1992

Dynamic performance characteristics of robot manipulators

M. Y. Ibrahim
University of Wollongong

Dynamic Performance Characteristics of Robot Manipulators

A thesis submitted in fulfillment of the requirements for the award of the degree

Doctor of Philosophy

from

THE UNIVERSITY OF WOLLONGONG

by

M. Y. Ibrahim
B.E. Mech. Eng., Zagazig University (1974)
M. Tech. Syst. Eng., Brunel University (1981)

Department of Mechanical Engineering
and
Department of Electrical and Computer Engineering

1992

Dynamic Performance Characteristics of Robot Manipulators

by

M. Y. Ibrahim

Submitted to the Department of Mechanical Engineering
and
Department of Electrical and Computer Engineering
on August, 1992, in fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Robot manipulators have the inherent characteristics of being highly non-linear and strongly coupled. The recent increasing demand for industrial robots compounded with their high complexity has provided the robot's designers with a new challenging and time-consuming problem. The reason behind it is that a robot's designer cannot easily predict the effect of changing an operational condition or a geometrical parameter on the dynamic performance.

In this dissertation a comprehensive study was conducted to analyse the dynamic characteristics of robot manipulators under different operational conditions and various structural and geometrical configurations. Therefore, both kinematic and dynamic modelling were extensively used in both iterative and closed form. The analysis was applied on two types of widely-used industrial robots: a) a SCARA type robot and b) articulated robots.

The results of the study are aimed at giving further insight into robot dynamic performance in pursuit of an optimal robot's design. That is achievable since the results are also aimed at helping a robot's designer to foresee the effect of changing an operational condition (e.g. velocity or payload) or a geometrical parameter (e.g. a joint's twist angle or a link's length). In this research, a computer plot package was especially developed and used to display the extensive analysis information in a compact form. This helps a robot's designer develop an intuitive feel for his problem and reinforces this by a visual display.

Thesis Supervisor: Prof. C. D. Cook
Title: Head, Department of Electrical and Computer Engineering

Thesis Supervisor: A/Prof. A. K. Tieu
Title: Associate Professor, Department of Mechanical Engineering

Acknowledgements

I would like to express my deep appreciation to both my supervisors Prof. C. D. Cook and Dr. A. K. Tieu, my teachers, advisers and friends for their guidance and friendly advice throughout the years of research that led to this dissertation.

Also, I would like to thank my colleagues at Monash University College, Gippsland (MUCG). Special thanks are due to Prof. K. R. Spriggs and Dr. I. J. Spark for their empathy and support which enabled the continuation of this research at Gippsland. Also, the assistance of the MUCG's technical staff and those who helped in the practical and theoretical experimentation is greatly acknowledged.

Last, but not least, I would like to thank my wife Marion for her unthinkable patience and support during this research and my four lovely little children Timothy, Josephine, James and Elizabeth for all the evenings and nights I spent away from them throughout the years of this research.

Contents

List of Figures

List of Tables

List of Publications Resulting From the Thesis

- M. Y. Ibrahim, C. D. Cook, and A. K. Tieu. Dynamic behaviour of a SCARA robot with links subjected to different velocity trajectories. *International Journal of Robotics and Artificial Intelligence*, Robotica, 6(1):115–121, 1988.

- M. Yousef Ibrahim, C. D. Cook, and A. K. Tieu. Dynamic Characteristics of a SCARA robot subject to NC2 velocity trajectories with different payloads. *International Journal of Robotics & Computer-Integrated Manufacturing*, 6(3):259–264, 1989.

- M. Yousef Ibrahim, C. D. Cook, and A. K. Tieu. Computer Analysis on the Effect of a Robot Geometrical Parameters for Optimal Realtime Performance. Proc. *IEEE International Conference on Intelligent Control and Instrumentation*, pp 820–825, Singapore, February 1992.

- M. Yousef Ibrahim, C. D. Cook, and A. K. Tieu. Dynamic Performance Characteristics of an Articulated Robot Arm Under Time-varying Payload — Part 1: Simulation Study. Submitted for Publication in the *International Journal of Mechatronics*, Oxford, UK, 1992.

- M. Yousef Ibrahim, C. D. Cook, and A. K. Tieu. Dynamic Performance Characteristics of an Articulated Robot Arm Under Time-varying Payload

— Part 2: Experimental Approach. Submitted for Publication in the *International Journal of Mechatronics*, Oxford, UK, 1992.

Chapter 1

Introduction

1.1 Background and Dissertation Goals

Due to the ever-increasing demand for high productivity, quality and economical production methods, the use of industrial robots has grown tremendously in the last decade. That vast growth has led to a boom in robotics research. With emphasis on high-speed and precision operation of robots, dynamic behaviour has become one of the most significant design factors.

The industrial robot emerged in the late 1960's. Therefore it could be considered in its infancy (compared with other branches of science). The Robotics industry currently enjoys considerable importance due to the following facts:

- the industry's potential for growth is very high
- it has a considerable influence on flexible manufacturing in general and thus has a large indirect effect on the economical aspects of production.

Industrial robots are key components of flexible manufacturing technologies because their programmability allows them to be quickly adapted to changes in the

production process.

The following are some of the major benefits that can be gained from using robots in manufacturing:

1. increased productivity,
2. improved product quality (through quality consistency),
3. increased manufacturing flexibility,
4. reduced labour cost and
5. performing dangerous and undesirable jobs.

The above-mentioned benefits have led to a high expectation of robots usage. However the industrial applications of robotics in the past two decades were mainly in the following areas :

1. spot welding
2. arc welding
3. spray painting
4. material handling
5. machine loading and unloading
6. assembly
7. inspection

Due to the constantly-increasing demand for industrial implementation of robotics, the use of robotic manipulators grew tremendously in the last decade leading to a boom in robotics research. One emphasis of this research is on high-speed and precision operations.

1.1.1 Current challenges

Robotic research problems include :

Speed

Due to the non-linear effect of the velocity related (centripetal and Coriolis) forces, and due to the strong inter-joints coupling, high-speed operating robots offer a serious challenge to the servo and control systems. These must be fast enough to accommodate the rapid changes in the systems' parameters.

Accuracy

Most robot manufacturers omit the robot's accuracy from the specification manual. Most industrial robots are known to have accuracy no better than +/- 1.25mm, whereas if robots are to be programmed off line, much higher accuracy is required. Accuracy is difficult to achieve, especially at high-speed due to the arm's inertia, resulting in links' deflections and vibrations.

Payload

Current industrial robots tend to be large, massive and lack versatility. On average, industrial robots can manipulate payloads that are only about 10% of their own weight. There is scope for improvement in this area for an improved design to achieve better mass distribution.

Dexterity

Dexterity is the degree of robot's flexibility to reach the working space in any direction. The need for better dexterity is often prompted by fine assembly

or spray painting applications.

Better dexterity can be achieved by increasing the number of Degrees of Freedom (d.o.f.). However, redundant d.o.f.'s make the control of such manipulators a difficult task.

Control

The control problem remains one of the most challenging problems in robotics. Controllers need to be much more sophisticated in their ability to interact between manipulators and sensors in real time. Also there are still many other problems areas such as programming languages, sensors, etc. which need improvement before the benefits can be fully realised in manufacturing industry. Also, considerable work has been reported regarding the application of adaptive control techniques to the robotic control problem [1-12].

Drawback of adaptive control algorithms

The drawback of many of the adaptive control algorithms that have been proposed in the robotics literature is that they treat the joints as decoupled systems and assume that the robot's parameters are essentially constant.

Adaptive control is mainly applied in practice to linear systems with constant or slowly time-varying unknown parameters. There is a definite need for the continued analysis and design of adaptive control algorithms for non-linear (or linear time-varying) robot models. Since the performance of a control algorithm is limited by the accuracy of the system's model, it is worthwhile to put effort into developing a better physical understanding of robot dynamics.

Sensitivity analyses must be applied to assess the impact of model approximations on robot control. They are, also, essential for the development of adaptive control algorithms that can handle effectively the continuously increasing

demands of robot control.

Vukobratović and Stokić [13], in their survey of adaptive robot control algorithms, state that:

> "...few efforts have been made to analyse the necessity of adaptive control for manipulation systems. It seems that most of the parameter variations in practice could be compensated by sufficiently robust (classical) control."

Also, a general drawback to adaptive controllers is that the computational requirements for real-time parameter identification, and the sensitivities to numerical precision and observation noise, tend to grow undesirably as the number of system state variables increases [14].

The continuous increasing demands for enhanced productivity and improved precision have imposed special requirements on the control of industrial robots and caused a shift of emphasis towards the dynamic behaviour of robotic manipulators. This shift has led to the development of non-linear feedback control algorithms for robots [15]. The method takes the following approach: (i) Design of a global non-linear feedback algorithm that transforms the highly coupled and non-linear robot dynamics into equivalent, decoupled linear systems (one for each degree-of-freedom); and (ii) Synthesise local (joint) linear control algorithms in the framework of classical engineering [16]. The applied control signal is then the sum of the nominal control signal to cancel the non-linear dynamics and the augmented linear feedback control signal to specify the closed loop response. Non-linear feedback control algorithms are thus founded upon the hypothesis that an exact model of the robot dynamics can be implemented in the controller.

The first nonlinear feedback approach to robot control, which utilised the closed-form dynamic robot model, was the computed-torque control algorithm [17-24]. In this approach, the actuating forces/torques are computed as functions of

the desired trajectory (and its velocity and acceleration) in joint coordinates. The algorithm required the on-line evaluation of the robot dynamics which led to implementation problems. The development of customised algorithms and dedicated hardware [25-28], which compute the inverse robot dynamics, has revitalised interest in a computed-torque control. The non-linear torque controller of Sahba and Mayne [29] utilises essentially the same structure as the one proposed by Raibert and Horn [22], although in the Sahba and Mayne controller, on-line calculations rather than look-up tables are used to obtain the inverse robot dynamics.

Resolved-acceleration control [25] extends the computed-torque concept to robot control in end-effector coordinates. A related approach is the non-linear direct design method [30, 31], which is based on non-linear feedback decoupling [32] and arbitrary pole placement [16]. Time optimal torque control [33] is a variation of the direct-design method. The direct design method has also been extended to robot control in Cartesian coordinates [34].

Although the non-linear feedback control concept is appealing from the theoretical point-of-view there are practical problems in its industrial applications. This stems from the fact that the dynamic control model is only *relatively* accurate. Furthermore, modelling errors result in inexact cancellation of the nonlinearities, which degrade performance and can lead to instability [35, 29]. The high non-linearity and strong coupling of the robot's arm make it difficult to design a robot that is free from the above- mentioned problems.

1.2 Problem Description

The design of a general robotic arm is an expensive, time-consuming and challenging task. That is due to the large number of system design parameters. The magnitude of this task can be illustrated by noting that a general six degrees

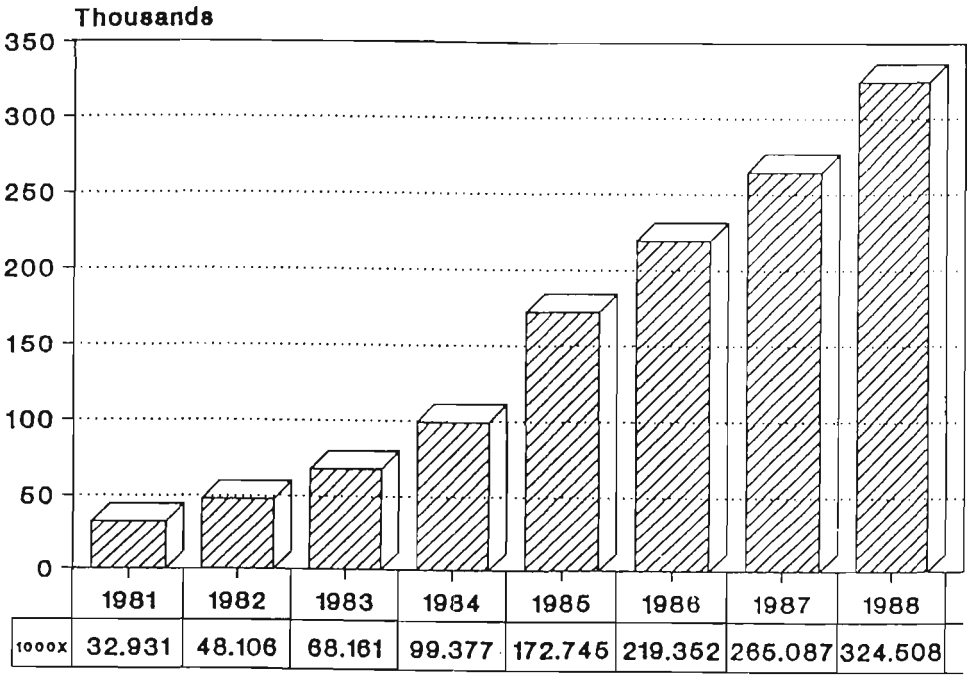


Figure 1-1: Robots population worldwide in the last decade.

of freedom serial manipulator can have up to 18 geometric parameters, 60 mass parameters, with 12 or more actuator parameters. The pre-production (design) cost of such a robot can be millions of dollars.

For example, it is reported that the Cincinnati Milacron T3 arm took 6 years to develop and an outlay of $6 million [36]. The NASA space shuttle manipulator is reported to have cost about $100 million [37]. As requirements for precision operation, cyclic speed of operation, external loads and complexity of geometry increase, the ability to meet complex design objectives becomes more critical. Also, since reliability is an inherent design characteristic, and robots is a growing field, (Figure 1-1), reliable design for reliable robots is a current great necessity [38]. Robot manipulator systems are difficult to design and one of the important means of easing this technological difficulty is the creation of a system of robust mathematical tools which is capable of making the design of new, more versatile systems feasible, and the fast and precise operation of these systems a reality.

1.3 Research Objectives

Because of the above problems and the complexity of design of robotic manipulators, this thesis contribution lies mainly in three different areas :

1. *Robot Dynamics*—to give a further insight into the performance characteristics of robot manipulators through :

   (a) an analytical study on the dynamic behaviour of robot arms under different velocity trajectories,

   (b) an analysis on the effect of links' dynamic balancing on the overall robot's dynamic behaviour, and

   (c) an analysis on the effect of load variation on the robot's dynamic characteristics.

2. *Robot Kinematics*—to establish quantitative measures of the effect of different kinematic parameters on the performance characteristics of a robotic arm.

3. *Robot Design*—to develop a new general design mechanism and computational methods which give a robot designer a quantitative feedback, with graphical illustration, regarding the influence of changing a robot's geometrical parameter on its optimal dynamic performance.

These three objectives represent the prime ingredients to establish reliable design tools. Through these design tools a robot designer can easily predict and foresee the impact of changing the operational conditions, (e.g. velocity or payload), or any of the kinematic parameters on the performance characteristics of a robotic manipulator.

In order to achieve the above-mentioned objectives, extensive software developments were required. The software was required to simulate the Lagrangian dynamics of both SCARA* and PUMA 560† robots. Also, software were developed for the kinematic analysis of PUMA 560 robot's arm. Also, the software function was for different velocity trajectories and robot's paths generation. Furthermore, the software was extended to allow for on line changes in the operating conditions (e.g. payload) and to study the effect of changing a robot's geometrical parameter on its dynamic behaviour.

1.4 Dissertation Outline

The work in this thesis includes mathematical modelling, software development, robot testing and computer simulation analysis. It also includes an analytical study on the dynamic response and sensitivity to changes in the kinematic parameters in order to develop design criteria and computational design tools for robotic manipulators. A plot package was especially developed as part of this research to reveal the special features of the study's outcome.

To accomplish the goals of this research, the dissertation is organised in the following manner :

- In Chapter 2, a review of robot dynamics is presented.
- In Chapter 3, the development of the kinematic model is presented and discussed.
- In Chapter 4, an analysis of the dynamic behaviour of a SCARA robot for different operating conditions under different velocity trajectories is pre-

*Selective Compliance Assembly Robot Arm.
†*PUMA* is a trade mark of UNIMATION.

sented. Also, the velocity trajectories used in the analysis are presented and discussed. Part of this Chapter has been published by Ibrahim *et al.* [39].

- In Chapters 5 and 6, the dynamic performance of a SCARA robot and a PUMA 560 were examined under different payloads and time-varying payload operating conditions. The PUMA 560 was further examined with its end-effector moving in a pre-specified Cartesian trajectory. Part of this work has been published by Ibrahim *et al.* [40] and another part is submitted for publication by Ibrahim *et al.* [41].

- In Chapter 7, a study was conducted using an experimental approach on the dynamic behaviour of a PUMA 560 with a time-varying payload condition. Also, a comparison between the computer simulation and real-time experiments are presented. Some of this work is submitted for publication by Ibrahim *et al.* [42].

- In Chapter 8, in pursuit of an optimal robot's design, a study on the extent of the effect of dynamic balancing of links was conducted.

- In Chapter 9, a performance indicator is developed to assist in the study of the effect of a robot's geometrical parameters on its dynamic behaviour. Part of this Chapter has been published by Ibrahim *et al.* [43].

- In Chapter 10, conclusions, discussion and recommendations for further study are presented.

Chapter 2

Dynamics Modelling

2.1 Introduction

An important initial step in analysing, designing or controlling a complex mechanical system, such as a robot, is to construct a representative model of the system. The model must be accurate enough to give results which satisfactorily describe the operation of the actual system, but is simple enough to be of practical use.

To satisfy these requirements, the rigid-links manipulator model is employed in this dissertation. The model simulates all the main forces that are present in any manipulator including the centrifugal/Coriolis forces terms.

The model, however, represents the main three terms affecting a robot's dynamics. These are :

- Inertial terms.
- Centripetal/Coriolis terms.
- Gravity terms.

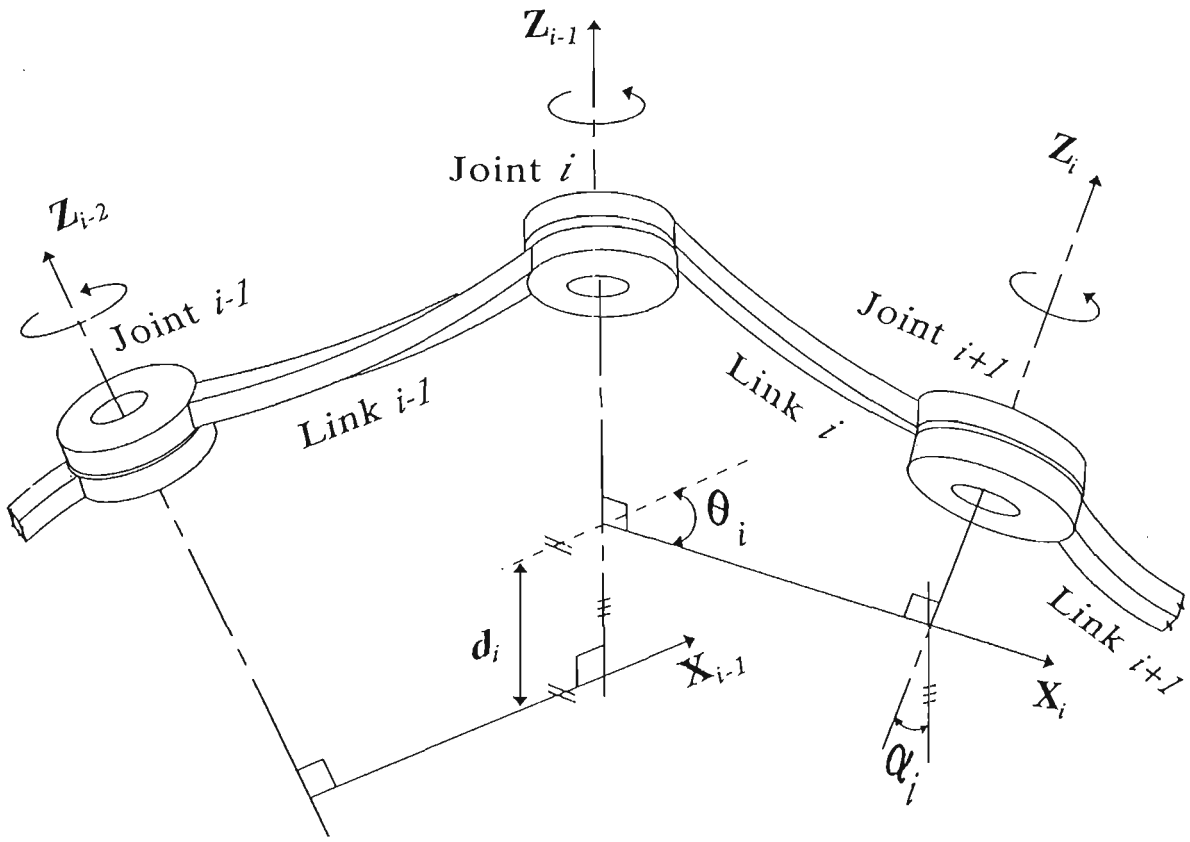


Figure 2-1: Link parameters θ, d, a and α.

The model employs the standard robot kinematics based on the Denavit-Hartenberg conventions [42] as explained below. More description on the derivation of the kinematic modelling of a robot's arm can be found in [43, 44, 17, 45-47]. The essential kinematic parameters of a revolute joint are schematically shown in Figure 2-1.

2.2 Review of Robot's Dynamics

As mentioned earlier, the question of dynamics and control cannot be altogether separated. Also, dynamics study plays an important role in the development of procedures for optimal design of industrial manipulators.

Manipulators are usually predetermined to work permanently or during certain time periods, under the same working conditions, the same environments or on the same programmed task. However, some manipulators operate under insufficiently defined working conditions or in environments with a degree of uncertainty. In this case on-line calculation of the dynamics and control parameters is necessary for calculating the programmed kinematics, within the limits of the kinematic and geometrical capabilities of the manipulator. In order to ensure and satisfy a good tracking quality of the trajectories; it is necessary also to calculate the required driving forces and/or torques, depending on other variables' conditions.

There are two main methods of forming the dynamic equations. These methods are the *Lagrangian* method and the *Newton-Euler* method.

The Lagrangian methods although slower than the Newton-Euler method, give a greater insight into the components of the dynamics model. Therefore, the Lagrangian method of modelling robot dynamics was adopted in this research.

In the following section the Lagrangian method will be presented first without proof. The proof can be easily followed in one or more references [17, 50, 51].

2.2.1 The Lagrangian model

The model of manipulator's dynamics via the Lagrangian method has an expression containing the effective inertias of each of the joints and the inertial coupling between them. Therefore, it allows one to determine :

1. The relationship between force/torque and acceleration at a joint.
2. The relationship between force/torque at a joint and acceleration at other joints.

The methods used in [51, 50] are based on simplified manipulator state equations, which are more useful than numerical techniques because the latter calculate the exact torques as function of positions, velocities and accelerations, but provide no insight into the system [17].

The general equation that describes the motion of an n-joint robot with one degree of freedom each is :

$$F_i = \sum_{j=1}^{n} D_{ij}\,\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\,\dot{q}_j\,\dot{q}_k + D_i \tag{2.1}$$

where :

$F_i \stackrel{def}{=}$ Force or torque acting at joint i.

$q,\ \dot{q},\ \ddot{q} \stackrel{def}{=}$ Position, velocity and acceleration of joint i variable.

$$D_{ij} = \sum_{p=max\,(i,j)}^{n} Tr\left(\frac{\partial T_p}{\partial q_j}\ J_p\ \left(\frac{\partial T_p}{\partial q_i}\right)^T\right)$$

$$D_{ijk} = \sum_{p=max\,(i,j,k)}^{n} Tr\left(\frac{\partial^2 T_p}{\partial q_j \partial q_k}\ J_p\ \left(\frac{\partial T_p}{\partial q_i}\right)^T\right)$$

$$D_i = \sum_{p=i}^{n} -m_p\, g^T \left(\frac{\partial T_p}{\partial q_i}\right) r_p$$

Then F_i can be expressed in a more expanded form as :

$$F_i = \sum_{j=1}^{n} \sum_{p=max\,(i,j)}^{n} Tr\left(\frac{\partial T_p}{\partial q_j}\, J_p\, \left(\frac{\partial T_p}{\partial q_i}\right)^T\right) \ddot{q}_j$$

$$+ \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{p=max\,(i,j,k)}^{n} Tr\left(\frac{\partial^2 T_p}{\partial q_j\, \partial q_k}\, J_p\, \left(\frac{\partial T_p}{\partial q_i}\right)^T\right) \dot{q}_j\, \dot{q}_k$$

$$+ \sum_{p=i}^{n} -m_p\, g^T \left(\frac{\partial T_p}{\partial q_i}\right) r_p \tag{2.2}$$

where :

$T_p \stackrel{def}{=} A_1 \times A_2 \times \cdots \times A_p$

$m_p \stackrel{def}{=}$ the mass of link p.

$r_p \stackrel{def}{=}$ the distance of the mass centre of link p with respect to link p's coordinate frame. It is a (4×1) row vector with the last component being equal to zero.

$Tr \stackrel{def}{=}$ Trace operator.

Also;

$$\frac{\partial T_p}{\partial q_j} = A_1 \times \cdots \times QA_j \times \cdots \times A_p \tag{2.3}$$

$$\frac{\partial^2 T_p}{\partial q_j\, \partial q_k} = A_1 \times \cdots \times QA_j \times \cdots \times QA_k \times \cdots \times A_p \tag{2.4}$$

where :

$Q \stackrel{def}{=}$ Partial derivative expressed as a matrix operator

Also, an infinite signal change in a transformation matrix for a revolute or prismatic joint can be defined as [17] :

$$\Delta_{revolute} = \begin{pmatrix} 0 & -d\theta & 0 & 0 \\ d\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.5}$$

or

$$\Delta_{prismatic} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & dd \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.6}$$

Hence, from equations (2.5) and (2.6), it is clear that the value of Q can be either :

1. For a revolute joint i (i.e. $q_i = \theta_i$) :

$$Q_{revolute} = \frac{\Delta_{revolute}}{d\theta} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.7}$$

thus when $i = k$:

$$Q_{\theta_j \theta_j} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.8}$$

or

2. For a prismatic joint i (i.e. $q_i = d_i$) :

$$Q_{prismatic} = \frac{\Delta_{prismatic}}{dd} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.9}$$

thus when $i = k$:

$$Q_{d_j d_j} = 0 \tag{2.10}$$

Proof of equations (2.7) and (2.9) can be found in [17, 47]. Also, in equation (2.2) :

$$J_p \stackrel{def}{=} 4 \times 4 \text{ pseudo inertia matrix} \tag{2.11}$$

Therefore;

$$J_p = m_p \begin{pmatrix} \left(\frac{-K^2_{pXX}+K^2_{pYY}+K^2_{pZZ}}{2}\right) & K^2_{pXY} & K^2_{pXZ} & x_p \\ K^2_{pXY} & \left(\frac{K^2_{pXX}-K^2_{pYY}+K^2_{pZZ}}{2}\right) & K^2_{pYZ} & y_p \\ K^2_{pXZ} & K^2_{pYY} & \left(\frac{K^2_{pXX}+K^2_{pYY}-K^2_{pZZ}}{2}\right) & z_p \\ x_p & y_p & z_p & 1 \end{pmatrix} \tag{2.12}$$

where :

$K^2_p ab \stackrel{def}{=}$ radius of gyration ab of link p, with $a, b = x, y, z$ being the Cartesian coordinates fixed in the same link.

$= \frac{I_p ab}{m_p}$ ($I_p ab$ is the inertia tensor)

$\left.\begin{matrix} x_p \\ y_p \\ z_p \end{matrix}\right\} \stackrel{def}{=}$ Components of $\vec{r}_p$, the mass centre vector of link p.

Based on the above explanation of robot's dynamics, the building blocks of equation (2.2) can be defined such that :

D_i The gravity load felt at joint (i).

D_{ii} The true inertia felt at joint (i).

D_{ij} The coupling inertia felt at joint (i), due to the acceleration of joint (j).

D_{ijj} The centripetal force at joint (i), due to the velocity of joint (j).

D_{ijk} The Coriolis force felt at joint (i), due to velocities of joints (j) and (k).

It is clear that robot dynamic model is lengthy, complicated and time consuming for on-line applications. It was estimated in [52] that to compute the forces/torques for one nominal point in the trajectory requires 7.9 seconds on a PDP 11/45 computer.

2.3 Dynamic Modelling

In this research two different dynamic models were developed :

- Iterative dynamic model.
- Closed-form dynamic model.

These two modelling techniques are used in this research and are discussed below.

2.3.1 Iterative model

The iterative dynamic model was used to reduce the running time of the programme. On the other hand, when it was necessary to obtain a single force/torque component the reduced closed-form model was used.

The iterative simulation model was based on the technique reported in [53, 54]. In this technique advantage has been taken of backward recurrence.

To explain the backward recurrence, consider that the generalised force equation (2.2) can be expressed as :

$$F_i = \sum_{j=i}^{n} \left[Tr \left(\frac{\partial T_j}{\partial q_i} \, J_j \, \ddot{T}_j^T \right) - m_j \, g^T \, \frac{\partial T_j}{\partial q_i} \, r_j \right]_{(i=1,\ldots,n)} \tag{2.13}$$

where :

$$\ddot{T} = \sum_{k=1}^{j} \frac{\partial T_j}{\partial q_k} \, \ddot{q}_k + \sum_{k=1}^{j} \sum_{l=1}^{j} \frac{\partial^2 T_j}{\partial q_k \, \partial q_l} \dot{q}_k \dot{q}_l$$

which yields the usual form for the components of the reaction and the Coriolis and centripetal forces.

However, it was found that it is better to leave F_i in its compact form without expanding $\ddot{T}$. It was also found that the velocities $\dot{T}_j$ and acceleration $\ddot{T}_j$ can be easily derived by the following straightforward differentiation :

$$T_j \;=\; T_{j-1} A_j \tag{2.14}$$

$$\begin{aligned}\dot{T}_j &= \dot{T}_{j-1}A_j + T_{j-1}\dot{A}_j \\ &= \dot{T}_{j-1}A_j + T_{j-1}\frac{\partial A_j}{\partial q_j}\dot{q}_j \end{aligned} \tag{2.15}$$

$$\begin{aligned}\ddot{T}_j &= \ddot{T}_{j-1}A_j + \dot{T}_{j-1}\dot{A}_j + \dot{T}_{j-1}\frac{\partial A_j}{\partial q_j}\dot{q}_j \\ &\quad + T_{j-1}\frac{d}{dt}\left(\frac{\partial A_j}{\partial q_i}\right)\dot{q}_j + T_{j-1}\frac{\partial A_j}{\partial q_j}\ddot{q}_j \\ &= \ddot{T}_{j-1}A_j + 2\dot{T}_{j-1}\frac{\partial A_j}{\partial q_j}\dot{q}_j \\ &\quad + T_{j-1}\frac{\partial^2 A_j}{\partial q_j^2}\dot{q}_j^2 + T_{j-1}\frac{\partial A_j}{\partial q_j}\ddot{q}_j \end{aligned} \tag{2.16}$$

Comparing (2.16) and (2.2), it is clear that this formula requires only calculation of $\frac{\partial^2 T_i}{\partial q_i^2}$ instead of all the matrices $\frac{\partial^2 T_j}{\partial q_k \partial q_l}$. This provides a considerable reduction in calculation requirement from an n^4 to an n^2 dependence.

To gain further computational efficiency, the following kinematical relationship was exploited [54] :

$$\frac{\partial T_j}{\partial q_i} = \frac{\partial T_i}{\partial q_i}\,{}^iT_j$$

Then the Lagrangian model can be expressed as follows :

$$\begin{aligned}F_i &= \sum_{j=1}^{n}\left[Tr\left(\frac{\partial T_i}{\partial q_i}T_j\, J_j\, \ddot{T}_j^T\right) - m_j\, g^T\frac{\partial T_i}{\partial q_i}T_j\, r_j\right] \\ &= Tr\left(\frac{\partial T_i}{\partial q_i}\sum_{j=1}^{n}{}^iT_j\, J_j\, \ddot{T}_j^T\right) - g^T\frac{\partial T_i}{\partial q_i}\sum_{j=1}^{n}m_j\,{}^iT_j\, r_j \end{aligned} \tag{2.17}$$

Forward recursion has been used very successfully in this formulation as follows :

Suppose that the first term of equation (2.17) contains B_i, where :

$$B_i = \sum_{j=1}^{n} {}^{i}T_j \, J_j \, \ddot{T}_j^T$$

Therefore

$$\begin{aligned} B_i &= {}^{i}T_i \, J_i \, \ddot{T}_i^T + \sum_{j=i+1}^{n} {}^{i}T_{i+1} \, {}^{i+1}T_j \, J_j \, \ddot{T}_j^T \\ &= J_i \ddot{T}_i^T + A_{i+1} \sum_{j=i+1}^{n} {}^{i+1}T_j \, J_j \, \ddot{T}_j^T \qquad (\text{since } {}^{i}T_i = I \text{ and } {}^{i}T_{i+1} = A_{i+1}) \\ &= J_i \ddot{T}_i^T + A_{i+1} \, B_{i+1} \end{aligned} \tag{2.18}$$

Also, if it is assumed in the second term of (2.17) that :

$$C_i = \sum_{j=i}^{n} m_j \, {}^{i}T_j \, r_j$$

Therefore

$$\begin{aligned} C_i &= m_i \, {}^{i}T_i \, r_i + \sum_{j=i+1}^{n} m_j \, A_{i+1} \, {}^{i+1}T_j \, r_j \\ &= m_i \, r_i + A_{i+1} \sum_{j=i+1}^{n} m_j \, {}^{i+1}T_j \, r_j \\ &= m_i \, r_i + A_{i+1} \, C_{i+1} \end{aligned} \tag{2.19}$$

By substituting from (2.18) and (2.19) in (2.17) :

$$F_i = Tr \left(\frac{\partial T_i}{\partial q_i} \, B_i \right) - g^T \frac{\partial T_i}{\partial q_i} \, C_i \tag{2.20}$$

Some of the simulation study in this thesis was based on the recursive technique to find the required force/torque. Two sets of successive computation were conducted in two different directions :

First

All the $\ddot{T}_i^T$ terms to be computed successively starting from $i = 1$ to $i = n$, using equation (2.19).

Then

Knowing the values of $\ddot{T}^T$, the terms B_i and C_i are computed successively starting from $i = n$ to $i = 1$, using equations (2.18,2.19).

2.3.2 Closed-form model

The above method was fast enough to conduct an off-line analytical study of a robot arm's dynamics. Due to the special geometrical configuration of the SCARA robot (Chapter 4) it was possible to use this technique to gain an insight into the effect on dynamic behaviour of different parameters. However, that was not the case with a geometrically articulated robot's arm (PUMA 560). Therefore, it was necessary to implement the closed-form model which is based on equation (2.2). To speed up the computational time, some parameter reduction took place in the specially developed software for robot performance analysis. The parameter reductions were as follows :

1. $\left.\begin{array}{rcl} D_{ij} & = & 0 \\ D_{ijk} & = & 0 \end{array}\right\}$ for $j, k < i$

   These terms can be omitted from the trace of chain products of the matrices.

2. $D_{ijk} = D_{ikj}$

   This can be seen from the fact that :

$$\frac{\partial^2 T_i}{\partial q_j \, \partial q_k} = \frac{\partial^2 T_i}{\partial q_k \, \partial q_j} \qquad \text{(see equation (2.4))}$$

3. $D_{ijk} = -D_{kji}$ for $i, k > j$

   This can be seen by noting that :

$$Q^T = -Q \qquad \text{(see equation (2.7))}$$

4. $D_{iji} = 0 \qquad \text{for } i \geq j$

   This is due to the fact that the trace of the product of a symmetric and a skew-symmetric matrix is identically zero. (A limiting case for this reduction is $D_{iii} = 0$, i.e, the centripetal force of a revolute joint is not felt by the motor of the same joint.)

Although the programme runs off-line and the time was not of critical importance, the above reductions helped in speeding the feedback for the analysis purposes.

2.4 Summary

Two dynamic modelling techniques were used in the course of the analytical study of this thesis. The first technique, based on the iterative Lagrangian algorithm, was used to study the dynamic behaviour of a SCARA robot under different velocity trajectories and time-varying payload conditions. The iterative technique was selected for two main reasons :

1. Its relative computational speed compared with the closed form model [54].

2. Given the special geometrical configuration of the SCARA robot, it was possible to use this technique to study the effect on dynamic behaviour of different terms.

The second technique was based on the closed form of the Lagrangian dynamic model. That particular technique was used in the study of the PUMA 560 robot arm. Despite the fact that it requires more calculation time than the iterative model, it helped in conducting the required analysis on the dynamic influence of each loading parameter in the articulated robot arm "PUMA 560".

Also, improvements in the calculation time was achieved by using the reduction mentioned in Section 2.3.2.

Chapter 3

Kinematic Modelling

3.1 Introduction

In order to compare the results of the simulation study on the dynamic behaviour under time-varying payload with the results of a real robot; it was necessary to model the kinematics of an available robot. The robot that became available through this research, is an articulated type "PUMA 560". The geometrical configuration of this type of robot is shown in Figure 3-1.

For this reason a complete kinematic model was developed for this type of robot to generate the inverse kinematic solution for alternative geometrical configurations. This kinematic model was coupled to the general dynamic model to form a more powerful research tool.

3.2 Cartesian Path

Given the initial and final Cartesian coordinates of the end-effector movement, which is assumed to be in a straight line, the length of the Cartesian motion

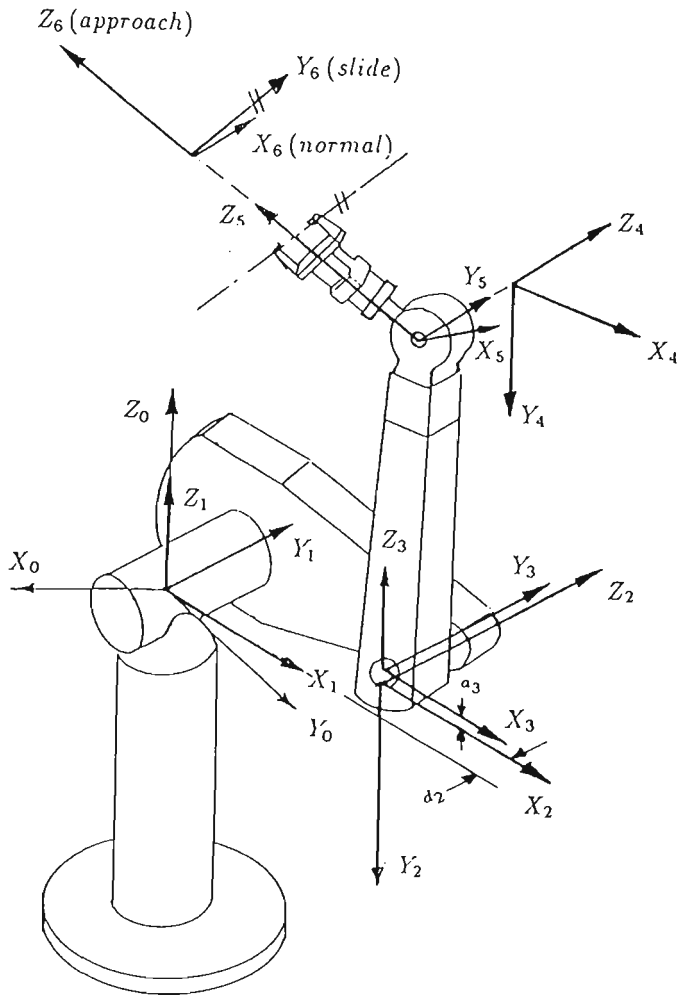


Figure 3-1: Kinematic parameters of "PUMA 560" robot manipulator.

$(\vec{\Re})$ can be calculated as :

$$\|\vec{\Re}\| = \sqrt{(X_1 - X_0)^2 + (Y_1 - Y_0)^2 + (Z_1 - Z_0)^2} \tag{3.1}$$

Where :

$\vec{\Re} \stackrel{def}{=}$ A vector representing the Cartesian path of the end-effector

$X_0, Y_0, Z_0 \stackrel{def}{=}$ The Cartesian coordinates of the starting position of the end-effector.

$X_1, Y_1, Z_1 \stackrel{def}{=}$ The Cartesian coordinates of the final position of the end-effector.

The Cartesian path vector's angles with the Cartesian axes can be calculated as follows : The angle $(\alpha_{\vec{\Re}})$ with the X-axis :

$$\alpha_{\vec{\Re}} = \tan^{-1} \frac{\sqrt{(Y_1 - Y_0)^2 + (Z_1 - Z_0)^2}}{X_1 - X_0} \tag{3.2}$$

Also, the angle $(\beta_{\vec{\Re}})$ with the Y-axis :

$$\beta_{\vec{\Re}} = \tan^{-1} \frac{\sqrt{(X_1 - X_0)^2 + (Z_1 - Z_0)^2}}{Y_1 - Y_0} \tag{3.3}$$

Similarly, the angle $(\gamma_{\vec{\Re}})$ with the Z-axis :

$$\gamma_{\vec{\Re}} = \tan^{-1} \frac{\sqrt{(X_1 - X_0)^2 + (Y_1 - Y_0)^2}}{Z_1 - Z_0} \tag{3.4}$$

Having completely defined the Cartesian path, the problem now is to conduct a time-based discretisation of the path. That leads to a set of points on the Cartesian path at equal time intervals rather than at equal distances. The positions of these points on the Cartesian path should also satisfy the desired velocity trajectory, as illustrated in Figure 3-3.

This time-based discretisation was achieved using the following three equations :

$$X_t = X_0 + P_t \cos \alpha_{\vec{\Re}} \tag{3.5}$$

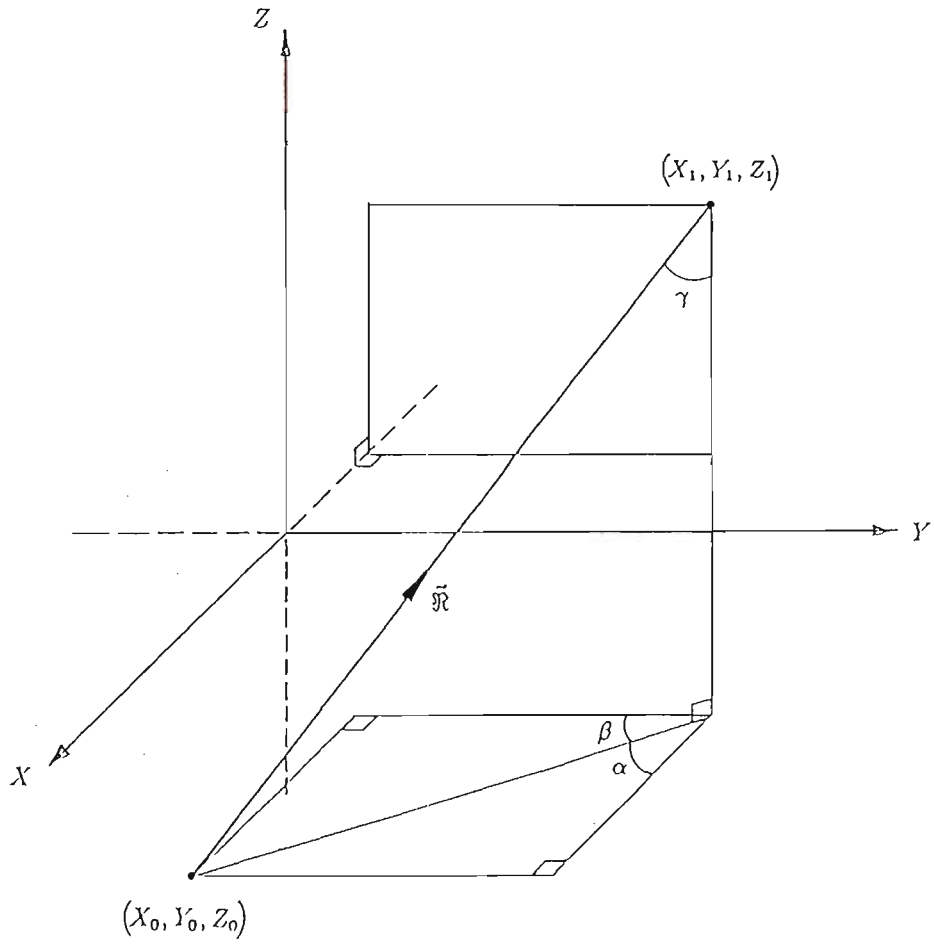


Figure 3-2: Cartesian path and its angles with respect to the world coordinate frame.

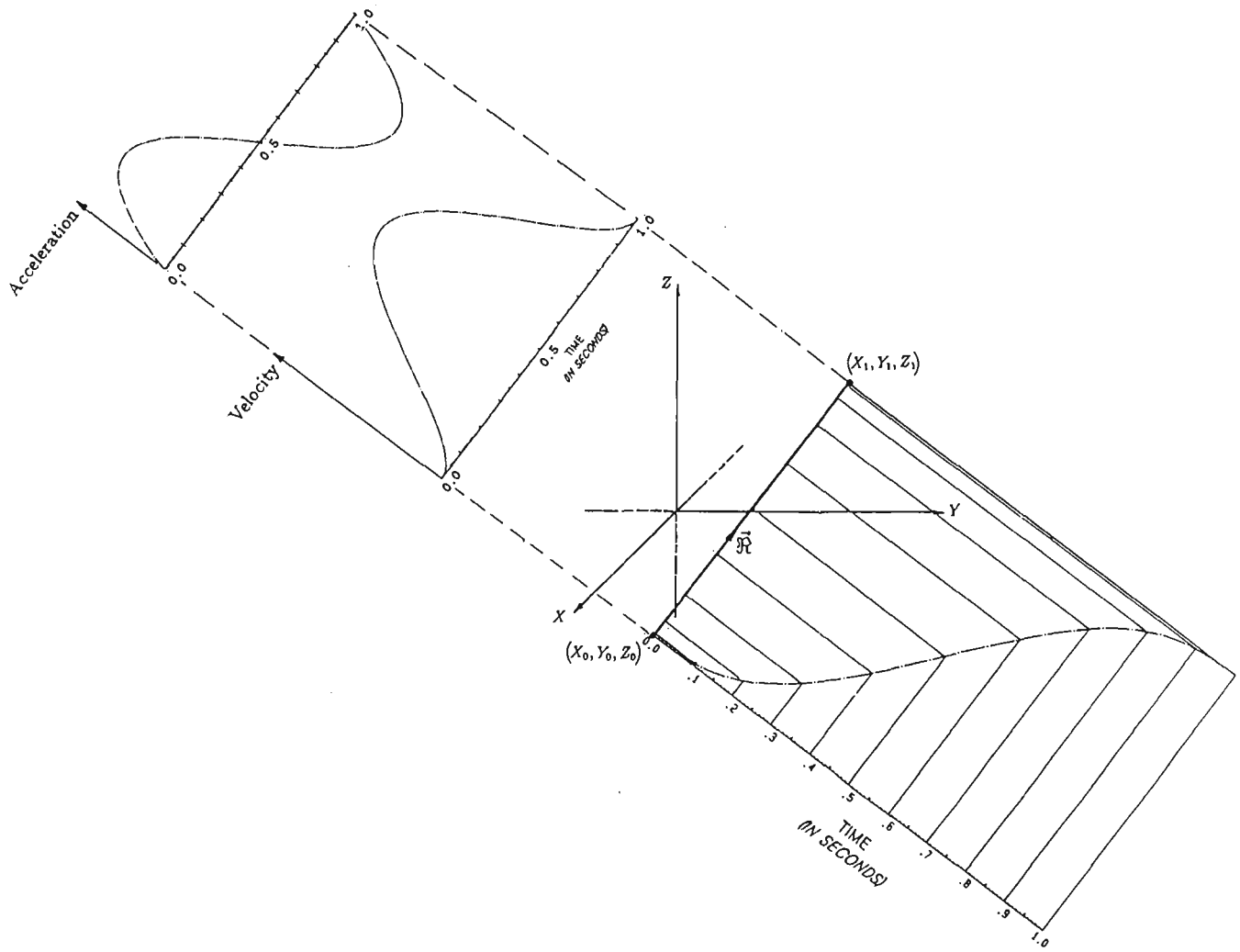


Figure 3-3: Time-based discretisation of the Cartesian trajectory.

$$Y_t = Y_0 + P_t \cos\beta_{\vec{R}} \tag{3.6}$$

$$Z_t = Z_0 + P_t \cos\gamma_{\vec{R}} \tag{3.7}$$

Where :

$X_t, Y_t, Z_t \stackrel{def}{=}$ The Cartesian coordinates of the end-effector at time (t).

$P_t \stackrel{def}{=}$ The linear displacement on the Cartesian path at time (t), with a zero starting value.

The position P_t can be obtained from a trajectory generator such as the polynomial trajectory explained on page 38. Knowing the Cartesian coordinates of these equal time-interval positions, the path positions can then be mapped into their corresponding joint positions. That can be achieved using the inverse kinematic solution.

3.3 Inverse Kinematic

In order to obtain the corresponding joints displacement that can lead to the resultant Cartesian trajectory, each position on the Cartesian path P_t is mapped to its corresponding joint position. That has been achieved numerically using the explicit inverse kinematic of the locally available robot (PUMA 560). Therefore, the computer model was extended to include the inverse kinematic model of the locally available robot PUMA 560. Each Cartesian position of the PUMA 560 has eight joint-space different solutions, as shown in Figure 3-4.

The geometrical approach to the solution of the inverse kinematic was used [55]. It was found that this solution gives more compatible results with those resulting from the robot's own controller than the analytical approach [45]. However, it was also found during the validation that there are inherent errors in the kinematic solution of the robot's controller. These errors were manifested

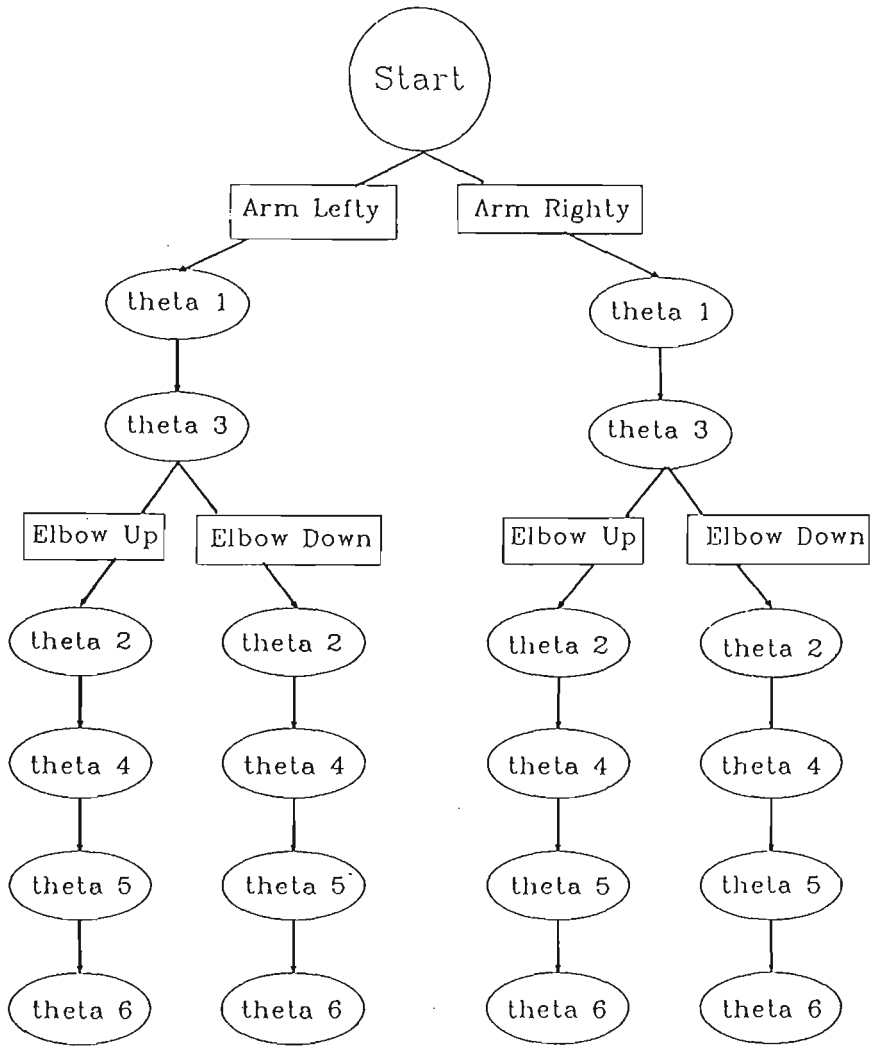


Figure 3-4: Inverse kinematic solution of "PUMA 560" robot manipulator.

in the discrepancy between the end-effector Cartesian positions to reach a certain point in space with elbow-up and elbow-down configurations.

The geometrical inverse kinematic solution was based on the following equations :

$$\theta_{1_t} = \tan^{-1} \frac{\pm Y_t \sqrt{X_t^2 + Y_t^2 + d_2^2} - d_2 X_t}{\pm X_t \sqrt{X_t^2 + Y_t^2 + d_2^2} + d_2 Y_t} \tag{3.8}$$

$$\theta_{2_t} = \tan^{-1} \left\{ \frac{-\left[Z_t (a_2 + d_4 S_3) + (d_4 C_3) \left(\pm\sqrt{X_t^2 + Y_t^2 - d_2^2}\right)\right]}{Z_t (d_4 C_3) - (a_2 + d_4 S_3) \left(\pm\sqrt{X_t^2 + Y_t^2 - d_2^2}\right)} \right\} \tag{3.9}$$

$$\theta_{3_t} = \tan^{-1} \left\{ \frac{X_t^2 + Y_t^2 + Z_t^2 - d_4^2 - a_2^2 - d_2^2}{\pm\sqrt{4 d_4^2 a_2^2 - (X_t^2 + Y_t^2 + Z_t^2 - d_4^2 - a_2^2 - d_2^2)^2}} \right\} \tag{3.10}$$

It should be noted that the two solutions of equation (3.8) resulting from the "$\pm$" sign are responsible for the two feasible configurations : arm *lefty* or arm *righty*. Similarly, the two solutions of equations (3.9) and (3.10) are responsible for the two alternative configurations : elbow *up* or elbow *down*. Proof of equations 3.8 to 3.10 can be found in [56]. Also the package calculates the kinematic solution for θ_{4_t}, θ_{5_t} and θ_{6_t}. However, these angles are responsible for the spatial-orientation of the end-effector. Therefore, their values are functions of the elements of the Cartesian orientation matrix. The orientation sub-matrix of T_6 can be expressed as [57] :

$$R(\vartheta,\varphi,\zeta)=\begin{pmatrix}(C\zeta\,C\varphi) & (C\zeta\,S\varphi\,S\vartheta-S\zeta\,C\vartheta) & (C\zeta\,S\varphi\,C\vartheta+S\zeta\,S\vartheta)\\ (S\zeta\,C\varphi) & (S\zeta\,S\varphi\,S\vartheta+C\zeta\,C\vartheta) & (S\zeta\,S\varphi\,C\vartheta-C\zeta\,S\vartheta)\\ -S\varphi & (C\varphi\,S\vartheta) & (C\varphi\,C\vartheta)\end{pmatrix} \tag{3.11}$$

Where :

ϑ , φ and ζ are the rotation angles around X-axis, Y-axis and Z-axis respectively of the end-effector's coordinate frame.

Following the evaluation of R matrix's elements; θ_{4_t}, θ_{5_t} and θ_{6_t} were calculated using the following algebraic solution :

$$\theta_{4_t} = \tan^{-1}\left\{\frac{-r_{13}\,S_1+r_{23}\,C_1}{-r_{13}\,C_1\,C_{23}-r_{23}s_1C_{23}+r_{33}S_{23}}\right\} \tag{3.12}$$

$$\theta_{5_t} = \tan^{-1}\left\{\frac{-\left[r_{13}\,(C_1C_{23}C_4+S_1S_4)+r_{23}\,(S_1C_{23}C_4-C_1S_4)\right.}{r_{13}\,(-C_1S_{23})+r_{23}\,(-S_1S_{23})+r_{33}\,(-C_{23})}\right.$$
$$\left.\frac{-r_{33}\,(S_{23}C_4)]}{+r_{33}\,(-C_{23})}\right\} \tag{3.13}$$

$$\theta_{6_t} = \tan^{-1}\left\{\frac{-r_{11}\,(C_1S_{23}S_4-S_1C_4)}{r_{11}\,[(C_1C_{23}C_4+S_1S_4)\,C_5-C_1S_{23}S_5]}\right.$$
$$\frac{-r_{21}\,(S_1C_{23}S_4+C_1C_4)}{+r_{21}\,[(S_1C_{23}C_4-C_1S_4)\,C_5-S_1S_{23}S_5]}$$
$$\left.\frac{+r_{31}\,(S_{23}S_4)}{-r_{31}\,(S_{23}C_4C_5+C_{23}S_5)}\right\} \tag{3.14}$$

Where :

$C_{ij} \overset{def}{=} \cos(\theta_i+\theta_j)$

$S_{ij} \overset{def}{=} \sin(\theta_i+\theta_j)$

$r_{ij} \overset{def}{=}$ element (ij) of the rotation matrix "R".

It is assumed that the robot completes its motion cycle within one second. Consequently, the cycle time is discretised in 61 points to contain 60 equal time intervals. The four feasible solutions for the six displacement angles are stored in a three dimensional array ($4 \times 6 \times 61$). The angles are then arranged according to their dependency in six separate arrays to give four different sets of displacement trajectories for the six robot's angles.

From the analysis made on the kinematics of the PUMA 560, it was found that the dependency of the joints angles is as follows :

$$\left.\begin{matrix}\theta_1 \\ \theta_3\end{matrix}\right\} \text{ are not functions of any other angle.}$$

$$\begin{aligned}\theta_2 &= f(\theta_3) \\ \theta_4 &= f(\theta_2) = f(\theta_3) \\ \theta_5 &= f(\theta_1, \theta_4, \theta_3) \\ \theta_6 &= f(\theta_1, \theta_4, \theta_5)\end{aligned}$$

The output of the inverse kinematic module of this package are six arrays for the six joint displacements. These arrays are then processed to get the corresponding velocity and acceleration of each link.

3.4 Summary

In order to conduct the simulation study on the dynamic behaviour of robot manipulators it was essential to develop the kinematic model and couple it to the dynamic model. Knowing the initial and final positions of the end effector, the trajectory's path was interpolated at equal time intervals. The software was developed to calculate each point on the trajectory's path such that it satisfies the desired cartesian displacement trajectory.

The calculated points on the cartesian trajectory were then mapped, via

the inverse kinematic model, to obtain the corresponding joint-space positions for the robot's possible configurations. The resulting position array of each joint was then processed to obtain the corresponding joint velocity and acceleration. Therefore the robot's inverse kinematic model had to be incorporated in the software to enable this process. The model was based on the geometrical approach to the solution of the inverse kinematic.

The integration of both kinematic and dynamic models in the software developed particularly for this thesis produced a powerful research tool which helped in conducting the analytical study of this thesis.

Chapter 4

Dynamic Behaviour Analysis of a Robot Subjected to Different Velocity Trajectories

4.1 Introduction

In robot design, it is often necessary to investigate the robot's performance when one or more working conditions vary over a given range. The analysis on which to base the design methods involves the multivariable mathematical relations between the design parameters and the manipulator's force and motion states. This analysis is complex, non-linear and highly coupled. The computer package developed in this thesis is specially designed to display the extensive analysis information in a compact format to help the designer develop an intuitive feel for his problem, re-enforced by the displays.

In this chapter, the dynamic behaviour of a SCARA robot will be examined for different operating velocity trajectories.

Since the first three links of any robot's arm are usually the longest and the heaviest, they have the greatest influence on the dynamic behaviour. This study will concentrate mainly on these links. The links of the manipulator have the geometrical configuration as shown in Figure 4-1. In this configuration links 1 and 2 move in a horizontal plane, Figures A.1 – A.2, while link 3 in Figure A.3, moves in a vertical plane.

To describe the translational and rotational relationship between the links of this manipulator, the Denavit-Hartenberg method [44, 58, 59] is used.

Typical SCARA robots are usually five or six degrees of freedom. In the simulation, the mass of the wrist components, the gripper and pay-load are represented by a simple spherical mass of 5 Kg. This mass is connected to the lower end of the third link as shown in Figure 4-1. The computer simulation model requires that the values of the link's inertia matrix are supplied as data. Therefore, a complete derivation of the inertia matrix values for each link was conducted. This derivation is detailed in Appendix A.

In order to study the dynamic behaviour and the effect of coupling on the required joint's torque/force two different trajectories were applied. A brief description of these trajectories is in the following section.

4.2 Applied Trajectories

The two types of velocity trajectories used in this study to examine different characteristics of the dynamic behaviour were namely :

1. Polynomial trajectories.

2. Numerical Control 2 (*NC2*) trajectories.

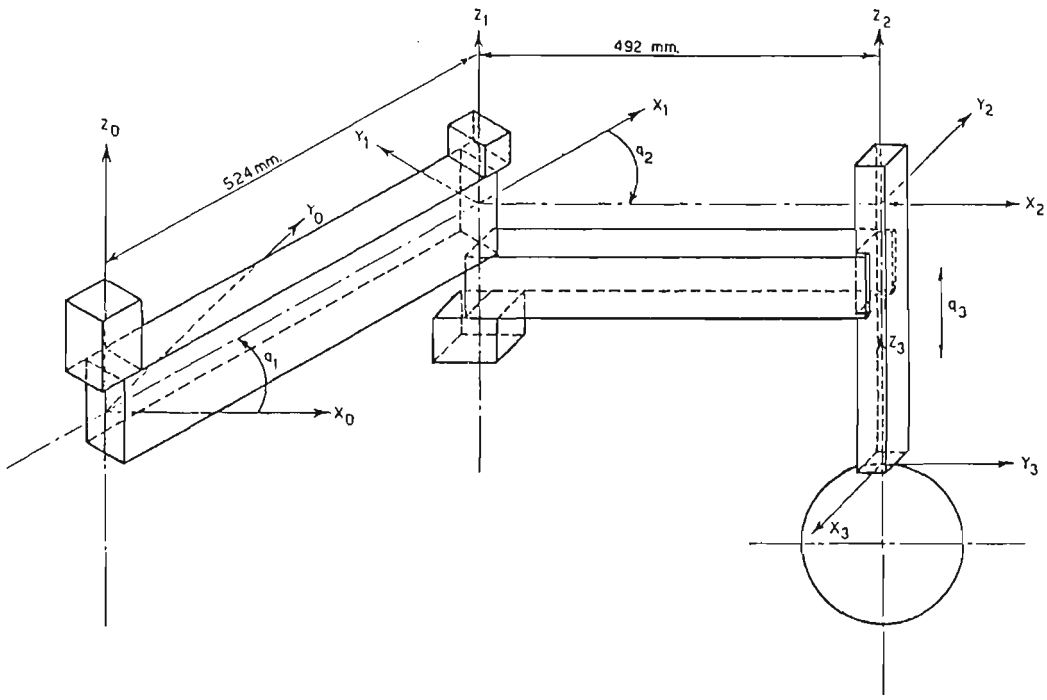


Figure 4-1: Schematic diagram of SCARA robot.

The main characteristics of these trajectories are as follows :

4.2.1 Polynomial trajectories

In these trajectories, the displacement of a link in space is a third order polynomial function in time. In this function the position should satisfy the following relationship [60, 61] :

$$\begin{aligned} q_t &= (1-t)^3\left\{q_0 + (3q_0 + \dot{q}_0)\,t + (\ddot{q}_0 + 6\dot{q}_0 + 12q_0)\,t^2/2\right\} \\ &\quad + t^3\{q_1 + (3q_1 + \dot{q}_1)\,(1-t) \\ &\quad + (\ddot{q}_1 + 6\dot{q}_1 + 12q_1)\,(1-t)^2/2\} \end{aligned} \tag{4.1}$$

$q_t \stackrel{\text{def}}{=}$ the position at time-instant (t)

$q_0 \stackrel{\text{def}}{=}$ the initial position in space at $(t = 0)$

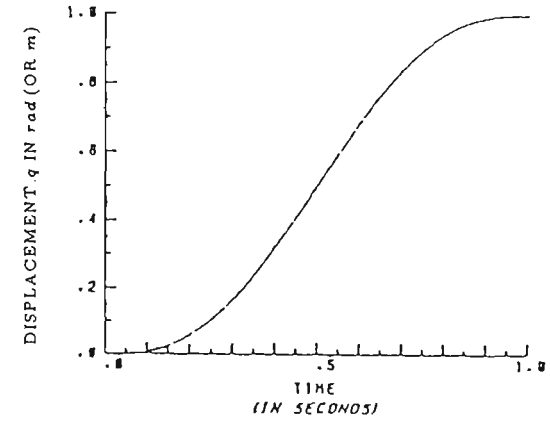
$q_1 \stackrel{\text{def}}{=}$ the final position reached at end of stroke at $(t = 1)$

The main characteristics of these types of trajectories, as shown in Figure 4-2, is that they give smooth position, velocity and acceleration progression. Its main limitation is that it does not satisfy the time optimality requirement for fast operation. For this reason the following type of velocity trajectory was also implemented in this research.

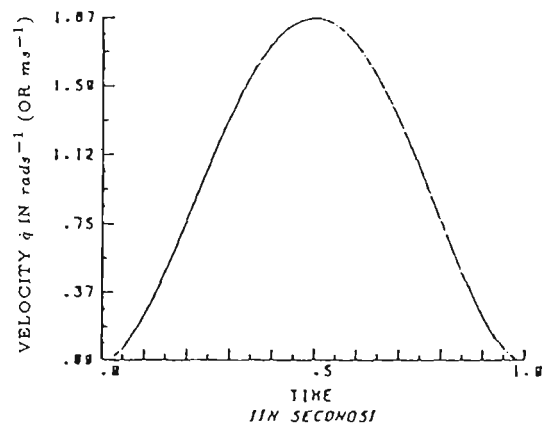
4.2.2 NC2 (Numerical Control 2) trajectories

In order to achieve minimal time displacement Bang-Bang trajectories must be used. The configuration of this trajectory, as shown in Figure 4-3, is constant acceleration – constant deceleration.

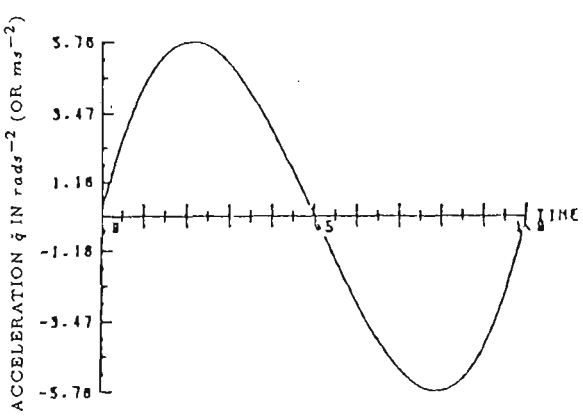
For a longer displacement and due to the actuator's limitations, the trapezoidal velocity trajectories in Figure 4-4 are the optimal time solution. The prob-



(a)



(b)



(c)

Figure 4-2: Polynomial trajectory.

lem associated with these kinds of trajectories is the vibration they cause due to the discontinuity at changeover points of the acceleration. This vibration can also affect the precision in position of the end effector at the end of the stroke.

To overcome the problem of vibration and to maintain, as much as possible the time optimality, the *NC2* velocity trajectory was introduced. This trajectory, as shown in Figure 4-5, has the following characteristics [39] :

1. There is no discontinuity in the acceleration except at the starting point to minimise the effect of vibration.
2. The maximum velocity occurs at 0.333 of the total displacement time, making the deceleration time twice as long as the acceleration time; thus further reducing any vibration effect.
3. The acceleration discontinuity at the end of stroke is eliminated.
4. This kind of trajectory is much closer to the time optimality than most of the commonly known trajectories mentioned in [61].

4.3 The Dynamic Behaviour Analysis

The study of the dynamic behaviour of the SCARA robot [39] was based on the complete Lagrangian model expressed in equation 2.2 as explained in Section 2.2.1.

Throughout this study, all links are specified to start and finish their individual motion at the same time. The study has been applied on a few operational conditions to examine the performance characteristics under those conditions. The outcome of three main applications is discussed below.

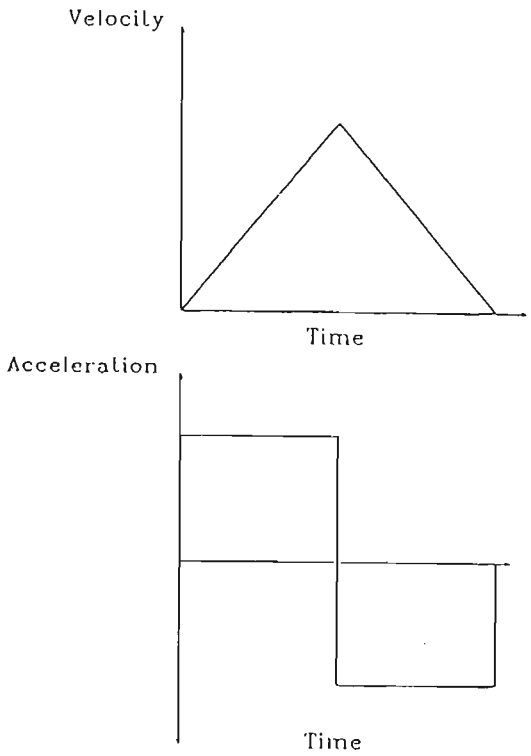


Figure 4-3: Bang-Bang trajectory.

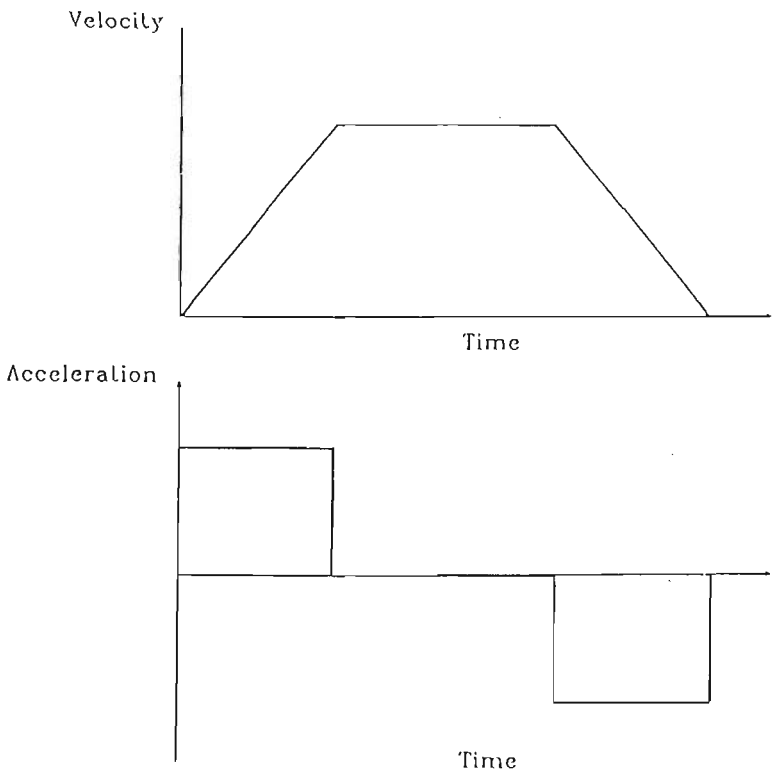
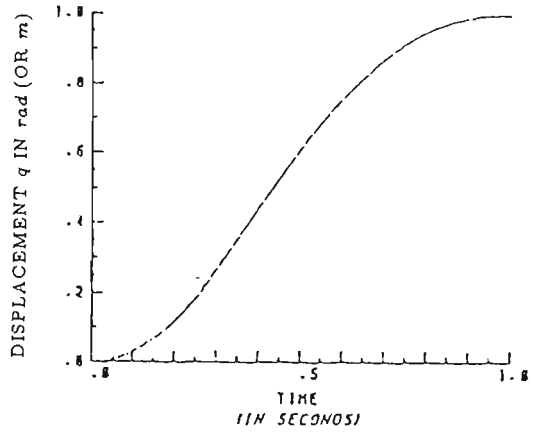
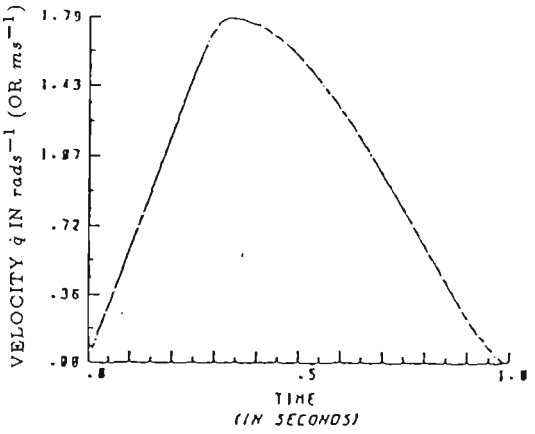


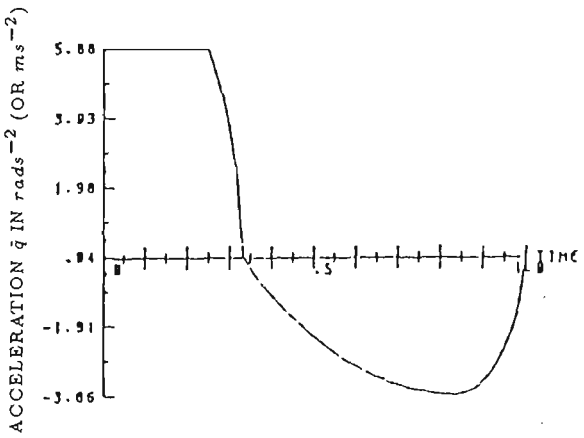
Figure 4-4: Trapezoidal trajectory.



(a)



(b)



(c)

Figure 4-5: *NC2* trajectory.

4.3.1 First application

To examine the effect of the velocity trajectories on the dynamics of the manipulator, various velocity trajectories of the polynomial type were applied on all links to examine the dynamic response for each trajectory. This procedure has been repeated using *NC2* type trajectories for comparison. All link trajectories are specified to start from the position $q_1 = q_2 = q_3 = 0$ and each link moves in its own positive direction to the position $q_1 = q_2 = q_3 = 1$ rad. . The results are shown in Figure 4-6 and Figure 4-7.

The results of the first two links will be discussed together because of the similarity of their motion and geometrical configuration, and those for the third link will be considered separately.

The response of the first two links

In the light of the results obtained corresponding to the increasing velocity trajectories, the following analysis can be made :

1. At a lower velocity trajectory, the required torque has taken the pattern of the acceleration. The physical interpretations for this are :

   (a) These links move in a horizontal plane; hence there is no movement in the vertical direction. Therefore, their actuators do not work against gravity and the third term of the Lagrangian model becomes zero.

   (b) The required torque at a lower velocity trajectory is dominated by the acceleration dependent inertia force as indicated by the curves for small torques shown in Figure 4-7-b for example.

2. It can be seen that the magnitude of the required torque, at the moment of zero acceleration (maximum velocity), becomes greater than zero and

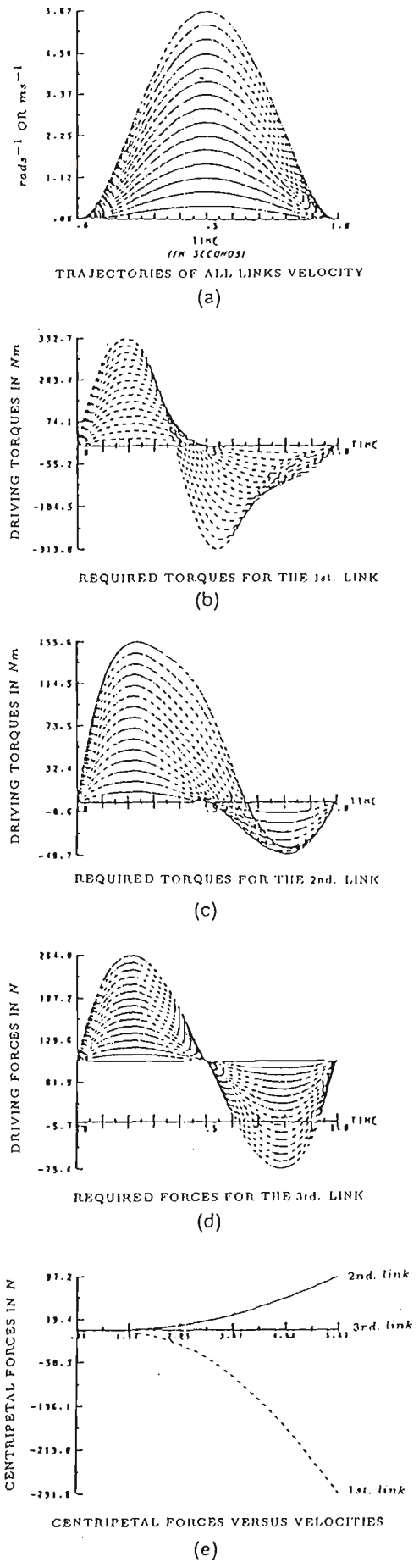


Figure 4-6: Dynamic behaviour for increasing polynomial trajectory.

