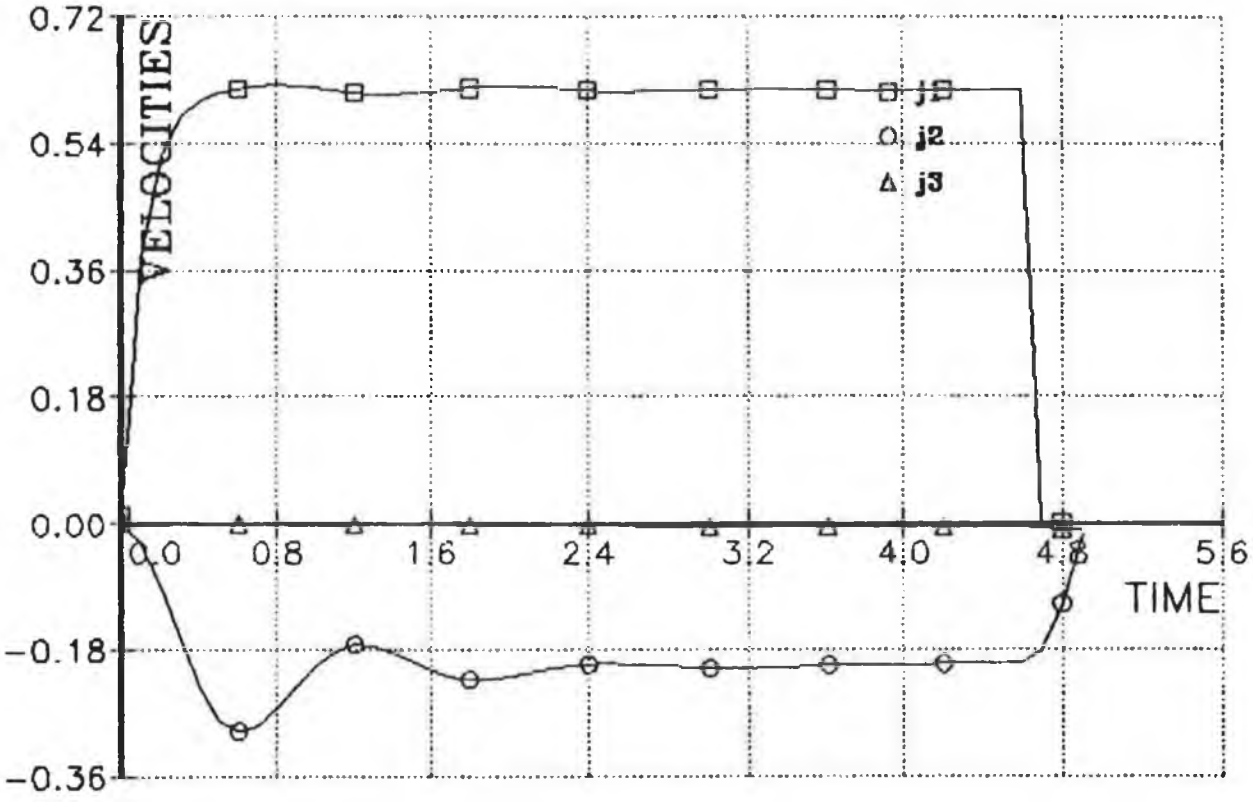
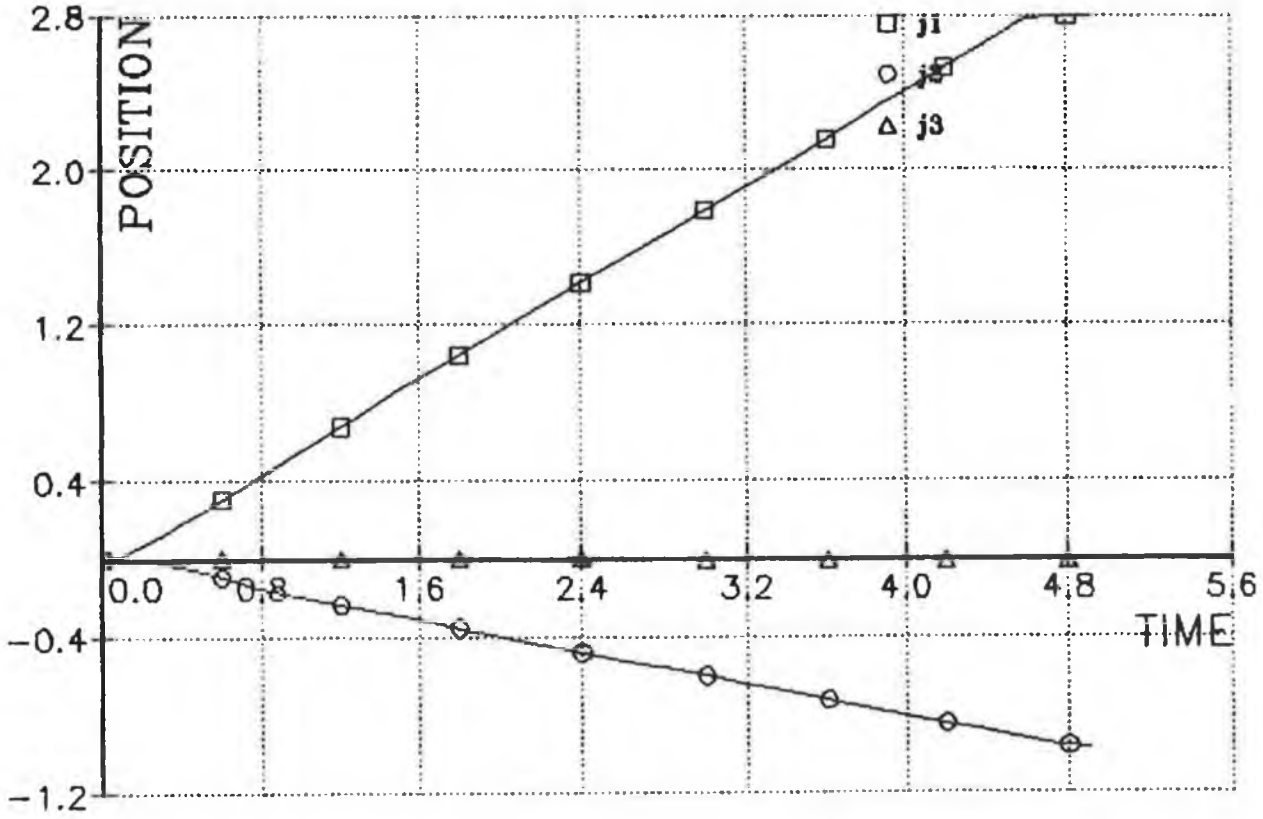


FIGURE 2.8 TEST 2 JOINT VELOCITIES (RAD/SEC) v TIME (SEC)

FIGURE 2.9 TEST 2 JOINT POSITIONS (RAD) v TIME (SEC)



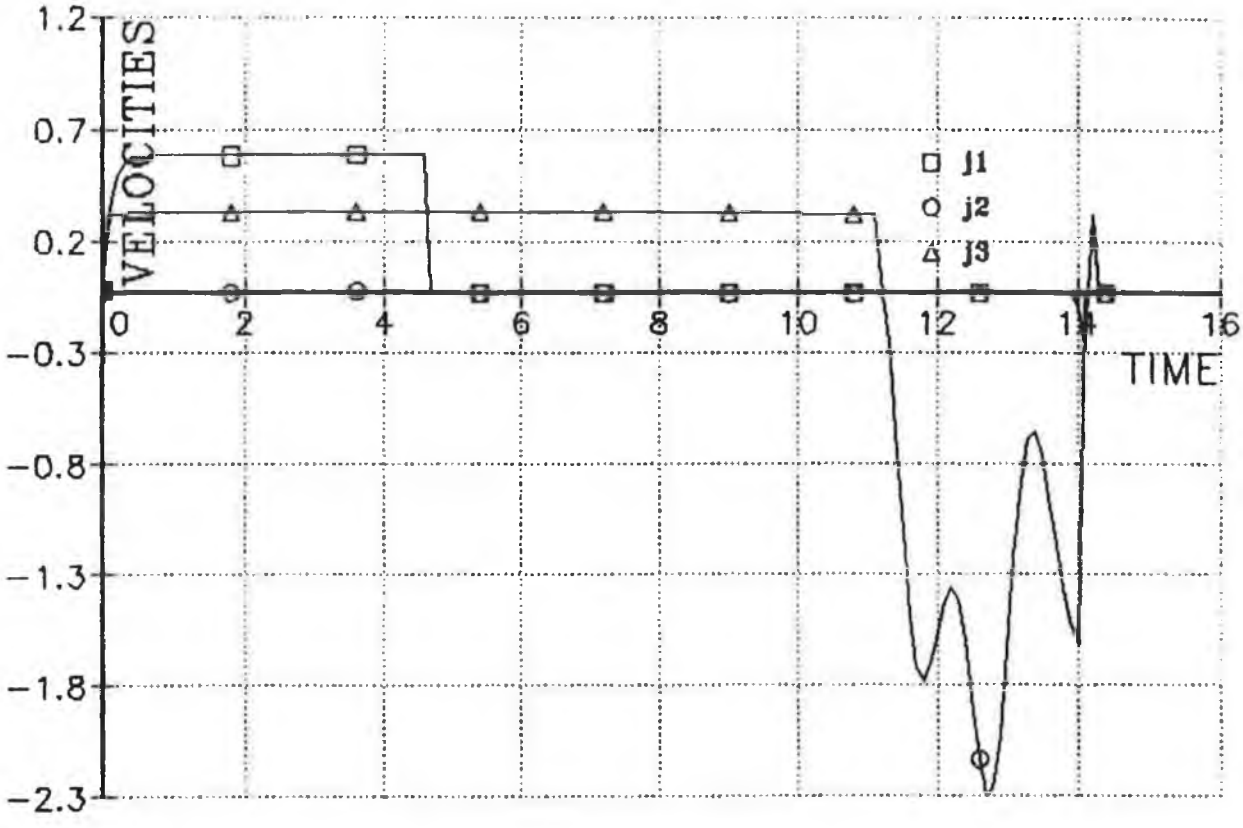
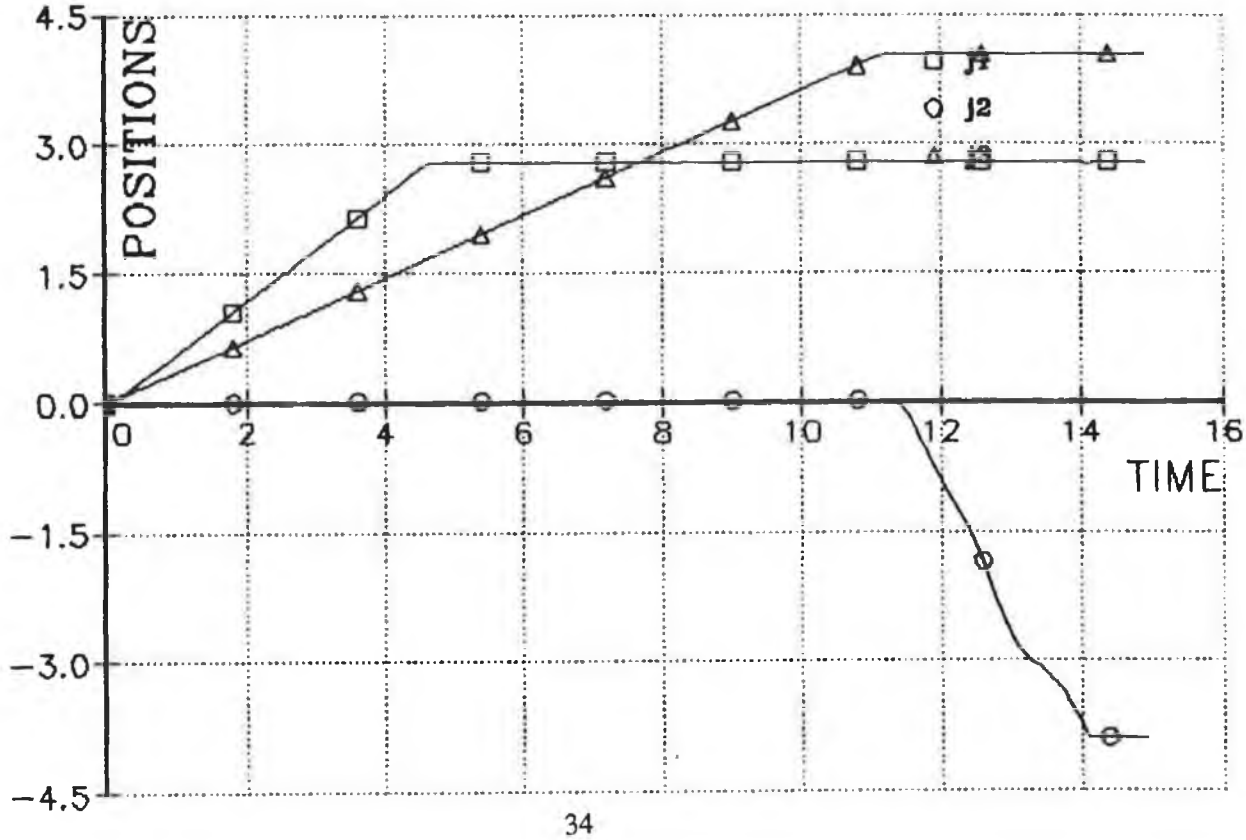


FIGURE 2.10 TEST 3 JOINT VELOCITIES (RAD/SEC) v TIME (SEC)

FIGURE 2.11 TEST 3 JOINT POSITIONS (RAD) v TIME (SEC)



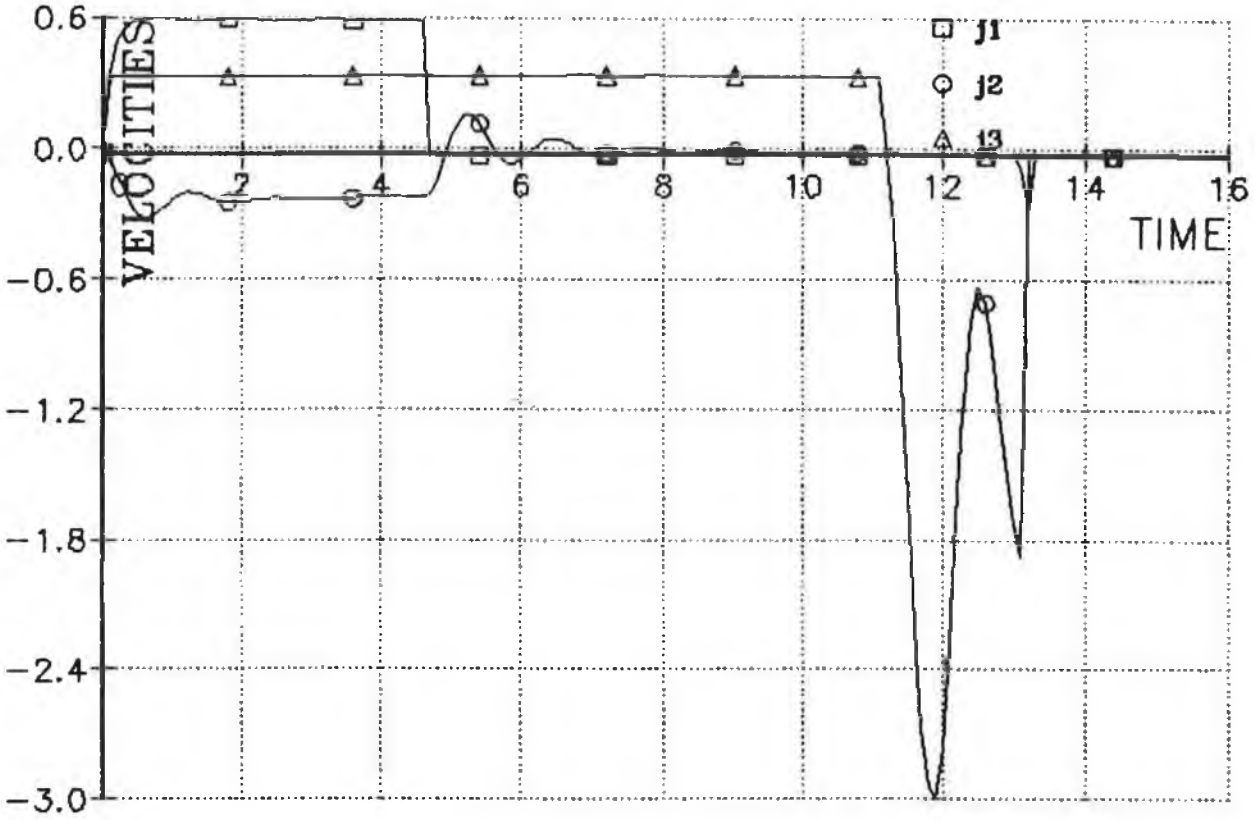
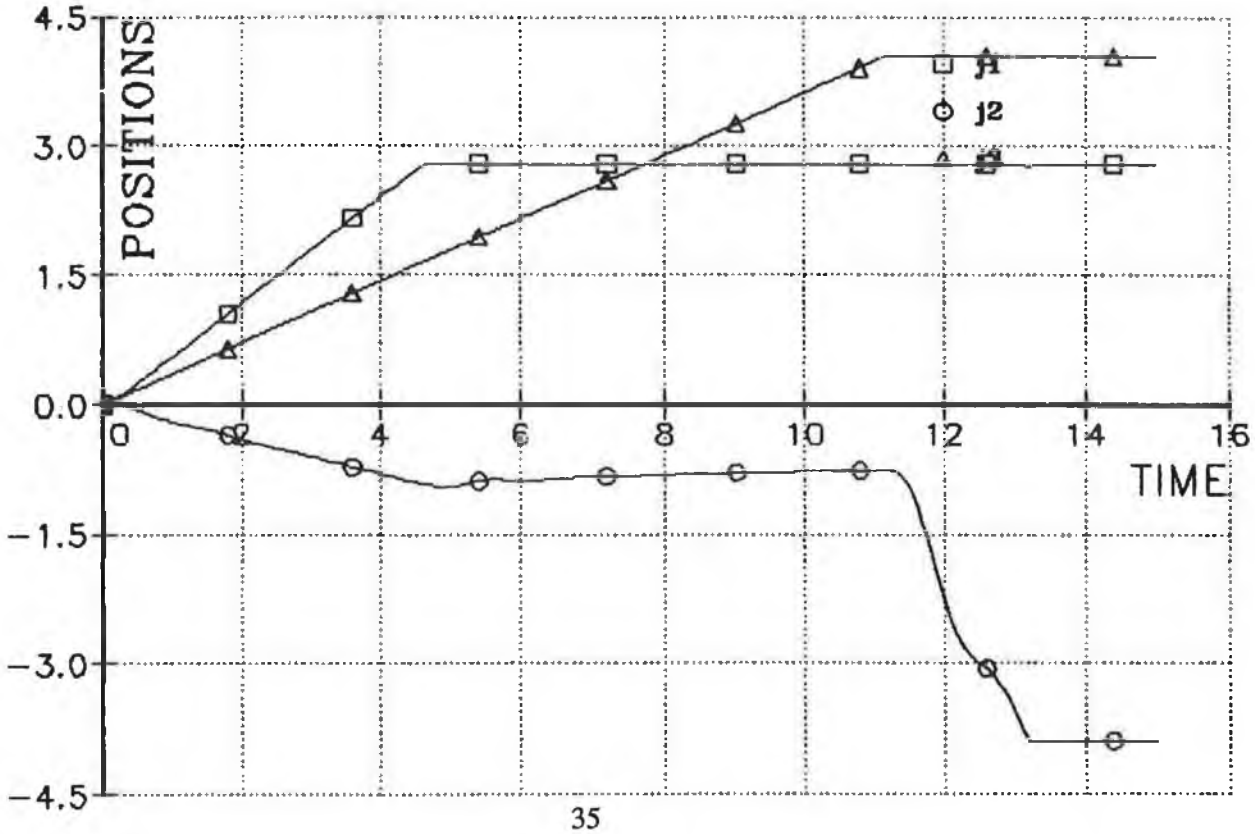


FIGURE 2.12 TEST 4 JOINT VELOCITIES (RAD/SEC) v TIME (SEC)

FIGURE 2.13 TEST 4 JOINT POSITIONS (RAD) v TIME (SEC)



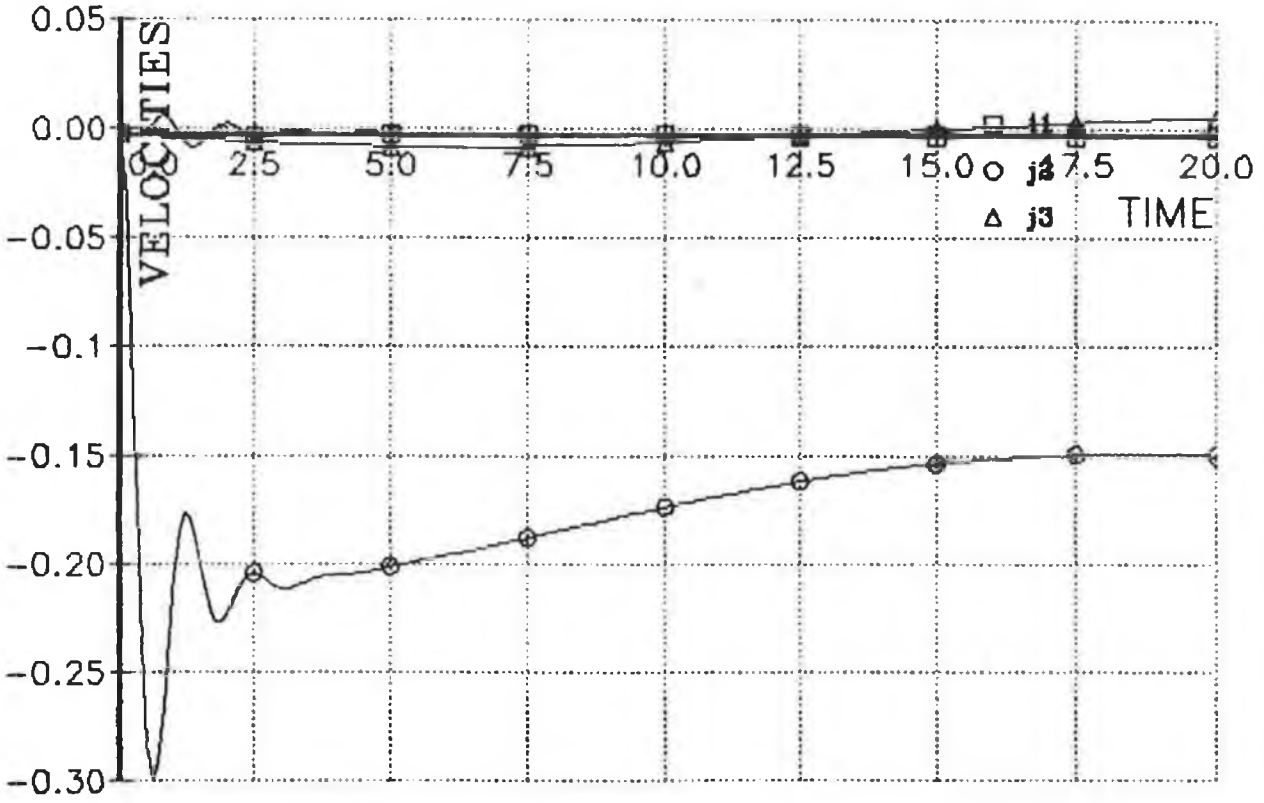
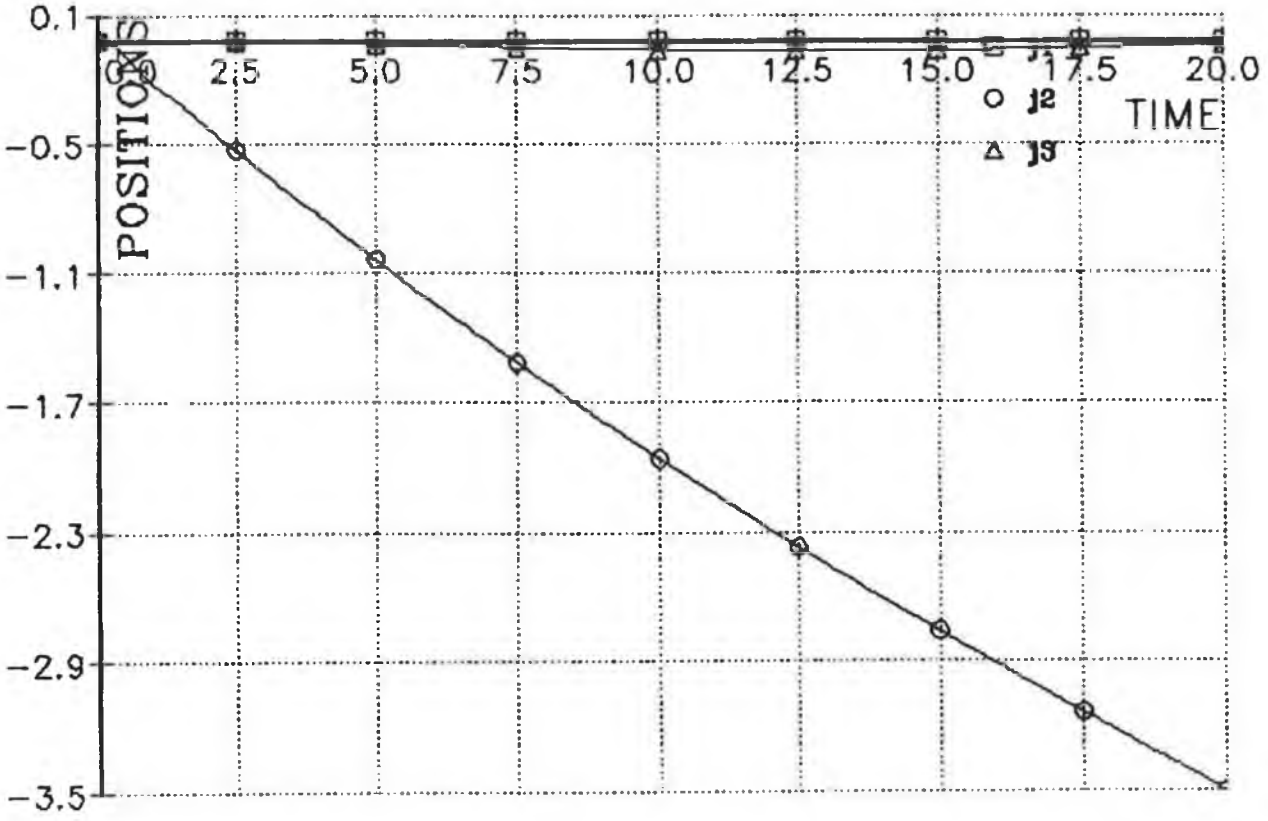


