Controller Estimation for the Adaptive Control of Robotic Manipulators

by

Guo Lin

(B Eng. Shanghai Jiao Tong University, PRC)

Department of Mechanical Engineering

McGill University

Montréal, Canada

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Master of Engineering

July 1987

ISBN 0-315-44478-9

To
my beloved parents
and Eva

Abstract

A method for estimating robotics systems controllers with time-varying parameters is presented in this thesis.

The robotic manipulator model is obtained using Kane's dynamical equations, which are then linearized about the desired trajectory. The linearized time-varying dynamical system, so derived, becomes the "plant" which is to be controlled.

A feedback control scheme, called *model assignment*, is derived. The feedback gains are designed to render the system a prescribed second-order stationary system by using *cancelling scheme*.

A recursive least-square identification method is introduced to perform the on-line parameter estimation for the feedback gains of the controller. Instead of estimating the parameters of the robotic dynamical model, the parameters of the controller are estimated. Therefore, the adaptive control algorithm is much simpler and computationally less demanding, compared with similar control schemes for robotic manipulators.

Dynamic modelling and adaptive control simulations were performed on a two-link manipulator, the Cincinnati Milacron Robot and PUMA 600 Robot. Satifactory results were obtained.

July 1987

Résumé

Cette thèse présente une méthode d'estimation de paramètres variables dans le temps, qui sert à la commande asservie de systèmes robotiques.

L'auteur obtient le modèle dynamique du robot manipulateur au moyen des équations dynamiques de Kané. Ces équations sont ensuite linéarisées autour de la trajectoire suivie par le robot, ce qui produit un système dynamique linéaire à coefficients variables avec le temps, sur lequel se base l'algorithme de commande faisant l'objet de la thèse. L'algorithme de commande asservie, appelé ici *algorithme d'assignation de modèle*, est obtenu à partir du modèle linéaire mentionné. En fait, les coefficients d'asservissement sont conçus façon à de rendre le système asservi un système linéaire stable à coefficients constants, ce qui est accompli au moyen d'une technique *d'annulation de termes*.

L'introduction d'un algorithme récursif d'identification de moindres carres permet d'estimer en lique les paramètres du système asservi. Ainsi, au lieu d'estimer les paramètres du modèle dynamique linéarisé du robot, l'algorithme en question estime les paramètres de la commande directement, ce qui le rend plus simple que les algorithmes similaires trouvés dans la littérature.

L'auteur inclue comme exemples d'application la simulation des systemes asservis d'un manipulateur à deux articulations, du robot PUMA 600, et du robot Cincinnati Milacron.

July 1987

Acknowledgements

I wish to thank Prof. J. Angeles for his guidance, encouragement and support which have been provided during the course of my research. His judicious suggestions made the successful completing of this thesis possible.

I wish to thank Mr. and Mrs Tsang for their kind help during my studies at McGill.

I wish to express my thanks to Ou Ma for his helpful suggestions in setting up the simulations, to Clément Gosselin for his help in translating the abstract of the thesis into French and to Edward Wielgus for prove reading the conclusions.

I would like to express my thanks to my colleagues and friends with whom I had the greatest times: Dr. Carlos López-Cajún and Dr. Lian-chai Zhao and Karen Anderson, Clément Gosselin, Zheng Cheng Lin, Zheng Liu, Ou Ma, Hai-Ming Qi and Jian Zhao for their pleasant personalities and humour.

I would finally like to thank McGill Research Center for Intelligent Machines for providing the facilities and plesent research environment.

July 1987

Table of Contents

List of Figures

Chapter 1 Dynamic Modelling Based on Kane's Equations

1.1 Introduction

Dynamic modelling is one of the most important parts of robotics research. Manipulators represent complicated dynamical systems and if we want to analyse their dynamics, we must take a very systematic approach for solving the problem We are interested in obtaining an efficient way to derive models of manipulators as this provides insight into the control problem.

The dynamical model for a six-link robot manipulator can be obtained by using physical laws such as the laws of Newtonian mechanics, Lagrangian mechanics and by using physical measurements. The task is to develop the dynamical equations of motion for the overall robot manipulator in terms of its parameters. Then, the procedures based either on Newton-Euler or Euler-Lagrange equations are used to develop the governing equations.

The use of the Euler-Lagrange formulation was studied by Uicker(1965), Paul(1972, 1981), Bejczy(1974), Lewis(1974) and Lee et al. (1982). The derivation of the dynamical model of a manipulator based on the Euler-Lagrange formulation is simple and systematic. The important feature of the work of the early researchers is their use of 4×4 rotation/translation matrices $\mathbf{B}_i$ to represent the motion of the kinematic chain. Matrix $\mathbf{B}_i$ transforms components of vectors and matrices in the i-th link coordinates to the base coordinates. The kinetic and potential energy of the kinematic chain is expressed in terms of the $\mathbf{B}_i$s and their derivatives. Then the Lagrange equations can be used to compute the generalized forces as followings:

$$\tau_i = \sum_{j=i}^{n} Tr(\frac{\partial \mathbf{B}_j}{\partial q_i} \mathbf{J}_j \ddot{\mathbf{B}}_j^T) - \sum_{j=i}^{n} m_j \mathbf{g}^T \frac{\partial \mathbf{B}_j}{\partial q_i} \mathbf{r}_j \qquad i = 1, ..., n \tag{1.1.1}$$

where $\mathbf{J}_j$ is the inertia tensor of the j-th link, q_i is the i-th joint variable (angle for rotation joint and length for prismatic joint),τ_i is the generalized force applied to the i-th link, Tr is, trace operator, $\mathbf{r}_j$ is a vector from the origin of the j-th link to its mass centre and $\mathbf{g}$ is the gravity vector.