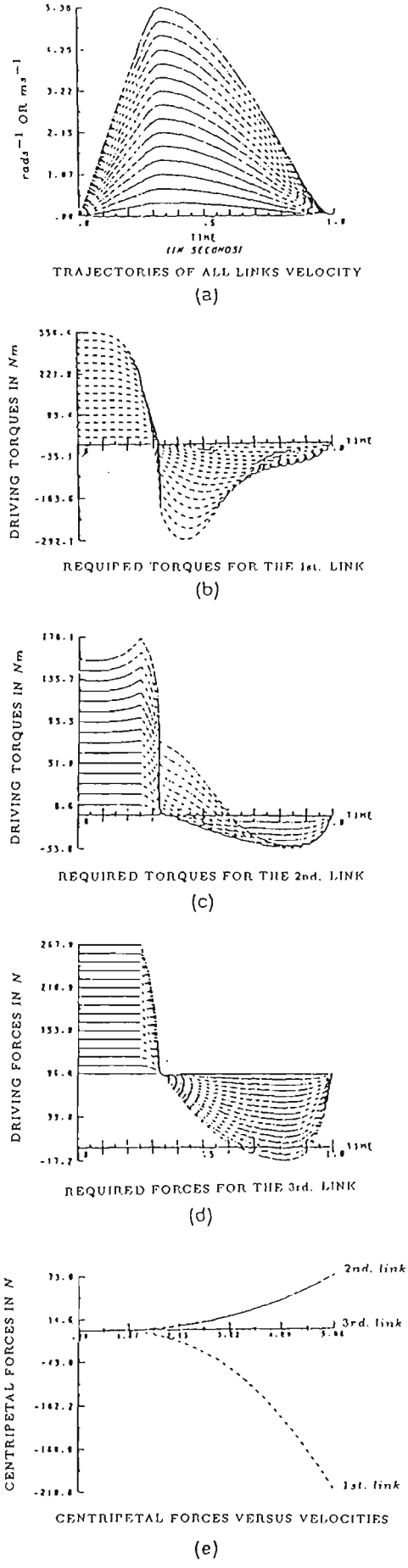


Figure 4-7: Dynamic behaviour for increasing *NC2* trajectory.

increases as the velocity increases as shown in Figures 4-6-c and 4-7-c. This is because at these particular moments both the acceleration-dependent inertia force and the gravitational force are equal to zero and so the non-zero value of the required force at these instances is due to the centripetal and Coriolis forces in the link 'i', which is a function of $\dot{q}_j, \dot{q}_k (j, k = 1, 2, \ldots, N)$. As a consequence of that, as the maximum velocity of the links increases, the required torque peak shifts its time position from maximum acceleration towards maximum velocity.

3. By comparing the required torque curves with their corresponding velocity trajectories, it is clear that a linear increment in maximum velocity causes a higher order increment in the centripetal and Coriolis force (Figures 4-6-e and 4-7-e). This is as expected from the second term of equation 2.2.

The response of the third link

Two main characteristics are demonstrated by the results shown in Figures 4-6-d and 4-7-d. These characteristics are :

1. There is always a constant required force added to the other velocity and acceleration dependent forces, i.e. the minimum required force is no longer zero. The physical interpretation for this non-zero minimum force is that since the link moves vertically in the gravitational field and all the preceding links have no vertical movement the gravitational force is constant.

2. Unlike the first two links, at the instance of zero acceleration (maximum velocity), the required torque remains zero and does not increase as the maximum velocity increases. This is because the third link is prismatic and so there are no velocity dependent centripetal and Coriolis forces. Since the link is under the effect of the inertial and gravitational forces only, the

Case	*Link*	*Applied Trajectory*
Case 1	1	Polynomial
	2	NC2
	3	Polynomial
Case 2	1	NC2
	2	Polynomial
	3	NC2

Table 4.1: Links' trajectories for case 1 and case 2 of dynamic coupling.

required force curves take the pattern of the acceleration curves with the gravitational forces added to them.

4.3.2 Second application

A robot's designer may consider applying different kinds of trajectories on different links to achieve different resultant end effector movements. In this case the study of the effect of dynamic coupling between links is important in forecasting how links will respond to each other's trajectories.

The SCARA robot has been examined in that respect in two ways as shown in Table 4.1 :

Firstly by applying polynomial trajectories on the first link, *NC2* trajectories on the second link and polynomial trajectories on the third link.

Secondly by applying *NC2* trajectories on the first link, polynomial trajectories on the second link and *NC2* trajectories on the third link.

The trajectories used are those shown in Figures 4-6-a and 4-7-a. The results obtained for these two cases, as shown in Figures 4-8 and 4-9, make the effect of coupling visible to the designer.

In the first case it is noticeable from Figure 4-8-a that the value of the required torque for the first link did not start at zero, although at this instant

the values of the first link variable, its first time derivative and its second time derivative are zeros (i.e. $q_{1_{t=0}} = \dot{q}_{1_{t=0}} = \ddot{q}_{1_{t=0}} = 0$). That was due to the interactive dynamics between the first and the second links, which becomes clear by comparing Figures 4-6-b and 4-8-a.

Also, by comparing the results shown in Figures 4-8 and 4-9 for the required torque of the first and second links with their corresponding results in Figures 4-6-b and 4-6-c and those in Figures 4-7-b and 4-7-c, it is found that :

1. The torque curve of a link does not necessarily follow the link's acceleration.
2. The value of the required torque is greater than zero at zero acceleration; this applies even with very low velocity trajectories.

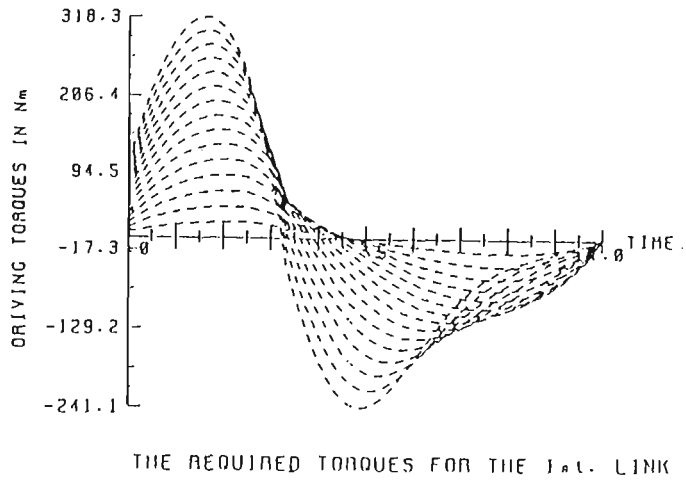
These changes in the required torque curves are due to the influence of the first link's trajectories on the second link and vice versa.

Also, Figures 4-8-c and 4-9-c show that the third link has not been affected by the dynamic coupling. This is as expected because the third link's motion and force are perpendicular to the plane in which the first two links move.

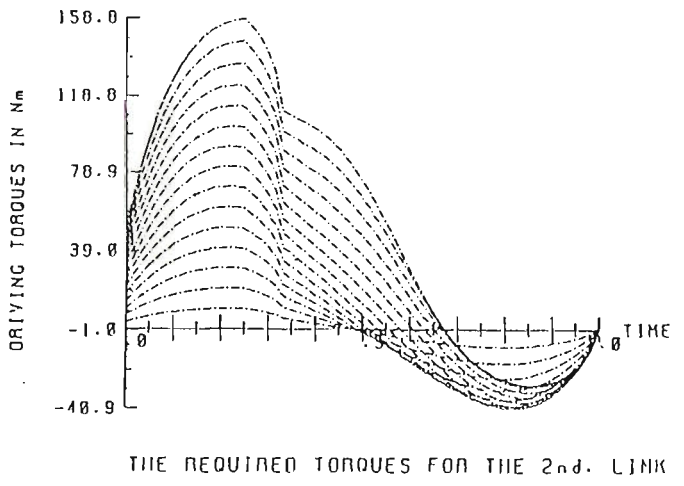
4.3.3 Third application

The dynamic behaviour of a robot's arm is not only a function of individual link velocity and acceleration but also a function of its position in space. For this reason it is important for a robot's designer to investigate the maximum required torque for each link in a robot's arm in the light of the expected operational paths which the links may take. Therefore, this chapter has been extended to include this investigation.

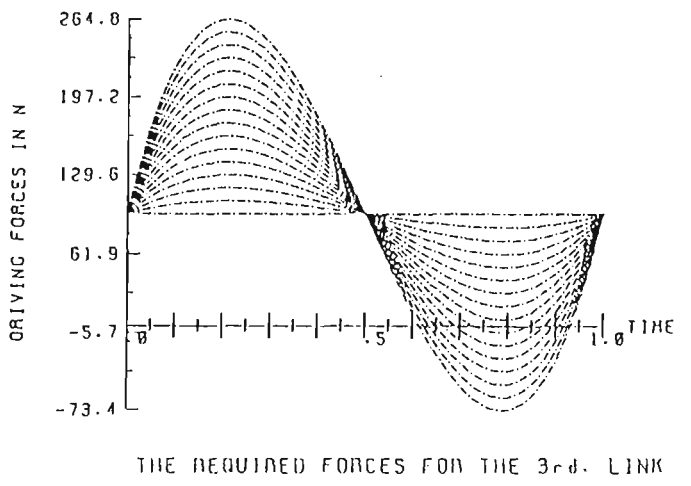
For consistency during this investigation a single velocity trajectory of polynomial type, with a maximum value of 3.75 rad/sec, was used for each link's



(a)

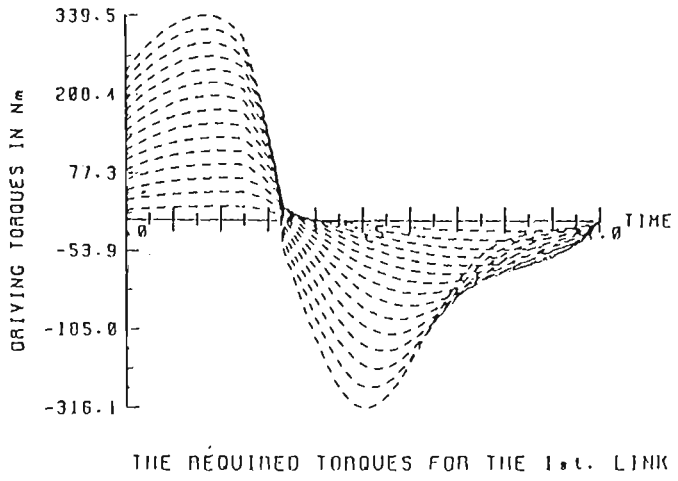


(b)

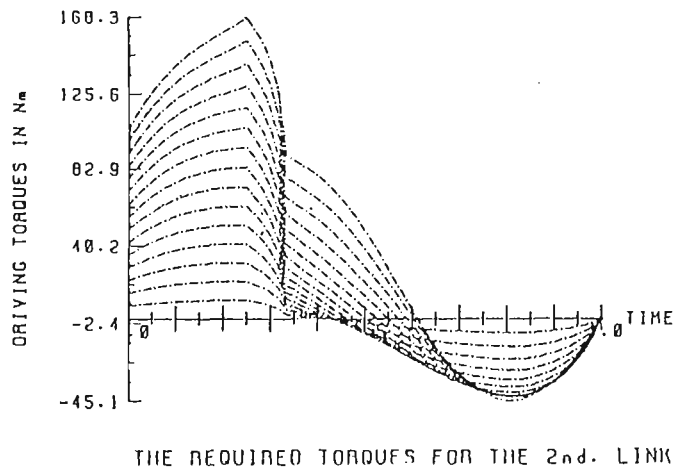


(c)

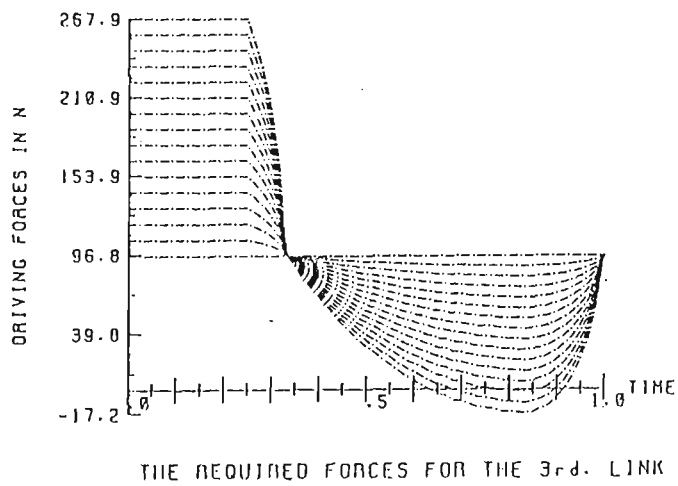
Figure 4-8: Effect of dynamic coupling (case 1).



(a)



(b)



(c)

Figure 4-9: Effect of dynamic coupling (case 2).

Case	*Link*	*Starting Position*	*Finishing Position*
Case 1	1	0.0	2 Rad.
	2	0.0	2 Rad.
Case 2	1	-1 Rad.	1 Rad.
	2	-1 Rad.	1 Rad.
Case 3	1	-1 Rad.	1 Rad.
	2	0.0	0.0

Table 4.2: Starting and finishing positions for the three cases.

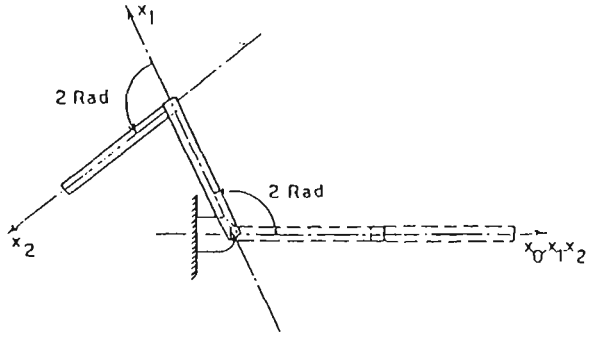
movement to give the same amount of displacement (2 radians). The starting and finishing positions of the links were varied as shown in Table 4.2.

Since it was shown in Section 4.3.2 that there is no dynamics coupling between the second and the third links, the investigation concentrated on the first two links only. For this reason the geometrical configuration data of the second link was modified to include the third link as a point mass at its end. Three of the cases examined during the course of this research will be discussed in this section. The results of the computer simulation are illustrated where Figures 4-10-a, 4-11-a and 4-12-a show the starting and finishing positions for the trajectories producing the torques shown in Figures 4-10-b, 4-11-b and 4-12-b respectively.

Based on these results the following analysis can be made:

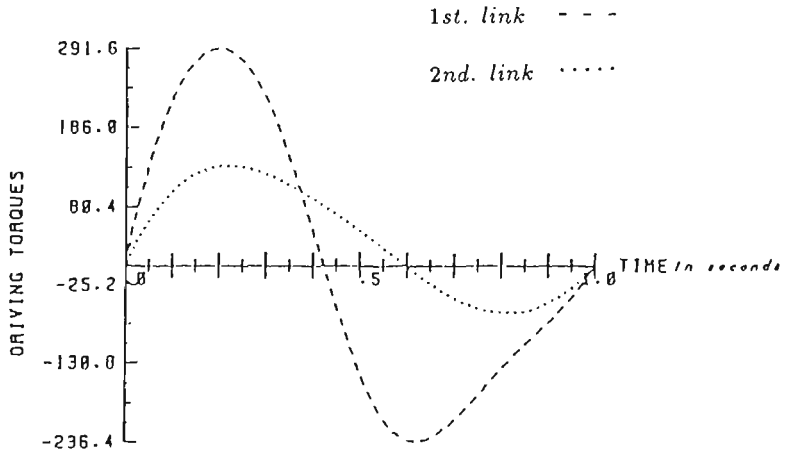
1. By comparing the results of the first and second cases, (Figures 4-10 and 4-11), one can notice the change in the maximum negative required torque. This change has occurred although both links have maintained the same velocity acceleration trajectories and they moved the same displacement (2 radians) in both cases. The only difference is the change in the starting and finishing positions.

2. Figure 4-11-b shows that for the start and finish positions in Figure 4-11-a there is no centripetal or Coriolis force when both q_1 and q_2 are equal to zero (at $t = 0.5$ sec). That is because the centripetal and Coriolis forces for both



Top view representation of both links motion

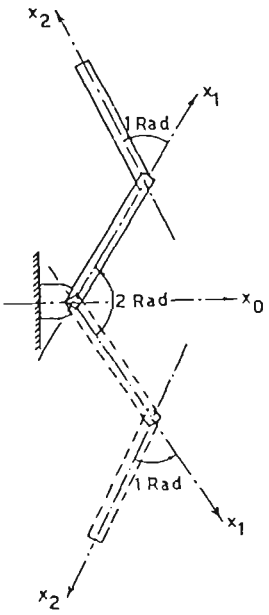
(a)



THE REQUIRED TORQUES FOR BOTH LINKS IN Nm

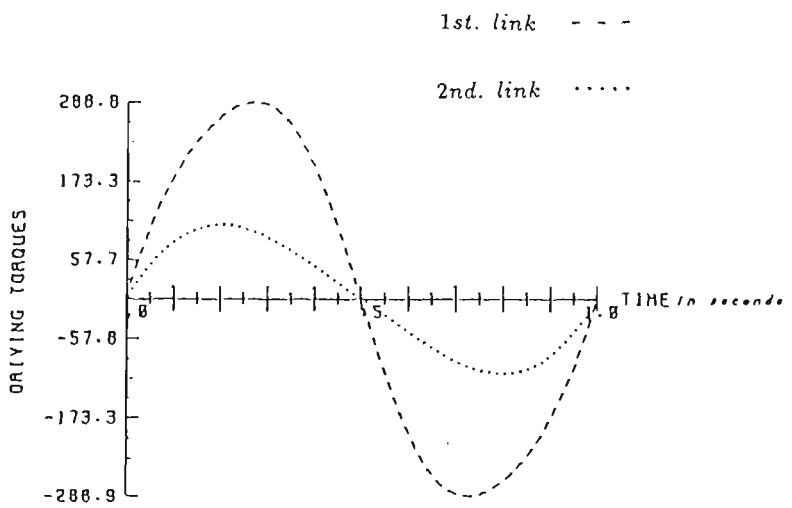
(b)

Figure 4-10: Kinetic effect of starting and finishing position (case 1).



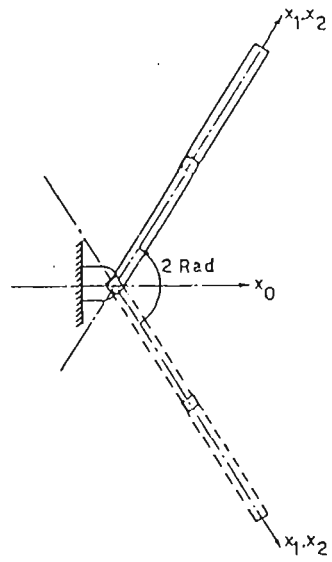
Top view representation of both links motion

(a)



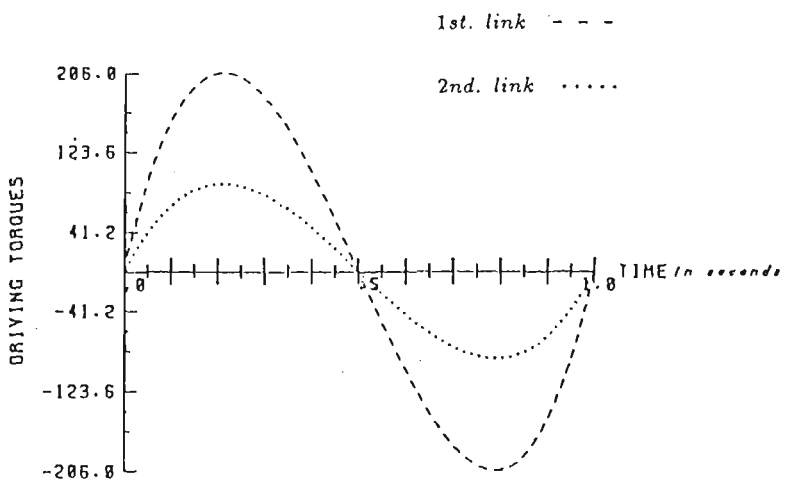
(b)

Figure 4-11: Kinetic effect of starting and finishing position (case 2).



Top view representation of both links motion

(a)



THE REQUIRED TORQUES FOR BOTH LINKS IN Nm

(b)

Figure 4-12: Kinetic effect of starting and finishing position (case 3).

links are functions of $(\sin q_2)$.

3. In the third case (Figure 4-12) the first link is the only moving link with the second link making a zero angle with the X axis of the first link coordinate frame. The comments on these results are :

   (a) Since the position variable of the second link remained equal to zero, there were no centripetal or Coriolis forces detected by either actuator.

   (b) Although the second link did not move relative to the first, its actuator is subjected to a torque due to the coupling discussed in 4.3.2

   (c) This torque sensed by the second link's actuator was in the same direction as that of the first link. That is expected since the inertia of both links are acting in the same direction.

4.4 Summary

In summary, this chapter presents an analysis of the SCARA robot's dynamic performance. The analysis was conducted and based on a specially developed computer package for the purpose of this research. It demonstrates that all terms of the complex, coupled and highly non-linear robot's dynamic must be considered when analysing high performance manipulators.

The numerical results obtained from this analytical study on the typical SCARA robot, which are represented in graphical forms, have given more quantitative insight into the dynamic behaviour of this type of robots [39]*. For example, it was shown from the cases studied that the non-linear velocity-related centripetal forces of the first link are three times that of the second link throughout the work-cycle. It was also shown that the effect of the velocity loading cannot be ignored

*[39] Ibrahim *et al.* Dynamic behaviour of a SCARA robot with links subjected to different velocity trajectories. *Robotica*, 6:115–121, 1988.

for a velocity beyond 1.12 rad/sec and 1 rad/sec for the first and second links respectively. This is particularly important if it is intended to omit these forces for on-line control purposes.

The analysis also enables the designer to quantitatively visualise the effect of coupling at every instant of the trajectory's time. In that context it was shown that a reduction in the maximum required torque for a link occurred when the neighboring links moved in different velocity trajectories with different points of maximum velocity. The reduction was 4.3% for link 1 (by comparing Figures 4-6-b and 4-8-a) and 8.7% for link 2 (from Figures 4-6-c and 4-7-c). However, it was found that the difference was too small to be of real significance taking into account the very high maximum velocity .

Chapter 5

Dynamic Characteristics of a SCARA Robot Under Different Payloads and Time-varying Payloads

5.1 Introduction

Robot manipulators are often required to perform tasks which entail different time-invariant payloads for different workcycles or a time-varying payload within a workcycle. This load variation usually affects the dynamic behaviour of a robot's arm. On the other hand, an industrial robot is a serial mechanism. Its dynamics are strongly coupled and highly nonlinear. Therefore, it is difficult to predict the effect of payload variation on the required torque/force for each link. Hence, in this thesis the research on the dynamic performance of robot manipulators has been extended to also cover the difficulties of predicting this effect.

The analysis was conducted using a SCARA robot type configuration, Figure 4-1, and an articulated robot (PUMA 560). Also in this phase of this research two velocity trajectories were used :

- NC2.
- Polynomial trajectories.

The conditions of the payload variation that were considered for this study were as follows :

1. different time-invariant payloads for different workcycles, such as in sorting and pick & place jobs.
2. a time-varying payload within a workcycle, such as in looming and welding processes.

Also in this investigation the dynamic simulation was based on the complete Lagrangian formulation, using the complete reverse dynamic model including the centripetal/Coriolis forces. The study was conducted off-line, hence the computational time has no critical importance.

Given the changes in payload characteristics, the mass of the last link (m_3), its pseudo-inertia tensor (J_3), and its center of masses ($\vec{r}_3$) had to be recalculated and updated at each instant of time within a workcycle. The effect of payload variation on the dynamic behaviour of each link was examined as outlined in the following analysis.

5.2 Effect of Different Time-invariant Payloads on a SCARA Robot Subject to NC2 Velocity Trajectories

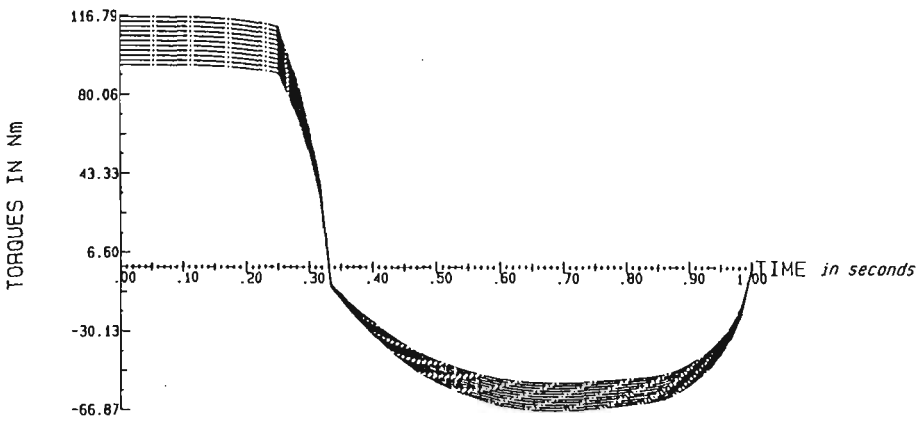
To study the effect of different time-invariant payloads on the dynamic behaviour of each link, a payload was represented by a spherical mass attached to the gripper. The required torque for each link was calculated at each point in time within an NC2 velocity trajectory. The same procedure was repeated for 11 workcycles with loads increasing in steps of 0.25 Kg so that for the first cycle the load was 2.5 Kg and for the eleventh cycle the load was 5 Kg.

5.2.1 The effect on the first link

For the *NC2* applied trajectory, at a maximum velocity point in time (0.33 sec.) the acceleration is zero. Therefore, the only torque felt by the actuator at this instant is due to the velocity-related centripetal and Coriolis forces, as shown in Figure 5-1. For this reason, the required torque at 0.33 sec. had a non zero value due to the effect of velocity. However, the effect of velocity on the dynamic behaviour of this robot is explained in Chapter 4.

It was also noticeable that the required torque for the first link was affected linearly by the increment in payload due to the inertia force. This is shown in Figure 5-1 at the maximum positive and maximum negative accelerations.

The required torque for the first link has also been affected linearly, due to centripetal/Coriolis force, by the change of payload in the robot's end-effector. Due to the geometrical configuration of the SCARA robot, the actuator of the first link does not sense any gravitational force. This is because the first link moves in



THE INERTIAL AND CENTRIPETAL TORQUES FOR LINK NO. 1
for
DIFFERENT PAYLOADS

Figure 5-1: Required torques for the first link under different time-invariant payload conditions.

a horizontal plane perpendicular to the gravity field, Figure 5-1.

5.2.2 The effect on the second link

In a similar way to the first link, the required torque of the second link is affected by the change in payload due to inertial and centripetal/Coriolis forces. The reason is that while the centripetal/Coriolis force is negative in the first link, it is positive in the second link, as shown at the zero acceleration points in Figures 5-1 and 5-2. The required torque of the second link has also been affected linearly by a change in payload, due to inertial force. It was found that while the percentage change in the maximum required torque for the first link was 23.25%, it was 27.47% for the second link. The small difference (4.22%) was due to the following :

1. the maximum required torque was dominated by the inertia force, and
2. the payload variation acted as a change in the second link's mass, while the first link's mass remained constant, thus the percentage change of the moving masses for the actuator of the second link is greater than that of the first link.

The second link also moves in a horizontal plane perpendicular to the gravity field. Thus, its actuator does not sense any gravitational force, Figure 5-2.

5.2.3 The effect on the third link

The third link in a SCARA robot is unlike the first two. It is a prismatic joint and moves vertically. Therefore, its actuator does not sense the centripetal/Coriolis force. Also, since a change in the kinetic energy of a moving link is a function of the acting forces [62, 63] the change in the kinetic energy of the

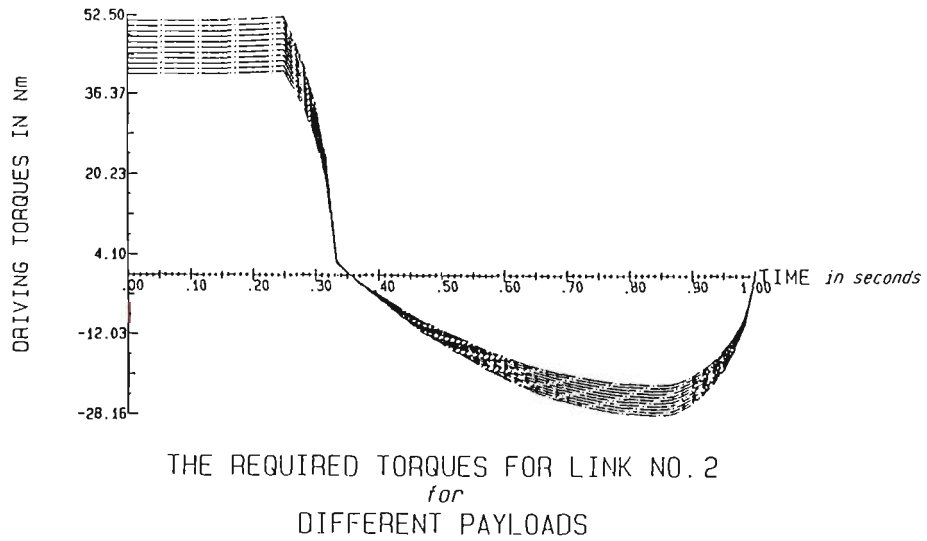


Figure 5-2: Required torques for the second link under different time-invariant payload conditions.

third link is dominated by the inertia force only. This is shown in the results obtained in Figure 5-3-a which shows that the kinetic forces cross over the zero line at the point of maximum velocity (zero acceleration). It is clear from the same figure that the inertia force has been affected linearly by the change in payload.

Since this link moves vertically, its actuator senses gravity forces. These gravity forces are proportional to the payloads, as shown in Figure 5-3-b. The influence of the payload changes on the total required force, according to the Lagrangian model (2.2), as shown in Figure 5-3-c.

The results plotted in Figure 5-4 show that the effect of a payload variation on the centripetal force is greater on the first link than on the second link, and is zero on the third link.

5.3 The Effect of a Time-varying Payload

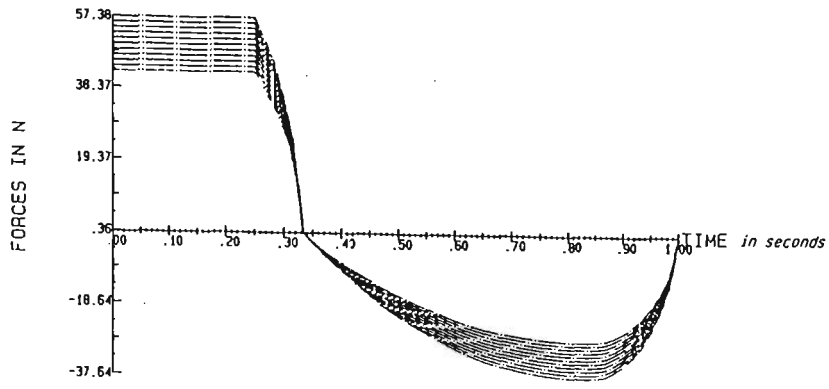
The effect of a time-varying payload was examined under two different types of variation:

1. discrete variation, and
2. continuous variation.

Discrete cases were studied for each kind of variation. For brevity, one case will be presented here for each type of variation.

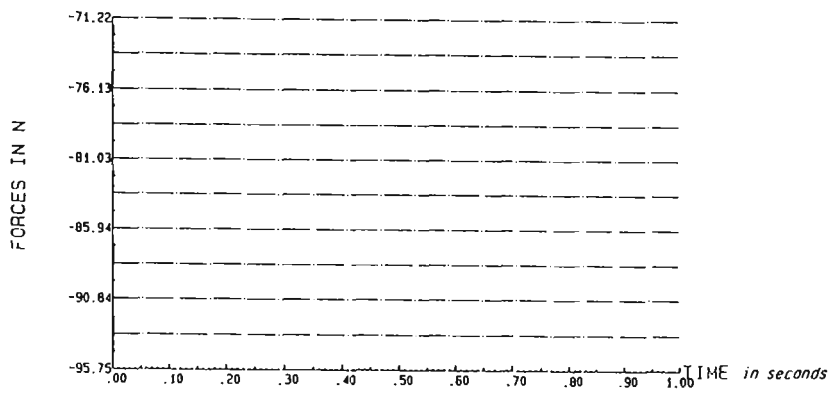
5.3.1 The effect of discrete variation

For technical analysis purposes, the interest was on the dynamic response for payload changes, irrespective of the pattern of changes. Therefore, to conveniently monitor that effect at certain instants of the cycle's time, payload changes



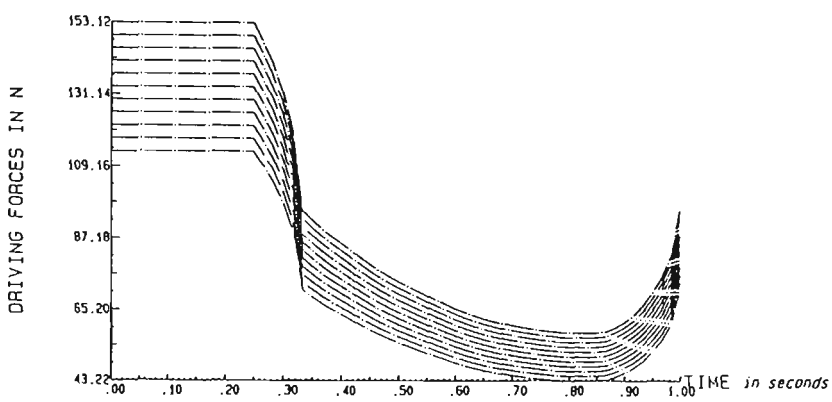
THE INERTIAL AND CENTRIPETAL FORCES FOR LINK NO. 3
for
DIFFERENT PAYLOADS

(a)



THE GRAVITATIONAL FORCES FOR LINK NO. 3
for
DIFFERENT PAYLOADS

(b)



THE REQUIRED FORCES FOR LINK NO. 3
for
DIFFERENT PAYLOADS

(c)

Figure 5-3: Inertial, gravitational and required forces for the third link under different time-invariant payload conditions.

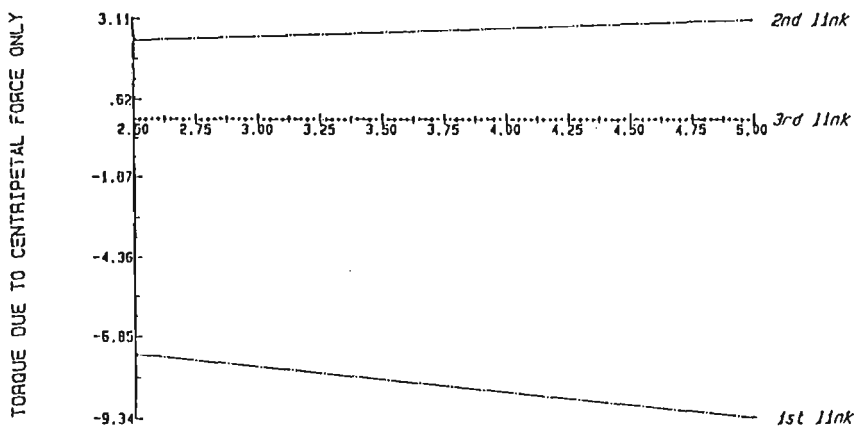


Figure 5-4: Torques due to centripetal forces versus payload.

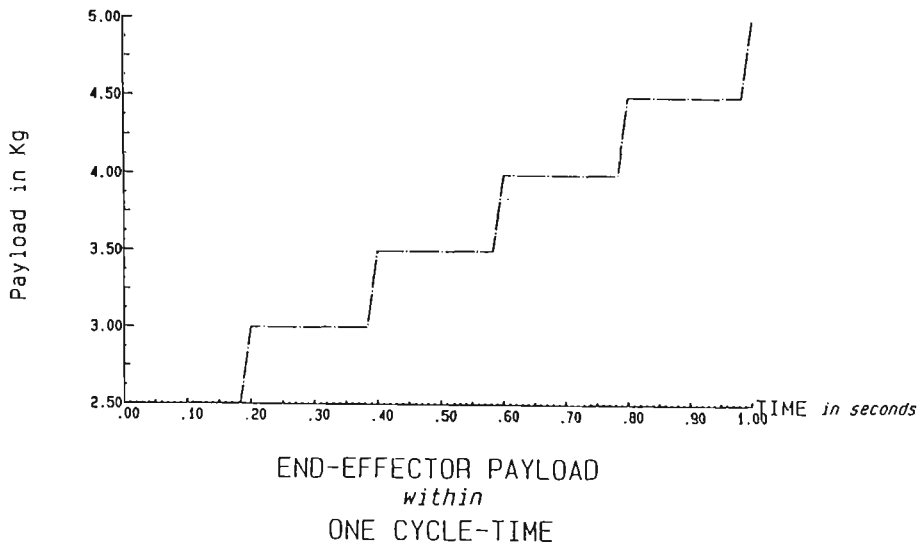


Figure 5-5: End-effector payload within one cycle-time.

were made discrete in time, and increments of payload were applied to obtain higher excitations.

In this case, the initial payload was set at 2.5 Kg at the beginning of the workcycle. An addition of 20% of the payload was made at equal intervals of 20% of the workcycle time. The payload at the end of the workcycle was double the payload at its beginning, i.e. 5 Kg, as shown in Figure 5-5.

The effect on the first link

The effect of load variation on the first link is shown by the successive increases in the required torques at the times of load addition, Figure 5-6. The figure shows that the additions of payload that took place at near maximum acceleration-time had a greater influence on the dynamic behaviour than those added at near maximum velocity-time. This agrees with the results discussed in Section 5.2.1.

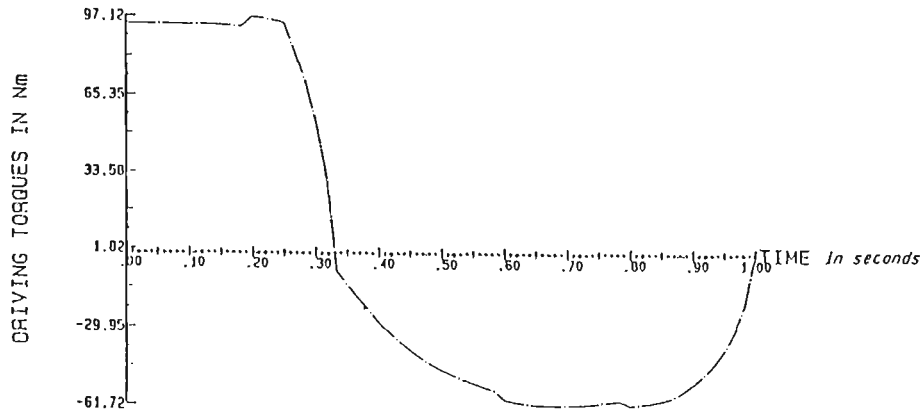


Figure 5-6: Required torque for link No. 1 for time-varying payload.

The effect on the second link

The effect on the second link, as shown in Figure 5-7, is similar in its pattern to that of the first link. In this case, the percentage change at the maximum acceleration point of the second link (7.4%) was greater than the effect of the payload change at the corresponding point of the first link (5.4%). That was due to the same reasons explained in Section 5.2.2.

The effect on the third link

Since this link moves prismatically, the kinetic forces are dominated by inertia force only. This inertia-resulting force is shown in Figure 5-8-a.

The effect of payload addition on the gravitational force component of the force for that link is shown in Figure 5-8-b. This was a result of the actuator moving

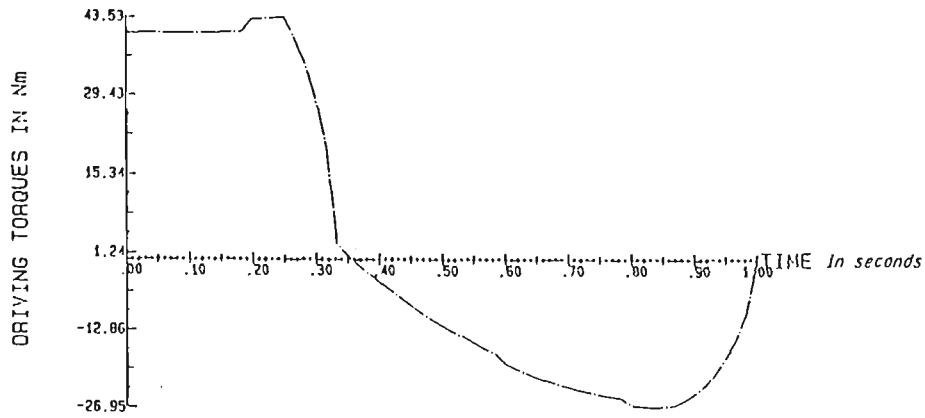


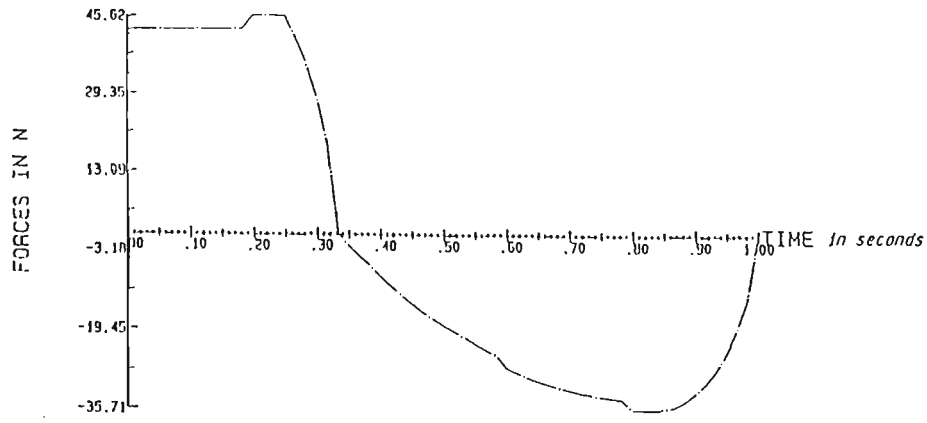
Figure 5-7: Required torque for link No. 2 for time-varying payload.

the link vertically and prismatically. The dynamic response of the gravitational force component was proportional to the payload addition within a workcycle. The effect of payload additions on the total required torque for the third link, i.e. on the gravitational and inertial forces, is plotted and shown in Figure 5-8-c. The stair-like curve in this figure shows the response of the actuator for the increments in payload with a zero velocity trajectory. The other curve represents the potential (gravitational) force subtracted from the kinetic (inertial) force.

5.3.2 Effect of a continuous time-varying payload

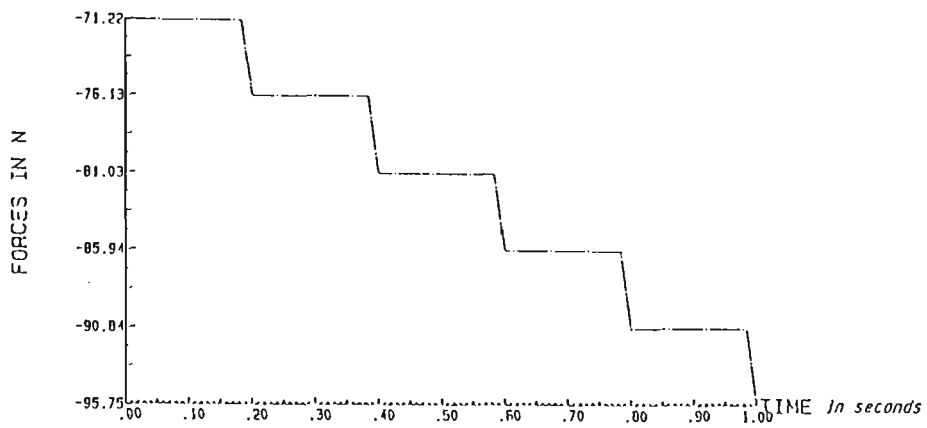
In this case, the initial payload was 2.5 Kg at the beginning of the work-cycle and increased linearly in time during the cycle. The payload reached its maximum at the end of the cycle to 5 Kg, Figure 5-9.

The response of the dynamic behaviour of the three links in this case was



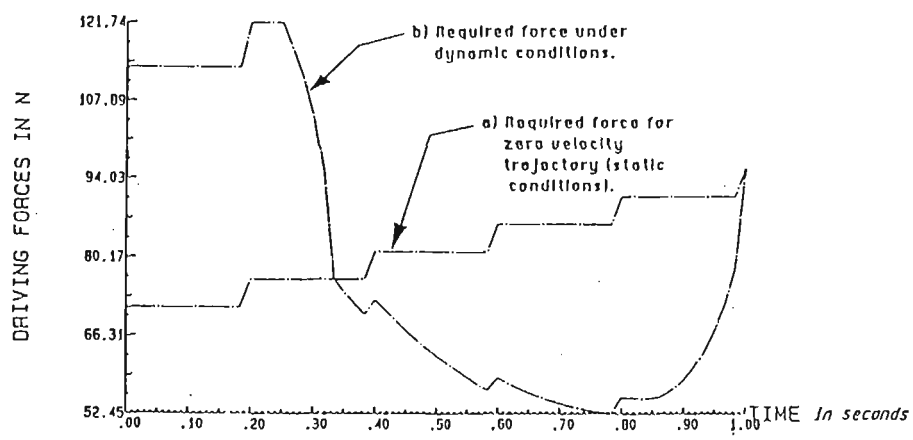
THE INERTIAL FORCES FOR LINK NO. 3

(a)



THE GRAVITATIONAL FORCES FOR LINK NO. 3

(b)



THE REQUIRED FORCES FOR LINK NO. 3

(c)

Figure 5-8: Inertial, gravitational and required forces for link No. 3 for time-varying payload.

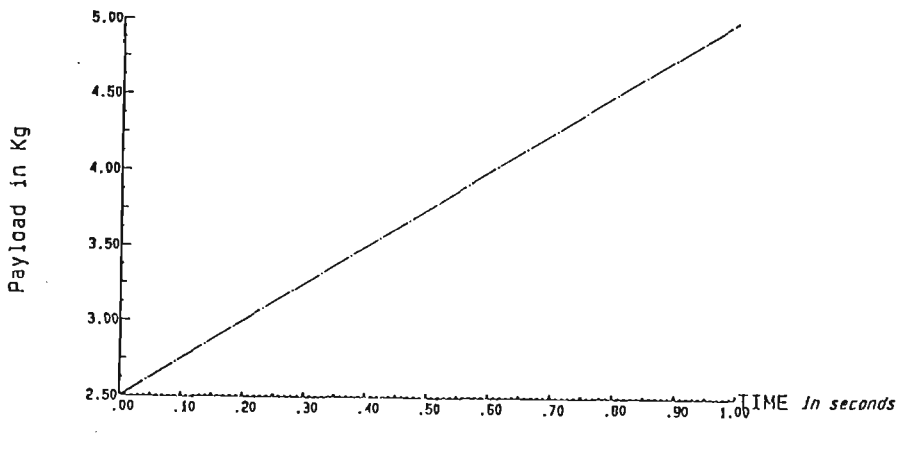


Figure 5-9: End-effector payload within one cycle-time.

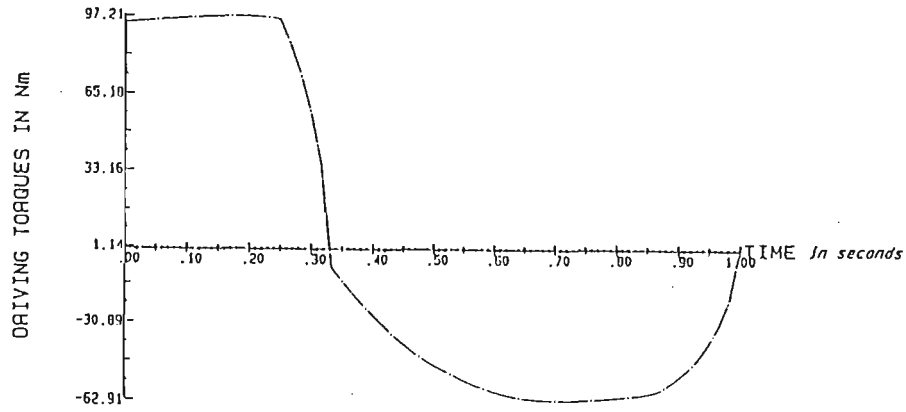


Figure 5-10: Required torques for link No. 1 for time-varying payload.

similar to the previous (discrete) case as shown in the results plotted in Figures 5-10 to 5-12. The dynamic response of the three links as shown in these figures was similar to the response in the case of discrete variation. That is, the effect was greater on the dynamic behaviour of the second link than on that of the first link.

5.4 Summary

Most of the work done on robotics to date is concerned either with aspects of control development or with the improvement of the kinematics and workspace. However, robot's dynamic characteristics has not received comparable attention in the literature. A software package was especially developed to analyse the dynamic behaviour of robot manipulators, under different operational conditions. Through this package it was possible to examine the dynamic behaviour of a robot's arm under different payload variation conditions, given the geometrical configuration and the velocity trajectory.

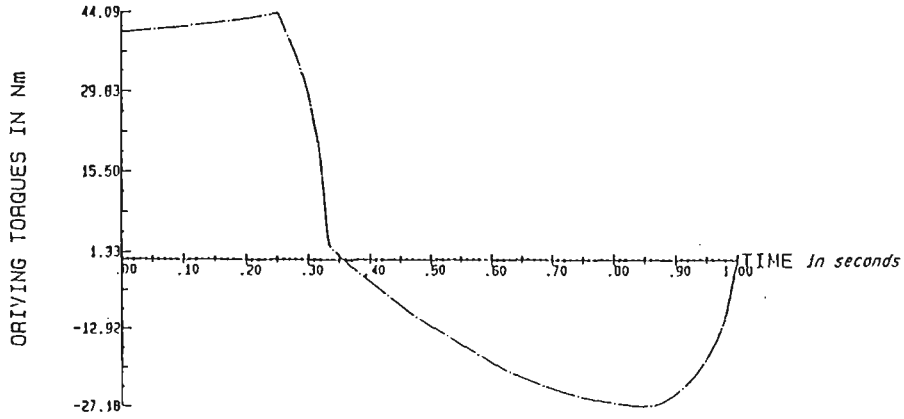


Figure 5-11: Required torques for link No. 2 for time-varying payload.

The results obtained for a robot of SCARA configuration with its links subjected to an NC2 velocity trajectory, which are summarised in [40]*, are briefly listed as follows:

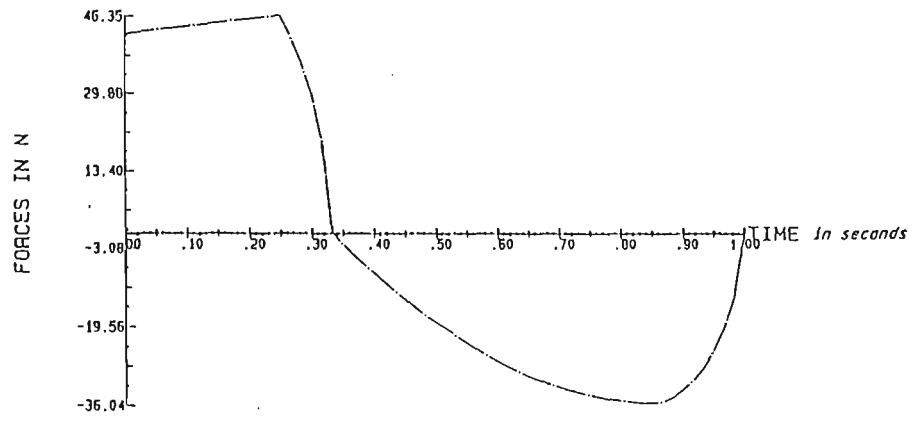
1. The changes in the required torque were linear in payload variation.

2. The percentage change of the required torque due to load variation was greater in the second link than in the first link.

3. The effect of payload variation on the required torque for a link, at near-maximum acceleration-time, was greater than the variation at near-maximum velocity-time.

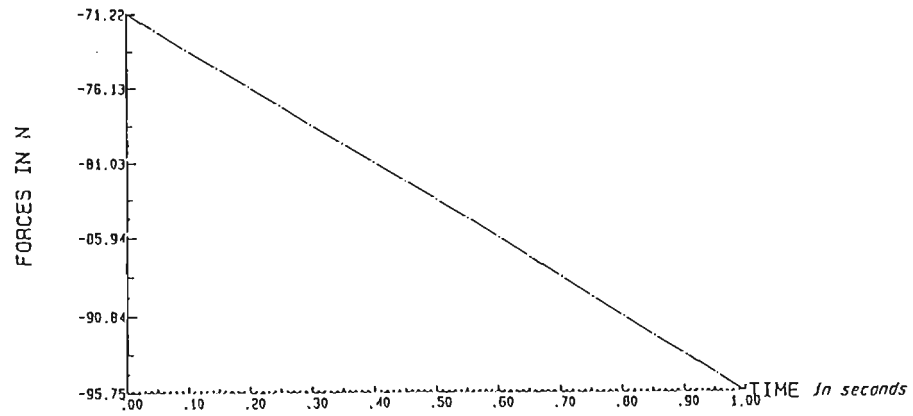
It was also found that the effect of payload changes at near-maximum acceleration-time and at near-maximum velocity-time are consistent irrespective of

*[40] Ibrahim et al. Dynamic Characteristics of a SCARA Robot Subject to NC2 Velocity Trajectories With Different Payloads. *Robotics & Computer-Integrated Manufacturing*, 6(3):259–264, 1989.



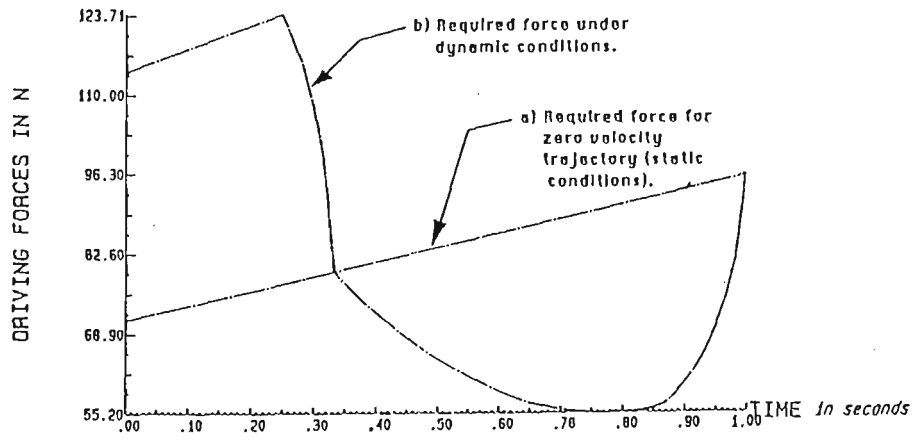
THE INERTIAL FORCES FOR LINK NO. 3

(a)



THE GRAVITATIONAL FORCES FOR LINK NO. 3

(b)



THE REQUIRED FORCES FOR LINK NO. 3

(c)

Figure 5-12: Inertial, gravitational and required forces for link No. 3 for time-varying payload.

the velocity trajectory used. This will be confirmed in Chapter 6 where a polynomial trajectory is applied on the cartesian motion of a PUMA 560. As mentioned in Chapter 4, if the velocity increases beyond 1.12 rad/sec and 1 rad/sec for the first and second link respectively, the non-linear velocity loading will increasingly dominate the dynamic behaviour. However, typical up-to-date industrial robots operate within these limits.

Because robot dynamics are inherently highly nonlinear and strongly coupled it is difficult to predict or deduce the response of a robot's dynamics behaviour to a time-varying payload. However, it is possible through the work presented here to give a robot designer a visual quantitative feeling about the required capability of a link's actuator to cope with payload variations at certain points within the workcycle.

Other terms can be added to the mathematical model of the required torque. These terms could represent the viscous friction and the motor's inertia. However, these terms are highly dependent on the detailed structure of the robot under investigation and so were excluded from the dynamic model because this study concentrated on the generalised forces that are usually felt by robot actuators of any configuration.

Chapter 6

Dynamic Behaviour of an Articulated Robot with End-effector Moving in a Cartesian Polynomial Trajectory Under a Time-varying Payload Condition

6.1 Introduction

In order to reduce vibration of a manipulated object at certain points in the workcycle, the joints' velocity trajectories can be designed such that the resultant end-effector movement produces a prespecified spatial velocity trajectory. In this research a computer simulation experiment has been conducted to examine

Parameter	*Joint*		
	1	2	3
α	-90^o	0	90^o
a	0	0.432 m	0
d	0	0.149 m	0

Table 6.1: Geometrical parameters of the first three links of a "PUMA 560".

the behaviour of an articulated robot's arm under the following conditions :

- End-effector moves in a polynomial trajectory and a straight-line Cartesian path.
- The payload is a discrete-time-varying payload.

The study implements the kinematic-dynamic integrated model that was developed specially for this research. The analysis of the conditional environment, constraints and output is discussed below.

6.2 The Cartesian Path

The Cartesian path of the end-effector is assumed to follow a third order polynomial trajectory. This polynomial trajectory is based on equation (4.1). Also, the robot under investigation is considered to be a PUMA-like robot. The geometrical parameters of the robot's three links are shown in 6.1. Also, the initial and final positions of the end-effector as well as its initial and final orientations are shown in Table 6.2.

Position	*Cartesian coordinates*			*Orientation angles*		
	X	Y	Z	α	β	γ
Starting point	0.4	0.6	-0.4	134^o	55^o	-58^o
End point	-0.5	0.5	0.5	-149^o	-12^o	173^o

Table 6.2: Initial and final positions and orientations of the end-effector.

The length of the straight line end-effector's path $\vec{\Re}$ was calculated using equation (3.1), and found to be 1.276 m. After applying the polynomial trajectory on that Cartesian path, it was then discretised to contain 60 equal time intervals. The discretisation was carried out using equations (3.5 – 3.7). This time-based discretisation has resulted in three arrays of X, Y and Z coordinates. The coordinates' arrays corresponding to the initial and final points stated in Table 6.2 are shown in Figure 6-1.

Each resulting Cartesian point of this discretisation was then mapped, through the inverse kinematic model, to the joint space positions. That produced four sets of different feasible solutions for every discretised position as explained in Section 3.3. A sample of the feasible joint-space solutions is shown in Figure 6-2 for the initial and final points on the Cartesian path.

The solution that corresponds to the required geometrical configuration is chosen. In this simulation experiment the following configuration was employed :

Arm *Lefty*

&

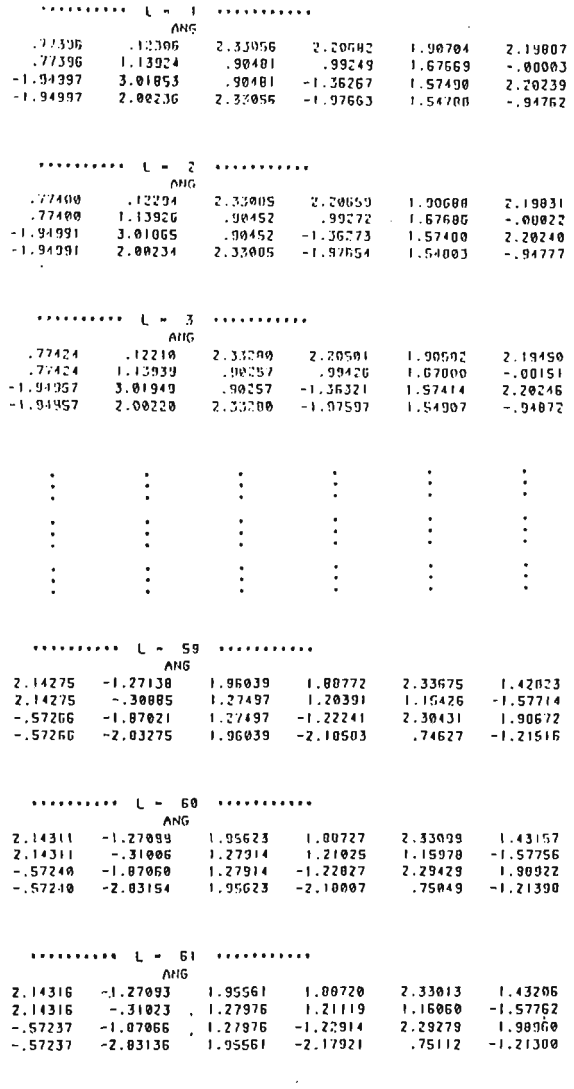
Elbow *Up*

This configuration resulted in the joints' angles $(\theta_1, \ldots, \theta_6)$ taking the values shown in Figure 6-3.

Since this research is focussing on the first three links; the values of the first three columns of Figure 6-3 $(\theta_1, \theta_2, \theta_3)$ were processed to obtain their corre-

```
Px = .400000000000000  Py= .600000000000000  Pz= -.400000000000000
Px = .399959368044074  Py= .599995405336967  Pz= -.399959368047712
Px = .399603111062041  Py= .599964790109516  Pz= -.399683111091144
Px = .398957687406801  Py= .599884187462158  Pz= -.398957687523216
Px = .397592888306826  Py= .599732543074060  Pz= -.397592888539657
Px = .395421005878598  Py= .599491222726647  Pz= -.395421006344259
Px = .392296000011265  Py= .599143999710213  Pz= -.392296000476927
Px = .388092658109963  Py= .598676961637102  Pz= -.388092659041286
Px = .382705773785710  Py= .598078418779187  Pz= -.382705775648356
Px = .376049307733774  Py= .597338811401278  Pz= -.376049311459065
Px = .368055550009012  Py= .596450615581125  Pz= -.368055553734303
Px = .358674296736717  Py= .595408253744245  Pz= -.358674300462009
Px = .347871994227171  Py= .594207997806370  Pz= -.347872001677752
Px = .335630947351456  Py= .592847081279806  Pz= -.335630954802036
Px = .321948447823524  Py= .591327602416277  Pz= -.321948455274105
Px = .306835936009884  Py= .589648434333503  Pz= -.306835943460464
Px = .290318192541599  Py= .587813129089773  Pz= -.290318207442760
Px = .272432574629784  Py= .585825837589800  Pz= -.272432589530945
Px = .253227972984314  Py= .583691992238164  Pz= -.253227987805475
Px = .232764253020287  Py= .581418244540691  Pz= -.232764253020287
Px = .211111107468605  Py= .579012338444591  Pz= -.211111122369766
Px = .188347586989403  Py= .576483058184305  Pz= -.188347601890564
Px = .164560878276825  Py= .573840089514852  Pz= -.164560908079147
Px = .139845901727676  Py= .571093978732824  Pz= -.139845901727676
Px = .114304000139236  Py= .568255989253521  Pz= -.114304029941559
Px = 8.804252147674566-002  Py= .565330047593032  Pz= -8.804255127906806-002
Px = 6.117370128631596-002  Py= .562352622300386  Pz= -6.117373109863836-002
Px = 3.381421566009526-002  Py= .559312677383423  Pz= -3.381424546241766-002
Px = 6.083542108535796-003  Py= .556231493502855  Pz= -6.083571910058186-003
Px = -2.189577221870426-002  Py= .553122679144144  Pz= 2.189574241630186-002
Px = -5.000001780139346-002  Py= .549999984353781  Pz= 4.999998807907106-002
Px = -7.810420393943786-002  Py= .546877293280708  Pz= 7.810417413711556-002
Px = -.106083607673645  Py= .543768471479416  Pz= .106083607673645
Px = -.133814251422882  Py= .540687287598048  Pz= .133814191818237
Px = -.161173856258392  Py= .537647331506014  Pz= .161173796653748
Px = -.188042497634888  Py= .534661924839020  Pz= .188042497634888
Px = -.214304065704346  Py= .531743972003460  Pz= .214304006099701
Px = -.239045967292706  Py= .528905978798866  Pz= .239045907688141
Px = -.264560854434967  Py= .526159881055355  Pz= .264560794830322
Px = -.288347578048706  Py= .523516914248466  Pz= .288347518444061
Px = -.311111247539520  Py= .520987620949745  Pz= .311111187934875
Px = -.332764303684235  Py= .518581724166070  Pz= .332764244079590
Px = -.353228068351746  Py= .516307963430881  Pz= .353228000747101
Px = -.372432625293732  Py= .514174124598503  Pz= .372432565689007
Px = -.390318131446838  Py= .512186846137047  Pz= .390318071842194
Px = -.406035949420929  Py= .510351537160026  Pz= .406035889816284
Px = -.421948409000505  Py= .508672370016575  Pz= .421948349475861
Px = -.435631012916565  Py= .507152079045773  Pz= .435630953311920
Px = -.447872018814087  Py= .505791969107223  Pz= .447871959209442
Px = -.458674347400665  Py= .504591709375381  Pz= .458674287796020
Px = -.468055522441864  Py= .503549358248711  Pz= .468055403232574
Px = -.476049200166626  Py= .502661159634590  Pz= .476049220561981
Px = -.482705867290497  Py= .501921540498733  Pz= .482705807605852
Px = -.488092015076007  Py= .501322990655899  Pz= .488092756271362
Px = -.492296075020923  Py= .500855965912342  Pz= .492296016216270
Px = -.495421087741852  Py= .500508739054203  Pz= .495420968532562
Px = -.497592961788177  Py= .500267422199249  Pz= .497592842578808
Px = -.498957729339600  Py= .500115773081779  Pz= .498957669734955
Px = -.499683117866516  Py= .500035180151463  Pz= .499683058261871
Px = -.499959385395050  Py= .500004483759403  Pz= .499959266185760
Px = -.500000035762787  Py= .499999968707561  Pz= .499999976158142
```

Figure 6-1: X, Y and Z coordinates of the Cartesian path for polynomial velocity trajectory fitting.



```
********** L = 1 **********
                    ANG
 .77396    .13396   2.33056   2.20642   1.90704   2.19807
 .77396   1.13924    .90481    .99249   1.67669   -.00003
-1.94997  3.01853    .90481  -1.36267   1.57490   2.20239
-1.94997  2.00236   2.33056  -1.97663   1.54708   -.94762

********** L = 2 **********
                    ANG
 .77400    .13204   2.33005   2.20659   1.90688   2.19831
 .77400   1.13926    .90452    .99272   1.67686   -.00022
-1.94991  3.01065    .90452  -1.36273   1.57400   2.20240
-1.94991  2.00234   2.33005  -1.97654   1.54803   -.94777

********** L = 3 **********
                    ANG
 .77424    .13210   2.33090   2.20581   1.90592   2.19450
 .77424   1.13939    .90257    .99426   1.67000   -.00151
-1.94957  3.01949    .90257  -1.36321   1.57414   2.20246
-1.94957  2.00220   2.33090  -1.97597   1.54907   -.94872

        :         :         :         :         :         :
        :         :         :         :         :         :
        :         :         :         :         :         :
        :         :         :         :         :         :
        :         :         :         :         :         :
        :         :         :         :         :         :

********** L = 59 **********
                    ANG
2.14275  -1.27138   1.96039   1.80772   2.33675   1.42823
2.14275   -.30885   1.27497   1.20391   1.15426  -1.57714
-.57266  -1.87021   1.27497  -1.22241   2.30431   1.90672
-.57266  -2.83275   1.96039  -2.10503    .74627  -1.21516

********** L = 60 **********
                    ANG
2.14311  -1.27089   1.95623   1.80727   2.33099   1.43157
2.14311   -.31006   1.27914   1.21025   1.15978  -1.57756
-.57240  -1.87060   1.27914  -1.22827   2.29429   1.90922
-.57240  -2.83154   1.95623  -2.10007    .75049  -1.21390

********** L = 61 **********
                    ANG
2.14316  -1.27093   1.95561   1.80720   2.33013   1.43206
2.14316   -.31023   1.27976   1.21119   1.16060  -1.57762
-.57237  -1.87066   1.27976  -1.22914   2.29279   1.90960
-.57237  -2.83136   1.95561  -2.17921    .75112  -1.21300
```

Figure 6-2: Four joint-space solutions for the Cartesian positions.

.77396	.12306	2.33056	2.20682	1.90704	2.19087
.77400	.12294	2.33005	2.20659	1.90688	2.19031
.77424	.12210	2.33280	2.20501	1.90582	2.19450
.77408	.11991	2.33790	2.20092	1.90302	2.18471
.77608	.11581	2.34739	2.19344	1.89774	2.16699
.77000	.10933	2.36222	2.18209	1.88929	2.14063
.76078	.10008	2.38304	2.16683	1.87703	2.10628
.78455	.08779	2.41011	2.14812	1.86030	2.06589
.78942	.07224	2.44340	2.12676	1.83840	2.02235
.79552	.05327	2.48262	2.10386	1.81049	1.97091
.80296	.03076	2.52726	2.08060	1.77549	1.93872
.81184	.00463	2.57667	2.05825	1.73190	1.90454
.82229	-.02525	2.63016	2.03811	1.67801	1.87070
.83440	-.05894	2.68696	2.02159	1.61101	1.86312
.84831	-.09656	2.74632	2.01038	1.52763	1.85917
.86413	-.13817	2.80750	2.00679	1.42384	1.86701
.88198	-.18381	2.86974	2.01439	1.29540	1.88377
.90199	-.23346	2.93231	2.03917	1.13946	1.89939
.92428	-.28690	2.99440	2.09224	.95789	1.88750
.94896	-.34412	3.05550	2.19560	.76213	1.77656
.97615	-.40442	3.11462	2.39327	.57005	1.31954
1.00594	-.46726	-3.11212	2.74943	.44973	.32018
1.03839	-.53178	-3.05915	-3.05850	.42936	-.14434
1.07356	-.59698	-3.01047	-2.69068	.51963	-.29464
1.11143	-.66175	-2.96691	-2.50545	.67637	-.35157
1.15194	-.72495	-2.92928	-2.44341	.87199	-.37266
1.19497	-.78556	-2.89837	-2.45031	1.09421	-.37137
1.24034	-.84270	-2.87487	-2.52708	1.32646	-.32115
1.28776	-.89577	-2.85938	-2.63725	1.54210	.01198
1.33689	-.94441	-2.85233	-2.70010	1.71615	1.35273
1.38733	-.98855	-2.85398	-2.94883	1.83031	1.72792
1.43861	-1.02827	-2.86435	-3.13955	1.91235	1.72043
1.49023	-1.06386	-2.88320	2.93004	1.94026	1.64446
1.54168	-1.09564	-2.91042	2.69323	1.95659	1.55330
1.59244	-1.12397	-2.94526	2.43000	1.94698	1.45749
1.64205	-1.14920	-2.98718	2.15390	1.92970	1.35238
1.69003	-1.17164	-3.03549	1.88484	1.91684	1.21067
1.73618	-1.19155	-3.08944	1.64696	1.91947	1.01665
1.78003	-1.20917	3.13486	1.45225	1.94317	.69992
1.82142	-1.22467	3.07170	1.30074	1.98782	.30122
1.86019	-1.23820	3.00518	1.18696	2.05041	-.03407
1.89623	-1.24990	2.93573	1.10512	2.12705	-.24089
1.92948	-1.25989	2.86405	1.05133	2.21362	-.35217
1.95996	-1.26825	2.79074	1.02428	2.30566	-.40473
1.98768	-1.27509	2.71637	1.02520	2.39813	-.41748
2.01270	-1.28040	2.64153	1.05751	2.48532	-.39403
2.03509	-1.28450	2.56678	1.12536	2.56091	-.32903
2.05496	-1.28722	2.49271	1.23003	2.61841	-.19936
2.07240	-1.28873	2.41997	1.36410	2.65234	.02499
2.08753	-1.28911	2.34924	1.50848	2.66022	.35197
2.10047	-1.28848	2.28130	1.63954	2.64401	.70400
2.11135	-1.28697	2.21704	1.74162	2.60926	.97720
2.12031	-1.28476	2.15750	1.81150	2.56282	1.15363
2.12750	-1.28207	2.10384	1.85400	2.51110	1.26265
2.13306	-1.27918	2.05734	1.87652	2.46000	1.33099
2.13717	-1.27641	2.01923	1.88621	2.41407	1.37473
2.14002	-1.27406	1.99042	1.88805	2.37715	1.40260
2.14180	-1.27237	1.97111	1.88856	2.35138	1.41942
2.14275	-1.27138	1.96039	1.88772	2.33675	1.42823
2.14311	-1.27099	1.95623	1.88727	2.33099	1.43157
2.14316	-1.27093	1.95561	1.88720	2.33013	1.43206

Figure 6-3: Joint-space solution corresponding to Arm-*Lefty* and Elbow-*Up*.

sponding velocity and acceleration. The velocity and acceleration were obtained using numerical differentiation methods [64]. To avoid error accumulation both velocity and acceleration were derived directly from the displacement values using the following equations :

$$v_t = \frac{\frac{1}{2}(q_{t+1} - q_{t-1})}{\Delta t} \tag{6.1}$$

$$a_t = \frac{q_{t-1} - 2q_t + q_{t-1}}{(\Delta t)^2} \tag{6.2}$$

Where :

$v_t \stackrel{def}{=}$ Velocity at time (t)
$a_t \stackrel{def}{=}$ Acceleration at time (t)
$q_t \stackrel{def}{=}$ Displacement at time (t)
$\Delta t \stackrel{def}{=}$ Time elapsed between two consecutive points in time

The sampling period of this simulated experiment was $\frac{1}{60}$ sec. Figures 6-4-a, 6-4-b and 6-4-c show the required displacement, velocity and acceleration of the first link in order to move the end effector in the desired polynomial velocity trajectory.

6.3 Dynamic Analysis of the First Link

The payload variation of the robot's hand was discrete. That was to enable examination of the robot's dynamic response to the load variation at certain points of the cycle's time. Therefore, at every 20% of the cycle time 20% of the payload was dropped. As shown in Figure 6-5, the initial payload was 2.5 Kg and the final payload was 0.0 Kg at the end of the cycle time (1 sec). Therefore there

was an equal decrement of the payload at 0.2, 0.4, 0.6, 0.8 and 1.0 sec.

The vertical component of the first link's motion was zero. Therefore, the gravitational force has no effect on the required torque for the first link. However, due to the inertial and Coriolis coupling (page 14), the change of the payload has influenced the kinetic-related torque of the first link. The effect of the velocity-related coupling of the second and third link is maximum when the arm is fully stretched horizontally. The torque on the first link due to the centripetal forces of the second and third links is maximum at these positions. Also, since the dynamic coupling is a result of the inertial interaction between links; the kinetic torque of the first link is sensitive to the direction and magnitude of both the first and second links' accelerations.