FIGURE 2.14 TEST 5 JOINT VELOCITIES (RAD/SEC) v TIME (SEC)

FIGURE 2.15 TEST 5 JOINT POSITIONS (RAD) v TIME (SEC)



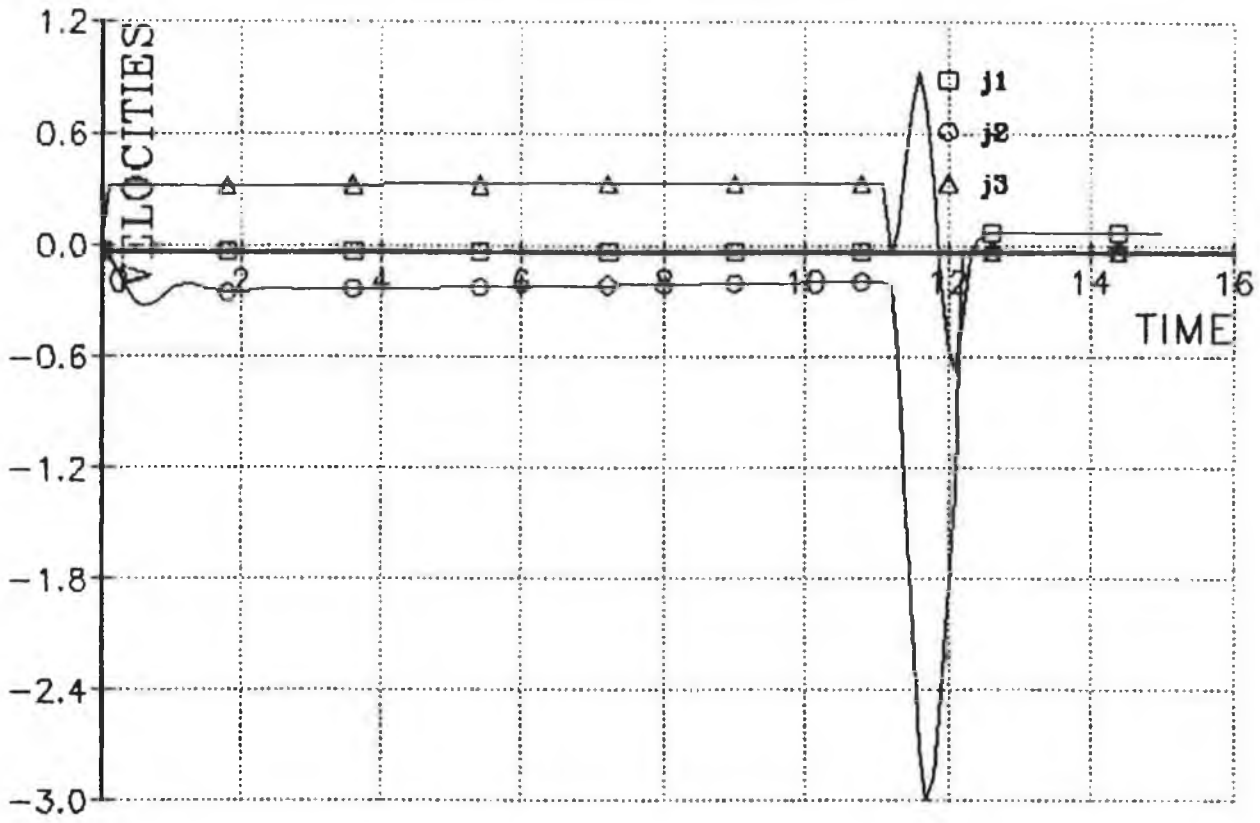
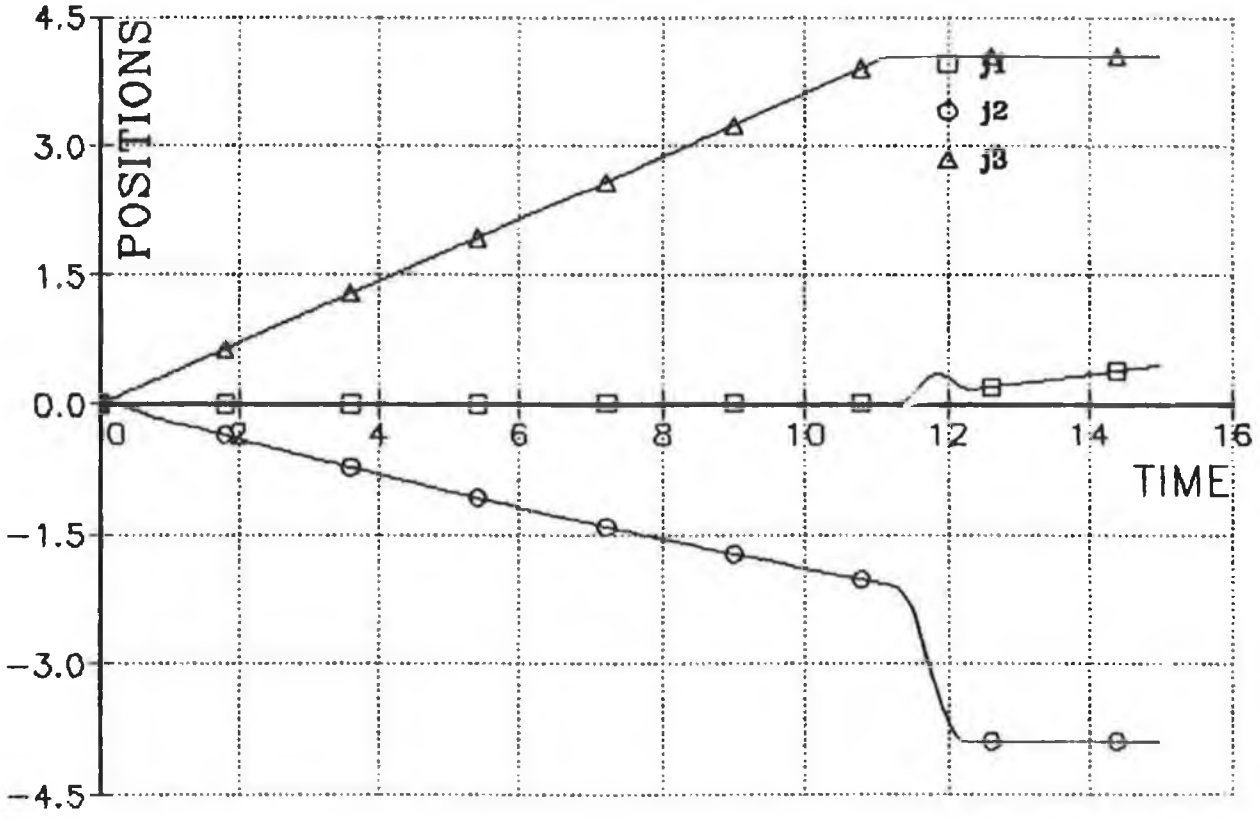


FIGURE 2.16 TEST 6 JOINT VELOCITIES (RAD/SEC) v TIME (SEC)

FIGURE 2.17 TEST 6 JOINT POSITIONS (RAD) v TIME (SEC)



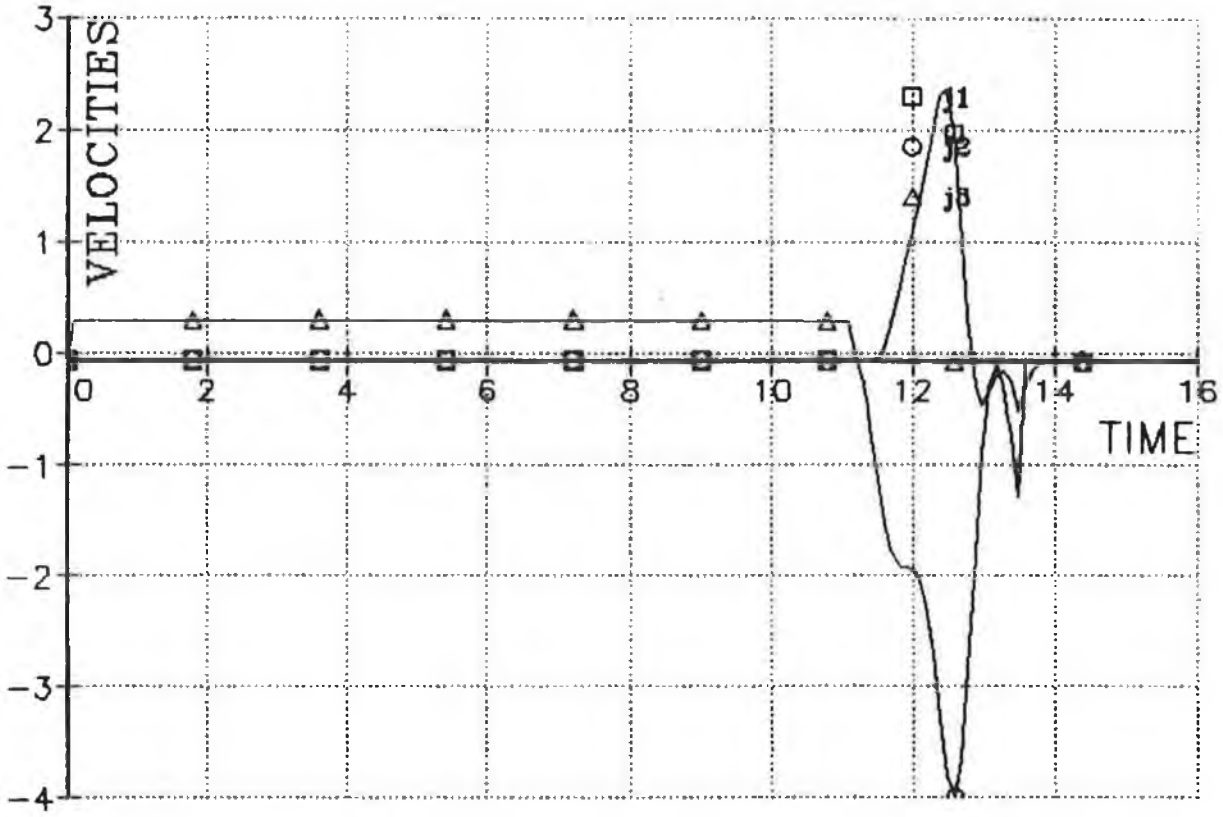
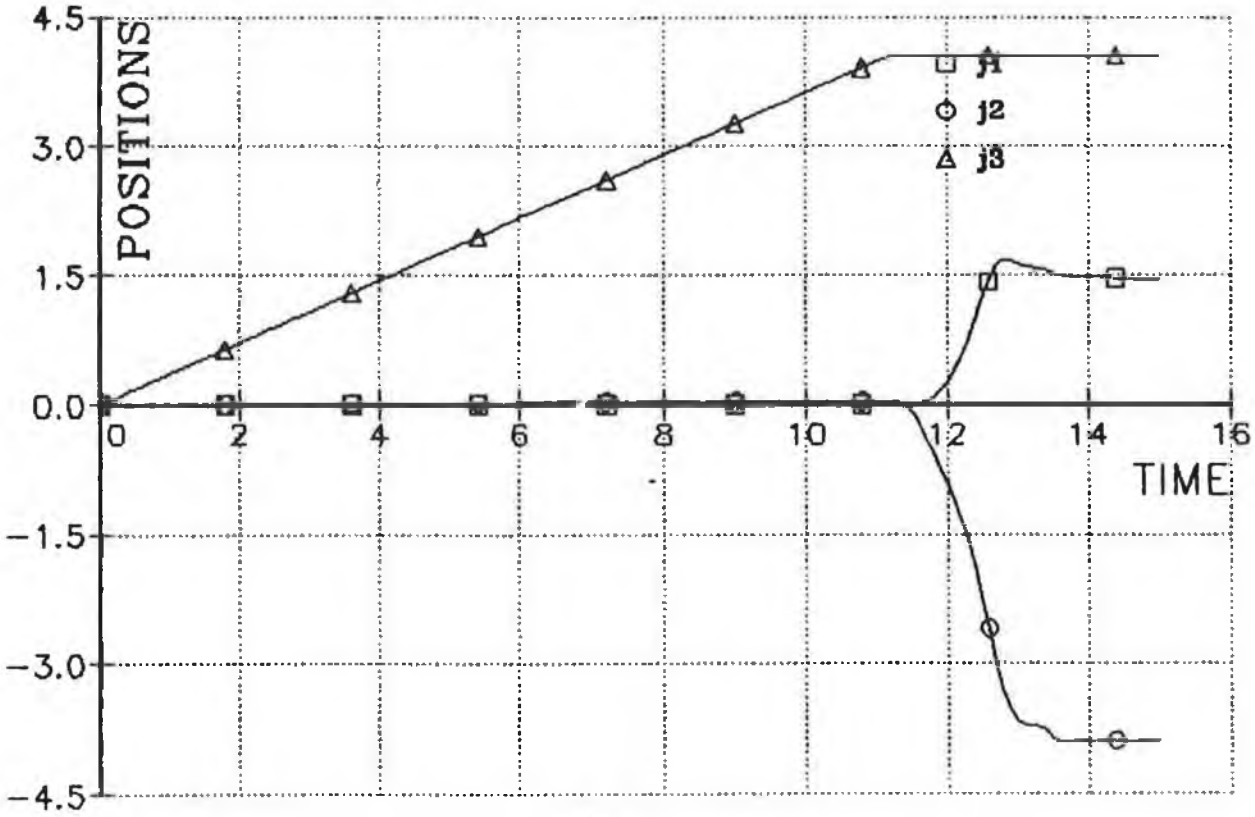


FIGURE 2.18 TEST 7 JOINT VELOCITIES (RAD/SEC) v TIME (SEC)

FIGURE 2.19 TEST 7 JOINT VELOCITIES (RAD) v TIME (SEC)



CHAPTER 3

The New Hardware Controller for the PUMA 560

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 in robotic research environments, however, the primary considerations are ease of modification, adaptability and programmability. These characteristics are essential to manufacture new products for the evaluation of a new sensor system or robot control algorithm design.

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 and more flexible controller for this robot. Before designing a new controller, it is essential to point out the shortcomings in the existing controller, to ensure these shortcomings do not re-appear in the new controller. These shortcomings can occur in three main areas:- the controller software; the controller hardware and in the control algorithm used to control the robot. Once these deficiencies have been recognised it becomes possible to draw up an ideal hardware specification for the new controller and to look at the hardware and software options available to fulfil the ideal specification.

3.1 The Existing PUMA 560 Software

In the case of the PUMA 560 industrial robot, a limited form of task-space control is provided by VAL2 (Victor's Assembly Language) [15]. VAL combines the features of an operating system and a programming language with the aim of allowing the user to teach the robot 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 [16] in terms of flexibility, expandability and is devoid of the ability to implement 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 EPROM, 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 path planning strategies can not be planned off-line.

To allow for large program creation, Unimation [17] suggest two possible alternatives:

1) connect a computer to the terminal port to simulate entry by a human user and
2) connect a computer to the disk port so that an existing program may be downloaded.

The first solution may appear to be more convenient but it still retains the system deficiencies listed above and it is relatively slow due to programming overheads. In order to efficiently gain more controller flexibility and the ability to program in a high level language it is necessary to break away from VAL as an operating system and as a means of programming.

3.2. The Existing Unimation PUMA 560 Control Hardware

The Unimation control hardware [18], see Figure 3.1, 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. From Figure 3.1 it can be seen that 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. 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 in order to carry out the command.

On-line interaction with the user includes parsing, interpreting and decoding VAL commands, as well as the monitoring of possible error messages. When a VAL command has been interpreted, various routines are called to perform scheduling and coordination functions such as :

1) inverse kinematic transforms,
2) joint-interpolated trajectory planning which involves sending new trajectory set-points to the 6503 μPs every 28 milliseconds,
3) acknowledging any messages from the 6503s ,and finally
4) providing lookahead instructions for performing continuous path interpolation if it is required.

At the lower level, the hardware hierarchy [18] consists of six digital servo boards, an analog servo board and six power amplifiers. The six 6503 μPs, reside on the digital servo boards with their EPROM and digital-to-analog converters (DACs). They communicate with the LSI-11/02 computer through a specially designed interface board that routes set-point information to each joint controller. The main functions of the 6503μPs are as follows:

1) to receive and acknowledge set-points every 28ms and provide path interpolation between the current joint and the desired value,
2) to read the encoder position every 28 ms.
3) to update the control error signal between the actual position and the desired position, and,
4) to convert the error signal to an analog control signal necessary to drive the joint motors.

This PUMA 560 hardware suffers from some quite severe limitations. These have been described by Goldenberg [17] :

1) both levels of the controller hierarchy contain only fixed point processors,
2) the existing memory in both hardware levels is inadequate to support large programs,
3) the instruction speed of the Rockwell 6503 μPs and the LSI 11/02 are inadequate to to implement computationally complex control algorithms in real time, 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 hardware.

From these limitations, it can be seen that if a more flexible hardware control structure, capable of implementing complex real time control is required, then the existing Unimation controller hardware must be replaced with a more flexible alternative.

3.3 The Existing PUMA 560 Control algorithm

The primary function of the Rockwell 6503 μP is to implement the existing control algorithm. In doing this the processor computes the error signal between the actual position and the ideal position set-point supplied by the upper level. This error is then sent to the analog servo board which has a lead-lag compensator designed for each joint. The feedback gain of the compensator is tuned to run at a "VAL speed" [17] of 100. There are two servo loops for the control of each joint : the outer loop provides position error information, see Figure 3.2, and is updated by the 6503 μP every 1ms and the inner loop consists of analog devices and a compensator with derivative feedback to put damping on the velocity variable. Both loop gains are constant and are tuned [17] for performance as a " critically damped joint system " at a normal VAL speed of 100 .

One of the main disadvantages of such a method is that the feedback gains are constant and prespecified. It does not have the capacity to update the feedback gains under varying payloads or operating conditions. An industrial robot such as the PUMA 560 is a highly non-linear system. By observing the model developed in Chapter 2, one can see that the these nonlinearities are due to inertial, gravity and other coupling effect. As a result the positions , velocities and accelerations of the PUMA's joints are dependent on the magnitude and variations in these effects. This control algorithm, with its fixed feedback gains, fails to take these variations into account. In fact, the PUMA moves with noticable vibrations at reduced speed [19] because of the controller gains being too high. This makes the robot suitable only for performing simple pick and place tasks that do not require a great deal of precision. To improve the performance of the robot it is, therefore, necessary to replace this control algorithm with one that is capable of tracking some or all of nonlinearities present.

3.4 The New Control Structure Specification

The PUMA 560 controller, because of its two distinct hardware levels, offers what is known as decentralized control structure. Such structures have been widely accepted [20] by the robotics industry due to ease of implementation and tolerence of failure. The main advantage, however, of such a structure is that it allows for the easier implementation of the the control layers discussed in Chapter 1. For this reason, it was decided that the new hardware structure should be primarily a decentralized control structure with capabilities of expansion to include some degree of parallelism. The new control structure should also offer the following:

1) floating point processors to perform mathemathical calculations with high precision and at an adequate speed for real-time control,
2) interfacing hardware which is compatable with the existing Unimation hardware,
3) software that can be written in a 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, and
7) the ability to connect sensory devices through serial, parallel or bus interfaces.

Finally, on top of all these requirements the new control structure must be economically viable. Otherwise it is not a realistic alternative to the existing control structure as far as the robot manufacturer is concerned.

3.5 The New Control Hardware

Numerous implementations of the control structure's upper level , including [21], [22] and [23], have replaced the existing upper level computer with a more powerful central computer. In the case of [21], the LSI-11/02 was replaced by the more powerful LSI-11/23 in conjunction with a Microvax. This combination

more powerful LSI-11/23 in conjunction with a Microvax. This combination provides the user with full floating-point capabilities, high-level language capabilities and an abundance of memory space. The implementation of such a system effectively doubles the cost of the original system [24], making it economically impractical. More recent implementation such as the TUNIS [22] and SIERA [23] 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.

The personal computer chosen was an Intel 80386-based IBM compatable PC [25]. 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 20 MHz,
3) the ability to add a floating-point coprocessor (80387),
5) 1 megabyte of RAM,
6) an 80 megabyte hard disk and
7) 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 a significant amount of floating-point calculations. The expansion slots offer the ability to add extra memory and the ability to interface the new lower level.

To replace the lower level of the controller architecture it was necessary, again, to choose a processor with high speed floating-point capabilities. One option considered was the option chosen by Goldenberg [17]. This implementation uses a

single PC to implement both hardware levels. This means that the tasks of the upper and lower levels have to be executed serially and not in parallel like the existing control hardware. Considering the high speed sampling required for robot control, this serial execution of tasks would limit the complexity of algorithms implemented on this system.

One solution which seems to be gaining in popularity in recent years is to use advanced signal processors (ASPs) to implement this level.. The reasons for thier increased popularity include [26] the reduction in operation and development time which they offer. Also, recent advances in VLSI technologies have meant cheaper ASP chips. For these reasons, it was decided to use an ASP configuration to implement the lower level of the controller. The ASP chosen for this level was the NEC μPD77230 [27]. The μPD77230 is capable of processing digital signals at high speeds and a good degree of accuracy. It can execute arithmethic operations with 32-bit, floating point data (8 bits for exponent and 24 bits of mantissa) or 24-bit, fixed-point data at 150 nS/instruction - including multiplication. Its internal circuitry[14], see Figure 3.3, consists of 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 Loughboro Sound Images Ltd. (LSI) [29]. 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 the internal RAM, the multiplier and the ALU. Such an arrangement makes the processor suitable [28] to process algorithms in which a few operations (such as addition of terms) occur repeatedly. These are the type of operations that occur in the more complex control algorithms such as the computed torque method [30]. In [28] 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 [28] that this algorithm could achieve a throughput of 2,220 setpoints per second. These figures produce controller sampling rates of 0.740 mS and 0.450 mS respectively. These sampling rates are much faster than the existing contoller which implements a much simpler PD based control algorithm. These timing statistics, coupled with the fact that the computed torque method is one of the most computationally complex robot control algorithms means that a μPD77230-based lower level is well capable of implementing control algorithms for in real time robotic control.