The Newton-Euler formulation is based on the laws governing the dynamics of rigid bodies. The relationship between the vector force acting at the mass centre of a given link and the acceleration of its mass centre is Newton's second law:

$$\mathbf{F} = m\dot{\mathbf{v}} \tag{1.1.2}$$

The vector moment about the mass centre is related to the angular velocity and angular acceleration of the rigid body by Euler's equation:

$$\mathbf{n} = \mathbf{J} \cdot \dot{\omega} + \omega \times (\mathbf{J} \cdot \omega) \tag{1.1.3}$$

where ω is the angular velocity and $\mathbf{J}$ is the inertia tensor The vector moment acting on the i-th link can be calculated by using Newton-Euler equation. Stepanenko and Vukobratovic(1976) derived the dynamics of human limbs by using the Newton-Euler equations In 1979 Orin et al revised Stepanenko and Vukobratovic's method to make it more efficient. They used the coordinate systems attached to the links instead of the fixed coordinates used by Stepaneko and Vukobratovic Luh, Walker and Paul(1980) focused on the efficiency of the foregoing computations They found that the computational time of the algorithm based on Newton-Euler grows linearly with the number of links, while the one of the algorithm based on the original Lagrange fomulation grows with the fourth power of the number of links.

In 1980, Hollerbach realized that the recursive nature of the Newton-Eulor formulation that makes it so efficient could be achieved with Lagranian formulations as well. Silver(1982) found that there is in fact no fundamental difference in computational efficiency between Lagrangian and Newton-Euler formulations.

Kane's dynamical equations (Kane, Likins and Levinson 1983), which were originally derived for nonholonomic systems, have been used in robotics area in the recent years. For a given robot, Kane's equations enable one to work systematically with dependent variables to eliminate effortless nonworking constraint forces and torques.

The package introduced here implements Kane's dynamical equations. Kinematic and dynamic formulae were derived by Rojas (1987)and the program based on these formulations was implemented and tested on a VAX-780 (McRCIM) by the author of this thesis. The simulation results are reported here.

1.2 Manipulator Kinematics

1.2.1 Hartenberg-Denavit Notation and Rotation Matrix

A systematic and generalized approach utilizing matrix algebra for describing the spatial geometry of coupled mechanical systems with respect to a fixed reference frame was developed by Hartenberg and Denavit in 1955. It has been widely used in the area of robotics in recent years. The advantage of their representation of the linkages is its algorithm universality in deriving the kinematic equations of a manipulator.

The parameters of the Hartenberg-Denavit notation are described as follows:

α_i:	twist angle from Z_{i-1} axis to Z_i-axis in the positive direction of axis X_{i-1}
a_i:	distance between Z_{i-1} and Z_i axes
b_i:	Z_{i-1} coordinate of the intersection of axes X_i and Z_{i-1}
θ_i:	angle from X_{i-1} to X_i about the Z_i axis using the right-hand rule

where $i = 1,...n$ and n is the number of joints

Hence, the rotation matrix $\mathbf{Q}_i$, which transforms vector components in the $(i+1)$-st link frame to the i-th link frame, is expressed as

$$\mathbf{Q}_i = \begin{pmatrix} \cos\theta_i & \sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & \cos\theta_i\sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{pmatrix} \tag{1.2.1 1}$$

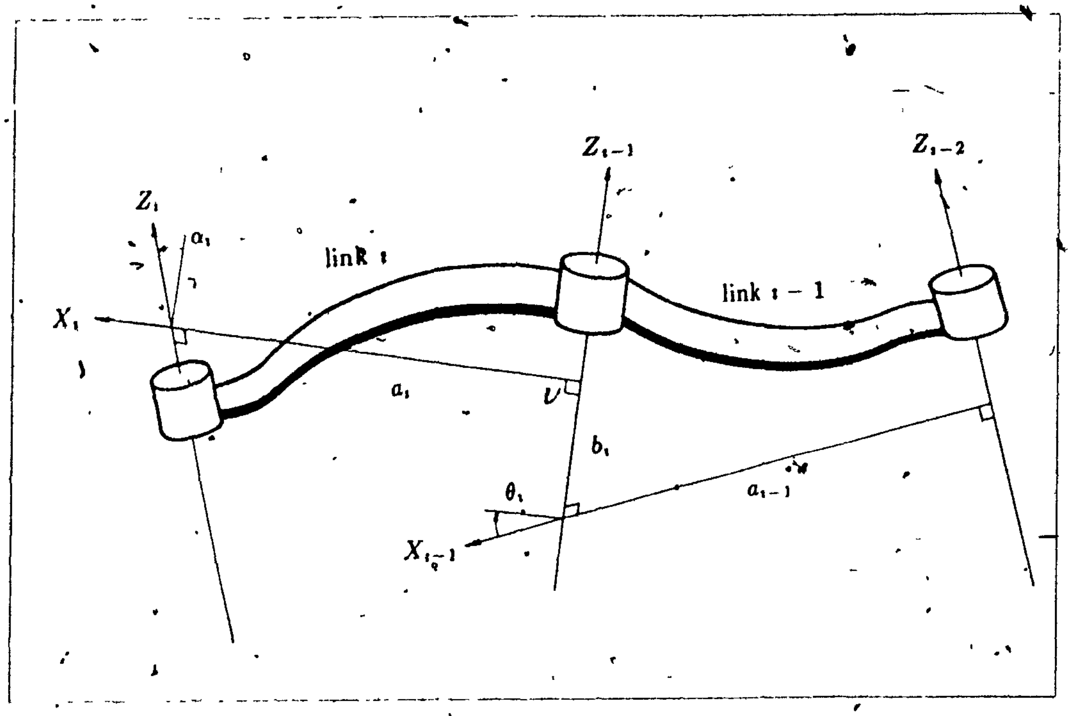

Figure 1.1 Hartenberg-Denavit Parameters

Let $[\mathbf{c}]_{i+1}$ be an arbitrary vector expressed in $(i+1)$-st coordinates, then, the following relationship holds:

$$[\mathbf{c}]_j = \mathbf{Q}_j \cdots \mathbf{Q}_i [\mathbf{c}]_{i+1} \qquad j < i \qquad (1.2.1.2)$$