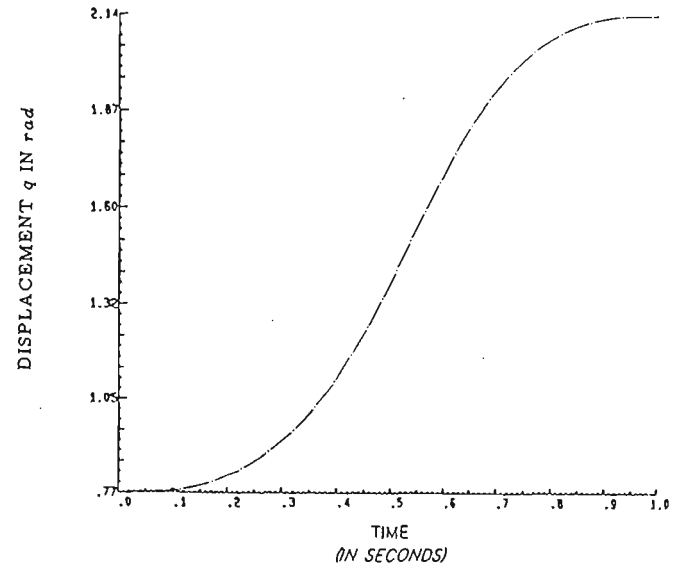
As a result of the above factors, the payload changes in the end-effector had the greatest influence on link 1 at 0.2 and 0.6 sec., as shown in Figure 6-6-b.

6.4 Dynamic Analysis of the Second Link

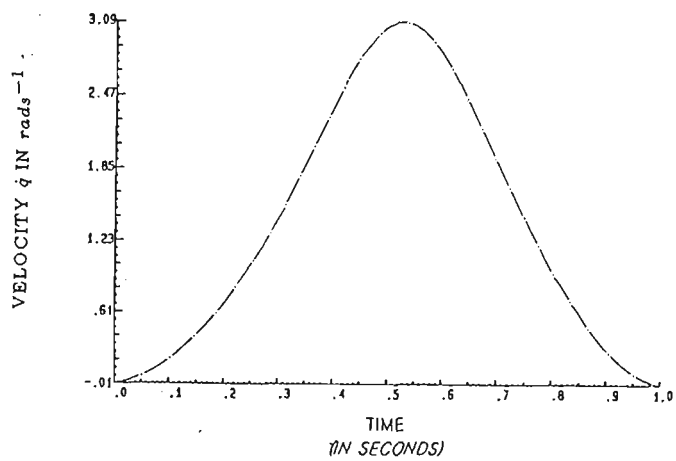
6.4.1 Second link's path trajectory

The desired end-effector path requires that the second joint's angular displacement moves in the negative direction (upward) and then retracts again. That is because the geometrical configuration is "$arm = lefty$" and "$elbow = up$". That was shown clearly on the output of the kinematic package, which is displayed in Figure 6-7-a. This figure shows that the change in the displacement direction occurs at 0.8 of the cycle's time. However, it was revealed as shown in Figure 6-7-b, that the maximum velocity of the second link occurs at 0.33 of the cycle's time. Also, the zero velocity of the same link occurs three times :

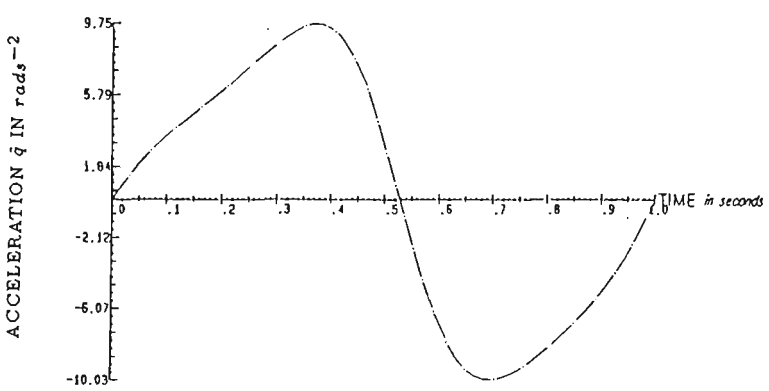
1. at the beginning of the cycle,



(a)



(b)



(c)

Figure 6-4: Displacement, velocity and acceleration of the first link.

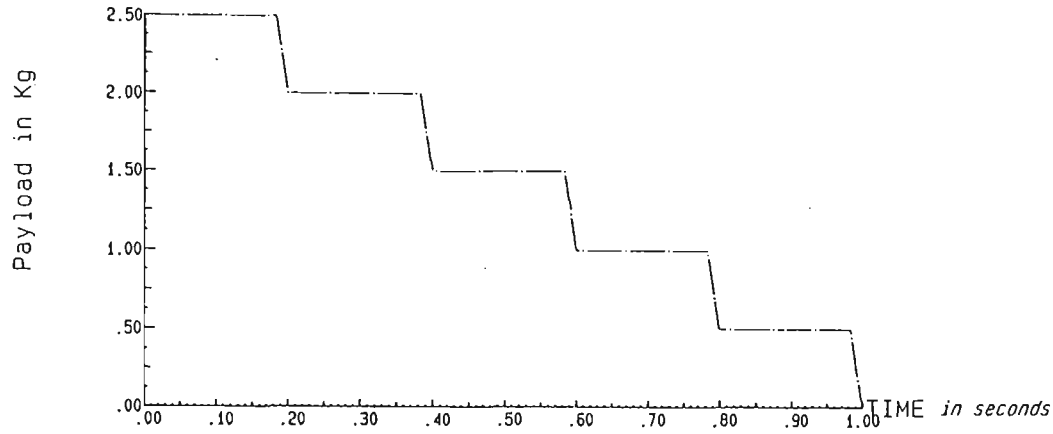


Figure 6-5: Payload variation within one cycle's time.

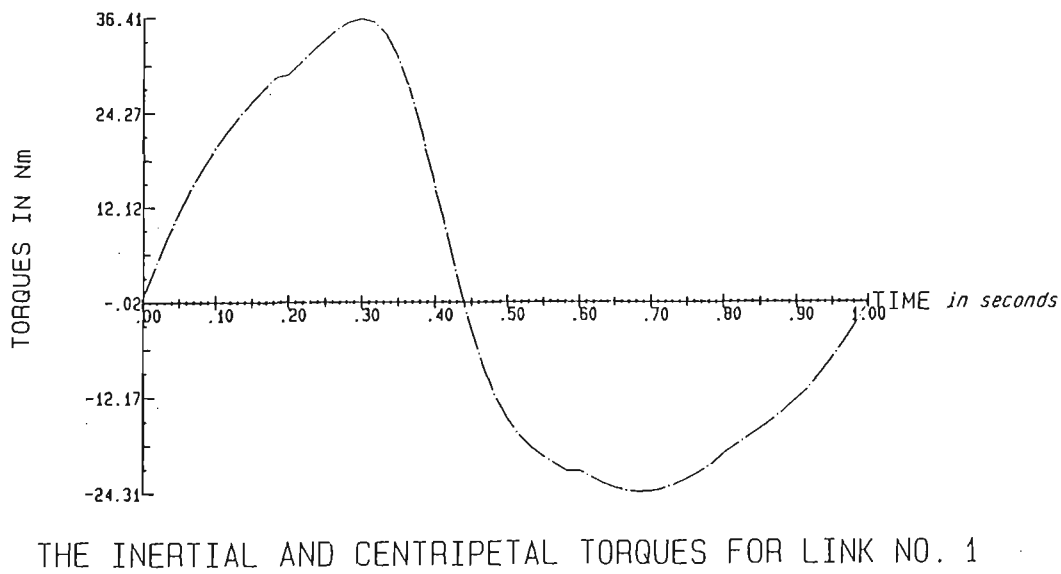


Figure 6-6: Dynamic effect on the first link.

2. at the end of the cycle and

3. at the point of changing displacement's direction.

6.4.2 Dynamic effect on the second link

The second link was moving upward. Hence, the gravity loading of the required torque was continuously reduced. That is physically interpreted by the continuous reduction in the centre of gravity's (C.G.) moment arm. Therefore, the gravitational loading component of the required torque for the link was more sensitive to the payload variation at the early stage of the workcycle. That is manifested through the computational result shown in Figure 6-8-a.

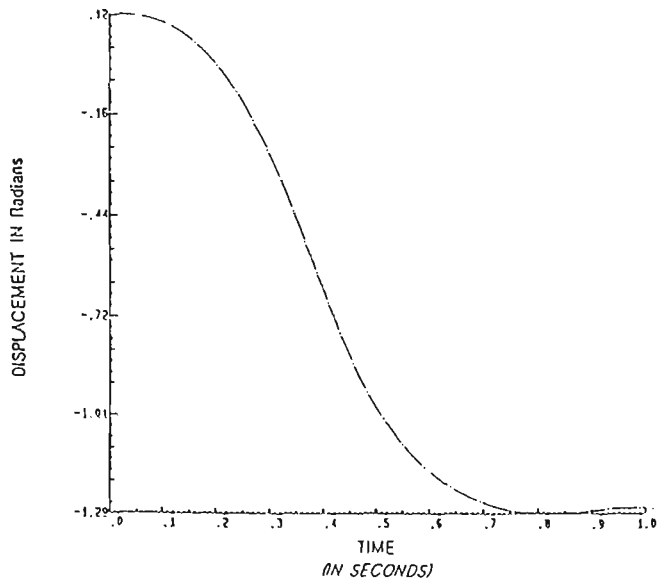
Also, the graphical output of the kinetic related torques of the second link has revealed the extent of dynamic coupling between the second and the third links. The payload changes at 0.2 sec. and 0.6 sec. have more kinetic influence on the link's dynamic behaviour than the changes at other times. The reason behind that is the *relatively* high acceleration at these points in time as found through the inverse kinematic analysis shown in Figure 6-7.

Also, the link has experienced a considerable kinetic-related torque at 0.9 sec. as shown in Figure 6-8-b. That was due to the dynamic coupling with the third link, since the latter has experienced its maximum acceleration at this point in time. The total required torque of link i at time t (τ_{i_t}) is obtained by subtracting the gravitational-related torque from the kinetic-related torque :

$$\tau_{i_t} = K_{i_t} - P_{i_t} \tag{6.3}$$

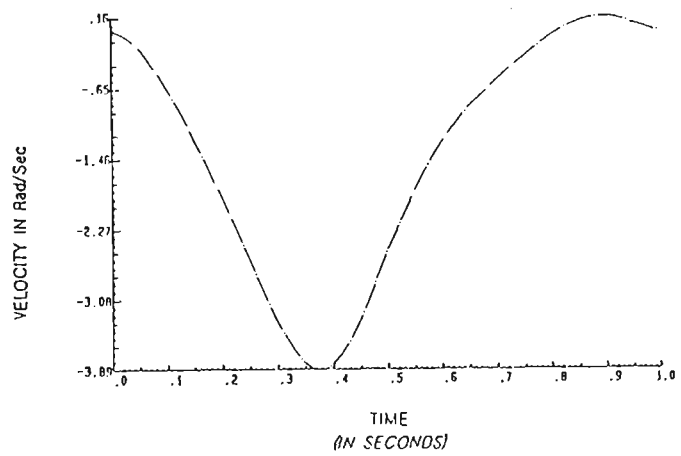
Where :

$\tau_{i_t} \stackrel{def}{=}$ The torque required for link i at time t.
$K_{i_t} \stackrel{def}{=}$ The kinetic energy of link i at link t.
$P_{i_t} \stackrel{def}{=}$ The potential energy of link i at time t.



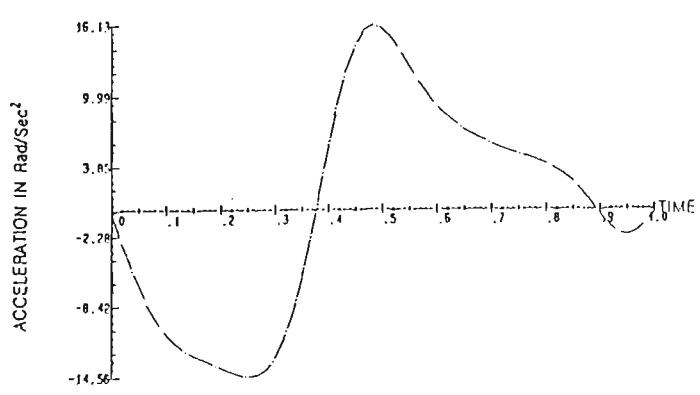
DISPLACEMENT TRAJECTORY OF LINK No. 2

(a)



VELOCITY TRAJECTORIES OF LINK NO. 2

(b)



ACCELERATION TRAJECTORY OF LINK No.2

(c)

Figure 6-7: Displacement, velocity and acceleration of the second link.

The total required torque for the second link is shown in Figure 6-8-c.

6.5 Dynamic Analysis of the Third Link

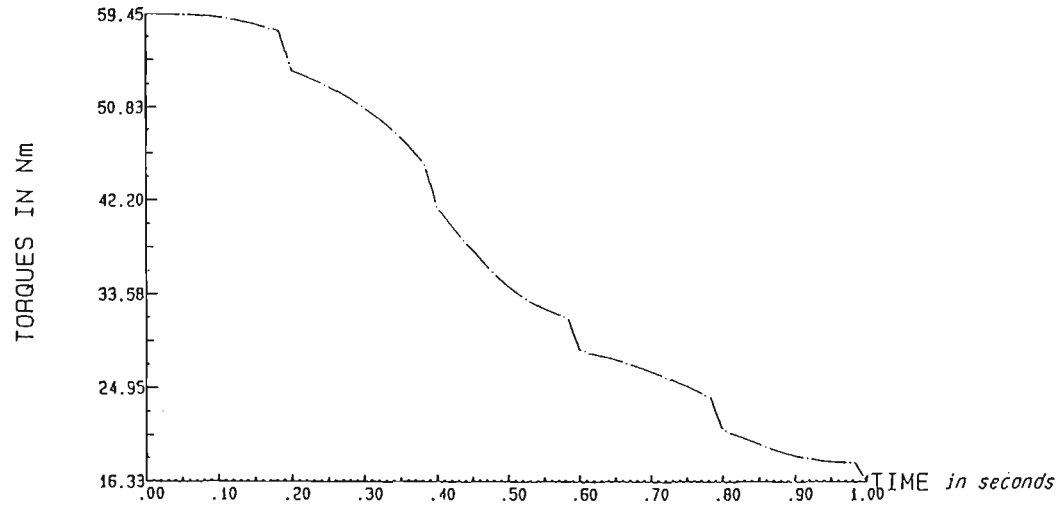
6.5.1 Third link's path trajectory

The third link displacement increases in the positive direction (downwards) during the first half of the cycle and retracts in the second half of the cycle, Figure 6-9-a. The software of the inverse kinematic gives a solution for q_i between $+\pi$ and $-\pi$. Therefore, if a displacement angle exceeds $+\pi$ (3.14 rad), the excess is expressed in the range $-\pi \Rightarrow 0$ as shown in Figure 6-9-a. Hence, the developed software had to accommodate these characteristics in the calculations of velocity and acceleration.

Also, in response to the end-effector polynomial trajectory, the velocity of the third link has two peaks, one positive and the other negative, as shown in Figure 6-9-b. The velocity trajectory behaves in a symmetrical way about the mid-cycle time where the velocity is zero.

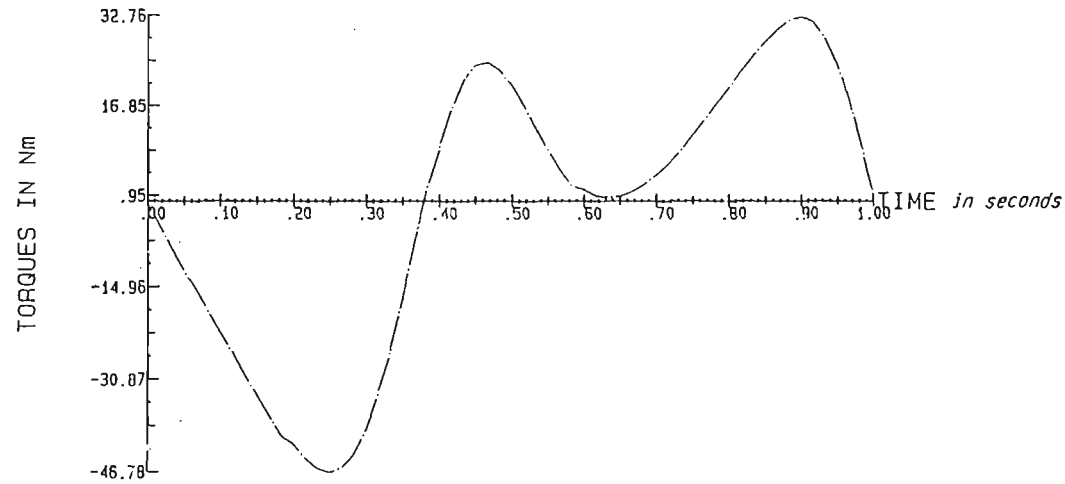
6.5.2 Dynamic effect on the third link

The gravity loading of the third link was dependent on its position with respect to a horizontal plane through the workcycle. The maximum sensitivity of the third link's gravitational loading to the payload changes was at 0.8 sec. Through the calculation of the instantaneous positions of both the first and second links at 0.8 sec; it was found that the third link was nearly horizontal at that moment in time. Thus its gravitational force had its maximum moment arm about its joint at that instant. This explains the gravity-related torque's sensitivity to



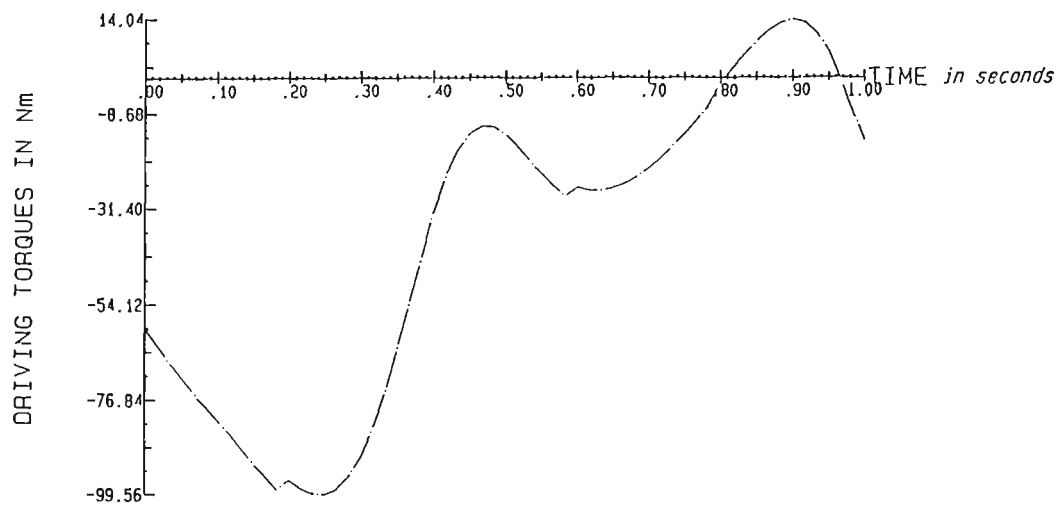
THE GRAVITATIONAL TORQUES FOR LINK NO. 2

(a)



THE INERTIAL AND CENTRIPETAL TORQUES FOR LINK NO. 2

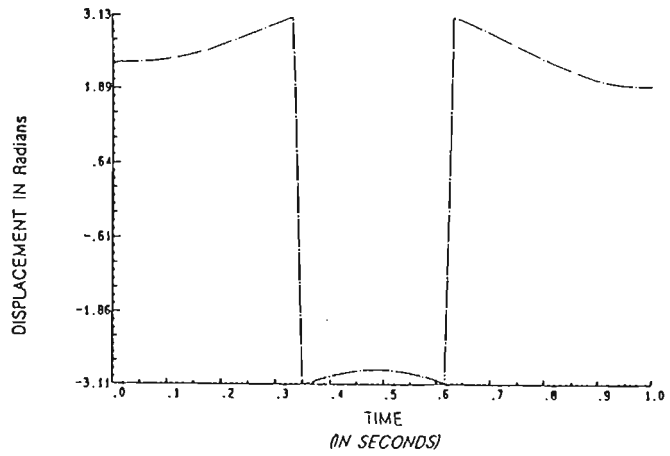
(b)



THE REQUIRED TORQUES FOR LINK NO. 2

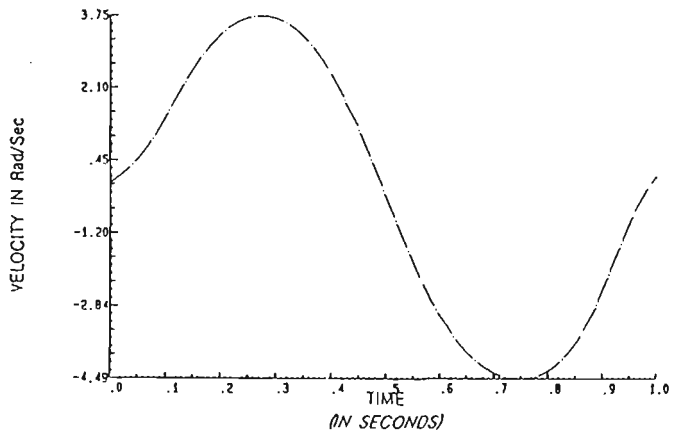
(c)

Figure 6-8: Gravitational, kinetic and total torque of the second link.



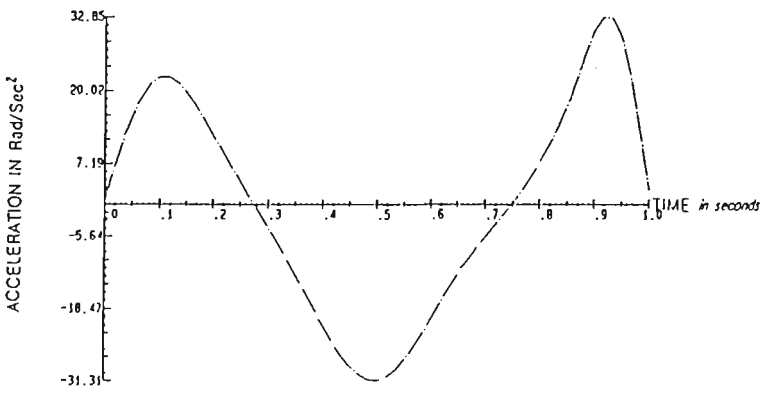
DISPLACEMENT TRAJECTORY OF LINK No.3

(a)



VELOCITY TRAJECTORIES OF LINK NO. 3

(b)



ACCELERATION TRAJECTORY OF LINK No.3

(c)

Figure 6-9: Displacement, velocity and acceleration of the third link.

the payload variation at that time.

The maximum step change in dynamic response, as a result of the payload variations, in the kinetic loading was at 0.4 sec., as shown in Figure 6-10-b. This is at near-maximum-acceleration point in time. It should be also noticed that at mid-cycle time (0.5 sec.), the second link exhibited the maximum positive acceleration while the third link exhibited the maximum negative acceleration.

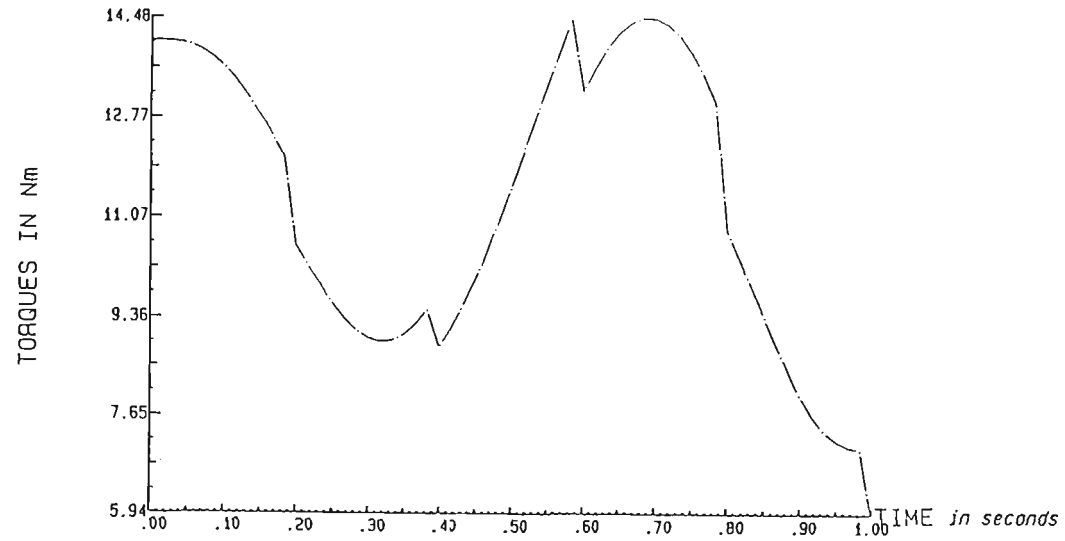
6.6 Summary

The dynamic behaviour analysis of an articulated type of robot, moving under particular operational conditions, was presented in this chapter. The robot was moving under a time-varying payload condition coupled with the polynomial velocity trajectory of the end-effector. Therefore, both the general dynamic model and the PUMA kinematic model were employed in this investigation. The displacement, velocity and acceleration trajectories of each link were examined and effect of payload variations on each link was also examined and analysed.

By cross examining both the kinematic and dynamic behaviour for each link it was found that the payload variation near maximum acceleration has more influence on the dynamic performance than at near maximum velocity. This was also true for an NC2 trajectory applied to a SCARA robot, as shown from the results obtained in Chapter 5.

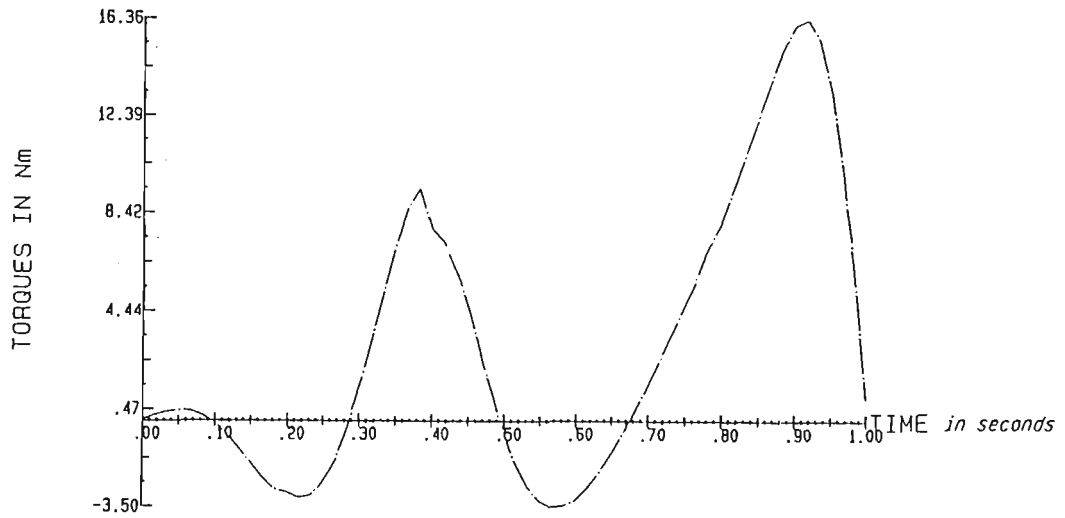
Although the maximum response to payload variation was limited to 9.6% of the total required torque, it is anticipated that the response will be more significant when robots become more capable of carrying payloads which are larger in proportion to their own weight.

The outcome of this analysis has proven to agree with work presented



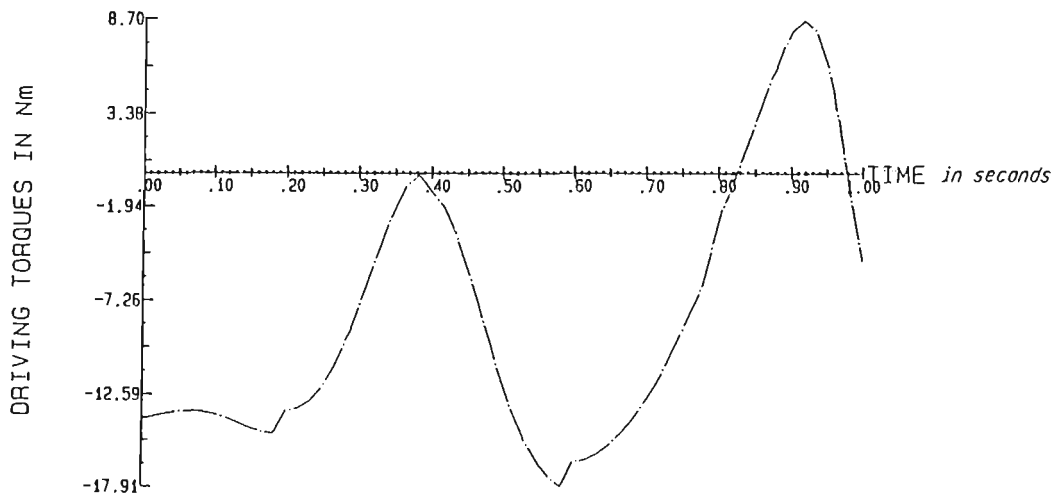
THE GRAVITATIONAL TORQUES FOR LINK NO. 3

(a)



THE INERTIAL AND CENTRIPETAL TORQUES FOR LINK NO. 3

(b)



THE REQUIRED TORQUES FOR LINK NO. 3

(c)

Figure 6-10: Gravitational, kinetic and total torque of the third link.

in Chapter 4 and summarised in [40]. Also, it enabled a better understanding of the dynamic behaviour of the robot's arm under the above-mentioned conditions. Furthermore, this analysis can help in the future design of a robot's Cartesian control.

Chapter 7

Experimental Approach to the Dynamic Analysis of an Articulated Robot Manipulator

The objective of this thesis is to analyse the *dynamic performance characteristics of robot manipulators* as the title suggests. In pursuit of this analytical study, an experimental approach was conducted to examine the dynamic behaviour of a PUMA 560 robot's arm. The experiments were conducted for different payload conditions. The results of the experiments were compared with those of the compatible conditions of the simulation study conducted in Chapter 6.

Also, the objective of the experiments was to determine the actual acceleration time percentage in a typical workcycle for a typical robot's arm. This information was required for the study related to links counter-balancing in the succeeding Chapter.

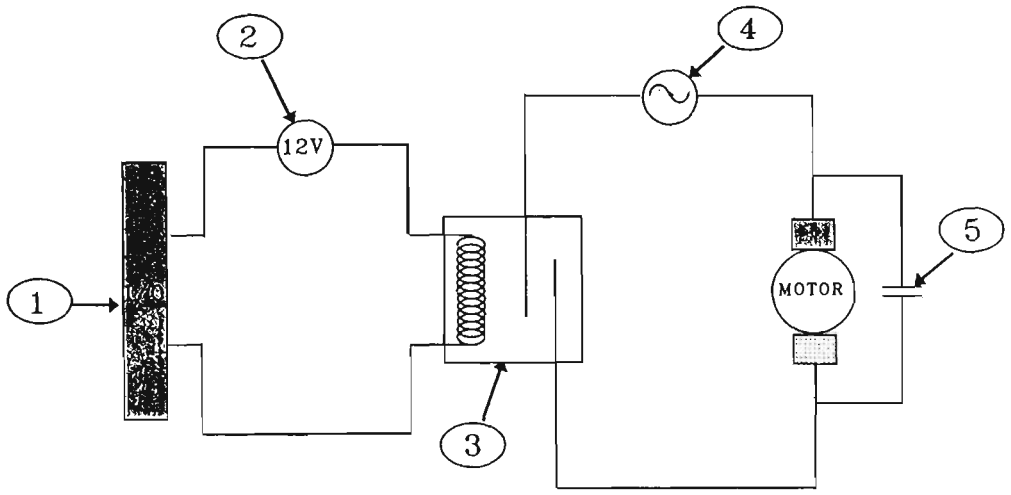
The experimental setup, conditions and results are discussed below.

7.1 Experimental Setup

Since the objective was to examine the robot's dynamic behaviour under discrete time-varying payload conditions; a payload ejection system was developed specifically for this purpose. The system in Figure 7-2 consists primarily of the following parts :

- A computer controlled AC-motor. This motor is controlled through the I/O interface module of the robot. A schematic diagram of the motor circuitry is shown in Figure 7-1.
- A flexible shaft to transfer the motion from the motor to the payload ejection device.
- A payload holding cylinder. Its main function is to hold payload parts and eject them at equal time intervals during the workcycle.
- Payload parts. These parts consist of four slices of a mild steel cylinder of equal weight. Since the maximum allowable payload for PUMA 560 is 2.5 Kg., each payload part is chosen to be 0.4 Kg. They were held through a threaded hole in the centre to the rotating screw in the centre of the payload ejection device.
- An AT-PC with a DAS16 data acquisition card connected to a universal screw terminal. The latter receives the required signals directly from the robot's controller.
- A soft body to catch the dropping payload parts (a mattress).

The payload ejection action is carried by the AC-motor which is activated by a signal received from the robot's controller via the robot's input/output module. The control instructions were written in the robot's " *VAL*" language [65].



1. Output signal from the robot's I/O module
2. 12V – DC supply
3. Relay (12V input / 240V output)
4. 240V AC supply
5. Capacitor for anti-clockwise turn

Figure 7-1: Payload ejection motor's circuit.

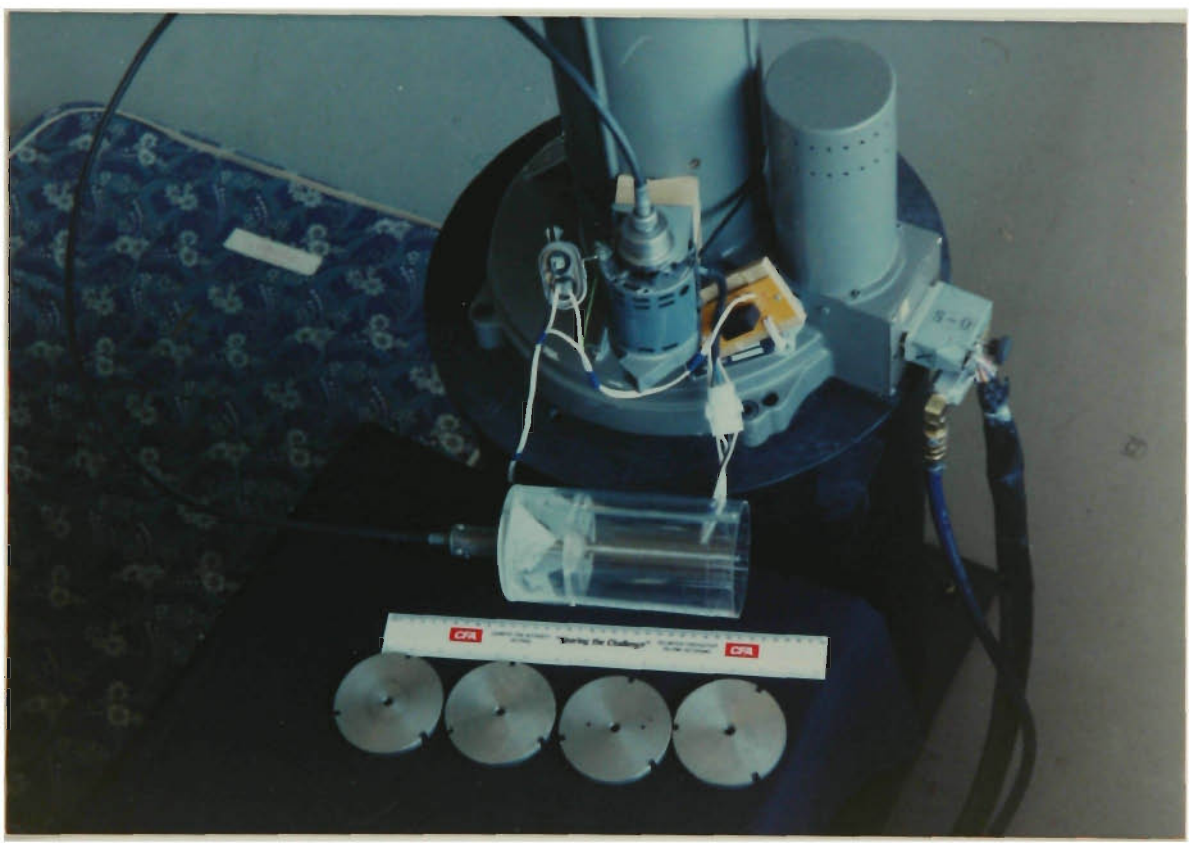


Figure 7-2: Payload ejection system.

Position	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Initial	-111.38^o	176.54^o	36.4^o	4.6^o	57.67^o	-207^o
Final	-41.93^o	-103.41^o	4.4^o	14.68^o	6.19^o	-21.97^o

Table 7.1: Initial and final joints' angles.

The orientation of the ejection screw axis with respect to the base coordinate frame was kept vertically down through the workcycle, as shown in Figure 7-3, for the following reasons :

1. To reduce the torsional load on joint 6.

2. To ease the payload-parts ejection.

The end-effector (ejection device) Cartesian path was similar to that used in the simulation study covered in Chapter 6. The joints angles of the initial and final positions are tabulated in Table 7.1.

7.2 Data Acquisition

The data was acquired through an AT personal computer. The sampling frequency and sampling duration were governed by the following equation :

$$f \times \Delta t = NR \tag{7.1}$$

Where :

$f \stackrel{def}{=}$ Sampling frequency
$\Delta t \stackrel{def}{=}$ Sampling period
$NR \stackrel{def}{=}$ Number of data records (the data file size)

The data file size limit was 8000 records, which is the maximum number of records that can be handled by "*Lotus 123*". Also, the required sampling time

Figure 7-3: Payload ejection device assembly.

for each channel was four seconds. Therefore, the sampling frequency per channel per second was 250.

Six data acquisition channels were used concurrently. Three channels were dedicated for the first three joints' encoder signals. The other three channels were assigned for the first three joint's motor current signals.

7.2.1 Data acquisition of displacement signals

In order to find both velocity and acceleration trajectories of the first three links of the PUMA robot, the motor's encoder signal was intercepted. The interception was through the encoder "0" on the analog servo-board of each joint. The location of the interception point on the robot's analog servo boards is shown in Figure 7-4. The displacement signal was written to data files through the data acquisition software "*LABTECH NOTEBOOK*" [66]. Subsequently, these data files were loaded to a HP-mainframe using the UNIX operating system for the required processing. The objectives of processing were as follows :

1. To filter the encoders signal from spikes and noise.

2. To count the pulses generated by the encoders.

3. To calculate the time duration for each encoder's pulse. It should be noted that the time duration of each pulse (Δt), can be calculated as :

$$\Delta t = \text{Number of samples within one encoder's pulse} \times \left(\frac{1}{f}\right)$$

Where f is the sampling frequency (constant.)

4. To have the displacement versus time curve.

5. To numerically differentiate the resulting displacement, using equations (6.1, 6.2), and obtain the corresponding velocity and acceleration.

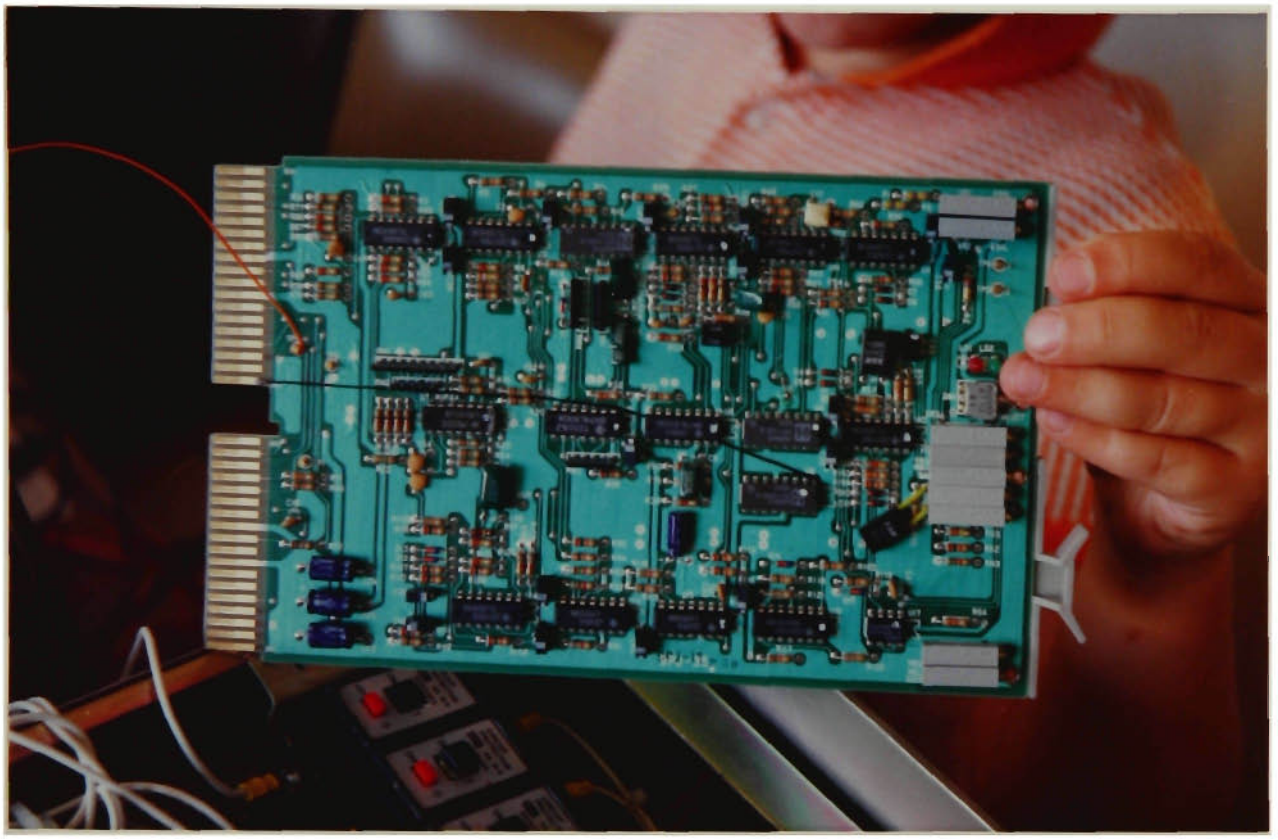


Figure 7-4: PUMA 560's analog servo-board and data interception point.

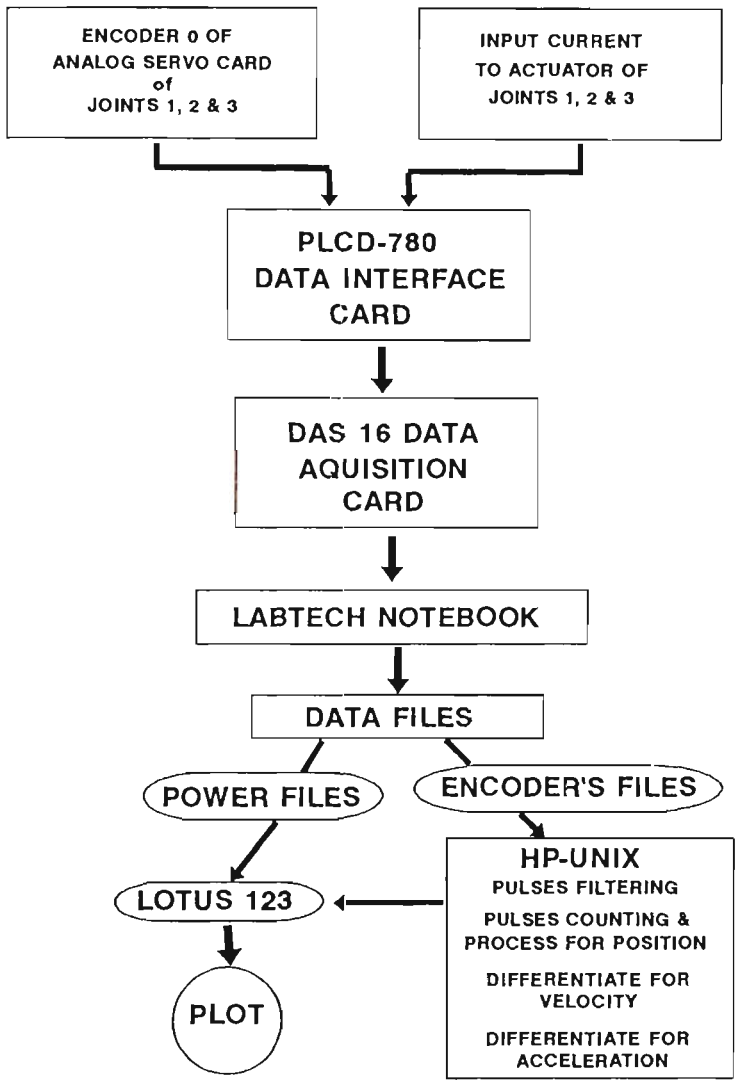


Figure 7-5: Data acquisition and processing flow diagram.

A flow chart of the data acquisition and analysis is shown in Figure 7-5. The speed of the robot during this experiment was 80% of the nominal factory setting speed. The collective time duration for the sampling of the six channels was 24 sec.

The experiment was repeated three times under three different payload conditions :

1. No payload.

2. Constant payload.

3. Discrete time-varying payload.

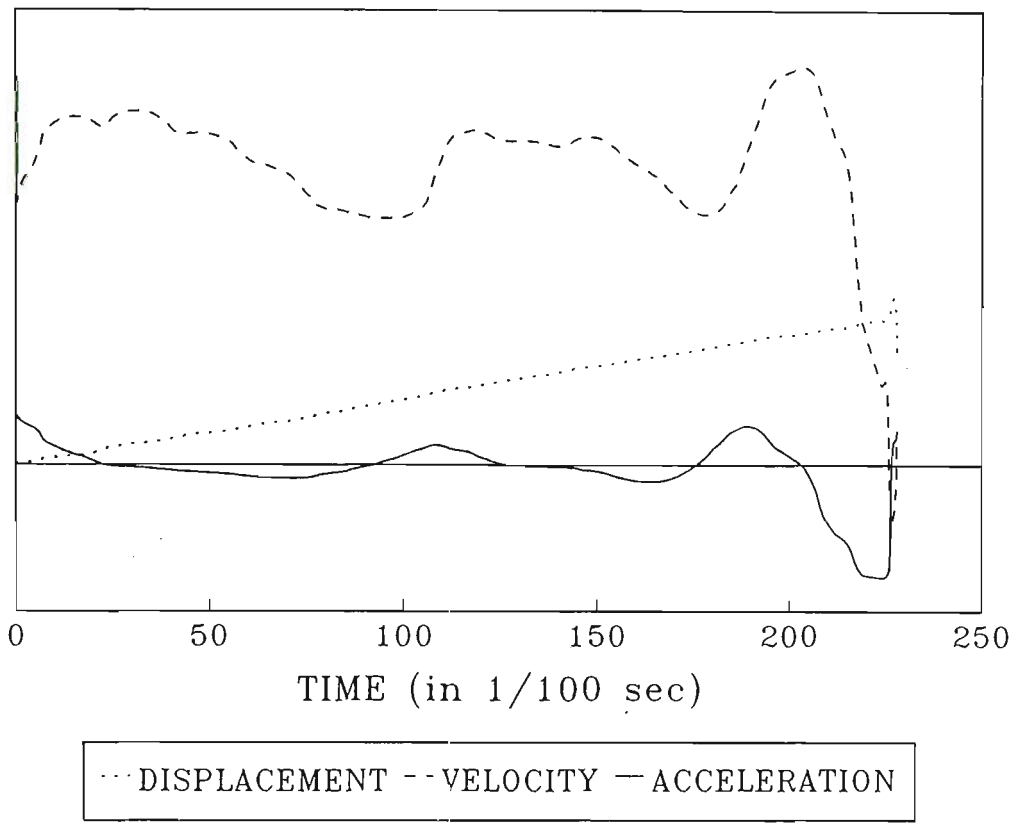


Figure 7-6: Displacement, velocity and acceleration profile of the first link.

Since the robot is position controlled, there was no substantial difference observed in the displacement, velocity and acceleration of the robot's links under the different payload conditions. The resulting links displacement, velocity and acceleration of the encoders signal processing are shown in Figures 7-6 – 7-8.

It was found, through the output of encoders signal processing, that the robot's third link moved at a constant velocity most of the cycle time. The acceleration time was relatively very short, and located at both ends of the cycle. In the meantime the second link's acceleration was smaller in magnitude and both negative and positive accelerations took a longer time in the workcycle. Also, the acceleration of the first link took the same pattern as that of the third link.

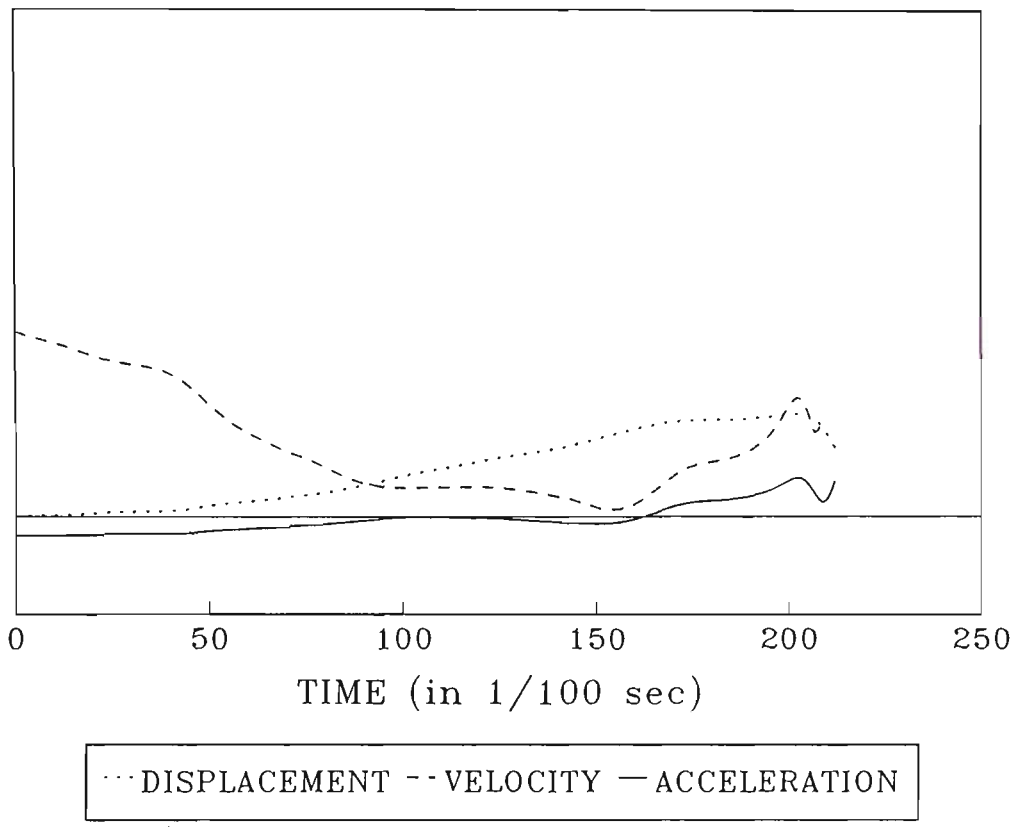


Figure 7-7: Displacement, velocity and acceleration profile of the second link.

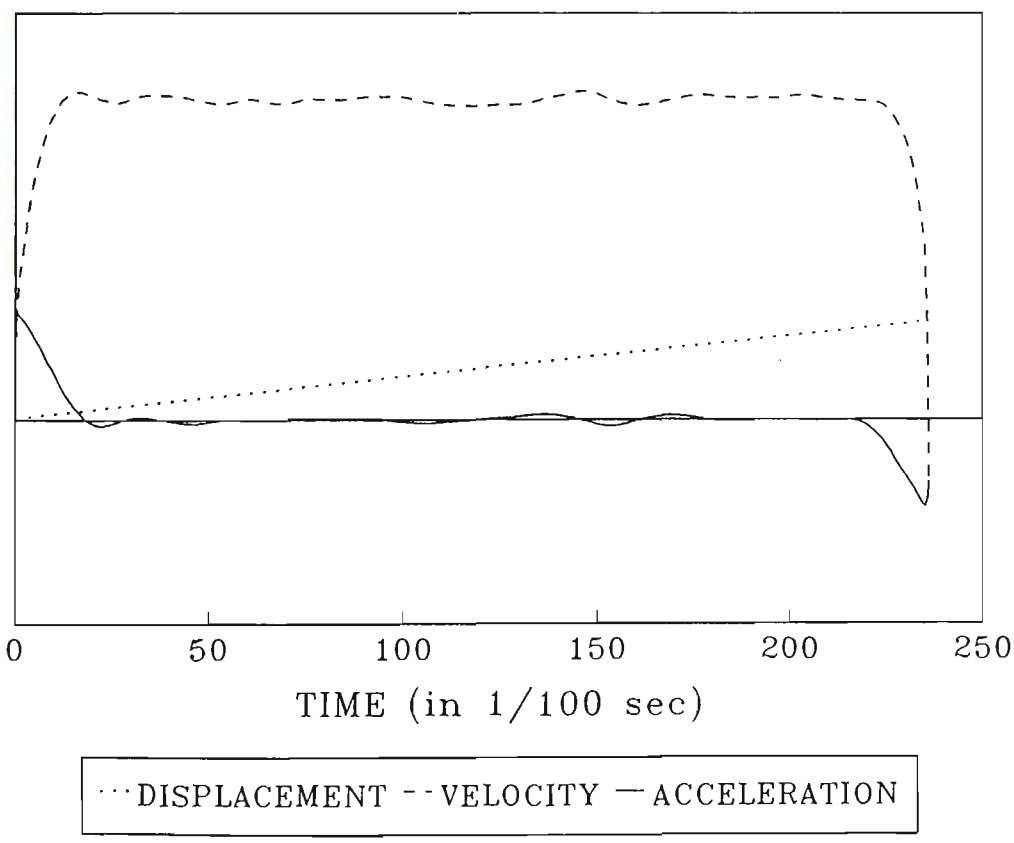


Figure 7-8: Displacement, velocity and acceleration profile of the third link.

7.2.2 Data acquisition of joint current

Since a joint's driving torque is a function of the supplied current, the joint's current was measured in this experiment to indicate the torque's value [6]. Therefore, three current guns were used to measure the joints' input current, Figure 7-9. The current readings were repeated three times, for the three payload conditions.

The outputs of the current reading for "no-payload", "constant payload" and "time-varying payload" are shown in Figures 7-10, 7-11 and 7-12 respectively. By comparing these figures, the real-time dynamic behaviour characteristics can be analysed as follows :

1. In the three payload condition cases, the second link's required torque was the most sensitive to the load variation. That was expected since it has the longest payload moment's arm.

2. Since there was no payload held in the payload ejection device, during the "no payload" case, the initial required torque of the second link was less than those of the other two cases.

3. The second link's torque, at the end of the cycle, was comparable in both "no payload" and "decreasing payload" operating conditions. However, the required torque at this moment of time was higher under the constant payload condition.

4. Under decreasing payload condition, the required torque trend shows comparable reduction. However, due to the limited current guns sensitivity (*1 v : 10 amp*), the exact reduction steps were not as clear as in the simulation study.

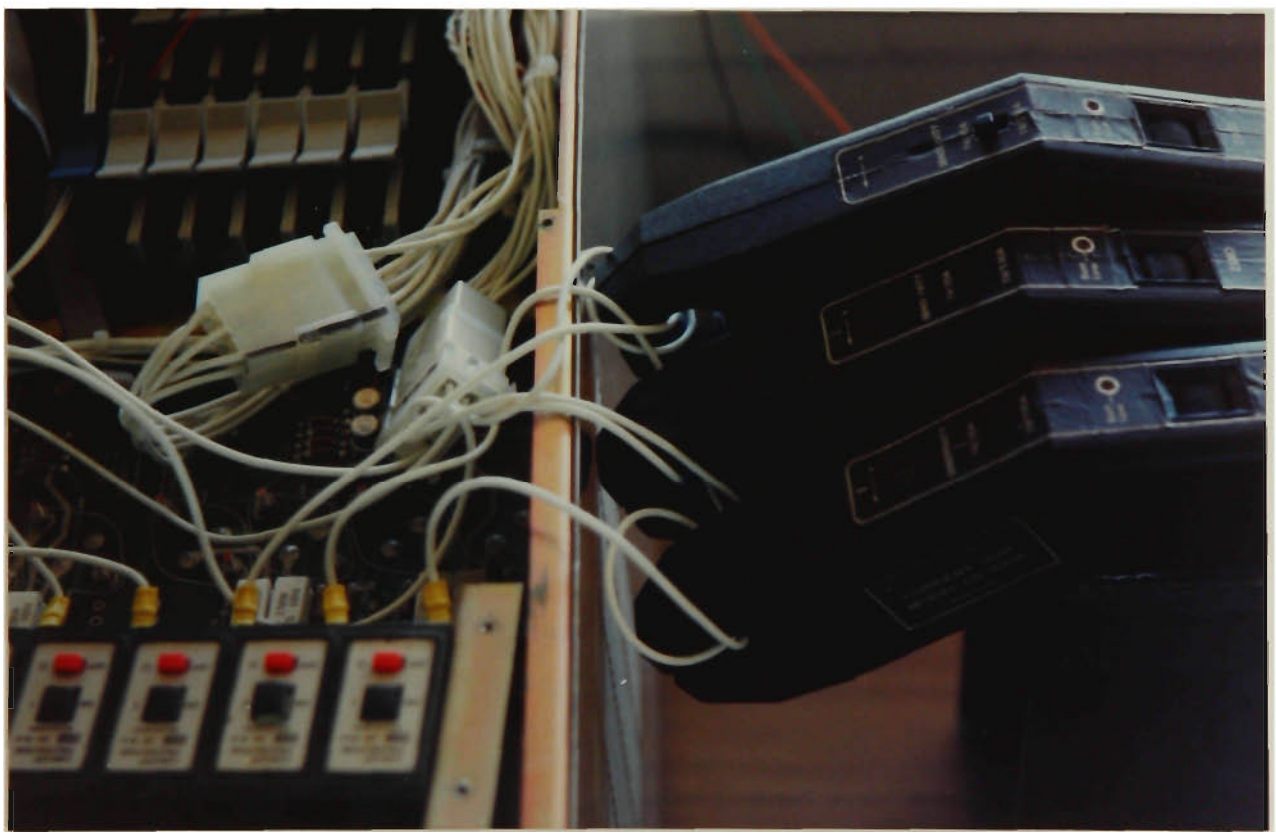


Figure 7-9: Joints current interception.

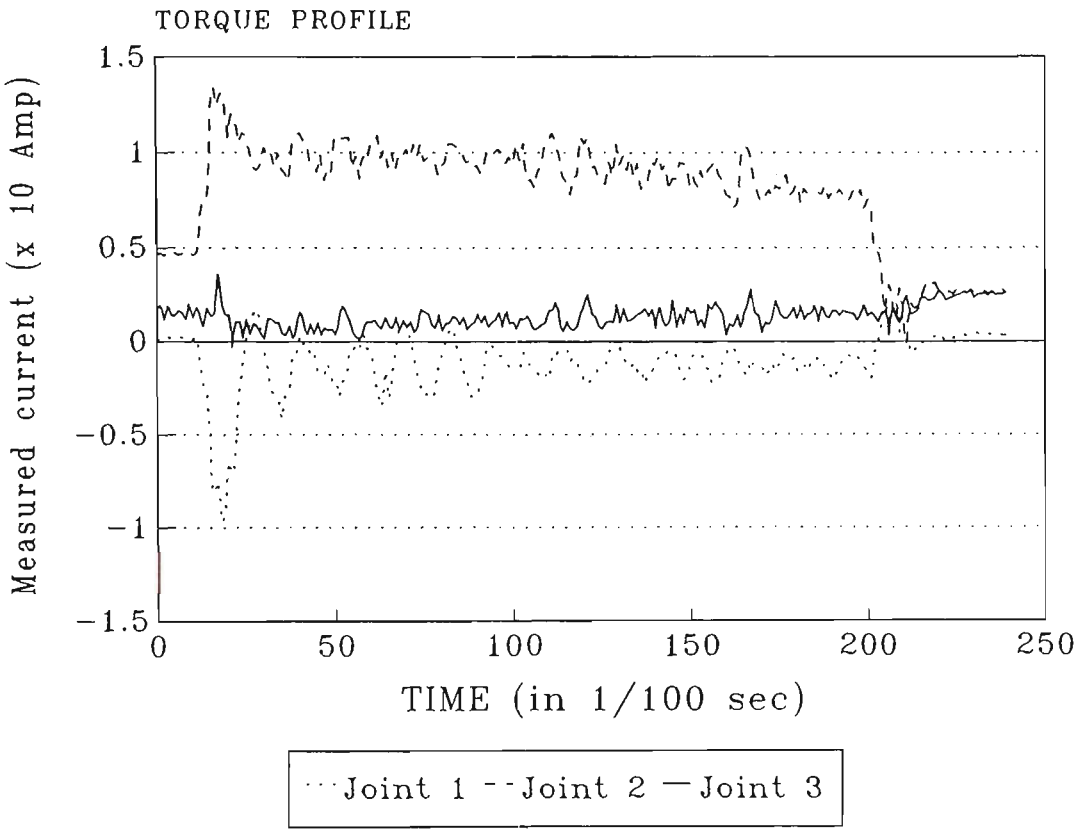


Figure 7-10: Joints torque performance under *no payload* condition.

5. The initial torque for the first link was zero. This link also required a considerable power at the beginning of the cycle. That was due to inertia-related torque. A smaller magnitude torque was required in the opposite direction, just before the end of the cycle, in order to stop the horizontal movement of the arm. That was shown as a small peak of the required torque before the the end of the cycle.

6. Through this practical experiments it was found that the maximum acceleration/deceleration time was only 18.18% of the total workcycle. This information is useful in determining the feasibility of links counter-balancing as will be discussed in Chapter 8.

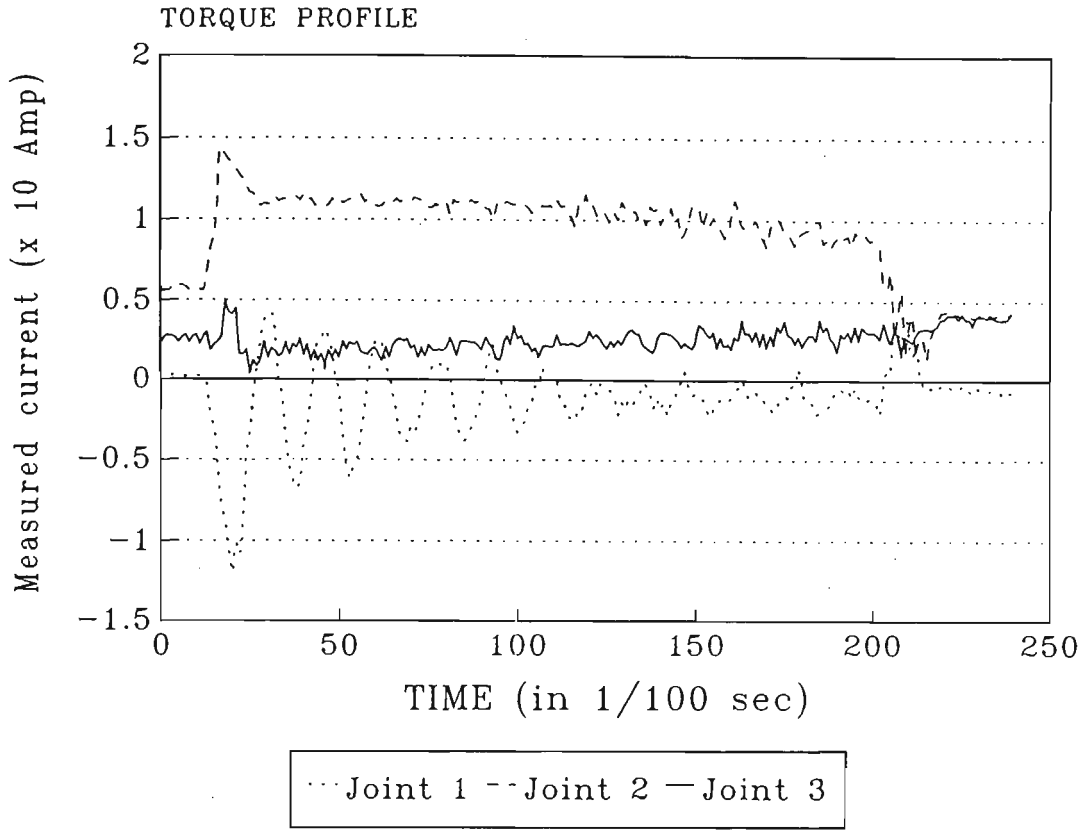


Figure 7-11: Joints torque performance under *constant payload* condition.

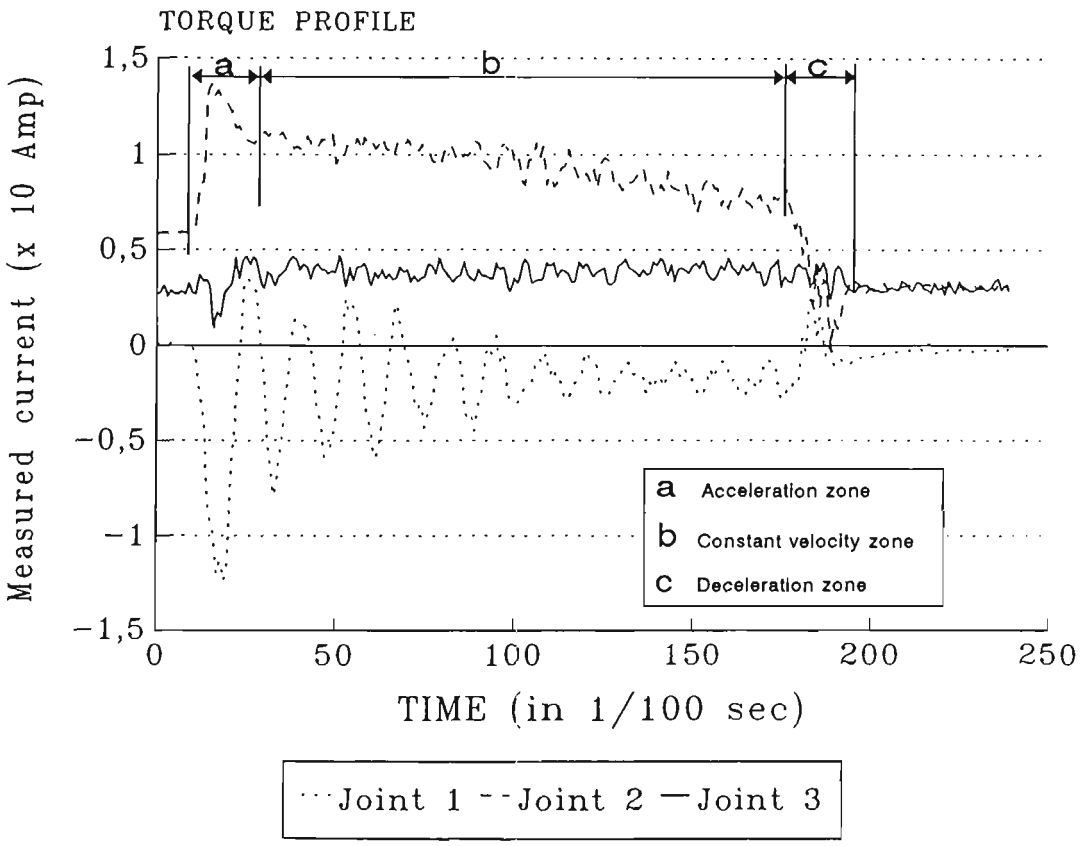


Figure 7-12: Joints torque performance under *time-varying payload* condition.

7.3 Summary

In this chapter the experimental approach to the dynamics analysis of a robot manipulator was presented. The results obtained through these experiments had some inherent inaccuracies. These were a result of several factors such as systematic errors (e.g. the current guns limited accuracy) and/or random errors. The random errors can be represented in the synchronisation of data collection with the robot's movement.

However, the trend and results obtained matched the dynamic behaviour results obtained through the simulation study presented earlier for similar work conditions. As an example, the second link's torque profile shown in Figure 7-12 for time-varying payload can be compared with the simulated results of the compatible case shown in Figure 6-8 for the acceleration and deceleration periods, taking into account that the image is flipped vertically. It should be noted that during the practical experiments the trajectory was dominated by a long constant velocity period. Therefore, the changes in payload took place during the zero acceleration period, while in the simulation, since there is no constant velocity period, the changes in payload and torques took place during an acceleration time. Also, it was found in the previous two Chapters that the dynamic behaviour of a robot's arm is more sensitive to payload changes near maximum-acceleration time than near maximum-velocity time. Accordingly, the payload changes in the practical experiments were of lesser effect than those of the simulation study. Also, since the changes in payload are very small compared with the total weight of links 2–6 which are carried by joint 2, the payload change at each step did not show a significant impact. However, the total change in the required torque due to payload variation was clear by comparing required torques in Figures 7-11 and 7-12 at the end of the constant velocity zone.

These practical experiments have given a true picture about the actual dynamic behaviour of a real industrial robot, thus enhancing the understanding of their dynamic behaviour. As an example, the real acceleration time was revealed through the encoders' signal processing. It was found that the maximum acceleration/deceleration time was only 18.18% of the total workcycle. The importance of this finding stems from the possibility of reducing the non-linear velocity loading via links counter-balancing as will be discussed in the following Chapter.

Chapter 8

Effect of Dynamic Balancing on the Performance Characteristics of an Articulated Robot

8.1 Introduction

This research is concerned with improving the dynamic performance of a robot arm by eliminating the non-linear velocity-related torques (centripetal/Coriolis forces). This potentially eliminates the non-linearities, and by eliminating the gravitational force, can help in further reducing the total required torque.

Therefore, investigation was conducted on the effect of link counter-balancing on the dynamic performance of an articulated robot. The investigation was conducted by comparing the robot's dynamic performance with and without counter-balance masses in order to visualise the extent of the effect of counter-balancing.

The robot chosen for this investigation is a PUMA 560 robot manipulator. The first link of this robot is already balanced. The design of the counter-balance masses for the second and third links is fully explained in Appendix B page 178.

8.2 Robot's Trajectories

The study is also aimed at assessing the effect of counter-balancing on the inter-link coupling forces. Therefore, two different path trajectories were used to demonstrate that effect.

In the first trajectory, both the first and the second links remained stationary at their own zero positions. However, the third link moved a full 180^o from its zero position within a period of one second. As a consequence, the third link should generate high centripetal forces. A schematic diagram of the link's movement in this trajectory is shown in Figure 8-1.

In the second trajectory, both the second and the third links remained stationary at their own zero positions. However, the first link moved a full 180^o from its zero position within a period of one second. As a consequence, the first link was exposed to high inertia forces. A schematic diagram of the link's movement in the second trajectory is shown in Figure 8-2.

The moving link's displacement, in both trajectories, was a third order polynomial function in time as described in equation (4.1). The maximum velocity (5.89 rad/sec) occurred at its mid-cycle point (0.5 sec.), as shown in Figure 8-3.

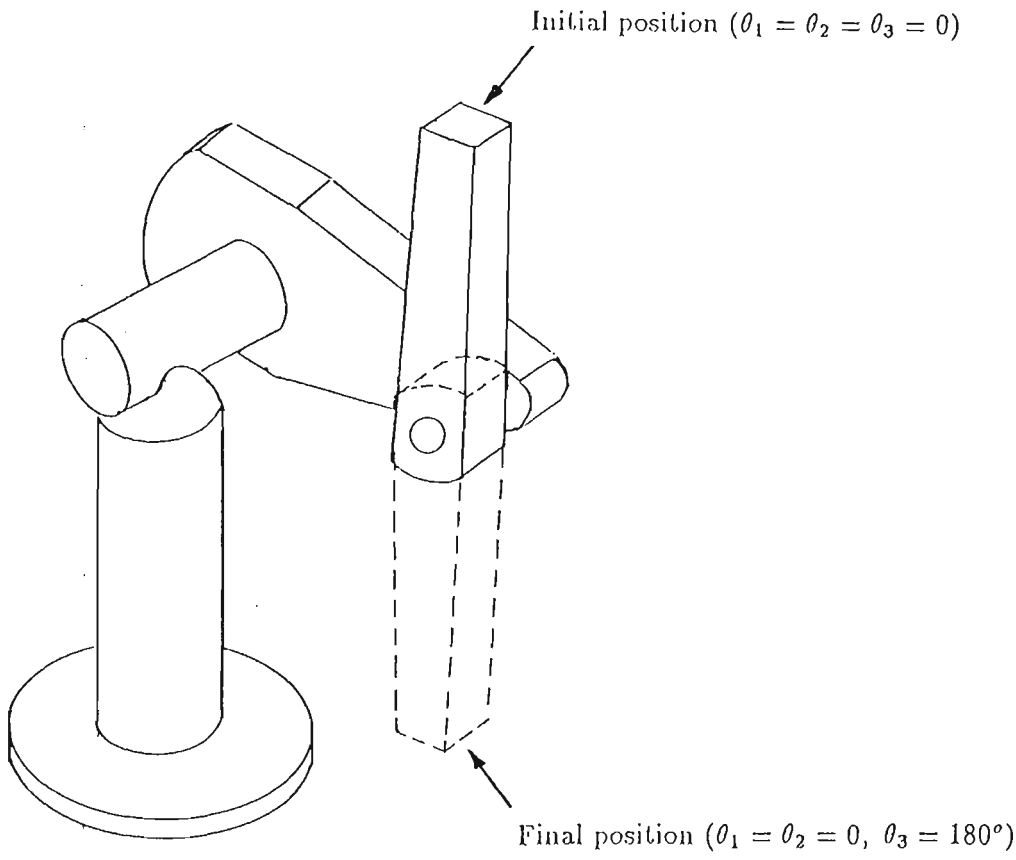


Figure 8-1: Schematic diagram of the first robot's path trajectory.