3.6 Summary

This chapter has demonstrated why and how the existing Unimation control hardware has been replaced with a new control structure. This demonstration took place in the following sequence:

1) the problems associated with the existing control structure were highlighted,
2) a design specification for a new control structure was discussed and finally,
3) the implementation and characteristics of the new control structure was discussed

The new control structure for the PUMA 560 consists of an Intel 80386-based host computer, three NEC μPD77230 floating-point processor cards. The control structure, like that of the existing PUMA 560 hardware, will be arranged in a hierarchical manner. At the top of the system will be the 80386 based computer which will serve as the supervisory computer and the lower level will consist of NEC μPD77230 processors.

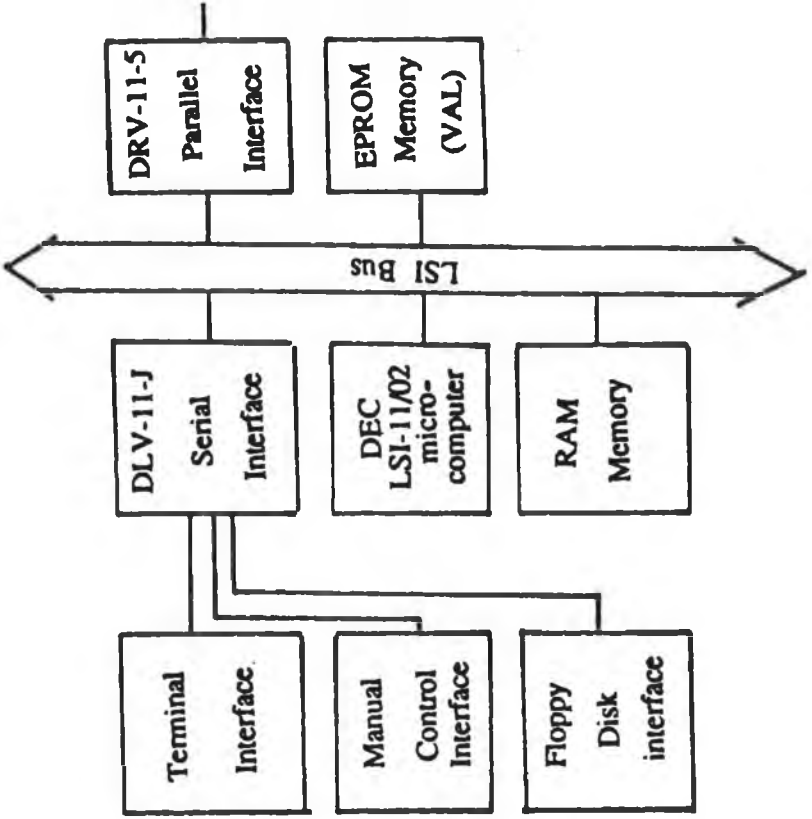
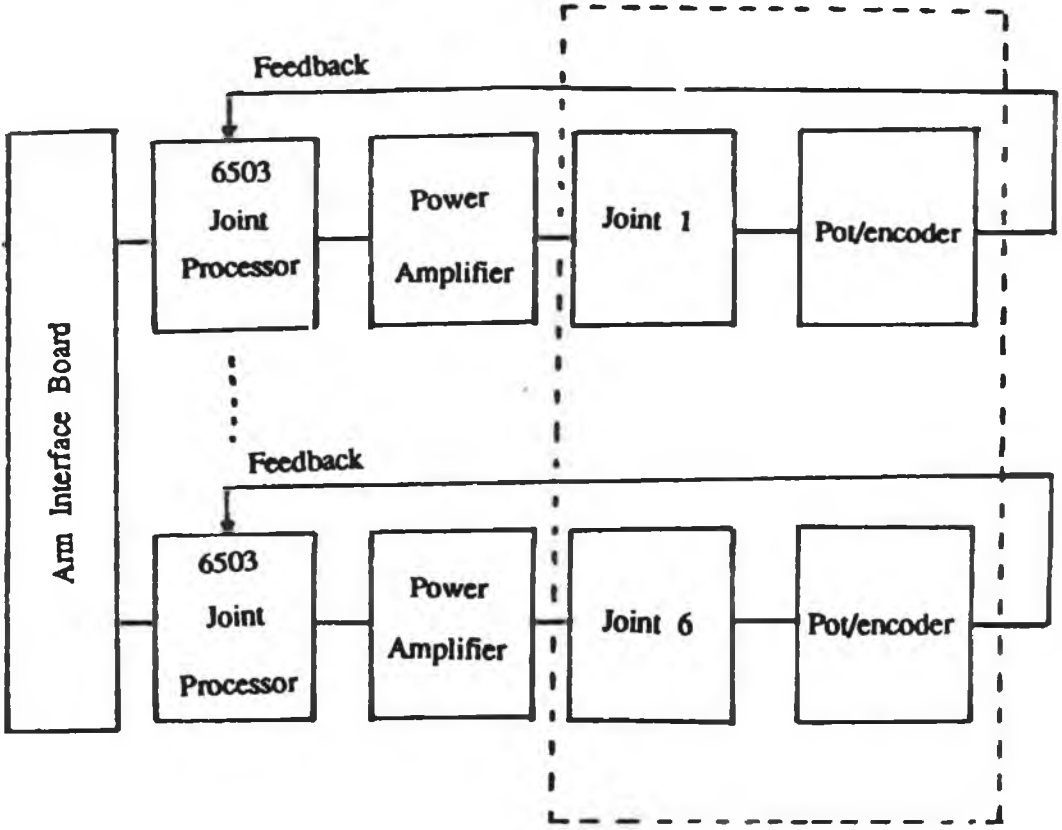


FIGURE 3.1 THE UNIMATION CONTROL HARDWARE


Feedback
6503
Joint
Processor
Power
Amplifier
Joint 1
Pot/encoder
Arm Interface Board
Feedback
6503
Joint
Processor
Power
Amplifier
Joint 6
Pot/encoder

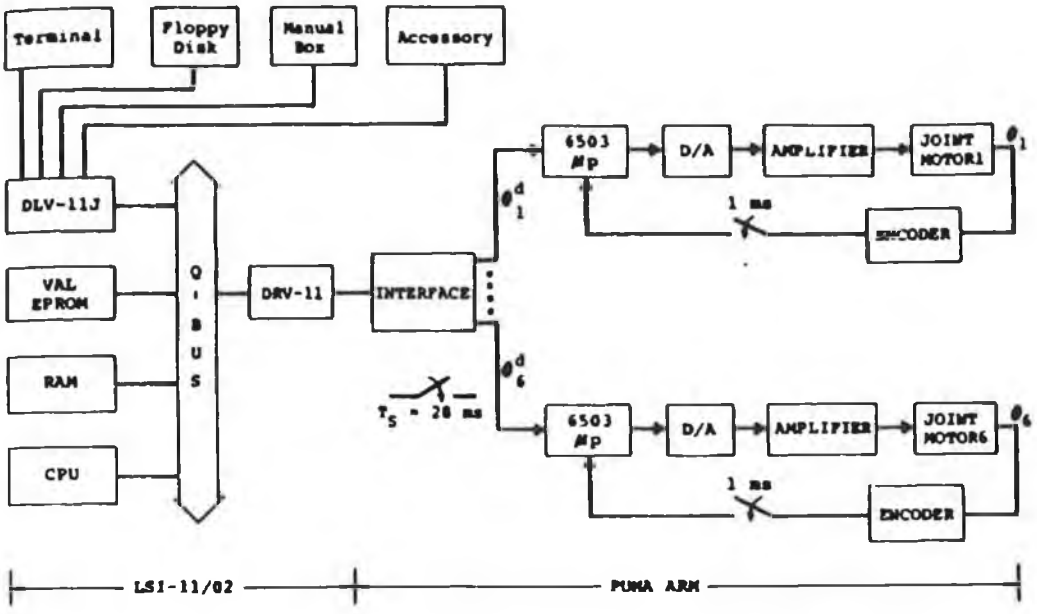


FIGURE 3.2 THE PUMA 560 SERVO LOOPS

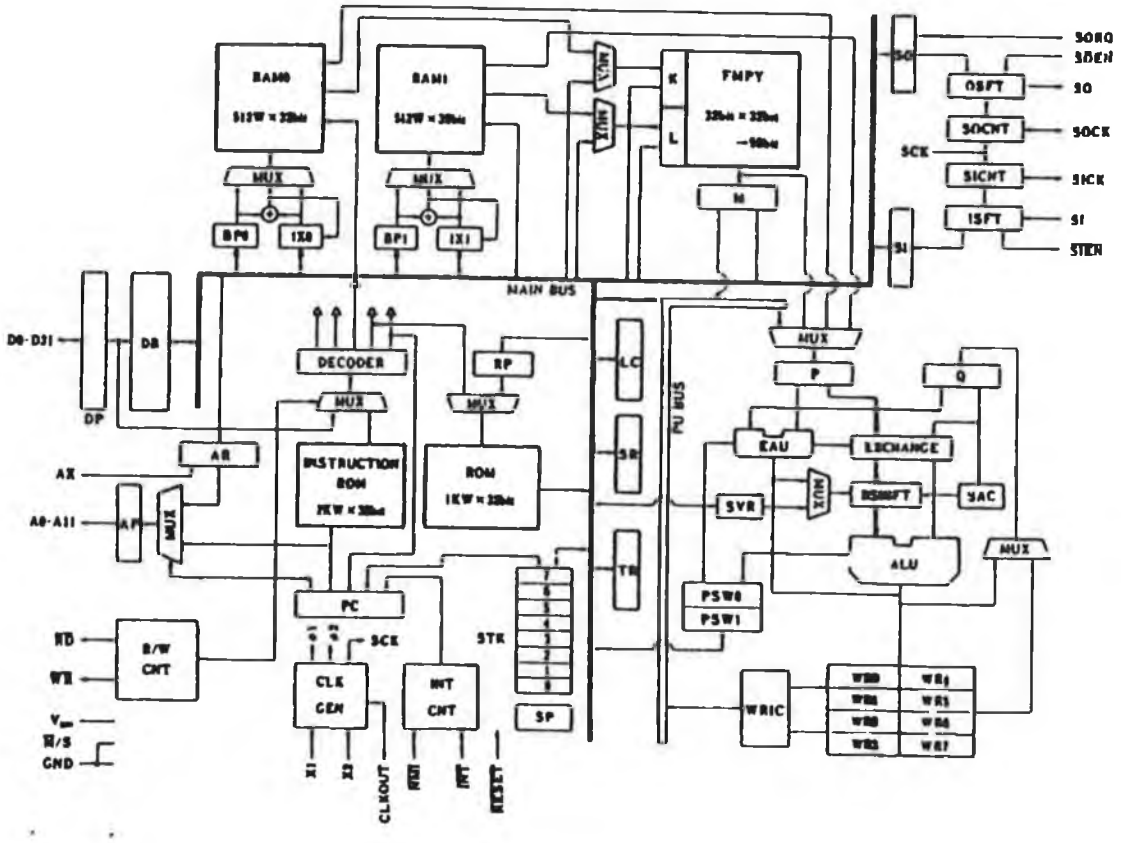


FIGURE 3.3 THE INTERNAL CIRCUITRY OF THE 77320 DSP

CHAPTER 4

Interfacing the New Control Hardware to the PUMA 560

The control hardware designed and implemented in this project, see Figure 4.1, consists of three basic elements: the personal computer; the processor boards and some special purpose interface hardware. The function of the personal 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.

This chapter begins by outlining the various elements involved in DC servo motor control of the PUMA 560's joints. From this outline, the design specification for a new interface between the new control hardware and PUMA 560 is drawn up. Finally, the hardware configuration used to implement the new interface specification is explained.

4.1 The D.C. Servo Motor Control for the PUMA 560

The control of a PUMA 560 arm is achieved through the control of the joint d.c. motors. The robot inputs necessary to control the PUMA 560 [31] are the input voltages used to drive the motors and the voltage signals necessary to apply motor brakes. The robot outputs necessary for control are the outputs of the potentiometric 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 Figure 4.2, 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. For example, if channel A leads channel B, in Figure 4.2, then the A channel state sequence will take the format 11001100 and the B channel sequence will be 10011001.

The Index channel, in conjunction with the position potentiometer, is used to find the initial position. The index channel produces a pulse every motor rotation. In the example given in Figure 4.3, it can be seen that the Index pulse is produced at regular intervals and that 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 absolute joint angle. The potentiometer itself 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 as been 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, and

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 value 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.

4.2 Design Requirements For The New Interface

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 design required falls into four main areas:

1) reading the incremental encoders,

2) reading the potentiometers,

3) driving the DC motors, and

4) applying the motor brakes.

These basic design requirements are discussed in the following subsections.

4.2.1 Reading the Incremental Encoders

The optical encoders are directly attached to the motor shaft and, because of the gear coupling, rotate several times when the joint is driven through its full motion. This gives a precise measurement of relative motion [32]. The A and B channels determine both the amount and direction of the rotation in discrete steps. The index channel produces a short pulse each motor revolution (360-degrees) which can be used by the system, in conjunction with the position potentoimeter data, to determine absolute position.

The way in which the output of the A and B channels detect the relative motion of the joint can be seen in Figure 4.2. The direction of rotation (clockwise or anti-clockwise) can be determined by observing the state transitions of these two channels. These transitions can be interpreted to perform three operations; i) increment joint position (A leads B); ii) decrement joint position (B leads A); and iii) remain at same position (no state change).

Almost all the PUMA 560 joints [32], 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 maximimum amount of transitions that are lightly to occur. For this reason 16-bit counters (maximum count 65526 = 2^{16}) are sufficient to keep track of the PUMA 560's joint movements.

4.2.2 Reading the Position Potentiometers

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° for 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 were produces 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).

An analog to digital converter (ADC) is used to measure the potentiometer voltage. This must have an input range of 0 to +5 volts and a resolution 0.046 volts. This corrosponds to an accuracy of 0.92% of full scale. Since, a 7-bit ADC has an accuracy of 0.78% over the full scale, an ADC with 7 or more bits of resolution is suitable for this application. Since the potentiometer is not actually part of the dynamic control system associated with the arm, there is no constraint on the conversion speed. However, to provide more accuracy at a little extra cost a 12-bit ADC was used for the potentiometer measurement.

4.2.3 Driving the Robot Motors

The drive current and voltage needed to drive a DC motor is entirely motor dependent. It was therefore not necessary to design power amplifiers for the system since satisfactory ones already exists. 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 [33] can be conveniently used because they were designed explicitly with this robot in mind. Using these amplifiers simplifies external connection to the arm's joint motors. In addition, the Unimation power amplifier unit contains a Miscellaneous Functions Unit [33] (MFU) which provides useful safeguards which can be monitored to prevent the arm from being damaged. These safeguards include the ability to monitor the amplifiers' input current and temperature to see if they are operating within the values specified for the 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 resolution means that the new drive signal is 4 times more accurate than the original one.

2.2.4 Releasing the Robot's Brake

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. The brake is similar to a DC relay: when current flows through its coil, the brake is released otherwise it remains applied. Furthermore, 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 [33] are wired together. The MFU mentioned above contains the circuitry needed to apply or release the brake. This circuitry can be contolled by setting or resetting a digital input to the MFU known as BRAKE RELEASE ENABLE [33].

4.2.5 System Timing

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. One way to control the sampling interval is to use only the control software and time delay loops as a time base. This, however, is not very acceptable for numerous reasons. Firstly, the delay loop must be altered every time the controller software is adjusted. Also, control software often executes at speeds dependent on the sampled data itself, making constant sampling impossible. An altenative to this method is to interupt the CPU using a hardware timer. The hardware timer can take the form of a programable up-counter. This counter should be free running from an N Hz clock giving a clock period of 1/N sec. The sample period can therefore be set in

terms of an integral number of clock cycles, each clock cycle adding 1/N secs. The program that requires this sampling interval can then be written as an interupt service routine. Then if a timer interrupt occurs the program can commence execution.

4.3 Hardware Design of the New Interface Board

This section details how the design requirements above are used in the design of the new controller interface. From the above requirements, 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,
4) a sample rate generator and
5) the interface with the new lower level hardware

The block diagram in Figure 4.4 shows the basic subsystems which are involved in the new interface. From this, it can be seen that the new system for each joint consists of two main components; an analog input/output board and the New Interface Board (NIB). The subsystems 1, 2 and 5 above are the basic components of the NIB while the subsystems numbered 3 and 4 are present on the analog board. The following is an individual description of the operation of these two boards.

4.3.1 The Analog I/O Board

The analog boards used, supplied by LSI [34], 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 5 μs, while the two DACs have a settling time of 3 μs. One of the input channels present was used for reading the feedback potentiometer, while one of the output channels was used to drive the motor amplifier. The reason why there are more I/O channels than is necessary is to make the controlller 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 8 Mhz clock. This timer, upon completion of a sample period, has the ability to interrupt both the upper and lower levels of the contoller hardware. In the case of the PUMA 560, it must be possible to generate these intervals of between 125 nS and 30 milliseconds. These are well within the range of the sampling periods necessary for real-time control of the PUMA 560.

4.3.2 The New Interface Board (NIB)

The NIB is a an interface card designed specifically for this project. The function of this card is to interface the encoder counter subsystem, the PUMA analog signals and other control signals with the μPD77230 board. The following sections describe the subsystems which make up this board:

1) the encoder counter subsystem,
2) the encoder reset subsystem and
3) the processor board interface.

These subsystems are discussed in the following sections.

4.3.2.1 The Encoder Counter Subsystem

A 16-bit up-down counter, made up of four 4-bit cascaded counters, was 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 [35]. This counter uses a 1 MHz clock which is generated on the NIB by a 1 MHz crystal. This value of clock frequency was chosen becuase it is much greater than the maximum frequency of the encoder state changes. So all state changes will be sampled.

The enable and the up-down signals of the counter are derived from the channel A and B 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 decode logic. The basic idea of the scheme is presented here and illustrated in Figure 4.5. From Figure 4.5 it can be seen that the encoder signals A and B are both fed through 2-stage shift registers clocked by the 1 MHz clock. The output of the first stage (A', B') are synchronized versions of the A and B inputs since thay 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 determenine which direction the count must go: up or down. Table 4.1 shows all the possible combinations of these states and the decoded command signals for the counter. An EPROM was used to decode the counter inputs. This EPROM has as its address inputs the A, the B states and the counter reset line. The outputs are the decoded command signals for the counters generated from Table 4.1.

TABLE 4.1 Command Signal Generation for the Encoder Counters

EPROM ADDRESS	EPROM O/P (COUNTER I/PS)			OPERATION
	ENT	D/U	LOAD	
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.
15	0	0	1	INC.
14	0	1	1	DEC.
15	1	1	1	NOP.
16	0	0	0	CLEAR.

4.3.2.2. The Encoder Reset System

The basic scheme for the establishing the PUMA's initial position is to rotate each motor until the index is found and define this position as position zero. This can be done using software to sample the index signal at fixed intervals. This method can however prove to be quite slow if the sampling rates are not very high. An example of this lack of speed would be a sampling rate of 100 Hz. This

would mean that the motor would have to rotate at a speed of less than 100 degrees per second. When gearing ratios are taken into account, this would mean a joint speed of two degrees per second.

A hardware scheme was used to overcome this limitation. The circuit used is illustrated in Figure 4.6. Each new counter reset circuit has two flipflops and a NAND gate associated with it. The circuit is asynchronously 'armed' or enabled via an ARM RESET signal. Once armed, the next occurance of an index pulse generates a single reset pulse sent to the associated counter circuit. When the reset pulse is issued the circuit disarms itself so that further occurences 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 occured.

4.3.2.3. The Processor Board Interface

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 either the encoder counter outputs or the analog board. The design here involved the use of 74623 [35] octal bus tranceiver 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 were derived by decoding the 14 I/O addresses as shown in Table 4.2.

TABLE 4.2 Address DecodingI/O for the New Interface

	I/O PORT ADD.				TRANCEIVER CONTROL SIGNALS			
OPERATION	A3	A2	A1	A0	GAB1	GAB2	GAB2	GBA2
NO OPERATION	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

X = don't care state

In addition to the I/O ports the µPD77230 board has a number of digital I/O lines which were 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 of the encoder reset circuit. The input line, BIT IN, is used to monitor the ARM STATUS line to see if an index has occured. Finally, the TRIGGER IN input is used to check if an over-current or over-temperature error condition [3] has been found by the MFU.

4.4 Summary

This chapter has shown how the new control hardware of chapter 4 was interfaced to the PUMA 560 industrial robot. The interfacing procedure took the following steps:

1) a brief description of PUMA 560 sensors was given,
2) the design requirements for interfacing the new control hardware with the PUMA 560 were then presented,
3) finally, the hardware design used to implement the new interface is described.

The new interface is similar to the existing Unimation interface in that it uses the Unimation power amplifiers and MFU. It also adds a degree of flexiblity to the new control hardware not found in the Unimation interface. Its flexiblilty lies in the increased number of I/O channels it provides and the increased accuracy of these channels. It also provides a flexible sample rate timer which is capable of producing sample rate times in a range suitable for real time control.