Based on the rotation matrix represented by eq.(1.2.1.1), matrix $\mathbf{P}_i$ is defined as follows:

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{1} \\ \mathbf{P}_1 &= \mathbf{Q}_1 \\ \mathbf{P}_2 &= \mathbf{P}_1\mathbf{Q}_2 \\ &\vdots \\ \mathbf{P}_n &= \mathbf{P}_{n-1}\mathbf{Q}_n \end{aligned} \qquad (1.2.1.3)$$

1.2.2 Kinematic Analysis of the Robotic System

Based on the Hartenberg-Denavit notation described in the last sub-section, the kinematic analysis of the robotic system is performed. It is pointed out that a vector $\mathbf{v}$

or a matrix **A** represented in the i-th coordinates is indicated as $[\mathbf{v}]_i$ or $[\mathbf{A}]_i$, respectively. However, if no indication of coordinates is given, baseframe coordinates will be implied.

First of all, a definition, which will be proved to be very useful, is introduced. To any 3-dimensional vector **f**, one can attribute a *unique* 3×3 tensor **F** defined as

$$\mathbf{F} \equiv \frac{\partial(\mathbf{f} \times \mathbf{x})}{\partial \mathbf{x}} \tag{1.2.2.1}$$

where **x** is an arbitrary vector.

Alternatively, the forgoing tensor **F** can be defined as:

$$\mathbf{F} \equiv \mathbf{f} \times \mathbf{1} \equiv \mathbf{1} \times \mathbf{f} \tag{1.2.2.2}$$

where

$$\mathbf{f} = \mathrm{vect}(\mathbf{F}) \tag{1.2.2.2a}$$

Therefore, the following relationship holds:

$$\mathbf{f} \times \mathbf{g} = (\mathbf{f} \times \mathbf{1})\mathbf{g} = \mathbf{F}\mathbf{g} \tag{1.2.2.3}$$

where **g** is an arbitrary 3-dimensional vector.

Let $\mathbf{u}_i$ be the vector from the base coordinate origin O_1 to the i-th coordinate origin O_i, $\mathbf{r}_i$ be the vector from the base coordinate origin O_1 to the centroid of the i-th linkage C_i, ρ_{i+1} be the vector from the $(i+1)$-st origin to the centroid of the i-th linkage and $\mathbf{s}_{i+1}$ be the vector from C_i to the origin of the robot end effector O_{n+1}. The vector from the i-th origin to the end effector of the manipulator is defined as $\mathbf{v}_i$.

$$\begin{aligned}
[\mathbf{v}_n]_n &= [\mathbf{a}_n]_n \\
&\vdots \\
[\mathbf{v}_i]_i &= [\mathbf{a}_i]_i + \mathbf{Q}_i[\mathbf{v}_{i+1}]_{i+1} \qquad i = n-1, n-2, \ldots, 1 \\
&\vdots \\
\mathbf{v}_1 &= \mathbf{a}_1 + \mathbf{Q}_1[\mathbf{v}_2]_2
\end{aligned} \tag{1.2.2.4}$$

where $[\mathbf{a}_i]_i$ is the vector from the i-th origin O_i to the $(i+1)$-st origin O_{i+1} expressed in the i-th coordinates.