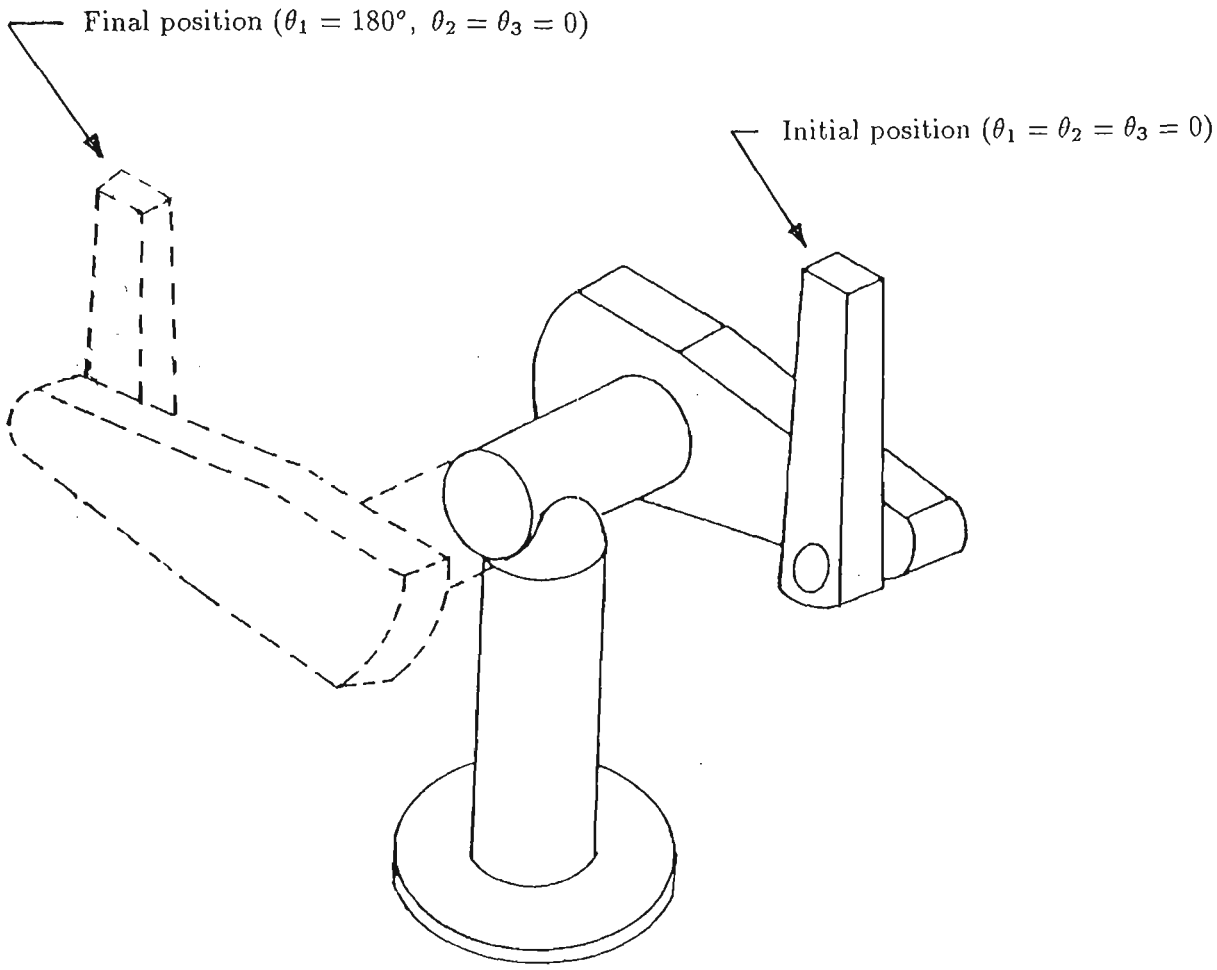
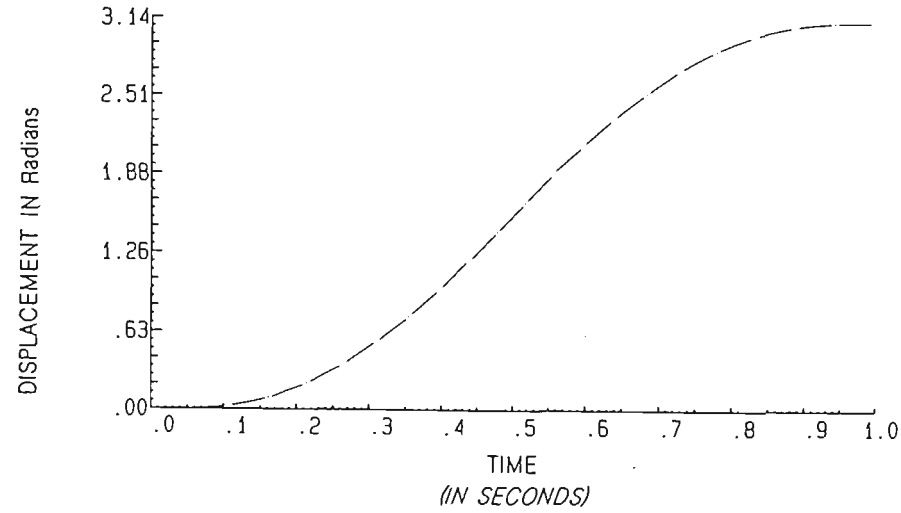
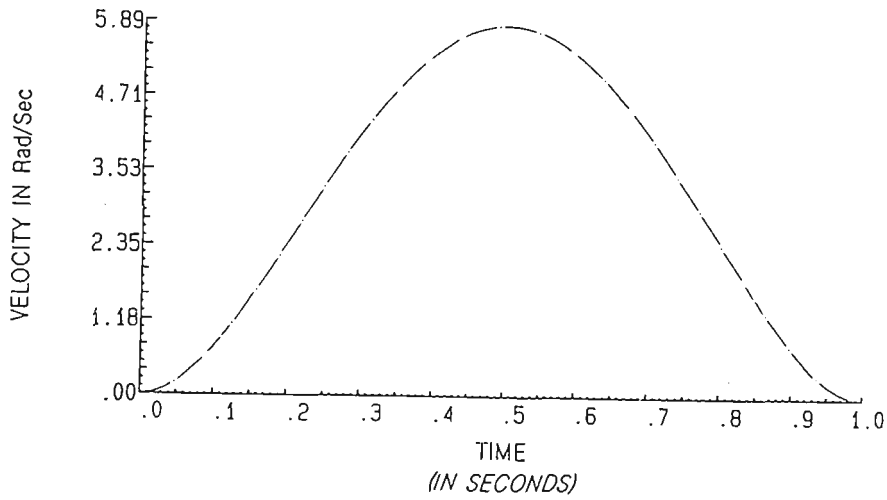


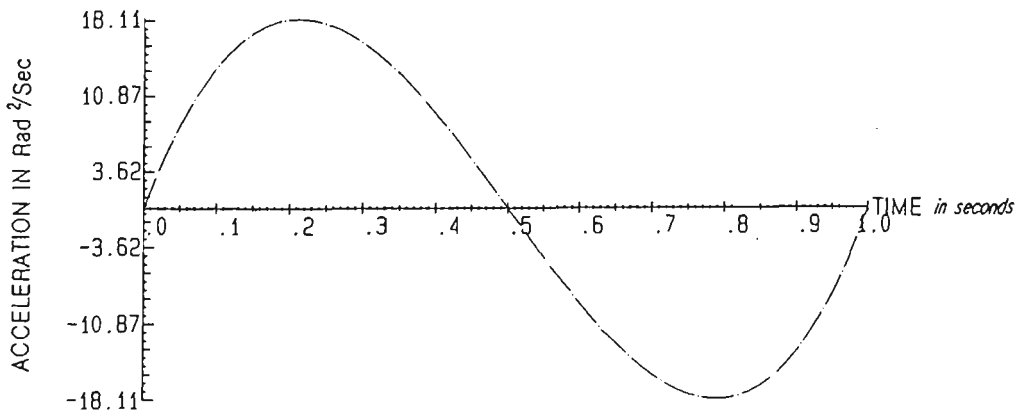
Figure 8-2: Schematic diagram of the second robot's path trajectory.



DISPLACEMENT TRAJECTORY OF LINK No.3



VELOCITY TRAJECTORIES OF LINK NO. 3



ACCELERATION TRAJECTORY OF LINK No.3

Figure 8-3: Link's motion characteristics.

8.3 Driving Torques for the First Trajectory

In order to have a clear understanding of the effect of counter-balance masses, it was necessary to study the influence of counter-balancing on the centripetal/Coriolis and inertia forces separately. Therefore, the closed-form dynamic model, explained on page 21, was employed in this investigation. Also in this study, the third link was augmented to include the wrist components as a point mass.

8.3.1 Dynamic performance without counter-balancing

The calculation of the required torque for each link was conducted using the reduced closed-form model as explained in Section 2.3.2. In this calculation the dynamics of each contributing component was individually calculated to check its response. Figure 8-4, shows the driving torques of the three links. Based on the results shown in this figure, the dynamic behaviour of the three links can be summarised as explained below.

Third link dynamic performance

Figure 8-4-a shows that the third link's actuator is not affected by its centripetal force. This is because the third link's centripetal force line of action passes through the joint centre line; hence the resulting moment's arm is zero. Therefore, the total driving torque for the third link is due to the inertial and gravitational forces only.

Second link dynamic performance

By analysing the driving torques of the second link, Figure 8-4-b, the following characteristics can be deduced :

- Although the second link did not move, its actuator has felt several types of torques due to the third link's movement.
- The inertia and Coriolis loading were entirely due to the dynamic coupling.
- The Coriolis force of the second link was zero at the third link's maximum velocity point (0.5 sec.). The reason behind that is that at the moment of the third link's maximum velocity, the arm was fully stretched horizontally. Thus the Coriolis force acting on link two due to the velocity of the third link acts through the joint of the second link. That causes the velocity-related torque, at the point of the third link's maximum velocity, on the second link to be zero.
- However, the maximum Coriolis force acting on the second link occurred at points before and after the midpoint of the trajectory. That was because this kind of force is a function of both $\dot{q}^2$ and the perpendicular distance between the line of action of the force and the joint.
- The inertial contribution in the required torque for the second link was due to the acceleration of the first link. Therefore, the inertial torque followed the same pattern as the acceleration curve of the third link, Figure 8-3-c.
- Also, the fluctuation in the gravity loading was due to the rate of change in the position-dependent, potential energy of the third link.

First link dynamic performance

While the effect of Coriolis forces on the second link is zero at the maximum velocity point, it is maximum on the first link at the same instant. This is due to the orthogonality of joint 1 axis Z_o and the line of action of the Coriolis force at this point in time. The torque arm in this case is the offset between the second link and the first link (d_2). The Coriolis force peaks at mid-cycle point (0.5 sec.) and has a mirror image about this point.

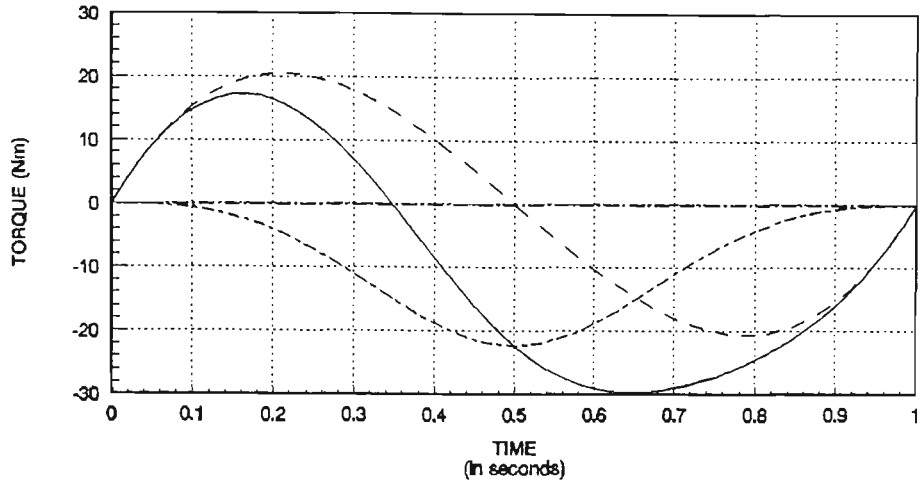
The first link has also experienced inertia loading due to the acceleration of the third link. The inertial torque was zero at 0.5 seconds due to the third link's acceleration value at this instant of time. The direction of the inertial coupling was the same throughout the cycle because the horizontal component of the third link's acceleration was in the same direction (away from the arm) throughout the cycle.

The first link's actuator did not experience any gravitational coupling because of the third link's movement. That was because the first joint's axis Z_0 was parallel to the gravitational force line of action, as shown in Figure 3-1.

As a result of the above facts the first link actuator was subjected only to coupling inertia and Coriolis forces due to the third link's movement. The resulting total torque on the first link's actuator is shown in Figure 8-4-c.

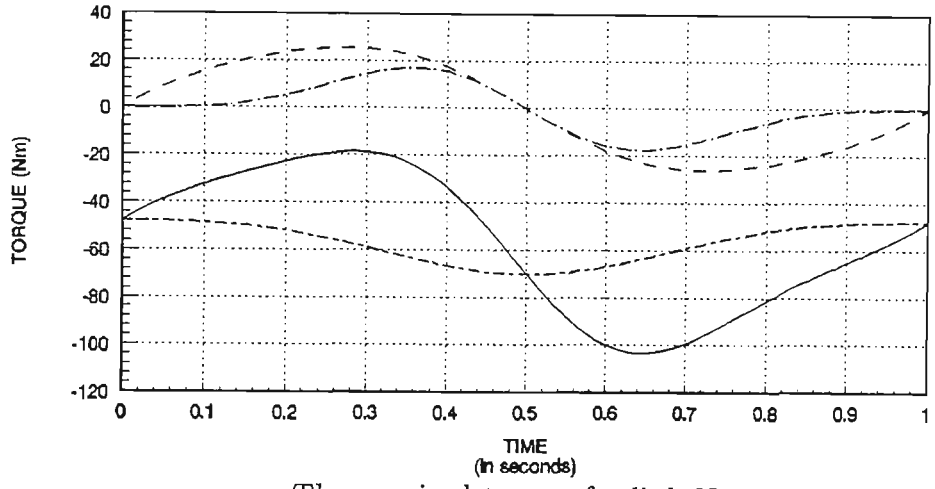
8.3.2 Dynamic performance with counter-balanced links

Counter-balance masses were designed for the third and second links (Appendix B). The first link did not need a counter-balance mass since it is already dynamically balanced. The movement trajectory of the third link was repeated, as in the previous case, while the second and third links remained stationary. The



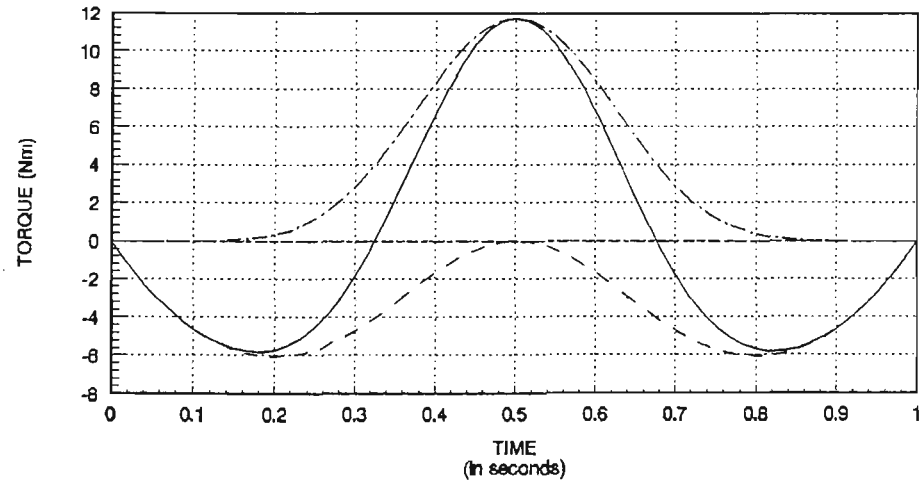
The required torque for link No. 3

(a)



The required torque for link No. 2

(b)



The required torque for link No. 1

(c)

Legend :

- Inertial Loading
- Centripetal/Coriolis Loading
- Gravitational Loading
- Resultant Torque

Figure 8-4: Dynamic performance without counter-balance masses (trajectory 1).

third link's movement followed the same velocity and acceleration trajectories as in the case of no counter-balancing. The results of the driving torques of each contributing force, for each link under the dynamic balancing condition, were obtained and plotted as shown in Figure 8-5.

Third link dynamic performance

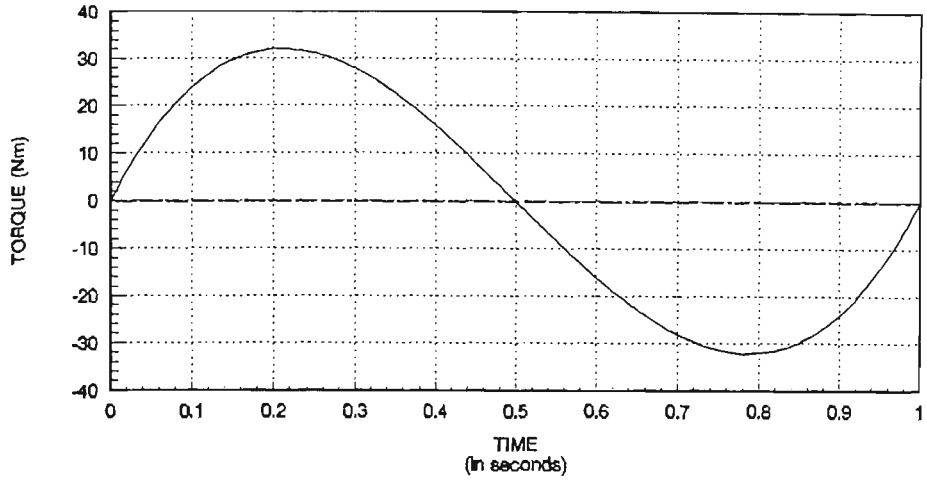
It was noticed that the centripetal/Coriolis forces and gravity forces were eliminated from the third link, Figure 8-5-a. The total torque for this link was due to the inertia force only.

Second link dynamic performance

The only dynamic coupling force that has affected the second link's actuator due to the third link's movement is the coupling inertia force. The coupling inertia on the second link was equivalent to the third link's inertia in magnitude and direction throughout the cycle, Figure 8-5-b.

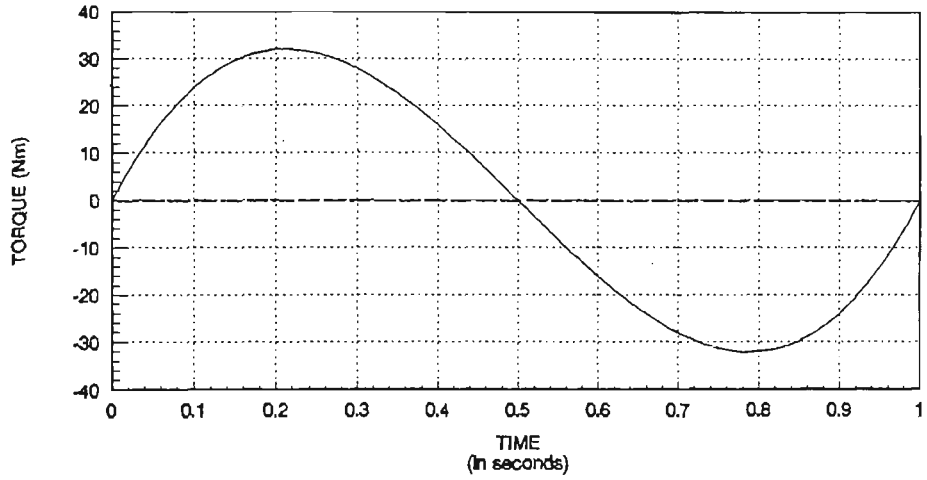
First link dynamic performance

The effect of dynamic balancing on the first link was quite substantial. The counter-balancing has eliminated the effect of both a) the Coriolis force and b) coupling inertia force on the first link. The coupling inertia were eliminated from the first link since the horizontal components of both the third link and its counter-balance mass accelerations were in opposite directions throughout the cycle, which is valid for any trajectory.



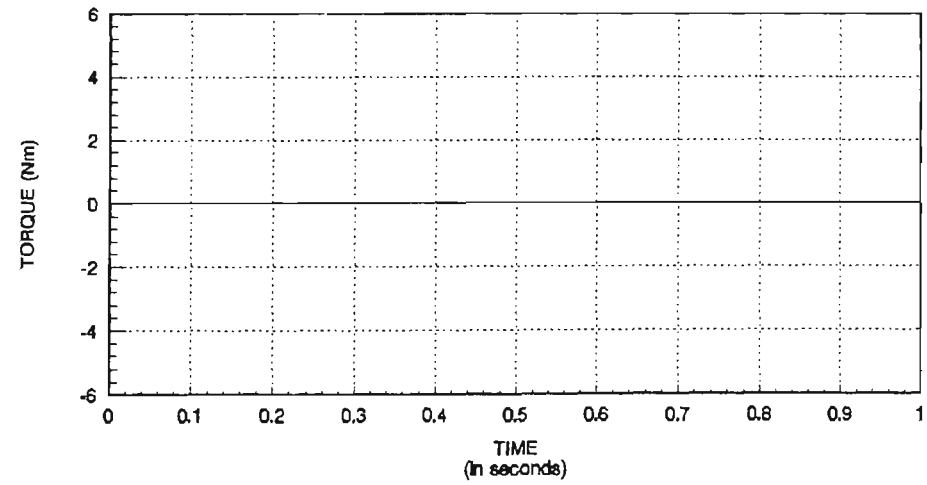
The required torque for link No. 3

(a)



The required torque for link No. 2

(b)



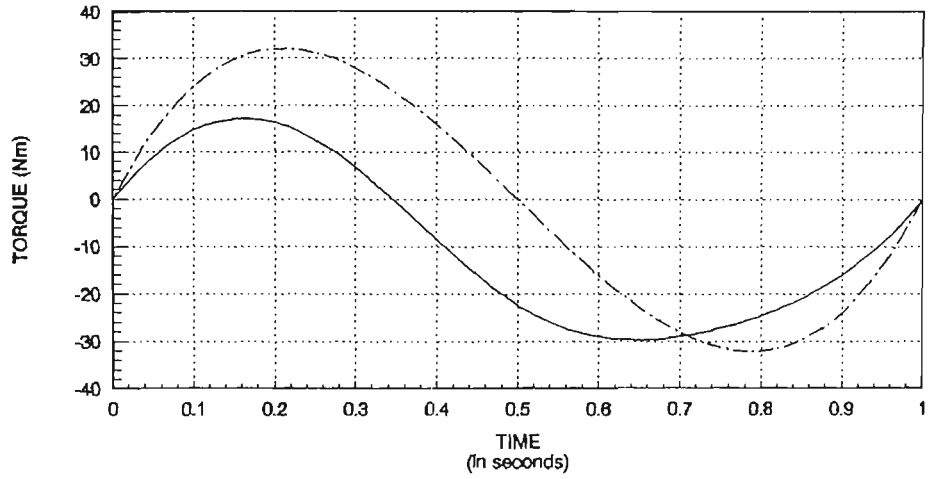
The required torque for link No. 1

(c)

Legend :

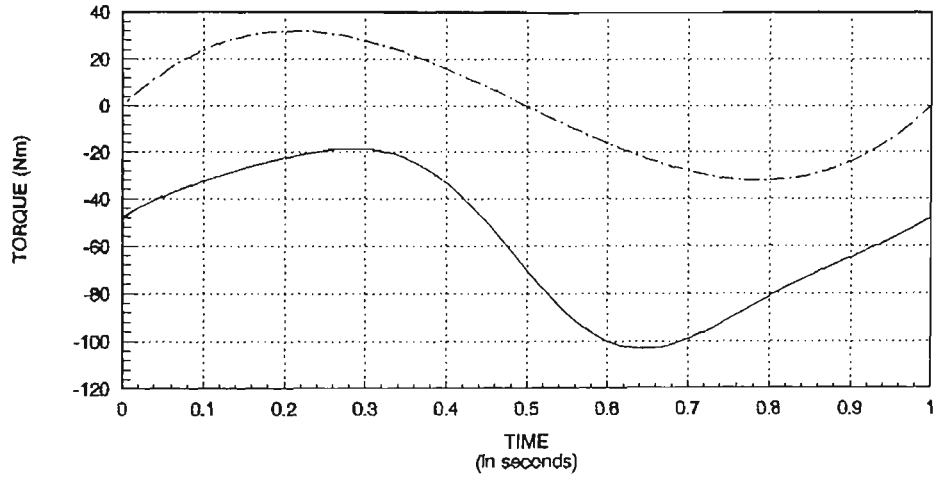
- Inertial Loading
- Centripetal/Coriolis Loading
- Gravitational Loading
- Resultant Torque

Figure 8-5: Dynamic performance with counter-balance masses (trajectory 1).



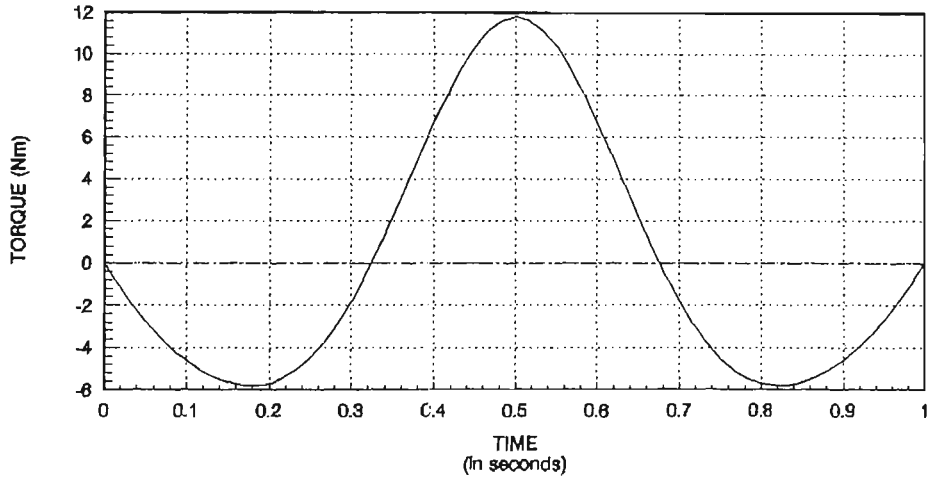
The required torque for link No. 3

(a)



The required torque for link No. 2

(b)



The required torque for link No. 1

(c)

Legend : —No Counter-balance
--- Counter-balanced

Figure 8-6: Total required torque for links with and without dynamic balancing (trajectory 1).

8.4 Driving Torques for the Second Trajectory

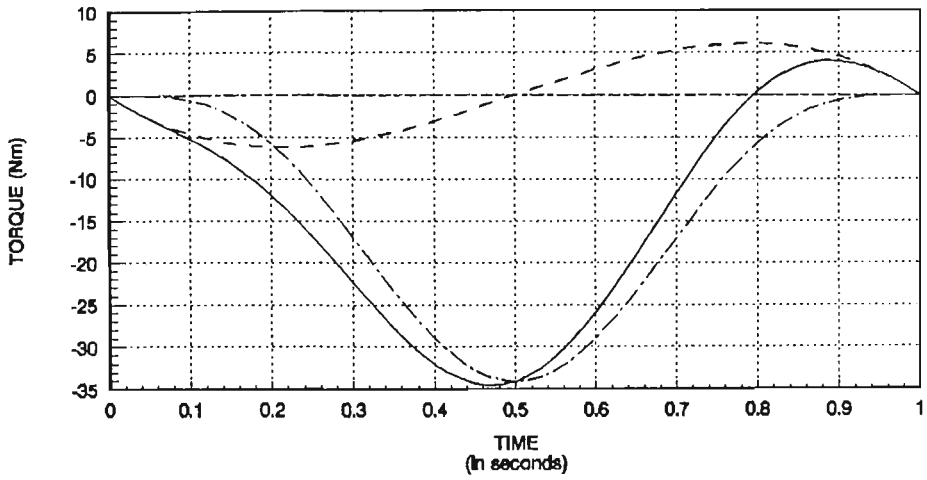
Robot manipulators are serial mechanisms. Therefore, an addition of a mass to a link affects the driving torques of the preceding links. Hence, it was important to quantitatively assess the extent of the effect of the inertia forces generated by the counter-balancing masses on the first link. The second path trajectory was thus designed such that the inertia forces will have the greatest effect on the first link.

8.4.1 Dynamic performance without counter-balancing

Figure 8-7-a shows that the third link was under the influence of both the velocity and inertia loading. Also, Figure 8-7-b shows that the second link was under the same coupling forces as the first link in addition to the gravitational loading. However, the first link was subjected only to the inertia forces. This is due to the orthogonality of joint 1 axis (Z_o) and the line of action of the Coriolis and gravitational forces of the second and third links. As shown in Figure 8-7-c, the maximum required torque for link 1 due to the inertia loading was 50 Nm.

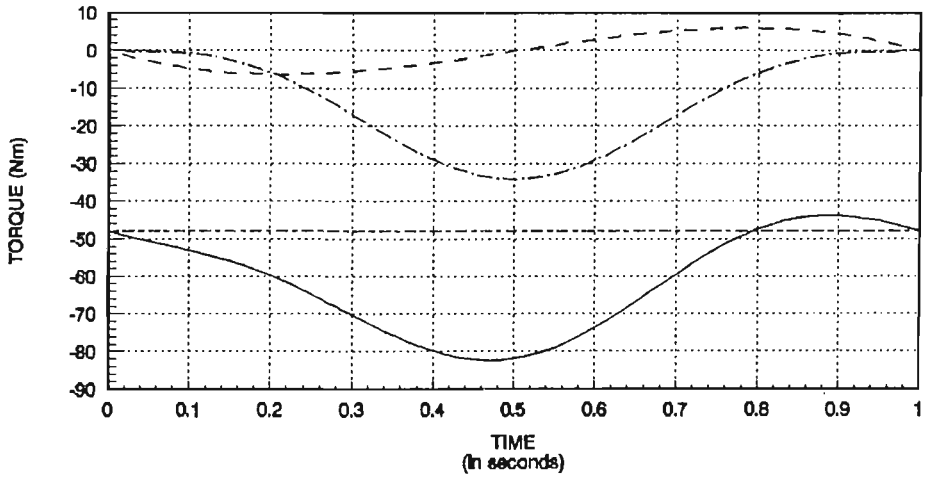
8.4.2 Dynamic performance with counter-balanced links

The second path trajectory was repeated with links counter-balanced. The driving torques were plotted in Figure 8-8. It was found through these results that while counter-balancing has eliminated the coupling forces of the third and the second links, the maximum inertia loading on the first link has increased from 50 Nm to 152 Nm. This means that the addition of the counter-balancing masses has resulted in a 304% increase in the inertia loading of the first link .



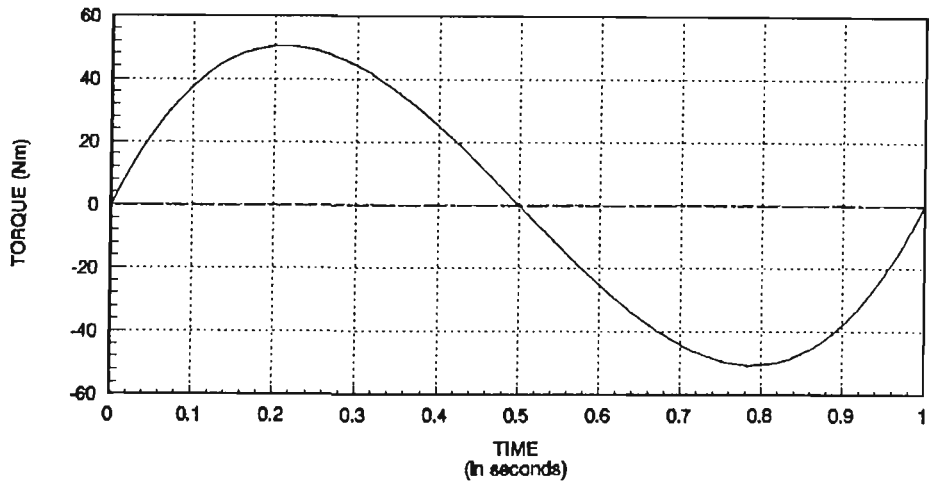
The required torque for link No. 3

(a)



The required torque for link No. 2

(b)



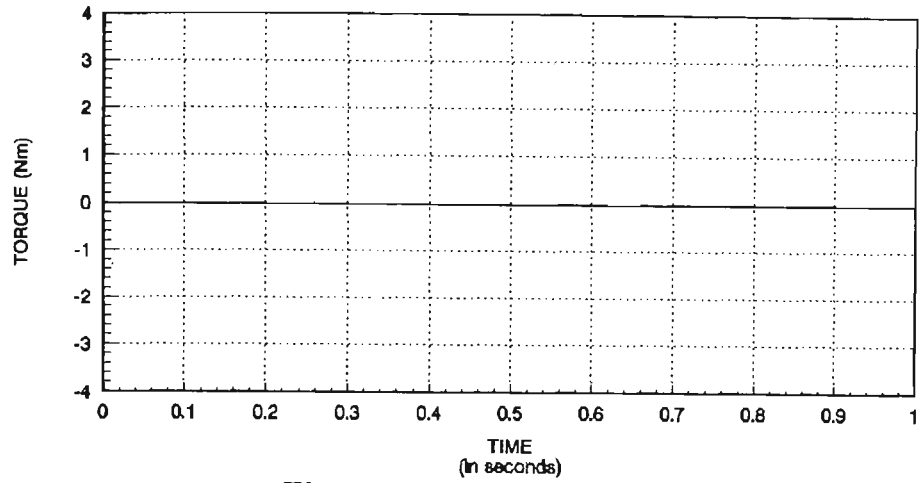
The required torque for link No. 1

(c)

Legend :

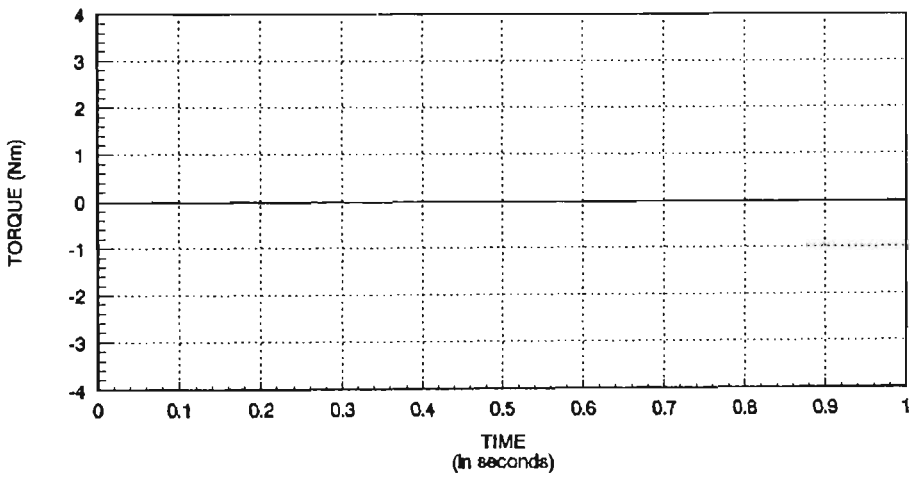
- Inertial Loading
- Centripetal/Coriolis Loading
- Gravitational Loading
- Resultant Torque

Figure 8-7: Dynamic performance without counter-balance masses (trajectory 2).



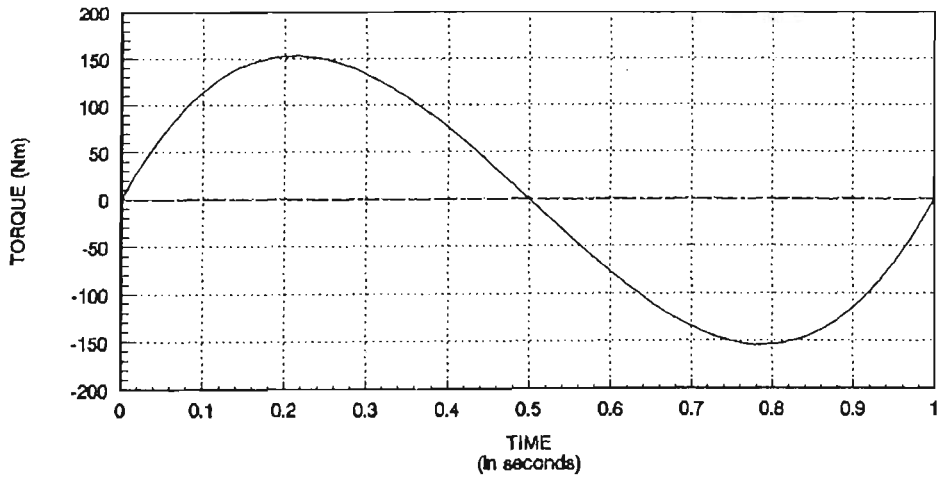
The required torque for link No. 3

(a)



The required torque for link No. 2

(b)



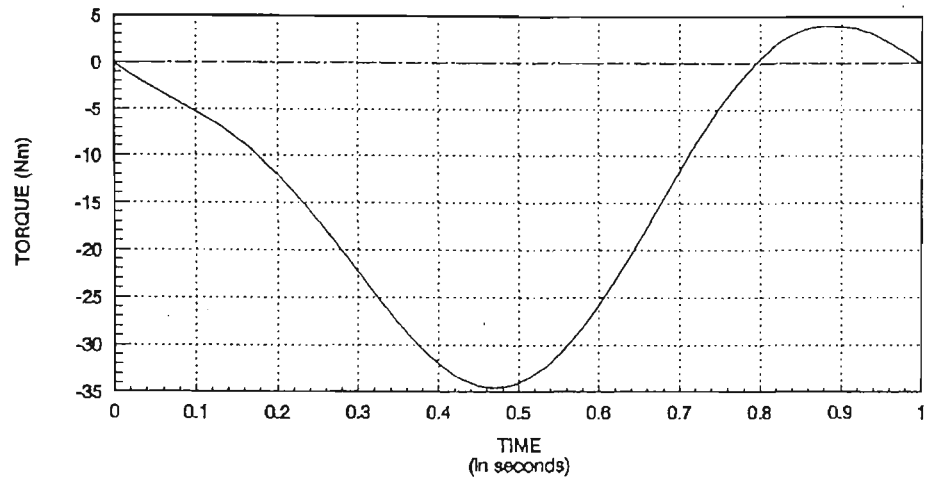
The required torque for link No. 1

(c)

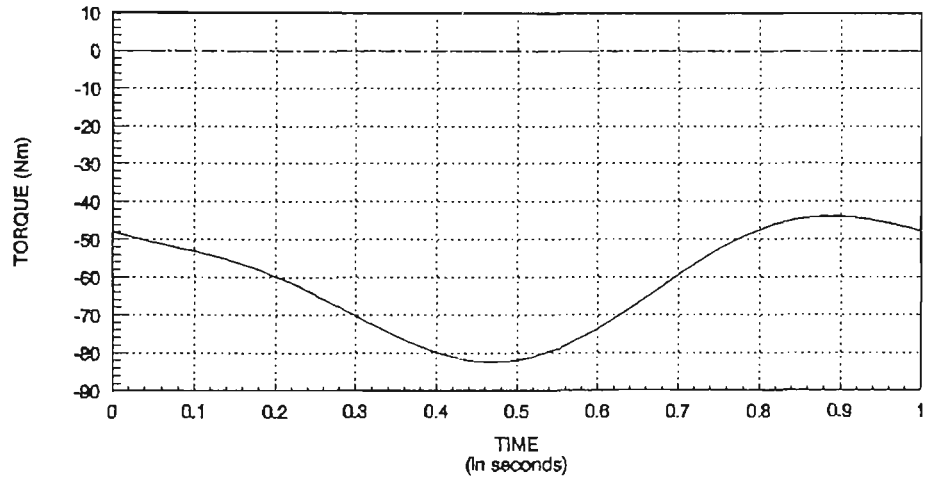
Legend :

- Inertial Loading
- Centripetal/Coriolis Loading
- Gravitational Loading
- Resultant Torque

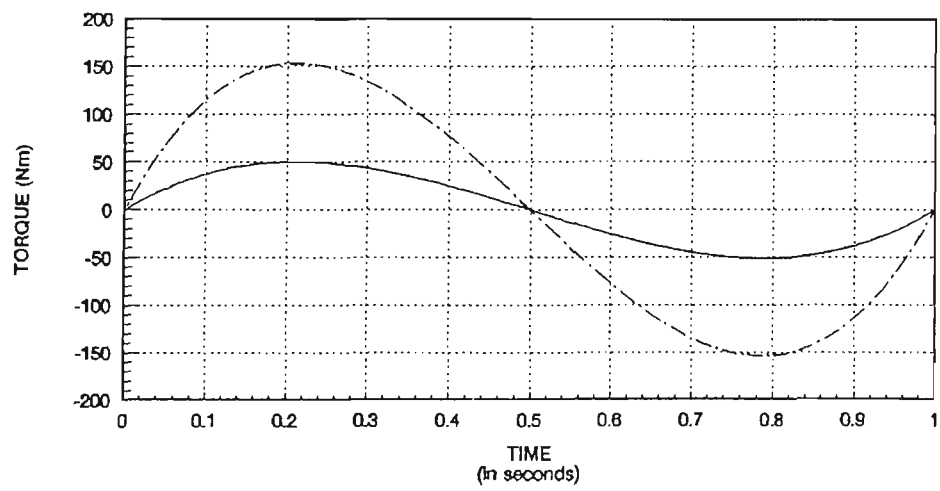
Figure 8-8: Dynamic performance with counter-balance masses (trajectory 2).



The required torque for link No. 3
(a)



The required torque for link No. 2
(b)



The required torque for link No. 1
(c)

Legend: — No Counter-balance
--- Counter-balanced

Figure 8-9: Total required torque for links with and without dynamic balancing (trajectory 2).

8.5 Summary

In this study two different path trajectory were used to examine the effect of dynamic balancing on the performance characteristics of a PUMA 560 robot arm. A comparison between the total required driving torques for the robot's links with and without balancing for the first path trajectory is shown in Figure 8-6. This comparison shows that the total required torque for the third link was greater with a counter-balance mass than without counter-balancing. That increment in the total required torque for the third link occurred, despite the elimination of the gravitational and centripetal forces, because of the increment in the inertia forces.

However, in this trajectory there was a substantial reduction in the total required torque for the second link. The maximum required torque was only 32 Nm with link counter-balanced compared to 103 Nm without counter-balancing. This difference will increase non-linearly as the speed increases.

Also, the effect of counter-balancing was even greater on the first link, where all the coupling forces were eliminated and the total required torque was zero.

Although there was an increment in the required torque in this path trajectory for the third link, due to the inertia forces generated by the addition of the counter balance masses, the sum of the total required torques for the three links was much less due to the great reduction in the coupling forces. It should also be mentioned that in the counter-balance masses design, the PUMA's maximum payload (2.5 Kg) was taken into account as a constant payload. However, because of the link's high weight to payload ratio, a payload variation would not have a considerable effect on the results obtained.

It should be noted that the trajectories used in this simulation study consisted of an acceleration followed by a deceleration without a constant velocity

zone. Since the effect of counter-balancing minimises the velocity loading on the required torque, the longer the maximum velocity zone (and/or the greater the maximum velocity value); the greater is the torque saving. With reference to the results of the practical experimentation reported in Chapter 7, it is evident that the links of practical industrial robots often move at constant velocity during most of the cycle time. That means a typical robot moves at a constant maximum velocity most of the cycle time, and the acceleration time is very minimal.

The disadvantage of counter-balancing involves the addition of more masses which increase the inertial load on the first link. That was demonstrated in the results of the second path trajectory as shown in Figure 8-9. It was found that despite the minimisation of the gravitational and nonlinear velocity loading of both links 2 and 3, link 1 encountered an increase in its required torque of up to 304% due to the high inertial loading. Therefore, for counter-balancing implementation, the first link should either move with a limited maximum acceleration/deceleration or be connected to a more powerful (bigger) actuator. A combination of these options would be a feasible solution since the actuator of the first link is not carried by the robot hence incrementing in its size does not incur a dynamic penalty on the robot's arm. Therefore, a robot's designer has to trade off the merits of counter-balancing and its effect to obtain a global optimality in the robot's performance based on the robot's work conditions and intended usage.

Chapter 9

Effect of a Robot's Geometrical Parameters on its Dynamic Performance

9.1 Introduction

In the previous chapters the dynamic performance of robot manipulators was examined and analysed under :

- different operational conditions; such as velocity trajectories or payloads and,
- structural modifications; such as link counter-balancing.

In searching for an optimal robot dynamic performance, and as an extension to the study conducted in the previous chapters on robots dynamic performance, the effect of different geometrical parameters was also examined.

There has been considerable research to analyse the effect of a robot's geometrical parameters on its workspace performance [67-69]. However, there is

no comparable work done on the effect of the geometrical parameters on the *dynamic* performance. Therefore, in this chapter the dynamic performance response to changes in various geometrical parameters was examined and the results summarised and graphically presented in 3-dimensional surface plots.

It was shown earlier (section 4.3.1) that a typical robot's dynamic performance is dominated by its inertia terms under normal operating conditions. Also, the centripetal/Coriolis forces become significant only at high velocity. Therefore, the sensitivity of the inertia matrix's eigenvalues to changes in a robot's geometrical parameters is taken as an indicator of performance.

Therefore, the closed-form dynamic model was further extended to include the necessary calculations for the inertia matrix's eigenvalues and its sensitivity. The study considered various geometrical parameters. The considered parameters were related to the first three links of an articulated robot's arm (PUMA 560). Since the inertia matrix is a function of the joint variable q; the performance was examined for a particular parameter over a range of joint angles.

9.2 Performance Measure

In order to investigate the effect of geometrical parameters on the dynamic performance, it was necessary to establish an indicator for that performance.

Since the robot is assumed to be a three link manipulator, the dimension of the inertia matrix was 3×3. Also, as explained on page 14, D_{ij} can be expressed as :

$$D_{ij} = \sum_{p=max\,(i,j)}^{n} Tr\left(\frac{\partial T_p}{\partial q_j}\, J_p\, \left(\frac{\partial T_p}{\partial q_i}\right)^T\right) \tag{9.1}$$

The inertia matrix of the three links can be expressed explicitly as follows :

$$D(q) = \begin{pmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{pmatrix} \tag{9.2}$$

Where :

$$\begin{aligned}
D_{11} &= Tr\left(U_{11}\, J_1\, U_{11}^T\right) + Tr\left(U_{21}\, J_2\, U_{21}^T\right) + Tr\left(U_{31}\, J_3\, U_{31}^T\right) \\
D_{12} &= Tr\left(U_{22}\, J_2\, U_{21}^T\right) + Tr\left(U_{32}\, J_3\, U_{31}^T\right) \\
D_{13} &= Tr\left(U_{33}\, J_3\, U_{31}^T\right)
\end{aligned}$$

$$\begin{aligned}
D_{21} &= Tr\left(U_{21}\, J_2\, U_{22}^T\right) + Tr\left(U_{31}\, J_3\, U_{32}^T\right) \\
D_{22} &= Tr\left(U_{22}\, J_2\, U_{22}^T\right) + Tr\left(U_{32}\, J_3\, U_{32}^T\right) \\
D_{23} &= Tr\left(U_{33}\, J_3\, U_{32}^T\right)
\end{aligned}$$

$$\begin{aligned}
D_{31} &= Tr\left(U_{31}\, J_3\, U_{33}^T\right) \\
D_{32} &= Tr\left(U_{32}\, J_3\, U_{33}^T\right) \\
D_{33} &= Tr\left(U_{33}\, J_3\, U_{33}^T\right)
\end{aligned}$$

Where :

$$U_{ij} = \frac{\partial T_i}{\partial q_j}$$

The inertia matrix D is a real symmetric matrix due to the following facts [70, 71] :

1. $Tr[A] = Tr[A]^T$ (for a square matrix A)

2. $Tr[ABC] = Tr[BCA] = Tr[CAB]$

By applying the above matrix trace characteristics in equation (9.1), the following relation can be obtained :

$$D_{ij} = D_{ji}$$

Therefore D is a symmetric matrix.

Also since every real matrix is similar to a diagonal matrix [71]; then D is similar to a diagonal matrix. Hence, D is a diagonable matrix.

Since the inertia matrix D is reducible to a diagonal matrix, its characteristics values remain unchanged represented by the elements of its principal diagonal in its reduced form. Therefore, the sensitivity of the inertia matrix w.r.t. a variable can be represented by the sensitivity of its eigenvalues w.r.t. the same variable. Also, the sensitivity of the inertia matrix eigenvalues to changes in a robot's geometrical parameter is simplified by the fact that the eigenvalues of a symmetrical matrix are real [72, 73].

The eigenvalues of the inertia matrix in equation (9.2) can be expressed as :

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} \tag{9.3}$$

Also, ξ_i was introduced to represent a robot's geometrical parameters for joint i, such that :

$$\xi_i \in \alpha_i, a_i, d_i$$

where α_i, a_i and d_i are the link i's geometrical parameters as shown in Figure 2-1.

In order to obtain the average response of the three eigenvalues; their Euclidean norm was calculated. This norm can be expressed as :

$$\mu = \sqrt{\sum_{j=1}^{N} \lambda_j^2} \qquad \text{(where N = number of d.o.f.)}$$

The Euclidean norm was used in order to obtain a positive average value irrespective of the sign of the individual eigenvalue. Also, since the search is focusing on finding the minimum sensitivity irrespective of the derivative's sign, the absolute values of the derivative were employed to calculate the performance indicator. This induced non-negative values for the eigenvalues' norm sensitivity.

This non-negativeness of the derivative values allowed usage of their logarithm as a performance measure. Thus, a wider spectrum of the dynamic response to geometrical parameter changes became available.

Therefore, the sensitivity of an inertia matrix's eigenvalues to any change in a link's geometrical parameter can be expressed as the derivative of their norm w.r.t. the parameter $\frac{\partial \mu}{\partial \xi_i}$; for $i = 1, 2, 3$.

Based on the above assumptions, a performance indicator for the sensitivity of the inertia matrix's eigenvalues to changes of a geometrical parameter ξ of link i was chosen to be Γ_{ξ_i}. Mathematically, the performance indicator can be expressed as follows:

$$\Gamma_{\xi_i} \triangleq f\left(col\left(q\right), \xi_i\right) \tag{9.4}$$

Therefore, Γ_{ξ_i} can be expressed explicitly as follows :

$$\left.\begin{array}{lll} \Gamma_{\xi_i} & = & \left\| \frac{\partial \mu}{\partial \xi_i} \right\|_{(i=1,\ldots,N)} \quad \text{(for a linear scale of } \Gamma_{\xi_i}\text{)} \\ \text{or} & & \\ \Gamma_{\xi_i} & = & \log_{10} \left\| \frac{\partial \mu}{\partial \xi_i} \right\|_{(i=1,\ldots,N)} \quad \text{(for a logarithmic representation of } \Gamma_{\xi_i}\text{)} \end{array}\right\} \tag{9.5}$$

Where :

$$col\left(q\right) = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$$

$$\mathrm{N} \stackrel{def}{=} \text{Number of joints.}$$

The geometrical parameters of the first three links of the PUMA 560 are shown in Table 6.1.

A minimum sensitivity of the inertia matrix eigenvalues w.r.t. a geometrical parameter, would indicate that the parameter has a minimal influence on the

dynamic performance. Also, from the adaptive control point of view, the smaller the sensitivity of the dynamic model, the less will be the required degree of adaptation. Therefore, minimal sensitivity is favourable from both design and control viewpoints. The sensitivity of the inertia matrix's eigenvalues to changes in the non-zero parameters of the PUMA 560 was examined in the course of this thesis. The applied velocity trajectory to the links motion was the third order polynomial trajectory discussed on page 38. The links motion took one second to complete its cycle. The results of this study are summarised and discussed below.

9.3 Effect of α_1 on the Dynamic Performance

The sensitivity of the inertia matrix's eigenvalues, Γ_{α_1}, to the changes in the twist angle α_1 was studied. The study was conducted over a range of adjacent configurations along different path trajectories. Two sets of results associated with two different motion trajectories are discussed below.

9.3.1 Dynamic performance under the first motion trajectory

In this investigation the motion displacement trajectory of the three links is schematically described in Figure 9-1 and its data tabulated in Table 9.1. The range of α_1 over which the sensitivity analysis was conducted is :

$$-90^o \leq \alpha_2 \leq 90^o$$

The dynamic sensitivity to changes in both α_1 and robot arm configuration is shown in Figure 9-2. It can be seen from these figures that the dynamic performance indicator, Γ_{α_1}, was highly dependent on the twist angle α_1. However,

during the specified path trajectory illustrated in Figure 9-1, the changes in the robot's configuration had no significant effect on eigenvalues sensitivity.

It was also revealed that the lowest sensitivity is attained in the region where $\alpha_1 = 0$. Furthermore, through usage of the logarithmic function shown in Figure 9-3, it was found that the absolute minimum sensitivity occurs when $\alpha_1 = 0$ $and\, t = 0$. It should be noticed that the robot's arm was fully stretched horizontally at $t = 0$ sec. through the robot's movement trajectory. That is because at that configuration (arm fully stretched horizontally) a small change in α_1 would not have a significant effect on the arm's inertia due to the perpendicularity of both the second and third links to Z_0. However, that perpendicularity will not hold for any small change in q_2 or q_3. Hence, at this new configuration, any small change in the twist angle (α_1) creates a significant change in the arm's inertia.

That further explains the above-mentioned perpendicularity characteristics of a robot's links. However, at α_1 equal to zero the robot exhibits a planar motion only, thus sacrificing the work space envelope. Therefore, a trade-off emerges between the choice of a geometrical parameter's value for optimal dynamic performance and the choice of a better work space. Also, the dynamic performance indicator, Γ_{α_1}, exhibited sensitivity to changes in the robot's configuration in some motion trajectories as shown in the following trajectory.

9.3.2 Dynamic performance under the second motion trajectory

The eigenvalues sensitivity to α_1 were examined during the course of a different robot's motion trajectory. The range of α_1 was identical to that of the previous trajectory, i.e. :

$$\alpha_1 = -90^0 \quad \Longrightarrow \quad \alpha_1 = 90^0$$

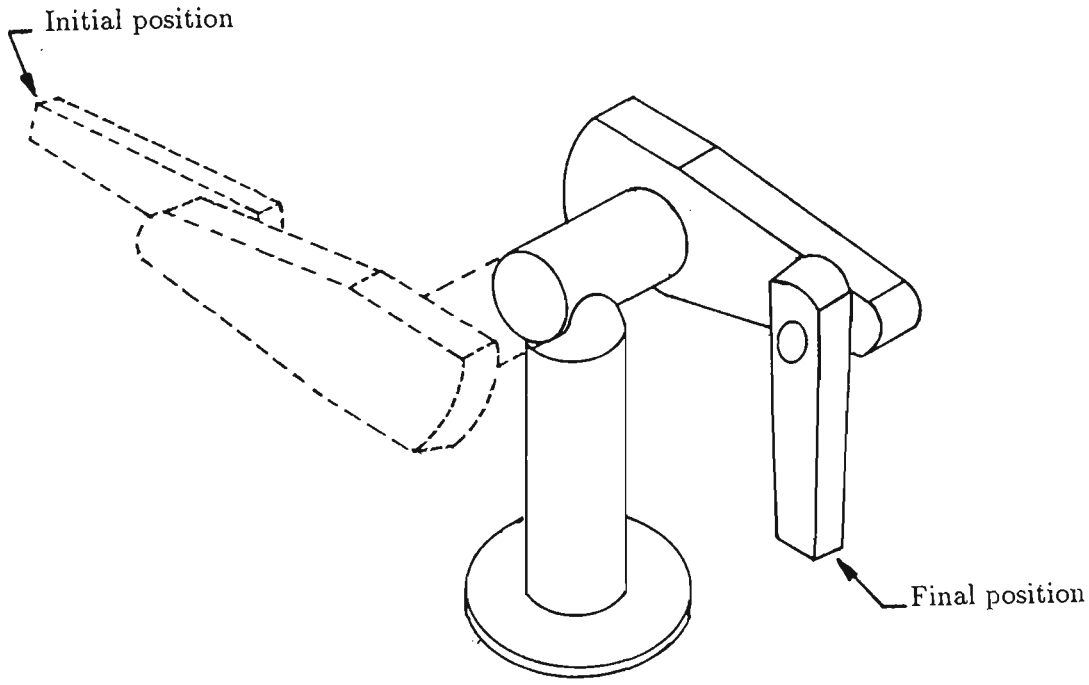


Figure 9-1: Schematic diagram of the first robot's path trajectory for Γ_{α_1}.

Total displacement of link 1	180^o
Starting position of link 1	-90^o
Total displacement of link 2	0.0
Starting position of link 2	0.0
Total displacement of link 3	90^o
Starting position of link 3	90^o

Table 9.1: Data for the first motion trajectory for Γ_{α_1}.

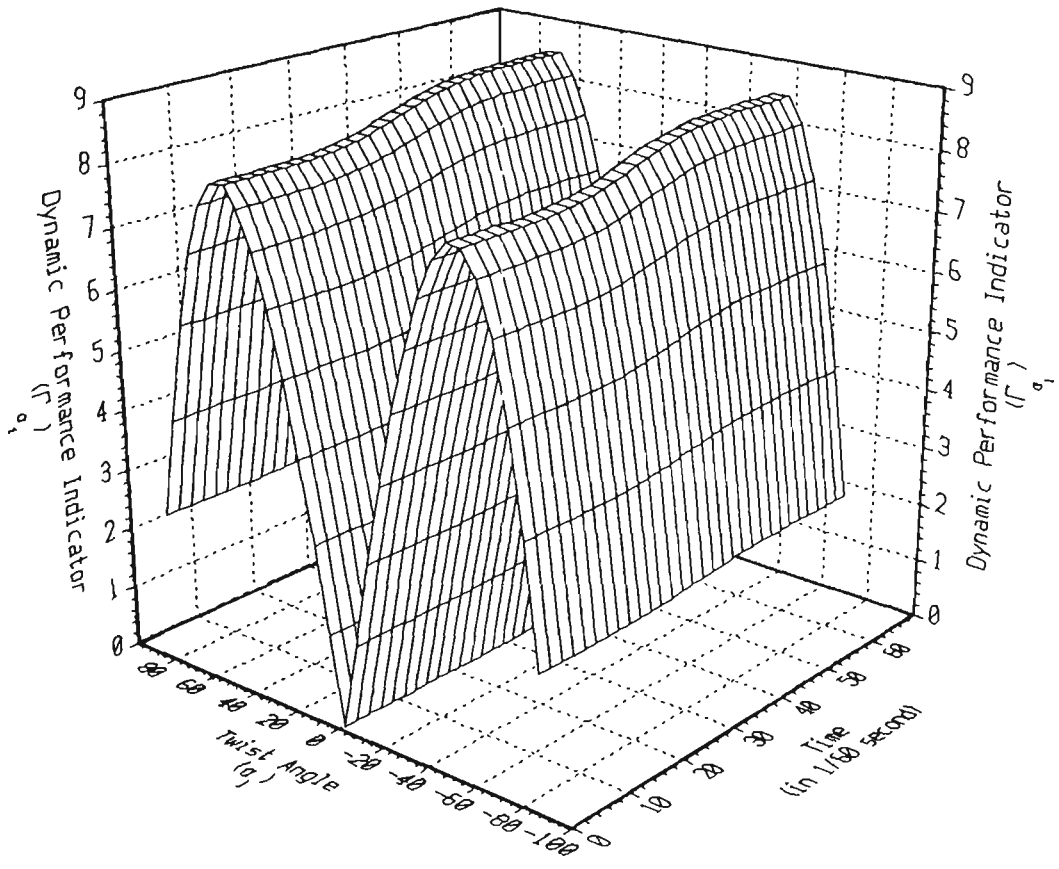


Figure 9-2: Dynamic performance indicator versus α_1 and time (trajectory 1).

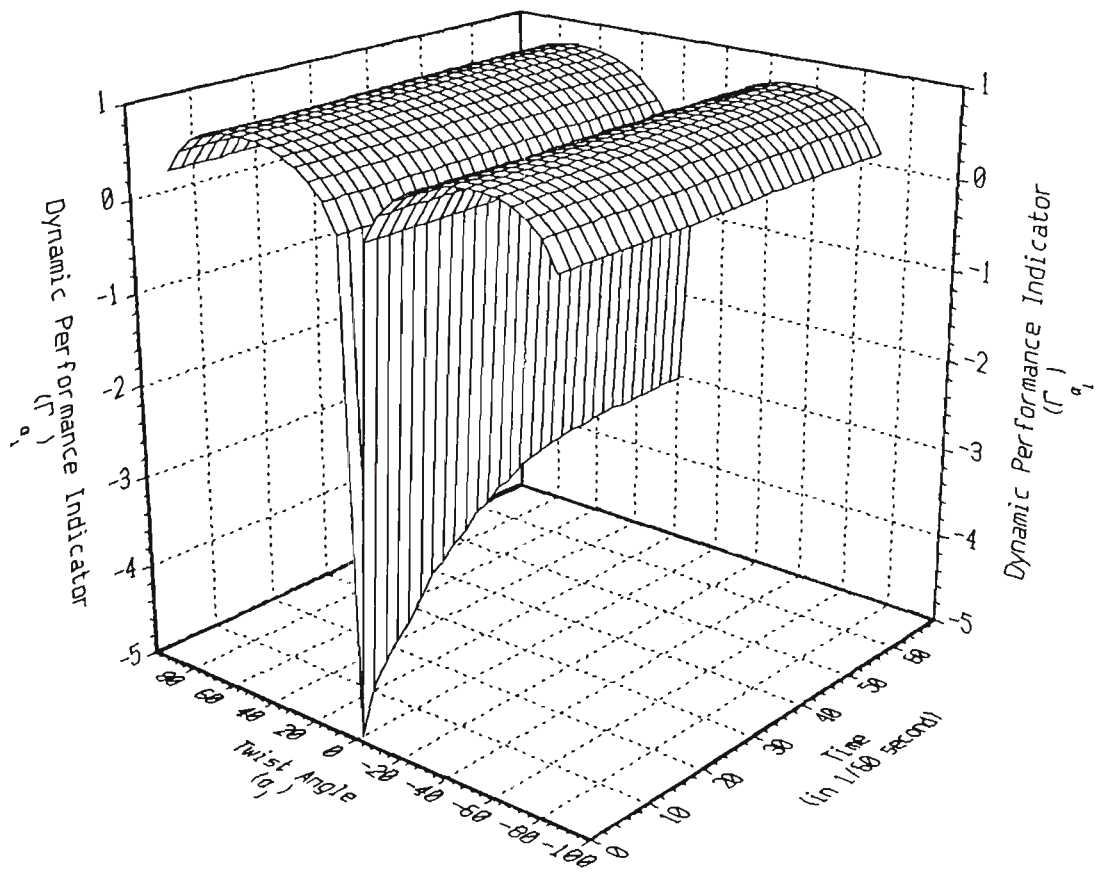


Figure 9-3: Logarithmic representation of Γ_{α_1} (trajectory 1).

In this motion trajectory the second link was stationary in a horizontal position, while the third link moved 180^o from -180^o position (vertically downward) to 0^o position (vertically upward). The links' motions are schematically shown in Figure 9-4 and their data are summarised in Table 9.2.

Figure 9-5 shows the inertia matrix $D(q)$'s eigenvalues sensitivity to both :

- changes in the geometrical parameter α_1, and
- changes in the robot's configuration through the motion trajectory.

From the plotted results in Figure 9-5, the eigenvalues sensitivity was again smaller about the zero value of α_1. It was also found that the sensitivity is symmetrical about $t = 0.5$ sec. That was due to the symmetry of the robot's path trajectory below and above $q_3 = 90^o$.

It is found from the logarithmic function of Γ_{α_1}, shown in Figure 9-6, that the absolute minimum sensitivity occurred at $t = 0.5$ sec. It should be noticed that the robot's arm was fully stretched horizontally at this point in time through the robot's movement trajectory. That further validates the above-mentioned perpendicularity characteristic of a robot's links (page 134).

9.4 Effect of α_2 on the Dynamic Performance

Further simulation experiments were conducted to investigate the effect of twist angle of the second link α_2 on the dynamic performance. The experiments were conducted for two different motion trajectories. In the first motion trajectory the robot's arm was stretched horizontally and rotated 180^o in a horizontal plane around Z_0. In the second trajectory the robot's arm was also stretched during the motion's cycle and moved 180^o about Z_1.

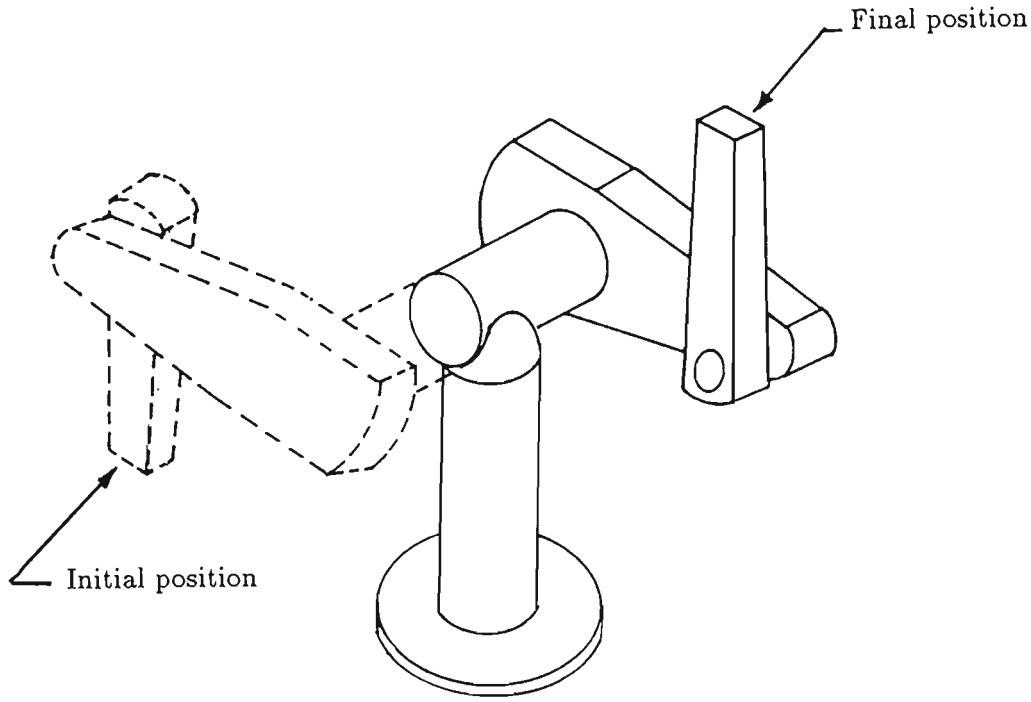


Figure 9-4: Schematic diagram of the second robot's path trajectory for Γ_{α_1}.

Total displacement of link 1	180^o
Starting position of link 1	-90^o
Total displacement of link 2	0.0
Starting position of link 2	0.0
Total displacement of link 3	-180^o
Starting position of link 3	180^o

Table 9.2: Data for the second motion trajectory for Γ_{α_1}.

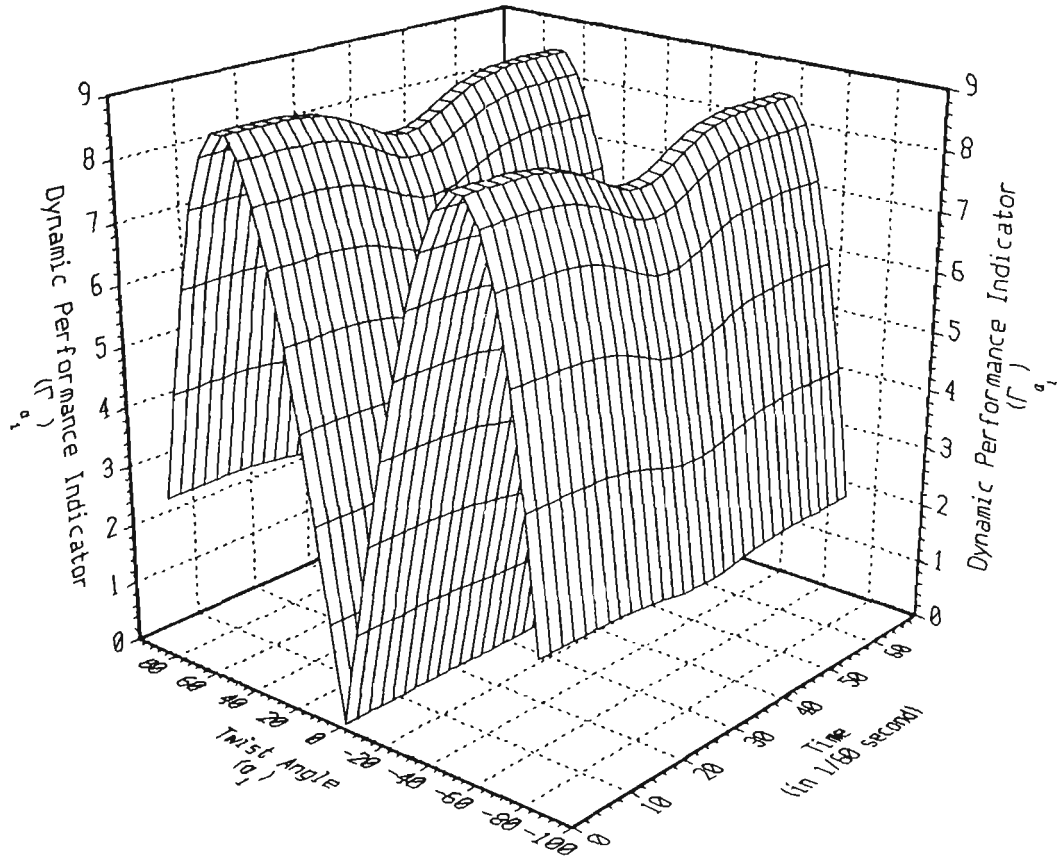


Figure 9-5: Dynamic performance indicator versus α_1 and time (trajectory 2).

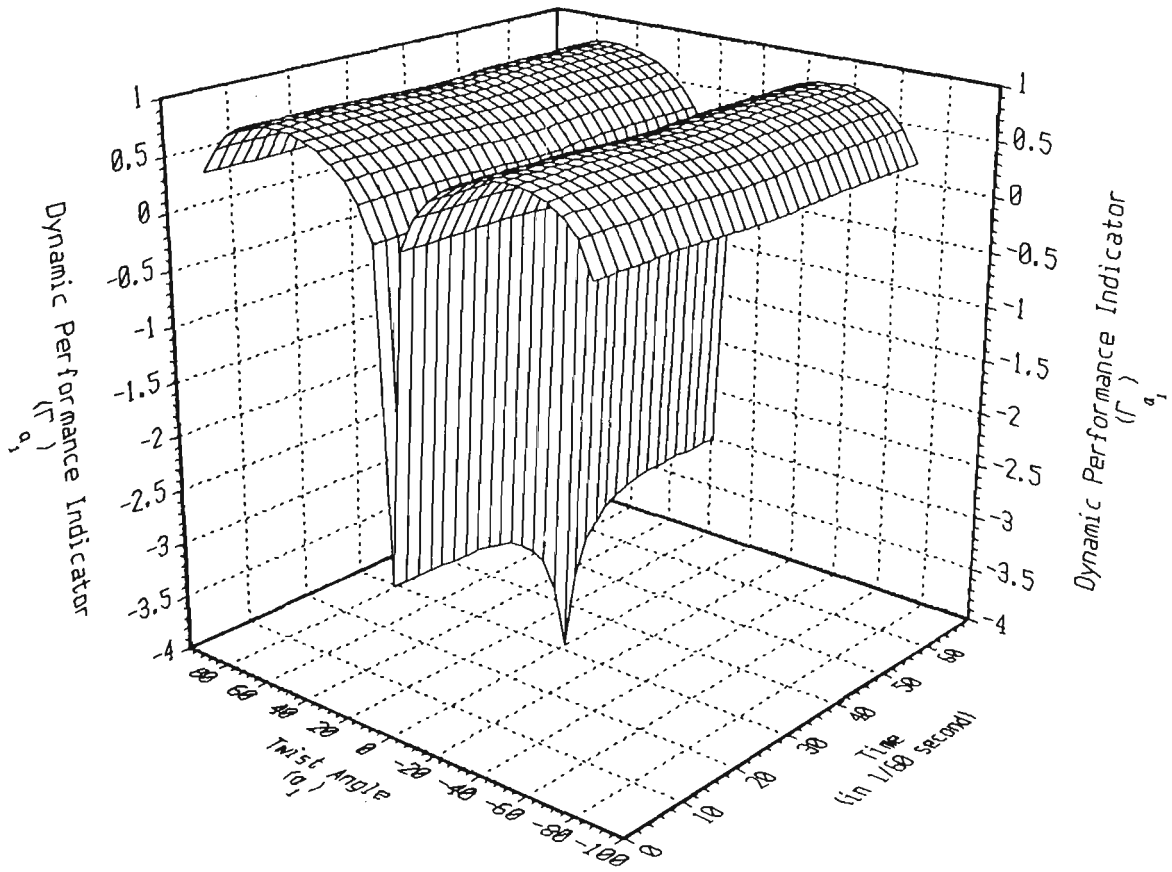


Figure 9-6: Logarithmic representation of Γ_{α_1} (trajectory 2).

9.4.1 Dynamic performance under the first motion trajectory

The sensitivity of the inertia matrix eigenvalues (Γ_{α_2}), was examined for different values of the twist angle α_2. The range of α_2 was as follows :

$$-90^o \leq \alpha_2 \leq 90^o$$

Also the robot's path trajectory of this case is schematically described in Figure 9-7 and its data tabulated in Table 9.3.

Since the arm was rotating fully stretched horizontally in a planar motion, the dynamic performance indicator was affected mainly by the changes in α_2. Also, Figure 9-8 shows that there was a high rate of change in the eigenvalues derivatives about $\alpha_2 = 0$. That was because the arm was fully stretched throughout the cycle. Therefore, any change in α_2 causes a significant reaction in the dynamic sensitivity due to the perpendicularity mentioned in section 9.3.1. It was found that the sensitivity increases about $\alpha_2 = 0^o$ and peaks at $\alpha_2 = \pm 45^o$ where it starts to change its direction. Furthermore, throughout the motion trajectory the arm was, theoretically, at singular positions.

It is clear from the obtained results that the most optimal value of α_2 is zero and the least optimal value is $\pm 45^o$. Also, the sensitivity's trend is generally uniform along the trajectory in which both q_2 and q_3 are constant, as demonstrated by the Logarithmic function of the sensitivity shown in Figure 9-9.

9.4.2 Dynamic performance under the second motion trajectory

In the second motion trajectory the first and second links were moving concurrently, each for 180°. During this trajectory the arm was continually

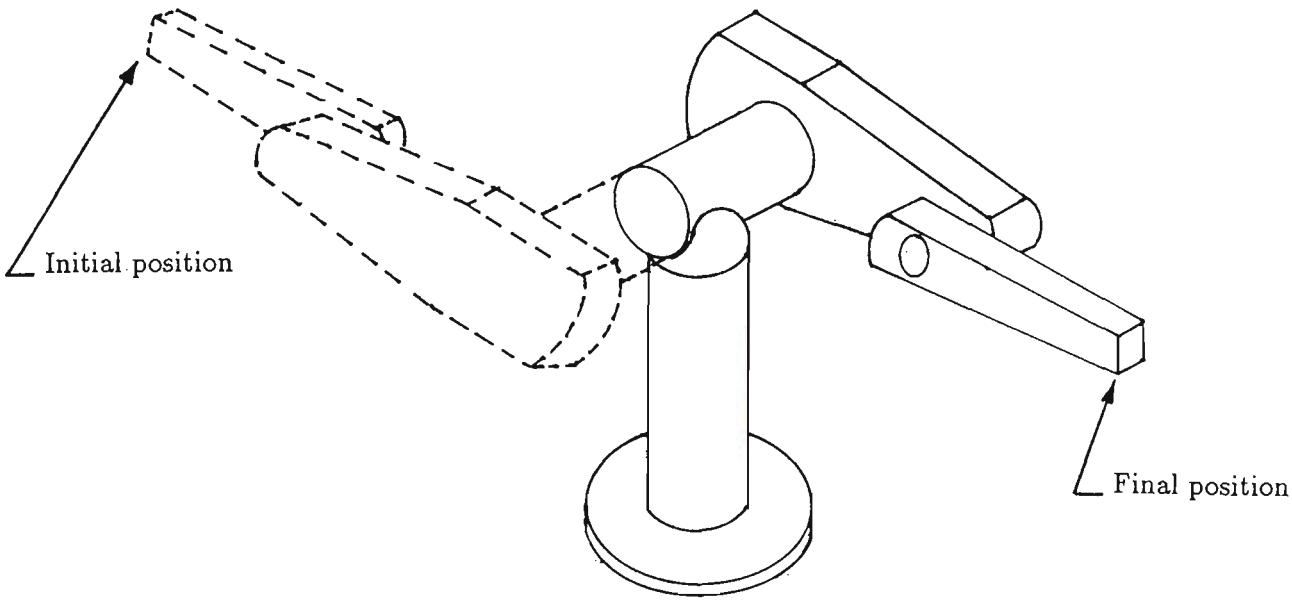


Figure 9-7: Schematic diagram of the first robot's path trajectory for Γ_{α_2}.

Total displacement of link 1	180^o
Starting position of link 1	-90^o
Total displacement of link 2	0.0
Starting position of link 2	0.0
Total displacement of link 3	0.0^o
Starting position of link 3	90^o

Table 9.3: Data for the first motion trajectory for Γ_{α_2}.