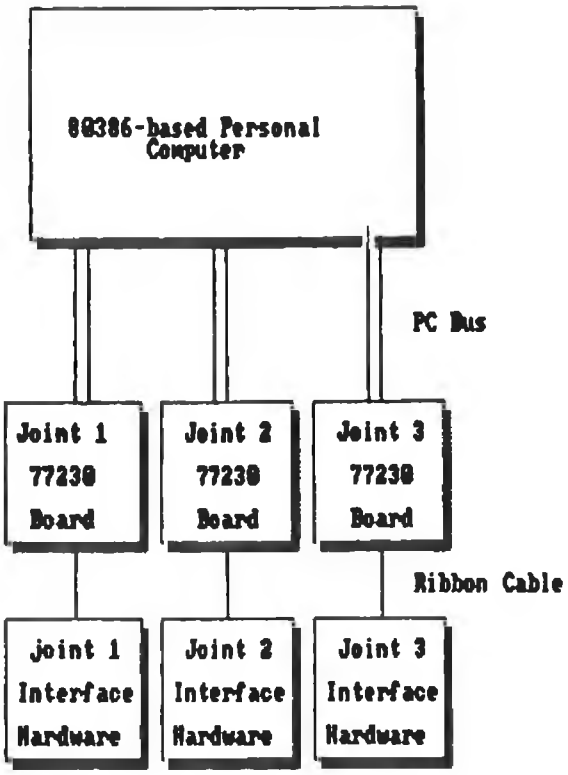


FIGURE 4.1 THE NEW CONTROL HARDWARE

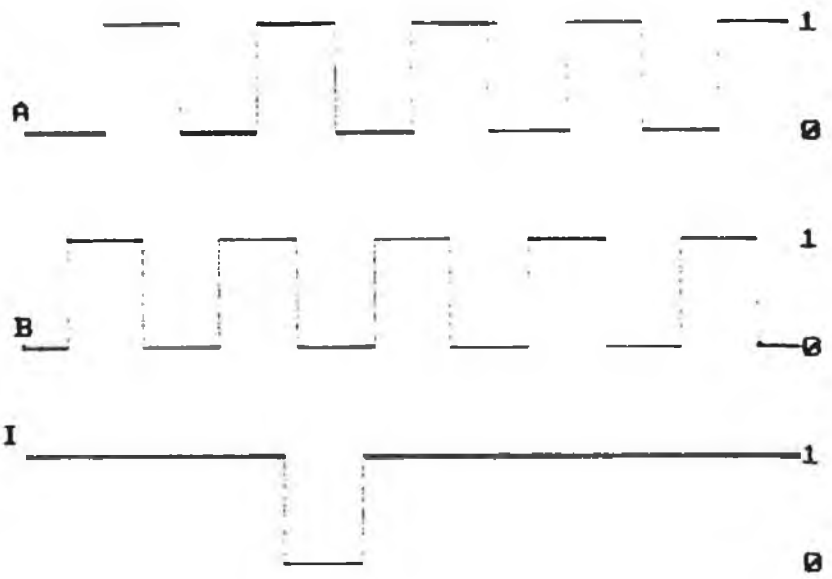


FIGURE 4.2 THE ENCODER A, B AND INDEX CHANNEL OUTPUTS

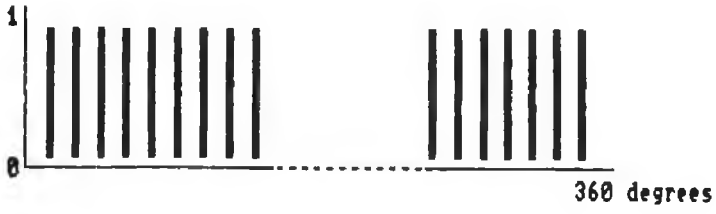
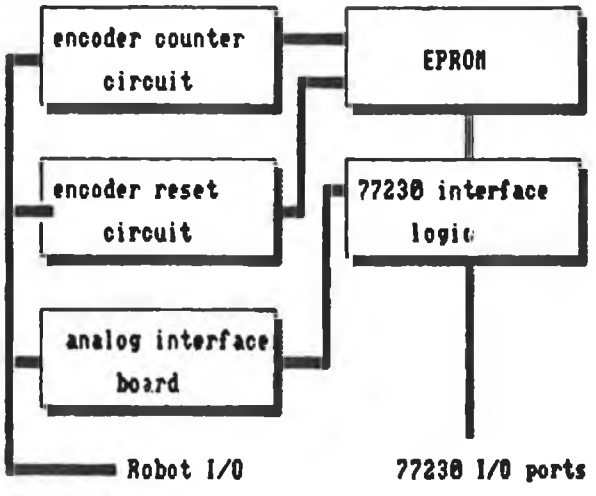


FIGURE 4.3 PERIODIC INDEX PULSE OCCURANCES

FIGURE 4.4 THE NEW INTERFACE SUBSYSTEMS



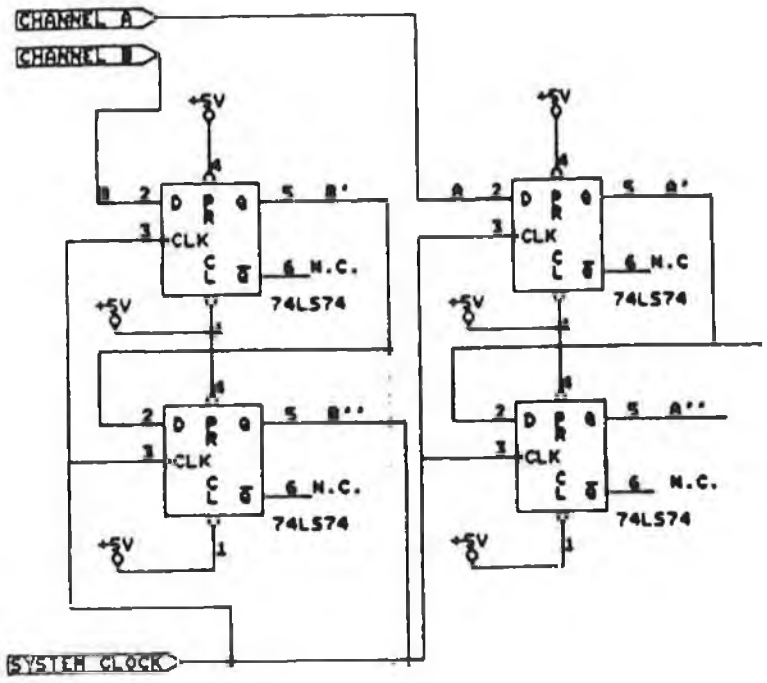


FIGURE 4.5 THE ENCODER COUNTER CIRCUITRY

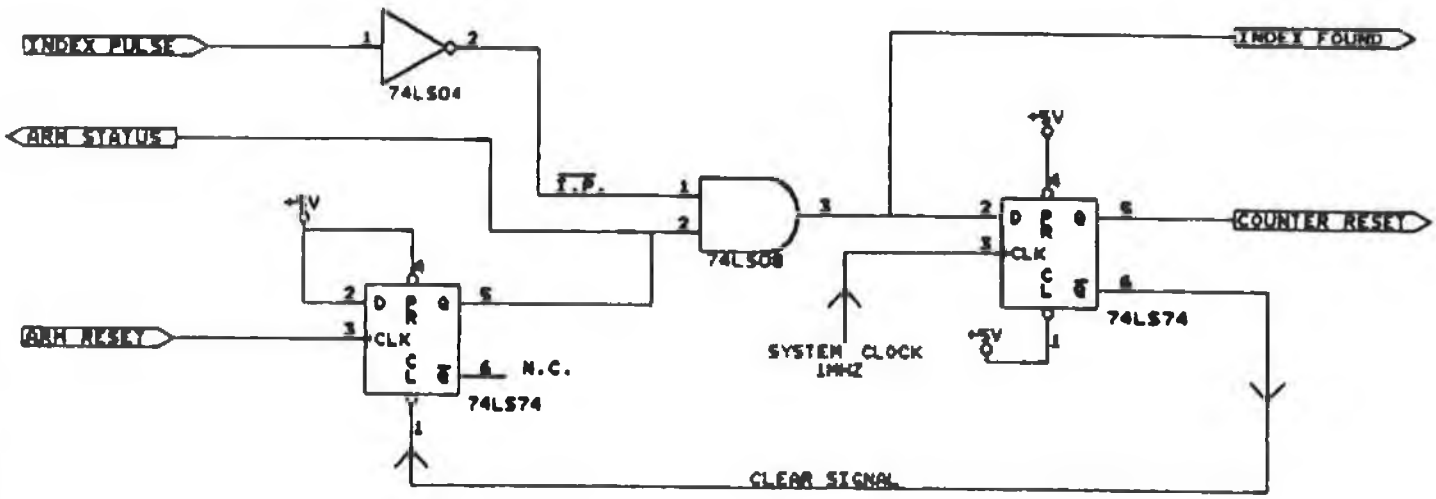


FIGURE 4.6 THE ENCODER RESET CIRCUITRY

CHAPTER 5

Software Considerations for the New Control Structure

The purpose of this chapter 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, use can be made of existing software tools to achieve the inter-processor communication desired.

The chapter begins by discussing the architectural considerations involved in the new control structure. Included in this is the choice of operating system and the choice of high level programming language. The role of computational elements in the new controller is then described. This involves examining the new control structure's real-time functionality under the headings of inter-processor communication, calculation speed and task coordination.

5.1 Architectural Considerations

The benefits of flexible automation through robotics can be achieved by using standard programmable elements [36] and by reserving task-specific activities to those tasks that require them. In the case of the new controller system, the standard programmable elements are the 80386-based personal computer and the μPD77230 boards - the interface hardware is task-specific (i.e., built specificially for the PUMA 560). 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 familarity 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 asfile management, batch file generation and on-line debugging tools are available.

Several standard computer operating systems have been developed in recent years. Two of the most commonly used of these are MSDOS and UNIX. Both these operating systems provide the support and availability of numerous languages, compilers and assemblers, as well as providing utilities for file manipulation, directory manipulation and networking. The operating system chosen for this controller was the MSDOS system. This is because MSDOS was cheaper and was found to support all the software tools required for the developent of the new controller software.

The software tools used for the the new controller consist of a Microsoft C language compiler, an NEC μPD77230 monitor [37] 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 Micosoft 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 a hexidecimal format using the object converter. In this format, the programs can be downloaded into the μPD77230 memory space and executed. The down loading and execution can be achieved by using either the LSI monitor or C drivers specificially written for this purpose that came with the boards. The LSI monitor has additional features which include editing and debugging facilities.

5.2 The Computational Elements of the New Controller

The computational elements of the new control structure involve a wide range of applications, including the role 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 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 functionality.

From an interface point-of-view, the functionality of the new controller can be described as the control structures ability to efficiently interpret information from the robot's feedback sensors and with its ability to generate drive signals for the robot amplifiers. The performance of these operations is dependent on the addressing system listed in Table 4.2. From this table it can be seen that by reading or writing to the 14 input/output addresses of the μPD77230 board's memory, one can interpret the sensor information or generate the drive signal for the motor amplifiers. These read and write operations can be performed efficiently by using assembly language modules stored in the processor's internal ROM [37]. By using these modules, the processing time taken to write or read one word, to or from the interface, amounts to 6 clock cycles or 750 nS. Since standard robot control sampling periods are in the range 0.5mS to 5 mS these operations take up less than 0.0015% of the time interval between samples.

Another 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 and the inter μPD77230 board data transfer rate. The upper to lower hardware communication involves the downloading of position setpoints to the μPD77230 boards from the personal computer. The μPD77230

boards are mapped into the input/output addressing area of the PC, see Figure 5.1. The address map of each µPD77230 board takes up 8 address in the PC's input/output area. The function of the control register shown in Table 5.1 is to enable or disable the processor and any interrupts to the PC. The status register shown in Table 5.1 is used to monitor the operation of the µPD77230. The downloading of the setpoints is achieved by the following sequence of events:

1) the memory address on the processor board where the setpoints is to be written is selected by writing to the Base +6 and the Base +7 addresses of the PC's I/O area.

2) an interupt is sent to the µPD77230 on a PC write.

3) the set-point is then written to the µPD77230 board by writing to the PC's I/O addresses Base + 1 to Base + 4.

TABLE 5.1 Address map for 77230-PC Interface

Address	Read	Write
Base + 0	Status Register	Control Register
Base + 1	L S Byte	L S Byte
Base + 2	Mid Byte	Mid Byte
Base + 3	M S Byte	M S Byte
Base + 4	Exponent	Exponent
Base + 5	Status Register	Control Register
Base + 6	Status Register	Address L S Byte
Base + 7	Status Register	Address M S Byte

This write operation causes an NMI interupt to the µPD77230 which is only cleared after the most significient byte of the setpoint has been transferred. Upon transferral of this byte the interrupt is cleared and the µPD77230 is free to access this new setpoint data. The entire write operation for one board consists of writing a single byte to each of the seven different PC I/O locations. A single write

operation was found take approximately 1 μs using a Microsoft C output function to write to one of the I/O port addresses. This means that the entire transferral of the data takes about 8 μs for the PC. Considering the fact that setpoints are usually downloaded at intervals of between 10 and 30 ms[36], this means that the downloading of the setpoints takes up a small percentage of the time needed for setpoint generation. The inter-μPD77230 communications take place using high-speed serial links. It is possible to use program modules stored in the proesssor's internal ROM to carry out the serial transfer of data words. These serial read and write operations take only 4 instruction cycles (500ηs) to execute.

The calculation functionality of the new hardware can be defined in terms of 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., 150 ns). Double-precision floating-point additions were found to take 10μs and multiplications approximately took 32 μs each.

In the lower level computational functionality involves 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 [37] or 150ns. In the floating-point case, addition and subtraction take 5 instruction cycles, with multiplication taking 6 instruction cycles. This means that the lower level is capable of performing thousands of additions and multiplications in one millisecond. The advantage can be seen more clearly if one examines the algorithms developed in [38], [39] and [40]. These algorithms are among some of the most computationally complex available, yet preliminary calculation suggest that these algorithms can be implemented in real-time using the μPD77230 boards. In the case of [38] and [39] these calculations show that both could implemented in times less than 0.5 ms, while [40] could be implemented in a time which is less than 0.8 ms. The same algorithms, if implemented on the

existing Rockwell 6503 μPs would require that the sampling interval be increased by a factor of 10. Such long sampling intervals would be unsuitable for real-time control, considering the speed [36] at which manipulator dynamics change.

5.3 Summary

This chapter has provided an insight into the computational aspects of the new controller. Among these aspects was the choice of operating system and the choice of a high-level programming language for the new control structure. The chapter also explained the efficiency of the new controller in performing the following tasks:

1) communication between the various processors of the new controller,
2) the transferral of data across PUMA 560-μPD77230 interface and finally,
3) the performance of floating point arithmethic.

The chapter has shown how the new hardware, in conjunction with the new software, is capable of performing all three of these tasks in real time and using a high level language.

CHAPTER 6

The Inverse Kinematic Problem for The PUMA 560

One of the main requirements mentioned for the new controller system in Chapter 3 was that the new controller should have the ability to implement real-time path planning. This chapter develops an inverse kinematic algorithm for use in such path planning applications.

Robot arm kinematics deals with the analysis of the motion geometry of a robot arm without regard for the forces which cause the motion. In other words, it deals with the analytical description of the spatial displacement of the robot as a function of time. This chapter explains how, given the joint angles of the PUMA 560 and the physical dimensions associated with the PUMA 560, it is possible to express the robot hand position and orientation with respect to a reference Cartesian coordinate system. This is known as the forward kinematic problem. The equations developed from the forward kinematics algorithm are then used to derive a set of joint angles from known Cartesian position coordinates. This is called the inverse kinematic problem for the PUMA 560.

6.1 The Forward Kinematics of the PUMA 560

To derive the forward kinematic equation for the PUMA 560 it is necessary to assign individual coordinate frames to each joint, as shown in Figure 6.1. By examining these coordinate frames it is possible to see that the movement of any joint i can be represented in terms of the coordinates of joint i-1 by using a series of joint rotations and linear transformations. Devanit and Hartenberg [41] proposed a system by which these transformations could be represented as a 4x4 transformation matrices. Each of these transformation has two frames associated with it: a reference frame and that of the joint whose movement it is required to

represent in the reference frame coordinates. For example T^a_b describes the transformation required to superimpose the coordinate frame of joint 'a' on to the coordinate frame of joint 'b'. The general form of the Transformation matrices as described by Davanit and Hartenberg is as follows:

$$T^{n-1}_n = \begin{vmatrix} \text{Cos}\theta n & -\text{Sin}\theta_n\text{Cos}\alpha_n & \text{Sin}\theta_n\text{Sin}\alpha_n & a_n\text{cos}\theta_n \\ \text{Sin}\theta_n & \text{Cos}\theta_n\text{Cos}\alpha_n & -\text{Cos}\theta_n\text{Sin}\alpha_n & a_n\text{sin}\theta_n \\ 0 & \text{Sin}\alpha_n & \text{Cos}\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (6.1)$$

where Θ_n = joint angle of joint n

d_n = the distance between links n and n-1,

a_n = the link length,

α_n = the link twist.

These quantities in the transformation matrix can be seen for the various joints of the PUMA 560 in Figure 6.1. The fourth row of this matrix serves only to square the matrix for inversion purposes and contains no information about the joint coordinate frames.

The position of any robot joint "n" can be expressed in terms of the coordinate frames of any joint "n-m" (m>1) by using these coordinate transformation. For example, if T^0_1 is the coordinate frame of joint 1 expresses in terms of robots base coordinates and T^1_2 is the coordinate frame of joint joint 2 expressed in terms of joint 1, then the product of these two matrices allows the position of joint 2 to be expressed in the base coordinates of the robot. It is common practice to choose the reference frame for all a robot's joints to be the base coordinates of the robot. The transformation matrix of joint 6 of the PUMA 560 robot can be expressed as the product of a number of individual joint transformation matrices as follows:

$$T^0_6 = T^0_1.T^1_2.T^2_3.T^3_4.T^4_5.T^5_6 \quad (6.2)$$

or

$$T^0_6 = \begin{bmatrix} u_x & o_x & a_x & p_x \\ u_y & o_y & a_y & p_y \\ u_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.3)$$

The quantities p_x, p_y, p_z make up a position vector p which contains the position of the robot's wrist with respect to the robots base coordinates. The quantities a_x, a_y and a_z make up a vector that describes the approach of the robot hand in terms of the base coordinates, while the quantities o_x, o_y and o_z can be used to describe the orientation of the robot hand in the same coordinates. The quantities u_x, u_y and u_z make up a third vector which is the cross product of the o and a vectors. The o, a and u vectors for the PUMA 560 hand can be seen in Figure 6.2.

The position of joint 3 in Cartesian base coordinates can be found by calculating the T^3_0 matrix as follows:

$$T^0_3 = T^0_1 . T^1_2 . T^2_3 \qquad (6.4)$$

The T^0_1, T^1_2 and T^2_3 for PUMA 560 can be derived by using the matrix in equation (6.1) and the link information for the PUMA 560 shown in Table 6.1. These matrices take the following forms:

$$T^0_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.5)$$

$$T^1_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T^2_3 = \begin{bmatrix} C_3 & 0 & S_2 & a_3C_3 \\ S_3 & 0 & -C_3 & a_2S_3 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where C_i and S_i are the sine and cosine of joint angle i.

TABLE 6.1 Link Parameters for the PUMA 560

joint	α	d	a
1	-90	0	0
2	0	0	a_2
3	90	d_3	a_3
4	-90	d_4	0
5	90	0	0
6	0	0	0

a_2 = 17cm, a_3 = 0.75cm, d_3 = 4.937cm and d_4 = 17.00cm

From the product of the three transformation matrices in equation (6.4) it is possible to obtain the position of the third joint. This is achieved by examining the first three elements of the fourth column of the $T^0{}_3$ matrix. The position of the third joint is therefore given by a position vector which has the form:

$$p_x = C_1.[\ d_3S_{23} + a_3C^2{}_3 + a_2C_2\ ] - S_1d_3 \qquad (6.6)$$

$$p_y = S_1.[\ d_4S_{23} + a_3C_{23} + a_2C_2\ ] - C_1d_3 \qquad (6.7)$$

$$p_z = -[\ d_4C_{23} + a_3S_{23} + a_2S_2\ ] \qquad (6.8)$$

where S_{23} and C_{23} are the $\sin(\Theta_2+\Theta_3)$ and $\cos(\Theta_2+\Theta_3)$, respectively. The quantities in equations (6.6), (6.7) and (6.8) are the solutions to the forward kinematic equations for the three primary joints of the PUMA 560.

6.2 The Inverse Kinematic Problem for the PUMA 560

The inverse kinematic problem for the three primary joints of PUMA 560 can be defined as the finding of the primary joint angles that will position the robot hand at a desired position in the robot's workspace. The inverse kinematic solution for a robot is used to generate the individual joint angles required to ensure that the robot hand moves along a particular trajectory.

Usually the inverse kinematic problem can be solved by using either an algebraic, iterative or geometric approach. Several investigations have tried to solve the problem for the PUMA 560 robot using an algebraic approach [42] [43] [44]. This approach suffers from the fact that the solution does not give a clear indication of how to select the correct solution from the several possible solutions for a particular arm configuration. The user must rely on intuition to select the correct solution. The iterative solution of [41] requires more computations than the algebraic approach and it does not guarantee convergence to the correct solution. If the manipulator under consideration has all revolute joints and the geometry of the first three joints and the last three joints meet at one point then a highly efficient geometric approach can be applied to solve the inverse kinematic problem [45]. Fortunately the PUMA 560 falls into this category of manipulator.

For the three primary joints PUMA 560 joints, there are 4 possible values of the first three joint angles that will give the same position vector for joint 3. The algorithm detailed in this section outlines how, with the use of configuration variables, it is possible to obtain a unique solution for the positioning of joint 3 of the robot.

The joint angle of joint 1 is defined by rearranging the terms in equation (6.6) and as follows:

$$S_1 = \frac{ARM.P_y(P_x^2 + P_y^2 - d_3^2)^{\frac{1}{2}} - P_y d_3}{P_x^2 + P_y^2} \qquad (6.9)$$

$$C_1 = \frac{ARM.P_x(P_x^2 + P_y^2 - d_3^2)^{\frac{1}{2}} - P_y d_3}{P_x^2 + P_y^2} \qquad (6.10)$$

The configuration variable ARM can have two possible values: +1 or -1. The reason for this is the fact that a robot can have either a right or left arm. The PUMA 560 used in this project is right armed. This means that the variable ARM is assigned a value of +1. In order to evaluate the angle Θ_1 an arctangent

function , atan2(x/y) which returns $\tan^{-1}(x/y)$ adjusted to the proper quadrant can be used. It is defined as:

$$\begin{aligned} \text{atan2}\ (y/x) \quad = \quad & 0 < \theta < 90 \quad \text{for } +x \text{ and } +y \\ & 90 < \theta < 180 \quad \text{for } -x \text{ and } +y \\ & 180 < \theta < -90 \quad \text{for } -x \text{ and } -y \\ & -90 < \theta < 0 \quad \text{for } +x \text{ and } -y \end{aligned} \qquad (6.11)$$

This function allows the joint angle of joint 1 to be calculated as follows:

$$\theta_1 = \text{atan2}(S_1/C_1) \qquad (6.12)$$

The angle for joint 2 can be defined by rearranging the equations (6.7), (6.8) and (6.9) to give the folowing equations:

$$S_2 = S_\alpha . C_\beta + (\text{ARM.ELBOW}) . C_\alpha . S_\beta \qquad (6.13)$$

$$C_2 = C_\alpha . C_\beta - (\text{ARM.ELBOW}) . S_\alpha . S_\beta \qquad (6.14)$$

Joint 2 of a right handed manipulator can be used to achieve the wrist positioning using two angle configurations. The ARM.ELBOW product in equations (6.13) and (6.14) can be used to define the two possibilities. The values which define these configurations are shown in Table 6.2. The RIGHT & ABOVE configuration in Table 6.2 refers to when the elbow of the RIGHT handed manipulator being positioned above the wrist.The RIGHT & BELOW configuration refers to when the manipulator wrist is positioned above its elbow. These configuration can be seen more clearly in Figure 6.3.

Table 6.2 Arm Configurutations for Joint 2 & 3 of the PUMA 560

Arm Configuration	ARM	ELBOW	ARM.ELBOW
RIGHT & ABOVE	+1	+1	+1
RIGHT & BELOW	+1	-1	-1

The terms S_α, S_β, C_β and C_α in (6.13) and (6.14) are the Sines and Cosines of angles α and β. These angles are defined as follows:

$$S_\alpha = \frac{-Pz}{R} \tag{6.16}$$

$$C_\alpha = \frac{-ARM.r}{R} \tag{6.17}$$

$$C_\beta = \frac{a_2^2 + R^2 + (d_4^2 + a_3^2)}{2.a_2R} \tag{6.18}$$

$$S_\beta = (1 - C_\beta^2) \tag{6.19}$$

where the R is the distance from the origin of the robots base coordinates, to the robots wrist position defined by:

$$R = (P_x^2 + P_y^2 + P_z^2 - d_3^2)^{\frac{1}{2}} \tag{6.20}$$

The quantity r is the perpendicular distance from the z axis of the base coordinates to the position of the robots wrist.

$$r = (P_x^2 + P_y^2 - d_3^2)^{\frac{1}{2}} \tag{6.21}$$

The joint angle of joint 2 can then be written as:

$$\theta_2 = \text{atan2}(S_2/C_2) \tag{6.22}$$

The joint angle for joint 3 can be defined using the following equations:

$$S_3 = S_\gamma C_\beta - S_\beta C_\gamma \tag{6.23}$$

$$C_3 = C_\gamma C_\beta + S_\gamma S_\beta \tag{6.24}$$

where the angles γ and β are defined by the following equations:

$$C_\gamma = \frac{a_2^2 + d_4^2 + a_3^2 - R^2}{2a_2 (d_4^2 + a_3^2)} \qquad (6.25)$$

$$S_\gamma = \text{ARM.ELBOW.}(1-C_\gamma^2)^{\frac{1}{2}} \qquad (6.26)$$

$$C_\beta = d_4/r \qquad (6.27)$$

$$C_\beta = a_3/r \qquad (6.28)$$

where R is defined as in equation (6.20) and the quantity r is defined by:

$$r = (d_4^2 + a_3^2)^{\frac{1}{2}} \qquad (6.29)$$

Using equations (6.24) through to (6.29), the solution of the angle of the third joint is obtained as follows:

$$\theta_3 = \text{atan2}(S_3/C_3) \qquad (6.30)$$

This completes the derivation of the inverse kinematic joint angles for the three primary joints of the PUMA 560.

6.3 Simulation of the Inverse Kinematic Equations

To show the advantage of using the inverse kinematics algorithm described in section 6.2 it was decided to simulate a joint trajectory in Cartesian coordinates and apply the inverse kinematic algorithm to solving the joint angles for this trajectory. The trajectory chosen was a circular trajectory similar to those used in arc welding applications. This type of trajectory cannot be generated by the existing Unimation controller without the use of a teach pendant to record points on circumference of the circular trajectory. The use of such a pendant means that the contoller must store a large number of circumference points to achieve any degree of accuracy.

The new control hardware offers an alternative to this by offering a means of generatng such trajectories on-line. The Carthesian circle chosen was defined by using the following polynomial equation to generate its circumference points:

$$\chi = 0.0512t^5 - 0.8274t^4 + 1.05t^3 \qquad (6.31)$$

where t is time. The position, approach and orientation vectors for this trajectory were defined as follows:

$$P_x(t) = (35 + 25C\chi)/4 \qquad (6.32)$$
$$P_y(t) = (35 + 25S\chi)/4$$
$$P_z(t) = 15/2$$

$$a_x(t) = a_y(t) = 0 \qquad (6.33)$$
$$a_z(t) = 1$$

$$o_x(t) = C\chi \qquad (6.34)$$
$$o_y(t) = S\chi$$
$$o_z(t) = 1$$

This trajectory was simulated and the inverse kinematic algorithm above was used to identify the joint angles necessary to realize the trajectory. The joint angles obtained for the three primary joints are shown in Figure 6.4. The angles obtained for this trajectory were then transformed using the forward kinematic equations (6.7), (6.8) and (6.9) to check the validity of the solutions. In all cases the positions found using the forward transformations were found to match those used to define the Carthesian circle. This means that the inverse kinematic solutions obtained by this method were correct. The advantage of implementing such a circle using this method on the new controller can be seen by considering the number of trajectory points that would be stored in the existing controller's

memory for use as control set points for the three joints. Since the time required to implement the circle was about 3s, see Figure 6.4, and the existing robot controller requires a joint angle setpoint approximately every 28 ms, this means that the existing robot controller would need over 300 memory locations to store setpoints for the joint angles necessary to achieve the same accuracy.