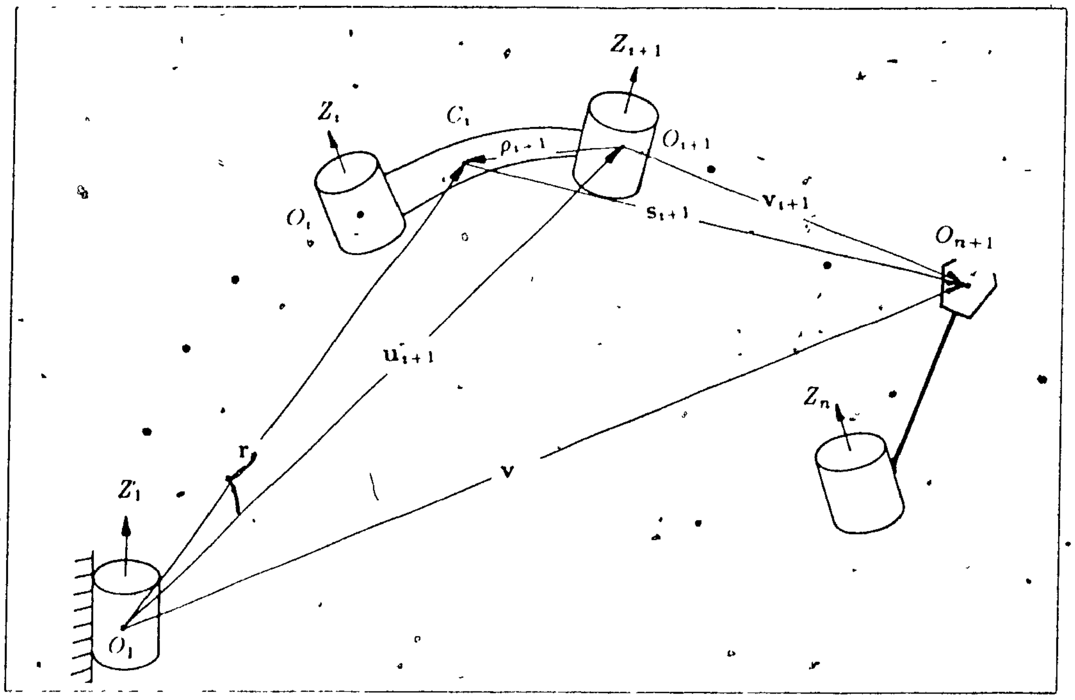

Figure 1.2 Vector definitions for the n-link manipulator

Let

$$\begin{aligned}\mathbf{v} = [\mathbf{v}_1]_1 &= [\mathbf{a}_1]_1 + \mathbf{Q}_1[\mathbf{v}_2]_2 \\ &= [\mathbf{a}_1]_1 + \mathbf{P}_1[\mathbf{a}_2]_2 + \mathbf{P}_2[\mathbf{a}_3]_3 + \dots + \mathbf{P}_{n-1}[\mathbf{a}_n]_n \\ &= [\mathbf{a}_1]_1 + [\mathbf{a}_2]_1 + \dots + [\mathbf{a}_n]_1 \end{aligned} \qquad (1.2.2.5)$$

Differentiation of eq (1.2.2.5) with respect to time leads to:

$$\dot{\mathbf{v}} = [\dot{\mathbf{a}}_1]_1 + [\dot{\mathbf{a}}_2]_1 + \dots + [\dot{\mathbf{a}}_n]_1 \qquad (1.2.2.6)$$

where

$$[\dot{\mathbf{a}}_{i+1}]_1 = \dot{\mathbf{P}}_i[\mathbf{a}_{i+1}]_{i+1} + \mathbf{P}_i[\dot{\mathbf{a}}_{i+1}]_{i+1} \qquad (1.2.2.7)$$

and $\mathbf{P}_i$ is defined in eq (1.2.1.3).

In order to find $[\dot{\mathbf{a}}_{i+1}]_1$, we have to calculate $[\dot{\mathbf{a}}_{i+1}]_{i+1}$ and $\dot{\mathbf{P}}_i$ first. They can be derived as follows:

$$[\dot{\mathbf{a}}_{i+1}]_{i+1} = \dot{\theta}_{i+1}\mathbf{e} \times [\mathbf{a}_{i+1}]_{i+1} \qquad (1.2.2.8)$$

Now $\dot{\mathbf{P}}_i$ is calculated as:

$$\dot{\mathbf{P}}_i = \dot{\theta}_1\frac{\partial \mathbf{P}_1}{\partial \theta_1} + \dots + \dot{\theta}_i\frac{\partial \mathbf{P}_i}{\partial \theta_i} \qquad (1.2.2.9)$$

and

$$\frac{\partial \mathbf{P}_i}{\partial \theta_j} = \mathbf{P}_{j-1}\mathbf{e} \times \mathbf{P}_i \qquad j \le i \tag{1.2.2.10}$$

where $\mathbf{e}$ is a unit vector parallel to the Z_i axis expressed in its own coordinate, namely, $\mathbf{e} = [\mathbf{e}_i]_i = [0,0,1]^T$ and θ_i is the joint variable of the i-th link.

The angular velocity and angular acceleration of each link can be expressed as:

$$\omega_i = \dot{\theta}_1\mathbf{e}_1 + \ldots + \dot{\theta}_i\mathbf{e}_i = [\mathbf{e}_1, \ldots, \mathbf{e}_i, 0, \ldots, 0]\dot{\theta} = \mathbf{A}_i\dot{\theta} \tag{1.2.2.11}$$

i.e.

$$\begin{aligned} \omega_1 &= \dot{\theta}_1\mathbf{e}_1 \\ \omega_2 &= \omega_1 + \dot{\theta}_2\mathbf{e}_2 \\ &\vdots \\ \omega_n &= \omega_{n-1} + \dot{\theta}_n\mathbf{e}_n \end{aligned} \tag{1.2.2.12}$$

where $\theta = [\theta_1, \ldots, \theta_n]^T$ is a vector of joint variables.

Therefore, the angular accelerations is in the following form:

$$\begin{aligned} \dot{\omega}_1 &= \ddot{\theta}_1\mathbf{e}_1 \\ \dot{\omega}_2 &= \dot{\omega}_1 + \ddot{\theta}_2\mathbf{e}_2 + \omega_1 \times \dot{\theta}_2\mathbf{e}_2 \\ &\vdots \\ \dot{\omega}_n &= \dot{\omega}_{n-1} + \ddot{\theta}_n\mathbf{e}_n + \omega_{n-1} \times \dot{\theta}_n\mathbf{e}_n \end{aligned} \tag{1.2.2.13}$$

Rewriting eq.(1.2.2.13) in a simple way, we get:

$$\dot{\omega}_i = \mathbf{A}_i\ddot{\theta} + \dot{\mathbf{A}}_i\dot{\theta} \tag{1.2.2.14}$$

where

$$\dot{\mathbf{A}}_i = [0, \omega_1 \times \mathbf{Q}_1\mathbf{e}, \ldots, \omega_i \times \mathbf{P}_i\mathbf{e}] \tag{1.2.2.15}$$