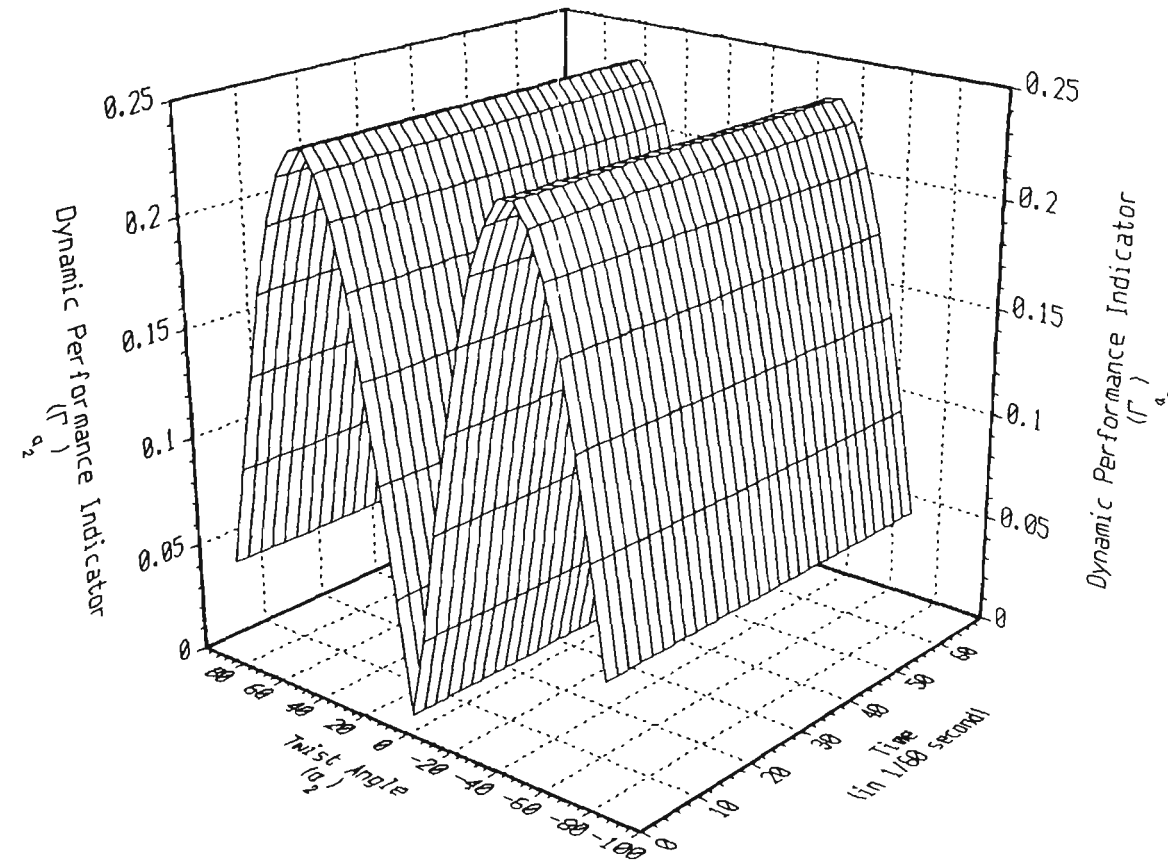


Figure 9-8: Dynamic performance indicator versus α_2 and time (trajectory 1).

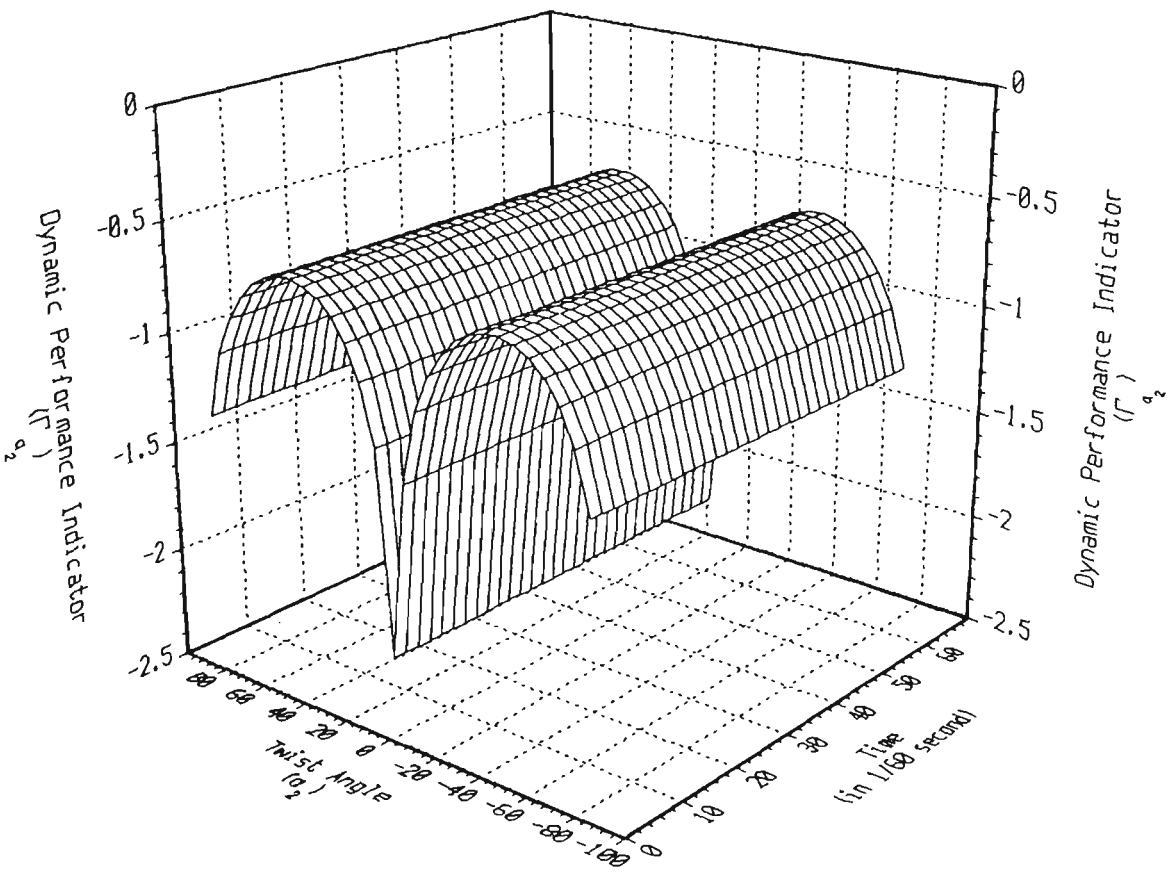


Figure 9-9: Logarithmic representation of Γ_{α_2} (trajectory 1).

stretched and moved from a vertically downward position to a vertically upward position. The motion's trajectory is schematically described in Figure 9-10 and its data are tabulated in Table 9.4.

In this case, the minimum sensitivity was attained under the following conditions :

- When $\alpha_2 = 0$.
- When $t = 0.5$ *sec.*.

A high rate of change in the robot's sensitivity Γ_{α_2} occurred immediately before and after the above-mentioned values for α_2 and time t, as shown in Figure 9-11. Also, in this motion's path trajectory the arm was fully stretched horizontally at $t = 0.5$ *sec.*.

However, from the sensitivity's logarithmic function shown in Figure 9-12, it was found that the minimum sensitivity occurs at $\alpha_2 = 0$.

The results shown in this experiment make it possible for a robot designer to choose the optimal workspace region together with suitable twist angles in order to obtain a robot's best possible dynamic performance.

9.5 Effect of a_2 on the Dynamic Performance

As shown in Table 6.1, the length of both the first and third links are zero, i.e. :

$$a_1 = a_3 = 0$$

Therefore, a test was conducted to examine the effect of the non-zero length (a_2) on the dynamic performance. The range of the tested a_2 values was :

$$0 \leq a_2 \leq 1 \ m$$

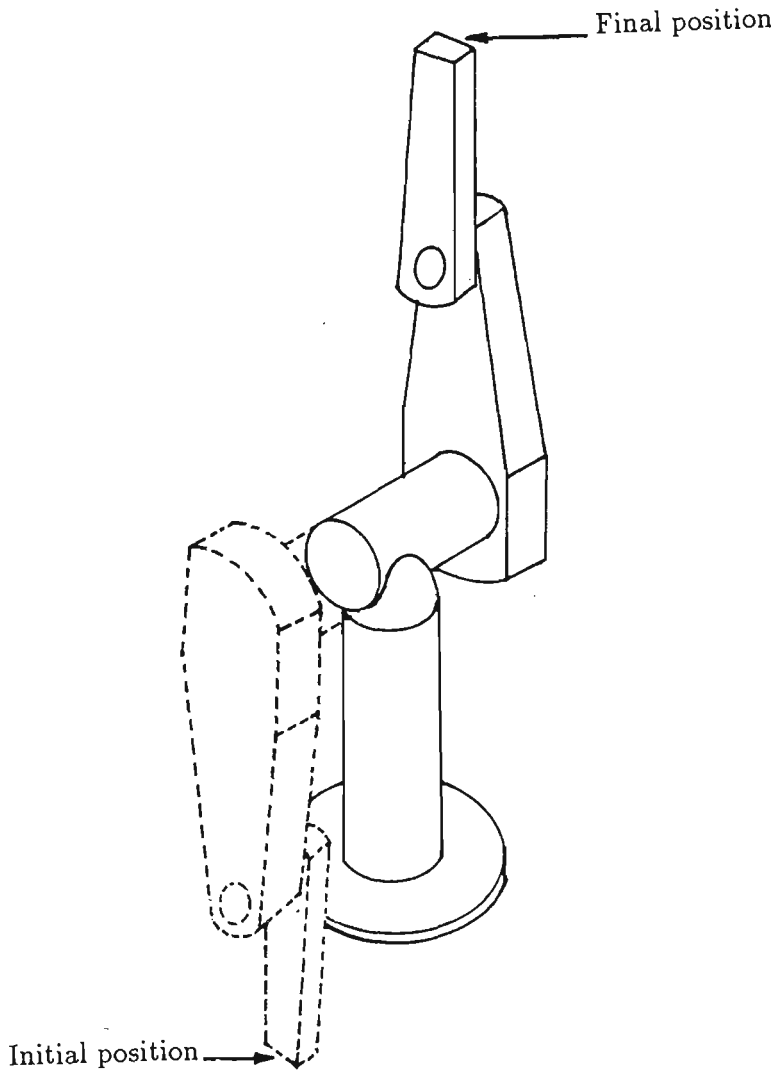


Figure 9-10: Schematic diagram of the second robot's path trajectory for Γ_{α_2}.

Total displacement of link 1	180^o
Starting position of link 1	-90^o
Total displacement of link 2	-180^o
Starting position of link 2	90^o
Total displacement of link 3	0.0^o
Starting position of link 3	90^o

Table 9.4: Data for the second motion trajectory for Γ_{α_2}.

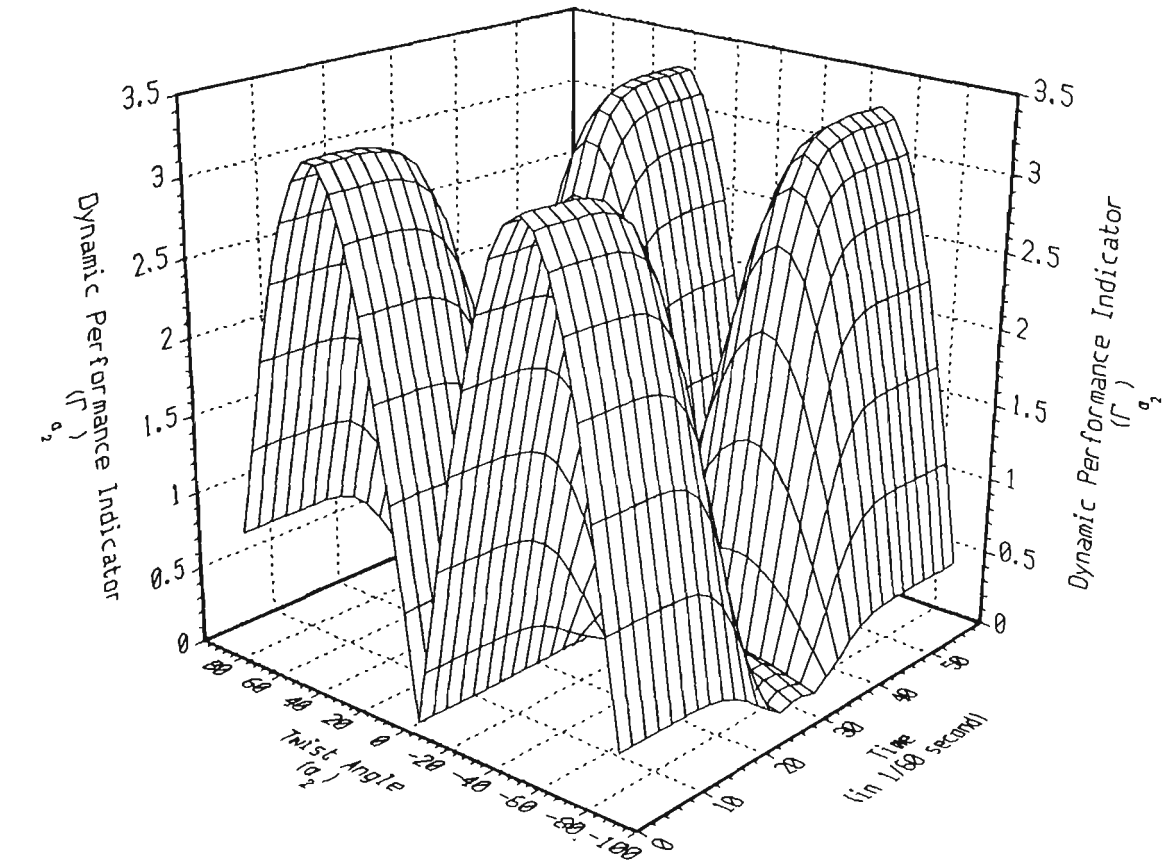


Figure 9-11: Dynamic performance indicator versus α_2 and time (trajectory 2).

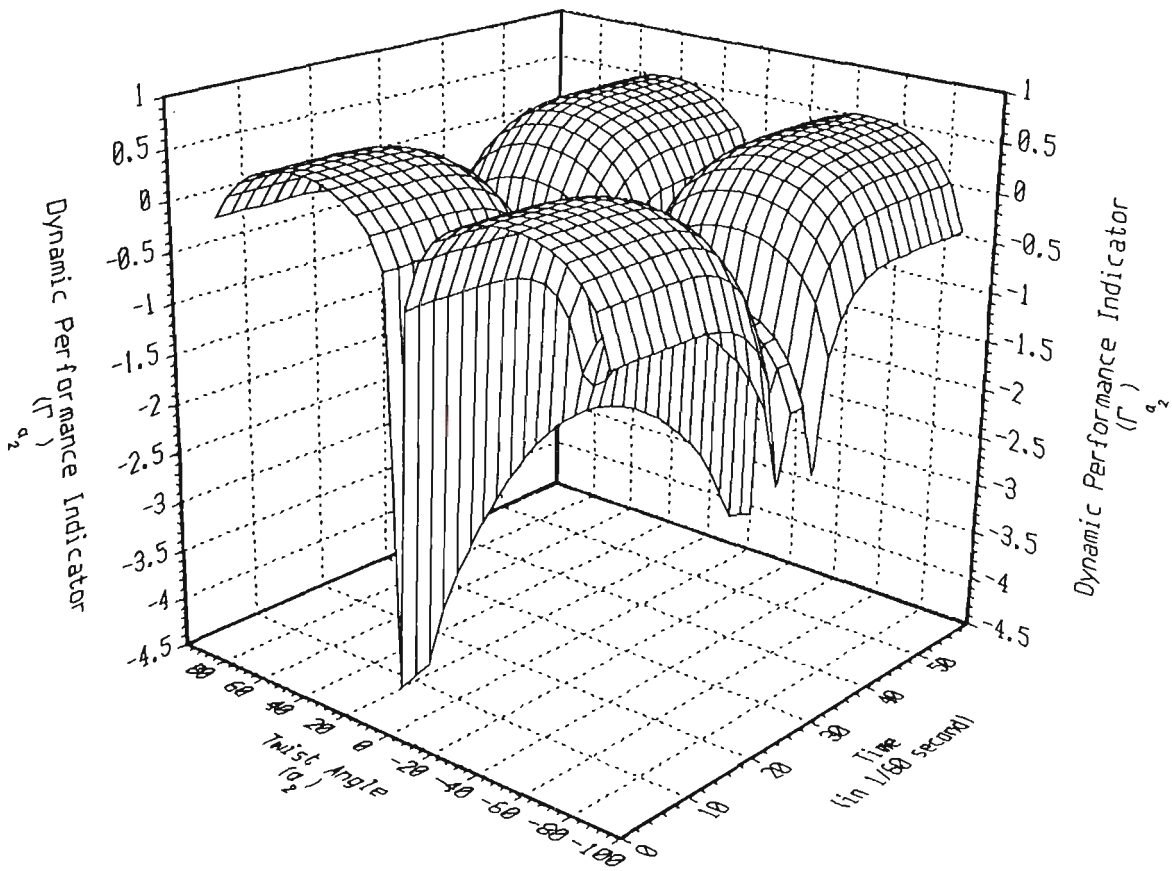


Figure 9-12: Logarithmic representation of Γ_{α_2} (trajectory 2).

The range was divided into 60 equal segments. During the investigation, a motion trajectory was applied to each length value. The motion trajectory data was similar to that described in Table 9.4. The results obtained of the dynamic performance indicator Γ_{a_2} were plotted versus both a_2 and the displacement as a function in time t, Figure 9-13.

It can be seen from the results obtained that the performance measure induced a mirror-like image about $t = 0.5$ sec. It can also be seen that at this particular configuration (arm fully stretched horizontally) the inertial eigenvalues are more sensitive to changes in the robot's arm length than at any other configuration on this trajectory. It should be also remarked that the arm was fully stretched throughout the cycle.

It can be briefly concluded here that the sensitivity of the dynamic performance to changes in the arm's length is greatly affected by the arm configuration in its course of action.

9.6 Summary

In this chapter a study was conducted to investigate the effect of a robot's geometrical parameters on its dynamic performance. Therefore, a performance measure indicator was introduced to give a quantifiable measure of the dynamic performance's response to changes in the links' geometrical parameters. The dynamic performance indicator was established using the logarithm of the Euclidean norm of the inertia matrix's eigenvalues. This was achieved by taking advantage of the symmetrical property of the inertia matrix.

The search was for a region in which there was minimum sensitivity to parameter changes over a specified path. However, as shown from the obtained results, large variations in the performance indicator do not, necessarily, mean a

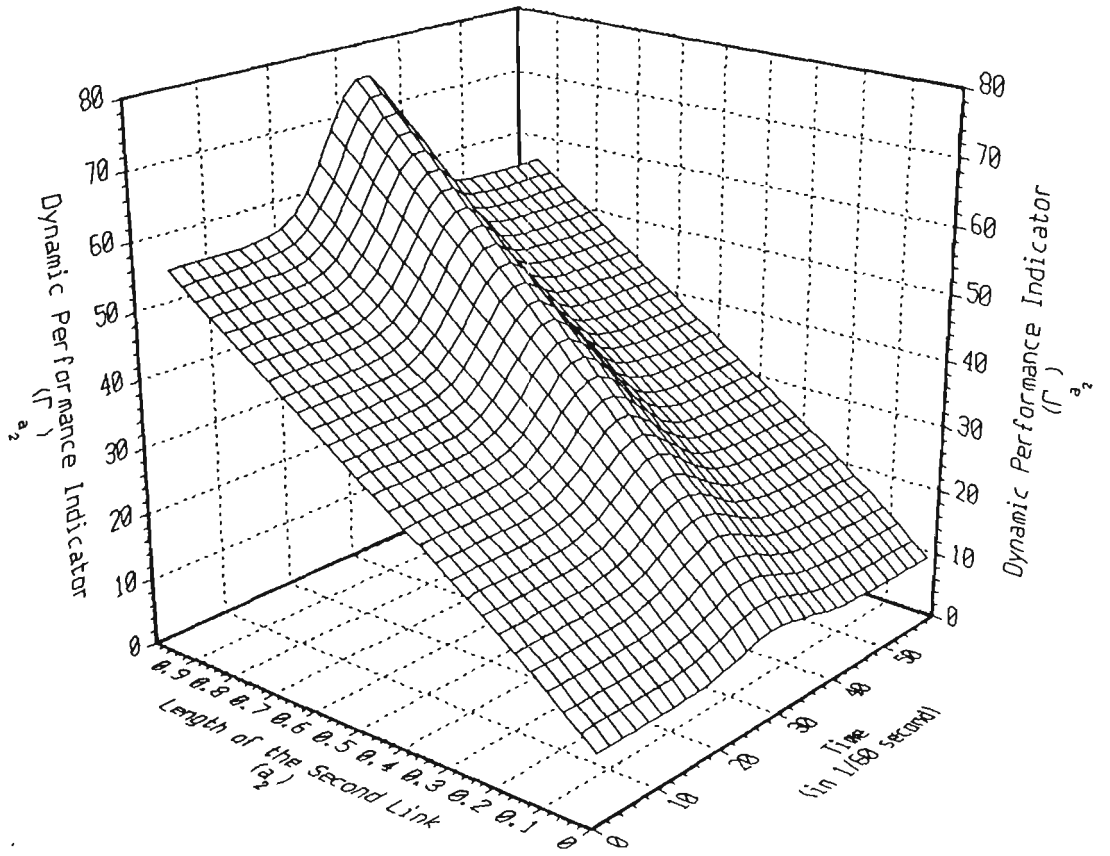


Figure 9-13: Dynamic performance indicator versus a_2 and time.

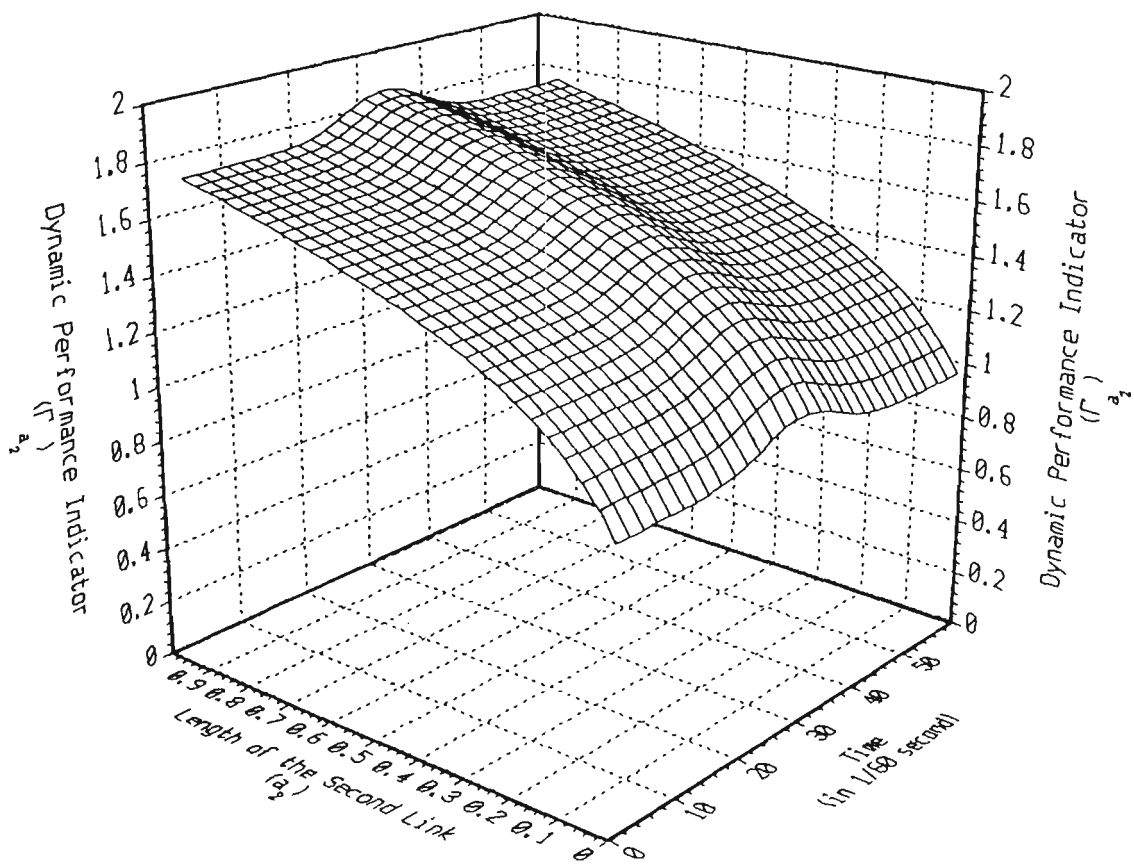


Figure 9-14: Logarithmic representation of Γ_{a_2}.

poor design from a workspace perspective. Therefore, a trade-off emerges between the choice of a geometrical parameter's value for optimal dynamic performance and the choice of a better work space. Therefore, a robot's designer has to weigh the merits of each option, depending on the intended application of the robot.

An important fact shown by this work is that the design of a robot for optimal dynamic performance can be achieved by taking into account two constraints :

1. geometrical parameter values such that the robot would have a workspace envelope necessary to execute its required task. Also, robots can be designed with variable geometrical parameters for dynamic versatility.

2. path constraints such that the robot would move through a low sensitivity corridor in its work envelope.

It was also revealed through the cases studied that the optimal value of a twist angle is zero. Furthermore, it was shown that α_1 and α_2 have most influence on the dynamic performance when their value is $\pm 45^o$. As demonstrated through the logarithmic representations of most cases studied, the performance sensitivity to changes in the twist angles is consistently high compared with the sensitivity to changes in the robot's configuration.

This new general design mechanism was established during the course of this research to give a robot designer a quantitative feedback, with graphical illustration, regarding the influence of changing a robot's geometrical parameters on its optimal dynamic performance.

Chapter 10

Conclusions and Recommendations

A systematic and comprehensive analytical study, addressing a range of issues on the dynamic performance of robot manipulators, was conducted in the course of this research. The two typical industrial robots considered in the study were a SCARA and an articulated robot arm (PUMA 560). Both iterative and closed-form dynamic models as well as the kinematic model of the PUMA 560 were developed and employed to facilitate the analysis of a robot's dynamic performance. The analysis was conducted to investigate the dynamic behaviour of robot manipulators under different operating conditions. These included :

Different velocity trajectory

Two trajectories with special characteristics were investigated. These were the *NC2* trajectory and a third order polynomial trajectory. The trajectories were used to investigate the dynamic coupling between links and the maximum velocity beyond which centripetal/Coriolis forces cannot be ignored. The analysis has shown quantitatively the extent of the effect of the non-linear velocity-related centripetal forces and coupling torques on a typical

robot for different velocity trajectories.

Time-varying or invariant payload operating conditions

An investigation was also conducted on robot dynamic characteristics under a time-varying payload operating condition. This showed the extent of the effect of payload variation on each link and also showed that the effect of payload variation is greater at near-maximum acceleration time than at near maximum-velocity time. That was found to be true for both types of robot investigated under NC2 and Polynomial trajectories.

This investigation was further conducted with the end effector moving in a prespecified Cartesian velocity trajectory using a trajectory generator and the reverse kinematic software developed in this thesis. The results showed that the maximum response to payload variation was 9.6% of the total required torque. However, it is anticipated that the response will be more significant when robots become more capable of carrying payloads of bigger proportion to their own weight.

Practical experiments were conducted for a robot moving under a time-varying payload and the results were compared with those of the simulated experiments. The trend of these results was found to support the results of relevant simulation experiments.

Since robot manipulators are being called upon to be more versatile in performing many new types of tasks involving higher speed and greater precision in advanced applications, an investigation was conducted on the possibility of eliminating the centripetal/Coriolis and gravitational forces by dynamically counterbalancing the robot's links. It was concluded that this technique is useful because it takes advantage of the short acceleration/deceleration periods in robots real applications. Also, the increase in the inertia loading of the first link can be met by a more powerful actuator which does not incur a dynamic penalty on the robot's

arm. However, dynamic balancing would be ineffective in job tasks where gravity and centripetal forces on a robot's arm are relatively small.

Furthermore, in pursuit of an in-depth analysis, a study was also conducted to examine the sensitivity of the robot's dynamic performance to changes in geometrical parameters. In order to quantify the dynamic performance in a unified form, a dynamic performance indicator was established. The indicator was based on the logarithm of the Euclidean norm of the inertia matrix eigenvalue derivatives with respect to the geometrical parameter under consideration. From the cases studied it was consistently found that smaller twist angles lead to a better performance criteria. Therefore it was recommended that twist angles be minimised where practical. This has prompted the suggestion that there are advantages in providing robots with adjustable twist angles.

The results obtained through the analysis of the topics covered in this dissertation can greatly expand the designer's understanding and reduce the design cycle-time. Computer generated plots which can compactly display extensive dynamic analysis information and graphics for illustrating robot input data and results have been used to provide enhanced control over the design process. These would give robot designers powerful tools to select rational design specifications and to interactively display the effect of changes in the operating conditions or the design parameters.

It should be also mentioned that this practical experimental setup and results can assist in future work aimed at optimising a robots design and performance.

For example, further research can be conducted to minimise gravitational and centripetal/Coriolis forces. As shown in Chapter 8, this problem can be addressed taking into account two conflicting constraints : a) the counterbalancing masses, and b) the limiting maximum acceleration.

masses, and b) the limiting maximum acceleration.

As mentioned earlier, robot manipulators to date can only carry up to 10% of their own weight. Therefore, researchers need to investigate methods of increasing a robot's lifting and manipulation efficiency. The work that has been done to date in this area such as the usage of a parallel and hybrid robot [74, 75], despite its promise for high precision does not yet show great potential due to the difficulty in modelling and controlling these devices. Therefore, these devices do not represent a better alternative to conventional serial robots.

Finally, it is hoped that the work on the topics covered in this dissertation has made a contribution towards the goal of helping designers produce robots with optimal dynamic performance.

Bibliography

[1] H. Seraji. An adaptive Cartesian control scheme for manipulators. In *Proceedings of the 1987 IEEE InternationalConference on Robotics and Automation*, pages 157–164, March 1987. Raleigh, NC.

[2] S. Dubowsky and D.T. DesForges. The application of model-referenced adaptive control to robotic manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, 101:193–200, September 1979.

[3] A.J. Koivo and T.H. Guo. Adaptive linear controller for robotic manipulators. *IEEE Transactions on Automatic Control*, AC-28(2):162–171, February 1983.

[4] G. G. Leininger. Adaptive control of manipulators using self-tuning methods. In *First International Symposium on Robotics Research, M. Brady and R. Paul Eds.*, 1984. Cambridge, MA:MIT Press.

[5] C.G.S. Lee and M.J. Chung. An adaptive control strategy for mechanical manipulators. *IEEE Transactions on Automatic Control*, AC-29(9):837–840, September 1984.

[6] Homayoun Seraji. Decentralized adaptive control of manipulators: Theory, simulation, and experimentation. *IEEE Journal of Robotics and Automation*, 5(2):183–201, April 1989.

[7] C. W. deSilva and J. Van Winssem. Least squares adaptive control for trajectory following robotics. *Trans. ASME*, 109:104–110, Summer 1987.

[8] J. Y. Han, H. Hemani, and S. Yurkovich. Nonlinear adaptive control of an n-link robot with unknown load. *The International Journal of Robotics Research*, 5:71–87, Fall 1987.

[9] K. Y. Lim and M. Eslam. Robust adaptive controller designs for robot manipulator systems. *IEEE Journal of Robotics and Automation*, RA-3:54–66, February 1987.

[10] J. J. Craig, P. Hsu, and S. S. Sasty. Adaptive control of mechanical manipulator. *The International Journal of Robotics Research*, 6:16–22, Summer 1987.

[11] John J. Craig. *Adaptive Control of Mechanical Manipulators.* Addison-Wesley, Reading, MA, 1987.

[12] J. K. Mills and A. A. Goldenberg. Robust control of robotic manipulators in the presence of dynamic parameter uncertainty. *ASME Journal of Dynamic Systems, Measurement and Control*, 111(3):444–451, September 1989.

[13] M. Vukobratović, D. Stokić, and N. Kirćanski. Towards nonadaptive and adaptive control of manipulation robots. *IEEE Transactions on Automatic Control*, AC-29(9):841–844, September 1984.

[14] A. H. Levis, S. I. Marcus, W. R. Perkins, P. Kokotovic, M. Athans, R. W. Brockett, and A. S. Willsky. Challenges to control: A collective view. *IEEE Transaction on Automatic Control*, AC-32:275–285, 1987.

[15] M. Vukobratović and D. Stokić. *Control of Manipulation Robots: Theory and Applications.* Springer-Verlag, New York, 1982.

[16] B.C. Kuo. *Automatic Control Systems.* Prentice-Hall, Englewood Cliffs, N.J., 1982.

[17] Richard P. Paul. *Robot Manipulators: Mathematics, Programming and Control.* MIT Press, Cambridge, MA, 1981.

[18] C. G. S. Lee, M. J. Chung, T. N. Mudge, and J. L. Turney. On the control of mechanical manipulators. In *Proceedings of 6th IFAC Symp. on Identification and System Parameter Estimation*, pages 1454–1459, June 1982. Washington, DC.

[19] Chae H. An, Christopher G. Atkeson, and John M. Hollerbach. *Model-Based Control of a Robot Manipulator.* MIT Press, Cambridge, MA, 1988.

[20] B.R. Markiewicz. Analysis of the computed-torque drive method and comparison with conventional position servo for a computer-controlled manipulator. Technical Report 33-601, Jet Propulsion Laboratory, Pasadena, CA, March 1973.

[21] A.K. Bejczy. Robot arm dynamics and control. Technical Report 33–669, Jet Propulsion Lab, Pasedena, CA, February 1974.

[22] Marc H. Raibert and B.K.P. Horn. Manipulator control using the configuration space method. *The Industrial Robot*, 5:69–73, June 1978.

[23] C. H. An, C. G. Atkeson, J. D. Griffiths, and j. M. Hollerbach. Experimental evaluation of feedforward and computer torque control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 165–168, April 1987. Raleigh, NC.

[24] Y. H. Chen. Robust computed torque schemes for mechanical manipulators: Nonadaptive versus adaptive. *ASME Journal of Dynamic Systems, Measurement and Control*, 113(2):324–327, June 1991.

[25] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, AC-25(3):468–474, June 1980.

[26] John M. Hollerbach and Gideon Sahar. Wrist partitioned inverse kinematic accelerations and manipulator dynamics. In *Proceedings of the* 1^{st} *International Conference on Robotics*, pages 152–161, March 1984. Atlanta, GA.

[27] Charles P. Neuman and Vassilios D. Tourassis. Robust discrete nonlinear feedback control for robotic manipulators. *Journal of Robotics Systems*, 4(1):115–143, 1987.

[28] Pradeep K. Khosla and Charles P. Neuman. Computational requirements of customized Newton-Euler algorithms. *Journal of Robotics Systems*, 2(3):309–327, Fall 1985.

[29] M. Sahba and D.Q. Mayne. Computer-aided design of nonlinear controllers for torque controlled robot arms. *IEE Proceedings*, 131, part D(1):8–14, January 1984.

[30] E. Freund and M. Syrbe. Control of industrial robots by means of microprocessors. In *Lecture Notes in Control and Information Science*, pages 167–185. Springer-Verlag, 1977. A.V. Balakrishnam and M. Thoma (editors).

[31] E. Freund. Fast nonlinear control for arbitrary pole-placement for industrial robots and manipulators. *The International Journal of Robotics Research*, 1(1):65–78, Spring 1982.

[32] Singh and W.J. Rough. Decoupling in a class of nonlinear systems by state-variable feedback. *ASME Journal of Dynamic Systems, Measurement and Control*, 94(4):323–329, December 1972.

[33] T.A.W Dwyer, G.K.F. Lee, and N. Chen. A terminal controller for a robotic manipulator arm with correction for perturbations. In *Proceeding of the Fourth Symposium on Robotics and Automation*, June 1984. Amesterdam, Holland.

[34] J.R. Hewit and J. Padovan. Decoupled feedback control of robot and manipulator arms. In *Proceedings of the Third International Symposium on Theory and Practice of Robots and Manipulators*, pages 251–266, September 1978. Udine, Italy.

[35] J.R. Hewit. Decoupled control of robot movement. *Journal of Electronics Letters*, 15(21):670–671, October 1979.

[36] D. Teaser and M. Thomas. Assessment for the mathematical formulation for design and digital control of programmable manipulator system. Technical report, Centre for Intelligent Machines and Robotics, University of Florida, Gainesville, FL, 1979.

[37] M. Thomas, H. C. Yuan-chou, and D. Teaser. Optimal actuator sizing for robotic manipulators based on local dynamic criteria. *ASME Journal of Mechanisms, Transmissions and Automation in Design*, 107:163–169, June 1985.

[38] M. Y. Ibrahim, K. R. Spriggs, and D. Jagger. Analysis of Robotics Reliability in Automotive Industry : a Practical Example. In *Proceeding of International Conference on Automation, Robotics and Computer Vision*, pages 351–355, September 1990. Singapore.

[39] M. Y. Ibrahim, C. D. Cook, and A. K. Tieu. Dynamic behaviour of a SCARA robot with links subjected to different velocity trajectories. *Robotica*, 6:115–121, 1988.

[40] M. Y. Ibrahim, C. D. Cook, and A. K. Tieu. Dynamics Characteristics of a SCARA Robot Subject to NC2 Velocity Trajectories With Different Payloads. *Robotics & Computer-Integrated Manufacturing*, 6(3):259–264, 1989.

[41] M. Yousef Ibrahim, C. D. Cook, and A. K. Tieu. Dynamic Performance Characteristics of an Articulated Robot Under Time-Varying Payload Condition—Part 1: Simulation Study. Submitted for Publication in the *International Journal of Mechatronics*, 1992.

[42] M. Yousef Ibrahim, C. D. Cook, and A. K. Tieu. Dynamic Performance Characteristics of an Articulated Robot Under Time-Varying Payload Condition—Part 2: Experimental Approach. Submitted for Publication in the *International Journal of Mechatronics*, 1992.

[43] M. Yousef Ibrahim, C. D. Cook, and A. K. Tieu. Effect of a Robot's Geometrical Parameters on its Optimal Dynamic Performance. In *Proceedings of IEEE Singapore International Conference on Intelligent Control and Instrumentation*, pages 820–825, February 1992. Singapore.

[44] Jack Denavit and R.S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, pages 215–221, June 1955.

[45] Richard P. Paul, Bruce Shimano, and Gordon E. Mayer. Kinematic control equations for simple manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11:449–455, June 1981.

[46] Richard P. Paul, Bruce Shimano, and Gordon E. Mayer. Differential kinematic control equations for simple manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11:456–460, June 1981.

[47] Phillip J. McKerrow. *Introduction to Robotics.* Addison-Wesley, Sydney, 1991.

[48] Parviz E. Nikravesh. *Computer-Aided Analysis of Mechanical Systems.* Prentice-Hall, Englewood Cliffs, N.J., 1988.

[49] F. N-Nagy and A. Siegler. *Engineering Foundations of Robotics.* Prentice-Hall, Englewood Cliffs, N.J., 1987.

[50] M. Vukobratović and V. Potkonjak. Contribution to computer construction of active chain models via lagrangian form. *Journal of Applied Mechanics*, pages 181–185, March 1979.

[51] J. J. Uicker, Jr. Dynamic force analysis of spatial linkages. In *Proceedings of ASME Mechanisms Conference*, October 1966. paper No. 66-Mech.-1, Lafayette, IN.

[52] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul. On-line computational scheme for mechanical manipulators. In 2^{nd} *IFAC/IFIP Symposium Information Problems, Manufacturing Technology*, October 1979. Stuttgart, Germany.

[53] R. C. Waters. Mechanical arm control. Technical Report Memo. 549, Artificial intelligence lab., MIT, October 1979.

[54] John M. Hollerbach. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-10(11):730–736, November 1980.

[55] K. Fu, R. C. Gonzalez, and C. S. G. Lee. *ROBOTICS: Control, Sensing, Vision, and Intelligence.* McGraw-Hill, New York, 1987.

[56] C. S. G. Lee and M. Ziegler. Geomertic approach in solving inverse kinematics of PUMA robots. *IEEE Transactions of Electrical Engineering and Computer Science*, AES-20(6), November 1984.

[57] John J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley, Reading, MA, 1986.

[58] C. S. George Lee. Robot arm kinematics, dynamics, and control. *Computer*, 15(12):62–80, September 1982.

[59] D. Myers and D. Gordon. Kinemaic equations for industrial manipulators. *The International Journal of Robotics Research*, pages 162–165, September 1982.

[60] M. S. Mujtoba. Discussion of trajectory calculation methods. Technical Report AIM 285.4, Stanford University, Artificial Intelligence Laboratory, 1977.

[61] Michael Brady, John M. Hollerbach, Timothy L. Johnson, Thomas Lozano-Perez, and Matthew T. Mason. *Robot Motion: Planning and Control.* MIT Press, Cambridge, MA, 1982.

[62] J. L. Meriam. *Engineering Mechanics: Dynamics.* John Wiley, New York, 1980.

[63] F. P. Beer and E. R. Johnston. *Vector Mechanics for Engineers: Dynamics.* McGraw-Hill, New York, 1977.

[64] G. J. Borce. *FORTRAN 77 and Numerical Methods for Engineers.* PWS Engineering, 1985.

[65] CONDEC Unimation Robotics, Danbury, CT 06810. *User's Guide to VAL,* June 1980.

[66] Laboratory Technologies Corporation, Wilmington, MA 01887. *LABTECH NOTEBOOK MANUAL,* October 1989.

[67] Y. C. Tsai and A. H. Soni. The effect of link parameter on the working space of general 3R robot arms. *Mechanism and Machine Theory,* 19(1):9–16, 1984.

[68] K. C. Gupta and B. Roth. Design considerations for manipulator workspace. *ASME Journal of Mechanical Design,* 104:704–711, October 1982.

[69] Tsuneo Yoshikawa. Dynamic manipulability of robot manipulators. *Journal of Robotics Systems,* 2(1):113–124, 1985.

[70] F.M. Stein. *Introduction to Matrices and Determinants.* Wadsworth, Belmont, California, 1967.

[71] H. Schneider and G.P. Barker. *Matrices and Linear Algebra.* Holt, Rinehart and Winston, New York, 1973.

[72] Erwin Kreyszig. *Advanced Engineering Mathematics.* John Wiley, New York, 1972.

[73] P.V. O'Neil. *Advanced Engineering Mathematics.* Wadsworth, Belmont, California, 1983.

[74] M. Mohamed. *Instantaneous Kinematics of Joint Displacement Analysis of Fully-Parallel Robot Devices.* PhD thesis, University of Florida, Gainesville, Florida, 1983.

[75] D. Tesar and D. J. Cox. The dynamic modelling and command signal formulation for parallel multiparameter robotic devices. Technical Report DOD Grant DE-AC05-79ER-10013, Center for Intelligent Machines and Robotics, University of Florida, Gainesville, Florida, September 1981.

Appendix A

Calculation of the Inertia Matrices of the SCARA Robot

In order to achieve reliable results for the analysis of the SCARA robot dynamic performance, practicable set of the links inertia matrices should be first developed. In this appendix the inertia matrices have been calculated for each link. Also, elements of the vectors representing the center of gravity (C.G.) for each link have been evaluated.

In this research the SCARA robot actuators are represented in the simulation by point masses at their C.G.'s locations. The links have the dimensions as indicated on Figures A.1, A.2 and A.3. The values of the point masses parameters are as shown in Table A.1.

A.1 Inertia Calculation of the First Link

The cross sectional dimension of link 1 and link 2 are shown in Figures A.1, A.2 and A.3. These hollowed rectangular tubes of both links one and two, are considered to be made of aluminium alloy which has a density of 2.17E3 Kg/m^3. The third link is considered to be made of solid steel bar, which

Parameter	*Description*	*Value*
m1	end point masses (e.g. motors, bearings,...etc.)	8 Kg
m2		1 Kg
m3		1.5 Kg
m4		1.5 Kg
m5	Parasitic masses (wiring, tubes,...etc.)	1.5 Kg
m6		0.5 Kg

Table A.1: Data of the SCARA robot's mass parameters.

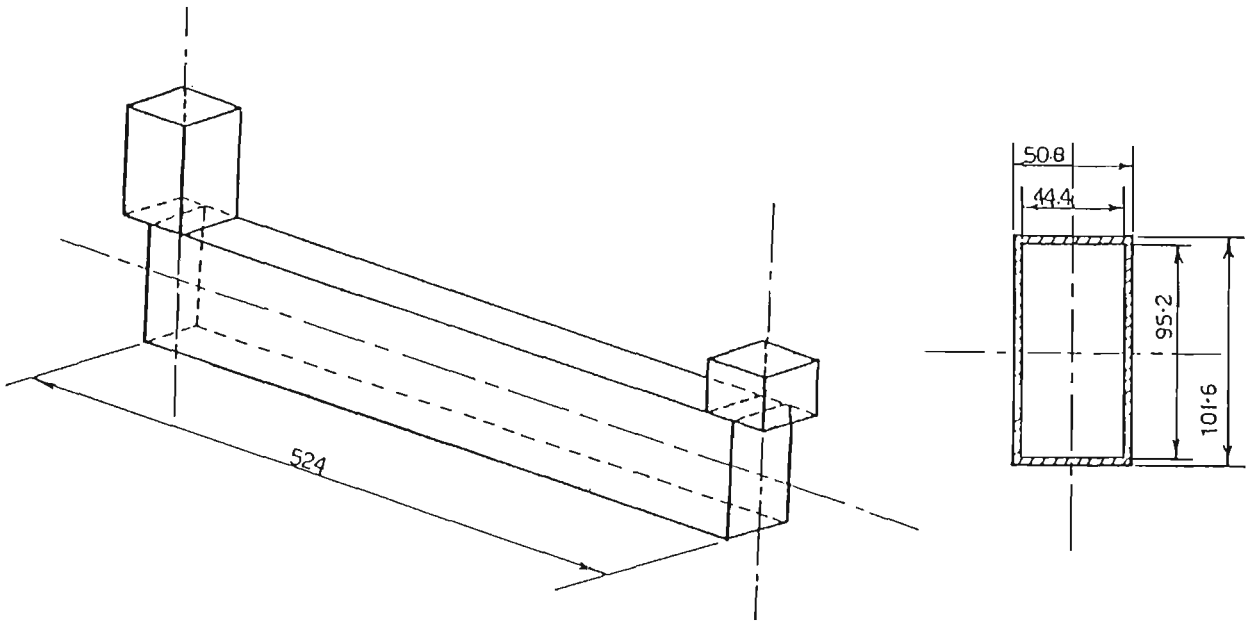


Figure A.1: Schematic diagram of SCARA's first link.

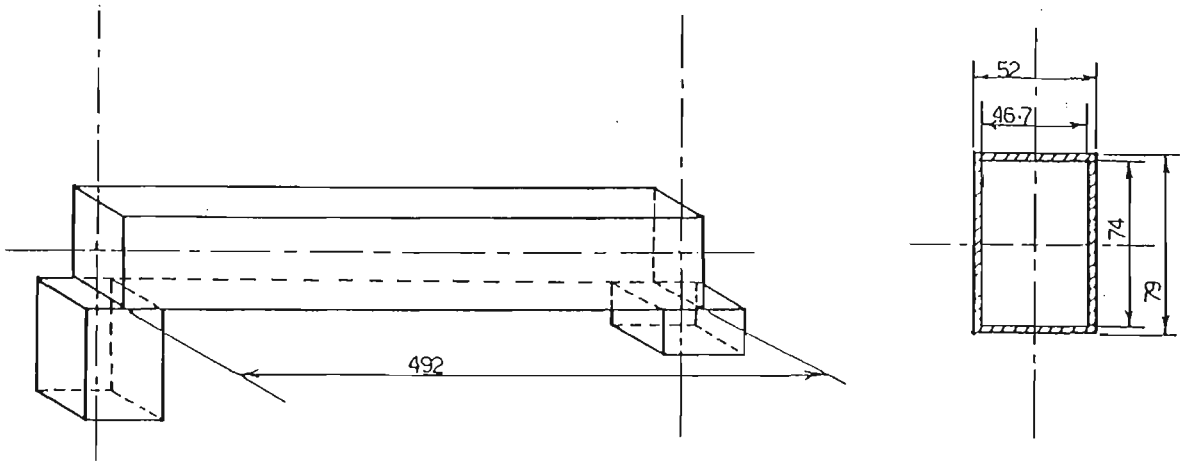


Figure A.2: Schematic diagram of SCARA's second link.

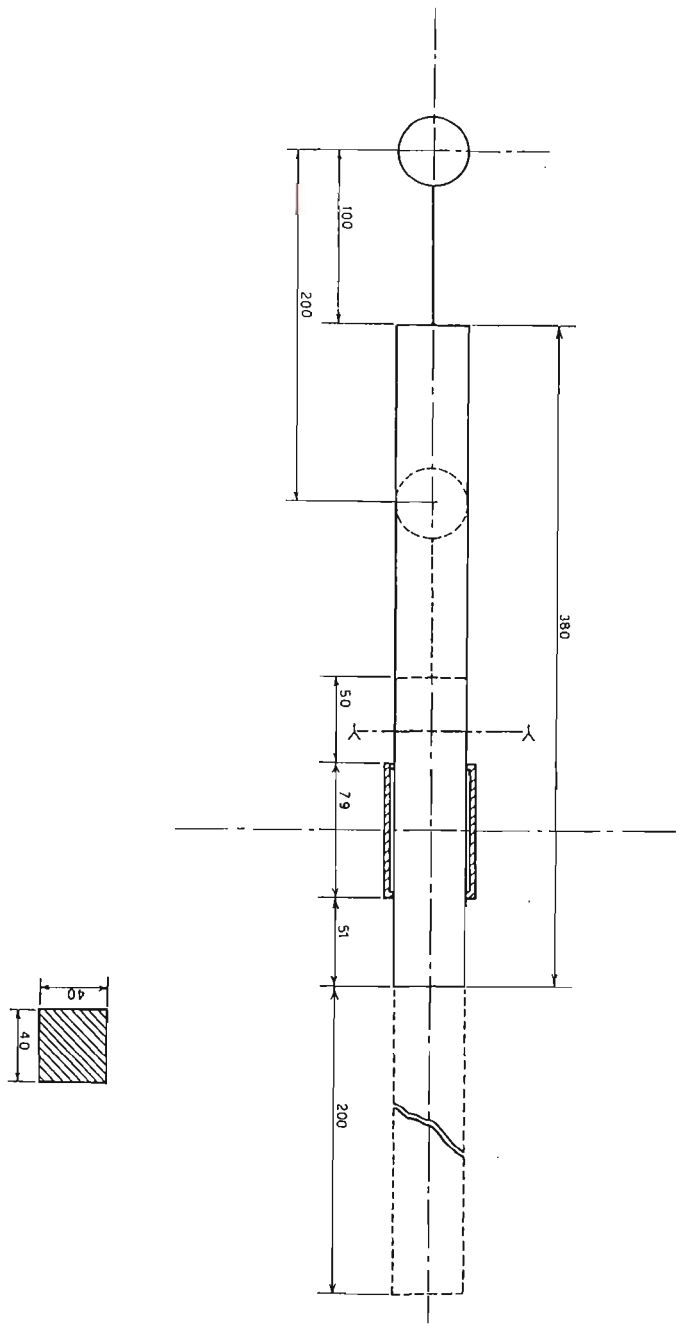


Figure A.3: Schematic diagram of SCARA's third link.

has a density of 7.8E3 Kg/m^3.

From the given data the mass of the main body of the first link can be calculated as follows :

$$\text{mass } m = \text{Volume } v \times \text{Density } \rho \quad \text{(A.1)}$$

$$m_{outer} = (0.0508 \times 0.1016)l_1 \times \left(2.71 \times 10^3\right) = (13.986l_1) \text{ Kg} \quad \text{(A.2)}$$

Similarly

$$m_{inner} = (0.0444 \times 0.0952)l_1 \times \left(2.71 \times 10^3\right) = (11.454l_1) \text{ Kg} \quad \text{(A.3)}$$

Where :

$m_{outer} \stackrel{def}{=}$ mass of link's outer dimension.

$m_{inner} \stackrel{def}{=}$ mass of link's inner dimension.

Therefore :

$$\begin{aligned} m_{link1} &= m_{outer} - m_{inner} = (13.986 - 11.454)l_1 \\ &= (2.532l_1) \text{ Kg} \\ &= 2.532 \times .524 = 1.327 \text{ Kg} \end{aligned} \quad \text{(A.4)}$$

A.1.1 Centre of gravity of link 1

In order to calculate the inertia matrix of the first link, its centre of gravity should be evaluated first. If it is assumed that the C.G. lies at point ϵ which is at a distance δ from the link's end.

Then the C.G. can be found by taking the moment about ϵ (as function of l_1) :

$$\begin{aligned} \sum M_\epsilon &= 1 \times \delta + (0.5l_1 + \delta)(1.5 + 2.532l_1) + 8(l_1 + \delta) = 0.0 \\ &= \delta + 0.75l_1 + 1.266(l_1)^2 + 1.5\delta + 2.532l_1\delta + 8l_1 + 8\delta = 0.0 \end{aligned}$$

i.e.

$$\delta(1 + 1.5 + 2.532l_1 + 8) = -\left(0.75l_1 + 1.266{l_1}^2 + 8l_1\right)$$

Therefore (A.5)

$$\begin{aligned} \delta &= -\frac{0.75l_1 + 1.266{l_1}^2 + 8l_1}{10.5 + 2.532l_1} \\ &= -\frac{8.75l_1 + 1.266{l_1}^2}{10.5 + 2.532l_1} \\ &= -0.0417\text{m} \qquad \text{(substitute for } l_1\text{=0.524 m)} \end{aligned} \quad \text{(A.6)}$$

Thus, the x, y and z coordinates of the first link's C.G. with respect the

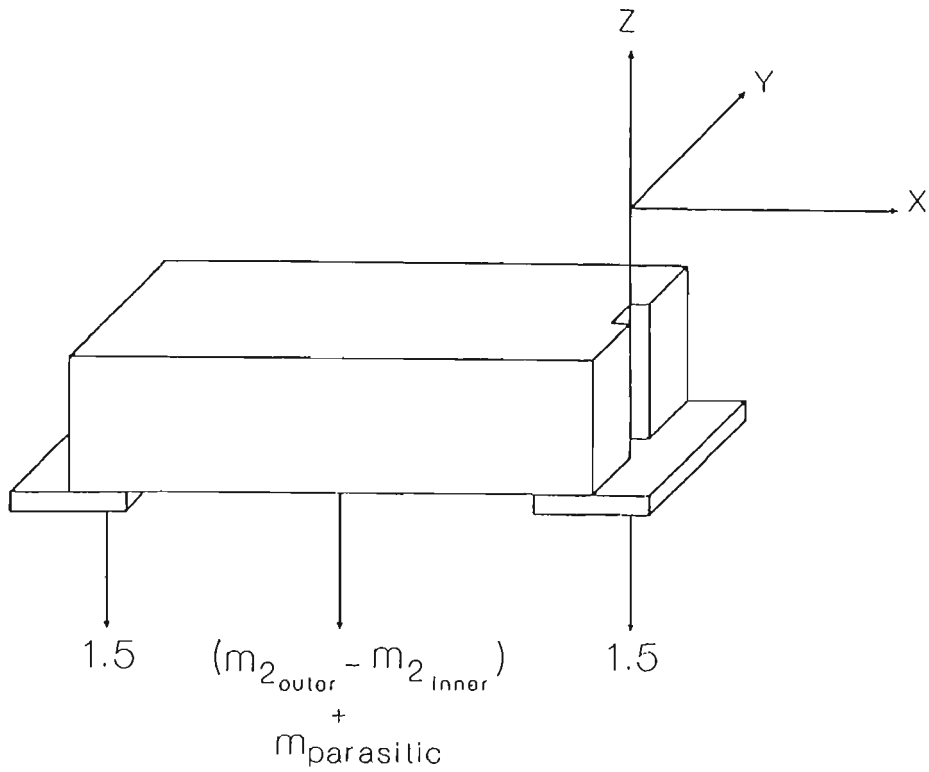


Figure A.4: Mass distribution of the first link.

link's fixed coordinate frame are :

$$\left.\begin{array}{rcl} x & = & -0.0417\ m \\ y & = & 0.0 \\ z & = & 0.0\ m \end{array}\right\} \tag{A.7}$$

A.1.2 Moment of inertia calculation

For the solid rectangular beam shown in Figure A.5, the following symbols are defined :

$X'Y'Z'$ is the coordinate frame of an arbitrary transverse slice of the beam.

$X_oY_oZ_o$ is the coordinate frame of the centre of gravity (C.G.) of the beam.

XYZ is the fixed coordinate frame of link 1, about which link 2 moves.

by taking the moment of inertia of the infinitismal transverse plane slice, indicated on Figure A.5, about Y'-axis :

$$dI_{Y'Y'} = (\rho dx)\left(\frac{1}{12}ab^3\right)$$

Also, by taking the moment of inertia of the same slice about Z' axis :

$$dI_{Z'Z'} = (\rho dx)\left(\frac{1}{12}a^3b\right)$$

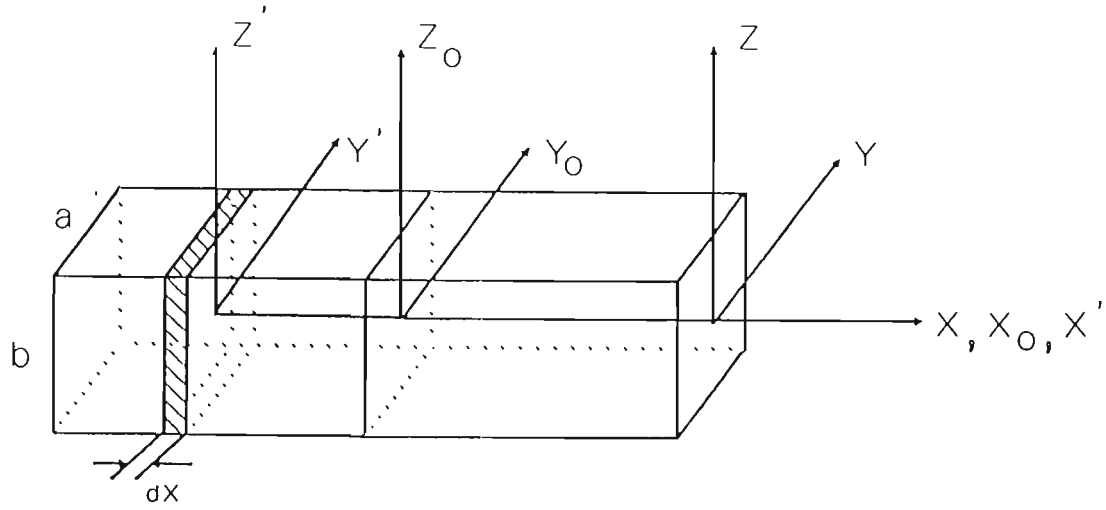


Figure A.5: Moment of inertia axes of a solid rectangular beam.

Since XX is collinear with $X'X'$, then;

$$dI_{XX} = dI_{Y'Y'} + I_{Z'Z'} = (\rho dx)\left(\frac{ab}{12}\right)(a^2 + b^2)$$

The above expression can be integrated to obtain the total beam moment of inertia about X-axis :

$$\begin{aligned} I_{XX} &= I_{X_oX_o} = \int dI_{XX} \\ &= \frac{\rho ab}{12}(a^2 + b^2)\int_0^l dx \\ &= \frac{1}{12}m(a^2 + b^2) \end{aligned}$$

Similarly

$$I_{Y_oY_o} = \frac{1}{12}m(b^2 + l^2)$$

Therefore :

$$I_{YY} = I_{Y_cY_o} + m\left(\frac{l}{2}\right)^2 = \frac{1}{12}m(b^2 + 4l^2)$$

Also :

$$I_{ZZ} = \frac{1}{12}m(a^2 + 4l^2)$$

Using equations A.2 and A.3 the moment of inertia of the link can be obtained by the substracting the inner dimension inertia from the outer dimension inertia. So, by calculating these moments as function of the link length l_1, the following results can be obtained :

$$I_{XX_{outer}} = \frac{1}{12}m(a^2 + b^2)_{outer}$$

$$= \frac{1}{12}(13.986l_1)\left[(0.0508)^2 + (0.1016)^2\right]$$
$$= 0.0150384l_1$$

Also :

$$I_{XX_{inner}} = \frac{1}{12}m\left(a^2 + b^2\right)_{inner}$$
$$= \frac{1}{12}(11.454l_1)\left[(0.0444)^2 + (0.0952)^2\right]$$
$$= 0.0105322l_1$$

Therefore :

$$I_{XX_{link}} = I_{XX_{outer}} - I_{XX_{inner}}$$
$$= 0.0150384l_1 - 0.0105322l_1$$
$$= 0.0045062l_1 \qquad \text{(A.8)}$$

Also, for calculating I_{YY} of the first link :

$$I_{YY} = I_{Y_oY_o} + m\left(\frac{l}{2}\right)^2 \qquad \text{(A.9)}$$

Calculating $I_{YY_{outer}}$:

$$I_{Y_oY_{o_{outer}}} = \frac{1}{12}m\left(b^2 + l^2\right)_{outer}$$
$$= \frac{1}{12}(13.986l_1)\left[(0.1016)^2 + l_1^2\right]$$
$$= 0.012031l_1 + 1.1655l_1^3 \qquad \text{(A.10)}$$

By substitution from (A.10) in (A.9) :

$$I_{YY_{outer}} = \left(0.012031l_1 + 1.1655l_1^3\right) + (13.986l_1)\left(\frac{l_1^2}{4}\right)$$
$$= 0.012031l_1 + 4.6625l_1^3 \qquad \text{(A.11)}$$

Similarly, for $I_{YY_{inner}}$:

$$I_{Y_oY_{o_{inner}}} = \frac{1}{12}m\left(b^2 + l^2\right)_{inner}$$
$$= \frac{1}{12}(11.545l_1)\left[(0.0592)^2 + l_1^2\right]$$
$$= 0.003345l_1 + 0.9545l_1^3 \qquad \text{(A.12)}$$

By substitution from (A.10) in (A.9) :

$$I_{YY_{inner}} = \left(0.003345l_1 + 0.9545l_1^3\right) + (11.454l_1)\left(\frac{l_1^2}{4}\right)$$
$$= 0.003345l_1 + 3.818l_1^3 \qquad \text{(A.13)}$$

$$\text{Since} \quad I_{YY} = I_{YY_{outer}} - I_{YY_{inner}}$$

Then from (A.11) and (A.13) :

$$\begin{aligned} I_{YY} &= \left(0.012031 l_1 + 4.6625 l_1^3\right) - \left(0.003345 l_1 + 3.818 l_1^3\right) \\ &= 0.008686 l_1 + 0.8445 l_1^3 \end{aligned} \tag{A.14}$$

$$I_{ZZ} = I_{Z_oZ_o} + m\left(\frac{l}{2}\right)^2 \tag{A.15}$$

Calculating $I_{ZZ_{outer}}$:

$$I_{Z_oZ_{o_{outer}}} = \frac{1}{12} m \left(a^2 + l^2\right)_{outer} \tag{A.16}$$

By substitution from (A.16) in (A.15) :

$$\begin{aligned} I_{ZZ_{outer}} &= \frac{1}{12} m \left(a^2 + 4l^2\right)_{outer} \\ &\quad \frac{1}{12}(13.986 l_1)\left[(0.0508)^2 + 4l^2\right] \\ &= 0.0030076 l_1 + 4.6625 l_1^3 \end{aligned} \tag{A.17}$$

Similarly, for $I_{ZZ_{inner}}$:

$$I_{Z_oZ_{o_{inner}}} = \frac{1}{12} m \left(a^2 + l^2\right)_{inner} \tag{A.18}$$

By substitution from (A.18) in (A.15) :

$$\begin{aligned} I_{ZZ_{inner}} &= \frac{1}{12} m \left(a^2 + 4l^2\right)_{inner} \\ &\quad \frac{1}{12}(11.454 l_1)\left[(0.0444)^2 + 4l^2\right] \\ &= 0.00188 l_1 + 43.818 l_1^3 \end{aligned} \tag{A.19}$$

$$\text{Since} \quad I_{ZZ} = I_{ZZ_{outer}} - I_{ZZ_{inner}}$$

Then from (A.17) and (A.19) :

$$\begin{aligned} I_{ZZ} &= \left(0.0030076 l_1 + 4.6625 l_1^3\right) - \left(0.00188 l_1 + 43.818 l_1^3\right) \\ &= 0.001127 l_1 - 0.842 l_1^3 \end{aligned} \tag{A.20}$$

Also, using the following axes translation rule :

$$I_{XY} = I_{X_oY_o} + m_{link}\ dx\ dy \tag{A.21}$$

where :

$I_{XY} \stackrel{\text{def}}{=}$ The product of inertia about w.r.t. the link's, fixed, coordinate frame.

$I_{X_oY_o} \stackrel{\text{def}}{=}$ The product of inertia about w.r.t. the C.G. coordinate frame.

$dx\ dy \stackrel{\text{def}}{=}$ The x and y coordinates of the C.G. point in the link's fixed coordinate frame.

The product of inertia of the first link can then be calculated as follow :

$$\begin{aligned} I_{XY} &= I_{X_oY_o} + m_{link1}\ dx\ dy \\ &= 0.0 + m_{link1} \times \left(\frac{l_1}{2}\right) \times 0.0 \\ &= 0.0 \end{aligned} \tag{A.22}$$

$$\begin{aligned} I_{XZ} &= I_{X_oZ_o} + m_{link1}\ dx\ dz \\ &= 0.0 + m_{link1} \times \left(\frac{l_1}{2}\right) \times 0.0 \\ &= 0.0 \end{aligned} \tag{A.23}$$

$$\begin{aligned} I_{YZ} &= I_{Y_oZ_o} + m_{link1}\ dy\ dz \\ &= 0.0 + m_{link1}(0.0)(0.0) \\ &= 0.0 \end{aligned} \tag{A.24}$$

Moment of inertia of point masses at fixed ends :

$$I_{XX} = 0.0 \tag{A.25}$$
$$I_{YY} = m \times l_1^2 = 8l_1^2 \tag{A.26}$$
$$I_{ZZ} = m \times l_1^2 = 8l_1^2 \tag{A.27}$$
$$I_{XY} = 0.0 \tag{A.28}$$
$$I_{XZ} = 0.0 \tag{A.29}$$
$$I_{YZ} = 0.0 \tag{A.30}$$

Moment of inertia due to the parasitic masses m_5 :

$$I_{XX} = 0.0 \tag{A.31}$$
$$I_{YY} = 1.5l_1^2 \tag{A.32}$$
$$I_{ZZ} = 1.5l_1^2 \tag{A.33}$$
$$\tag{A.34}$$

Also, for the same parasitic masses :

$$I_{XY} = I_{XZ} = I_{YZ} = 0.0 \tag{A.35}$$

Therefore, the total inertia of the first link can be calculated as follows :

$$I_{XX} = I_{XX_{link1}} + I_{XX_{point\ masses}} + I_{XX_{parasitic\ masses}}$$

From (A.8),(A.25) and (A.31) :

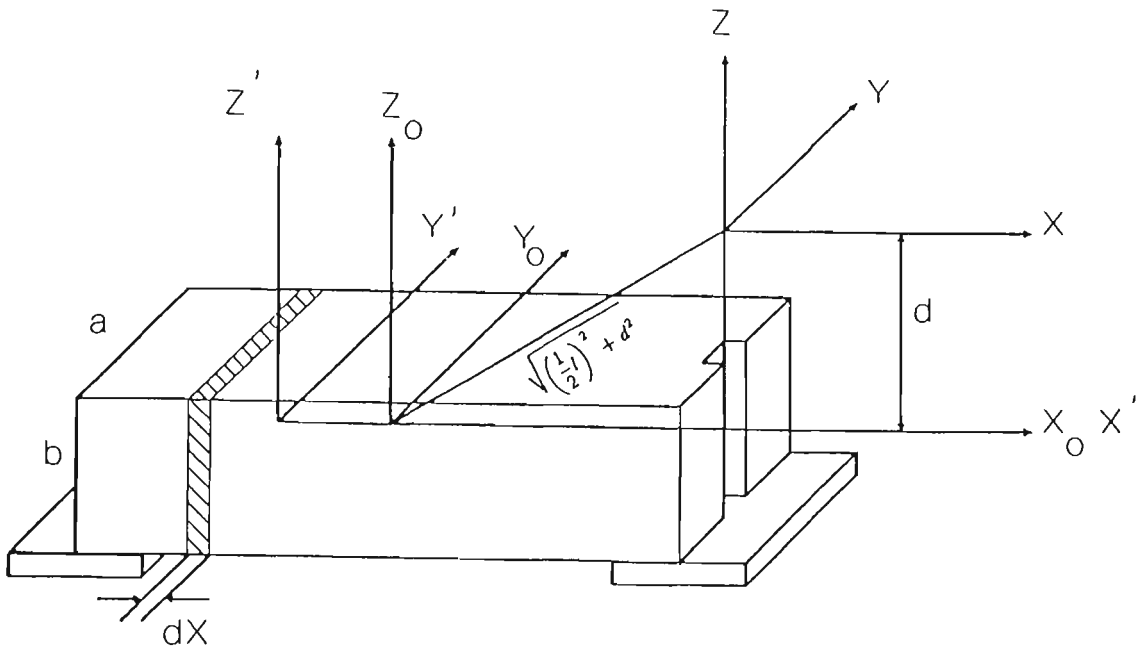


Figure A.6: Second link inertial parameters.

$$I_{XX} = 0.0045062l_1 + 0.0 + 0.0 = 0.0045062l \quad \text{(A.36)}$$

$$I_{YY} = I_{YY_{link1}} + I_{YY_{point\ masses}} + I_{YY_{parasitic\ masses}}$$

From (A.14),(A.26) and (A.32) :

$$\begin{aligned} I_{YY} &= \left(0.008686l_1 + 0.8445l_1^3\right) + 8l_1^2 + 1.5l_1^2 \\ &= 0.008686l_1 + 9.5l_1^2 + 0.8445l_1^3 \end{aligned} \quad \text{(A.37)}$$

$$I_{ZZ} = I_{ZZ_{link1}} + I_{ZZ_{point\ masses}} + I_{ZZ_{parasitic\ masses}}$$

From (A.20),(A.27) and (A.33) :

$$\begin{aligned} I_{ZZ} &= \left(0.001127l_1 + 0.842l_1^3\right) + 8l_1^2 + 1.5l_1^2 \\ &= 0.001127l_1 + 9.5l_1^2 + 0.842l_1^3 \end{aligned} \quad \text{(A.38)}$$

The resultant product inertia for link 1 :

$$I_{XY} = I_{XZ} = I_{YZ} = 0.0 \quad \text{(A.39)}$$

A.2 Inertia Calculation of the Second Link

The mass distribution of the second link is shown in Figure A.6. Due to the geometrical similarity of the first and second link The steps taken in the previous section will be followed here.

Data:

$$\begin{aligned} l_2 &= 0.492 & \text{density}(\rho) &= 2.71E3\ Kg/m^3 \\ a_{2_{outer}} &= 0.052 & a_{2_{inner}} &= 0.0467 \\ b_{2_{outer}} &= 0.079 & b_{2_{inner}} &= 0.074 \end{aligned}$$

A.2.1 Centre of gravity of link 2

Due to the symmetric distribution of the second link's masses, Figure A.6, its C.G. lies in on the centre-lines intersection point of the link. Thus the x, y and z coordinates of the C.G. with respect to the link's fixed coordinate frame are :

$$\left.\begin{aligned} x &= -\tfrac{l_2}{2} = -0.245\ m \\ y &= 0.0 \\ z &= -d = -0.1\ m \end{aligned}\right\} \qquad \text{(A.40)}$$

Using the above data :

$$\begin{aligned} m_{2_{outer}} &= l \times a_{outer} \times b_{outer} \times \rho \\ &= 0.492 \times 0.052 \times 0.079 \times \rho \\ &= 5.477Kg \\ m_{2_{inner}} &= l \times a_{inner} \times b_{inner} \times \rho \\ &= 0.492 \times 0.0467 \times 0.074 \times \rho \\ &= 4.607Kg \end{aligned}$$

Total mass of link 2 :

$$\begin{aligned} m_2 &= (m_3 + m_4)_{end-points\ masses} + (m_{2_{outer}} - m_{2_{inner}}) + m_{6_{parasitic}} \\ &= (1.5 + 1.5) + (5.477 - 4.607) + 0.5 = 4.37\ Kg \end{aligned}$$

A.2.2 Moment of inertia calculation

By taking the moment of inertia about the C.G. coordinate frame of the second link :

$$\begin{aligned} I_{X_oX_o} &= \left(\frac{1}{12}\right) m\left(a_2^2 + b_2^2\right) \\ I_{Y_oY_o} &= \left(\frac{1}{12}\right) m\left(b_2^2 + l_2^2\right) \\ I_{Z_oZ_o} &= \left(\frac{1}{12}\right) m\left(a_2^2 + l_2^2\right) \end{aligned}$$

By using the parallel axis theorem, the moment of inertia of the link's body about its fixed coordinate frame can be obtained :

$$\begin{aligned} I_{XX} &= I_{X_oX_o} + md^2 \\ I_{YY} &= I_{Y_oY_o} + m\left(\left(\frac{1}{2}l_2\right)^2 + d_2^2\right) \\ I_{ZZ} &= I_{Z_oZ_o} + m\left(\frac{1}{2}l_2\right)^2 \end{aligned}$$

Also, the inertia of the end-points masses are :

$$\begin{aligned} I_{m3_{XX}} &= 1.5d_2^2 & \qquad I_{m4_{XX}} &= 1.5d_2^2 \\ I_{m3_{YY}} &= 1.5\left(l_2^2 + d_2^2\right) & \qquad I_{m4_{YY}} &= 1.5d_2^2 \\ I_{m3_{ZZ}} &= 1.5l_2^2 & \qquad I_{m4_{ZZ}} &= 0.0 \end{aligned}$$

Also, the inertia of the second link's parasitic masses ($m6$) is :

$$\begin{aligned} I_{m6_{XX}} &= 0.5d_2^2 \\ I_{m6_{YY}} &= 0.5\left(\left(\frac{1}{2}l_2\right)^2 + d_2^2\right) \\ I_{m6_{ZZ}} &= 0.0 \end{aligned}$$

Therefore, the total moment of inertia of the second link about its fixed coordinate frame can be obtained as follows :

$$\begin{aligned} I_{XX} &= (I_{XX_{mouter}} - I_{XX_{minner}}) + I_{XX_{m4}} + I_{XX_{m5}} + I_{XX_{m6}} \\ &= \left\{\left[\frac{1}{12}(5.477)\left((0.052)^2 + (0.079)^2\right)\right] - \left[\frac{1}{12}(4.607)\left((0.0467)^2 + (0.074)^2\right)\right]\right\} \\ &= +1.5\,(1.5)^2 + 1.5\,(1.5)^2 + 0.5\,(0.1)^2 \\ &= \{0.00408 - 0.00291\} + 0.015 + 0.015 + 0.005 \\ &= 0.03617\ Kg\ m^2 \end{aligned} \tag{A.41}$$

$$\begin{aligned} I_{YY} &= (I_{YY_{mouter}} - I_{YY_{minner}}) + I_{YY_{m4}} + I_{YY_{m5}} + I_{YY_{m6}} \\ &= \left\{\left[\frac{1}{12}(5.477)\left((0.079)^2 + (0.492)^2\right) + 5.477\left((0.5 \times 0.492)^2 + (0.1)^2\right)\right]\right. \\ &\quad \left. - \left[\frac{1}{12}(4.607)\left((0.074)^2 + (0.492)^2\right) + 4.607\left((0.5 \times 0.492)^2 + (0.1)^2\right)\right]\right\} \\ &\quad +1.5\left((0.492)^2 + (0.1)^2\right) + 1.5\,(0.1)^2 + 0.5\left((0.5 \times 0.492\,)^2 + (0.1)^2\right) \\ &= \{[0.1133 + 0.3862] - [0.0950 + 0.3249]\} \\ &\quad +0.3780 + 0.015 + 0.0352 \\ &= 0.5078\ Kg\ m^2 \end{aligned} \tag{A.42}$$

$$
\begin{aligned}
I_{ZZ} &= \left(I_{ZZ_{m_{outer}}} - I_{ZZ_{m_{inner}}}\right) + I_{ZZ_{m4}} + I_{ZZ_{m5}} + I_{ZZ_{m6}} \\
&= \left\{\left[\frac{1}{12}(5.477)\left((0.052)^2 + (0.492)^2\right) + 5.477\,(0.5 \times 0.492)^2\right]\right. \\
&\quad \left. - \left[\frac{1}{12}(4.607)\left((0.0467)^2 + (0.492)^2\right) + 4.607\left((0.5 \times 0.492)^2 + (0.1)^2\right)\right]\right\} \\
&\quad +1.5\left((0.492)^2 + 0.5\left(0.5 \times 0.492\right)^2\right. \\
&= (0.1117 + 0.3314) - (0.0937 + 0.2788) + 0.3631 + 0.03025 \\
&= 0.4639\ Kg\ m^2 \qquad \text{(A.43)}
\end{aligned}
$$

Product inertia calculation :

The product of inertia of the second link about its fixed coordinate frame can be obtained using equation A.21 as follows :

$$
\begin{aligned}
I_{XY_{m_{outer}}} &= 0.0 + 5.477\left(-\frac{0.492}{2}\right)(0.0) \\
&= 0.0 \\
&\text{Similarly;} \\
I_{XY_{m_{inner}}} &= 0.0 \\
&\text{Also,} \\
I_{YZ_{m_{outer}}} &= 0.0 + 5.477\,(0.1)\,(0.0) \\
&= 0.0 \\
&\text{Similarly;} \\
I_{YZ_{m_{inner}}} &= 0.0 \\
&\text{Also,} \\
I_{XZ} &= 0.0
\end{aligned}
$$

So, the final result of the inertia product of the second link can be expressed in the following equation :

$$I_{XY} = I_{YZ} = I_{XZ} = 0.0 \qquad \text{(A.44)}$$

A.3 Inertia Calculation of the Third Link

The third link of the SCARA robot in this dissertation is augmented to consists of :

- A solid rectangular steel bar, with the dimension shown in Figure A.3.
- The wrist components and the load are represented by a homogeneous spherical mass of 5 Kg. This mass is attached to the lower end of link 3*.