6.4 Summary

This chapter details the development of an algorithm for solving the inverse kinematics problem for the PUMA 560. The alogrithm was developed by first deriving the forward kinematic equations for the PUMA 560. These equations were then used to develop the inverse kinematic algorithm. The algorithm was then tested by simulation to show that it could produce the joint angles required for a real time path planning application which can not be implemented on the existing Unimation controller, without the use of a teach pendant.

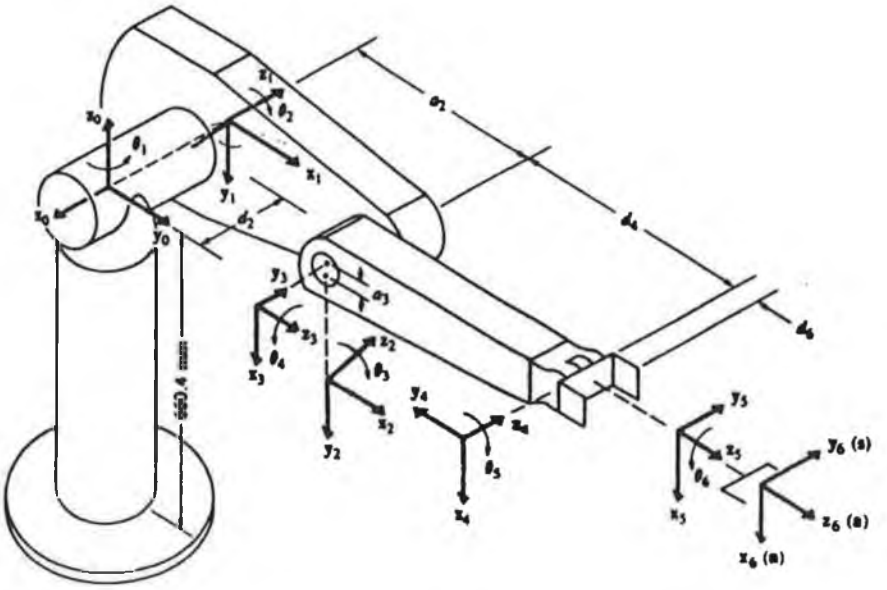
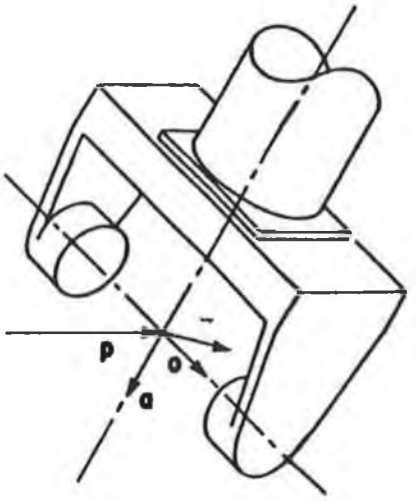


FIGURE 6.1 PUMA 560 LINK PARAMETERS

FIGURE 6.2 PUMA 560 HAND POSITION VECTORS



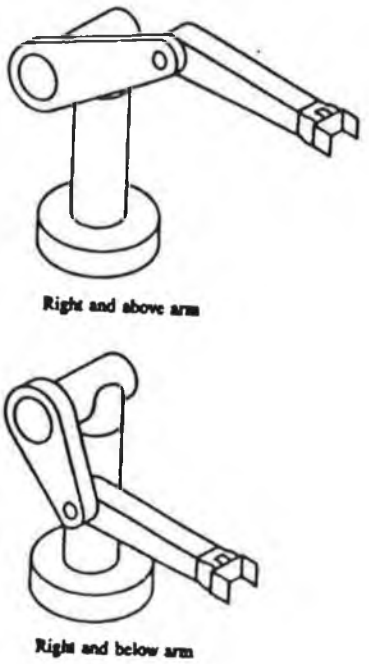
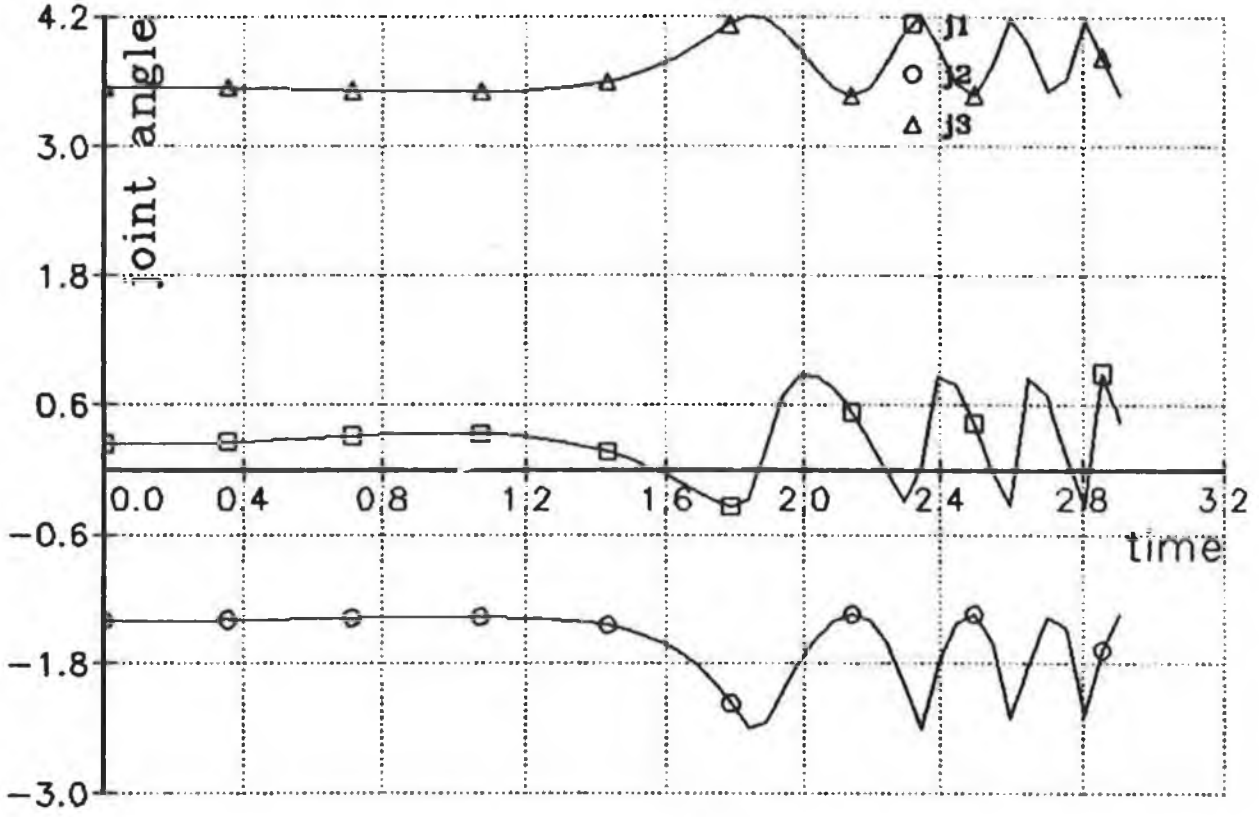


FIGURE 6.3 PUMA 560 RIGHT ARM CONFIGURATIONS.

FIGURE 6.4 JOINT ANGLES FOR CARTHESIAN CIRCLE



CHAPTER 7

Parameterization of the PUMA 560 Robot Model

The dynamic control of an industrial manipulator involves the determination of the inputs (torques or voltages) for the actuators which operate at the joints so that a set of desired values for the positions and velocities for the manipulator are achieved. Virtually all forms of dynamic control involve the use of a system model for the design of control algorithms. In the case of adaptive/self-tuning control, the model used is a discretized one which takes the form of an autoregressive time series model.

In this chapter such a time series model for the motion of the joints of PUMA 560 is developed from the continuous time model developed in Chapter 2. Various least squares-based methods for determining the parameters that best fit this model are implemented. These methods are then tested using, input/output data obtained from the actual robot, to see how well they identify the dynamics of the PUMA 560.

7.1 A General Time Series Model of PUMA 560

The equations of motion for the manipulator may be developed by the direct application of the classical Euler-Lagrange method of dynamic modelling as demonstrated in Chapter 2. For the three primary joint of PUMA 560, the model can be written in the following way:

$$V_i = \frac{R_i F_i}{k_i^t} + \frac{L_i \dot{F}_i}{k_i^t} + k_i^e N_i q_i \qquad (7.1)$$

where the terms in this equation are the same as those described in Chapter 2. This robot model can be discretized using a method such as Euler [8]. This method discretizes the joint position, velocities and accelerations as follows:

$$\dot{q}_i(t) = \frac{q_i(t) - q_i(t-h)}{h} \qquad (7.2)$$

$$\ddot{q}_i(t) = \frac{\ddot{q}_i(t) - \ddot{q}_i(t-1)}{h} \qquad (7.3)$$

$$\dddot{q}_i(t) = \frac{\ddot{q}_i(t) - \ddot{q}_i(t-1)}{h} \qquad (7.4)$$

where h is the discretization intervel. By backsubstituting equations (7.2), (7.3) and (7.4) into equation (7.1) it is possible to obtain a time series model for a robot withvoltage inputs and joint position outputs. The joint torques and their derivatives in equation (7.1) contain terms which are product and sums of all the joint position, velocities and accelerations. These terms can be represented as a nonlinear forcing term [47] in the time series model. The time series model of each joint has the form:

$$\begin{aligned} y(kT) = {} & A_0 + A_1 y[(k-1)T] \\ & + A_2 y[(k-2)T] \ \ldots\ldots + B_1 u[(k-1)T] \\ & + B_2 u[(k-2)T] \ldots\ldots\ldots f[kT] + m(kT). \end{aligned} \qquad (7.5)$$

where u(kT) is the model input, or joint voltage, and y(kT) is output or joint position at time kT. A_i and B_i are coefficients of the linear portion of the model, $f(.)$ is the joint nonlinear forcing term containing joint nonlinearities due to joint coupling and m(.) represents modelling errors.

An auotregressive model [47] can then be assumed from the model in (7.5) for each joint. It takes the form of the following difference equation.

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + h(k) + e(k) \qquad (7.6)$$

where k refers to the sampling interval and d is a positive integer specifing a positive time delay. The term h(k) represents a forcing term intended to include

nonlinear effects and the modelling errors of equation (7.5). The equation error e(k) represents a zero mean white noise. $A(q^{-1})$ and $B(q^{-1})$ are polynomials with q^{-1} being the backward shift operator.

7.2 Parameterization of the Autoregressive Model

To obtain comprehensive information about the parameters of the autoregressive model in equation (7.6) it was decided to use similar operational environments to those used in [46] and [47]. These test environments were used because according to [47] they provide the most insight into the dynamic characteristics of the PUMA 560 robot. The tests can be broken down into two blocks:

1) slow trajectory unloaded, and
2) fast trajectory unloadeed.

The fast trajectories were generated by driving the joints of the PUMA robot at their maximum VAL speed setting [48], while the slow trajectories were generated by driving the joints at 50% of maximum VAL speed. Numerous tests involving driving the three primary joints simultanously at the same speed from a rest position were performed. A selection of the robot trajectories, both slow and fast, used is shown in Figure 7.1 and 7.2. These trajectories were obtained by using the new control hardware to sample the position of the PUMA 560 joints. The new hardware was also used to sample the robot's input voltages.

The parameters of the time series model were estimated from the input/output pairs using four different on-line estimation methods. The following sections describe these methods in detail and evaluate their performance in identifying the parameters of the autoregressive robot model.

7.2.1 Method 1: Recursive Least Squares (RLS).

When optimal control theory is applied to the development of robot controllers, a common simplification of the model in equation (7.6) is to assume that the coupling terms due to the other joints can be neglected [47][49][50][7]. By assuming this and that the PUMA 560s system parameters are slowly time-varying [47], it is possible to apply the simplest form of RLS to the identification of this robot's parameters. The model identified by this algorithm takes the form of the block diagram in Figure 7.3. This model can be written as:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + e(k) \qquad (7.7)$$

If the parameter vector Θ and the regressor information vector Φ are defined as

$$\theta^T = (a_1,\ldots,a_n;b_1,\ldots,b_n) \qquad (7.8)$$

and

$$\Phi(k-1)^T = [y(k-1),\ldots,y(k-n-1);u(k-1),\ldots,u(k-n-1)] \qquad (7.9)$$

the model can then be written as:

$$y(k) = \theta^T.\Phi(k-1) + e(k). \qquad (7.10)$$

The parameter estimation problem is to find the estimates of the unknown parameters which minimize the cumulative loss function:

$$E(\theta) = \frac{1}{N+1} \sum_{k=0}^{N} [e(k)]^2 \qquad (7.11)$$

where N represents the number of measurements taken and e(k) is the prediction error at time k.

The principle underlying least square 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:

$$\theta(k) = \theta(k-1) + P(k)\Phi(k-1) \cdot [\, y(k) - \theta^T(k-1)\Phi(k-1)] \qquad (7.12)$$

$$P(k) = \frac{1}{\mu}\left[P(k-1) - \frac{P(k-1)\Phi(k-1)\Phi^T(k-1)P(k-1)}{\mu + \Phi^T(k-1)P(k-1)\Phi(k-1)} \right] \qquad (7.13)$$

where P(.) is the covarience matrix of the estimation errors and where μ is what is known as the forgetting factor. The P matrix is an $R^{2nx\,2n}$ matrix where 2n is the total number of parameters being estimated. This matrix is the 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 if the initial estimates are poor to ensure that its elements do not tend to zero too rapidly. If this occurs equation (7.12) reduces to

$$\theta(k) = \theta(k-1) \qquad (7.14)$$

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 [51] of 1000 on the diagonal elements of the P matrix should prevent this problem occuring. 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 problem, [51] suggests the use of a forgetting factor (μ). This factor can be used to implement an exponential weighting of past data to allow tracking of slow drift which might occur 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 are 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 varience monitoring, [51] 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:

$$\mu(t) = 0.95\mu(t-1) + 0.05 \qquad (7.15)$$

$$\text{with } \mu(0) = 0.95$$

To apply this method to the joints of the PUMA 560, a second order autoregressive model structure was used. This means the n value in equations (7.8) and (7.9) was given a value of 2. So the total number of parameters to be identified was four. This structure was used because an autoregressive model of order 3 was found to produce parameters which were of the order of 10^{-2} smaller than any of those identified using the the second order model.

The trajectory tests described in section 7.2 were carried on the three primary joints of the robot using the RLS algorithm. An example of the parameters obtained is shown in Figures 7.4 and 7.5. From these it can be seen that the parameters a_1 and a_2 converge rapidly to a constant value while the b_1 and b_2 parameters converged after a period of about 1 second. By comparing Figures 7.6 and 7.5 it can be seen that the parameters estimated varied in value and speed of convergence from one joint to the next. Figure 7.6 also shows that that the convergence of the parameters also depended on the joint speed. In fact, joint 1 was found to have the most rapid convergence at both speed values, while joint 3 was found to have the slowest.

A measure of the accuracy with which the RLS had modelled the actual robot dynamics was obtained by examining the cumulative loss functions, see equation (7.11), of all three over the slow and fast trajectories. These loss functions provide a good indicator of the model accuracy because they provide information about the estimation errors at every point on the joint trajectories. It can be seen from Figure 7.7 that the loss functions of each joint increases with joint speed. They can also be seen to increase from joint to joint with joint 1 having the smallest loss function values while and joint 3 having the largest. In all cases, this function seemed to increase rapidly with time. This indicates that the model estimated was becoming a less accurate representation of the true system model as the joints moved along their trajectories. This decrease in accuracy was not, however, due to the estimation method falling asleep because the elements of the diagonal elements of covarience matrix were found to have a value greater than 1 throughout the estimation run. In an attempt to reduce the prediction error initial estimates were used. These estimates were chosen by examining the estimation errors at each sampling instant and choosing the initial estimates to be the estimates where the estimation error was a minimum for that trajectory. These initial estimates were found to reduce the cumulative Loss function by a factor of approximately 3. This would seem to indicate that good initial estimates are required to ensure a more rapid convergence for RLS.

The parameter inaccuracies can be explained in terms of model of equation (7.1). Any movement of the robot's joints will involve changes in velocities and accelerations of these joints. These changes will cause the torque terms F and F of equation (7.1) to vary because of their dependence on velocities and accelerations. It is the inability of RLS to track these torque-dependent variations which appears to render RLS inadequate for the identification of the robot model parameters.

7.2.2 Method 2: Modifed Recursive Least Squares (MRLS)

This method of MRLS is based on the least squares model described in section 7.2.1. It differs , however, in that it uses the model [10] shown in Figure 7.8. This more comprehensive autoregressive model can be written as:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + h(k) + e(k) \qquad (7.16)$$

where h(k) is a forcing term intended to include the nonlinear effects of torque-dependent terms in the robot model. In this case, the parameter estimates and the regressors can be written in the following vector format:

$$\theta^T = (a_1,\ldots,a_n;b_1,\ldots,b_n,h_1) \qquad (7.17)$$

and,

$$\Phi_{(k-1)}{}^T = [\ y(k-1),\ldots,y(k-n);\ u(k),\ldots,u(k-n+1),1] \qquad (7.18)$$

From this we can see that the autoregressive model of equation (7.16) can be again written as:

$$y(k) = \theta^T.\Phi(k-1) + e(k). \qquad (7.19)$$

This is the format required to apply the least squares estimation algorithm shown in section 7.2.1. This results in the parameters being identified by equations (7.12) and (7.13). To ensure that this estimation method had the same ability as the RLS algorithm to track time varying parameters the same forgetting factor was used.

A second order model structure was used to apply this method to parameterization of the robot model. This meant that there were five parameters to be identified instead of the four used in the RLS algorithm. As with the RLS method, an increase in the number of parameter estimates beyond this value had little or no effect on the minimization of the loss function in equation (7.11).

The algorithm for MRLS was tested using the same input/output data gathered from the PUMA 560 robot for the RLS method. The estimates for the a_1, a_2, b_1 and b_2 parameters were found to be the same, in all cases, as those obtained for the RLS method. The h_1 parameter, for all the trajectories was found to take the form of a peak, see Figure 7.9 and 7.9. By comparing figures 7.1 and 7.9 it can be seen that these peaks reached their maximum amplitude when joint acceleration was a maximum. It can also be seen by comparing Figures 7.9, 7.10 and 7.11 that the amplitude increases with joint speed and varies form joint to joint.

The accuracy of the MRLS model was examined by observing the cumulative loss function used in the RLS method. Figure 7.12 the loss functions obtained from the joints moving along the same test trajectories used for the RLS method. From this information, it can be seen that the loss function was found to be about 3 times smaller than those measured over the same trajectories using RLS. The loss functions for all three joints tended to increase at a much slower rate than the loss functions in the RLS case, indicating that estimation error has reached a value very close to zero. Although the a_i and b_i parameters were the same in both the RLS and MRLS, an MRLS-based adaptive controller which incorporates the h_i parameter in the minimization its performance criterion should produce a more optimal controller.

The reason for the improvement in the parameters was the ability of the forcing term h_j to identify the residual due to the nonlinearities in the robot model which remained unidentified in the the RLS algorithm. In anattempt to reduce the effect of the initial, rather rapid, increase in the loss function, shown in Figure 7.12 it was decided to use initial non-zero estimates. The estimates used for each joint were chosen at a point in the joints trajectory where the estimation error was approximately zero. This was to ensure the best possible initial estimates for the identification. The introduction of the initial estimates had the effect of decreasing the initial rate of increase in the loss function and decreasing the convergence time of the parameters.

Although the MRLS model is a more accurate model of the robot than that produced by RLS, the estimates of MRLS fail to converge parameters to their correct value until the joints have stopped their initial acceleration and are moving at a constant velocity. It would appear from this that a more accurate model of the robot is required to increase the model parameter convergence.

7.2.3 Method 3: Extended Least Squares (ELS)

This method attempts to estimate a model for the noise present in any system, as well as the system model itself. It does this by formulating the autoregressive model [51] in the way shown in Figure 7.13. This model can be written time series form as follows:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})u(k) + C(q^{-1})e(k) \qquad (7.20)$$

where $C(q^{-1})$is the polynomial containing the parameters of the noise model.

In this case, the parameter estimates and the regressors can be written in the following vector format:

$$\theta^T = (a_1,\ldots,a_n;b_1,\ldots,b_n,c_1,\ldots,c_n) \qquad (7.21)$$

and,

$$\Phi(k-1)^T = [\ y(k-1),\ldots,y(k-n);\ u(k),\ldots,u(k-n),\ e(k),\ldots,e(k-n)] \qquad (7.22)$$

From this we can see that the autoregressive model can be written as:

$$y(k) = \theta^T.\Phi(k-1) + e(k). \qquad (7.23)$$

This means that the equations (7.12) and (7.13) can be used to update the parameter estimates of the model. Once again the same variable forgetting factor was used to track parameter variations due to time.

A second order model structure was used for both the noise and the system model itself. This meant a total of six parameters had to be estimated. The PUMA 560 input/output data used was the same as that used for the RLS and MRLS methods. Figures 7.14 and 7.15 show the parameters obtained from joint 2 traveling at a fast speed. From this we can see that all the model parameters were found to converge in less than 0.4 seconds of the joint starting to move. By comparing the estimates obtained in Figures 7.15 and 7.16 it can be seen that the convergence time and the parameters varied somewhat from one joint to the next, with joint 3 being the worst. The convergence rates for all three joints was found to be much faster than the rates observed for the previous two methods.

The accuracy of this model was again assessed by examining the loss functions. The loss functions for this method are shown in Figure 7.17. These were found to be as much as 3 times smaller than those found when the MRLS method was used. The loss functions of the three joints were found to increase at a much slower rate than those of the RLS or MRLS, indicating that the estimation error was decreasing more rapidly over the trajectories.

The improvement in parameter accuracy and convergence provided by the ELS method can be accredited to the way in which the algorithm models the residual. The convergence of the residual model seems to indicate that the non-linear torque terms can, in fact, be approximated quite accurately using a linear noise model. Because of the rapid convergence of the ELS parameters and the small estimation errors present, this method appears to provide good estimation errors even with no initial estimates provided. The provision of initial estimates was found to reduce the prediction errors for all the joints by a factor of about 20% in each case.

7.2.4 Method 4: Nonlinear Extended Least Squares (NELS)

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 Hammerstien nonlinear model [52] shown in Figure 7.18. This model can be written as follows:

$$y(k) = A(q^{-1})y(k) + B(q^{-1})x(k) + C(q^{-1})e(k) \qquad (7.24)$$

where $C(q^{-1})$is the polynomial containing the parameters of the noise model and is a nonlinear polynomial defined by:

$$x(k) = n_{01}u(k) + n_{02}u^2(k) + n_{03}u^3(k) + n_{11}u(k) + n_{12}u^2(k) + n_{13}u^3(k) \qquad (7.25)$$

In this case, the parameter estimates and the regressors can be written in the following vector format:

$$\theta^T = (n_{00},\ldots n_{02};n_{11},\ldots n_{12};a_1,\ldots,a_n; \\ b_1,\ldots,b_n,c_1,\ldots,c_n) \qquad (7.26)$$

and,

$$\Phi(k-1)^T = [\ u(k),u^2(k),u^3(k),u(k-1),u^2(k-1),u^3(k-1); \\ y(k-1),\ldots,y(k-n);\ u(k),\ldots,u(k-n); \\ e(k),\ldots,e(k-n)] \qquad (7.27)$$

From this we can see that the autoregressive model of can be again written as:

$$y(k) = \theta^T.\Phi(k-1) + e(k). \qquad (7.28)$$

A second order model structure for the system model, the noise model and the nonlinearity. This meant a total of 10 parameters had to be estimated. The robot input/output data used was the same as that used for testing ELS, MRLS and RLS methods. When the method was put through these test it was found that the parmeters n_{11} and n_{12} were found to be of an order approximately 10^{-6} smaller than the next to smallest parameters estimated for each joint. For this reason it was decided not to estimate these terms.