From eqs.(1.2.2.9) and (1.2.2.10), we can obtain an expression for $\dot{\mathbf{P}}_i$ as follows:

$$\begin{aligned} \dot{\mathbf{P}}_i &= \dot{\theta}_1\mathbf{e}_1 \times \mathbf{P}_i + \dot{\theta}_2\mathbf{e}_2 \times \mathbf{P}_i + \cdots + \dot{\theta}_i\mathbf{e}_i \times \mathbf{P}_i \\ &= (\dot{\theta}_1\mathbf{e}_1 + \dot{\theta}_2\mathbf{e}_2 + \cdots + \dot{\theta}_i\mathbf{e}_i) \times \mathbf{P}_i \\ &= \omega_i \times \mathbf{P}_i \end{aligned} \tag{1.2.2.16}$$

Substitution of eqs (1.2.2.7), (1.2.2.8) and (1.2.2.16) into eq.(1.2.2.6) yields.

$$\begin{aligned}\dot{\mathbf{v}} &= \dot\theta_1\mathbf{e}_1 \times \mathbf{a}_1 + (\dot\theta_1\mathbf{e}_1 + \dot\theta_2\mathbf{e}_2) \times [\mathbf{a}_2]_1 + \ldots + (\dot\theta_1\mathbf{e}_1 + \ldots + \dot\theta_n\mathbf{e}_n) \times [\mathbf{a}_n]_1 \\ &= \dot\theta_1\mathbf{e}_1 \times (\mathbf{a}_1 + [\mathbf{a}_2]_1 + \ldots + [\mathbf{a}_n]_1) + \dot\theta_2\mathbf{e}_2 \times ([\mathbf{a}_2]_1 + \ldots + [\mathbf{a}_n]_1) + \ldots + \dot\theta_n\mathbf{e}_n \times [\mathbf{a}_n]_1 \\ &= \dot\theta_1\mathbf{e} \times \mathbf{v}_1 + \dot\theta_2\mathbf{P}_1(\mathbf{e} \times \mathbf{v}_2) + \ldots + \dot\theta_n\mathbf{P}_{n-1}(\mathbf{e} \times \mathbf{v}_n)\end{aligned} \qquad (1.2.2.17)$$

On the other hand, $\dot{\mathbf{v}}$ can be written as

$$\begin{aligned}\dot{\mathbf{v}} &= \frac{\partial\mathbf{v}}{\partial\theta_1}\dot\theta_1 + \ldots + \frac{\partial\mathbf{v}}{\partial\theta_n}\dot\theta_n \\ &= \frac{\partial\mathbf{v}}{\partial\boldsymbol\theta}\dot{\boldsymbol\theta}\end{aligned} \qquad (1.2.2.18)$$

Hence,

$$\begin{aligned}\frac{\partial\mathbf{v}}{\partial\theta_1} &= \mathbf{e} \times \mathbf{v} \\ \frac{\partial\mathbf{v}}{\partial\theta_2} &= \mathbf{P}_1(\mathbf{e} \times \mathbf{v}_2) \\ &\vdots \\ \frac{\partial\mathbf{v}}{\partial\theta_n} &= \mathbf{P}_{n-1}(\mathbf{e} \times \mathbf{v}_n)\end{aligned} \qquad (1.2.2.19)$$

From eq.(1.2.2.4) we have $[\mathbf{v}_i]_i = [\mathbf{a}_i]_i + \mathbf{Q}_i[\mathbf{v}_{i+1}]_{i+1}$, hence, the derivative of $[\mathbf{v}_i]_i$ with respect to θ_j $(j > i)$ is obtained as

$$\begin{aligned}\frac{\partial[\mathbf{v}_i]_i}{\partial\theta_j} &= 0 + \mathbf{Q}_i\frac{\partial[\mathbf{v}_{i+1}]_{i+1}}{\partial\theta_j} \\ &= \mathbf{Q}_i \cdots \mathbf{Q}_{j-1}\frac{\partial[\mathbf{a}_j]_j}{\partial\theta_j} + \mathbf{Q}_i \cdots \mathbf{Q}_{j-1}\frac{\partial\mathbf{Q}_j}{\partial\theta_j}\mathbf{Q}^T\mathbf{Q}\mathbf{v}\end{aligned} \qquad (1.2.2.20)$$

Notice that, for the rotation matrix $\mathbf{Q}_i$, the following relationship holds:

$$\frac{\partial\mathbf{Q}_i}{\partial\theta_i}\mathbf{Q}_i^T = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \equiv \mathbf{E} = \mathbf{1} \times \mathbf{e} \qquad (1.2.2.21)$$

Therefore, from eqs (1.2.2.3), (1.2.2.20) and (1.2.2.21) we can get:

$$\begin{aligned}\frac{\partial[\mathbf{v}_i]_i}{\partial\theta_j} &= \mathbf{Q}_i, \ldots, \mathbf{Q}_{j-1}\mathbf{e} \times ([\mathbf{a}_j]_j + \mathbf{Q}_j[\mathbf{v}_{j+1}]_{j+1}) \\ &= \mathbf{Q}_i \cdots \mathbf{Q}_{j-1}(\mathbf{e} \times [\mathbf{v}_j]_j)\end{aligned} \qquad (1.2.2.22)$$