*This assumption is for constant payload condition.

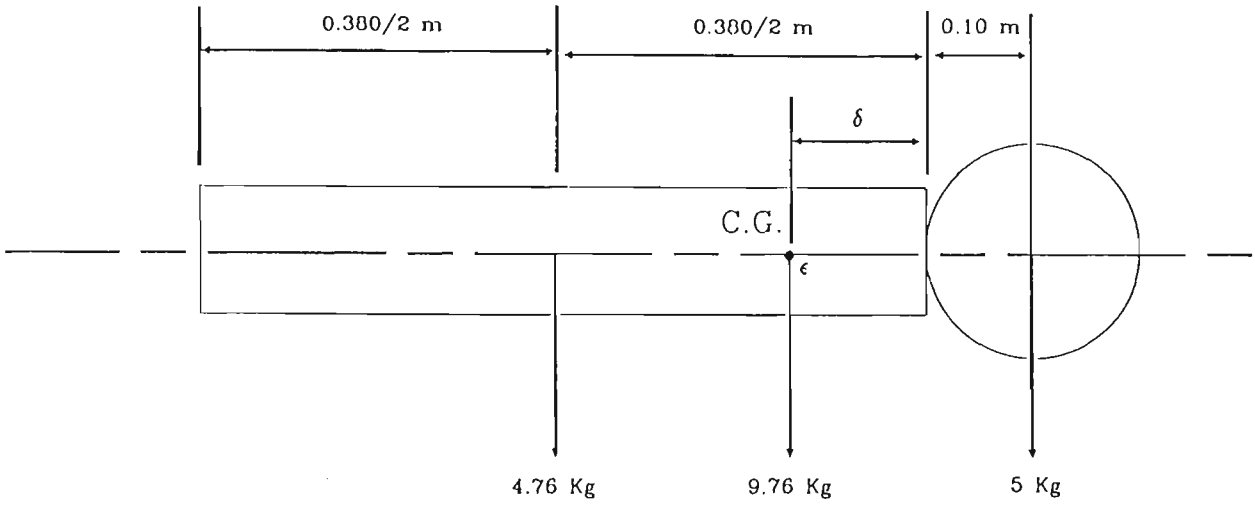


Figure A.7: Mass distribution of the third link.

Therefore, the moment of inertia are calculated for each part separately and then resultant inertia is obtained.

A.3.1 Centre of gravity of link 3

The mass of the main link-body can be obtained through the following equation :

$$\begin{aligned} \text{The link's mass} \quad m_3 &= a_3 \times b_3 \times l_3 \times \rho_{steel} \\ &= 0.04 \times 0.04 \times 0.38 \times 7830 \\ &= 4.76\ Kg \end{aligned}$$

Therefore, the total mass of the augmented third link is :

$$m_{3_{total}} = 4.76 + 5 = 9.76\ Kg \tag{A.45}$$

If it is assumed that the third link's C.G. lies at point ϵ which is at a distance δ from the link's end, Figure A.7.

Then the C.G. can be found by taking the moment about ϵ (as function of δ) :

$$\sum M_\epsilon = 0.0$$

Therefore;

$$\begin{aligned} 5\,(\delta + 0.1) - 4.76\,(0.190 - \delta) &= 0.0 \\ 5\delta + 0.5 - 0.9044 + 4.76\delta &= 0.0 \\ 9.76\delta &= 0.4044 \\ \text{Then :} \quad \delta &= 0.0414\ m \end{aligned} \tag{A.46}$$

Thus the x, y and z coordinates of the C.G. with respect to the link's fixed coordinate frame are :

$$\left.\begin{array}{rcl} x & = & 0.0\ m \\ y & = & 0.0 \\ z & = & 0.041\ m \end{array}\right\} \tag{A.47}$$

A.3.2 Moment of inertia calculation

Moment of inertia of the link's main-body can be obtained through the following set of equations.

$$\begin{aligned} \text{Since} \quad I_{XX} &= I_{X_oX_o} + m\ dx\ dx \\ \text{Then} \quad I_{XX} &= \frac{1}{12} m_3 \left(a_3^2 + l_3^2\right) + m_3 \left(\frac{1}{2} l_3\right)^2 \\ &= \frac{1}{12} m_3 \left((0.04)^2 + (0.38)^2\right) + m_3 \left(0.5 \times 0.38\right)^2 \\ &= 0.01217 \times m_3 + 0.0361 \times m_3 \\ &= 0.2298\ Kg.m^2 \end{aligned} \tag{A.48}$$

$$\begin{aligned} \text{Similarly;} \quad I_{YY} &= \frac{1}{12} m_3 \left(b_3^2 + l_3^2\right) + m_3 \left(\frac{1}{2} l_3\right)^2 \\ &= \frac{1}{12} m_3 \left((0.04)^2 + (0.38)^2\right) + m_3 \left(0.5 \times 0.38\right)^2 \\ &= 0.01217 m_3 + 0.0361 m_3 = 0.04827 m_3 \\ &= 0.2298\ Kg.m^2 \end{aligned} \tag{A.49}$$

$$\begin{aligned} \text{Also;} \quad I_{ZZ} &= \frac{1}{12} m_3 \left(a_3^2 + b_3^2\right) \\ &= \frac{1}{12} m_3 \left((0.04)^2 + (0.04)^2\right) \\ &= m_3 \times 0.000267 = 4.76 \times 0.000267 \\ &= 0.001269\ Kg.m^2 \end{aligned} \tag{A.50}$$

Augmenting link 3

It is assumed that the payload and the wrist components have a mass 5 Kg. represented by homogeneous sphere of 0.1m radius, as schematically shown in Figure A.8.

$$I_{sphere} = \frac{2}{5}\ m\ r^2$$

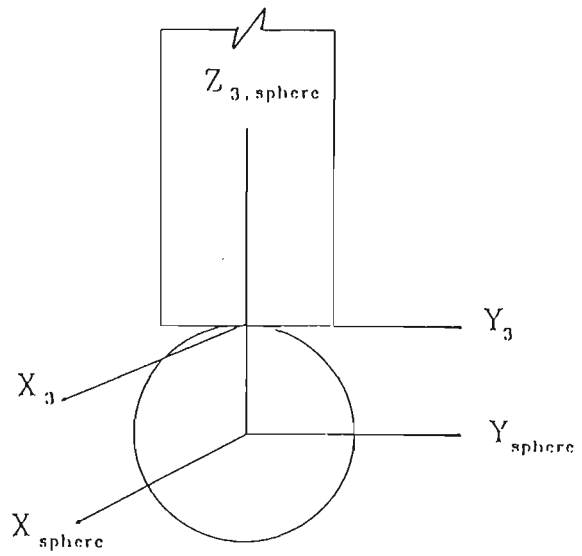


Figure A.8: Third link's lower end.

Then;

$$\begin{aligned} I_{XX_{sphere}} &= \frac{2}{5} \times 5 \times (0.1)^2 \\ &= 0.02\ Kg.m^2 \\ \text{Similarly;}\quad I_{YY_{sphere}} &= 0.02\ Kg.m^2 \\ \text{Also;}\quad I_{ZZ_{sphere}} &= 0.02\ Kg.m^2 \end{aligned}$$

By transfering the sphere's moment of inertia from the sphere's C.G. to the third link's fixed coordinate frame, then:

$$I_{X_3X_{3_{sphere}}} = 0.02 + 5\,(0.1)^2 = 0.07\ Kg.m^2 \tag{A.51}$$

$$I_{Y_3Y_{3_{sphere}}} = 0.02 + 5\,(0.1)^2 = 0.07\ Kg.m^2 \tag{A.52}$$

$$I_{ZZ_{sphere}} = 0.02\ Kg.m^2 \tag{A.53}$$

The total moment of inertia then can be obtained by adding the link's main body's inertia, to the spherical mass inertia.

Therefore, by adding (A.48) and (A.51) :

$$I_{XX_3} = 0.2298 + 0.07 = 0.2998\ Kg.m^2 \tag{A.54}$$

By adding (A.49) and (A.52) :

$$I_{YY_3} = 0.2298 + 0.07 = 0.2998\ Kg.m^2 \tag{A.55}$$

By adding (A.50) and (A.53) :

$$I_{ZZ_3} = 0.001269 + 0.02 = 0.021269\ Kg.m^2 \tag{A.56}$$

Similar to the first and second links, the product of inertia of the third link are negligible, i.e :

$$I_{XY} = I_{YZ} = I_{XZ} = 0.0 \tag{A.57}$$

A.4 Dynamic Model Data

The data in this appendix with the necessary information about each link's centre of gravity vector ($\vec{r}$), and their pseudo inertia matrices (J) are used in the dynamic model.

A.4.1 First link data

From equation (A.7), the first link's centre of gravity is :

$$\vec{r}_1 = \begin{pmatrix} -0.417 \\ 0.0 \\ 0.0 \\ 1 \end{pmatrix} \tag{A.58}$$

Also, from equation (2.12), page 17, the pseudo inertia matrix (J) is

$$J_i = m_i \begin{pmatrix} \left(\frac{-K_{iXX}^2+K_{iYY}^2+K_{iZZ}^2}{2}\right) & K_{iXY}^2 & K_{iXZ}^2 & x_i \\ K_{iXY}^2 & \left(\frac{K_{iXX}^2-K_{iYY}^2+K_{iZZ}^2}{2}\right) & K_{iYZ}^2 & y_i \\ K_{iXZ}^2 & K_{iYY}^2 & \left(\frac{K_{iXX}^2+K_{iYY}^2-K_{iZZ}^2}{2}\right) & z_i \\ x_i & y_i & z_i & 1 \end{pmatrix} \tag{A.59}$$

From (A.36), (A.37) and (A.38) :

$$\left.\begin{array}{rcl} K_{1XX}^2 = \frac{I_{1XX}}{m_1} & = & 0.000228669 \\ K_{1YY}^2 = \frac{I_{1YY}}{m_1} & = & 0.2648 \\ K_{1ZZ}^2 = \frac{I_{1ZZ}}{m_1} & = & 0.2644 \end{array}\right\} \tag{A.60}$$

Also, from equation (A.39) :

$$K_{1XY}^2 = K_{1XZ}^2 = K_{1YZ}^2 = 0.0 \tag{A.61}$$

Then, by substituting from (A.60) and (A.61) in (A.59) the first link's pseudo-inertia matrix is obtained :

$$J_1 = 10.326 \begin{pmatrix} \frac{1}{2}(0.52897) & 0.0 & 0.0 & -0.417 \\ 0.0 & \frac{1}{2}(-0.0001714) & 0.0 & 0.0 \\ 0.0 & 0.0 & \frac{1}{2}(0.0006286) & 0.0 \\ -0.417 & 0.0 & 0.0 & 1 \end{pmatrix} \tag{A.62}$$

A.4.2 Second link data

From equation (A.40), the second link's centre of gravity is :

$$\vec{r}_2 = \begin{pmatrix} -0.246 \\ 0.0 \\ -0.1 \\ 1 \end{pmatrix} \tag{A.63}$$

Also, from (A.41), (A.42) and (A.43) :

$$\left.\begin{array}{rcl} K^2_{2XX} = \frac{I_{2XX}}{m_2} & = & 0.008277 \\ K^2_{2YY} = \frac{I_{2YY}}{m_2} & = & 0.11620 \\ K^2_{2ZZ} = \frac{I_{2ZZ}}{m_2} & = & 0.10615 \end{array}\right\} \tag{A.64}$$

Also, from equation (A.44) :

$$K^2_{2XY} = K^2_{2XZ} = K^2_{2YZ} = 0.0 \tag{A.65}$$

Then, by substituting from (A.64) and (A.65) in (A.59) the second link's pseudo-inertia matrix is obtained :

$$J_2 = 4.37 \begin{pmatrix} \frac{1}{2}(0.214073) & 0.0 & 0.0 & -0.246 \\ 0.0 & \frac{1}{2}(-0.001773) & 0.0 & 0.0 \\ 0.0 & 0.0 & \frac{1}{2}(0.018327) & -0.1 \\ -0.246 & 0.0 & -0.1 & 1 \end{pmatrix} \tag{A.66}$$

A.4.3 Third link data

From equation (A.47), the third link's centre of gravity is :

$$\vec{r}_3 = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.041 \\ 1 \end{pmatrix} \tag{A.67}$$

Also, from (A.54), (A.55) and (A.56) :

$$\left.\begin{array}{rcl} K^2_{3XX} = \frac{I_{3XX}}{m_3} & = & 0.0307 \\ K^2_{3YY} = \frac{I_{3YY}}{m_3} & = & 0.0307 \\ K^2_{3ZZ} = \frac{I_{3ZZ}}{m_3} & = & 0.002179 \end{array}\right\} \tag{A.68}$$

Also, from equation (A.57) :

$$K^2_{3XY} = K^2_{3XZ} = K^2_{3YZ} = 0.0 \tag{A.69}$$

Then, by substituting from (A.68) and (A.69) in (A.59) the second link's pseudo-inertia matrix is obtained :

$$J_3 = 9.76 \begin{pmatrix} \frac{1}{2}(0.002179) & 0.0 & 0.0 & 0.0 \\ 0.0 & \frac{1}{2}(0.002179) & 0.0 & 0.0 \\ 0.0 & 0.0 & \frac{1}{2}(0.059221) & 0.041 \\ 0.0 & 0.0 & 0.041 & 1 \end{pmatrix} \tag{A.70}$$

Appendix B

Counter Balance Masses

In order to achieve a practical dynamic balancing for PUMA 560 robot's arm, the centrifugal and gravity forces should be eliminated or minimized by installing counter-balance masses for both link 2 and link 3. Link 1 is already balanced and therefore it does not need a counter balance mass.

Links 4, 5 and 6 are considered, in this dissertation, as a constituent mass of link 3. Since the full weight of link 3 is carried by link 2, the counter balance mass for link 3 has to be found first.

Since the model was extended to include the study of the effect of the geometrical parameters on the robot's dynamic behaviour, the counter balance masses were driven as functions of each link's length.

B.1 Counter Balance Mass of the Third Link

The third link will be dynamically balanced if its centre of gravity lies on its axis of rotation. To achieve this, a counter balance mass is attached to the link as illustrated in Figure B.1.

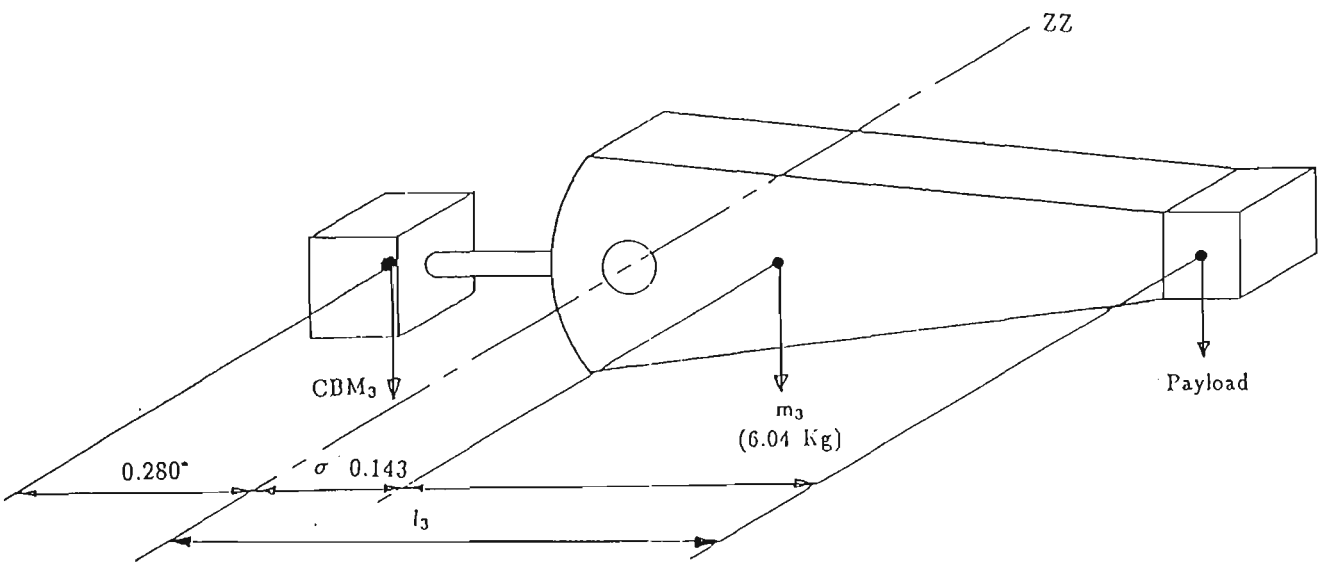
With reference to Figure B.1, by taking the moment around ZZ the counter balance mass of the third link CBM_3 and its total mass m_{3_t} can be expressed as functions of its variable length l_3 as follows :

$$(0.280 \times CBM_3) = \left(6.04\left(\frac{0.143}{0.565}\right)l_3\right) + (2.5l_3)$$

$$CBM_3 = \left(\frac{\left(6.04\left(\frac{0.143}{0.565}\right)l_3\right) + (2.5l_3)}{0.280}\right)$$

Also,

$$m_{3_t} = 6.04 + 2.5 + CBM_3$$



* This is a fixed distance due to the geometrical constraints of the PUMA 560.

$l_3 \stackrel{def}{=}$ Nominal length of link 3 (including the last 3 joints).

Figure B.1: Geometrical representation of counter balancing the third link.

It should be noted that the distance between the centre of gravity of the counter balance mass and the centre of gravity of link 3 is fixed $(0.280m)$ due to the geometrical constraints of the PUMA 560.

B.2 Counter Balance Mass of the Second Link

To dynamically balance the second link of PUMA 560 robot's arm, its centre of gravity must lie on its axis of rotation. That can be achieved by attaching a counter balance mass to it as illustrated in Figure B.2 below. This link carries both the third link and its counter balance mass. Therefore the third link should be taken into account when calculating the counter balance mass of the second link.

By taking the moment about XX

$$(0.425 \times CBM_2) = \left(\left(\frac{0.068}{l_2}\right) l_2 \times 17.4\right) + (0.432 \times m_{3_t})$$

Therefore;

$$CBM_2 = \left(\frac{\left(\left(\frac{0.068}{l_2}\right) l_2 \times 17.4\right) + (0.432 \times m_{3_t})}{0.425}\right)$$

B.3 Second Link's Moment of Inertia

Assumptions: In order to make the moment of inertia of any link about its main axes as functions of its variable links, the following assumptions are made :

- The link's weight is fixed, though its length is variable. That allows a check on the effect of the geometrical shape of a link on its dynamic behaviour, irrespective of its weight.
- The centre of gravity of a link is proportional to and a function of the link's length.

Also, the following notation should be defined :

$\mathrm{CBM}_2 \stackrel{\text{def}}{=}$ counter balance mass of the third link.

$\mathrm{m}_2 \stackrel{\text{def}}{=}$ mass of the second link

$\mathrm{m}_{2_t} \stackrel{\text{def}}{=} \mathrm{CBM}_2 + m_2$

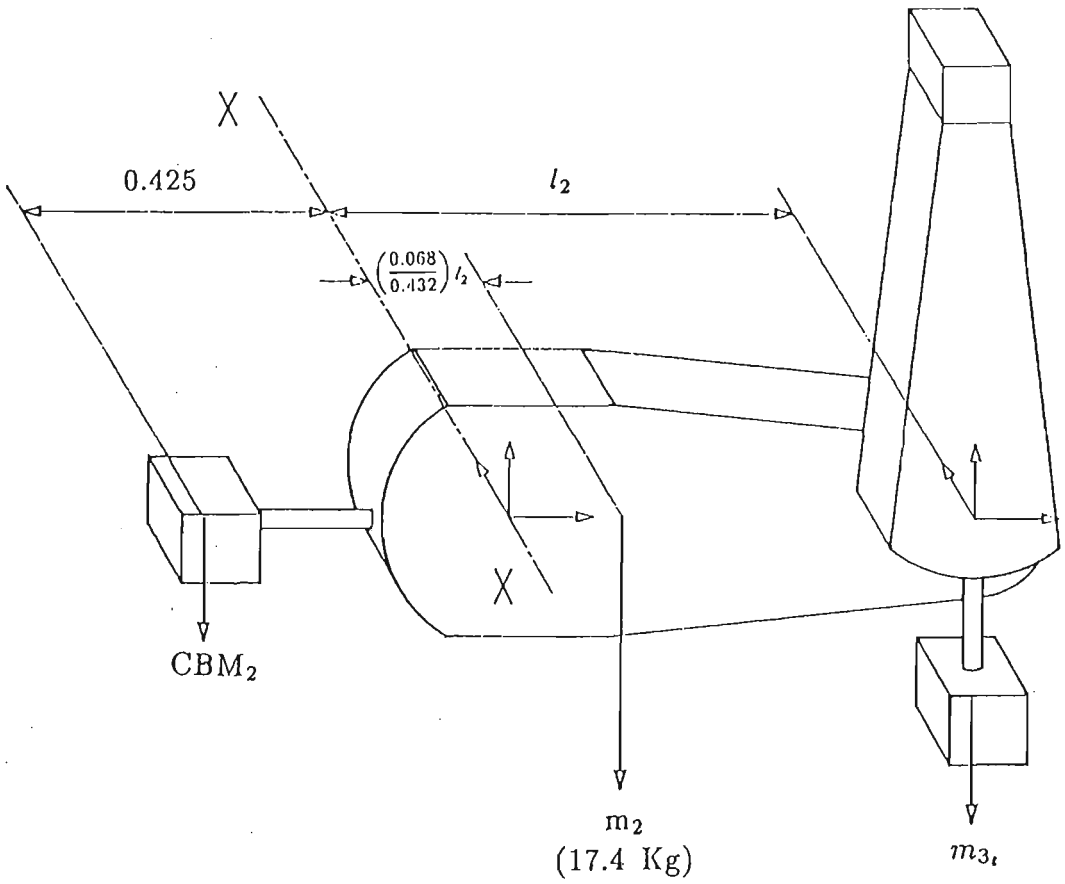


Figure B.2: Geometrical representation of counter balancing the second link.

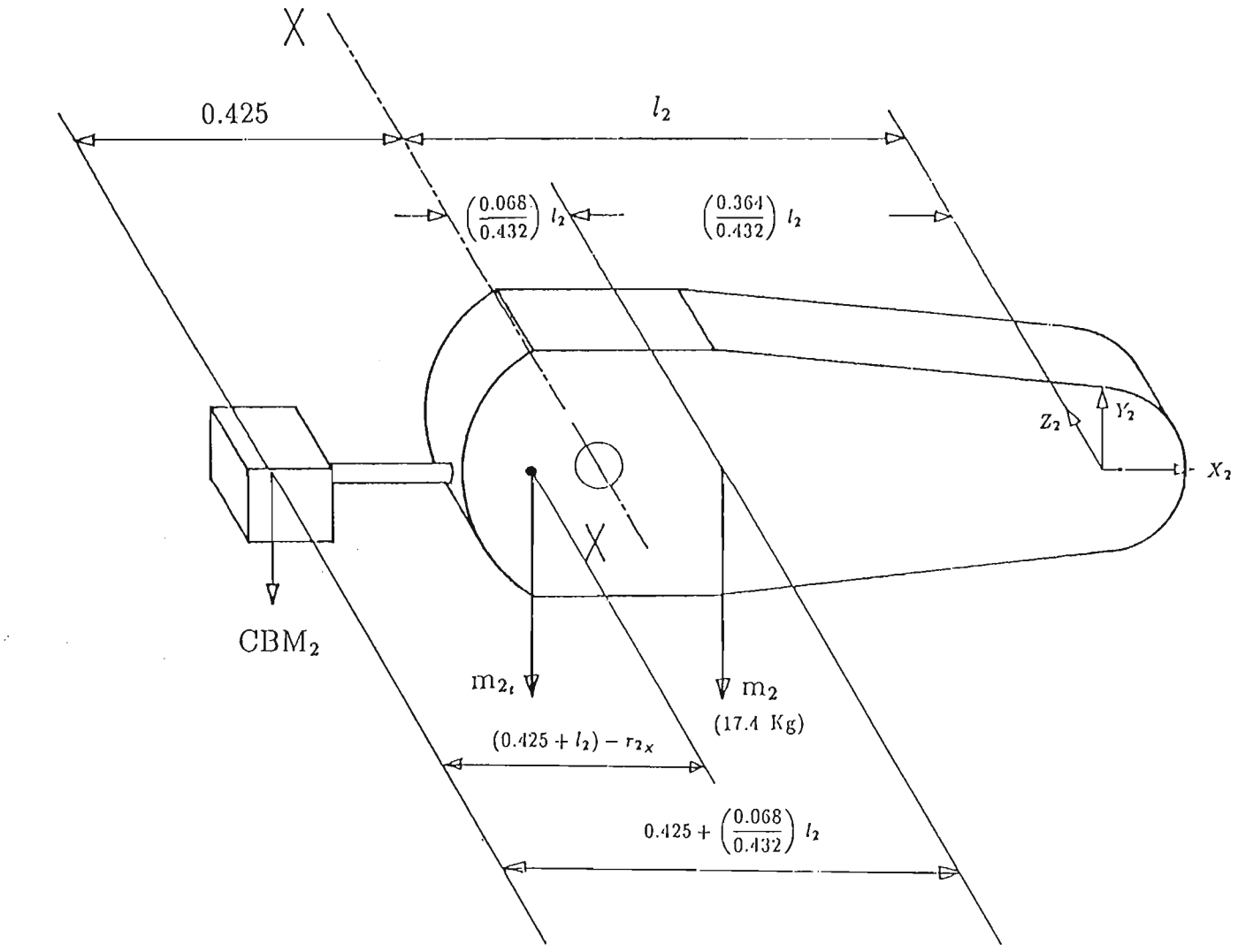


Figure B.3: Inertia-related parameters of the second link.

With reference to Figure (B.3), by taking the moment about axis AA (M_{AA}) :

$$M_{AA}\} \text{ due to } CBM_2 + m_2 = M_{AA}\} \text{ due to } m_{2_t}$$

Therefore :

$$((0.425 + l_2) - r_{2_X})\,(CBM_2 + m_2) = \left(0.425 + \left(\frac{0.068}{0.432}\right) l_2\right)\, m_2$$

i.e.

$$(0.425 + l_2)\,(CBM_2 + m_2) - r_{2_X}\,(CBM_2 + m_2) = \left(0.425 + \left(\frac{0.068}{0.432}\right) l_2\right)\, m_2$$

i.e.

$$r_{2_X} = \left[\frac{(0.425 + l_2)\,(CBM_2 + m_2) - \left(0.425 + \left(\frac{0.068}{0.432}\right) l_2\right)\, m_2}{(CBM_2 + m_2)}\right]$$

Then, the moment of inertia of the link about the three main axes are determined as follows :

$$I_{XX_{CBM_2}} = I_{YY_{CBM_2}} = I_{ZZ_{CBM_2}} = \frac{bh^3}{12}$$

i.e.

$$I_{XX_{CBM_2}} = \frac{(0.15)(0.15)^3}{12} = 4.2187510^{-5} \approx 0$$

$$I_{XX} = I_{XX_{link}} + I_{XX_{CBM_2}} = 0.130 + 0 = 0.130$$

$$I_{YY} = 0.524\left(\frac{\left(\frac{0.364}{0.432}\right) l_2}{0.364}\right)^2 + m_2\left(r_{2_X} - \left(\frac{0.364}{0.432}\right) l_2\right)^2 + CBM_2(0.425 + l_2 + r_{2_X})^2$$

$$I_{ZZ} = 0.539 + 17.4\left(r_{2_X} - \left(\frac{0.364}{0.432}\right) l_2\right)^2 + CBM_2(0.425 + l_2 + r_{2_X})^2$$

B.4 Third Link's Moment of Inertia

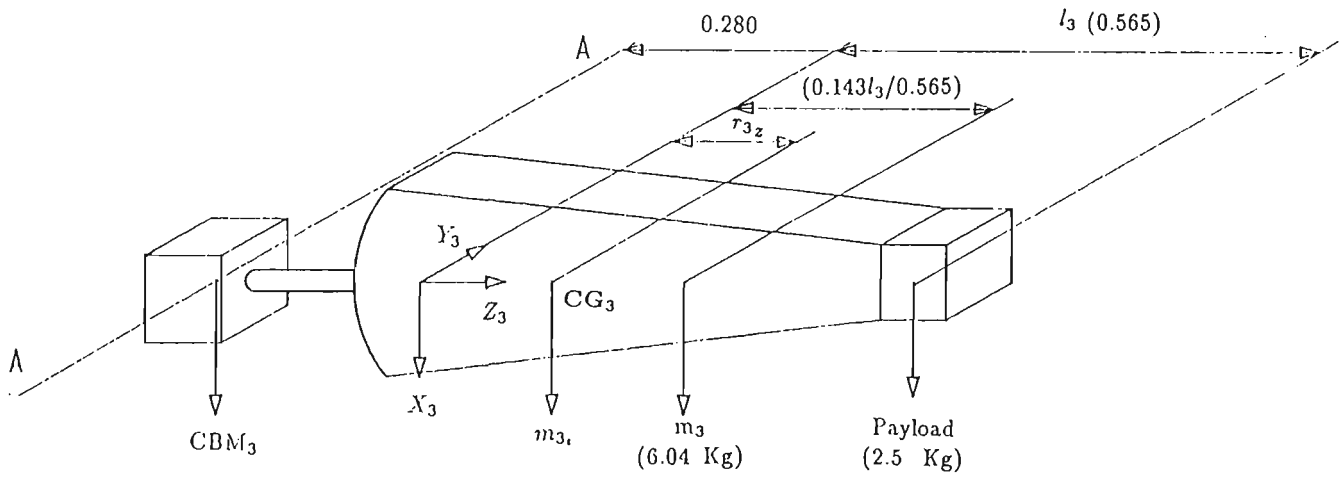


Figure B.4: Inertia-related parameters of the third link.

In finding the inertia of the third link the following conventions will be used :

$$\begin{aligned} \mathrm{CBM_3} &\stackrel{\text{def}}{=} \text{counter balance mass of the third link.} \\ \mathrm{m_3} &\stackrel{\text{def}}{=} \text{mass of the third link (6.04 Kg)} \\ \mathrm{PL} &\stackrel{\text{def}}{=} \text{payload mass} \\ \mathrm{m_{3_t}} &\stackrel{\text{def}}{=} \mathrm{CBM_3} + m_3 + PL \end{aligned}$$

In this section also, the moment of inertia will be found by taking the moment about AA; so by taking the moment of m_{3_t} about AA :

$$M_{AA} = (CBM_3 + m_3 + PL)\,(r_Z + 0.280\,)$$

Also, by taking the moment of the individual component :

$$M_{AA} = m_3 \left(0.280 + \left(\frac{0.143}{0.565}\right) l_3\right) + PL\,(0.280 + l_3)$$

Therefore;

$$r_Z\,(CBM_3 + m_3 + PL) = \left(\frac{0.143 l_3}{0.565}\right) m_3 + l_3 PL - 0.280 CBM_3$$

$$r_Z\,(CBM_3 + 6.04 + PL) = \left(\frac{0.143 l_3}{0.565}\right) m_3 + l_3 PL - 0.280 CBM_3$$

i.e.

$$r_Z = \frac{\left(\left(\frac{0.143 l_3}{0.565}\right) m_3 + l_3\,PL - 0.280 CBM_3\right)}{(CBM_3 + 6.04 + PL)}$$

$$\left.\begin{aligned} r_Y &= 0.014 \\ r_X &= 0 \end{aligned}\right\} \text{from the PUMA 560 geometrical data}$$

The moment of inertia of the link about the main three axes are determined as follows :

$$\begin{aligned} I_{XX_{(CBM_3)}} &= I_{YY_{(CBM_3)}} = I_{ZZ_{(CBM_3)}} \approx 0 \\ I_{XX_{(PL)}} &= I_{YY_{(PL)}} = I_{ZZ_{(PL)}} \approx 0 \end{aligned}$$

$$\begin{aligned} I_{XX} &= I_{XX_{link3}} + m_3\left(\left(\tfrac{0.143 l_3}{0.565}\right) - r_Z\right)^2 \\ &\quad + PL(l_3 - r_Z)^2 + CBM_3(0.280 + r_Z)^2 \end{aligned}$$

$$\begin{aligned} I_{YY} &= I_{YY_{link3}} + m_3\left(\left(\tfrac{0.143 l_3}{0.565}\right) - r_Z\right)^2 \\ &\quad + PL(l_3 - r_Z)^2 + CBM_3(0.280 + r_Z)^2 \end{aligned}$$

Since $I_{YY_{link3}} = 0.212$ (for the nominal length of link 3) therefore :

$$\begin{aligned} I_{YY} &= 0.212\left(\tfrac{l_3}{0.565}\right) + m_3\left(\left(\tfrac{0.143 l_3}{0.565}\right) - r_Z\right)^2 \\ &\quad + PL(l_3 - r_Z)^2 + CBM_3(0.280 + r_Z)^2 \end{aligned}$$

$\mathrm{I}_{ZZ} = I_{ZZ_{link3}} = 0.0154$

Appendix C

Software package

Software development has been a major part of this thesis to allow the fulfilment of its analysis objectives. That has included the development of the following packages :

1. Iterative general dynamic model.
2. An explicit closed-form dynamic model of PUMA 560.
3. Inverse kinematic model of PUMA 560.
4. A HPGL code-generating plot package.
5. A third order polynomial velocity trajectory generator.
6. An *NC2* velocity trajectory generator.

That was in addition to other programmes for other specified tasks such as inertia matrix's eigenvalues analysis and the third link's mass centre relocation. The software were written using the extended ANSI FORTRAN 77. It was run on an HP9000 computer using the unix based *UX* operating system.