Figures 7.19 and 7.20 show the parameters obtained from joint 2 traveling at the fast speed. From this we can see that all the estimated parameters were found to be similar in their convergence to the ELS method. In fact, this was found to be the case in all the tests that were undertaken using this method. It can be seen from the two nonlinear parameters n_{01} and n_{02}, in Figure 7.21 that the robot

model have some considerable dependency on the square of the input voltage and this dependency is much less for the cube of the robot input.

The accuracy of this model was again assessed by examining the loss functions. The cumulative loss functions for this method are shown in Figure 7.21. The loss function of the three joints were found to increase at a slower rate, over the entire trajectories, than those of the ELS method. This indicates that the estimation error was decreasing more rapidly over the trajectories.

The improvement in parameter accuracy and convergence provided by the NELS method is due to the modeling of the input product terms in the system. The convergence of NELS model seems to indicate that that the nonlinear torque-dependent terms can, in fact, be modelled more accurately by including nonlinear functions of the robot inputs in the identification model. The provision of initial estimates was found to reduce the prediction errors for all the joints.

7.3 Summary

This chapter has been concerned with the parameterization of the PUMA 560 robot model developed in Chapter 2. It presents a time series model for the robot and shows how this time series model can be written in an autoregressive model. Various least squares methods were then applied to input/output data obtained from the PUMA 560 robot to identify the parameters of this autoregressive model. These methods included:

1) Recursive least squares,
2) Modified Recursive least squares,
3) Externed Least Squares and,
4) Nonlinear extended least squares.

The models identified by these methods were examined to test their accuracy and convergence. From these examinations various insights into the suitability of these methods for robot control were gained. It was seen that the RLS method was found to be unsuitable for identification of the model parameters of the robot. The method of MRLS was found to model the robot more accurately but it failed to show any substantial improvement in convergence time without good initial estimates. The ELS method was found to model the robot more accurately than then previous methods and showed rapid convergence even without good initial estimates. The method of NELS was found model the robot more accurately than the three other methods while showing similar convergence to the the ELS method. This would seem to indicate the suitability of using a nonlinear identification method for the development of adaptive controllers for robotic systems.

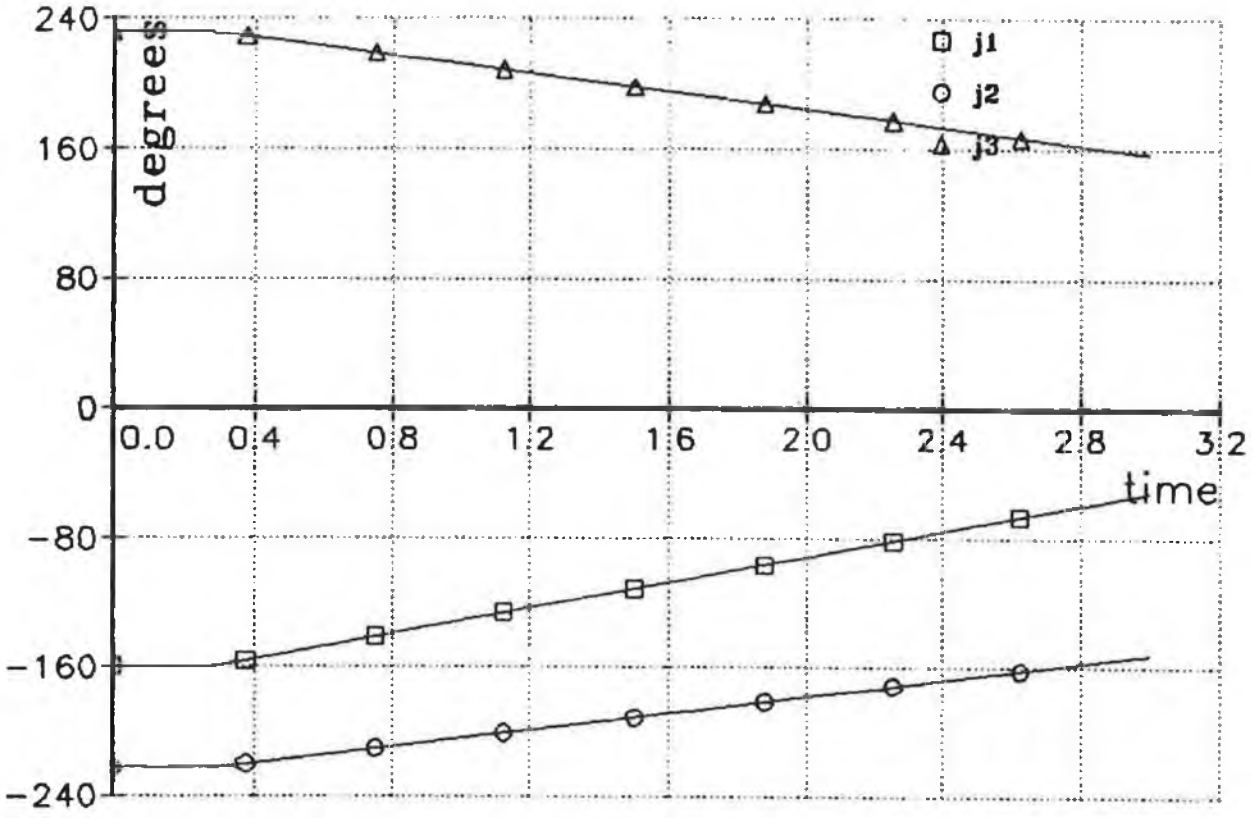
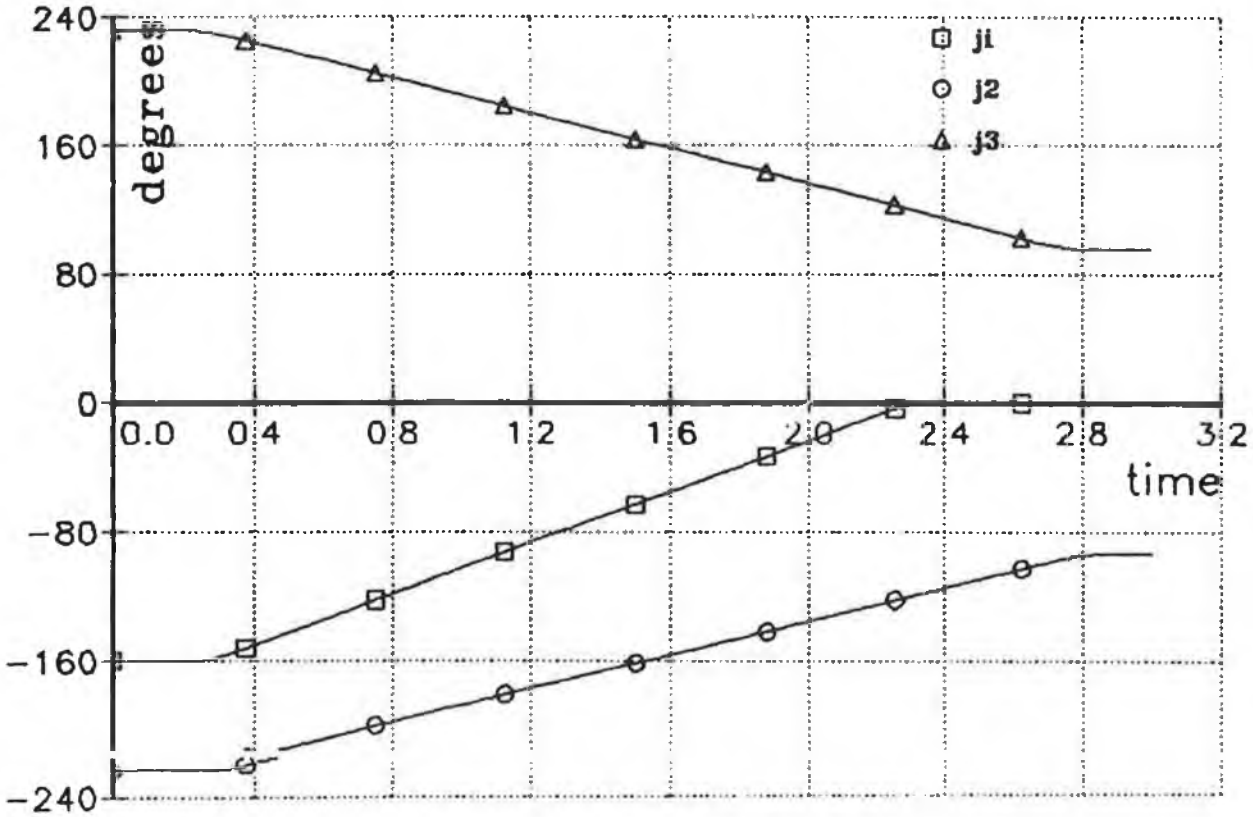


FIGURE 7.1 JOINT TRAJECTORIES SLOW V TIME (SECS)

FIGURE 7.2 JOINT TRAJECTORIES FAST V TIME (SECS)



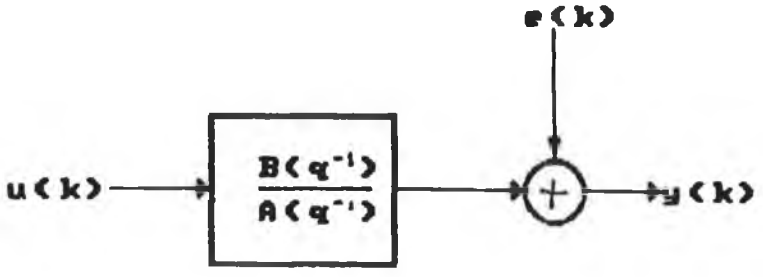
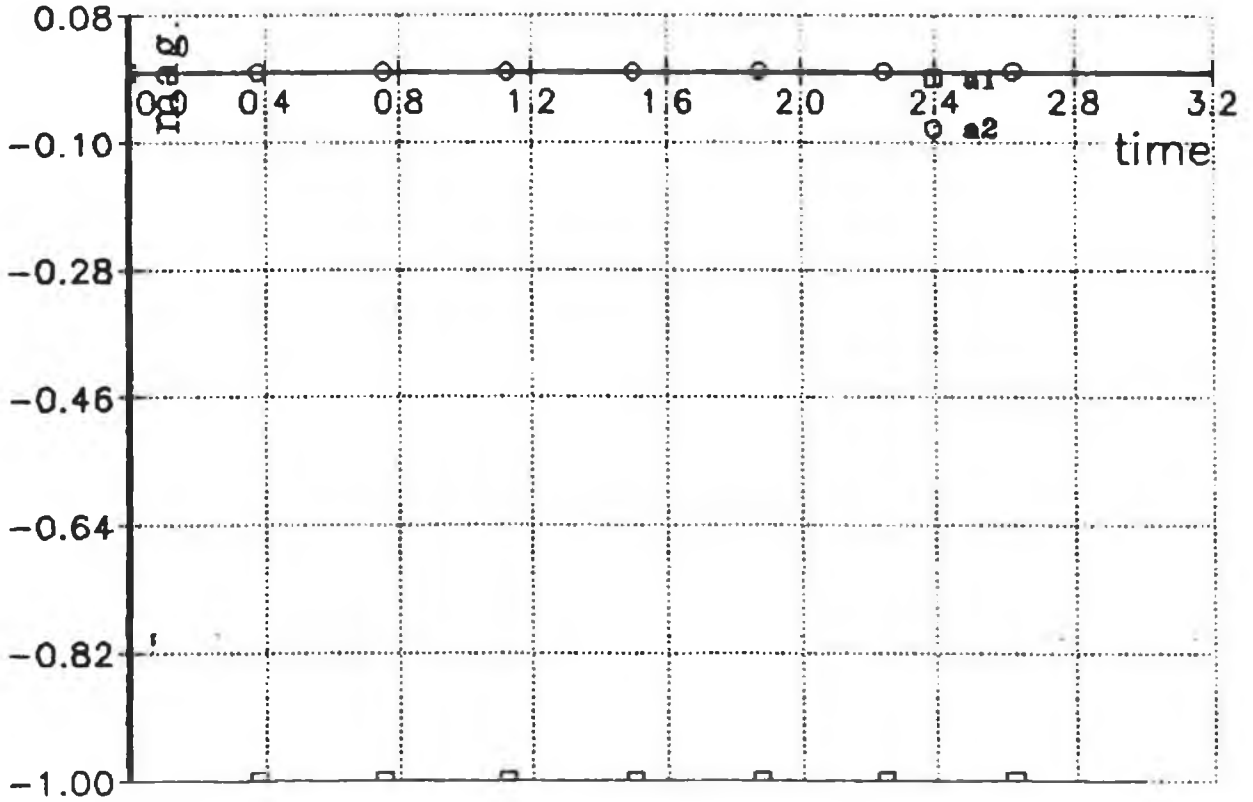


FIGURE 7.3 RLS BLOCK DIAGRAM

FIGURE 7.4 RLS J2 FAST a_1 AND a_2 PARAMETERS



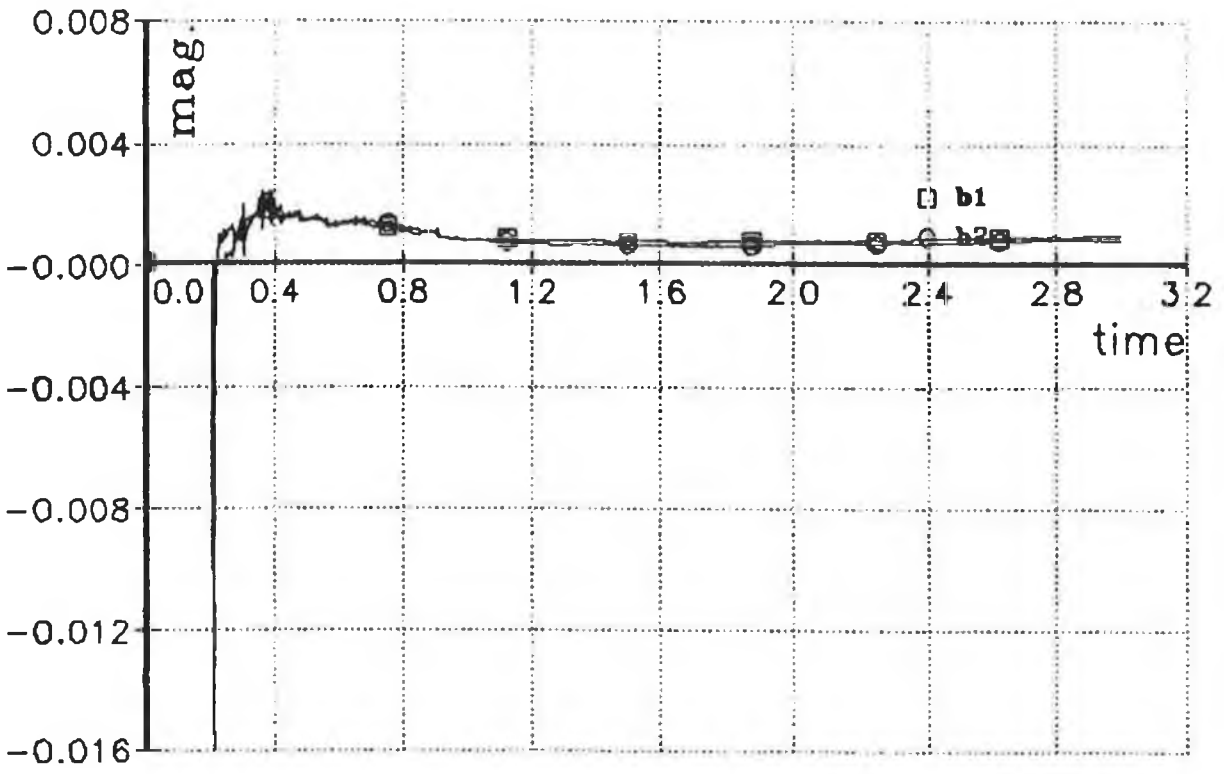
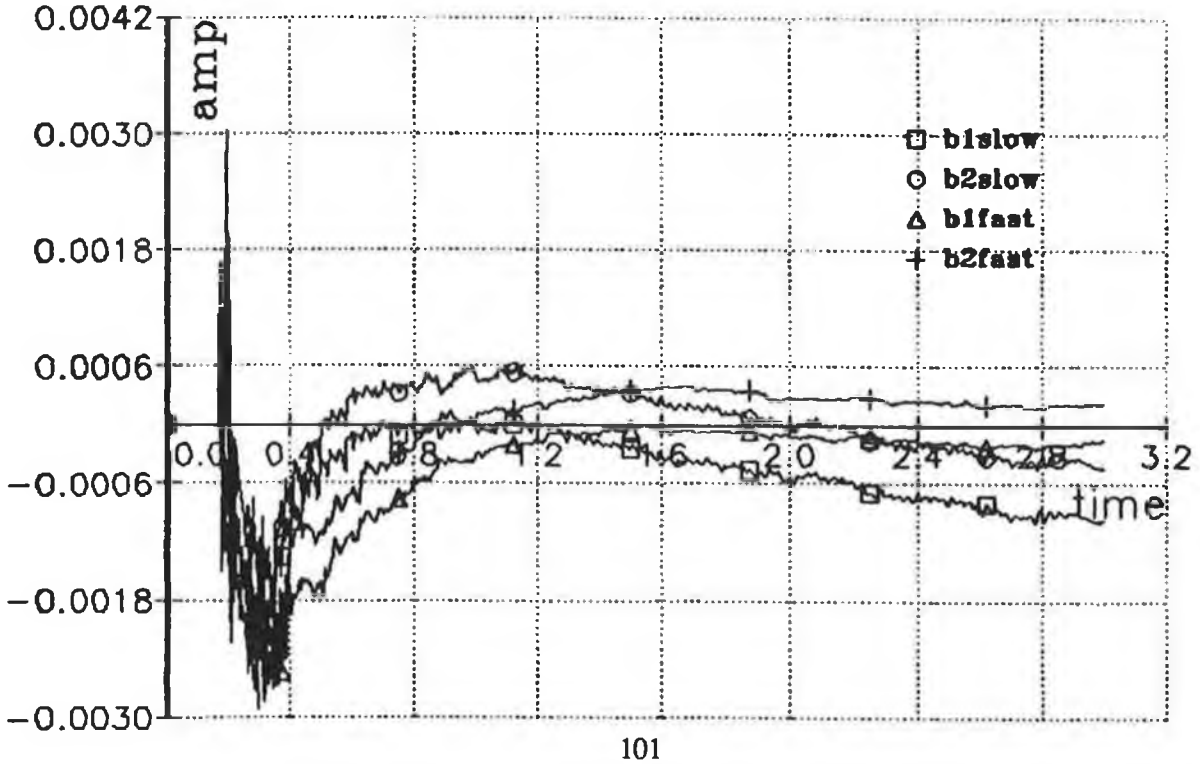


FIGURE 7.5 RLS J2 FAST b_1 AND b_2 PARAMETERS

FIGURE 7.6 RLS J3 FAST AND SLOW b_1 AND b_2 PARAMETERS



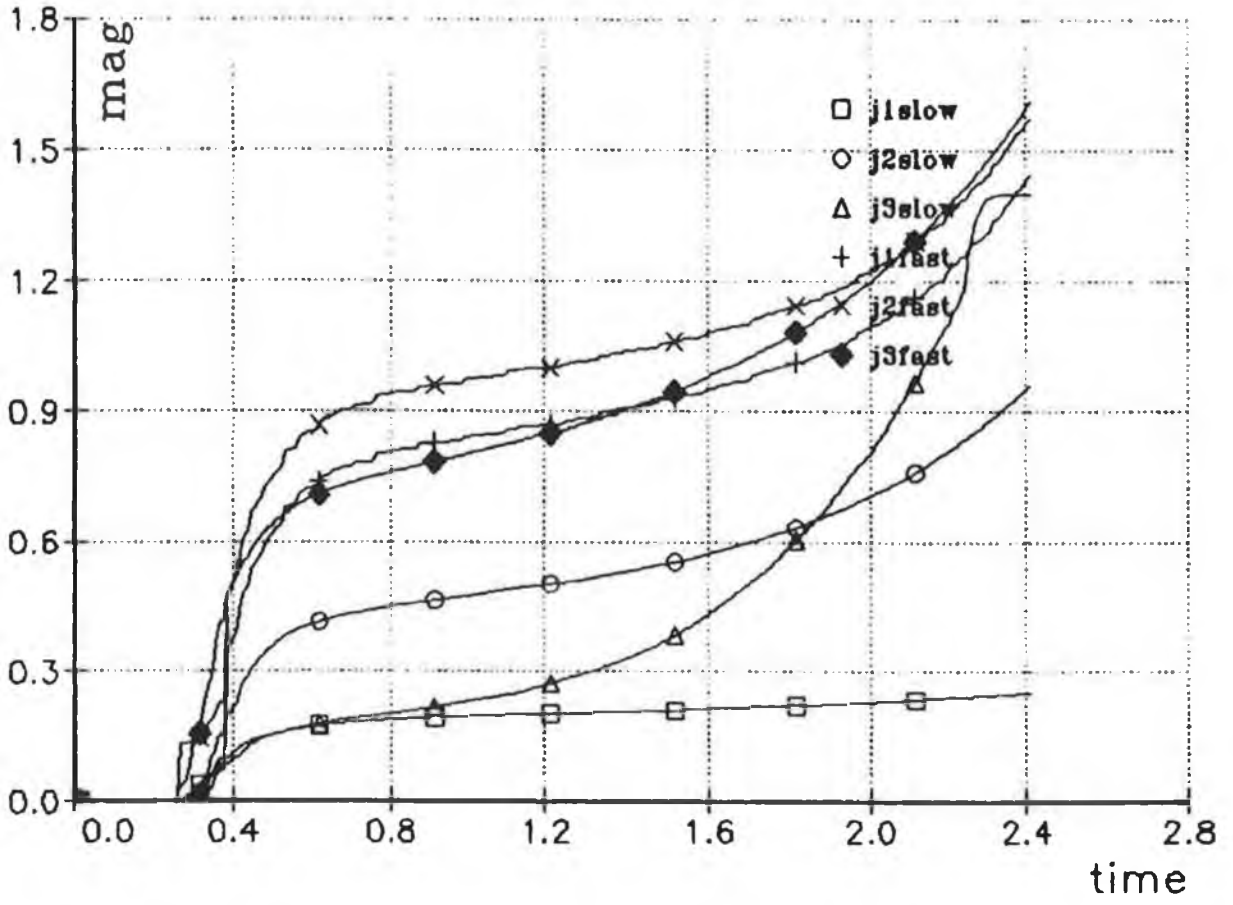
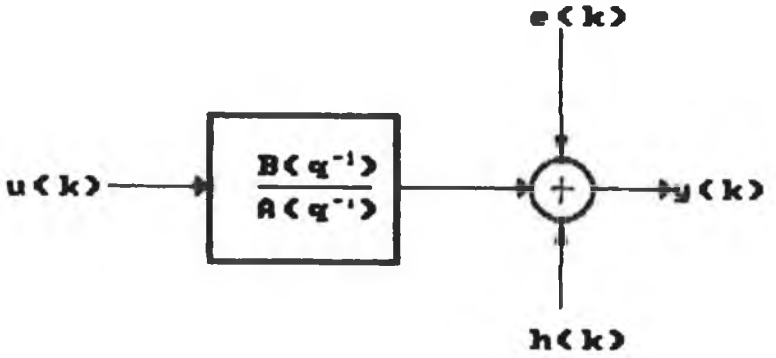