Some other kinematic formulae, which will be used in the modelling of the robotic manipulator, are derived as follows:

$$\mathbf{u}_{i+1} = \mathbf{v} - \mathbf{P}_i[\mathbf{v}_{i+1}]_{i+1} \tag{1.2.2.23}$$

$$\mathbf{r}_i = \mathbf{u}_{i+1} + \mathbf{P}_i[\rho_{i+1}]_{i+1} \tag{1.2.2.24}$$

From eqs.(1.2.2.23) and (1.2.2.24) we get:

$$\begin{aligned}\mathbf{r}_i &= \mathbf{v} - \mathbf{P}_i([\mathbf{v}_{i+1}]_{i+1} - [\rho_{i+1}]_{i+1}) \\ &= \mathbf{v} - \mathbf{P}_i[\mathbf{s}_{i+1}]_{i+1}\end{aligned} \tag{1.2.2.25}$$

By applying eqs.(1.2.2.5) and (1.2.2.25), $\mathbf{r}_i$ can be expanded as:

$$\begin{aligned}\mathbf{r}_i &= \mathbf{a}_1 + \mathbf{Q}_1[\mathbf{a}_2]_2 + \dots + \mathbf{P}_{n-1}[\mathbf{a}_n]_n \\ &\quad - \mathbf{P}_i([\mathbf{a}_{i+1}]_{i+1} + \mathbf{Q}_{i+1}[\mathbf{a}_{i+2}]_{i+2} + \dots + \mathbf{Q}_{i+1}\cdots\mathbf{Q}_{n-1}[\mathbf{a}_n]_{n+1} - [\rho_{i+1}]_{i+1}) \\ &= \mathbf{a}_1 + \mathbf{Q}_1[\mathbf{a}_2]_1 + \dots + \mathbf{P}_{i-1}[\mathbf{a}_i]_i + \mathbf{P}_i[\rho_{i+1}]_{i+1}\end{aligned} \tag{1.2.2.26}$$

From eq.(1.2.2.25), $\dot{\mathbf{r}}_i$ can be derived as

$$\begin{aligned}\dot{\mathbf{r}}_i &= \dot{\mathbf{v}} - \dot{\mathbf{P}}_i[\mathbf{s}_{i+1}]_{i+1} - \mathbf{P}_i[\dot{\mathbf{v}}_{i+1}]_{i+1} + \mathbf{P}_i[\dot{\rho}_{i+1}]_{i+1} \\ &= \frac{\partial \mathbf{v}}{\partial \theta}\dot{\theta} - \omega_i \times \mathbf{P}_i[\mathbf{s}_{i+1}]_{i+1} - \mathbf{P}_i\frac{\partial[\mathbf{v}_{i+1}]_{i+1}}{\partial \theta}\dot{\theta} \\ &= \dot{\theta}_1\mathbf{e} \times \mathbf{v} + \dot{\theta}_2\mathbf{P}_1(\mathbf{e} \times [\mathbf{v}_2]_2) + \dots + \dot{\theta}_n\mathbf{P}_{n-1}(\mathbf{e} \times [\mathbf{v}_n]_n) + \\ &\quad - (\dot{\theta}_1\mathbf{e} + \dot{\theta}_2\mathbf{P}_1\mathbf{e} + \dots + \dot{\theta}_i\mathbf{P}_{i-1}\mathbf{e}) \times \mathbf{P}_i[\mathbf{s}_{i+1}]_{i+1} - \dot{\theta}_{i+1}\mathbf{P}_i\mathbf{e} \times [\mathbf{v}_{i+1}]_{i+1} + \\ &\quad - \dot{\theta}_{i+2}\mathbf{P}_i\mathbf{Q}_{i+1}\mathbf{e} \times [\mathbf{v}_{i+2}]_{i+2} - \dots - \dot{\theta}_n\mathbf{P}_{n-1}\mathbf{e} \times [\mathbf{v}_n]_{n+1}\end{aligned} \tag{1.2.2.27}$$

Notice that, in eq.(1.2.2.27), since ρ_{i+1} is the constant in its own coordinate, hence, $[\dot{\rho}_{i+1}]_{i+1} = 0$.

However, $\dot{\mathbf{r}}_i$ can be presented in another way as follows:

$$\begin{aligned}\dot{\mathbf{r}}_i(\theta) &= \frac{\partial \mathbf{r}_i}{\partial \theta_1}\dot{\theta}_1 + \dots + \frac{\partial \mathbf{r}_i}{\partial \theta_i}\dot{\theta}_i \\ &= \frac{\partial \mathbf{r}_i}{\partial \theta}\dot{\theta}\end{aligned} \tag{1.2.2.28}$$

By comparing eqs.(1.2.2.27) and (1.2.2.28), we get:

$$\frac{\partial \mathbf{r}_i}{\partial \theta_j} = \mathbf{P}_{j-1}\mathbf{e} \times [\mathbf{v}_j - \mathbf{Q}_j \ldots \mathbf{Q}_i [\mathbf{s}_{i+1}]_{i+1}] \qquad j < i$$
$$= 0 \qquad j > i$$
$$\frac{\partial \mathbf{r}_n}{\partial \theta_j} = \frac{\partial \mathbf{v}}{\partial \theta_j} \qquad (1.2.2.29)$$

Up to now, the kinematic formulae which are needed to perform the dynamical modelling have been derived. The dynamical analysis will be described in the next section.