The above-mentioned software is listed below.

```
$OPTION  CROSSREF ON
$OPTION  SYMTABLE ON
$OPTION LIST ON
C
      PROGRAM ROBOT_KIDY
C
C   This program was last modefied on 19/10/89
C   It includes the smoothest NC2 curve !!!
C   It runs for either PUMA or SCARA robot. It allows for:
C       1- Different velocity trajectories.
C       2- Different types of velocity trajectories :-
C            (a) Third order polynomial velocity trajectory.
C            (b) NC2 velocity trajectory.
C       3- Different starting angles.
C       4- Fixed or Time Varying Pay-Load. The later can be either:
C            a- Continous variation in time,
C          or
C            b- discrete variation in time.
C   This program, also, resolves the inverse kinematics of the PUMA
C   560 robot's arm.
C
      INCLUDE 'variables_2'
C
      CALL ROBOT_TYPE (ROBOT,NLINK)
C
      CALL GET_DATAFILE (NLINK,ROT,RM,SMALLA,TWIST,DEE,G,R,RI)
C
      CALL INIT_Q (QROT,QSLID)
C
      CALL INERT_K (NLINK,SQRK,RI,RM,PSINER,R,RJ)
C
      CALL LOAD_INFO (RLOAD,VRLD,CHANGE,DECR,PERCENT,
     & IFRACTION,sr88)
C
      CALL SPACE_TYPE (SPACE)
      IF (SPACE(1:1) .EQ. 'C' .OR. SPACE(1:1) .EQ. 'c') THEN
C
         CALL CART_TRJTYP (TRJ,NUMTRJ)
         CALL ROBOT_CONFIG (ARM,ELBOW)
         CALL END_POSs (Px,Py,Pz,ALPHA1,BETA1,GAMMA1,
     &                          ALPHA61,BETA61,GAMMA61)
      ENDIF
C
      IF (SPACE(1:1) .EQ. 'J' .OR. SPACE(1:1) .EQ. 'j') THEN
         CALL JOINT_TRAJ (NLINK,TRAJ,NUMTRJ,DEGREES,ENCREM,XX,ROT)
      ENDIF
C
C****  START PROCESSING A LOOP WITHEN THE TRAJECTORY LOOP  ****
C
      DO III= 1,NUMTRJ
        IF (SPACE(1:1) .EQ. 'J' .OR. SPACE(1:1) .EQ. 'j') THEN
          CALL JOINTRAJ_PVA (NLINK,XX,DEGREES,TRAJ,PP,VV,AA,
     &                                  P_ALL,V_ALL,A_ALL,III)
        ELSE
          CALL CART_TRJ (Px,Py,Pz,ALPHA1,BETA1,GAMMA1,
     &     ALPHA61,BETA61,GAMMA61,ALPHA,BETA,GAMMA,TRJ,
     &     P_ALL,V_ALL,A_ALL,ARM,ELBOW,III)
        ENDIF
C
        VARIATION = 0.0
        CURR_LD = RLOAD
        RMLINK3 = RM(3)
C
899     DO L = 1,61
C
          CALL CURRENT_LD (sr88,sr99,CHANGE,CURR_LD,PERCENT,RLOAD,
     &    IFRACTION,VARIATION,L)
C
          PAYLD (L) = CURR_LD
          RM(3) = RMLINK3 + CURR_LD
C
          CALL GET_R_RI_LINKN (ROBOT,R,RI,RM,NLINK,SMALLA,CURR_LD)
C
          CALL GET_RJ_LINKN (SQRK,NLINK,RI,R,PSINER,RM,RJ)
C
          SS = L
          X(L-1) = (SS-1.)/60.
C
          CALL KINEM_VAR(VAR,VEL,ACCEL,P_ALL,V_ALL,A_ALL,III,NLINK,L)
C
```

```
C
C         **********************
C         *                    *
C         * BACKWARD RECURSION *
C         *                    *
C         **********************
C
          CALL BACK_RECURSION(NLINK,ROT,VAR,VEL,ACCEL,SMALLA,TWIST,DEE,
     &    QROT,QSLID,A,QA,QQA,U,T,TDOT,TDOTS)
C
C         **********************
C         *                    *
C         * FORWARD RECURSION  *
C         *                    *
C         **********************
C
          CALL FD_RECURSION (NLINK,A,TDOTS,RJ,RM,R,U,G,F_ALL,
     &    K_ALL,G_ALL,III,L)
        ENDDO
        WRITE (6,*) III
      ENDDO
      CALL GETMAXMIN (FMAX,FMIN,GMAX,GMIN,KMAX,KMIN,AMAX,AMIN,
     & VMAX,VMIN,PMAX,PMIN,F_ALL,K_ALL,G_ALL,A_ALL,V_ALL,P_ALL,
     & PAYLD,RLDMAX,RLDMIN,NUMTRJ)
C
C     *** START PLOTTING ***
C
cc      CALL PLOTPOS (X,P_ALL,PMIN,PMAX,NUMTRJ)
cc      CALL PLOTVEL (X,V_ALL,VMIN,VMAX,NUMTRJ)
cc      CALL PLOTACC (X,A_ALL,AMIN,AMAX,NUMTRJ)
cc      CALL PLOTLOAD (X,PAYLD,RLMIN,RLMAX)
      CALL PLOTGRAVF (X,G_ALL,GMIN,GMAX,NLINK,NUMTRJ,ROT)
      CALL PLOTKINF (X,K_ALL,KMIN,KMAX,NLINK,NUMTRJ,ROT)
      CALL PLOTTORQUE (X,F_ALL,FMIN,FMAX,NLINK,NUMTRJ,ROT)
      CALL CLOSE
C
      STOP
      END



      SUBROUTINE ML2BY3 (D,E,F,K)
      REAL D(4,4),E(4,4,6),F(4,4,6)
C
C     SUBROUTINE TO MULTIPLY TWO DIMENSIONS MATRICES BY
C     THREE DIMENSIONS MATRICES
C
      DO I= 1,4
         DO J= 1,4
            F(I,J,K) = 0.0
            DO K2=1,4
               F(I,J,K) = F(I,J,K) + D(I,K2)*E(K2,J,K)
            ENDDO
         ENDDO
      ENDDO
      RETURN
      END



      SUBROUTINE ML3BY3 (D,E,F,K1,K2,K3)
      REAL D(4,4,6),E(4,4,6),F(4,4,6)
C
C     SUBROUTINE TO MULTIPLY THREE DIMENSIONS MATRICES BY
C     THREE DIMENSIONS MATRICES
C
      DO  I =1,4
         DO  J =1,4
            F(I,J,K3)=0.0
            DO  K= 1,4
               F(I,J,K3) = F(I,J,K3) + D(I,K,K1)*E(K,J,K2)
            ENDDO
         ENDDO
      ENDDO
      RETURN
      END



      SUBROUTINE MATRNS (A,B,K)
```

```
      REAL A(4,4,6),B(4,4,6)
C
C     SUBROUTINE TO TRANSFORM SQUARE MATRICES
C
      DO J=1,4
         DO I=1,4
            B(I,J,K) = A(J,I,K)
         ENDDO
      ENDDO
      RETURN
      END



      SUBROUTINE ML3BY2 (A,B,C,K1,K2,K3)
      DIMENSION A(4,4,6),B(4,6),C(4,6)
C
C     SUBROUTINE TO MULTIPLY THREE DIMENSIONS MATRICES BY
C     TWO DIMENSIONS MATRICES
C
      DO  I = 1,4
         C(I,K3)=0.0
         DO  J=1,4
            C(I,K3)= C(I,K3) + A(I,J,K1) * B(J,K2)
         ENDDO
      ENDDO
      RETURN
      END



      SUBROUTINE TRJPOLY (P,VEL,ACC)
      COMMON RATIO
      DIMENSION P(61),VEL(61),ACC(61),X(0:61)
C
C     SUBROUTINE TO CALCULATE THE TRAJECTORY OF A LINK'S POSITION AS
C     A THIRD ORDER POLYNOMIAL FUNCTION IN TIME.
C
      P(1) = 0.0
      VEL(1) = 0.0
      ACC(1) = 0.0
      P(61) = 1.0
      VEL(61) = 0.0
      ACC(61)= 0.0
      DO I = 2,60
        S = I-1
        T = S/60.
        P(I) = ((1.-T)**3)*(P(1)+(3.*P(1)+VEL(1))*T+(ACC(1)
     &         +6.*VEL(1)+12.*P(1))*(T**2/2.))
     &         +(T**3)*(P(61)+(3.*P(61)-VEL(61))*(1.-T)
     &         +(ACC(61)-(6.*VEL(61))+12.*P(61))*((1.-T)**2/2.))
      ENDDO
      DO I = 1,61
        P(I) = P(I) * RATIO
      ENDDO
      X(0) = 0.0
      DO I=1,60
        SS = I
        X(I) = SS/60.
      ENDDO
      VEL(1) = 0.0
      ACC(1) = 0.0
      DO I=2,60
           VEL(I) = (.5* (P(I+1)-P(I-1)))/(1./60.)
           ACC(I) = (P(I+1) - 2.*P(I) + P(I-1))/((1./60.)**2)
      ENDDO
      VEL(1) = 0.0
      ACC(1) = 0.0
      VEL(61) = VEL(1)
      ACC(61) = ACC(1)
      RETURN
      END



      SUBROUTINE TRJNC2 (P,V,A)
      DIMENSION A(61),V(61),P(61),X(61)
      COMMON RATIO
C
C     THIS SUBROUTINE CALCULATES THE "Numerical Control 2" NC2
```

```
C     VELOCITY TRAJECTORY OF A ROBOT'S LINK.
C
      DO I = 1,16
        A(I) = 1.19
      ENDDO
      A(17)= 1.065
      A(18)= .933
      A(19)= .75
      A(20)= .498
      A(21)= 0.0
      A(22)= -.06
      A(23)= -.115
      A(24)= -.165
      A(25)= -.21
      A(26)= -.255
      A(27)= -.3
      A(28)= -.342
      A(29)= -.384
      A(30)= -.42
      A(31)= -.453
      A(32)= -.483
      A(33)= -.513
      A(34)= -.543
      A(35)= -.572
      A(36)= -.599
      A(37)= -.626
      A(38)= -.649
      A(39)= -.67
      A(40)= -.689
      A(41)= -.704
      A(42)= -.719
      A(43)= -.731
      A(44)= -.743
      A(45)= -.752
      A(46)= -.761
      A(47)= -.767
      A(48)= -.773
      A(49)= -.776
      A(50)= -.778
      A(51)= -.78
      A(52)= -.776
      A(53)= -.768
      A(54)= -.742
      A(55)= -.705
      A(56)= -.66
      A(57)= -.60
      A(58)= -.519
      A(59)= -.42
      A(60)= -.279
      A(61)= 0.0
C
      DO I=1,61
        A(I) = A(I) * RATIO * 4.944688
      ENDDO
C
      V(1) = 0.0
      V(2)=.5 * (A(2) + A(1)) * (1./60.)
C
      DO I = 3,61
        V(I) = ((1./60.)*(A(I)+A(I-1))*.5)+V(I-1)
      ENDDO
      V(61) = 0.0
C
      P(1) = 0.0
      P(2)=.5 * (V(2) + V(1)) * (1./60.)
C
      DO  I = 3,61
        P(I) = ((1./3.)*(1./60.)*(V(I-2)+4.*V(I-1)+V(I)))+P(I-2)
       ENDDO
C
      RETURN
      END


      SUBROUTINE VELACC (P,V,A,TRJ)
      DIMENSION P(61),V(61),A(61),VEL(61),ACC(61),X(61),DF(61)
      DIMENSION B(61),C(61),D(61)
      CHARACTER*4 TRJ
C
```

```
C     THIS SUBROUTINE CALCULATES THE TRAJECTORY OF A LINK'S
C     VELOCITY AND ACCELERATION GIVEN ITS DISPLACEMENT TRAJECTORY
C
      PI = 3.14159274
      VEL(1) = 0.0
      DO  I=2,60
        POS1 = P(I-1)
        POS2 = P(I)
        POS3 = P(I+1)
C
C       *** if it crosses the 2nd qudrant to the 3rd quadrant  ***
C
        IF ((P(I) .GT. (PI/2.) .AND. P(I) .LE. PI) .AND.
     &     (P(I+1) .LT. -(PI/2.) .AND. P(I+1) .GE. -PI)) THEN
         POS3 = PI - (-PI - P(I+1))
C
        ELSE IF ((P(I-1) .GT. (PI/2.) .AND. P(I-1) .LE. PI) .AND.
     &     (P(I) .LT. -(PI/2.) .AND. P(I) .GE. -PI)) THEN
          POS1 = -PI -(PI-P(I-1))
C
C       *** if it crosses the 3rd qudrant to the 2nd quadrant  ***
C
        ELSE IF ((P(I) .LT. -(PI/2.) .AND. P(I) .GE. -PI) .AND.
     &     (P(I+1) .GT. (PI/2.) .AND. P(I+1) .LE. PI)) THEN
          POS3 = -PI - (PI - P(I+1))
C
        ELSE IF ((P(I-1) .LT. -(PI/2.) .AND. P(I-1) .GE. -PI) .AND.
     &     (P(I) .GT. (PI/2.) .AND. P(I) .LE. PI)) THEN
          POS1 = PI + (PI+ P(I-1))
        ENDIF
        VEL(I) = (.5* (POS3-POS1))/(1./60.)
        ACC(I) = (POS3 - 2.*P(I) + POS1)/((1./60.)**2)
      ENDDO
      X(1) = 0.0
      DO I = 2,61
         SS = I - 1
         X(I) = SS/60.
      ENDDO
      STEP = 1.0/60.0
      NX   = 61
      SM   = 0.5
      DO I = 1,61
         DF(I) = 1.0
      ENDDO
      VEL(1) = 0.0
      VEL(61) = VEL(1)
      IF (TRJ(1:1) .EQ. 'P' .OR. TRJ(1:1) .EQ. 'p') THEN
        ACC(1) = 0.0
        ACC(61) = ACC(1)
      ELSE
        ACC(1) = ACC(2)
        ACC(61)= 0.0
      ENDIF
      CALL DCSSMO (STEP,NX,X,VEL,DF,SM,V,B,C,D)
      CALL DCSSMO (STEP,NX,X,ACC,DF,SM,A,B,C,D)
C
      RETURN
      END



      SUBROUTINE ROBOT_TYPE (ROBOT,NLINK)
      CHARACTER*5 ROBOT
      INTEGER NLINK
C
C     THIS SUBROUTINE READS THE ROBOT'S TYPE (SCARA/PUMA),
C     AND THE NUMBER OF LINKS TO BE PROCESSED.
C
38      WRITE (6,37)
37      FORMAT (' WHAT IS THE KIND OF YOUR ROBOT ??'/
     &  ' For SCARA robot type',T60,"'SCARA'"/
     &  ' For PUMA robot type',T60,"'PUMA'   "/)
        READ (5,*) ROBOT
C
C
      IF(ROBOT(1:1) .EQ. 'S' .OR. ROBOT(1:1) .EQ. 's') THEN
        open(unit=1,file='vardata')
      ELSE IF(ROBOT(1:1) .EQ. 'P' .OR. ROBOT(1:1) .EQ. 'p') THEN
        open(unit=1,file='pumadata2')
      ELSE
```

```
          PRINT*,'Your robot is neither SCARA nor PUMA.    '
          PRINT*,'I have no data file for this robot    .'
          STOP
        ENDIF
C
C           ******* READ NO. OF LINKS *******
C
        WRITE (6,666)
666     FORMAT (//," HOW MANY LINKS IN YOUR ROBOT'S ARM ??    "/)
        READ (5,*) NLINK
        RETURN
        END


        SUBROUTINE GET_DATAFILE (NLINK,ROT,RM,SMALLA,TWIST,DEE,G,R,RI)
        INTEGER ROT(6)
        REAL RM(6),SMALLA(6),TWIST(6),DEE(6),G(6,4),R(4,6),RI(4,4,6)
C
C       THIS SUBROUTINE READ THE GEOMETRICAL PARAMETERS OF A ROBOT'S
C       ARM FROM A DATA FILE.
C
C           ** READ TYPE OF JOINT **
C
        READ (1,*) (ROT(K),K=1,NLINK)
C
C           ** READ MASSES **
C
        READ (1,*) (RM(K),K=1,NLINK)
C
C           ** READ LENGTH **
C
        READ (1,*) (SMALLA(K),K=1,NLINK)
C
C
C       ** READ TWIST ANGLES **
C
        READ (1,*) (TWIST(K),K=1,NLINK)
C
C           ** READ OFF-SET **
C
        READ (1,*) (DEE(K),K=1,NLINK)
C
C           ** READ GRAVITY **
C
        DO  K = 1,NLINK
           READ (1,*) (G(K,J),J=1,4)
        ENDDO
C
C           ** READ CENTRES OF MASSES **
C
        DO  I = 1,4
           READ (1,*) (R(I,K),K=1,3)
        ENDDO
C
C           ** READ INERTIA MATRICES **
C
        DO  K =1,NLINK
           DO  I =1,3
              READ (1,*) (RI(I,J,K),J=1,3)
           ENDDO
        ENDDO
C
        RETURN
        END


        SUBROUTINE INIT_Q (QROT,QSLID)
        REAL QROT(4,4),QSLID(4,4)
C
C       SUBROUTINE TO INITIALIZE MATRICES QROT AND QSLID
C
        QROT(1,2) = -1
        QROT(2,1) =  1
        QSLID(3,4)=  1
C
        RETURN
        END
```

```
      SUBROUTINE INERT_K (NLINK,SQRK,RI,RM,PSINER,R,RJ)
      REAL SQRK(4,4,6),RI(4,4,6),RM(6),PSINER(4,4,6),
     & R(4,6),RJ(4,4,6)
      INTEGER NLINK
C
C     SUBROUTINE TO CALCULATE THE PSEUDO INERTIA MATRIX FOR
C     EACH LINK, AND MULTIPLY EACH OF THESE MATRICES BY THE
C     CORRESPONDING LINK'S MASS.
C
      DO  K=1,NLINK-1
         DO  J=1,3
            DO  I=1,3
               SQRK (I,J,K) =  RI(I,J,K)/ RM(K)
               IF (I .NE. J) THEN
                  PSINER (I,J,K) = SQRK (I,J,K)
               ENDIF
            ENDDO
         ENDDO
         PSINER(1,1,K) = .5*(-SQRK(1,1,K)+SQRK(2,2,K)+SQRK(3,3,K))
         PSINER(2,2,K) = .5*(SQRK(1,1,K) - SQRK(2,2,K) + SQRK(3,3,K))
         PSINER(3,3,K) = .5*(SQRK(1,1,K) + SQRK(2,2,K) - SQRK(3,3,K))
         PSINER(1,4,K) = R(1,K)
         PSINER(4,1,K) = R(1,K)
         PSINER(2,4,K) = R(2,K)
         PSINER(4,2,K) = R(2,K)
         PSINER(3,4,K) = R(3,K)
         PSINER(4,3,K) = R(3,K)
         PSINER(4,4,K) = 1
      ENDDO
C
      DO  K=1,NLINK-1
         DO  J=1,4
            DO  I=1,4
               RJ(I,J,K) = PSINER(I,J,K) * RM(K)
            ENDDO
         ENDDO
      ENDDO
C
      RETURN
      END


      SUBROUTINE LOAD_INFO (RLOAD,VRLD,CHANGE,DECR,PERCENT,
     & IFRACTION,sr88)
C
      INTEGER IFRACTION
      REAL RLOAD,PERCENT
      CHARACTER*4 VRLD,DECR,CHANGE,sr88*1,sr99*1
C
C     THIS SUBROUTINE IS TO READ THE MASS AND CHARACTERISTICS
C     OF THE PAYLOAD.
C
      WRITE (6,1)
1     FORMAT (" WHAT IS LOAD IN ROBOT'S HAND (in Kg's) ??   "/
     &" It should be a real number !   "/)
      READ(5,*) RLOAD
C
C
      WRITE (6,122)
122   FORMAT (/,' Is it Time-Variant Load ?   '//)
      VRLD=" "
      READ (5,*) VRLD
      sr99=VRLD
C
      IF (sr99 .EQ. 'Y' .OR. sr99 .EQ. 'y')  THEN
43       WRITE  (6,44)
44       FORMAT(/,' IS THE CHANGE IN PAY-LOAD IS CONTINOUS OR DISCRETE?'
     &   /' ',' For continous change type',T60,"'CONT'"/
     &   /' ',' For discrete change type',T60,"'DISC   '"/)
         READ (5,*) CHANGE
         IF (CHANGE(1:1) .EQ. 'C' .OR. CHANGE(1:1) .EQ. 'c') THEN
            WRITE (6,133)
133         FORMAT (/,' Is it Decreasing Load ?   '//)
            DECR= " "
            READ (5,*) DECR
            sr88 = DECR
            WRITE (6,144)
144         FORMAT(/,' WHAT IS THE PERCENTAGE OF VARIATION/ CYCLE ??'
```

```
     &   /' ',' (e.g. 0.01 or 0.99)'//)
            READ (5,*) PERCENT
         ELSE IF(CHANGE(1:1).EQ.'D' .OR. CHANGE(1:1).EQ.'d')THEN
            WRITE (6,45)
45          FORMAT (/,' HOW OFTEN DOES THE CHANGE OCCURE ?'/
     &       ' It should be as a percentage of the cycle time'/
     &       ' and a real number, e.g., 0.1 or 0.25 ..etc.    '/)
            READ (5,*) FRACTION
            IFRACTION = 60. * FRACTION
            WRITE (6,46)
46          FORMAT (/,' IS IT DECREASING PAY-LOAD ?   '/)
            READ (5,*) DECR
            sr88 = DECR
            WRITE (6,47)
47          FORMAT(/,' WHAT IS THE PAY-LOAD VARIATION AT EACH STEP ?'/
     &   ' It should be as a percentage of the initial Pay-Load'/
     &       ' and a real number   '/)
            READ (5,*) PERCENT
         ELSE
            PRINT *,"The answer was niether 'CONT' nor 'DISC'"
            GO TO 43
         ENDIF
       ENDIF
C
       RETURN
       END



       SUBROUTINE SPACE_TYPE (SPACE)
       CHARACTER*5 SPACE
C
C      THIS SUBROUTINE IS TO DETERMINE WHETHER THE GIVEN DATA
C      REPRESENT A CARTISIAN- OR JOINT-SPACE.
C
          WRITE  (6,44)
44        FORMAT(/,' WHAT IS THE TYPE OF THE GIVEN SPACE TRAJECTORY?'
     &    /' ',' For cartesian-space .. type',T60,"'CART'"/
     &    /' ',' For joint-space .. type',T60,"'JOINT   '"/)
          READ (5,*) SPACE
C
       RETURN
       END



       SUBROUTINE CART_TRJTYP (TRJ,NUMTRJ)
       INTEGER NUMTRJ
       CHARACTER*4 TRJ
C
C      THIS SUBROUTINE IS TO DETERMINE THE TYPE OF THE VELOCITY
C      TRAJECTORY OF THE END-EFFECTOR ALONG THE CARTISIAN PATH.
C
       NUMTRJ = 1
C
       WRITE (6,363)
363    FORMAT (//,'  WHAT IS THE TYPE OF THE TRAJECTORY OF',
      &' CARTESIAN PATH ??   '/' ',' FOR POLYNOMIAL FUNCTION IN',
      &' TIME DISPLACEMENT TYPE',T55,"'POLY'"/' ',' FOR TRAJECTORY ',
      &'USED IN NUMERICAL CONTROL II TYPE',T55,"'NC2'"/' ',' FOR EXIT'
      &,' TYPE',T55,'EXIT'/)
       READ (5,*) TRJ
C
       RETURN
       END



       SUBROUTINE ROBOT_CONFIG (ARM,ELBOW)
       CHARACTER*6 ARM, ELBOW
C
C      THIS SUBROUTINE IS TO DETERMINE THE PUMA 560 GEOMETRICAL
C      CONFIGURATION WHILE IN MOTIN FOR THE KINEMATIC SOLUTION.
C
       WRITE (6,213)
213    FORMAT (//,' ','WHAT IS THE ARM CONFIGURATION    ??'/' ',
      &' FOR LEFTY CONFIGURATION TYPE',T55,"'LEFTY' or 'L'"/' ',
      &' FOR RIGHTY CONFIGURATION TYPE',T55,"'RIGHTY' or 'R'"/)
       READ (5,*) ARM
C
```

```
      WRITE (6,321)
321   FORMAT (//,' ','WHAT IS THE CONFIGURATION OF THE ELBOW    ??'/' ',
     &' FOR ABOVE-THE-WRIST CONFIGURATION TYPE',T55,"'UP' or 'U'"/' ',
     &' FOR UNDER-THE-WRIST CONFIGURATION TYPE',T55,"'DOWN' or 'D'"/)
      READ (5,*) ELBOW
C
      RETURN
      END



      SUBROUTINE JOINT_TRAJ (NLINK,TRAJ,NUMTRJ,DEGREES,ENCREM,XX,ROT)
      INTEGER NLINK,NUMTRJ,ROT(6)
      REAL ENCREM(6),DEGREES(6),XX(6,0:25)
      CHARACTER*5 TRAJ(6)
C
C     THIS SUBROUTINE IS TO READ THE TYPE OF THE VELOCITY
C     TRAJECTORY OF EACH LINK IN THE JOINT-SPACE MODE.  IT
C     ALSO READ THE NUMBER OF THE TRAJECTORIES OF EACH TYPE FOR
C     EACH LINK WITH A FIXED INCREAMENT.
C
      WRITE (6,222)
222   FORMAT (//,'  HOW MANY SPEED TRAJECTORY FOR EACH LINK ??   ')
      READ (5,*) NUMTRJ
C
      DO  K = 1,NLINK
         WRITE (6,363) K
363      FORMAT (//,'  WHAT IS THE TYPE OF THE TRAJECTORY FOR',
     &  ' LINK No.',I2,' ??   '/' ',' FOR POLYNOMIAL FUNCTION IN',
     &  ' TIME DISPLACEMENT TYPE',T60,"'POLY'"/' ',' FOR TRAJECTORY ',
     &  'USED IN NUMERICAL CONTROL II TYPE',T60,"'NC2'"/' ',' FOR EXIT'
     &  ,' TYPE',T60,'EXIT'/)
         READ (5,*) TRAJ(K)
         IF (TRAJ(K) .EQ. 'EXIT') STOP
         IF (ROT(K) .EQ. 1) THEN
          IPRELINK = K -1
          WRITE (6,213) K, IPRELINK
213       FORMAT(//,' WHAT IS THE INITIAL STARTING ANGLE OF LINK No.',I2,
     &    ' ?   '/' ',' (w.r.t. Link',I2,' Co-ordinate Frame)'/' ',
     &    ' N.B. It should be in DEGREES and REAL'/)
          READ (5,*) DEGREES(K)
         ENDIF
67       WRITE (6,321) K
321      FORMAT (//,'  WHAT IS THE DESIRED ICREMENT FOR THE SPEED TRAJECTOR
     &   ES FOR LINK No.',I2/' ',' AFTER THE ZERO TRAJECTORY ??   '/' ',
     &   ' (IT SHOULD BE A REAL NO. say between 0.0 & 0.2)'/)
         READ (5,*) ENCREM(K)
      ENDDO
C
      DO  K=1,NLINK
         DO  I = 1,(NUMTRJ-1)
            XX(K,I) = XX (K,(I-1)) + ENCREM (K)
         ENDDO
      ENDDO
C
      RETURN
      END



      SUBROUTINE END_POSs (Px,Py,Pz,ALPHA1,BETA1,GAMMA1,
     &                     ALPHA61,BETA61,GAMMA61)
      REAL*8 Px(61),Py(61),Pz(61)
      REAL ALPHA1,BETA1,GAMMA1,ALPHA61,BETA61,GAMMA61
C
C   THIS SUBROUTINE IS TO READ STARTING AND FINISHING POSITIONS
C   OF THE END-EFFECTOR AND ITS ORIENTATIONS
C
      WRITE (6,12)
12    FORMAT(/' ',' WHAT ARE THE CARTESIAN COORDINATES (x,y,z) AND'/' '
     &,'THE ORIENTATION (ALPHA,BETA,GAMMA) OF THE STARTING POINT    ?'/)
      READ (5,*) Px(1),Py(1),Pz(1),ALPHA1,BETA1,GAMMA1
      PRINT *,'MAIN', Px(1),Py(1),Pz(1),ALPHA1,BETA1,GAMMA1
C
      WRITE (6,13)
13    FORMAT (/' ',' WHAT ARE THE CARTESIAN COORDINATES (x,y,z) AND'/
     &' ',' THE ORIENTATION (ALPHA,BETA,GAMMA) OF THE LAST POINT ?   '/)
      READ (5,*) Px(61),Py(61),Pz(61),ALPHA61,BETA61,GAMMA61
      PRINT *,'MAIN', Px(61),Py(61),Pz(61),ALPHA61,BETA61,GAMMA61
C
```

```
      RETURN
      END


      SUBROUTINE JOINTRAJ_PVA (NLINK,XX,DEGREES,TRAJ,PP,VV,AA,
     &                         P_ALL,V_ALL,A_ALL,III)
      REAL XX(6,0:25),DEGREES(6),PP(61),VV(61),AA(61),
     & P_ALL(61,25,61),V_ALL(61,25,61),A_ALL(61,25,61)
      INTEGER NLINK,III
      CHARACTER*5 TRAJ(6)
      COMMON RATIO,DEG
C
C     SUBROUTINE TO CALL THE RIGHT TRAJECTORY GENERATOR AND
C     TO SUPPLY THE CALLING PROGRAM WITH THE INSTANTENOUS
C     POSITION, VELOCITY AND ACCELERATION.
C

      DO  K = 1, NLINK
         RATIO = XX(K,(III-1))
         DEG = DEGREES(K)
         IF(TRAJ(K) .EQ. 'POLY' .OR. TRAJ(K) .EQ. 'poly') THEN
            CALL TRJPOLY (PP,VV,AA)
         ELSE
            CALL TRJNC2 (PP,VV,AA)
         ENDIF
         DO  L=1,61
            P_ALL(K,III,L) = PP(L)
            V_ALL(K,III,L) = VV(L)
            A_ALL(K,III,L) = AA(L)
         ENDDO
      ENDDO
C
      RETURN
      END

C
      SUBROUTINE CART_TRJ (Px,Py,Pz,ALPHA1,BETA1,GAMMA1,
     &      ALPHA61,BETA61,GAMMA61,ALPHA,BETA,GAMMA,TRJ,
     &                     P_ALL,V_ALL,A_ALL,ARM,ELBOW,III)
      REAL*8 Px(61),Py(61),Pz(61),P_ALL(61),V_ALL(61),A_ALL(61),
     & ALPHA61,BETA61,GAMMA61,ALPHA1,BETA1,GAMMA1,
     & ALPHA,BETA,GAMMA
      REAL ANG_A(6,4,61),ANG_2(6,61)
      CHARACTER*6 ARM , ELBOW
      CHARACTER*4 TRJ
C
C     SUBROUTINE TO CALCULATE THE INVERSE KINEMATICS OF A
C     PUMA 560 ROBOT'S ARM GIVEN THE INETIAL AND FINAL CARTESIAN
C     POSITIONS OF ITS HAND HAND IN SPACE.
C
      CALL CART_PATH (Px(1),Py(1),Pz(1),ALPHA1,BETA1,GAMMA1,
     & Px(61),Py(61),Pz(61),ALPHA61,BETA61,GAMMA61,
     & Px, Py, Pz, ALPHA, BETA, GAMMA, TRJ)
C
      CALL GETANG_A (Px,Py,Pz,ALPHA,BETA,GAMMA,ANG_A)
C
      CALL PATH (ARM, ELBOW, ANG_A, ANG_2)
C
C
C    ***   FIND VEL & ACC OF EACH JOINT   ***
C
      CALL LNK_VELACC (ANG_2,P_ALL,V_ALL,A_ALL,TRJ,III)
C
      RETURN
      END


      SUBROUTINE GETANG_A (Px,Py,Pz,ALPHA,BETA,GAMMA,ANG_A)
      REAL*8 Px(61),Py(61),Pz(61),Pxx,Pyy,Pzz,a2,a3,d3,d4
      REAL ALPHA(61),BETA(61),GAMMA(61),ANG(6,4),ANG_A(6,4,61)
C
C     SUBROUTINE TO CALCULATE THE JOINTS ANGLES IN THE FOUR
C     POSSIBLE GEOMETRICAL CONFIGURATIONS (LEFTY,RIGHTY,ELBOW UP,
C     ELBOW DOWN), FOR EVERY INSTANT OF TIME.
C
      DO  L=1,61
         Pxx = Px(L)
         Pyy = Py(L)
```

```
            Pzz = Pz(L)
            CART_ALPHA = ALPHA(L)
            CART_BETA  = BETA (L)
            CART_GAMMA = GAMMA(L)
C
            a2 = .4318
            a3 = -.02032
            d3 = 0.1495
            d4 = 0.43307
C
            CALL ROBOT_K (a2,a3,d3,d4,Pxx,Pyy,Pzz,CART_ALPHA,
     &                                 CART_BETA,CART_GAMMA,ANG)
C
            CALL FILLANG_A (ANG, ANG_A,L)
C
         ENDDO
C
         RETURN
         END



         SUBROUTINE PRINT_ANG (ANG,L)
         REAL ANG(6,4)
C
C        SUBROUTINE TO PRINT THE CALCULATED JOINTS ANGLES FOR
C        FOUR DIFFERENT GEOMETRICAL CONFIGURATIONS OF THE ROBOT'S
C        ARM AND FOR 60 TIME INERVALS WITHIN A SECOND.
C
         PRINT *,' '
         PRINT *,'   **********  L = ',L,'  ***********'
         PRINT *,'                  ANG'
         DO  JJJJ = 1,4
            WRITE (6,4521) (ANG(KKKK,JJJJ),KKKK=1,6)
4521        FORMAT (' ', 6(F8.5,3X))
         ENDDO
         PRINT *,' '
         PRINT *,' '
C
         RETURN
         END



         SUBROUTINE LNK_VELACC (ANG_2,P_ALL,V_ALL,A_ALL,TRJ,III)
         REAL ANG_2(6,61),P_ALL(6,25,61),V_ALL(6,25,61),
        & A_ALL(6,25,61),PP(61),VV(61),AA(61)
         INTEGER K,III,L
         CHARACTER*4 TRJ
C
C        SUBROUTINE TO CALCULATE THE EACH JOINT'S VELOCITY AND
C        ACCELERATION FROM THE CALCULATED DISPLACEMENT TRAJECTORY.
C
         DO  K=1, 6
            DO  L=1,61
               P_ALL(K,III,L) = ANG_2(K,L)
               PP(L) = P_ALL (K,III,L)
            ENDDO
            CALL VELACC (PP,VV,AA,TRJ)
            DO  L=1,61
               V_ALL (K,III,L) = VV(L)
               A_ALL (K,III,L) = AA(L)
            ENDDO
         ENDDO
C
         RETURN
         END



         SUBROUTINE PRINTANG_2 (ANG_2)
         REAL ANG_2(6,61)
C
C        SUBROUTINE TO PRINT THE CALCULATED JOINTS ANGLES FOR
C        A GEOMETRICAL CONFIGURATION OF THE ROBOT'S ARM AND FOR
C        60 TIME INERVALS WITHIN A SECOND.
C
         DO  L = 1,61
            WRITE (6,1) (ANG_2(K,L),K=1,6)
1           FORMAT (' ', 6(F8.5,3X))
```

```
      ENDDO
C
      RETURN
      END


      SUBROUTINE FILLANG_A (ANG, ANG_A,L)
      REAL ANG(6,4),ANG_A(6,4,61)
C
C     SUBROUTINE TO FILL THE 3-D JOINTS DISPLACEMENT MATRIX
C     WITH THE CORRECT ANGLES VALUE FOR DIFFERENT CONFIGURATIONS
C     AT EACH INTERVAL OF TIME.
C
      DO  K=1,6
         DO  J=1,4
            ANG_A(K,J,L) = ANG (K,J)
         ENDDO
      ENDDO
C
      RETURN
      END


      SUBROUTINE CURRENT_LD (sr88,sr99,CHANGE,CURR_LD,PERCENT,RLOAD,
     & IFRACTION,VARIATION,L)
      INTEGER IFRACTION,L
      REAL PERCENT,CURR_LD,VARIATION
      CHARACTER*4 CHANGE,sr88*1,sr99*1
C
C     SUBROUTINE TO CALCULATE THE VALUE OF THE PAY-LOAD AT EACH
C     INSTANT OF TIME FOR A ROBOT'S WORK-CYCLE.
C
      RL=L
      IF (sr99 .EQ. 'Y' .OR. sr99 .EQ. 'y')  THEN
         IF (CHANGE(1:1) .EQ. 'C' .OR. CHANGE(1:1) .EQ. 'c') THEN
           IF (sr88 .EQ. 'Y' .OR. sr88 .EQ. 'y')  THEN
C            PRINT *,'*************** LD IS DECREASING ***************'
             CURR_LD = RLOAD -((RL-1.)*PERCENT/60.*RLOAD)
           ELSE
             CURR_LD = RLOAD +((RL-1.)*PERCENT/60.*RLOAD)
C            PRINT *,'*************** LD IS INCREASING ***************'
           ENDIF
         ELSE
         IF ((((L-1)/IFRACTION)*IFRACTION) .EQ. (L-1)) THEN
             IF((L-1) .NE. 0) VARIATION = VARIATION + (PERCENT * RLOAD)
               IF (sr88 .EQ. 'Y' .OR. sr88 .EQ. 'y') THEN
                 CURR_LD = RLOAD - VARIATION
               ELSE
                 CURR_LD = RLOAD + VARIATION
               ENDIF
             ENDIF
         ENDIF
      ENDIF
C
      RETURN
      END


      SUBROUTINE GET_R_RI_LINKN (ROBOT,R,RI,RM,NLINK,SMALLA,
     & CURR_LD)
      INTEGER NLINK
      REAL R(4,6),RI(4,4,6),RM(6),SMALLA(6),CURR_LD
      CHARACTER*5 ROBOT
C
C     SUBROUTINE TO CALCULATE THE C.G. VECTOR AND INERTIA MATRIX
C     FOR THE LAST LINK AT EACH INSTANT OF TIME AS A FUNCTION OF
C     THE VARYING PAYLOAD AND DEPENDENT ON THE ROBOT'S TYPE.
C
      IF (ROBOT(1:1) .EQ. 'S' .OR. ROBOT(1:1) .EQ. 's') THEN
        R(3,NLINK) = ((RM(NLINK)*SMALLA(NLINK)/2.0)-(0.1*CURR_LD))/
     &  (CURR_LD+RM(NLINK))
        RI(1,1,3)=0.002298+(2./5.*(0.1**2)*CURR_LD)+(CURR_LD*(0.1**2))
        RI(2,2,3)=0.002298+(2./5.*(0.1**2)*CURR_LD)+(CURR_LD*(0.1**2))
        RI(3,3,3) = 0.001269 + (2./5.*(0.1**2)*CURR_LD)
      ELSE
C
C *  For 6 d.o.f. PUMA robot, consider that the 3rd. joint   *
```

```
C *  coordinate frame will remain the same.  Hence, update    *
C *  the centre of gravity and the inertia of the 3rd.link     *
C *  accordingly.                                              *
C                                                              *
         R(3,NLINK) = ((0.433 * CURR_LD)+1.407)/(CURR_LD+6.5)
         RI(1,1,NLINK) = 0.52186+ ((2./5.)* CURR_LD * (0.1**2))
         RI(2,2,NLINK) = 0.51877+ ((2./5.)* CURR_LD * (0.1**2))
         RI(3,3,NLINK) = 0.0048 + ((2./5.)* CURR_LD * (0.1**2))
       ENDIF
C
       RETURN
       END



       SUBROUTINE GET_RJ_LINKN (SQRK,NLINK,RI,R,PSINER,RM,RJ)
       REAL SQRK(4,4,6),RI(4,4,6),R(4,6),PSINER(4,4,6),RM(6),RJ(4,4,6)
       INTEGER NLINK
C
C      SUBROUTINE TO CALCULATE THE PSEUDO INERTIA MATRIX
C      FOR THE LAST LINK AT EACH INSTANT OF TIME AS A FUNCTION OF
C      THE VARYING PAYLOAD AND DEPENDENT ON THE ROBOT'S TYPE.
C
       DO  J=1,3
          DO  I=1,3
             SQRK (I,J,NLINK) =  RI(I,J,NLINK)/ RM(NLINK)
             IF (I .NE. J) THEN
                PSINER (I,J,NLINK) = SQRK (I,J,NLINK)
             ENDIF
          ENDDO
       ENDDO
       PSINER(1,1,NLINK) = .5*(-SQRK(1,1,NLINK)+SQRK(2,2,NLINK)
      & +SQRK(3,3,NLINK))
       PSINER(2,2,NLINK) = .5*(SQRK(1,1,NLINK) - SQRK(2,2,NLINK)
      & +SQRK(3,3,NLINK))
       PSINER(3,3,NLINK) = .5*(SQRK(1,1,NLINK) + SQRK(2,2,NLINK)
      & - SQRK(3,3,NLINK))
       PSINER(1,4,NLINK) = R(1,NLINK)
       PSINER(4,1,NLINK) = R(1,NLINK)
       PSINER(2,4,NLINK) = R(2,NLINK)
       PSINER(4,2,NLINK) = R(2,NLINK)
       PSINER(3,4,NLINK) = R(3,NLINK)
       PSINER(4,3,NLINK) = R(3,NLINK)
       PSINER(4,4,NLINK) = 1
C
       DO  J=1,4
          DO  I=1,4
             RJ(I,J,NLINK) = PSINER(I,J,NLINK) * RM(NLINK)
          ENDDO
       ENDDO
C
       RETURN
       END



       SUBROUTINE KINEM_VAR(VAR,VEL,ACCEL,P_ALL,V_ALL,A_ALL,III,NLINK,L)
       REAL VAR(6),VEL(6),ACCEL(6),
      &     P_ALL(6,25,61),V_ALL(6,25,61),A_ALL(6,25,61)
       INTEGER K,III,NLINK,L
C
C      SUBROUTINE TO GET THE TIME-DEPENDENT KINEMATIC VARIABLES FOR
C      JOINT'S TORQUE CALCULATION.
C
       DO  K =1,NLINK
          VAR(K) = P_ALL (K,III,L)
          VEL(K) = V_ALL (K,III,L)
          ACCEL(K)=A_ALL (K,III,L)
       ENDDO
C
       RETURN
       END



       SUBROUTINE BACK_RECURSION (NLINK,ROT,VAR,VEL,ACCEL,SMALLA,TWIST,
      & DEE,QROT,QSLID,A,QA,QQA,U,T,TDOT,TDOTS)
       REAL VAR(6),VEL(6),ACCEL(6),SMALLA(6),TWIST(6),DEE(6),
      &     QROT(4,4),QSLID(4,4),A(4,4,6),QA(4,4,6),QQA(4,4,6),U(4,4,6),
      &     T(4,4,6),TDOT(4,4,6),TDOTSA(4,4,6),TDOTS(4,4,6),TERM3(4,4,6)
```

```
      INTEGER NLINK,ROT(6)
C
C     SUBROUTINE TO DO THE BACK RECURSION PART OF THE DYNAMIC MODEL
C     TO FIND THE REQUIRED JOINT TORQUES AT EACH SINGLE INSTANT OF
C     THE ROBOT WORK CYCLE.
C
      DO  K = 1, NLINK
         CALL A_QA_QQA (K,ROT,VAR,VEL,ACCEL,SMALLA,TWIST,DEE,
     &   QROT,QSLID,A,QA,QQA)
C
         IF (K .EQ. 1) THEN
C           CALCULATE [T],[U],[TDOT] AND [TDOTS]  FOR THE FIRST LINK
C                                    (K=1)
            CALL T1 (K,A,T)
            CALL U1 (K,QA,U)
            CALL TDOT1 (K,U,VEL,TDOT)
            CALL TDOTS1 (K,ROT,VEL,QQA,U,ACCEL,TDOTS)
         ELSE
C           CALCULATE [T],[U],[TDOT] AND [TDOTS]  FOR LINK NO. K
C                                    (K>1)
            CALL T_U_TDOT_K (K,A,T,QA,U,VEL,TDOT)
            CALL GETTERM3 (K,TDOT,QA,TDOTQA,VEL,TERM3)
            CALL ML3BY3 (TDOTS,A,TDOTSA,(K-1),K,K)
            CALL TDOTSK (K,ROT,T,QQA,TERM3,TDOTSA,U,VEL,ACCEL,TDOTS)
         ENDIF
C
      ENDDO
      RETURN
      END



      SUBROUTINE A_QA_QQA (K,ROT,VAR,VEL,ACCEL,SMALLA,TWIST,DEE,
     &QROT,QSLID,A,QA,QQA)
      REAL VAR(6),VEL(6),ACCEL(6),SMALLA(6),TWIST(6),DEE(6),
     &     QROT(4,4),QSLID(4,4),A(4,4,6),QA(4,4,6),QQA(4,4,6)
      INTEGER K,ROT(6)
C
C     SUBROUTINE TO CALCULATE MATRICES [A], [QA],[QQA] FOR
C     EITHER REVOLUTE OR SLIDIND JOINT.
C
        IF (ROT(K) .EQ. 1) THEN
          CALL A_ROT (K,VAR,VEL,ACCEL,SMALLA,TWIST,DEE,A)
          CALL ML2BY3 (QROT,A,QA,K)
          CALL ML2BY3 (QROT,QA,QQA,K)
        ELSE
          CALL A_SLID (K,VAR,VEL,ACCEL,SMALLA,TWIST,DEE,A)
          CALL ML2BY3 (QSLID,A,QA,K)
C         ** NOTE THAT [QQA] FOR SLIDING JOINT = 0.0  **
        ENDIF
C
      RETURN
      END



      SUBROUTINE A_ROT (K,VAR,VEL,ACCEL,SMALLA,TWIST,DEE,A)
      REAL VAR(6),VEL(6),ACCEL(6),SMALLA(6),TWIST(6),DEE(6),A(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE THE TRANSFORMATION MATRIX [A] FOR
C     A REVOLUTE JOINT.
C
      A(1,1,K) = COS (VAR(K))
      A(1,2,K) = -SIN(VAR(K)) * COS (TWIST(K))
      A(1,3,K) =  SIN(VAR(K)) * SIN (TWIST(K))
      A(1,4,K) =  SMALLA(K) * COS (VAR(K))
      A(2,1,K) =  SIN (VAR(K))
      A(2,2,K) =  COS (VAR(K)) * COS (TWIST(K))
      A(2,3,K) = -COS (VAR(K)) * SIN (TWIST(K))
      A(2,4,K) =  SMALLA(K) * SIN (VAR(K))
      A(3,1,K) =  0.0
      A(3,2,K) =  SIN(TWIST(K))
      A(3,3,K) =  COS(TWIST(K))
      A(3,4,K) =  DEE(K)
      A(4,1,K) =  0.0
      A(4,2,K) =  0.0
      A(4,3,K) =  0.0
      A(4,4,K) =  1.0
C
```

```
      RETURN
      END


      SUBROUTINE A_SLID (K,VAR,VEL,ACCEL,SMALLA,TWIST,DEE,A)
      REAL VAR(6),VEL(6),ACCEL(6),SMALLA(6),TWIST(6),DEE(6),A(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE THE TRANSFORMATION MATRIX [A] FOR
C     A SLIDING JOINT.
C
      A(1,1,K) = COS (VAR(K-1))
      A(1,2,K) = -SIN(VAR(K-1)) * COS (TWIST(K))
      A(1,3,K) =  SIN(VAR(K-1)) * SIN (TWIST(K))
      A(1,4,K) =  SMALLA(K) * COS (VAR(K))
      A(2,1,K) =  SIN (VAR(K-1))
      A(2,2,K) =  COS (VAR(K-1)) * COS (TWIST(K))
      A(2,3,K) = -COS (VAR(K-1)) * SIN (TWIST(K))
      A(2,4,K) =  SMALLA(K) * SIN (VAR(K))
      A(3,1,K) =  0.0
      A(3,2,K) =  SIN(TWIST(K))
      A(3,3,K) =  COS(TWIST(K))
      A(3,4,K) =  VAR(K)
      A(4,1,K) =  0.0
      A(4,2,K) =  0.0
      A(4,3,K) =  0.0
      A(4,4,K) =  1.0
C
      RETURN
      END


      SUBROUTINE T1 (K,A,T)
      REAL A(4,4,6),T(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO ASSIGN [T(1)] TO [A(1)]
C
      DO  J = 1,4
         DO  I = 1,4
            T(I,J,K) = A(I,J,K)
         ENDDO
      ENDDO
C
      RETURN
      END


      SUBROUTINE U1 (K,QA,U)
      REAL QA(4,4,6),U(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO ASSIGN [U(1)] TO [QA(1)]
C
      DO  J = 1,4
         DO  I = 1,4
            U(I,J,K) = QA(I,J,K)
         ENDDO
      ENDDO
C
      RETURN
      END


      SUBROUTINE TDOT1 (K,U,VEL,TDOT)
      REAL U(4,4,6),VEL(6),TDOT(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE THE FIRST DRIVATIVE OF THE
C     TRANSFORMATION MATRIX [T] FOR THE FIRST LINK.
C
      DO  J = 1,4
         DO  I = 1,4
            TDOT(I,J,K) = U(I,J,K) * VEL(K)
         ENDDO
      ENDDO
```

```
C
      RETURN
      END



      SUBROUTINE TDOTS1 (K,ROT,VEL,QQA,U,ACCEL,TDOTS)
      REAL VEL(6),ACCEL(6),QQA(4,4,6),U(4,4,6),TDOTS(4,4,6)
      INTEGER K,ROT(6)
C
C     SUBROUTINE TO CALCULATE THE SECOND DRIVATIVE OF THE
C     TRANSFORMATION MATRIX [T] OF THE FIRST LINK.  THIS IS
C     DONE FOR EITHER REVOLUTE OR SLIDING JOINT.
C
      IF (ROT(K) .EQ. 1) THEN
         CALL TDOTSROT1 (K,VEL,QQA,U,ACCEL,TDOTS)
      ELSE
         CALL TDOTSSLID1 (K,U,ACCEL,TDOTS)
      ENDIF
C
      RETURN
      END



      SUBROUTINE TDOTSROT1 (K,VEL,QQA,U,ACCEL,TDOTS)
      REAL VEL(6),VELSQR(6),ACCEL(6),QQA(4,4,6),U(4,4,6),
     &     TDOTS(4,4,6),QQAVL2(4,4,6),QAACEL(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE THE SECOND DRIVATIVE OF THE
C     TRANSFORMATION MATRIX [T] OF THE FIRST JOINT (REVOLUTE
C     JOINT ONLY)
C
      VELSQR(K) = VEL(K)**2
      DO  J = 1,4
         DO  I = 1,4
            QQAVL2(I,J,K) = QQA(I,J,K) * VELSQR(K)
         ENDDO
      ENDDO
C
      DO  J = 1,4
         DO  I = 1,4
            QAACEL(I,J,K) = U(I,J,K) * ACCEL(K)
         ENDDO
      ENDDO
C
      DO  J = 1,4
         DO  I = 1,4
            TDOTS(I,J,K) = QQAVL2(I,J,K) + QAACEL(I,J,K)
         ENDDO
      ENDDO
C
      RETURN
      END



      SUBROUTINE TDOTSSLID1 (K,U,ACCEL,TDOTS)
      REAL U(4,4,6),ACCEL(6),TDOTS(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE THE SECOND DRIVATIVE OF THE
C     TRANSFORMATION MATRIX [T] OF THE FIRST JOINT (PRISMATIC
C     JOINT ONLY)
C
      DO  J = 1,4
         DO  I = 1,4
            TDOTS(I,J,K) = U(I,J,K) * ACCEL(K)
         ENDDO
      ENDDO
C
      RETURN
      END



      SUBROUTINE T_U_TDOT_K (K,A,T,QA,U,VEL,TDOT)
      REAL A(4,4,6),T(4,4,6),QA(4,4,6),U(4,4,6),VEL(6),TDOT(4,4,6),
     &     TDOTA(4,4,6),UVEL(4,4)
```

```
      INTEGER K
C
C     SUBROUTINE TO CALCULATE THE FIRST DRIVATIVE OF THE
C     TRANSFORMATION MATRIX [T] FOR JOINT NUMBER (>1).
C  ** CALCULATE [T] FOR K>1 **
      CALL ML3BY3 (T,A,T,(K-1),K,K)
C
C  ** CALCULATE [U] FOR K>1 **
      CALL ML3BY3 (T,QA,U,(K-1),K,K)
C
C  ** CALCULATE [TDOT] FOR K>1 **
      CALL ML3BY3 (TDOT,A,TDOTA,(K-1),K,K)
C
      DO  J = 1,4
         DO  I = 1,4
            UVEL(I,J) = U(I,J,K) * VEL(K)
         ENDDO
      ENDDO
C
      DO  J = 1,4
         DO  I = 1,4
            TDOT(I,J,K) = TDOTA(I,J,K)+UVEL(I,J)
         ENDDO
      ENDDO
C
      RETURN
      END



      SUBROUTINE GETTERM3 (K,TDOT,QA,TDOTQA,VEL,TERM3)
      REAL TDOT(4,4,6),QA(4,4,6),TERM3(4,4,6),TDOTQA(4,4,6),VEL(6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE A CERTAIN TERM USED BY THE
C     BACK RECURSION PART OF THE DYNAMIC MODEL FOR JOINT
C     NUMBER (K) > 1.
C
      CALL ML3BY3 (TDOT,QA,TDOTQA,(K-1),K,K)
      DO  J = 1,4
         DO  I = 1,4
            TERM3(I,J,K) = TDOTQA(I,J,K)*2.0* VEL(K)
         ENDDO
      ENDDO
C
      RETURN
      END



      SUBROUTINE TDOTSK (K,ROT,T,QQA,TERM3,TDOTSA,U,VEL,ACCEL,TDOTS)
      REAL T(4,4,6),QQA(4,4,6),TERM3(4,4,6),TDOTSA(4,4,6),U(4,4,6),
     &     VEL(6),ACCEL(6),TDOTS(4,4,6)
      INTEGER ROT(6),K
C
C     SUBROUTINE TO CALCULATE THE SECOND DRIVATIVE OF THE
C     TRANSFORMATION MATRIX FOR EITHER REVOLUTE OR PRISMATIC JOINT
C     (JOINT NUMBER (K) > 1)
C
      IF (ROT(K) .EQ. 1) THEN
         CALL TDOTS_ROTK (K,VEL,T,QQA,TDOTSA,TERM3,U,ACCEL,TDOTS)
      ELSE
         CALL TDOTS_SLIDK (K,TDOTSA,U,ACCEL,TERM3,TDOTS)
      ENDIF
C
      RETURN
      END



      SUBROUTINE TDOTS_ROTK (K,VEL,T,QQA,TDOTSA,TERM3,U,ACCEL,TDOTS)
      REAL VEL(6),VELSQR(6),T(4,4,6),QQA(4,4,6),TQQA(4,4,6),
     &     TDOTSA(4,4,6),TERM3(4,4,6),TERM2(4,4,6),U(4,4,6),ACCEL(6),
     &     TDOTS(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE THE SECOND DRIVATIVE OF THE
C     TRANSFORMATION MATRIX [T] FOR A REVOLUTE JOINT NUMBER K (>1)
C
         VELSQR(K) = VEL(K)**2
```

```
          CALL ML3BY3 (T,QQA,TQQA,(K-1),K,K)
C
       DO  J = 1,4
          DO  I = 1,4
             TERM2(I,J,K) = TQQA(I,J,K) * VELSQR(K)
          ENDDO
       ENDDO
C
       DO  J = 1,4
         DO  I = 1,4
           TDOTS(I,J,K)=TDOTSA(I,J,K) + TERM2(I,J,K) +(U(I,J,K)*ACCEL(K))
     &       + TERM3(I,J,K)
         ENDDO
       ENDDO
C
       RETURN
       END



       SUBROUTINE TDOTS_SLIDK (K,TDOTSA,U,ACCEL,TERM3,TDOTS)
       REAL TDOTSA(4,4,6),U(4,4,6),ACCEL(6),TERM3(4,4,6),TDOTS(4,4,6)
       INTEGER K
C
C      SUBROUTINE TO CALCULATE THE SECOND DRIVATIVE OF THE
C      TRANSFORMATION MATRIX [T] FOR A PRISMATIC JOINT NUMBER K (>1)
C
       DO  J = 1,4
          DO  I = 1,4
       TDOTS(I,J,K) = TDOTSA(I,J,K) + U(I,J,K)*ACCEL(K)
     &              + TERM3(I,J,K)
          ENDDO
       ENDDO
C
       RETURN
       END



       SUBROUTINE FD_RECURSION (NLINK,A,TDOTS,RJ,RM,R,U,G,F_ALL,
     & K_ALL,G_ALL,III,L)
       REAL TDOTS(4,4,6),RJ(4,4,6),RM(6),R(4,6),U(4,4,6),G(6,4),
     &      F_ALL(6,25,61),K_ALL(6,25,61),G_ALL(6,25,61),
     &      D(4,4,6),TRNDTS(4,4,6),UD(4,4,6),C(4,6),UC(4,6),
     &      A(4,4,6)

       INTEGER NLINK
C
C      SUBROUTINE TO CALCULATE THE TERMS REQUIRED BY THE FORWARD
C      RECURSION  FOR THE DYNAMIC MODEL.
C
       DO K= NLINK ,1,-1
          CALL MATRNS (TDOTS,TRNDTS,K)
          CALL CALC_D_C (RJ,TRNDTS,RM,R,A,D,C,NLINK,K)
C
C         ** CALCULATE THE FORCE F **
C
          CALL ML3BY3 (U,D,UD,K,K,K)
C
C
C
          TR = 0.0
          DO I = 1,4
             TR = TR + UD(I,I,K)
          ENDDO
C
          CALL ML3BY2 (U,C,UC,K,K,K)
C
          FPRT2 = 0.0
          DO I=1,4
             FPRT2 = FPRT2 + G(K,I) * UC(I,K)
          ENDDO
C
       IF (L .EQ. 3 .AND. K .EQ. 1 .AND. III .EQ. 1) THEN
       END IF
          F_ALL(K,III,L) = TR - FPRT2
          G_ALL(K,III,L) = FPRT2
          K_ALL(K,III,L) = TR
C
       PRINT *,'F_ALL(',K,',',III,',',L,') = ',F_ALL(K,III,L)
```

```
      ENDDO
C
      RETURN
      END


      SUBROUTINE CALC_D_C (RJ,TRNDTS,RM,R,A,D,C,NLINK,K)
      REAL RJ(4,4,6),TRNDTS(4,4,6),RM(6),R(4,6),A(4,4,6),
     &     D(4,4,6),C(4,6)
      INTEGER NLINK,K
C
C     SUBROUTINE TO CALCULATE TERMS D & C THAT ARE REQUIRED BY THE
C     FORWARD RECURSION FOR EITHER THE LAST JOINT OR OTHER LINK.
C
      IF (K .EQ. NLINK) THEN
         CALL D_C_LINKN (RJ,TRNDTS,RM,R,D,C,K)
      ELSE
         CALL D_C_LINKK (RJ,TRNDTS,RM,R,A,D,C,K)
      ENDIF
C
      RETURN
      END


      SUBROUTINE D_C_LINKN (RJ,TRNDTS,RM,R,D,C,K)
      REAL RJ(4,4,6),TRNDTS(4,4,6),RM(6),R(4,6),D(4,4,6),C(4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE TERMS D & C THAT ARE REQUIRED BY THE
C     FORWARD RECURSION FOR THE LAST JOINT ONLY.
C
         CALL ML3BY3 (RJ,TRNDTS,D,K,K,K)
         DO  I=1,4
            C(I,K) = RM(K) * R(I,K)
         ENDDO
C
      RETURN
      END


      SUBROUTINE D_C_LINKK (RJ,TRNDTS,RM,R,A,D,C,K)
      REAL RJ(4,4,6),TRNDTS(4,4,6),RM(6),R(4,6),D(4,4,6),C(4,6),
     &     DPRT1(4,4,6),DPRT2(4,4,6),CPRT1(4,6),CPRT2(4,6),
     &     A(4,4,6)
      INTEGER K
C
C     SUBROUTINE TO CALCULATE TERMS D & C THAT ARE REQUIRED BY THE
C     FORWARD RECURSION FOR JOINT NUMBER K (< NLINK)
C
         CALL ML3BY3 (RJ,TRNDTS,DPRT1,K,K,K)
         CALL ML3BY3 (A,D,DPRT2,(K+1),(K+1),K)
         DO J=1,4
            DO I=1,4
               D(I,J,K) = DPRT1(I,J,K) + DPRT2(I,J,K)
            ENDDO
         ENDDO
C
         DO  I=1,4
            CPRT1(I,K) = RM(K) * R(I,K)
         ENDDO
         CALL ML3BY2 (A,C,CPRT2,(K+1),(K+1),K)
         DO  I=1,4
            C(I,K) = CPRT1(I,K) + CPRT2(I,K)
         ENDDO
C
      RETURN
      END


      SUBROUTINE GETMAXMIN(FMAX,FMIN,GMAX,GMIN,KMAX,KMIN,AMAX,AMIN,
     & VMAX,VMIN,PMAX,PMIN,F_ALL,K_ALL,G_ALL,A_ALL,V_ALL,P_ALL,
     & PAYLD,RLDMAX,RLDMIN,NUMTRJ)
C
      REAL FMAX(6),FMIN(6),GMAX(6),GMIN(6),KMAX(6),KMIN(6),AMAX(6),
     &     AMIN(6),VMAX(6),VMIN(6),PMAX(6),PMIN(6),
     &     F_ALL(6,25,61),K_ALL(6,25,61),G_ALL(6,25,61),
```

```
     &     FCENT(6,25),VCENT(6,25),
     &     A_ALL(6,25,61),V_ALL(6,25,61),P_ALL(6,25,61),
     &     PAYLD(61),RLDMAX,RLDMIN,CENTMAX,CENTMIN
      INTEGER NUMTRJ
      CHARACTER*5 TRAJ(6),SPACE
      CHARACTER*4 TRJ
C
C      **********  FIND MAX MIN  ***********
C
C     #### INITIALISE ####
C
      DO  K = 1, 6
         FMAX(K) = F_ALL (K,NUMTRJ,1)
         FMIN(K) = F_ALL (K,NUMTRJ,1)
         GMAX(K) = G_ALL (K,NUMTRJ,1)
         GMIN(K) = G_ALL (K,NUMTRJ,1)
         KMAX(K) = K_ALL (K,NUMTRJ,1)
         KMIN(K) = K_ALL (K,NUMTRJ,1)
         AMAX(K) = A_ALL (K,NUMTRJ,1)
         AMIN(K) = A_ALL (K,NUMTRJ,1)
         VMAX(K) = V_ALL (K,NUMTRJ,1)
         VMIN(K) = V_ALL (K,NUMTRJ,1)
         PMAX(K) = P_ALL (K,NUMTRJ,1)
         PMIN(K) = P_ALL (K,NUMTRJ,1)
      ENDDO
C
      DO  K=1,6
         DO  III=1,NUMTRJ
            DO  L=1,61
              IF (F_ALL(K,III,L) .GT. FMAX(K)) FMAX(K)=F_ALL(K,III,L)
              IF (F_ALL(K,III,L) .LT. FMIN(K)) FMIN(K)=F_ALL(K,III,L)
              IF (G_ALL(K,III,L) .GT. GMAX(K)) GMAX(K)=G_ALL(K,III,L)
              IF (G_ALL(K,III,L) .LT. GMIN(K)) GMIN(K)=G_ALL(K,III,L)
              IF (K_ALL(K,III,L) .GT. KMAX(K)) KMAX(K)=K_ALL(K,III,L)
              IF (K_ALL(K,III,L) .LT. KMIN(K)) KMIN(K)=K_ALL(K,III,L)
              IF (A_ALL(K,III,L) .GT. AMAX(K)) AMAX(K)=A_ALL(K,III,L)
              IF (A_ALL(K,III,L) .LT. AMIN(K)) AMIN(K)=A_ALL(K,III,L)
              IF (V_ALL(K,III,L) .GT. VMAX(K)) VMAX(K)=V_ALL(K,III,L)
              IF (V_ALL(K,III,L) .LT. VMIN(K)) VMIN(K)=V_ALL(K,III,L)
              IF (P_ALL(K,III,L) .GT. PMAX(K)) PMAX(K)=P_ALL(K,III,L)
              IF (P_ALL(K,III,L) .LT. PMIN(K)) PMIN(K)=P_ALL(K,III,L)
              IF (SPACE(1:1) .EQ. 'C' .OR. SPACE(1:1) .EQ. 'c') THEN
                 IF(TRJ .EQ.'POLY')
     &                FCENT(K,III)=F_ALL(K,III,31)-F_ALL(K,III,61)
                 IF(TRJ .EQ.'NC2')
     &                FCENT(K,III)=F_ALL(K,III,21)-F_ALL(K,III,61)
              ELSEIF (SPACE(1:1).EQ.'J' .OR. SPACE(1:1).EQ.'j') THEN
                 IF(TRAJ(K) .EQ.'POLY')
     &                FCENT(K,III)=F_ALL(K,III,31)-F_ALL(K,III,61)
                 IF(TRAJ(K) .EQ.'NC2')
     &                FCENT(K,III)=F_ALL(K,III,21)-F_ALL(K,III,61)
              ENDIF
              IF (FCENT(K,III) .GT. CNTMAX)   CNTMAX = FCENT(K,III)
              IF (FCENT(K,III) .LT. CNTMIN)   CNTMIN = FCENT(K,III)
              IF (SPACE(1:1) .EQ. 'C' .OR. SPACE(1:1) .EQ. 'c') THEN
                 IF (TRJ .EQ. 'POLY')   VCENT(K,III) = V_ALL(K,III,31)
                 IF (TRJ .EQ. 'NC2' )   VCENT(K,III) = V_ALL(K,III,21)
              ELSEIF (SPACE(1:1).EQ.'J' .OR. SPACE(1:1).EQ.'j') THEN
                 IF (TRAJ(K) .EQ. 'POLY') VCENT(K,III)=V_ALL(K,III,31)
                 IF (TRAJ(K) .EQ. 'NC2' ) VCENT(K,III)=V_ALL(K,III,21)
              ENDIF
              IF (VCENT(K,III) .GT. VCENTMX) VCENTMX=VCENT(K,III)
              IF (VCENT(K,III) .LT. VCENTMN) VCENTMN=VCENT(K,III)
            ENDDO
         ENDDO
      ENDDO
C
      RLDMAX = -100.
      RLDMIN =  100.
      DO I = 1,61
        IF(PAYLD(I) .GE. RLDMAX)  RLDMAX = PAYLD(I)
        IF(PAYLD(I) .LE. RLDMIN)  RLDMIN = PAYLD(I)
      ENDDO
C
      RETURN
      END


      SUBROUTINE PLOTPOS (X,P_ALL,PMIN,PMAX,NUMTRJ)
```

```
      REAL X(0:61),P_ALL(6,25,61),PTRJ(61),PMIN(6),PMAX(6)
      INTEGER NUMTRJ
C
C     SUBROUTINE TO PLOT DISPLACEMENT VERSUS TIME WITHIN ONE SECOND
C     FOR ALL THE JOINTS
C
      CALL BEGIN
      DO  K=1, 6
         CALL DISPLAY ('A4',200.,200., 900.,650.)
         CALL SCALE (0.,1.,PMIN(K),PMAX(K))
         CALL PEN(2)
         CALL XAXIS('X',0.,PMIN(K),1.,1,11,1,4)
         CALL CHR_SIZE (.15,.3)
         CALL AXLBL ('X',0.,PMIN(K),1.,1,11,'F4.1')
         CALL YAXIS('Y',0.,PMIN(K),PMAX(K),1,6,2,2)
         CALL AXLBL ('Y',0.,PMIN(K),PMAX(K),1,6,'F6.2')
         CALL PEN(3)
            DO  I=1, NUMTRJ
               DO  L= 1, 61
                  PTRJ(L) = P_ALL(K,I,L)
               ENDDO
               CALL CURVE (X,PTRJ,61,'4',5.)
            ENDDO
         CALL SCALE_OFF
         CALL CHR_STYLE (45,35,'SS')
         CALL CHR_SIZE (.25,.4)
         CALL PEN(1)
         CALL TEXT ('A4','OFF',550.,140.,'TIME$$',4)
         CALL CHR_SLOPE (20.)
         CALL CHR_SIZE (.25,.4)
         CALL TEXT ('A4','OFF',550.,110.,'(IN SECONDS)$$',4)
         CALL SLOPE_OFF
         CALL CHR_SIZE (.25,.4)
         CALL TEXT_DIR (90.)
         CALL TEXT ('A4','OFF',90.,450.,
     &   'DISPLACEMENT IN Radians OR Meters$$',4)
         CALL DIR_OFF
         CALL CHR_SIZE (.3,.6)
         CALL PEN(1)
         CALL TEXT ('A4','OFF',550.,50.,
     &   ' DISPLACEMENT TRAJECTORY OF LINK No.    $$',4)
         CALL INTEG_NUM ('A4','OFF',800.,50.,K,'I1',4)
         CALL TEXT ('A4','OFF',550.,10.,'WITHIN ONE SECOND$$',4)
         CALL NEXT('A4')
      ENDDO
C
      RETURN
      END



      SUBROUTINE PLOTVEL (X,V_ALL,VMIN,VMAX,NUMTRJ)
      REAL X(0:61),V_ALL(6,25,61),VTRJ(61),VMIN(6),VMAX(6)
      INTEGER NUMTRJ
C
C     SUBROUTINE TO PLOT VELOCITY VERSUS TIME WITHIN ONE SECOND
C     FOR ALL THE JOINTS
C
      CALL BEGIN
      DO K = 1,6
         CALL DISPLAY ('A4',200.,200., 900.,650.)
         CALL SCALE (0.,1.,VMIN(K),VMAX(K))
         CALL PEN(2)
         CALL XAXIS('X',0.,VMIN(K),1.,1,11,1,4)
         CALL CHR_SIZE (.15,.3)
         CALL AXLBL ('X',0.,VMIN(K),1.,1,11,'F4.1')
         CALL YAXIS('Y',0.,VMIN(K),VMAX(K),1,6,2,2)
         CALL AXLBL ('Y',0.,VMIN(K),VMAX(K),1,6,'F7.2')
         CALL PEN(3)
         DO I =1, NUMTRJ
            DO L= 1, 61
               VTRJ(L) = V_ALL(K,I,L)
            ENDDO
            CALL CURVE (X,VTRJ,61,'4',5.)
         ENDDO
         CALL SCALE_OFF
         CALL CHR_STYLE (45,35,'SS')
         CALL CHR_SIZE (.25,.4)
         CALL PEN(1)
         CALL TEXT ('A4','OFF',550.,140.,'TIME$$',4)
```

```
          CALL CHR_SLOPE (20.)
          CALL CHR_SIZE (.25,.4)
          CALL TEXT ('A4','OFF',550.,110.,'(IN SECONDS)$$',4)
          CALL SLOPE_OFF
          CALL CHR_SIZE (.25,.4)
          CALL TEXT_DIR (90.)
          CALL TEXT ('A4','OFF',90.,450.,
     &    'VELOCITY IN Rad/Sec OR M/Sec$$',4)
          CALL DIR_OFF
          CALL CHR_SIZE (.3,.6)
          CALL PEN(1)
          CALL TEXT ('A4','OFF',550.,50.,
     &    ' VELOCITY TRAJECTORIES OF LINK NO.    $$',4)
          CALL INTEG_NUM ('A4','OFF',800.,50.,K,'I1',4)
          CALL TEXT ('A4','OFF',550.,10.,'WITHIN ONE SECOND$$',4)
          CALL NEXT('A4')
        ENDDO
C
        RETURN
        END



        SUBROUTINE PLOTACC (X,A_ALL,AMIN,AMAX,NUMTRJ)
        REAL X(0:61),A_ALL(6,25,61),ATRJ(61),AMIN(6),AMAX(6)
        INTEGER NUMTRJ
C
C       SUBROUTINE TO PLOT ACCELERATION VERSUS TIME WITHIN ONE SECOND
C       FOR ALL THE JOINTS
C
        CALL BEGIN
        DO K=1, 6
           CALL DISPLAY ('A4',200.,200., 900.,650.)
           CALL SCALE (0.,1.,AMIN(K),AMAX(K))
           CALL PEN(2)
           CALL XAXIS('X',0.,0.,1.,0,11,1,4)
           CALL CHR_SIZE (.15,.3)
           CALL AXLBL ('X',0.,0.,1.,1,11,'F4.1')
           CALL YAXIS('Y',0.,AMIN(K),AMAX(K),1,6,2,2)
           CALL AXLBL ('Y',0.,AMIN(K),AMAX(K),1,6,'F6.2')
           CALL PEN(3)
              DO I =1, NUMTRJ
                 DO L= 1, 61
                    ATRJ(L) = A_ALL(K,I,L)
                 ENDDO
              CALL CURVE (X,ATRJ,61,'4',5.)
           ENDDO
           CALL CHR_STYLE (45,35,'SS')
           CALL CHR_SIZE (.25,.4)
           CALL PEN(1)
           CALL TEXT ('A4','ON',X(60),0.0,'TIME$$',12)
           CALL CHR_SLOPE (20.)
           CALL CHR_SIZE (.18,.35)
           CALL TEXT ('A4','ON',X(60),0.0,'      in seconds$$',12)
           CALL SCALE_OFF
           CALL SLOPE_OFF
           CALL CHR_SIZE (.25,.4)
           CALL TEXT_DIR (90.)
           CALL TEXT('A4','OFF',90.,425.,
     &     'ACCELERATION IN Rad/Sec  or M/Sec $$',4)
           CALL CHR_SIZE (.15,.3)
           CALL TEXT ('A4','OFF',80.,425.,
     &     '                                2             2$$',4)
           CALL DIR_OFF
           CALL CHR_SIZE (.3,.55)
           CALL PEN(1)
           CALL TEXT ('A4','OFF',550.,100.,
     &     ' ACCELERATION TRAJECTORY OF LINK No.    $$',4)
           CALL INTEG_NUM ('A4', 'OFF', 800.,100., K, 'I1', 4)
           CALL TEXT ('A4','OFF',550.,50.,'WITHIN ONE SECOND$$',4)
           CALL NEXT('A4')
        ENDDO
C
        RETURN
        END



        SUBROUTINE PLOTLOAD (X,PAYLD,RLMIN,RLMAX)
        REAL X(0:61),PAYLD(61),RLMIN,RLMAX
```

```
C
C     SUBROUTINE TO PLOT PAYLOAD VERSUS TIME WITHIN ONE WORK-CYCLE
C
      CALL BEGIN
      CALL DISPLAY ('A4',200.,200.,900.,650.)
      CALL SCALE (0.,1.,RLDMIN,RLDMAX)
      CALL PEN(2)
      CALL XAXIS ('X',0.,RLDMIN,1.,1,11,1,4)
      CALL YAXIS ('Y',0.,RLDMIN,RLDMAX,1,6,1,1)
      CALL CHR_SIZE (.15,.3)
      CALL AXLBL ('X',0.,RLDMIN,1.,1,11,'F4.2')
      CALL AXLBL ('Y',0.,RLDMIN,RLDMAX,1,6,'F7.2')
      CALL PEN(3)
      CALL CURVE (X,PAYLD,61,'4',5.)
      CALL CHR_SIZE (.25,.4)
      CALL PEN(1)
      CALL TEXT ('A4','ON',X(60),RLDMIN,'TIME$$',12)
      CALL CHR_SLOPE (20.)
      CALL CHR_SIZE (.18,.35)
      CALL TEXT ('A4','ON',X(60),RLDMIN,'       in seconds$$',12)
      CALL SCALE_OFF
      CALL SLOPE_OFF
      CALL CHR_SIZE (.25,.4)
      CALL TEXT_DIR (90.)
      CALL TEXT ('A4','OFF',90.,425.,
     &'Payload in Kg$$',4)
      CALL DIR_OFF
      CALL CHR_SIZE (.3,.55)
      CALL PEN(1)
      CALL TEXT ('A4','OFF',550.,130.,
     &"  END-EFFECTOR PAYLOAD  $$",4)
      CALL CHR_SIZE (.25,.45)
      CALL CHR_SLOPE (15.)
      CALL TEXT ('A4','OFF',550.,100.,'within$$',4)
      CALL SLOPE_OFF
      CALL CHR_SIZE (.3,.55)
      CALL TEXT ('A4','OFF',550.,60.,
     &'ONE CYCLE-TIME$$',4)
      CALL NEXT('A4')
C
      RETURN
      END



      SUBROUTINE PLOTGRAVF (X,G_ALL,GMIN,GMAX,NLINK,NUMTRJ,ROT)
      REAL X(0:61),G_ALL(6,25,61),GTRJ(61),GMIN(6),GMAX(6)
      INTEGER NUMTRJ,ROT(6)
C
C     SUBROUTINE TO PLOT THE GRAVITITONAL LOADING ON THE REQUIRED
C     TORQUE FOR EACH LINK
C
      CALL BEGIN
      DO K=1,NLINK
         IF(GMAX(K) .NE. 0.0 .OR. GMIN(K) .NE. 0.0) THEN
            CALL DISPLAY ('A4',200.,200.,900.,650.)
            CALL SCALE (0.,1.,GMIN(K),GMAX(K))
            CALL PEN(2)
            CALL XAXIS ('X',0.,GMIN(K),1.,1,11,1,4)
            CALL YAXIS ('Y',0.,GMIN(K),GMAX(K),1,6,1,1)
            CALL CHR_SIZE (.15,.3)
            CALL AXLBL ('X',0.,GMIN(K),1.,1,11,'F4.2')
            CALL AXLBL ('Y',0.,GMIN(K),GMAX(K),1,6,'F7.2')
            CALL PEN(3)
               DO I=1,NUMTRJ
                  DO J=1,61
                     GTRJ(J) = G_ALL(K,I,J)
                  ENDDO
                  CALL CURVE (X,GTRJ,61,'4',5.)
               ENDDO
            CALL CHR_SIZE (.25,.4)
            CALL PEN(1)
            CALL TEXT ('A4','ON',X(60),GMIN(K),'TIME$$',12)
            CALL CHR_SLOPE (20.)
            CALL CHR_SIZE (.18,.35)
            CALL TEXT ('A4','ON',X(60),GMIN(K),'       in seconds$$',12)
            CALL SCALE_OFF
            CALL SLOPE_OFF
            CALL CHR_SIZE (.25,.4)
            CALL TEXT_DIR (90.)
```

```
            IF (ROT(K) .EQ. 0) THEN
              CALL TEXT ('A4','OFF',90.,425.,
     &        ' FORCES IN N$$',4)
              CALL DIR_OFF
              CALL CHR_SIZE (.3,.55)
              CALL PEN(1)
              CALL TEXT ('A4','OFF',550.,130.,
     &        '  THE GRAVITATIONAL FORCES FOR LINK NO.   $$',4)
            ELSE
              CALL TEXT ('A4','OFF',90.,425.,
     &        ' TORQUES IN Nm$$',4)
              CALL DIR_OFF
              CALL CHR_SIZE (.3,.55)
              CALL PEN(1)
              CALL TEXT ('A4','OFF',550.,130.,
     &        '  THE GRAVITATIONAL TORQUES FOR LINK NO.   $$',4)
            ENDIF
            CALL PEN(4)
            CALL INTEG_NUM ('A4','OFF',900.,130.,K,'I1',4)
            CALL PEN (1)
            CALL CHR_SIZE (.25,.45)
            CALL CHR_SLOPE (15.)
            CALL TEXT ('A4','OFF',550.,100.,'for$$',4)
            CALL SLOPE_OFF
            CALL CHR_SIZE (.3,.55)
            CALL TEXT ('A4','OFF',550.,65.,
     &      'TIME-VARYING PAYLOAD$$',4)
C     &     'VARIOUS VELOCITY TRAJECTORIES$$',4)
C           CALL TEXT ('A4','OFF',550.,45.,'WITHIN ONE SECOND$$',4)
            CALL NEXT('A4')
         ENDIF
       ENDDO
C
       RETURN
       END



       SUBROUTINE PLOTKINF (X,K_ALL,KMIN,KMAX,NLINK,NUMTRJ,ROT)
       REAL X(0:61),K_ALL(6,25,61),KTRJ(61),KMIN(6),KMAX(6)
       INTEGER NUMTRJ,ROT(3)
C
C      SUBROUTINE TO PLOT THE KINETIC FORCE LOADING ON THE REQUIRED
C      TORQUE FOR EACH LINK
C
       CALL BEGIN
       DO K=1,NLINK
          CALL DISPLAY ('A4',200.,200.,900.,650.)
          CALL SCALE (0.,1.,KMIN(K),KMAX(K))
          CALL PEN(2)
          CALL XAXIS ('X',0.,0.,1.,0,11,1,4)
          CALL YAXIS ('Y',0.,KMIN(K),KMAX(K),1,6,1,1)
          CALL CHR_SIZE (.15,.3)
          CALL AXLBL ('X',0.,0.,1.,1,11,'F4.2')
          CALL AXLBL ('Y',0.,KMIN(K),KMAX(K),1,6,'F7.2')
          CALL PEN(3)
          DO I=1,NUMTRJ
             DO J=1,61
                KTRJ(J) = K_ALL(K,I,J)
             ENDDO
             CALL CURVE (X,KTRJ,61,'4',5.)
          ENDDO
          CALL CHR_SIZE (.25,.4)
          CALL PEN(1)
          CALL TEXT ('A4','ON',X(60),0.0,'TIME$$',12)
          CALL CHR_SLOPE (20.)
          CALL CHR_SIZE (.18,.35)
          CALL TEXT ('A4','ON',X(60),0.0,'       in seconds$$',12)
          CALL SCALE_OFF
          CALL SLOPE_OFF
          CALL CHR_SIZE (.25,.4)
          CALL TEXT_DIR (90.)
          IF (ROT(K) .EQ. 0) THEN
             CALL TEXT ('A4','OFF',90.,425.,
     &       ' FORCES IN N$$',4)
             CALL DIR_OFF
             CALL CHR_SIZE (.3,.55)
             CALL PEN(1)
             CALL TEXT ('A4','OFF',550.,130.,
     &       '  THE INERTIAL AND CENTRIPETAL FORCES FOR LINK NO.   $$',4)
```

```
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     LINK'S LENGTH (L3) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF L3    ?'
      READ *, SMALLA(3,1)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF L3    ?'
      READ *, SMALLA(3,60)
      PRINT*,' '
C
      STEP = (SMALLA(3,60)-SMALLA(3,1))/60.0
      DO 1 III=2,59
         SMALLA(3,III) = SMALLA(3,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = SMALLA(3,III)
2     CONTINUE
C
      RETURN
      END



      SUBROUTINE PRAM_D1 (DEE,PARAMETER)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     JOINT'S OFFSET (D1) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF D1    ?'
      READ *, DEE(1,1)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF L3    ?'
      READ *, DEE(1,60)
      PRINT*,' '
C
      STEP = (DEE(1,60)-DEE(1,1))/60.0
      DO 1 III=2,59
         DEE(1,III) = DEE(1,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = DEE(1,III)
2     CONTINUE
C
      RETURN
      END



      SUBROUTINE PRAM_D2 (DEE,PARAMETER)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     JOINT'S OFFSET (D2) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF D2    ?'
      READ *, DEE(2,1)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF D2    ?'
      READ *, DEE(2,60)
      PRINT*,' '
C
      STEP = (DEE(2,60)-DEE(2,1))/60.0
      DO 1 III=2,59
         DEE(2,III) = DEE(2,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = DEE(2,III)
2     CONTINUE
C
      RETURN
      END
```

```
          ELSE
             CALL TEXT ('A4','OFF',90.,425.,
     &       ' TORQUES IN Nm$$',4)
             CALL DIR_OFF
             CALL CHR_SIZE (.3,.55)
             CALL PEN(1)
             CALL TEXT ('A4','OFF',550.,130.,
     &       '  THE INERTIAL AND CENTRIPETAL TORQUES FOR LINK NO.   $$',4)
          ENDIF
          CALL PEN (4)
          CALL INTEG_NUM ('A4','OFF',1000.,130.,K,'I1',4)
          CALL CHR_SIZE (.25,.45)
          CALL PEN(1)
          CALL CHR_SLOPE (15.)
          CALL TEXT ('A4','OFF',550.,100.,'for$$',4)
          CALL SLOPE_OFF
          CALL CHR_SIZE (.3,.55)
          CALL TEXT ('A4','OFF',550.,65.,
     &    'TIME-VARYING PAYLOAD$$',4)
C         CALL TEXT ('A4','OFF',550.,45.,'WITHIN ONE SECOND$$',4)
          CALL NEXT('A4')
       ENDDO
C
       RETURN
       END



       SUBROUTINE PLOTTORQUE (X,F_ALL,FMIN,FMAX,NLINK,NUMTRJ,ROT)
       REAL X(0:61),F_ALL(6,25,61),FTRJ(61),FMIN(6),FMAX(6)
       INTEGER ROT(6),NLINK,NUMTRJ
C
C      SUBROUTINE TO PLOT THE TOTAL REQUIRED TORQUE FOR EACH LINK
C      WITHIN ONE WORK-CYCLE
C
       CALL BEGIN
       DO K=1,NLINK
          CALL DISPLAY ('A4',200.,200.,900.,650.)
          CALL SCALE (0.,1.,FMIN(K),FMAX(K))
          CALL PEN(2)
          IF ((FMIN(K) .LE. 0.0 .AND. FMAX(K) .LE. 0.0) .OR.
     &       (FMIN(K) .GT. 0.0 .AND. FMAX(K) .GT. 0.0))  THEN
             CALL XAXIS ('X',0.,FMIN(K),1.,1,11,1,4)
             CALL CHR_SIZE (.15,.3)
             CALL AXLBL ('X',0.,FMIN(K),1.,1,11,'F4.2')
          ELSE
             CALL XAXIS ('X',0.,0.,1.,0,11,1,4)
             CALL CHR_SIZE (.15,.3)
             CALL AXLBL ('X',0.,0.,1.,1,11,'F4.2')
          ENDIF
          CALL YAXIS ('Y',0.,FMIN(K),FMAX(K),1,6,1,1)
          CALL CHR_SIZE (.15,.3)
          CALL AXLBL ('Y',0.,FMIN(K),FMAX(K),1,6,'F7.2')
          CALL PEN(3)
          DO I=1,NUMTRJ
             DO J=1,61
                FTRJ(J) = F_ALL(K,I,J)
             ENDDO
             CALL CURVE (X,FTRJ,61,'4',5.)
          ENDDO
          IF (K .EQ. 3 .AND. ROT(K) .EQ. 0) THEN
             CALL CHR_SIZE (.25,.4)
             CALL PEN(1)
             CALL TEXT ('A4','ON',X(60),FMIN(K),'TIME$$',12)
             CALL CHR_SLOPE (20.)
             CALL CHR_SIZE (.18,.35)
             CALL TEXT ('A4','ON',X(60),FMIN(K),'       in seconds$$',12)
          ELSE
             CALL CHR_SIZE (.25,.4)
             CALL PEN(1)
             CALL TEXT ('A4','ON',X(60),0.0,'TIME$$',12)
             CALL CHR_SLOPE (20.)
             CALL CHR_SIZE (.18,.35)
             CALL TEXT ('A4','ON',X(60),0.0,'       in seconds$$',12)
          ENDIF
          CALL SCALE_OFF
          CALL SLOPE_OFF
          CALL CHR_SIZE (.25,.4)
          CALL TEXT_DIR (90.)
          IF (ROT(K) .EQ. 0) THEN
```

```
            CALL TEXT ('A4','OFF',90.,425.,
     &      'DRIVING FORCES IN N$$',4)
            CALL DIR_OFF
            CALL CHR_SIZE (.3,.55)
            CALL PEN(1)
            CALL TEXT ('A4','OFF',550.,130.,
     &   '   THE REQUIRED FORCES FOR LINK NO.    $$',4)
          ELSE
            CALL TEXT ('A4','OFF',90.,425.,
     &      'DRIVING TORQUES IN Nm$$',4)
            CALL DIR_OFF
            CALL CHR_SIZE (.3,.55)
            CALL PEN(1)
            CALL TEXT ('A4','OFF',550.,130.,
     &   '   THE REQUIRED TORQUES FOR LINK NO.    $$',4)
          ENDIF
          CALL PEN(4)
          CALL INTEG_NUM ('A4','OFF',850.,130.,K,'I1',4)
          CALL PEN(1)
          CALL CHR_SIZE (.25,.45)
          CALL CHR_SLOPE (15.)
          CALL TEXT ('A4','OFF',550.,100.,'for$$',4)
          CALL SLOPE_OFF
          CALL CHR_SIZE (.3,.55)
          CALL TEXT ('A4','OFF',550.,65.,
     &    'TIME-VARYING PAYLOAD$$',4)
C         CALL TEXT ('A4','OFF',550.,30.,'WITHIN ONE SECOND$$',4)
          CALL NEXT('A4')
        ENDDO
C
        RETURN
        END
```

```
      SUBROUTINE CART_PATH (X1,Y1,Z1,ALPHA1,BETA1,GAMMA1,
     &X61,Y61,Z61,ALPHA61,BETA61,GAMMA61,Px,Py,Pz,ALPHA,BETA,GAMMA,TRJ)
      REAL*8 X1,Y1,Z1,X61,Y61,Z61, Px(61),Py(61),Pz(61)
       DIMENSION ALPHA(61),BETA(61),GAMMA(61)
      DIMENSION P(61), VEL(61), ACC(61)
      CHARACTER*4 TRJ
      COMMON RATIO
C
C THIS SUBROUTINE IS TO CALCULATE THE POINTS ON THE CART. PATH
C   AT 60 EQUAL TIME INTERVALS
C
          DEGRAD = 3.14159 / 180.0
          RADDEG = 180.0 / 3.14159
      PRINT *, X1,Y1,Z1,ALPHA1,BETA1,GAMMA1,
     &X61,Y61,Z61,ALPHA61,BETA61,GAMMA61
      R_LEN = SQRT((X61-X1)**2
     &              +(Y61-Y1)**2
     &              +(Z61-Z1)**2)
C
         ALPHA_R = ATAN2 (SQRT((Y61-Y1)**2 + (Z61-Z1)**2),(X61-X1))
C
         BETA_R  = ATAN2 (SQRT((X61-X1)**2 + (Z61-Z1)**2),(Y61-Y1))
C
        GAMMA_R = ATAN2 (SQRT((X61-X1)**2 + (Y61-Y1)**2) , (Z61-Z1))
C
      PRINT *,'ALPHA_R =',ALPHA_R,' BETA_R=',BETA_R,' GAMMA_R=',GAMMA_R
      PRINT *,'COS ALPHA_R =',COS(ALPHA_R)
C
      RATIO = R_LEN
      PRINT *, 'RATIO = R_LEN = ',RATIO
C
C
C
      IF (TRJ(1:1) .EQ. 'P' .OR. TRJ(1:1) .EQ. 'p') THEN
       CALL TRJPOLY (P,VEL,ACC)
      ELSE
       CALL TRJNC2 (P,VEL,ACC)
      ENDIF
C
C
C
      DO I=1,61
         Px(I) = X1 + (P(I)*(COS(ALPHA_R)))
         Py(I) = Y1 + (P(I)*(COS(BETA_R)))
         Pz(I) = Z1 + (P(I)*(COS(GAMMA_R)))
         PRINT *,'Px =',Px(I),'  Py=',Py(I),'  Pz=',Pz(I)
      ENDDO
C
C
C         ********* ORIENTATION INTERPOLATION *********
C
      ALPHA1_LOC = ALPHA1 * DEGRAD
      BETA1_LOC  = BETA1  * DEGRAD
      GAMMA1_LOC = GAMMA1 * DEGRAD
      ALPHA61_LOC = ALPHA61 * DEGRAD
      BETA61_LOC  = BETA61  * DEGRAD
      GAMMA61_LOC = GAMMA61 * DEGRAD
      ROT_X = (ALPHA61_LOC - ALPHA1_LOC)
      ROT_Y = (BETA61_LOC - BETA1_LOC)
      ROT_Z = (GAMMA61_LOC - GAMMA1_LOC)
C
      RATIO = ROT_X
      IF (TRJ(1:1) .EQ. 'P' .OR. TRJ(1:1) .EQ. 'p') THEN
       CALL TRJPOLY (P,VEL,ACC)
      ELSE
       CALL TRJNC2 (P,VEL,ACC)
      ENDIF
      DO I=1,61
        ALPHA(I) = ALPHA1_LOC + P(I)
        PRINT *,'ALPHA =',ALPHA(I)
      ENDDO
C
      RATIO = ROT_Y
      IF (TRJ(1:1) .EQ. 'P' .OR. TRJ(1:1) .EQ. 'p') THEN
       CALL TRJPOLY (P,VEL,ACC)
      ELSE
       CALL TRJNC2 (P,VEL,ACC)
      ENDIF
      DO I=1,61
         BETA(I) = BETA1_LOC + P(I)
```

```
         PRINT *,'BETA =',BETA(I)
      ENDDO
C
      RATIO = ROT_Z
      IF (TRJ(1:1) .EQ. 'P' .OR. TRJ(1:1) .EQ. 'p') THEN
       CALL TRJPOLY (P,VEL,ACC)
      ELSE
       CALL TRJNC2 (P,VEL,ACC)
      ENDIF
      DO I=1,61
         GAMMA(I) = GAMMA1_LOC + P(I)
         PRINT *,'GAMMA =',GAMMA(I)
      ENDDO
C
      RETURN
      END



      SUBROUTINE ROBOT_K (a2,a3,d3,d4,Px,Py,Pz,ALPHA,BETA,GAMMA,ANG)
C
       REAL*4 ANG(6,4),ROT(3,3),GAMMA,BETA,ALPHA,INT(6)
       REAL*8 Px,Py,Pz,a2,a3,d3,d4
C
C
C
C
C
       CALL THETA1( ANG , Px ,Py ,d3 )
C
       CALL THETA32(ANG,Px,Py,Pz,a2,a3,d3,d4 )
C
       CALL THETA22(ANG,a2,a3,d3,d4,Px,Py,Pz )
C
       CALL ROTAT(ALPHA, BETA, GAMMA, ROT)
C
       CALL THETA4(ROT, ANG, Px, Py, Pz, a2, a3, d3, d4, INT)
C
       CALL THETA5(ROT, ANG, INT)
C
       CALL  THETA6(ROT, ANG, INT)
C
       RETURN
       END




      SUBROUTINE THETA1 (ANG,Px,Py,d3)
      REAL*4 VAL(3),ANG(6,4)
      REAL*8 d3,Px,Py
C
C     THIS  SUBROUTINE  CALCULATES  THE  JOINT  ANGLE  FOR  THE
C                       FIRST  LINK.
C
      VAL(1) = ATAN2(Py,Px)
100   IF ((Px*Px + Py*Py - d3*d3).LT.0.0)  THEN
         d3 = d3-0.001
         GOTO 100
      ENDIF
      VAL(2) = ATAN2 (d3 , SQRT(Px*Px + Py*Py - d3*d3))
      VAL(3) = ATAN2 (d3 , -SQRT(Px*Px + Py*Py - d3*d3))
      ANG(1,1) = VAL(1) - VAL(2)
      ANG(1,2) = ANG(1,1)
      ANG(1,3) = VAL(1) - VAL(3)
      ANG(1,4) = ANG(1,3)
C
      RETURN
      END
```

```
      SUBROUTINE THETA22 (ANG,a2,a3,d3,d4,Px,Py,Pz)
      REAL*4 VAL(15),ANG(6,4)
      REAL*8 a2,a3,d3,d4,Px,Py,Pz
C
C     THIS  SUBROUTINE  COMPUTES  THE  JOINT  ANGLE  FOR  THE
C           SECOND LINK USING GEOMETRICAL APPROACH.
C
      VAL(1) = SQRT(Px*Px + Py*Py + Pz*Pz - d3*d3)
      VAL(2) = SQRT(Px*Px + Py*Py - d3*d3)
      VAL(3) = -Pz/VAL(1)
C
      VAL(4) = -1.0*(-1.0)*VAL(2)/VAL(1)
      VAL(5) = -1.0*( 1.0)*VAL(2)/VAL(1)
C
      VAL(6) = (a2*a2 + VAL(1)*VAL(1) - (d4*d4 + a3*a3))/(2.0*a2*VAL(1))
      VAL(7) = SQRT(1.0 - VAL(6)*VAL(6))
C
      VAL(8) = VAL(3)*VAL(6) + (-1.0)*VAL(4)*VAL(7)
      VAL(9) = VAL(3)*VAL(6) + ( 1.0)*VAL(4)*VAL(7)
      VAL(10) = VAL(3)*VAL(6) + ( 1.0)*VAL(5)*VAL(7)
      VAL(11) = VAL(3)*VAL(6) + (-1.0)*VAL(5)*VAL(7)
C
C
      VAL(12) = VAL(4)*VAL(6) - (-1.0)*VAL(3)*VAL(6)