FIGURE 7.7 RLS LOSS FUNCTIONS SLOW AND FAST

FIGURE 7.8 MRLS BLOCK DIAGRAM



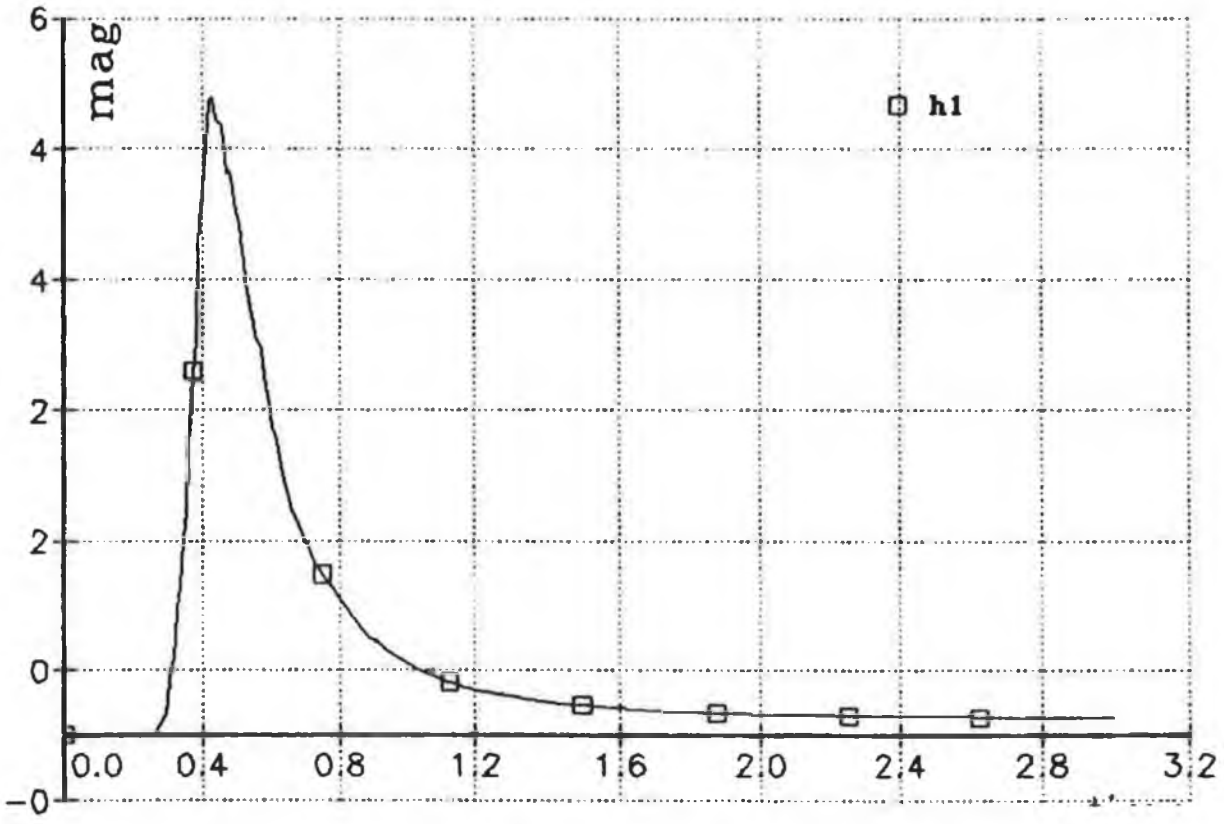
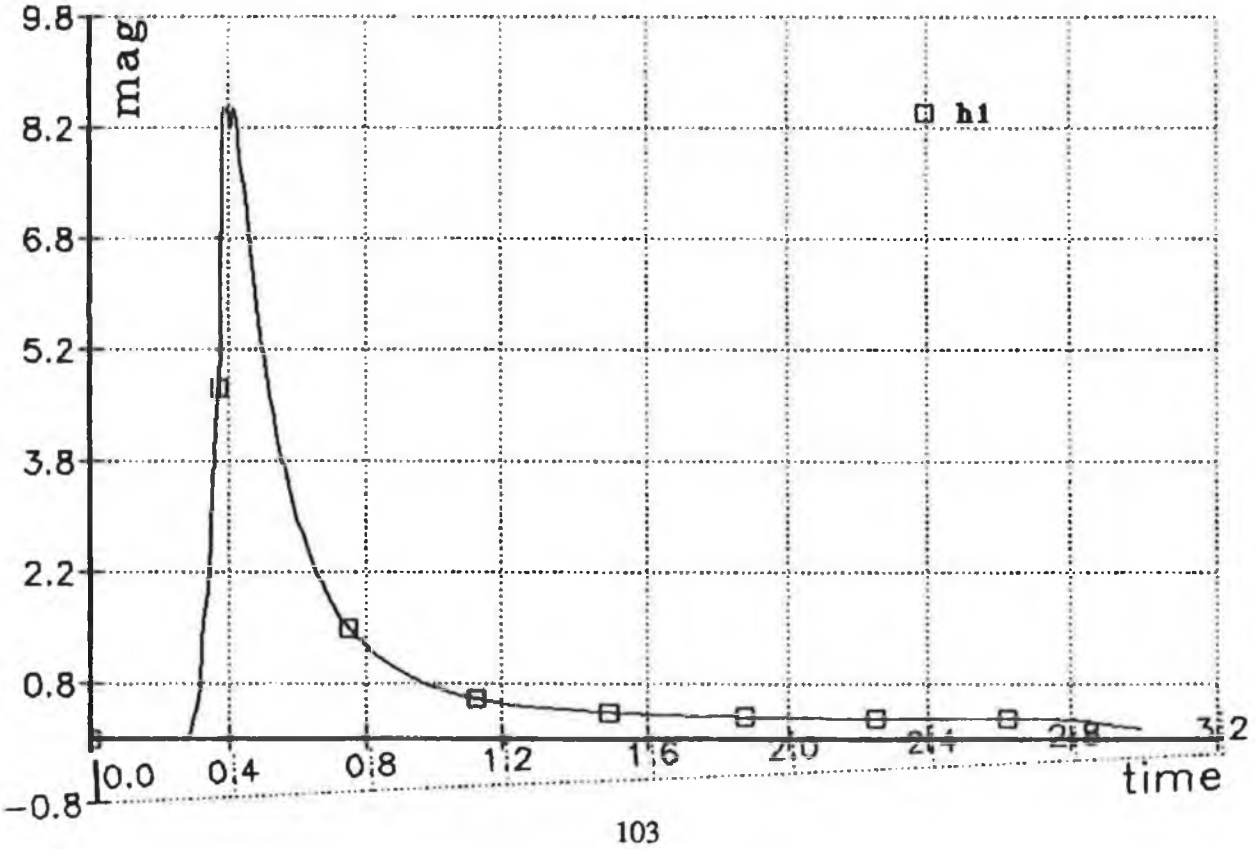


FIGURE 7.9 MRLS J2 FAST h_1 PARAMETER

FIGURE 7.10 MRLS J2 FAST h_1 PARAMETER



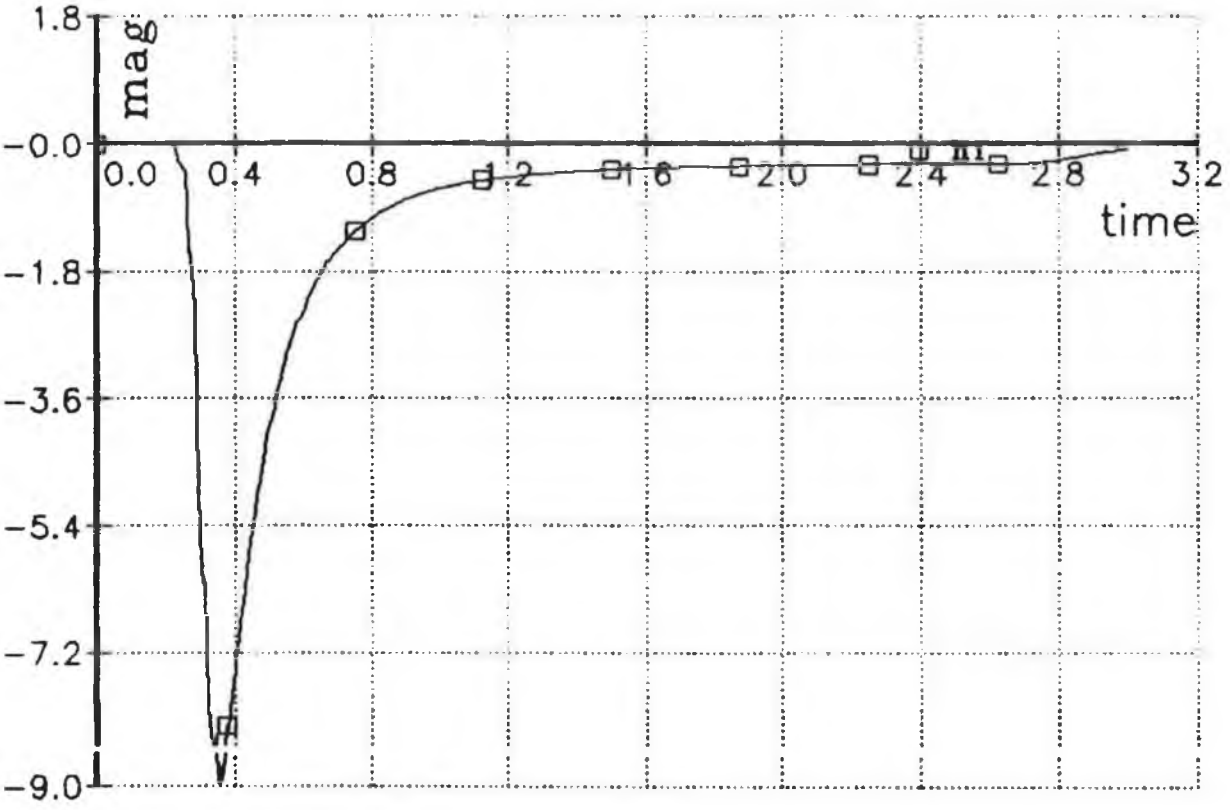
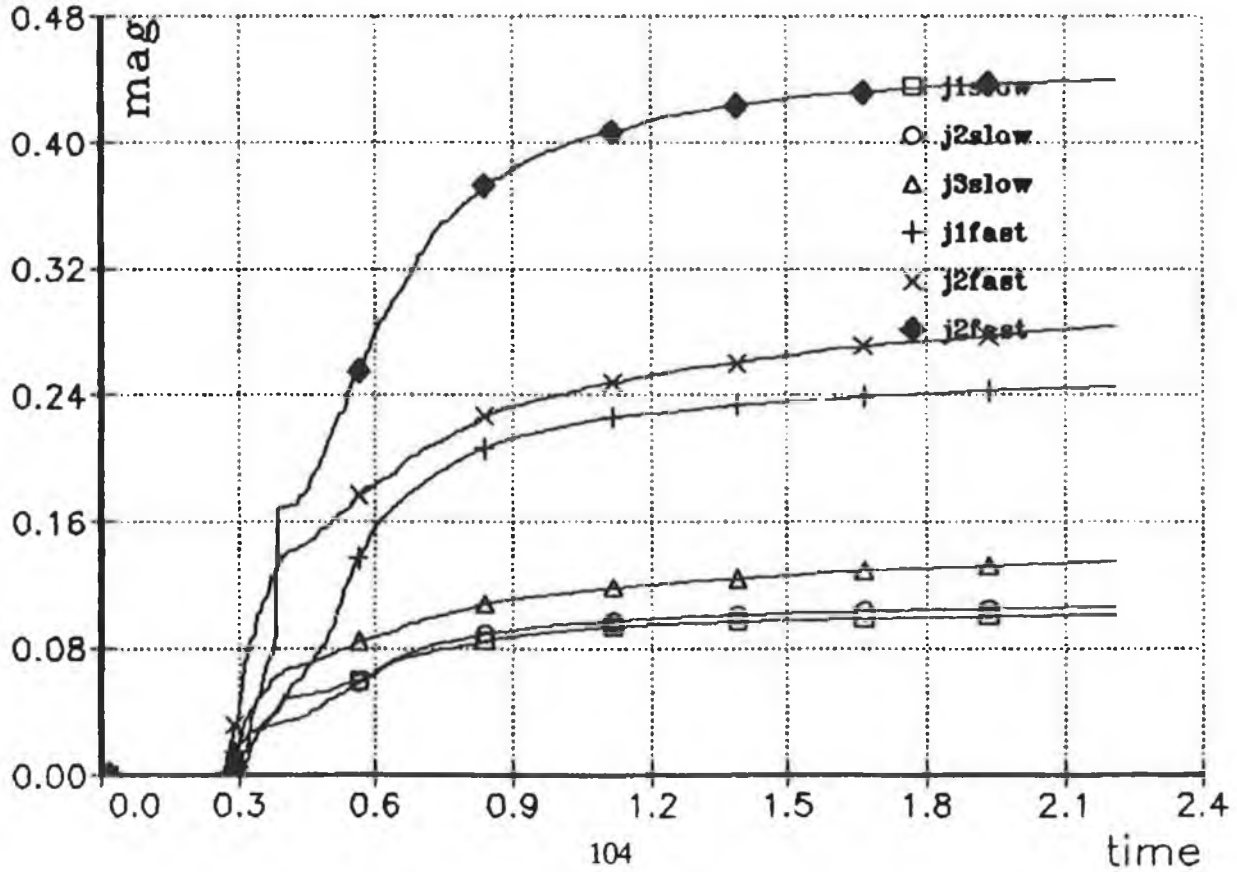


FIGURE 7.11 MRLS J3 FAST h_1 PARAMETER

FIGURE 7.12 MRLS LOSS FUNCTIONS SLOW AND FAST



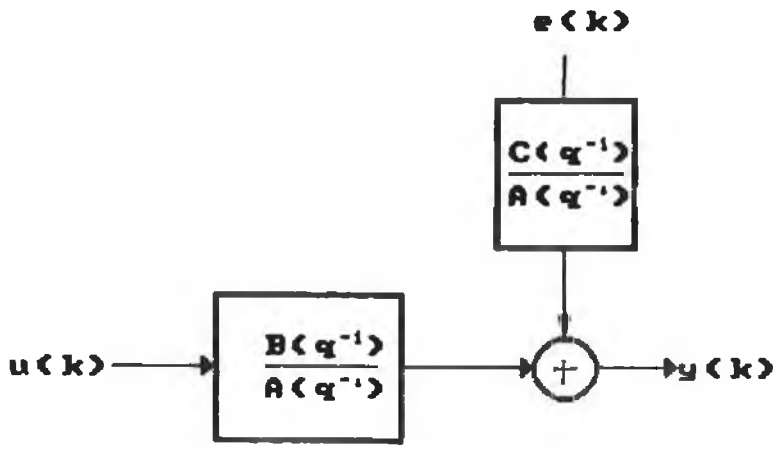
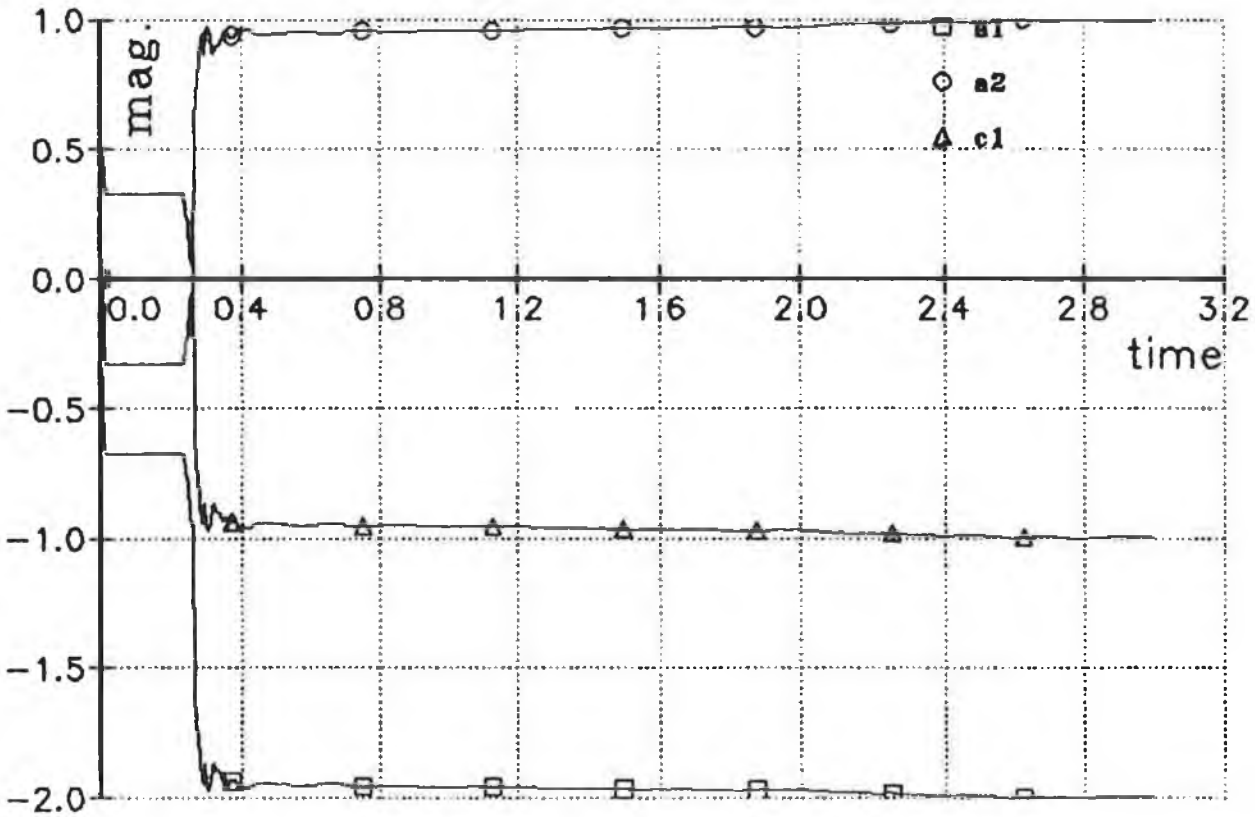


FIGURE 7.13 ELS BLOCK DIAGRAM

FIGURE 7.14 ELS J2 FAST a_1,a_2 & c_1 PARAMETERS



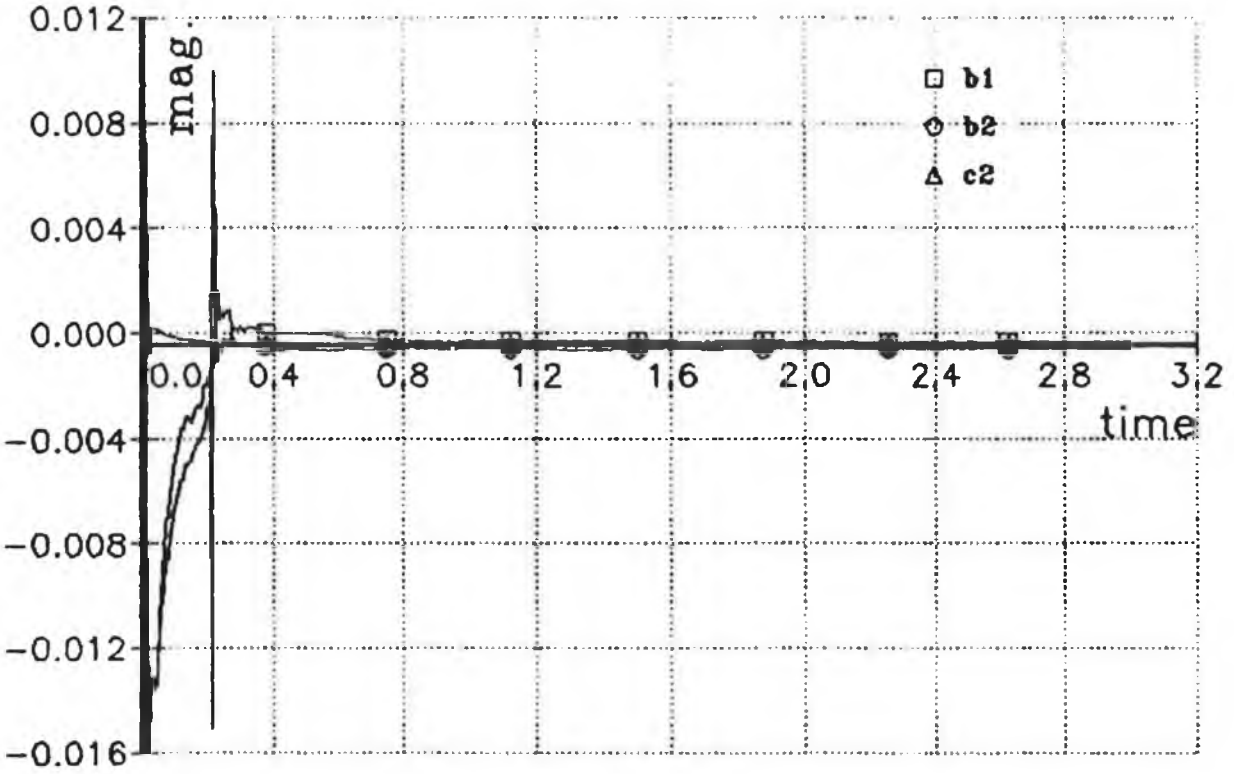
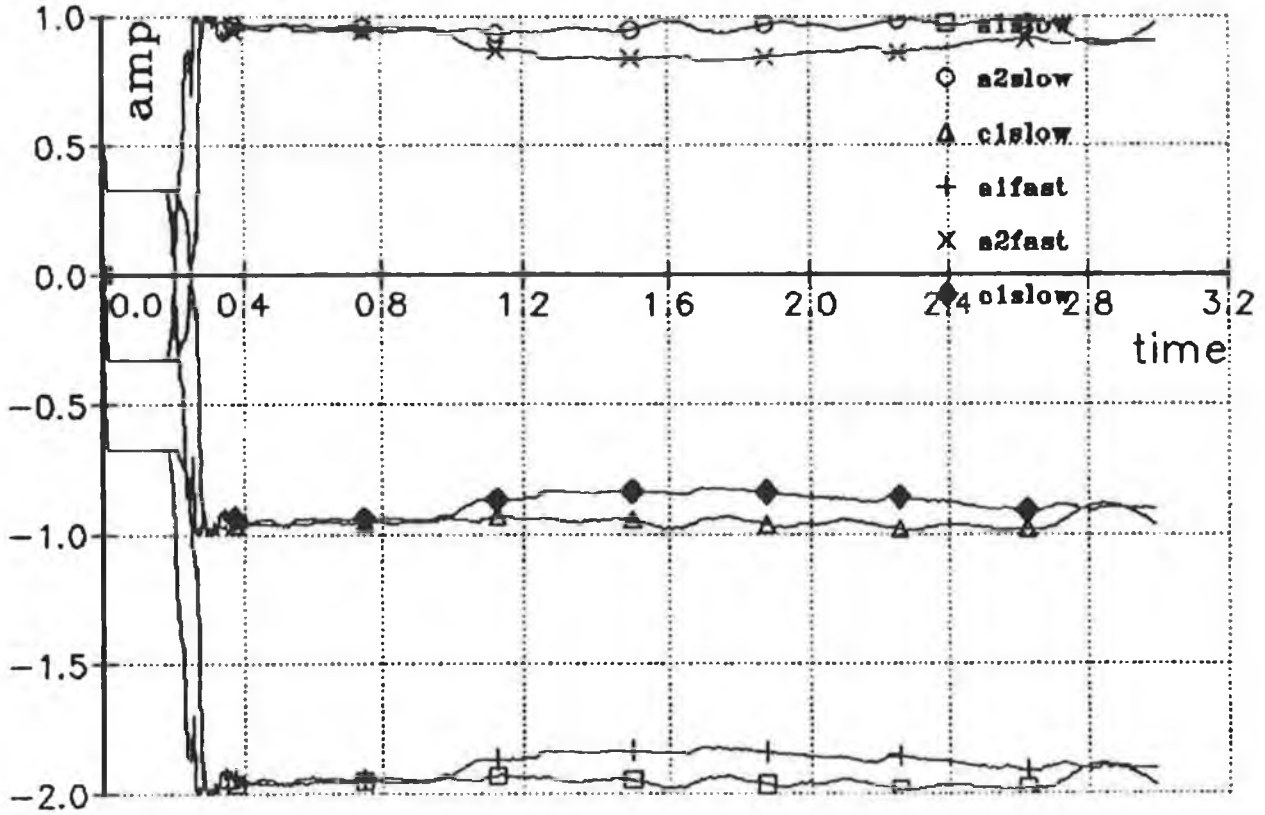