1.3 Dynamic Modelling

1.3.1 Kinetic Energy

The kinetic energy $\mathcal{T}$ of a robotic manipulator in a reference coordinate is defined as

$$\mathcal{T} = \mathcal{T}_\omega + \mathcal{T}_v \qquad (1.3.1.1)$$

$\mathcal{T}_\omega$ is called rotational kinetic energy and $\mathcal{T}_v$ is called translational kinetic energy. They depend, respectively, on the angular velocity and the velocity of the mass centre $\dot{\mathbf{r}}$. So, $\mathcal{T}$ can be written as

$$\mathcal{T} = \frac{1}{2}\sum_{i=1}^{n}(m_i \dot{\mathbf{r}}_i^T \mathbf{r}_i + \omega_i^T \mathbf{J}_i \omega_i) \qquad (1.3.1.2)$$

Substitution of eqs.(1.2.2.11) and (1.2.2.28) into eq.(1.3.1.2) yields:

$$\mathcal{T} = \frac{1}{2}\dot{\theta}^T \mathbf{J}(\theta)\dot{\theta} \qquad (1.3.1.3)$$

where

$$\mathbf{J}(\theta) = \sum_{i=1}^{n}[m_i (\frac{\partial \mathbf{r}_i}{\partial \theta})^T \frac{\partial \mathbf{r}_i}{\partial \theta} + \mathbf{A}_i^T \mathbf{J}_i \mathbf{A}_i \qquad (1.3.1.4)$$

In the above equations, m_i is mass of the i-th link and $\mathbf{A}_i$ is expressed in eq.(1.2.2.11).

1.3.2. Potential Engery

Let the total potential energy of a robotic manipulator be W. Then, the potential energy of the i-th link is in the following form:

$$W_i = -m_i \mathbf{g}^T \mathbf{r}_i \qquad i = 1, ...n \qquad (1.3.2.1)$$

where $\mathbf{g}$ is the gravity acceleration vector of $[g_x, g_y, g_z]^T$ which is expressed in the base coordinates. For a system at sea-level base, $\mathbf{g} = [0, 0, -9.81]^T$ providing Z_1 is the vertical. The total potential energy of the manipulator can be obtained by summing all the potential energies of the links, i.e.,

$$W = -\sum_{i=1}^{n} m_i \mathbf{g}^T \mathbf{r}_i \qquad (1.3.2.2)$$

1.3.3 Dynamical equations

Given a robotic system having n degrees of freedom in a Newtonian reference frame, the motion of the system is governed by following equations:

$$\tau_j + \tau_j^* = 0 \qquad j = 1, ..., n \qquad (1.3.3.1)$$

where $\tau_1, ..., \tau_n$ are the generalized active forces and $\tau_1^*, .., \tau_n^*$ are the generalized inertia forces. These equations are called *Kane's dynamical equations.*

The generalized inertia force acting on the j-th link can be expressed as

$$\tau_j^* = -\left(\frac{\partial T}{\partial \theta_j} - \frac{d}{dt}\frac{\partial T}{\partial \theta_j} - \frac{\partial W}{\partial \theta_j}\right) \qquad j - 1, \ldots n \qquad (1.3.3.2)$$

Now we compute each part of eq. (1.3 3 2)

$$\frac{\partial T}{\partial \theta} = \mathbf{J}(\theta)\dot{\theta} \qquad (1.3.3.3)$$

For each link, we have.

$$\frac{\partial T}{\partial \dot{\theta}_j} = \sum_{i=1}^{n} [m_i \frac{\partial \mathbf{r}_i}{\partial \theta_j} \frac{\partial \mathbf{r}_i}{\partial \theta} \dot{\theta} + \mathbf{P}_{j-1}\mathbf{e} \cdot \mathbf{J}_i \mathbf{A}_i \dot{\theta}]$$

$$= \sum_{i=1}^{n} [m_i \frac{\partial \mathbf{r}_i}{\partial \theta_j} \cdot \dot{\mathbf{r}}_i + \mathbf{P}_{j-1}\mathbf{e} \cdot \mathbf{J}_i \omega_i] \qquad (1.3.3.4)$$

Differentiation of both sides of eq. (1.3.3.4) with respect to time yields:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot\theta_j}\right) = \sum_{i=1}^{n}[m_i\dot{\mathbf{r}}_i \cdot \frac{d}{dt}\left(\frac{\partial \dot{\mathbf{r}}_i}{\partial \dot\theta_j}\right) + \frac{\partial \mathbf{r}_i}{\partial \theta_j} \cdot m_i\ddot{\mathbf{r}}_i + \dot{\mathbf{P}}_{j-1}\mathbf{e} \cdot \mathbf{J}_i\omega_i + \mathbf{P}_{j-1}\mathbf{e} \cdot \dot{\mathbf{J}}_i\omega_i + \mathbf{P}_{j-1}\mathbf{e} \cdot \mathbf{J}_i\dot\omega_i] \tag{1.3.3.5}$$

where

$$\frac{d}{dt}\left(\frac{\partial \mathbf{r}_i}{\partial \theta_j}\right) = \frac{\partial \dot{\mathbf{r}}_i}{\partial \theta_j} \tag{1.3.3.6}$$

and

$$\dot{\mathbf{J}}_i = \omega_i \times \mathbf{J}_i - \mathbf{J}_i \times \omega_i \tag{1.3.3.7}$$

Hence, from eqs (1.2.2.16), (1.3.3.6) and (1.3.3.7), eq.(1.3.3.5) can be written as:

$$\begin{aligned}\frac{d}{dt}\left(\frac{\partial T}{\partial \dot\theta_j}\right) = \sum_{i=1}^{n}[m_i\dot{\mathbf{r}}_i \cdot \frac{\partial \dot{\mathbf{r}}_i}{\partial \theta_j} + \omega_{j-1} \times \mathbf{P}_{j-1}\mathbf{e} \cdot (\mathbf{J}_i\omega_i) + m_i\frac{\partial \mathbf{r}_i}{\partial \theta_j}m_i\ddot{\mathbf{r}}_i + \\ + \mathbf{P}_{j-1}\mathbf{e} \cdot (\omega_i \times \mathbf{J}_i\omega_i) + \mathbf{P}_{j-1}\mathbf{e} \cdot \mathbf{J}_i\dot\omega_i)]\end{aligned} \tag{1.3.3.8}$$