      VAL(13) = VAL(4)*VAL(6) - ( 1.0)*VAL(3)*VAL(6)
      VAL(14) = VAL(5)*VAL(6) - ( 1.0)*VAL(3)*VAL(6)
      VAL(15) = VAL(5)*VAL(6) - (-1.0)*VAL(3)*VAL(6)
C
C
      ANG(2,1) = ATAN2 (VAL(8) , VAL(12))
      ANG(2,2) = ATAN2 (VAL(9) , VAL(13))
      ANG(2,3) = ATAN2 (VAL(10), VAL(14))
      ANG(2,4) = ATAN2 (VAL(11), VAL(15))
C
      RETURN
      END
```

```
      SUBROUTINE THETA32 (ANG,Px,Py,Pz,a2,a3,d3,d4)
      REAL*4 VAL(10),ANG(6,4)
      REAL*8 d3,d4,a2,a3,Px,Py,Pz
C
C     THIS  SUBROUTINE  COMPUTES THE  JOINT  ANGLE  FOR  THE
C           THIRD LINK USING GEOMETRICAL APPROACH.
C
100   VAL(1) = SQRT (Px*Px + Py*Py + Pz*Pz - d3*d3 )
      VAL(2) = a2*a2 + d4*d4 + a3*a3 - VAL(1)*VAL(1)
      VAL(3) = 2.0 * a2 * (SQRT(d4*d4 + a3*a3))
      VAL(4) = VAL(2) / VAL(3)
      VAL(5) = (-1.0)*SQRT(1.0-(VAL(4)*VAL(4)))
      VAL(6) = ( 1.0)*SQRT(1.0-(VAL(4)*VAL(4)))
      VAL(7) = ( 1.0)*SQRT(1.0-(VAL(4)*VAL(4)))
      VAL(8) = (-1.0)*SQRT(1.0-(VAL(4)*VAL(4)))
C
      VAL(9) = d4/(SQRT(d4*d4 + a3*a3))
      VAL(10)= SQRT(a3*a3)/SQRT(d4*d4 + a3*a3)
C
      ANG(3,1) = ATAN2((VAL(5)*VAL(10)-VAL(4)*VAL(9)) ,
     &                  VAL(4)*VAL(10)+VAL(5)*VAL(9))
C
      ANG(3,2) = ATAN2((VAL(6)*VAL(10)-VAL(4)*VAL(9)) ,
     &                  VAL(4)*VAL(10)+VAL(6)*VAL(9))
C
      ANG(3,3) = ATAN2((VAL(7)*VAL(10)-VAL(4)*VAL(9)) ,
     &                  VAL(4)*VAL(10)+VAL(7)*VAL(9))
C
      ANG(3,4) = ATAN2((VAL(8)*VAL(10)-VAL(4)*VAL(9)) ,
     &                  VAL(4)*VAL(10)+VAL(8)*VAL(9))
C
      RETURN
      END
```

```
      SUBROUTINE ROTAT (ALPHA,BETA,GAMMA,ROT)
      REAL GAMMA,BETA,ALPHA,ROT(3,3),VAL(3)
C
C     THIS SUBROUTINE COMPUTES THE ROTATION MATRIX GIVEN
C     THE ORIENTATION OF THE ROBOT WRIST.
C
      VAL1 = GAMMA
      VAL2 = BETA
      VAL3 = ALPHA
      ROT(1,1) = COS(VAL3) * COS(VAL2)
      ROT(1,2) = (COS(VAL3)*SIN(VAL2)*SIN(VAL1))-(SIN(VAL3)*COS(VAL1))
      ROT(1,3) = (COS(VAL3)*SIN(VAL2)*COS(VAL1))+(SIN(VAL3)*SIN(VAL1))
      ROT(2,1) = SIN(VAL3)*COS(VAL2)
      ROT(2,2) = (SIN(VAL3)*SIN(VAL2)*SIN(VAL1))+(COS(VAL3)*COS(VAL1))
      ROT(2,3) = (SIN(VAL3)*SIN(VAL2)*COS(VAL1))-(COS(VAL3)*SIN(VAL1))
      ROT(3,1) = -SIN(VAL2)
      ROT(3,2) = COS(VAL2)*SIN(VAL1)
      ROT(3,3) = COS(VAL2)*COS(VAL1)
C
      RETURN
      END
```

```
      SUBROUTINE THETA4 (ROT,ANG,Px,Py,Pz,a2,a3,d3,d4,INT)
      REAL*4 ROT(3,3), ANG(6,4), VAL(25), INT(8)
      REAL*8 Px,Py,Pz,a2,a3,d3,d4
C
C     THIS  SUBROUTINE  COMPUTES  THE  JOINT  ANGLE  FOR  THE
C                       FOURTH  LINK.
C
      VAL(1)=(-a3-a2*COS(ANG(3,1)))*Pz + (COS(ANG(1,1))*Px+SIN(ANG(1,1))
     &       *Py)*(a2*SIN(ANG(3,1))-d4)
      VAL(2)=(-a3-a2*COS(ANG(3,2)))*Pz + (COS(ANG(1,2))*Px+SIN(ANG(1,1))
     &       *Py)*(a2*SIN(ANG(3,2))-d4)
      VAL(3)=(-a3-a2*COS(ANG(3,3)))*Pz + (COS(ANG(1,3))*Px+SIN(ANG(1,2))
     &       *Py)*(a2*SIN(ANG(3,3))-d4)
      VAL(4)=(-a3-a2*COS(ANG(3,4)))*Pz + (COS(ANG(1,4))*Px+SIN(ANG(1,2))
     &       *Py)*(a2*SIN(ANG(3,4))-d4)
C
      VAL(5)=Pz*Pz + (COS(ANG(1,1))*Px + SIN(ANG(1,1))*Py)**2
      VAL(6)=Pz*Pz + (COS(ANG(1,3))*Px + SIN(ANG(1,3))*Py)**2
C
C
C                 INT(1) IS MADE UP OF ANG(3,1) AND ANG(1,1)
C                 INT(2) IS MADE UP OF ANG(3,2) AND ANG(1,2)
C                 INT(3) IS MADE UP OF ANG(3,3) AND ANG(1,3)
C                 INT(4) IS MADE UP OF ANG(3,4) AND ANG(1,4)
C
C
      INT(1) = VAL(1)/VAL(5)
      INT(2) = VAL(2)/VAL(5)
      INT(3) = VAL(3)/VAL(6)
      INT(4) = VAL(4)/VAL(6)
C
C
      VAL(12)=(a2*SIN(ANG(3,1))-d4)*Pz - (-a3-a2*COS(ANG(3,1)))*
     &       (COS(ANG(1,1))*Px+SIN(ANG(1,1))*Py)
      VAL(13)=(a2*SIN(ANG(3,2))-d4)*Pz - (-a3-a2*COS(ANG(3,2)))*
     &       (COS(ANG(1,2))*Px+SIN(ANG(1,2))*Py)
      VAL(14)=(a2*SIN(ANG(3,3))-d4)*Pz - (-a3-a2*COS(ANG(3,3)))*
     &       (COS(ANG(1,3))*Px+SIN(ANG(1,3))*Py)
      VAL(15)=(a2*SIN(ANG(3,4))-d4)*Pz - (-a3-a2*COS(ANG(3,4)))*
     &       (COS(ANG(1,4))*Px+SIN(ANG(1,4))*Py)
C
C
C               INT(5) IS MADE UP OF ANG(3,1) AND ANG(1,1)
C               INT(6) IS MADE UP OF ANG(3,2) AND ANG(1,2)
C               INT(7) IS MADE UP OF ANG(3,3) AND ANG(1,3)
C               INT(8) IS MADE UP OF ANG(3,4) AND ANG(1,4)
C
C
      INT(5) = VAL(12)/VAL(5)
      INT(6) = VAL(13)/VAL(5)
      INT(7) = VAL(14)/VAL(6)
      INT(8) = VAL(15)/VAL(6)
C
C
      VAL(20) = -ROT(1,3)*SIN(ANG(1,1)) + ROT(2,3)*COS(ANG(1,1))
      VAL(21) = -ROT(1,3)*SIN(ANG(1,3)) + ROT(2,3)*COS(ANG(1,3))
C
      VAL(22) = -ROT(1,3)*COS(ANG(1,1))*INT(5)-ROT(2,3)*SIN(ANG(1,1))*
     &          INT(5)+ROT(3,3)*INT(1)
      VAL(23) = -ROT(1,3)*COS(ANG(1,2))*INT(6)-ROT(2,3)*SIN(ANG(1,2))*
     &          INT(6)+ROT(3,3)*INT(2)
      VAL(24) = -ROT(1,3)*COS(ANG(1,3))*INT(7)-ROT(2,3)*SIN(ANG(1,3))*
     &          INT(7)+ROT(3,3)*INT(3)
      VAL(25) = -ROT(1,3)*COS(ANG(1,4))*INT(8)-ROT(2,3)*SIN(ANG(1,4))*
     &          INT(8)+ROT(3,3)*INT(4)
C
C
      ANG(4,1) = ATAN2 (VAL(20),VAL(22))
      ANG(4,2) = ATAN2 (VAL(20),VAL(23))
      ANG(4,3) = ATAN2 (VAL(21),VAL(24))
      ANG(4,4) = ATAN2 (VAL(21),VAL(25))
C
C
      RETURN
      END
```

```
      SUBROUTINE THETA5(ROT, ANG, INT)
      REAL*4 ROT(3,3), ANG(6,4), VAL(60), INT(8)
C
C     THIS  SUBROUTINE  COMPUTES  THE  JOINT  ANGLE  FOR  THE
C                        FIFTH  LINK.
C
C              CONFIGURATION = LEFTY
C              VARIABLE = INT(5), INT(6), (4,1), (4,2)
C
       VAL(1) = ROT(1,3)*(COS(ANG(1,1))*INT(5)*COS(ANG(4,1)) +
     &          SIN(ANG(1,1))*SIN(ANG(4,1)))
       VAL(2) = ROT(1,3)*(COS(ANG(1,2))*INT(6)*COS(ANG(4,2)) +
     &          SIN(ANG(1,2))*SIN(ANG(4,2)))
C
C              CONFIGURATION = RIGHTY
C              VARIABLE = INT(7), INT(8), (4,3), (4,4)
C
       VAL(3) = ROT(1,3)*(COS(ANG(1,3))*INT(7)*COS(ANG(4,3)) +
     &          SIN(ANG(1,3))*SIN(ANG(4,3)))
       VAL(4) = ROT(1,3)*(COS(ANG(1,4))*INT(8)*COS(ANG(4,4)) +
     &          SIN(ANG(1,4))*SIN(ANG(4,4)))
C
C
C
C
C                      PART  2
C
C               CONFIGURATION = LEFTY
C               VARIABLE = INT(5), INT(6), (4,1), (4,2)
C
       VAL(9) = ROT(2,3)*(SIN(ANG(1,1))*INT(5)*COS(ANG(4,1)) -
     &          COS(ANG(1,1))*SIN(ANG(4,1)))
       VAL(10) = ROT(2,3)*(SIN(ANG(1,2))*INT(6)*COS(ANG(4,2)) -
     &          COS(ANG(1,2))*SIN(ANG(4,2)))
C
C              CONFIGURATION = RIGHTY
C               VARIABLE = INT(7), INT(8), (4,3), (4,4)
C
       VAL(11) = ROT(2,3)*(SIN(ANG(1,3))*INT(7)*COS(ANG(4,3)) -
     &          COS(ANG(1,3))*SIN(ANG(4,3)))
       VAL(12) = ROT(2,3)*(SIN(ANG(1,4))*INT(8)*COS(ANG(4,4)) -
     &          COS(ANG(1,4))*SIN(ANG(4,4)))
C
C
C
C                      PART  3
C
C
       VAL(17) = ROT(3,3)*INT(1)*COS(ANG(4,1))
       VAL(18) = ROT(3,3)*INT(2)*COS(ANG(4,2))
       VAL(19) = ROT(3,3)*INT(3)*COS(ANG(4,3))
       VAL(20) = ROT(3,3)*INT(4)*COS(ANG(4,4))
C
C
C
C                      PART  4
C
C
      VAL(37) = ROT(1,3)*(-COS(ANG(1,1))*INT(1))
      VAL(38) = ROT(1,3)*(-COS(ANG(1,2))*INT(2))
      VAL(39) = ROT(1,3)*(-COS(ANG(1,3))*INT(3))
      VAL(40) = ROT(1,3)*(-COS(ANG(1,4))*INT(4))
C
C
C
C                      PART  5
C
C
      VAL(41) = ROT(2,3)*(-SIN(ANG(1,1))*INT(1))
      VAL(42) = ROT(2,3)*(-SIN(ANG(1,2))*INT(2))
      VAL(43) = ROT(2,3)*(-SIN(ANG(1,3))*INT(3))
      VAL(44) = ROT(2,3)*(-SIN(ANG(1,4))*INT(4))
C
C
C
C                      PART  6
C
C
      VAL(45) = ROT(3,3)*(-INT(5))
      VAL(46) = ROT(3,3)*(-INT(6))
```

```
      VAL(47) = ROT(3,3)*(-INT(7))
      VAL(48) = ROT(3,3)*(-INT(8))
C
C
C
C                    NUMERATOR  OF  ATAN  EXPRESSION
C
C
      VAL(49) = -(VAL(1) + VAL(9) - VAL(17))
      VAL(50) = -(VAL(2) + VAL(10) - VAL(18))
      VAL(51) = -(VAL(3) + VAL(11) - VAL(19))
      VAL(52) = -(VAL(4) + VAL(12) - VAL(20))
C
C
C
C                    DENOMINATOR  OF  ATAN  EXPRESSION
C
      VAL(53) = VAL(37) + VAL(41) + VAL(45)
      VAL(54) = VAL(38) + VAL(42) + VAL(46)
      VAL(55) = VAL(39) + VAL(43) + VAL(47)
      VAL(56) = VAL(40) + VAL(44) + VAL(48)
C
C
C                   FINAL  JOINT  ANGLES
C
C
      ANG(5,1) = ATAN2 (VAL(49),VAL(53))
      ANG(5,2) = ATAN2 (VAL(50),VAL(54))
      ANG(5,3) = ATAN2 (VAL(51),VAL(55))
      ANG(5,4) = ATAN2 (VAL(52),VAL(56))
C
      RETURN
      END
```

```
      SUBROUTINE THETA6 (ROT,ANG,INT)
      REAL*4 ROT(3,3), ANG(6,4), VAL(50), INT(8)
C
C     THIS SUBROUTINE COMPUTES THE JOINT ANGLE FOR THE
C                       SIXTH LINK.
C
C
C
C
C                       PART  1
C
      VAL(1) = (-ROT(1,1))*(COS(ANG(1,1))*INT(1)*SIN(ANG(4,1))-
     &          SIN(ANG(1,1))*COS(ANG(4,1)))
      VAL(2) = (-ROT(1,1))*(COS(ANG(1,2))*INT(2)*SIN(ANG(4,2))-
     &          SIN(ANG(1,2))*COS(ANG(4,2)))
      VAL(3) = (-ROT(1,1))*(COS(ANG(1,3))*INT(3)*SIN(ANG(4,3))-
     &          SIN(ANG(1,3))*COS(ANG(4,3)))
      VAL(4) = (-ROT(1,1))*(COS(ANG(1,4))*INT(4)*SIN(ANG(4,4))-
     &          SIN(ANG(1,4))*COS(ANG(4,4)))
C
C
C
C
C                       PART  2
C
      VAL(5) = ROT(2,1)*(SIN(ANG(1,1))*INT(5)*SIN(ANG(4,1))-
     &          COS(ANG(1,1))*COS(ANG(4,1)))
      VAL(6) = ROT(2,1)*(SIN(ANG(1,2))*INT(6)*SIN(ANG(4,2))-
     &          COS(ANG(1,2))*COS(ANG(4,2)))
      VAL(7) = ROT(2,1)*(SIN(ANG(1,3))*INT(7)*SIN(ANG(4,3))-
     &          COS(ANG(1,3))*COS(ANG(4,3)))
      VAL(8) = ROT(2,1)*(SIN(ANG(1,4))*INT(8)*SIN(ANG(4,4))-
     &          COS(ANG(1,4))*COS(ANG(4,4)))
C
C
C
C                       PART  3
C
      VAL(9) = ROT(3,1)*INT(1)*SIN(ANG(4,1))
      VAL(10) = ROT(3,1)*INT(2)*SIN(ANG(4,2))
      VAL(11) = ROT(3,1)*INT(3)*SIN(ANG(4,3))
      VAL(12) = ROT(3,1)*INT(4)*SIN(ANG(4,4))
C
C
C
C                       PART  4 (S6)
C
      VAL(13) = VAL(1) - VAL(5) + VAL(9)
      VAL(14) = VAL(2) - VAL(6) + VAL(10)
      VAL(15) = VAL(3) - VAL(7) + VAL(11)
      VAL(16) = VAL(4) - VAL(8) + VAL(12)
C
C
C
C                       PART  5
C
      VAL(17) = (COS(ANG(1,1))*INT(5)*COS(ANG(4,1))+SIN(ANG(1,1))*
     &    SIN(ANG(4,1)))*COS(ANG(5,1))-COS(ANG(1,1))*INT(1)*SIN(ANG(5,1))
      VAL(17) = ROT(1,1)*VAL(17)
C
      VAL(18) = (COS(ANG(1,2))*INT(6)*COS(ANG(4,2))+SIN(ANG(1,2))*
     &  SIN(ANG(4,2)))*COS(ANG(5,2))-COS(ANG(1,2))*INT(2)*SIN(ANG(5,2))
      VAL(18) = ROT(1,1)*VAL(18)
C
      VAL(19) = (COS(ANG(1,3))*INT(7)*COS(ANG(4,3))+SIN(ANG(1,3))*
     &  SIN(ANG(4,3)))*COS(ANG(5,3))-COS(ANG(1,3))*INT(3)*SIN(ANG(5,3))
      VAL(19) = ROT(1,1)*VAL(19)
C
      VAL(20) = (COS(ANG(1,4))*INT(8)*COS(ANG(4,4))+SIN(ANG(1,4))*
     &  SIN(ANG(4,4)))*COS(ANG(5,4))-COS(ANG(1,4))*INT(4)*SIN(ANG(5,4))
      VAL(20) = ROT(1,1)*VAL(20)
C
C
C                       PART  6
C
C
```

```
      VAL(21) = (SIN(ANG(1,1))*INT(5)*COS(ANG(4,1))-COS(ANG(1,1))*
     & SIN(ANG(4,1)))*COS(ANG(5,1))-SIN(ANG(1,1))*INT(1)*SIN(ANG(5,1))
      VAL(21) = ROT(2,1)*VAL(21)
C
      VAL(22) = (SIN(ANG(1,2))*INT(6)*COS(ANG(4,2))-COS(ANG(1,2))*
     & SIN(ANG(4,2)))*COS(ANG(5,2))-SIN(ANG(1,2))*INT(2)*SIN(ANG(5,2))
      VAL(22) = ROT(2,1)*VAL(22)
C
      VAL(23) = (SIN(ANG(1,3))*INT(7)*COS(ANG(4,3))-COS(ANG(1,3))*
     & SIN(ANG(4,3)))*COS(ANG(5,3))-SIN(ANG(1,3))*INT(3)*SIN(ANG(5,3))
      VAL(23) = ROT(2,1)*VAL(23)
C
      VAL(24) = (SIN(ANG(1,4))*INT(8)*COS(ANG(4,4))-COS(ANG(1,4))*
     & SIN(ANG(4,4)))*COS(ANG(5,4))-SIN(ANG(1,4))*INT(4)*SIN(ANG(5,4))
      VAL(24) = ROT(2,1)*VAL(24)
C
C
C                  PART  7
C
C
      VAL(25) = (ROT(3,1))*(INT(1)*COS(ANG(4,1))*COS(ANG(5,1))+
     &           INT(5)*SIN(ANG(5,1)))
      VAL(26) = (ROT(3,1))*(INT(2)*COS(ANG(4,2))*COS(ANG(5,2))+
     &           INT(6)*SIN(ANG(5,2)))
      VAL(27) = (ROT(3,1))*(INT(3)*COS(ANG(4,3))*COS(ANG(5,3))+
     &           INT(7)*SIN(ANG(5,3)))
      VAL(28) = (ROT(3,1))*(INT(4)*COS(ANG(4,4))*COS(ANG(5,4))+
     &           INT(8)*SIN(ANG(5,4)))
C
C
C
C                     PART  8  (C6)
C
C
C
      VAL(29) = VAL(17) + VAL(21) - VAL(25)
      VAL(30) = VAL(18) + VAL(22) - VAL(26)
      VAL(31) = VAL(19) + VAL(23) - VAL(27)
      VAL(32) = VAL(20) + VAL(24) - VAL(28)
C
C                    FINAL JOINT ANGLES
C
      ANG(6,1) = ATAN2 (VAL(13),VAL(29))
      ANG(6,2) = ATAN2 (VAL(14),VAL(30))
      ANG(6,3) = ATAN2 (VAL(15),VAL(31))
      ANG(6,4) = ATAN2 (VAL(16),VAL(32))
C
C
      RETURN
      END
```

```
      SUBROUTINE ARRANGE (A,B)
      DIMENSION A(6,4), B(6,4)
C
C     THIS SUBROUTINE IS TO ARRANGE THE THE SET OF ANGLES (4 SETS)
C     SO THAT EVERY SET WILL BE BASED UPON THIER DEPENDANCY
C
      DO I=1,6
       DO J=1,4
        B(I,J) = A(I,J)
       ENDDO
      ENDDO
C
      B(1,2) = A(1,1)
      B(1,3) = A(1,2)
      B(1,4) = A(1,2)
C
      RETURN
      END



      SUBROUTINE  PATH (ARM,ELBOW,A, B)
      DIMENSION A(6,4,61), B(6,61)
      CHARACTER*6 ARM,ELBOW
C
C     THIS SUPROUTINE IS TO CHOOSE THE RIGHT PATH FOR EACH JOINT
C     GIVEN THE FOUR POSSIBLE SOLUTIONS AT EACH SINGLE INSTANT OF
C     THE TRAJECTORY'S TIME
C
      IF(((ARM(1:1) .EQ. 'L') .OR. (ARM(1:1) .EQ. 'l')) .AND.
     &    ((ELBOW(1:1) .EQ. 'U') .OR. (ELBOW(1:1) .EQ. 'u'))) J_PATH=1
C
      IF(((ARM(1:1) .EQ. 'L') .OR. (ARM(1:1) .EQ. 'l')) .AND.
     &    ((ELBOW(1:1) .EQ. 'D') .OR. (ELBOW(1:1) .EQ. 'd'))) J_PATH=2
C
      IF(((ARM(1:1) .EQ. 'R') .OR. (ARM(1:1) .EQ. 'r')) .AND.
     &    ((ELBOW(1:1) .EQ. 'U') .OR. (ELBOW(1:1) .EQ. 'u'))) J_PATH=3
C
      IF(((ARM(1:1) .EQ. 'R') .OR. (ARM(1:1) .EQ. 'r')) .AND.
     &    ((ELBOW(1:1) .EQ. 'D') .OR. (ELBOW(1:1) .EQ. 'd'))) J_PATH=4
C
      DO  II=1,61
       DO  I=1,6
        B(I,II) = A(I,J_PATH,II)
       ENDDO
      ENDDO
      RETURN
      END
```

```
      PROGRAM CLOSED_FORM
      INCLUDE 'VARIABLES'
C
C     THIS PROGRAM CALCULATES THE REQUIRED TORQUE FOR EACH
C     OF THE FIRST THREE LINKS OF THE PUMA ROBOT ARM TO TRAVEL
C     A DESCRIBED TRAJECTORY.
C
C
      CALL DATA (TWIST,SMALLA,DEE)
      CALL CONDITION (TWIST,SMALLA,DEE,PARAMETER,BALANCE,RATIOS,
     &                                                   DEGREES)
C
      DO 1 III=1,60
         IF (BALANCE(1:1).EQ.'Y' .OR. BALANCE(1:1).EQ.'y') THEN
           CALL CBMASSES (CBM2,CBM3,PAYLD,SMALLA,III)
         ENDIF
         CALL CALCINERTIA (R,IXX2,IYY2,IZZ2,IXX3,IYY3,IZZ3,CBM2,CBM3,
     &                        PAYLD,SMALLA,III,BALANCE)
         CALL MATRIXJ (JI,R,IXX2,IYY2,IZZ2,CBM2,CBM3,PAYLD,IXX3,
     &                  IYY3,IZZ3,III)
         CALL TRAJECTORY (POSDUM,VELDUM,ACCDUM,RATIOS,DEGREES)
C
         DO 2 TIME=1,61
           CALL SUPPLY (POSDUM,VELDUM,ACCDUM,POS,VEL,
     &                  ACCEL,TIME)
           CALL DATAINFO (TWIST,SMALLA,DEE,POS,VEL,ACCEL,
     &                    GRAV,CENT,D,JI,R,CBM2,CBM3,PAYLD,III)
           CALL COMPONENTS (GRAV,CENT,D,ACCEL,INERTF,GRAVF,
     &                      VELF,TIME)
           CALL TORQUELINK (GRAV,CENT,D,TIME,ACCEL,TORQ)
           CALL EIG_INERT (LAMBDA1,D,TIME,III)
2        CONTINUE
1     CONTINUE
C
      CALL INDEX_INER (INDEX1,LAMBDA1,PARAMETER)
C
C      CALL PLOTPOS (POSDUM)
C      CALL PLOTVEL (VELDUM)
C      CALL PLOTACC (ACCDUM)
C      CALL PLOTTORQUE (TORQ,GRAVF,VELF,INERTF)
C
      STOP
      END



      SUBROUTINE CALCINERTIA(R,IXX2,IYY2,IZZ2,IXX3,IYY3,IZZ3,CBM2,CBM3,
     &                       PAYLD,SMALLA,III,BALANCE)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE CALCULATES THE INERTIA OF EACH
C     LINK OF THE ROBOT ARM AROUND THE X,Y AND Z AXIS.
C
      PAYLD=2.5
      IF (BALANCE(1:1).NE.'Y' .OR. BALANCE(1:1).NE.'y') THEN
         CBM2=0.0
         CBM3=0.0
      ENDIF
C
      R(4,1,III)=1.0
      R(1,2,III) = (((0.425+SMALLA(2,III))*(CBM2+17.4))
     &             -(0.425+(0.068/0.432)*SMALLA(2,III))*17.4)
     &             / (CBM2+17.4)
      R(2,2,III)=0.0
      R(3,2,III)=0.0
      R(4,2,III)=1.0
      R(1,3,III)=0.0
      R(2,3,III)=0.0
C
      R(3,3,III) = ((0.143/0.565*SMALLA(3,III) * 6.04)
     &           + (SMALLA(3,III)*PAYLD)
     &           - 0.280*CBM3)/(6.04+PAYLD+CBM3)
C
      R(4,3,III)=1.0
C
C
      IXX2 = 0.130
      IYY2 = 0.524 *(SMALLA(2,III)/0.364)**2
     &       + 17.4 * (R(1,2,III) - (0.364/0.432*SMALLA(2,III)))**2
     &       + CBM2 * (0.425 + SMALLA(2,III) - R(1,2,III))**2
```

```
      IZZ2 = 0.539
     &       +17.4*((R(1,2,III)-((0.354/0.432)*SMALLA(2,III)))**2)
     &       +CBM2*((0.425+SMALLA(2,III)) - R(1,2,III))
C
C
C
      IXX3 = 0.192
     &     + 6.04 *((0.143/0.565*SMALLA(3,III)) - R(3,3,III))**2
     &     + PAYLD * (SMALLA(3,III) - R(3,3,III))**2
     &     + CBM3 * (0.280 + R(3,3,III))**2
C
      IYY3 = 0.212
     &     + 6.04 *((0.143/0.565*SMALLA(3,III)) - R(3,3,III))**2
     &     + PAYLD * (SMALLA(3,III) - R(3,3,III))**2
     &     + CBM3 * (0.280 + R(3,3,III))**2
C
      IZZ3 = 0.0154
C
      R(1,2,III)=-R(1,2,III)
C
      RETURN
      END



      SUBROUTINE MATRIXJ (JI,R,IXX2,IYY2,IZZ2,CBM2,CBM3,PAYLD,IXX3,
     &                    IYY3,IZZ3,III)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE INERTIA
C     TENSOR MATRICES.
C
C
      DO 1 X=1,4
        DO 2 Y=1,4
          DO 3 Z=1,3
            JI(X,Y,Z)=0.0
3         CONTINUE
2       CONTINUE
1     CONTINUE
C
      CALL MATRIXJ1 (JI)
      CALL MATRIXJ2 (JI,R,CBM2,IXX2,IYY2,IZZ2,III)
      CALL MATRIXJ3 (JI,R,PAYLD,CBM3,IXX3,IYY3,IZZ3,III)
C
      RETURN
      END



      SUBROUTINE MATRIXJ1 (JI)
      REAL JI(4,4,3)
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE INERTIA
C     TENSOR MATRIX FOR LINK 1..
C
C
      JI(1,1,1)=0.175
      JI(2,2,1)=0.175
      JI(3,3,1)=-0.175
C
      RETURN
      END



      SUBROUTINE MATRIXJ2 (JI,R,CBM2,IXX2,IYY2,IZZ2,III)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE INERTIA
C     TENSOR MATRIX FOR LINK 2.
C
C
      JI(1,1,2)=(-IXX2+IYY2+IZZ2)/2+(17.4+CBM2)*(R(1,2,III)**2)
      JI(1,4,2)=(17.4+CBM2)*R(1,2,III)
      JI(2,2,2)=(IXX2-IYY2+IZZ2)/2+(17.4+CBM2)*(R(2,2,III)**2)
      JI(2,4,2)=(17.4+CBM2)*R(2,2,III)
      JI(3,3,2)=(IXX2+IYY2-IZZ2)/2+(17.4+CBM2)*(R(3,2,III)**2)
      JI(3,4,2)=(17.4+CBM2)*(R(3,2,III))
      JI(4,1,2)=(17.4+CBM2)*R(1,2,III)
```

```
      JI(4,2,2)=(17.4+CBM2)*R(2,2,III)
      JI(4,3,2)=(17.4+CBM2)*(R(3,2,III))
      JI(4,4,2)=17.4+CBM2
C
      RETURN
      END


      SUBROUTINE MATRIXJ3 (JI,R,PAYLD,CBM3,IXX3,IYY3,IZZ3,III)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE INERTIA
C     TENSOR MATRIX FOR LINK 3.
C
C
      JI(1,1,3)=(-IXX3+IYY3+IZZ3)/2+(6.04+PAYLD+CBM3)*R(1,3,III)**2
      JI(1,4,3)=(6.04+PAYLD+CBM3)*R(1,3,III)
      JI(2,2,3)=(IXX3-IYY3+IZZ3)/2+(6.04+PAYLD+CBM3)*(R(2,3,III)**2)
      JI(2,4,3)=(6.04+CBM3+PAYLD)*R(2,3,III)
      JI(3,3,3)=(IXX3+IYY3-IZZ3)/2+(6.04+PAYLD+CBM3)*(R(3,3,III)**2)
      JI(3,4,3)=(6.04+CBM3+PAYLD)*R(3,3,III)
      JI(4,1,3)=(6.04+PAYLD+CBM3)*R(1,3,III)
      JI(4,2,3)=(6.04+CBM3+PAYLD)*R(2,3,III)
      JI(4,3,3)=(6.04+CBM3+PAYLD)*R(3,3,III)
      JI(4,4,3)=6.04+CBM3+PAYLD
C
      RETURN
      END


      SUBROUTINE SUPPLY (POSDUM,VELDUM,ACCDUM,POS,VEL,
     &                   ACCEL,TIME)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FROM THE
C     TRAJECTORY INPUT TO THE DATAINFO FILE.
C
      DO 1 X=1,3
         POS(X)=0.0
         VEL(X)=0.0
         ACCEL(X)=0.0
         POS(X)=POSDUM(X,TIME)
         VEL(X)=VELDUM(X,TIME)
         ACCEL(X)=ACCDUM(X,TIME)
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE TORQUELINK (GRAV,CENT,D,TIME,ACCEL,TORQ)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE CALCULATES THE TORQUE OF ALL
C     THREE LINKS.
C
      DO 1 K=1,3
         TORQ(K,TIME)= D(K,1)*ACCEL(1)
     &                +D(K,2)*ACCEL(2)
     &                +D(K,3)*ACCEL(3)
     &                +CENT(K)+GRAV(K)
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE COMPONENTS (GRAV,CENT,D,ACCEL,INERTF,
     &                       GRAVF,VELF,TIME)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE INDIVIDUAL TORQUE
C     COMPONENTS FOR EACH LINK TO THE TORQUE PLOT
C     PACKAGE.
C
      DO 1 K=1,3
        GRAVF(K,TIME)=GRAV(K)
        VELF(K,TIME)=CENT(K)
```

```
          INERTF(K,TIME)=D(K,1)*ACCEL(1)+
     &                   D(K,2)*ACCEL(2)+
     &                   D(K,3)*ACCEL(3)
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE TORQDATA (TORQUE,TORQ,LEN)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE PLACES THE TORQUES OF THE ROBOT
C     ARM FOR  FIVE  TRAJECTORIES IN THE ONE ARRAY FOR
C     USE IN THE PLOT PACKAGE.
C
      DO 1 X=1,3
        DO 2 Y=1,61
           TORQUE(X,Y,LEN)=TORQ(X,Y)
2       CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE TRAJECTORY (POSDUM,VELDUM,ACCDUM,RATIOS,DEGREES)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUINE CALCULATES THE TRAJECTORY OF THE
C     LINKS AS A FUNCTION OF TIME.
C
      DO 1 K=1,3
       RATIO = RATIOS(K)
       DEG = DEGREES(K)
       CALL TRAJJOINT (POSDUM,VELDUM,ACCDUM,RATIO,DEG,K)
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE TRAJJOINT (POSDUM,VELDUM,ACCDUM,RATIO,DEG,K)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE CALCULATES THE TRAJECTORY OF A
C     LINK POSITION AS A FUNCTION OF TIME.
C
      POSDUM(K,1)=0.0
      VELDUM(K,1)=0.0
      ACCDUM(K,1)=0.0
      POSDUM(K,61)=1.0
      VELDUM(K,61)=0.0
      ACCDUM(K,61)=0.0
C
      DO 1 I=2,60
         S=I-1
         T=S/60.0
         POSDUM(K,I)=((1.0-T)**3)*(POSDUM(K,1)+(3.0*
     &   POSDUM(K,1)+VELDUM(K,1))*T+(ACCDUM(K,1)+6.0*
     &   VELDUM(K,1)+12.0*POSDUM(K,1))*(T**2/2.0))+
     &   (T**3.0)*(POSDUM(K,61)+(3.0*POSDUM(K,61)-
     &   VELDUM(K,61))*(1.0-T)+(ACCDUM(K,61)-(6.0*
     &   VELDUM(K,61))+12.0*POSDUM(K,61))*((1.0-T)**2/2.0))
1     CONTINUE
C
      DO 2 I=1,61
         POSDUM(K,I)=POSDUM(K,I)*RATIO
         POSDUM(K,I)=POSDUM(K,I)+DEG
2     CONTINUE
C
      VELDUM(K,1)=0.0
      ACCDUM(K,1)=0.0
C
      DO 3 I=2,60
         VELDUM(K,I)=(0.5*(POSDUM(K,I+1)-POSDUM(K,I-1)))/
     &   (1.0/60.0)
```

```
C
          ACCDUM(K,I)=(POSDUM(K,I+1)-2.0*POSDUM(K,I)+
     &    POSDUM(K,I-1))/((1.0/60.0)**2)
3     CONTINUE
C
      VELDUM(K,1)=0.0
      ACCDUM(K,1)=0.0
      VELDUM(K,61)=VELDUM(K,1)
      ACCDUM(K,61)=ACCDUM(K,1)
C
      RETURN
      END


      SUBROUTINE DATAINFO (TWIST,SMALLA,DEE,POS,VEL,ACCEL,
     &                     GRAV,CENT,D,JI,R,CBM2,CBM3,PAYLD,III)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE CALCULATES THE INERTIA,THE INVERSE
C     OF THE INERTIA, THE CENTRIFUGAL AND THE GRAVITY
C     TERMS THAT DESCRIBE THE DYNMAMIC BEHAVIOUR OF A
C     ROBOT ARM.
C
      CALL MATRIXA (A,POS,TWIST,SMALLA,DEE,III)
      CALL MATRIXQ (Q)
      CALL INERTIA (A,Q,JI,D)
      CALL CENTRIFUGAL (CENT,A,JI,Q,VEL)
      CALL GRAVITY (A,Q,GRAV,R,CBM2,CBM3,PAYLD,III)
C
      RETURN
      END


      SUBROUTINE MATRIXA (A,POS,TWIST,SMALLA,DEE,III)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE ASSIGNS DATA TO 4*4 TRANSFOR-
C     MATION MATRICES WHICH DESCRIBE THE RELATIONSHIP
C     BETWEEN THE LINKS OF THE ROBOT ARM.
C
      CALL INITIALIZEA (A)
      CALL MATA (A,TWIST,SMALLA,DEE,POS,III)
      CALL MATAkk (A)
      CALL MATA20 (A)
      CALL MATA31 (A)
      CALL MATA30 (A)
C
      RETURN
      END


      SUBROUTINE INITIALIZEA (A)
      REAL A(4,4,3,4)
      INTEGER I,J,K1,K2
C
C     THIS SUBROUTINE SETS ALL THE 4*4 TRANSFORMATION
C     MATRICES TO ZERO.
C
      DO 1 I=1,4
        DO 2 J=1,4
          DO 3 K1=1,3
            DO 4 K2=1,4
              A(I,J,K1,K2)=0.0
4           CONTINUE
3         CONTINUE
2       CONTINUE
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE MATAkk (A)
      INCLUDE 'VARIABLES'
C
C     This SUBROUTINE SETS THE TRANSFORMATION MATRICES A00,
```

```
C     A11,A22 TO IDENTY MATRICES.
C
      DO 1 K=1,3
         DO 2 I=1,4
            A(I,I,K,K) = 1.0
2        CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE MATA (A,TWIST,SMALLA,DEE,POS,III)
      REAL A(4,4,3,4),TWIST(3,60),SMALLA(3,60),DEE(3,60),POS(3)
      INTEGER III
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE TRANS-
C     FORMATION MATRICES WHICH RELATE EACH LINK's COORDINATE
C     FRAME TO THE PREVIOUS LINK'S COORDINATE FRAME.
C             IT DETERMINES A10, A21, A32
C
      DO 1 K=1,3
         A(1,1,K,(K+1)) =  COS (POS(K))
         A(1,2,K,(K+1)) = -SIN(POS(K)) * COS (TWIST(K,III))
         A(1,3,K,(K+1)) =  SIN(POS(K)) * SIN (TWIST(K,III))
         A(1,4,K,(K+1)) =  SMALLA(K,III) * COS (POS(K))
         A(2,1,K,(K+1)) =  SIN (POS(K))
         A(2,2,K,(K+1)) =  COS (POS(K)) * COS (TWIST(K,III))
         A(2,3,K,(K+1)) = -COS (POS(K)) * SIN (TWIST(K,III))
         A(2,4,K,(K+1)) =  SMALLA(K,III) * SIN (POS(K))
         A(3,1,K,(K+1)) =  0.0
         A(3,2,K,(K+1)) =  SIN(TWIST(K,III))
         A(3,3,K,(K+1)) =  COS(TWIST(K,III))
         A(3,4,K,(K+1)) =  DEE(K,III)
         A(4,4,K,(K+1)) =  1.0
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE MATA20 (A)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE TRANS-
C     FORMATION MATRIX WHICH RELATES LINK 2 TO THE
C     ROBOT'S BASE.
C
      DO 1 I = 1,4
         DO 2 J=1,4
            A(I,J,1,3) = 0.0
            DO 3 M=1,4
               A(I,J,1,3)=A(I,J,1,3)+A(I,M,1,2)*A(M,J,2,3)
3           CONTINUE
2        CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE MATA31 (A)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE TRANS-
C     FORMATION MATRIX WHICH RELATES LINK 3 TO LINK 1..
C
      DO 1 I = 1,4
         DO 2 J=1,4
            A(I,J,2,4) = 0.0
            DO 3 M=1,4
               A(I,J,2,4)=A(I,J,2,4)+A(I,M,2,3)*A(M,J,3,4)
3           CONTINUE
2        CONTINUE
1     CONTINUE
C
      RETURN
```

```
      END


      SUBROUTINE MATA30 (A)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE TRANS-
C     FORMATION MATRIX WHICH RELATES LINK 3 TO THE
C     ROBOT BASE..
C
      DO 1 I = 1,4
         DO 2 J=1,4
            A(I,J,1,4) = 0.0
            DO 3 M=1,4
               A(I,J,1,4)=A(I,J,1,4)+A(I,M,1,2)*A(M,J,2,4)
3           CONTINUE
2        CONTINUE
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE MATRIXQ (Q)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE SUPPLIES THE DATA FOR THE ROTARY
C     JOINT MATRIX OPERATOR.
C
      DO 1 X=1,4
        DO 2 Y=1,4
          Q(X,Y)=0.0
2       CONTINUE
1     CONTINUE
C
      Q(1,2)=-1.0
      Q(2,1)=1.0
C
      RETURN
      END


      SUBROUTINE INERTIA (A,Q,JI,D)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE  CALCULATES THE 3*3 INERTIA OR
C     ACCELERATION RELATED MATRIX FOR A 3 LINK ROBOT ARM.
C
C                          SET INERTIA MATRIX TO ZERO.
      DO 1 X=1,3
        DO 2 Y=1,3
          D(X,Y)=0.0
2       CONTINUE
1     CONTINUE
C
      DO 3 i=1,3
        DO 4 j=1,3
          m=0
          IF (i.GT.j) THEN
             m=m+i
             D(i,j)=D(i,j)+D(j,i)
          ELSE
            m=m+j
            CALL DMATRIX (D,A,Q,JI,i,j,m)
          ENDIF
4       CONTINUE
3     CONTINUE
C
      RETURN
      END


      SUBROUTINE DMATRIX (D,A,Q,JI,i,j,m)
      INCLUDE 'VARIABLES'
C
C     THIS  SUBROUTINE CALCULATES EACH INDIVIDUAL TERM
```

```
C     OF THE INERTIA MATRIX.
C
      p=0
      n=0
      DO 1 p=m,3
         n=j
         CALL AQMATRIX (AQ,A,Q,n)
         CALL UMATRIX (U,AQ,A,n,p)
         CALL TERM1 (T1,U,JI,p)
         n=i
         CALL AQMATRIX (AQ,A,Q,n)
         CALL UMATRIX (U,AQ,A,n,p)
         CALL TRANSPOSE (UT,U)
         CALL TERM2 (T2,T1,UT)
         D(i,j)=D(i,j)+T2(1,1)+T2(2,2)+T2(3,3)+T2(4,4)
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE AQMATRIX (AQ,A,Q,n)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES A SPECIFIED TRANSFORM-
C     ATION MATRIX BY THE OPERATING MATRIX.
C
C     [AQ] IS A 4*4 MATRIX WHICH IS THE PRODUCT OF THE
C     TRANSFORMATION AND OPERATING MATRICES.
C
C                         SET AQ MATRIX TO ZERO.
      DO 1 X=1,4
        DO 2 Y=1,4
          AQ(X,Y)=0.0
            DO 3 Z=1,4
               AQ(X,Y)=AQ(X,Y)+A(X,Z,1,n)*Q(Z,Y)
3           CONTINUE
2       CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE UMATRIX (U,AQ,A,n,p)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES THE AQ MATRIX BY A
C     SPECIFIED TRANSFORMATION MATRIX AGAIN.
C
C     [U] IS A 4*4 MATRIX WHICH IS THE PRODUCT OF THE
C     AQ AND TRANSFORMATION MATRICES.
C
      DO 1 X=1,4
        DO 2 Y=1,4
C                          SET U MATRIX TO ZERO
          U(X,Y)=0.0
            DO 3 Z=1,4
              U(X,Y)=U(X,Y)+AQ(X,Z)*A(Z,Y,n,(p+1))
3           CONTINUE
2       CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE TERM1 (T1,U,JI,p)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES THE U MATRIX BY THE
C     SPECIFIED INERTIA TENSOR MATRIX.
C
C     [T1] IS A 4*4 MATRIX WHICH IS THE PRODUCT OF THE
C     U AND INERTIA TENSOR MATRICES.
C
      DO  1 X=1,4
```

```
          DO 2 Y=1,4
            T1(X,Y)=0.0
              DO 3 Z=1,4
                T1(X,Y)=T1(X,Y)+U(X,Z)*JI(Z,Y,p)
3             CONTINUE
2         CONTINUE
1       CONTINUE
C
        RETURN
        END



        SUBROUTINE TRANSPOSE (UT,U)
        INCLUDE 'VARIABLES'
C
C       THIS SUBROUTINE FINDS THE TRANSPOSE OF THE 'U'
C       MATRIX .
C
        DO 1 X=1,4
          DO 2 Y=1,4
C                           SET TRANSPOSE MATRIX TO ZERO.
            UT(X,Y)=0.0
            UT(X,Y)=UT(X,Y)+U(Y,X)
2         CONTINUE
1       CONTINUE
C
        RETURN
        END



        SUBROUTINE TERM2 (T2,T1,UT)
        INCLUDE 'VARIABLES'
C
C       THIS SUBROUTINE MULTIPLIES THE MATRIX T1 DETERMINED
C       FROM SUBROUTINE TERM1 BY MATRIX UT DETERMINED
C       FROM SUBROUTINE TRANSPOSE.
C
C       [M] IS A 4*4 MATRIX AND IS A PRODUCT OF THE 'T1'
C       AND 'UT' MATRICES.
C
        DO 1 X=1,4
          DO 2 Y=1,4
C                           SET T2 MATRIX TO ZERO.
            T2(X,Y)=0.0
              DO 3 Z=1,4
                T2(X,Y)=T2(X,Y)+T1(X,Z)*UT(Z,Y)
3             CONTINUE
2         CONTINUE
1       CONTINUE
C
        RETURN
        END




        SUBROUTINE CENTRIFUGAL (CENT,A,JI,Q,VEL)
        INCLUDE 'VARIABLES'
C
C       THIS SUBROUTINE CALCULATES THE VELOCITY RELATED
C       CORIOLIS AND CENTRIFUGAL OUTPUT TERMS. THE OUTPUT
C       IS IN THE FORM OF A 1*3 COLLUMN VECTOR.
C
        DO 1 i=1,3
          DO 2 j=1,3
            DO 3 k=1,3
C
              H(i,j,k)=0.0
              IF (j.GT.k) THEN
                 H(i,j,k)=H(i,j,k)+H(i,k,j)
              ELSE
                 CALL HMATRIX (i,j,k,H,A,Q,JI)
              ENDIF
C
3           CONTINUE
2         CONTINUE
C
```

```
        CALL MULTVELOCITY (H,VEL,CENT,i)
C
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE HMATRIX (i,j,k,H,A,Q,JI)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE CALCULATES EACH INDIVIDUAL TERM OF
C     THE CENTRIFUGAL ACCELERATION MATRIX.
C
       CALL MAX1 (i,j,k,m)
      p=0
      n=0
      DO 1 p=m,3
C
        n=j
        CALL AQMATRIX (AQ,A,Q,n)
        CALL AQAMATRIX (AQ,A,AQA,j,k)
        CALL AQAQMATRIX (AQ,AQA,Q)
        n=k
        CALL UMATRIX (U,AQ,A,n,p)
        CALL TERM1 (T1,U,JI,p)
        n=i
        CALL AQMATRIX (AQ,A,Q,n)
        CALL UMATRIX (U,AQ,A,n,p)
        CALL TRANSPOSE (UT,U)
        CALL TERM2 (T2,T1,UT)
C
        H(i,j,k)=H(i,j,k)+T2(1,1)+T2(2,2)+T2(3,3)+T2(4,4)
C
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE MAX1 (i,j,k,m)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE CALCULATES THE MAXIMUM OF THE
C     COUNTERS i,j,k.
c
      m=0
      IF (i.GE.j.AND.i.GE.k) THEN
         m=m+i
      ELSE IF (j.GE.i.AND.j.GE.k) THEN
         m=m+j
      ELSE
         m=m+k
      ENDIF
C
      RETURN
      END


      SUBROUTINE AQAMATRIX (AQ,A,AQA,j,k)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES THE MATRIX AQ DETERNINED
C     IN SUBROUTINE AQ MATRIX BY THE SPECIFIED TRANSFORM-
C     ATION MATRIX.
C
C     [AQA] IS A 4*4 MATRIX
C
      DO 1 X=1,4
        DO 2 Y=1,4
C                       SET AQA MATRIX TO ZERO.
          AQA(X,Y)=0.0
            DO 3 Z=1,4
C
              AQA(X,Y)=AQA(X,Y)+AQ(X,Z)*A(Z,Y,j,k)
C
3           CONTINUE
```

```
2       CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE AQAQMATRIX (AQ,AQA,Q)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLES THE AQA MATRIX DETERMINED
C     IN THE SUBROUTINE AQAMATRIX BY THE OPERATING MATRIX.
C
      DO 1 X=1,4
        DO 2 Y=1,4
          AQ(X,Y)=0.0
            DO 3 Z=1,4
C
              AQ(X,Y)=AQ(X,Y)+AQA(X,Z)*Q(Z,Y)
C
3           CONTINUE
2       CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE MULTVELOCITY (H,VEL,CENT,i)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES THE CENTRIFUGAL MATRIX
C     BY THE ACCELERATION OF ALL LINKS TO GIVE THE
C     CENTRIFUGAL FORCE AT EACH LINK.
C
      DO 1 X=1,3
        CI(X)=0.0
        CI(X)=VEL(1)*H(i,X,1)+VEL(2)*H(i,X,2)+VEL(3)*H(i,X,3)
1     CONTINUE
C
      CENT(i)=0.0
      CENT(i)=CI(1)*VEL(1)+CI(2)*VEL(2)+CI(3)*VEL(3)
C
      RETURN
      END



      SUBROUTINE GRAVITY (A,Q,GRAV,R,CBM2,CBM3,PAYLD,III)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE DETERMINES THE GRAVITY LOADINGS
C     ON EACH LINK OF THE ROBOT ARM AND THE OUTPUT IS
C     IN THE FORM OF A 1*3 COLLUMN VECTOR.
C
      CALL GRAVZERO (GRAV,MASS,CBM2,CBM3,PAYLD,ACC)
C
      n=0
      DO 1 i=1,3
        GRAV(i)=0.0
        DO 2 p=i,3
C
          CALL MGMATRIX (MG,MASS,ACC,p)
          n=i
          CALL AQMATRIX (AQ,A,Q,n)
          CALL UMATRIX (U,AQ,A,n,p)
          CALL MGUMATRIX (MGU,MG,U)
          CALL MGURMATRIX (MGUR,MGU,R,p,III)
C
          GRAV(i)=GRAV(i)+MGUR
C
2       CONTINUE
1     CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE GRAVZERO (GRAV,MASS,CBM2,CBM3,PAYLD,ACC)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE INITIALIZES THE LINK MASSES AND
C     THE ACCELERATION DUE TO GRAVITY.
C
      MASS(1)=0.0
      MASS(2)=-(17.4+CBM2)
      MASS(3)=-(6.04+CBM3+PAYLD)
C
      ACC(3)=-9.81
C
      RETURN
      END



      SUBROUTINE MGMATRIX (MG,MASS,ACC,p)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES THE MATRIX MASS BY
C     THE ACC MATRIX
C
C     [MG] IS A 4*1 ROW VECTOR
C
      DO 1 X=1,4
        MG(X)=0.0
        MG(X)=MG(X)+MASS(p)*ACC(X)
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE MGUMATRIX (MGU,MG,U)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES THE MG MATRIX DETERM-
C     INED FROM SUBROUTINE MGMATRIX BY THE U MATRIX.
C
C     [MGU] IS A 4*1 ROW VECTOR
C
      DO 1 X=1,4
        MGU(X)=0.0
          DO 2 Y=1,4
C
            MGU(X)=MGU(X)+MG(Y)*U(Y,X)
2         CONTINUE
1     CONTINUE
C
      RETURN
      END



      SUBROUTINE MGURMATRIX (MGUR,MGU,R,p,III)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE MULTIPLIES THE MGU MATRIX DETER-
C     MINED FROM SUBROUTINE MGUMATRIX BY THE R OF
C     GYRATION OF THE SPECIFIED LINK.
C
C     [MGUR] IS THE GRAVITY LOADING FELT AT A LINK
C     DUE TO THE POSITION OF ANOTHER LINK.
C
      MGUR=0.0
      DO 1 X=1,4
        MGUR=MGUR+MGU(X)*R(X,p,III)
1     CONTINUE
C
      RETURN
      END
```

```
C*****************************************************************


      SUBROUTINE DATA (TWIST,SMALLA,DEE)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE ROBOT'S INITIAL GEOMETRICAL
C     PARAMETERS. IT ALSO INITIALIZE THE PARAMETERS ARRAYS.
C
      TWIST(1,1) = -90.0 * 3.14/180.0
      TWIST(2,1) = 0.0
      TWIST(3,1) = 90.0 * 3.14/180.0
      SMALLA(1,1)= 0.0
      SMALLA(2,1)= 0.432
      SMALLA(3,1)= 0.0
      DEE(1,1)   = 0.0
      DEE(2,1)   = 0.1495
      DEE(3,1)   = 0.0
C
      DO 1 III=2,60
        TWIST(1,III) = TWIST(1,1)
        TWIST(2,III) = TWIST(2,1)
        TWIST(3,III) = TWIST(3,1)
        SMALLA(1,III)= SMALLA(1,1)
        SMALLA(2,III)= SMALLA(2,1)
        SMALLA(3,III)= SMALLA(3,1)
        DEE(1,III)   = DEE(1,1)
        DEE(2,III)   = DEE(2,1)
        DEE(3,III)   = DEE(3,1)
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE CONDITION(TWIST,SMALLA,DEE,PARAMETER,BALANCE,RATIOS,
     &                                                      DEGREES)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO DETERMINE WHICH IS THE VARIABLE
C     PARAMETER AND THE DEGREE OF VARIATION. IT ALSO FINDS
C     THE TOTAL AND INITIAL DISPLACEMENT FOR EACH LINK.
C
      PRINT *,'WHICH IS THE VARYING PARAMETER ?   '
      PRINT *,'TYPE EITHER  : ALPHA1 or ALPHA2 or ALPHA3'
      PRINT *,'             or   L1  or    L2  or    L3'
      PRINT *,'             or   D1  or    D2  or    D3'
      PRINT *,'             or        NONE  for no change'
      READ *, VAR
C
      PRINT *,'IS IT DYNAMICALLY BALANCED    ?(Y/N)'
      READ *, BALANCE
C
      IF (VAR(1:6).EQ.'ALPHA1' .OR. VAR(1:6).EQ.'alpha1') THEN
        CALL PRAM_ALPHA1 (TWIST,PARAMETER)
      ELSEIF (VAR(1:6).EQ.'ALPHA2' .OR. VAR(1:6).EQ.'alpha2') THEN
        CALL PRAM_ALPHA2 (TWIST,PARAMETER)
      ELSEIF (VAR(1:6).EQ.'ALPHA3' .OR. VAR(1:6).EQ.'alpha3') THEN
        CALL PRAM_ALPHA3 (TWIST,PARAMETER)
      ELSEIF (VAR(1:2).EQ.'L1' .OR. VAR(1:6).EQ.'l1') THEN
        CALL PRAM_L1 (SMALLA,PARAMETER)
      ELSEIF (VAR(1:2).EQ.'L2' .OR. VAR(1:6).EQ.'l2') THEN
        CALL PRAM_L2 (SMALLA,PARAMETER)
      ELSEIF (VAR(1:2).EQ.'L3' .OR. VAR(1:6).EQ.'l3') THEN
        CALL PRAM_L3 (SMALLA,PARAMETER)
      ELSEIF (VAR(1:2).EQ.'D1' .OR. VAR(1:6).EQ.'d1') THEN
        CALL PRAM_D1 (DEE,PARAMETER)
      ELSEIF (VAR(1:2).EQ.'D2' .OR. VAR(1:6).EQ.'d2') THEN
        CALL PRAM_D2 (DEE,PARAMETER)
      ELSEIF (VAR(1:2).EQ.'D3' .OR. VAR(1:6).EQ.'d3') THEN
        CALL PRAM_D3 (DEE,PARAMETER)
      ELSEIF (VAR(1:4).EQ.'NONE' .OR. VAR(1:6).EQ.'none') THEN
        RETURN
      ENDIF
C
      DO 1 K=1,3
       PRINT *,'Total displacement of link no. ',K,' (in degrees)?  '
       READ *,DUMRATIO
```

```
        RATIOS(K) = DUMRATIO
        RATIOS(K) = RATIOS(K) /(180.0/3.14)
        PRINT *,'Starting position of link no. ',K,' (in degrees)?  '
        READ *,DUMDEG
        DEGREES(K) = DUMDEG
        DEGREES(K) = DEGREES(K) * (3.14/180.)
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE PRAM_ALPHA1 (TWIST,PARAMETER)
      REAL TWIST(3,60),PARAMETER(60)
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     TWIST ANGLE (ALPHA1) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF ALPHA1 (in degrees)    ?'
      READ *, TWIST(1,1)
      TWIST(1,1) = TWIST(1,1)*(3.14/180.0)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF ALPHA1 (in degrees)    ?'
      READ *, TWIST(1,60)
      TWIST(1,60) = TWIST(1,60)*(3.14/180.0)
      PRINT*,' '
C
      STEP = (TWIST(1,60)-TWIST(1,1))/60.0
      DO 1 III=2,59
         TWIST(1,III) = TWIST(1,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = TWIST(1,III)
2     CONTINUE
C
      RETURN
      END


      SUBROUTINE PRAM_ALPHA2 (TWIST,PARAMETER)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     TWIST ANGLE (ALPHA2) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF ALPHA2 (in degrees)    ?'
      READ *, TWIST(2,1)
      TWIST(2,1) = TWIST(2,1)*(3.14/180.0)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF ALPHA2 (in degrees)    ?'
      READ *, TWIST(2,60)
      TWIST(2,60) = TWIST(2,60)*(3.14/180.0)
      PRINT*,' '
C
      STEP = (TWIST(2,60)-TWIST(2,1))/60.0
      DO 1 III=2,59
         TWIST(2,III) = TWIST(2,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = TWIST(2,III)
2     CONTINUE
C
      RETURN
      END


      SUBROUTINE PRAM_ALPHA3 (TWIST,PARAMETER)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     TWIST ANGLE (ALPHA3) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
```

```
      PRINT *,' INPUT THE INITIAL VALUE OF ALPHA2 (in degrees)    ?'
      READ *, TWIST(3,1)
      TWIST(3,1) = TWIST(3,1)*(3.14/180.0)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF ALPHA2 (in degrees)    ?'
      READ *, TWIST(3,60)
      TWIST(3,60) = TWIST(3,60)*(3.14/180.0)
      PRINT*,' '
C
      STEP = (TWIST(3,60)-TWIST(3,1))/60.0
      DO 1 III=2,59
         TWIST(3,III) = TWIST(3,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = TWIST(3,III)
2     CONTINUE
C
      RETURN
      END



      SUBROUTINE PRAM_L1 (SMALLA,PARAMETER)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     LINK'S LENGTH (L1) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF L1    ?'
      READ *, SMALLA(1,1)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF L1    ?'
      READ *, SMALLA(1,60)
      PRINT*,' '
C
      STEP = (SMALLA(1,60)-SMALLA(1,1))/60.0
      DO 1 III=2,59
         SMALLA(1,III) = SMALLA(1,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = SMALLA(1,III)
2     CONTINUE
C
      RETURN
      END



      SUBROUTINE PRAM_L2 (SMALLA,PARAMETER)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     LINK'S LENGTH (L2) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF L2    ?'
      READ *, SMALLA(2,1)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF L2    ?'
      READ *, SMALLA(2,60)
      PRINT*,' '
C
      STEP = (SMALLA(2,60)-SMALLA(2,1))/60.0
      DO 1 III=2,59
         SMALLA(2,III) = SMALLA(2,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = SMALLA(2,III)
2     CONTINUE
C
      RETURN
      END



      SUBROUTINE PRAM_L3 (SMALLA,PARAMETER)
```

```
      SUBROUTINE PRAM_D3 (DEE,PARAMETER)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO READ THE TERMINALS VALUES OF THE
C     JOINT'S OFFSET (D3) ARRAY AND CALCULATE THE VALUES OF
C     THE OTHER ELEMENTS.
C
      PRINT *,' INPUT THE INITIAL VALUE OF D3    ?'
      READ *, DEE(3,1)
      PRINT*,' '
      PRINT *,' INPUT THE FINAL VALUE OF D3    ?'
      READ *, DEE(3,60)
      PRINT*,' '
C
      STEP = (DEE(3,60)-DEE(3,1))/60.0
      DO 1 III=2,59
         DEE(3,III) = DEE(3,(III-1)) + STEP
1     CONTINUE
C
      DO 2 III=1,60
         PARAMETER(III) = DEE(3,III)
2     CONTINUE
C
      RETURN
      END



      SUBROUTINE CBMASSES (CBM2,CBM3,PAYLD,SMALLA,III)
      INCLUDE 'VARIABLES'
C
      PAYLD=2.5
C
      CBM3=((6.04*(0.143/0.565)*SMALLA(3,III))+(2.5*SMALLA(3,III)))
      TOT_M3 = 6.04 + 2.5 + CBM3
C
      CBM2 = ((((0.068/0.432)*SMALLA(2,III))*17.4) +
     &       (SMALLA(2,III)*TOT_M3))/0.425
C
      RETURN
      END



      SUBROUTINE EIG_INERT (LAMBDA1,D,TIME,III)
C
C     THIS SUBROUTINE IS TO CALCULATE THE EIGEN VALUES OF THE
C     [D] MATRIX FOR EVERY TIME ITERATION WITHIN THE PARAMETER
C     ITERATIONS.
C
      REAL D(3,3),LAMBDA1(3,61,60)
      DOUBLE PRECISION AA(6,6),Z(6,6),WR(6),WI(6),FV1(6)
      INTEGER   TIME,III,I,J,N,MATZ,IV1(6),ierr
      MATZ = 0
      NM = 6
      N = 3
      DO 1 I=1,3
         DO 2 J=1,3
            AA(I,J) = D(I,J)
2        CONTINUE
1     CONTINUE
c
c *** compute eigenvalues and eigenvectors
c
      CALL RG (NM,N,AA,WR,WI,MATZ,Z,IV1,FV1,ierr)
      if (ierr .ne. 0) write (6,6003) ierr
 6003 format (1h0/1h0,11h  rg ierr =,i5)
C
      DO 3 K=1,3
        LAMBDA1(K,TIME,III) = WR(K)
3     CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE INDEX_INER (INDEX1,LAMBDA1,PARAMETER)
      INCLUDE 'VARIABLES'
      REAL IND_COPY1(61,60),IND_COPY2(61,60),IND_COPY3(61,60)
C
C
      CALL DIFF_INER (LAMBDA1,PARAMETER,DIFF1)
C
      DO 1 TIME=1,61
         DO 2 III=1,60
           INDEX1(TIME,III) = SQRT( (DIFF1(1,TIME,III))**2
     &                             +(DIFF1(2,TIME,III))**2
     &                             +(DIFF1(3,TIME,III))**2)
C     print *,'time =',TIME,'parameter=',PARAMETER(III),
C     &        'index=',INDEX1(TIME,III)
2        CONTINUE
1     CONTINUE
      DO 3 I=1,61
        DO 4 J=1,60
          IND_COPY1(I,J) = INDEX1(I,J)
          IND_COPY2(I,J) = INDEX1(I,J)
          IND_COPY3(I,J) = INDEX1(I,J)
4       CONTINUE
3     CONTINUE
      CALL PPBGN('PLOT_INDEX')
      CALL PLOT_INDEX_PARAM (INDEX1, PARAMETER)
      CALL LOG_INDEX (INDEX1)
      CALL PLOT_INDEX_PARAM (INDEX1, PARAMETER)
C
      CALL SMOX_INDEX (IND_COPY1)
      CALL PLOT_INDEX_PARAM (IND_COPY1, PARAMETER)
      CALL LOG_INDEX (IND_COPY1)
      CALL PLOT_INDEX_PARAM (IND_COPY1, PARAMETER)
C
      CALL SMOXY_INDEX (IND_COPY2,PARAMETER)
      CALL PLOT_INDEX_PARAM (IND_COPY2, PARAMETER)
      CALL LOG_INDEX (IND_COPY2)
      CALL PLOT_INDEX_PARAM (IND_COPY2, PARAMETER)
      RETURN
      END



      SUBROUTINE DIFF_INER (LAMBDA1,PARAMETER,DIFF1)
      INCLUDE 'VARIABLES'
C
C     THIS SUBROUTINE IS TO DIFFERENTIATE THE EIGEN VALUE (LAMBDA1)
C     WITH RESPECT TO THE VARYING GEOMETRICAL PARAMETER.
C
      DO 1 L=1,61
       DO 2 K=1,3
        DO 3 III=2,59
         DIFF1(K,L,III)=(0.5*(LAMBDA1(K,L,III+1)-LAMBDA1(K,L,III-1)))/
     &                  (PARAMETER(III)-PARAMETER(III-1))
3       CONTINUE
2      CONTINUE
1     CONTINUE
C
      DO 4 L=1,61
        DO 5 K=1,3
          DIFF1(K,L,2)=DIFF1(K,L,1)
          DIFF1(K,L,60)=DIFF1(K,L,59)
5       CONTINUE
4     CONTINUE
C
      RETURN
      END



      SUBROUTINE SMOX_INDEX (INDEX)
      REAL INDEX(61,60),DUMMY(61),SMDUMMY(61),X(61),DF(61)
      REAL B(61),C(61),D(61)
      DO 1 J=1,60
         DO 2 I=1,61
            DUMMY(I) = INDEX(I,J)
2        CONTINUE
c
         X(1) = 0.0
         DO 3 I = 2,61
```

```
            SS = I - 1
            X(I) = SS/60.
3         CONTINUE
          STEP = 1.0/60.0
          NX   = 61
          SM   = 0.00001
          DO 4 I = 1,61
             DF(I) = 1.0
4         CONTINUE
          CALL DCSSMO (STEP,NX,X,DUMMY,DF,SM,SMDUMMY,B,C,D)
C
          DO 5 I=1,61
            INDEX(I,J) = SMDUMMY(I)
5         CONTINUE
1       CONTINUE
C
        RETURN
        END


        SUBROUTINE SMOXY_INDEX (INDEX, PARAMETER)
        REAL INDEX(61,60),DUMMY(61),SMDUMMY(61),X(61),DF(61),Y(60)
        REAL B(61),C(61),D(61),PARAMETER(60)
        DO 1 J=1,60
           DO 2 I=1,61
              DUMMY(I) = INDEX(I,J)
2          CONTINUE
C
           X(1) = 0.0
           DO 3 I = 2,61
             SS = I - 1
             X(I) = SS/60.
3          CONTINUE
           STEP = 1.0/60.0
           NX   = 61
           SM   = 0.01
           DO 4 I = 1,61
              DF(I) = 1.0
4          CONTINUE
           CALL DCSSMO (STEP,NX,X,DUMMY,DF,SM,SMDUMMY,B,C,D)
C
           DO 5 I=1,61
             INDEX(I,J) = SMDUMMY(I)
5          CONTINUE
1       CONTINUE
C
C   SMOOTHING ALONG Y-AXIS
C
        DO 6 I=1,61
           DO 7 J=1,60
              DUMMY(J) = INDEX(I,J)
7          CONTINUE
C
           DO 8 J = 1,60
             Y(J) = PARAMETER(J)
8          CONTINUE
           STEP = (PARAMETER(60)-PARAMETER(1))/60.
           NX   = 60
           SM   = 0.01
           DO 9 J = 1,60
              DF(J) = 1.0
9          CONTINUE
           CALL DCSSMO (STEP,NX,Y,DUMMY,DF,SM,SMDUMMY,B,C,D)
C
           DO 10 J=1,60
             INDEX(I,J) = SMDUMMY(J)
10         CONTINUE
6       CONTINUE
        RETURN
        END


        SUBROUTINE LOG_INDEX (INDEX)
        REAL INDEX(61,60)
        DO 1 I=1,61
           DO 2 J=1,60
              INDEX(I,J) = INDEX(I,J) * 100.0
              IF (INDEX(I,J) .LT. 1.0) INDEX(I,J)=1.0
C             IF (INDEX(I,J) .GT. 100.0) INDEX(I,J)=100.0
              INDEX(I,J) = LOG10 (INDEX(I,J))
```

```
2        CONTINUE
1     CONTINUE
C
      RETURN
      END


      SUBROUTINE PLOT_INDEX_PARAM (INDEX, PARAMETER)
      REAL INDEX(61,60), PARAMETER(60), INDEXMAX, INDEXMIN
C
C     THIS SUBROUTINE IS TO PLOT THE EUCLEDIAN NORM OF THE
C     EIGEN VALUES SENSITIVITY AGAINST BOTH ALPHA2 AND THETA.
C
      INDEXMAX = INDEX(1,1)
      INDEXMIN = INDEX(1,1)
ccc   print *,'INDEX            TIME             PARAM_INDEX'
      DO 1 I= 1,61
        DO 2 J= 1,60
ccc        print *,INDEX(I,J),'        ',I,'         ',J
           IF (INDEX(I,J) .GT. INDEXMAX) INDEXMAX=INDEX(I,J)
           IF (INDEX(I,J) .LT. INDEXMIN) INDEXMIN=INDEX(I,J)
2       CONTINUE
1     CONTINUE
ccc   print *,'INDEXMIN = ',INDEXMIN,'        INDEXMAX =',INDEXMAX
C
      XCAMERA = -20.0
      YCAMERA = -15.0 * PARAMETER(60)
      ZCAMERA = 7.0 * INDEXMAX
      XAIM = 0.5
      YAIM = PARAMETER(23)
      ZAIM = 0.0
C
      CALL PPALL ('DISPLAY',100.,100.,900.,900.)
      CALL PJSET (0.0,(INDEXMAX-INDEXMIN),0.0,(INDEXMAX-INDEXMIN),
     &                                        INDEXMIN,INDEXMAX)
      CALL PJSCAL (2)
      CALL PJHIDE
      CALL PJCAMP (XCAMERA ,YCAMERA ,ZCAMERA )
      CALL PJAIMP (XAIM ,YAIM ,ZAIM )
      CALL PJDRAW (INDEX,61,60,61,0,0)
      CALL PPNEXT
      CALL PPALL ('DISPLAY',100.,100.,900.,900.)
      CALL PPALL ('USER',0.,PARAMETER(1),1.0,PARAMETER(60))
      CALL PPCONT(INDEX,61,60,61,INDEXMIN,INDEXMAX,-5,'CONTOUR',7,' ',0)
      CALL PPNEXT
      RETURN
      END
```

```
      SUBROUTINE XAXIS (A,XSTRT,YSTRT,END,ISIDE,MAJ,MID,MIN)
      CHARACTER*1 A
      DOUBLE PRECISION SPSMIN,DXSTRT,DYSTRT,DEND,RMAJ,RMID,RMIN
C
C     SUBROUTINE TO DRAW X-AXIS
C
      WRITE (6,111) XSTRT,YSTRT
111   FORMAT (' PU;PA',F20.6,F20.6,';')
      WRITE (6,222) END,YSTRT,XSTRT,YSTRT
222   FORMAT (' PD',F20.6,F18.6,';PU',F15.6,F18.6,';')
      DXSTRT = XSTRT
      DYSTRT = YSTRT
      DEND   = END
      RMAJ=MAJ
      RMID=MID
      RMIN=MIN
      SPCMIN = (END-XSTRT)/((RMAJ-1.)*(RMID+1.)
     &  *(RMIN+1.))
      IF (MAJ .EQ. 0) SPCMIN = (END-XSTRT)/((RMID-1.)
     &  *(RMIN+1.))
      DO 7 I= 1,MAJ-1
         IF (MAJ .EQ. 0) GO TO 10
         IF (I .NE. 1) GO TO 10
         IF (ISIDE .NE. 1) GO TO 1
         WRITE (6,333) XSTRT,YSTRT
333      FORMAT (' TL1.5;PA',F20.6,',',F20.6,';XT;')
         GO TO 10
1        IF (ISIDE .NE. 0) GO TO 22
         WRITE (6,444) XSTRT,YSTRT
444      FORMAT (' TL1.5,1.5;PA',F20.6,',',F20.6,';XT;')
         GO TO 10
22       WRITE (6,4444) XSTRT, YSTRT
4444     FORMAT (' TL0.0,1.5;PA',F20.6,',',F20.6,';XT;')
10          DO 8 J=1,MID+1
               IF (MAJ .EQ. 0 .AND. J .EQ. 1) THEN
               IF (ISIDE .NE. 1) GO TO 5
               WRITE (6,555) XSTRT,YSTRT
555            FORMAT (' TL1.5;PA',F20.6,',',F20.6,';XT;')
               GO TO 20
5              IF (ISIDE .NE. 0) GO TO 33
               WRITE (6,666) XSTRT,YSTRT
666            FORMAT (' TL1.5,1.5;PA',F20.6,',',F20.6,';XT;')
               GO TO 20
33             WRITE (6,5555) XSTRT, YSTRT
5555           FORMAT (' TL0.0,1.5;PA',F20.6,',',F20.6,';XT;')
               END IF
20             IF (MIN .EQ. 0) GO TO 12
               DO 9 K= 1,MIN
                  IF (ISIDE .NE. 1) GO TO 2
                  WRITE (6,777)SPCMIN
777               FORMAT (' TL.5;PR',F26.12,', 0.0 ;XT;')
                  GO TO 9
2                 IF (ISIDE .NE. 0) GO TO 44
                  WRITE (6,888)  SPCMIN
888               FORMAT(' TL.5,.5;PR',F26.12,', 0.0 ;XT;')
                  GO TO 9
44                WRITE (6,6666) SPCMIN
6666              FORMAT (' TL0.0,0.5;PR',F26.12,',0.0 ;XT;')
9              CONTINUE
12             IF (MID .EQ. 0) GO TO 17
               IF (J .EQ. (MID+1)) GO TO 17
               IF (ISIDE .NE. 1) GO TO 3
               WRITE (6,999) SPCMIN
999            FORMAT (' TL1.0;PR',F26.12,', 0.0 ;XT;')
               GO TO 8
3              IF (ISIDE .NE. 0) GO TO 55
               WRITE (6,1111) SPCMIN
1111           FORMAT (' TL1.0,1.0;PR',F26.12,', 0.0;XT;')
               GO TO 8
55             WRITE (6,7777) SPCMIN
7777           FORMAT (' TL0.0,1.0;PR',F26.12,', 0.0;XT;')
8           CONTINUE
17       IF (ISIDE .NE. 1) GO TO 4
         WRITE (6,2222) SPCMIN
2222     FORMAT (' TL1.5;PR',F26.12,' , 0.0 ;XT;')
         GO TO 7
4        IF (ISIDE .NE. 0) GO TO 66
         WRITE (6,3333) SPCMIN
3333     FORMAT (' TL1.5,1.5;PR',F26.12,' , 0.0;XT;')
         GO TO 7
```

```
66        WRITE (6,8888) SPCMIN
8888      FORMAT (' TL0.0,1.5;PR',F26.12,', 0.0;XT;')
7       CONTINUE
C
        RETURN
        END


        SUBROUTINE YAXIS (A,XSTRT,YSTRT,END,ISIDE,MAJ,MID,MIN)
        CHARACTER*1 A
        DOUBLE PRECISION SPSMIN,DXSTRT,DYSTRT,DEND,RMAJ,RMID,RMIN
C
C       SUBROUTINE TO DRAW Y-AXIS
C
        WRITE (6,111) XSTRT,YSTRT
111     FORMAT (' PU;PA',F20.6,F20.6,';')
        WRITE (6,222) XSTRT,END,XSTRT,YSTRT
222     FORMAT (' PD',F15.6,F20.6,';PU',F15.6,F18.6,';')
        DXSTRT = XSTRT
        DYSTRT = YSTRT
        DEND   = END
        RMAJ=MAJ
        RMID=MID
        RMIN=MIN
        SPCMIN = (END-YSTRT)/((RMAJ-1.)*(RMID+1.)
       &  *(RMIN+1.))
        IF (MAJ .EQ. 0) SPCMIN = (END-YSTRT)/((RMID-1.)
       &  *(RMIN+1.))
           DO 7 I= 1,MAJ-1
              IF (MAJ .EQ. 0) GO TO 10
              IF (I .NE. 1) GO TO 10
              IF (ISIDE .NE. 1) GO TO 1
              WRITE (6,333) XSTRT,YSTRT
333           FORMAT (' TL1.5;PA',F20.6,',',F20.6,';YT;')
              GO TO 10
1             IF (ISIDE .NE. 0) GO TO 22
              WRITE (6,444) XSTRT,YSTRT
444           FORMAT (' TL1.5,1.5;PA',F20.6,',',F20.6,';YT;')
              GO TO 10
22            WRITE (6,4444) XSTRT, YSTRT
4444          FORMAT (' TL0.0,1.5;PA',F20.6,',',F20.6,';YT;')
10            DO 8 J=1,MID+1
                 IF (MAJ .EQ. 0 .AND. J .EQ. 1) THEN
                   IF (ISIDE .NE. 1) GO TO 5
                   WRITE (6,555) XSTRT,YSTRT
555                FORMAT (' TL1.5;PA',F20.6,',',F20.6,';YT;')
                   GO TO 20
5                  IF (ISIDE .NE. 0) GO TO 33
                   WRITE (6,666) XSTRT,YSTRT
666                FORMAT (' TL1.5,1.5;PA',F20.6,',',F20.6,';YT;')
                   GO TO 20
33                 WRITE (6,5555) XSTRT, YSTRT
5555               FORMAT (' TL0.0,1.5;PA',F20.6,',',F20.6,';YT;')
                 END IF
20               IF (MIN .EQ. 0) GO TO 12
                 DO 9 K= 1,MIN
                    IF (ISIDE .NE. 1) GO TO 2
                    WRITE (6,777)SPCMIN
777                 FORMAT (' TL.5;PR 0.0,',F26.12,';YT;')
                    GO TO 9
2                   IF (ISIDE .NE. 0) GO TO 44
                    WRITE (6,888)SPCMIN
888                 FORMAT (' TL.5,.5;PR 0.0,',F26.12,';YT;')
                    GO TO 9
44                  WRITE (6,6666) SPCMIN
6666                FORMAT (' TL0.0,.5;PR 0.0,',F26.12,';YT;')
9                CONTINUE
12               IF (MID .EQ. 0) GO TO 17
                 IF (J .EQ. (MID+1)) GO TO 17
                 IF (ISIDE .NE. 1) GO TO 3
                 WRITE (6,999) SPCMIN
999              FORMAT (' TL1.0;PR 0.0,',F26.12,';YT;')
                 GO TO 8
3                IF (ISIDE .NE. 0) GO TO 55
                 WRITE (6,1111)SPCMIN
1111             FORMAT (' TL1.0,1.0;PR 0.0,',F26.12,';YT;')
                 GO TO 8
55               WRITE (6,7777) SPCMIN
7777             FORMAT (' TL0.0,1.0;PR 0.0,',F26.12,';YT;')
```

```
8           CONTINUE
17        IF (ISIDE .NE. 1) GO TO 4
          WRITE (6,2222) SPCMIN
2222      FORMAT (' TL 1.5;PR 0.0,',F26.12,';YT;')
          GO TO 7
4         IF (ISIDE .NE. 0) GO TO 66
          WRITE (6,3333) SPCMIN
3333      FORMAT (' TL 1.5,1.5;PR 0.0,',F26.12,';YT;')
          GO TO 7
66        WRITE (6,8888) SPCMIN
8888      FORMAT (' TL0.0,1.5;PR 0.0,',F26.12,';YT;')
7       CONTINUE
C
        RETURN
        END


        SUBROUTINE DISPLAY(PGSIZE,P1X,P1Y,P2X,P2Y)
        CHARACTER*2 PGSIZE
        INTEGER EXT
C
C       SUBROUTINE TO SET THE GRAPH DISPLAY AREA
C
        EXT = 3
        IF (PGSIZE .NE. 'A4') GO TO 10
        UNIT = 10.00
        X1 = P1X * UNIT
        Y1 = P1Y * UNIT
        X2 = P2X * UNIT
        Y2 = P2Y * UNIT
        GO TO 30
10      IF (PGSIZE .NE. 'A3') GO TO 20
        UNIT = 15.00
        X1 = P1X * UNIT
        Y1 = P1Y * UNIT
        X2 = P2X * UNIT
        Y2 = P2Y * UNIT
        GO TO 30
20      IF (PGSIZE .NE. 'A1') GO TO 999
        UNIT = 30.0
        X1 =(P1X * UNIT) - 15000.0
        Y1 =(P1Y * UNIT) - 10500.0
        X2 =(P2X * UNIT) - 15000.0
        Y2 =(P2Y * UNIT) - 10500.0
C
30      WRITE (6,111) X1,Y1,X2,Y2
111     FORMAT (' IP',F15.6,F15.6,F15.6,F15.6,';')
        GO TO 9999
999     WRITE (6,12) EXT
12      FORMAT (' LBERROR IN PGSIZE',1R1)
C
9999    RETURN
        END


        SUBROUTINE BEGIN
        INTEGER ESCAPE
        CHARACTER*1 Y
C
C       SUBROUTINE TO INITIALIZE THE PLOTTER
C
        Y = 'Y'
        ESCAPE = 27
        WRITE (6,10) ESCAPE , Y
10      FORMAT (' ',1R1,'.',A1)
        WRITE (6,*) 'IN;DF;SP1;'
        RETURN
        END


        SUBROUTINE CLOSE
        INTEGER ESCAPE
C
C       SUBROUTINE TO DISACTIVATE THE PLOTTER
C
        CHARACTER*1 Z
        Z = 'Z'
```

```
      ESCAPE = 27
      WRITE (6,10)ESCAPE,Z
10    FORMAT (' ',1R1,'.',A1)
      RETURN
      END



      SUBROUTINE AXLBL (A,XSTRT,YSTRT,END,ISIDE,MAJ,FORM)
      CHARACTER FORMA*20
      CHARACTER*5 FORM
      DIMENSION XY(100)
      CHARACTER*1 A
      integer cont_c
      cont_c=3
      FORMA(1:20)="(' LB',     ,r1)"
C
      do 40 j=5,1,-1
         if(form(j:j).ge.'0'.and.form(j:j).le.'9')go to 50
40    continue
      stop 'Invalid format in AXLBL!'
50    forma(8:12)=form(1:j)
      RMAJ = MAJ
      WRITE (6,111) XSTRT,YSTRT
111   FORMAT (' PU',F20.6,F20.6,';')
      IF (A .NE. 'X') GO TO 100
      SPCMIN =  (END - XSTRT)/(RMAJ -1.0 )
      XY(1) = XSTRT
      DO 10 I = 2,MAJ
         RI = I
         XY(I) = XSTRT + ((RI -1.0) * SPCMIN)
10    CONTINUE
      DO 30 I=1,MAJ
         IF (ISIDE .NE. 1) GO TO 2
         WRITE (6,222) XY(I),YSTRT
222      FORMAT (' PA',F20.6,F20.6,';LO16;')
         GO TO 12
2        WRITE (6,333) XY(I),YSTRT
333      FORMAT (' PA',F20.6,F20.6,';LO14;')
12       WRITE (6,FORMA)   XY(I),cont_c
30    CONTINUE
      GO TO 999
C
C
100   SPCMIN = (END-YSTRT)/(RMAJ - 1.)
      XY(1)  = YSTRT
      DO 110 I=2,MAJ
          RI = I
          XY(I) = YSTRT + ((RI - 1.) * SPCMIN)
110   CONTINUE
      DO 300 I=1,MAJ
         IF (ISIDE .NE. 1) GO TO 22
         WRITE (6,444) XSTRT,XY(I)
444      FORMAT (' PA',F20.6,F20.6,';LO18;')
         GO TO 14
22       WRITE (6,555) XSTRT,XY(I)
555      FORMAT (' PA',F20.6,F20.6,';LO12;')
14       WRITE (6,FORMA)  XY(I),cont_c
300   CONTINUE
C
999       RETURN
          END



      SUBROUTINE CURVE (X,Y,NUM,LTYPE,PATLEN)
      DIMENSION  X(*),Y(*)
      CHARACTER*1 LTYPE
C
C     SUBROUTINE TO DRAW A LNEAR CURVE TO FIT A SET OF DATA POINTS
C
      IF (LTYPE(1:1) .EQ. ' ' ) THEN
          WRITE (6,11) X(1), Y(1)
11        FORMAT (' PU;PA',F20.6,F20.6,';CV1;LT;')
      ELSE
          WRITE (6,111)  X(1),Y(1),LTYPE,PATLEN
111       FORMAT (' PU;PA',F20.6,F20.6,';CV1;LT',A3,',',F10.3,';')
      END IF
      DO 10 I = 1,NUM
         WRITE (6,222) X(I),Y(I)
```

```
222      FORMAT (' PD',F20.6,F20.6,';')
10     CONTINUE
       WRITE (6,333) X(1),Y(1)
333    FORMAT (' CV0;PU',F20.6,F20.6,';')
C
       RETURN
       END


       SUBROUTINE LINE (X1,Y1,X2,Y2,LTYPE,PATLEN)
       CHARACTER*1 LTYPE
C
C      SUBROUTINE TO DRAW A STRAIGHT LINE CONNECTING TOW POINTS
C
       WRITE (6,111)  X1,Y1
111    FORMAT (' PA;PU',F20.6,F20.6,';')
       IF (LTYPE(1:1) .EQ. ' ' ) THEN
           WRITE (6,22) X2, Y2
22         FORMAT (' LT;PD',F20.7,F20.7,';')
       ELSE
           WRITE (6,222)  LTYPE,PATLEN,X2,Y2
222        FORMAT (' LT',A3,F10.3,';PD',F20.7,F20.7,';')
       END IF
       WRITE (6,*) 'LT;'
C
       RETURN
       END


       SUBROUTINE CHR_STYLE (ISTD,IALT,SELECT)
       CHARACTER*2 SELECT
C
C      SUBROUTINE TO SELECT THE CHOOSEN CHARACTER SYLE
C
       WRITE (6,*) 'CS',ISTD,';CA',IALT,';',SELECT,';'
       RETURN
       END


       SUBROUTINE TEXT_DIR (ANGLE)
C
C      SUBROUTINE TO SELECT THE CHOOSEN CHARACTER SYLE
C
       RAD = (ANGLE / 180.) * 3.14
       COSRAD = COS(RAD)
       SINRAD = SIN(RAD)
       WRITE (6,111) COSRAD,SINRAD
111    FORMAT (' DI',F20.6,F20.6,';')
C
       RETURN
       END


       SUBROUTINE NEXT (PG_SIZE)
       CHARACTER*2 PG_SIZE
C
C      SUBROUTINE TO ADVANCE THE PAGE (IF THIS FACILITY IS AVAIL-
C      ABLE IN THE USED PLOTTER)
C
       IF (PG_SIZE .NE. 'A4') GO TO 10
       WRITE (6,*) 'PG;IN;SP1;'
       GO TO 20
10     WRITE (6,*) 'NR;IN;SP1;'
C
20     RETURN
       END


       SUBROUTINE CHR_SIZE (W,H)
C
C      SUBROUTINE TO SET THE CHARACTER'S WIDTH AND HIGHT
C      (N.B.: THE WIDTH (W) & THE HIGHT (H) SHOULD BE IN CM.!)
C
       WRITE (6,*)  'SI',W,H,';'
       RETURN
```

```
      END


      SUBROUTINE TEXT (PGSIZE,SCL_STATE,X,Y,STRING,IORIGIN)
      CHARACTER FORMA*100
      CHARACTER   STRING*80
      CHARACTER*2 PGSIZE
      CHARACTER*3 SCL_STATE
      INTEGER EXT
C
C     SUBROUTINE TO DRAW A TEXT STRING
C
      EXT = 3
      J = INDEX (STRING , '$$')
C
      IF (SCL_STATE(1:2) .EQ. 'ON')  GO TO 40
15    IF (PGSIZE .NE. 'A4') GO TO 10
          PX = X * 10.00
          PY = Y * 10.00
      GO TO 30
C
10    IF (PGSIZE .NE. 'A3') GO TO 20
          PX = X * 15.00
          PY = Y * 15.00
      GO TO 30
C
20    IF (PGSIZE .NE. 'A1') GO TO 99
          PX =(X * 30.00) - 15000.0
          PY =(Y * 30.00) - 10500.0
C
30    WRITE (6,111)  PX,PY,IORIGIN
111   FORMAT (' PU;PA',F20.6,F20.6,';LO',I3,';')
      GO TO 50
40    WRITE (6,222)   X, Y,IORIGIN
222   FORMAT (' PU;PA',F20.6,F20.6,';LO',I3,';')
50    FORMA = "(' LB','"//STRING(1:J-1)//"',1R1)"
      WRITE (6,FORMA)  EXT
      GO TO 999
99    WRITE (6,12) EXT
12    FORMAT (' LBERROR IN PGSIZE!',1R1)
C
999   RETURN
      END


      SUBROUTINE PEN (IPEN)
C
C     SUBROUTINE TO SELECT THE CHOOSEN PEN NUMBER
C
      WRITE (6,*)'SP',IPEN,';'
C
      RETURN
      END


      SUBROUTINE DIR_OFF
C
C     SUBROUTINE TO TURN THE TEXT DIRECTION OFF
C
      WRITE (6,*) 'DI;'
      RETURN
      END


      SUBROUTINE SCALE_OFF
C
C     SUBROUTINE TO TURN THE SCALE OF THE PLOTTING OFF
C
      WRITE (6,*) 'SC;'
C
      RETURN
      END


      SUBROUTINE CHR_SLOPE (ANGLE)
```

```
C
C     SUBROUTINE TO SET THE CHARACTER SLOPE
C
      RAD = (ANGLE / 180.) * 3.14
      TANRAD = TAN (RAD)
      WRITE (6,111) TANRAD
111   FORMAT (' SL',F20.6,';')
C
      RETURN
      END


      SUBROUTINE SLOPE_OFF
C
C     SUBROUTINE TO TURN THE CHARACTER SLOPE OFF
C
      WRITE (6,*)'SL;'
      RETURN
      END


      SUBROUTINE SCALE (XSTRT,XEND,YSTRT,YEND)
      WRITE (6,111)  XSTRT,XEND,YSTRT,YEND
111   FORMAT (' SC',F20.6,F20.6,F20.6,F20.6,';')
C
      RETURN
      END


      SUBROUTINE SYMBOLS (X,Y,N,SYMB)
      DIMENSION X(*), Y(*)
      CHARACTER*1  SYMB
      NUM = N
      PRINT*,'PU',X(1),Y(1),';SM',SYMB,';'
      DO 10 I=1, NUM
        PRINT*,'PA',X(I),Y(I),';'
10    CONTINUE
      PRINT*,'SM;'
C
      RETURN
      END


      SUBROUTINE BROKEN_LINE (X,Y,LTYPE,PATLEN)
      DIMENSION X(*), Y(*)
        CHARACTER*1 LTYPE
        IF (LTYPE(1:1) .EQ. ' ' )  THEN
          WRITE (6,11) X(1), Y(1)
11        FORMAT (' PA;PU',F20.6,F20.6,';LT;')
        ELSE
          WRITE (6,111) X(1), Y(1), LTYPE, PATLEN
111       FORMAT (' PA;PU',F20.6,F20.6,';LT',A3,',',F10.3,';')
        END IF
      N = NUM
      DO 10 I = 2, N
      WRITE (6,222) X(I), Y(I)
222   FORMAT (' PD',F20.6,F20.6,';')
10    CONTINUE
      WRITE (6,333) X(1), Y(1)
333   FORMAT (' LT;PU',F20.6,F20.6,';')
C
      RETURN
      END


      SUBROUTINE INTEG_NUM (PGSIZE,SCL_STATE,X,Y,IVAR,FORM,IORIGIN)
      CHARACTER FORMA*25
      CHARACTER FORM*5
      CHARACTER*2 PGSIZE
      CHARACTER*3 SCL_STATE
      INTEGER EXT
      EXT = 3
          do 44 j=5,1,-1
               if(FORM(j:j).ge.'0'.and.FORM(j:j).le.'9')go to 55
44        continue
```

```
          stop 'Invalid format in AXLBL!'
55      FORMA = "(' LB'," //FORM(1:j)// ",1R1)"
C
        IF (SCL_STATE(1:2) .EQ. 'ON')  GO TO 40
15      IF (PGSIZE .NE. 'A4') GO TO 10
            PX = X * 10.00
            PY = Y * 10.00
        GO TO 30
C
10      IF (PGSIZE .NE. 'A3') GO TO 20
            PX = X * 15.00
            PY = Y * 15.00
        GO TO 30
C
20      IF (PGSIZE .NE. 'A1') GO TO 99
            PX =(X * 30.00) - 15000.0
            PY =(Y * 30.00) - 10500.0
C
30      WRITE (6,111)  PX,PY,IORIGIN
111     FORMAT (' PU;PA',F20.6,F20.6,';LO',I3,';')
        GO TO 50
40      WRITE (6,222)   X, Y,IORIGIN
222     FORMAT (' PU;PA',F20.6,F20.6,';LO',I3,';')
50      WRITE (6,FORMA) IVAR, EXT
        GO TO 999
99      WRITE (6,12) EXT
12      FORMAT (' LBERROR IN PGSIZE!',1R1)
C
999     RETURN
        END



        SUBROUTINE REAL_NUM (PGSIZE,SCL_STATE,X,Y,VAR,FORM,IORIGIN)
        CHARACTER FORMA*25
        CHARACTER FORM*5
        CHARACTER*2 PGSIZE
        CHARACTER*3 SCL_STATE
        INTEGER EXT
C
C       SUBROUTINE TO DRAW A REAL NUMBER
C
        EXT = 3
            do 44 j=5,1,-1
                if(FORM(j:j).ge.'0'.and.FORM(j:j).le.'9')go to 55
44          continue
            stop 'Invalid format in AXLBL!'
55      FORMA = "(' LB'," //FORM(1:j)// ",1R1)"
C
        IF (SCL_STATE(1:2) .EQ. 'ON')  GO TO 40
15      IF (PGSIZE .NE. 'A4') GO TO 10
            PX = X * 10.00
            PY = Y * 10.00
        GO TO 30
C
10      IF (PGSIZE .NE. 'A3') GO TO 20
            PX = X * 15.00
            PY = Y * 15.00
        GO TO 30
C
20      IF (PGSIZE .NE. 'A1') GO TO 99
            PX =(X * 30.00) - 15000.0
            PY =(Y * 30.00) - 10500.0
C
30      WRITE (6,111)  PX,PY,IORIGIN
111     FORMAT (' PU;PA',F20.6,F20.6,';LO',I3,';')
        GO TO 50
40      WRITE (6,222)   X, Y,IORIGIN
222     FORMAT (' PU;PA',F20.6,F20.6,';LO',I3,';')
50      WRITE (6,FORMA) VAR, EXT
        GO TO 999
99      WRITE (6,12) EXT
12      FORMAT (' LBERROR IN PGSIZE!',1R1)
C
999     RETURN
        END
```