FIGURE 7.15 ELS J2 FAST b_1,b_2 & c_2 PARAMETERS

FIGURE 7.16 ELS J3 FAST AND SLOW a_1 a_2 & c_1 PARAMETERS



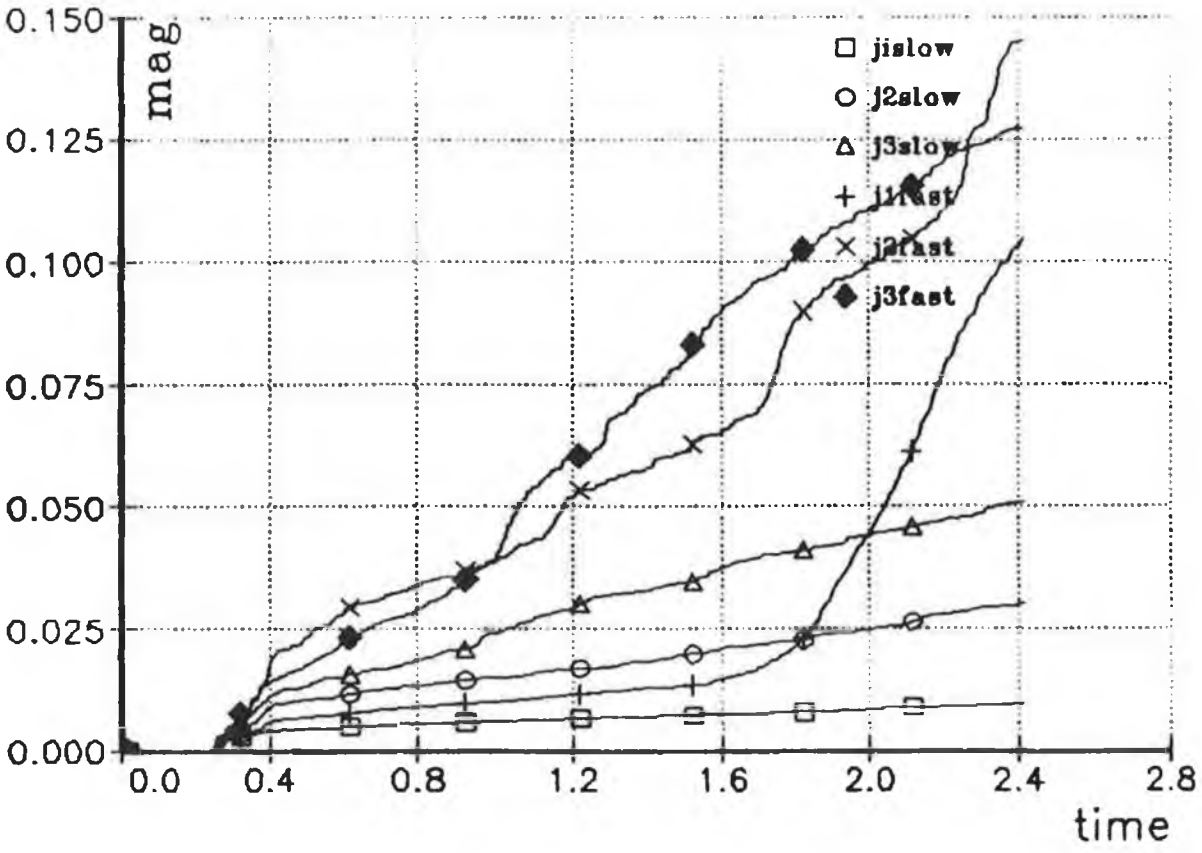
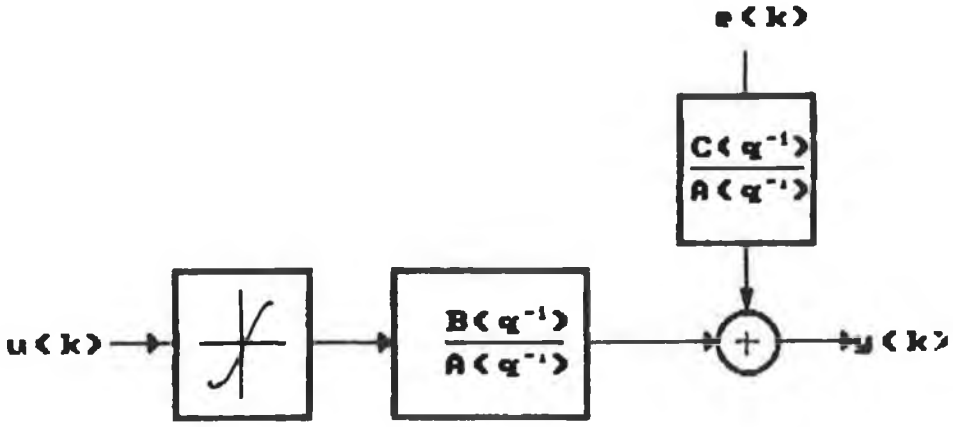


FIGURE 7.17 ELS LOSS FUNCTIONS SLOW AND FAST

FIGURE 7.18 NELS BLOCK DIAGRAM



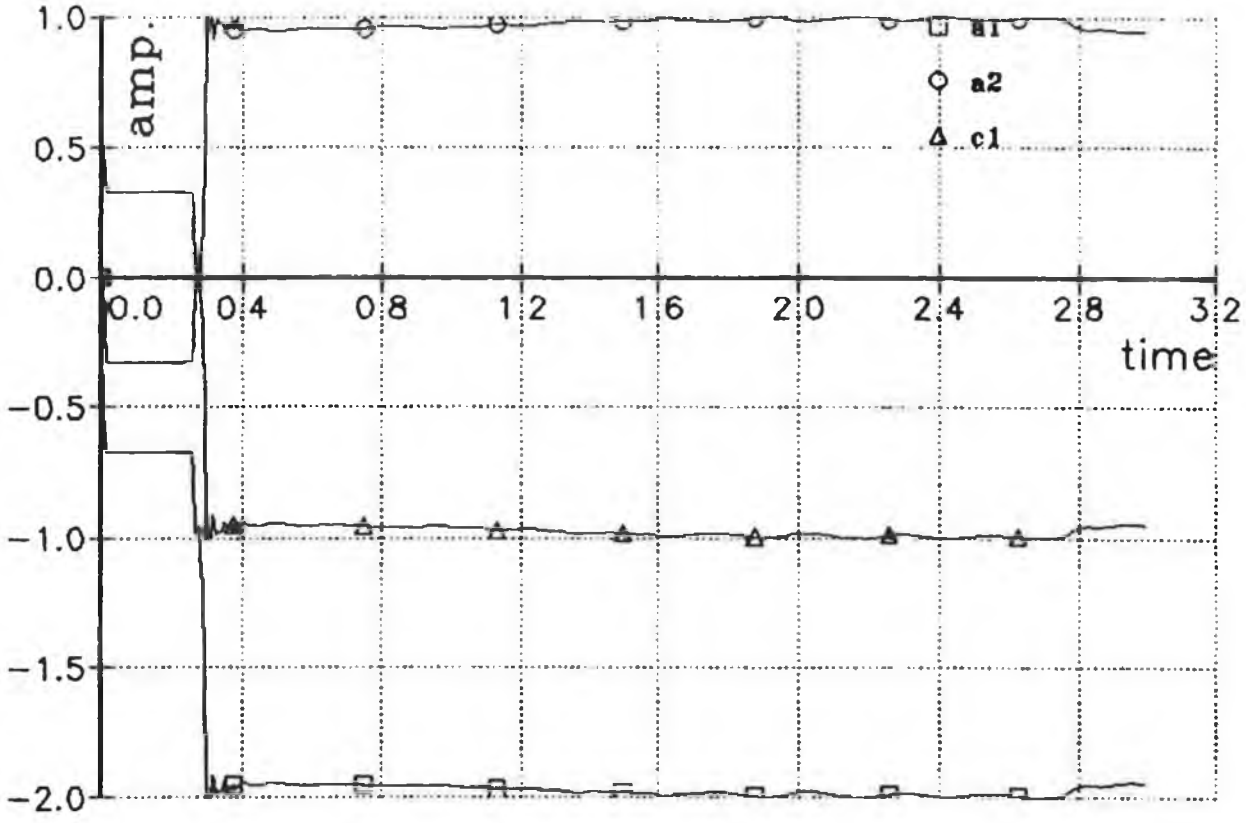
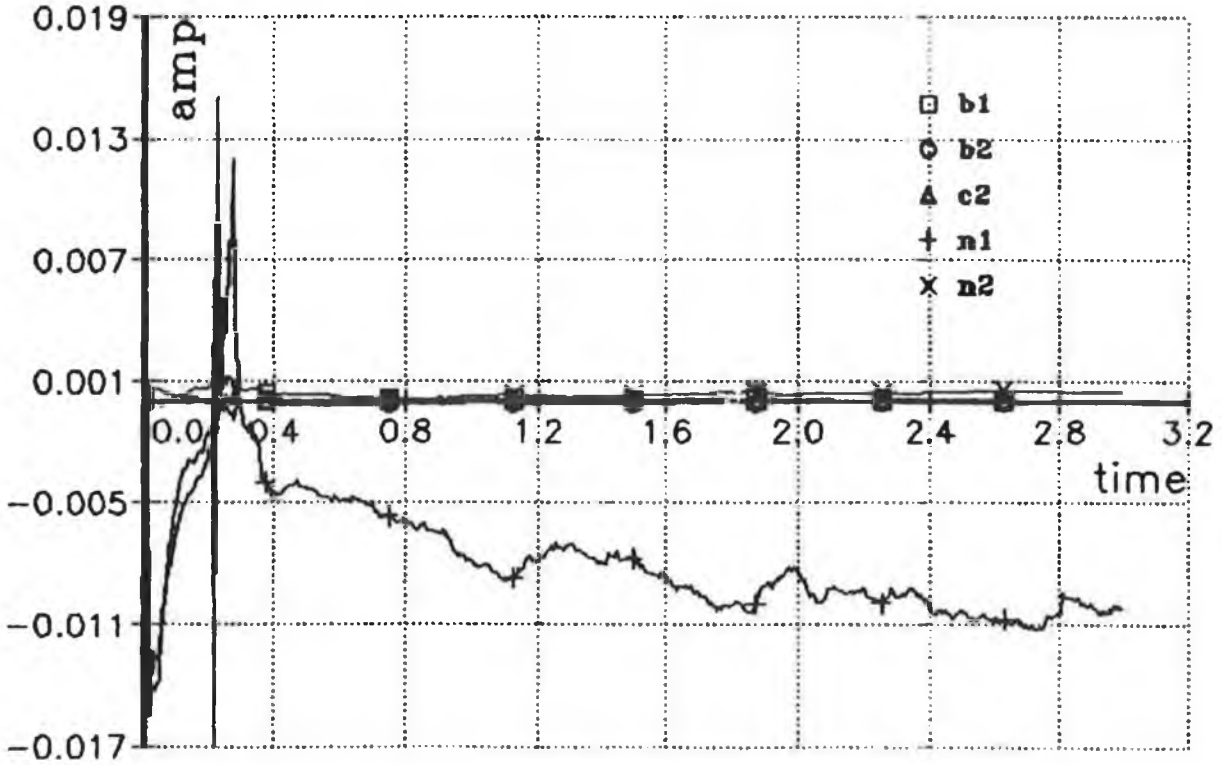


FIGURE 7.19 NELS J2 FAST a_1,a_2 & c_1 PARAMETERS

FIGURE 7.20 NELS J2 FAST b_1,b_2,c_1,n_1 & n_2 PARAMETERS



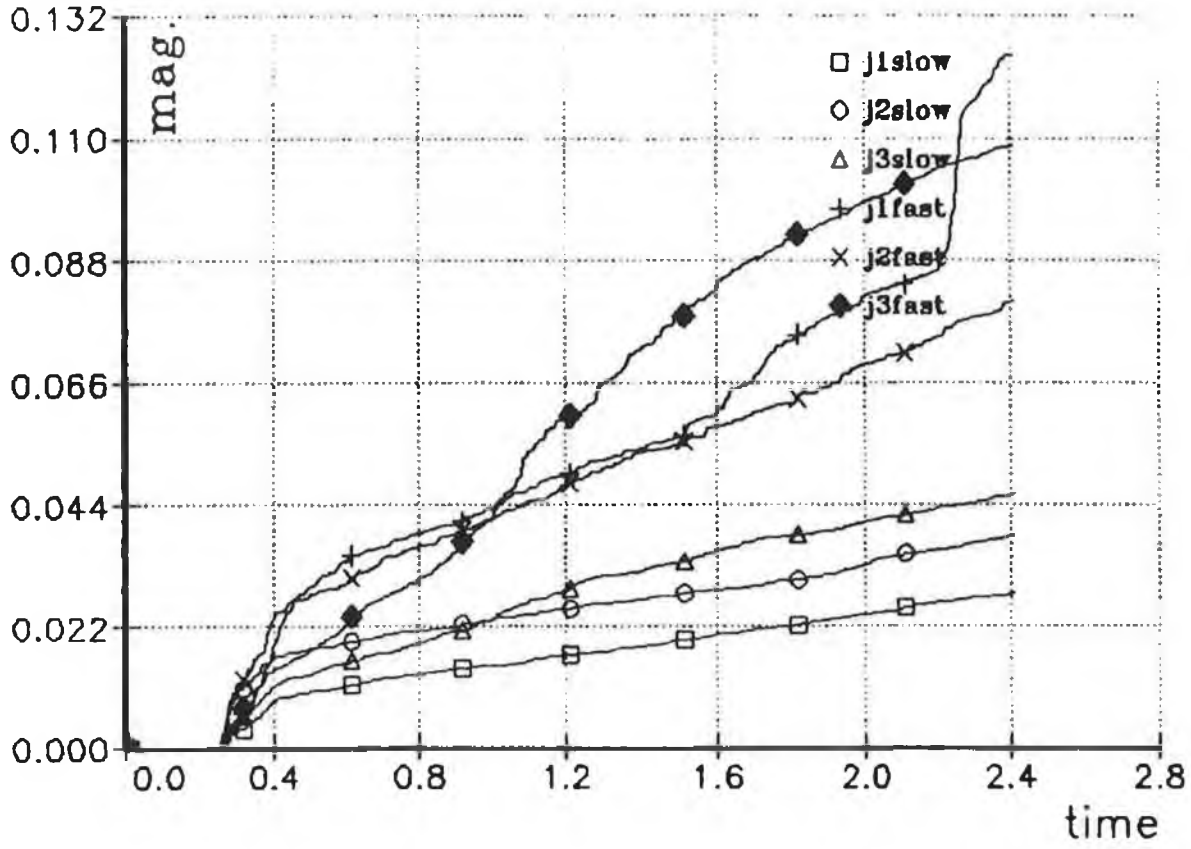


FIGURE 7.21 NELS LOSS FUNCTIONS SLOW AND FAST

CHAPTER 8

Conclusions

A complete dynamic model has been developed for the three primary joints of the PUMA 560 industrial manipulator. The modelling exercise involved the development of a third order model based on the Euler-Lagrange equations of motion for the PUMA 560. The Euler-Lagrange method models the manipulator as a set of second-order differential equations. The inputs to these equations are the joint torques while the equation outputs are the positions, velocities and accelerations of the robot joints. Since the only inputs to the PUMA 560 are the joint voltages and currents necessary to drive the joint DC motors, it was decided to include the dynamics of these motors to gain a more complete robot model. The inclusion of the actuator dynamics led to a third order model with voltage inputs and position, velocity and acceleration outputs.

Thismodel was simulated on a digital computer using a ninth-order state-space representation. The model was then validated by comparing its operation with that of the actual robot in a number of carefully chosen test conditions.

The complete design and implementation of a hierarchical control structure, using special purpose processors for the control of the three primary joints of the PUMA 560, has been presented in this thesis. The system involved consists of a general purpose personal computer operating as a supervisory or host machine with attached digital signal processor (DSP) boards capable of performing the numerically complex calculations involved in some real-time robot path planning and control algorithms.

The supervisory computer used in the new controller is 80386-based PC with a large memory capacity and an extremely fast clock speed (50ηs). This computer can be used to perform trajectory planning, coordinate transformations and task coordination between the personal computer and the DSP procesors used. The lower level of the new controller's hierachy consists of three NEC μPD77230 boards each with a large on-board memory. These boards are capable of implementing numerially complex joint control algorithm in real time. The advantages of this new system over the existing Unimation PUMA 560 include:

1) faster operation speeds at both levels,
2) a considerably larger memory capacity in both levels,
3) the ability to program in a high-level language,
4) full floating-point capabilites in both levels

In addition to these advantages, the new contol system provides a flexible interface to the PUMA 560. This flexiblity of this interface allows for the addition of vision and tactile sensors, if required, at some later research stage.

The thesis develops an inverse kinematic algorithm for the three primary joints of the PUMA 560 robot. The algorithm uses a geometric approach to provide a unique solution to the inverse kinematic problem. The algorithm developed is capable of realising complex robot paths which up to now were only realizable using the PUMA 560's teach pendant.

Finally, the thesis presents a time series model for the PUMA 560. A number of linear and nonlinear least squares identification methods were used to parameterize this model. The methods were implemented using the new hardware structure and tested under the headings of parameter convergence and identified model accuracy. Conclusions on the results obtained can be made as follows:

1) the method of RLS was found to be inaccurate for identifing the robot model,

2) the method of MRLS was found to produce more accurate model robot model than RLS but it was found to have an poor parameter convergence rates in the absense of a good initial estimated,

3) the method of ELS was found produce a more accurate model of the robot than the MRLS and RLS methods and showed rapid convergence of parameters even in the absene of good initial estimates,

4) the method of NELS was found to produce the most accurate model of the robot and showed covergence rates similate to those found using the ELS method.

This project has been successful in that it has managed to develop a comprehensive robot model for the PUMA 560 and also in the development of a new, more flexible, robot control hardware system to serve as an implementation tool for the future development of more computationally complex robot control algorithms. The results obtained from the parameterization of the robot model indicate that future development of adaptive controllers for robotic systems, based on nonlinear identification techniques, could lead to more accurate controllers.

REFERENCES

[1] Vukobratovic M. & Stokic D., "Control of Robotic Manipulation Robots Theory and Application", Springer-Verlag Berlin, 1982.

[2] Luh J.Y.S, "Conventional Controller Design for Industrial Robots - a Tutorial", IEEE Trans. on Systems, Man, and Cybernetics, Volume SMC-13, No. 3, May/June 1983.

[3] Hsia T.S., "Adaptive Control of Manipulators - A Review", IEEE Trans. on Robotics and Automation, June 1986.

[4] Koivo A.J, "Adaptive Linear Controller for Robotic Manipulators", IEEE Trans. on Automatic Control", Volume AC-28, No. 2, Feb. 1983.

[5] Craig J.J, "Adaptive Control of Mechanical Manipulators", The International Journal of Robot Research, Volume 6, No. 2, Summer 1987.

[6] Lieninger G.G, "Self-Tuning Adaptive Control of Manipulators", Advanced Software in Robotics, Elsevier Science Publishers b.v. (North-Holland), 1984.

[7] Dubowsky S. et al, "Application of Model Reference Adaptive Control to Robotic Manipulators", Journal of Dynamic Systems Measurement and Control, Volume 101, 1979.

[8] Lee C.S.G & Chung M., "An Adaptive Control Strategy for Mechanical Manipulators", IEEE Computer Science Press, Silver Spring, MD. 20910, USA.

[9] Bejczy A.K., "Robot Arm Dynamics and Their Control", Techical Memo 33-669, Jet propulsion Laboratory, Feb. 1984.

[10] 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 Muanufacturing, University of Michigan.

[11] Hollerbach J.M., "A Recursive Langrian Formulation for a Comparitive Study of Dynamic Formulation Complexity", IEEE Trans. on Systems, Man. and Cybernetics, Vol SMC-10, No. 11, 1980.

[12] 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.

[13] 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.

[14] Anderson G.P. "Modelling and Simulation of a PUMA 560 robot for Control System Appraisal", N.I.H.E. Dublin, M.Eng. Thesis 1988.

[15] Unimation Inc.,"Programming Manual User's Guide to VAL11 398T1", Version 1.1, August 1987.

[16] Fu K.S. et al, "Robotics Control, Sensing, Vision and Intelligence", McGraw-Hill, 1987.

[17] Melidy A. & Goldenberg A.A., "Operation of the PUMA 560 Without VAL", Robotics Proc. Robots 9, 1985

[18] Unimation (Europe) Ltd., "PUMA 560 Mk2 Robot System Technical Manual", Sept. 1985.

[19] Lee C.S.G et al, "Hierarchical Control Structure Using Special Purpose Processors for the Control of Robot Arms", Proc. of the IEEE Pattern Recognition and Image Processing Conference, pp 634-640, 1982.

[20] Paul R., "Robot Manipulators", MIT Press,1981.

[21] Unimation Inc., "Breaking away from VAL", Danbury Connecticut, 1982.

[22] Kananzides P., Wasti H and Wolovich W.A., "A Multiprocessor System for Real Time Robot Control: Design and Application", Proc. IEEE Int. Conf. Robotics and Automation, 1987.

[23] Penny D., "Control of the PUMA Robot Without VAL", Univ Toronto, RAL Tech. Rep., April 1985.

[24] Ringwood J.V., "Control Strategies for Robotic Manipulators", Proc. IMC-6 Conference on Advanced Manufacturing Technology, Dublin City University, Sept 1989

[25] Olivetti, "M380 Users Guide", 1989.

[26] Guruasavaraj K.H., "Implementation of a Self-Tuning Controller Using Digital Signal Processing Chips", IEEE Control Systems Magazine, June 1989.

[27] NEC, "Digital Signal Processors - Product Description", 1989.

[28] Kabuka M. & Escoto R., "Robot Arm Controller", IEEE Micro, Feb 1989.

[29] Loughborough Sound Images Ltd., "LSI 77230 PC Processor Board Hardware Manual", Version 2, Sept 1988.

[30] Khosla P.K. & Kanada T., "Experimental Evaluation of Feedforward Compensation and Computed-Torque Control Schemes", Proc. of the American Control Conf., Seatle, WA., June 1987.

[31] Bihn D.G. & Steve Hsia T.C., "Universal Six Joint Robot Controller", IEEE Control Systems Magazine, February 1988.

[32] Bihn D.G., "A Universal Six Joint Robot Controller", M.S. Thesis, Department of Electrical And Computer Engineering, University of California, Davis, California, 1986.

[33] Unimation, "Puma 500 Mk 2 Electrical and Mechanical Drawings", July 1985.

[34] LSI Ltd, "4 Channel Analog Interface Card Users Manual", Version 2.1, November 1988.

[35] Texas Instruments, "The TTL Data Book for Design Engineers", Volume 1988.

[36] Taylor P.M., "Robotic Control",Macmillan New Electronics Series, 1990.

[37] NEC, "Digital Signal Processor Development Tools", 1989.

[38] Astrom S.J., "LQG Self-Tuners", IFAC Adaptive Systems, San Francisco, 1983.

[39] Grimble M., "Implicit and Explicit LQG Self Tuning Controllers", Automatica, Vol 20, No. 5, 1984.

[40] Lelic M.A & Wellstand, "A generalized Pole Placement Self-Tuning Controller - An Application to Manipulator Control", Control Systems Centre Report No. 658, UMIST, Manchester, August 1986.

[41] Elgazzer E., "Efficient Kinematic Transforms for the PUMA 560 robot", IEEE Journal of Robotics and Automation, Vol RAI-1, No.3, Sept. 1985.

[42] Bazerghi A. & Goldenberg A.A., "An Exact Kinematic Model of the PUMA 600 Manipulator", IEEE Trans. on Systems Man, and Cybernetics, Vol SMC-14, May/June 1984.

[43] Crochetiere W.J., "Locating the Wrist of an Elbow-Type Manipulator", IEEE Trans. on Systems Man, and Cybernetics, Vol SMC-14, No.3, May/June 1984.

[44] Ersu E., "A Numerical Solution of the General Kinematic Problem", Institute of Control Engineering, Univ. of Darmstadt, FGR, 1984.

[45] Lee C.S.G. & Ziegler, "Geometric Approah in Solving Inverse Kinematics for the PUMA Robot", IEEE trans. on Aerospace and Electronic Systems, Volume AES-20, No. 6, Nov. 1984.

[46] Leahy M.B. & Saridis G.N, "Compensation of Unmodeled PUMA Manipulator Dynamics", Proc. Intl. Conf. on Robotics and Automation, Mar. 1987.

[47] Leahy M.B Jnr, "Performance Characterization of a PUMA 600 Robot", RAL Technical Report, No.56, RPI, Sept. 1985.

[48] Unimation, "User's Guide to VAL2", Version 1.1 August 1984.

[49] Young K.K.D., "Controller Design for a Manipulator Using Theory of Variable Structure Systems", IEEE Trans, Syst, Man. and Cybernetics, Vol. SMC-8, Feb. 1987.

[50] Book W.J. et al, "Feedback Control of a Two Beam, Two Joint System with Distributed Flexibility", Journal of Dynamic System Measurement and Control, Dec. 1975.

[51] Ljung L., & Soderstrom T., "Theory and Practice of Recursive Identification", MIT Press, 1984.

[52] Anbumani K. & Sarma I.G., "Self-Tuning Control of Nonlinear Systems Characterized by Hammerstein Models", IFAC Control Science and Technology (8th Triennial World Conference), Kyoto, Japan, 1981.