The partial derivative of the kinetic energy T with respect to θ_j is.

$$\frac{\partial T}{\partial \theta_j} = \sum_{i=1}^{n}[m_i\dot{\mathbf{r}}_i \cdot \frac{\partial \dot{\mathbf{r}}_i}{\partial \theta_j} + \frac{\partial \omega_i}{\partial \theta_j} \cdot (\mathbf{J}_i\omega_i) + \frac{1}{2}\omega_i^T\frac{\partial \mathbf{J}_i}{\partial \theta_j}\omega_i] \tag{1.3.3.9}$$

From eq.(1.2.2.10) we can get

$$\begin{aligned}\frac{\partial \omega_i}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}[\dot\theta_1\mathbf{e} + \dot\theta_2\mathbf{P}_1\mathbf{e} + \ldots + \dot\theta_i\mathbf{P}_{i-1}\mathbf{e}] \\ &= \dot\theta_j\frac{\partial \mathbf{P}_{j-1}}{\partial \theta_j}\mathbf{P}_{j-1}^T\mathbf{P}_{j-1}\mathbf{e} + \ldots + \dot\theta_i\frac{\partial \mathbf{P}_{i-1}}{\partial \theta_j}\mathbf{P}_{i-1}^T\mathbf{P}_{i-1}\mathbf{e} \\ &= \mathbf{P}_{j-1}\mathbf{e} \times (\dot\theta_j\mathbf{P}_{j-1}\mathbf{e} + \ldots + \dot\theta_i\mathbf{P}_{i-1}\mathbf{e}) \\ &= \mathbf{P}_{j-1}\mathbf{e} \times (\omega_i - \omega_{j-1}) \qquad , j \le i\end{aligned} \tag{1.3.3.10}$$

And $\partial\mathbf{J}_i/\partial\theta_j$ is derived as:

$$\begin{aligned}\frac{\partial \mathbf{J}_i}{\partial \theta_j} &= \frac{\partial \mathbf{P}_i}{\partial \theta_j}[\mathbf{J}_i]_i\mathbf{P}_i^T + \mathbf{P}_i[\mathbf{J}_i]_i\left(\frac{\partial \mathbf{P}_i}{\partial \theta_j}\right)^T \\ &= \mathbf{P}_{j-1}\mathbf{e} \times \mathbf{P}_i[\mathbf{J}_i]_i\mathbf{P}_i^T + \mathbf{P}_i[\mathbf{J}_i]_i(\mathbf{P}_{j-1}\mathbf{e} \times \mathbf{P}_i)^T \\ &= \mathbf{P}_{j-1}\mathbf{e} \times \mathbf{J}_i - \mathbf{J}_i \times \mathbf{P}_{j-1}\mathbf{e}\end{aligned} \tag{1.3.3.11}$$

where $\mathbf{J}_i = \mathbf{P}_i [\mathbf{J}_i]_i \mathbf{P}_i^T$ and $[\mathbf{J}_i]_i$ is the inertia matrix of the i-th link about its centroid and expressed in its own coordinates.

By substituting eqs.(1.3.3.10) and (1.3.3.11) into eq.(1.3.3.9) and rearranging the arising equation, we obtain:

$$\begin{aligned}\frac{\partial T}{\partial \theta_j} &= \sum_{i=1}^{n}[m_i \dot{\mathbf{r}}_i \cdot \frac{\partial \dot{\mathbf{r}}_i}{\partial \theta_j} + \mathbf{P}_{j-1}\mathbf{e} \times (\omega_i - \omega_{j-1}) \cdot \mathbf{J}_i \omega_i + \frac{1}{2}\omega_i^T (\mathbf{P}_{j-1}\mathbf{e} \times \mathbf{J}_i - \mathbf{J}_i \times \mathbf{P}_{j-1}\mathbf{e})\omega_i] \\ &= \sum_{i=1}^{n}[m_i \dot{\mathbf{r}}_i \cdot \frac{\partial \dot{\mathbf{r}}_i}{\partial \theta_j} + \omega_{j-1} \times \mathbf{P}_{j-1}\mathbf{e} \cdot (\mathbf{J}_i \omega_i)] \qquad (1.3.3.12)\end{aligned}$$

From eqs.(1.3.3.12) and (1.3.3.8), the following is derived:

$$\frac{d}{dt}(\frac{\partial T}{\partial \dot{\theta}_j}) - \frac{\partial T}{\partial \theta_j} = \sum_{i=1}^{n}[\frac{\partial \mathbf{r}_i}{\partial \theta_j} m_i \cdot \ddot{\mathbf{r}}_i + \mathbf{P}_{j-1}\mathbf{e} \cdot (\omega_i \times \mathbf{J}_i \omega_i + \mathbf{J}_i \dot{\omega}_i)] \qquad (1.3.3.13)$$

By performing the derivative of the potential energy expressed in eq.(1.3.1.2), the last term of the dynamical equation can be obtained as:

$$\frac{\partial W}{\partial \theta_j} = -\mathbf{g}^T (\sum_{i=1}^{n} m_i \frac{\partial \mathbf{r}_i}{\partial \theta_j}) \qquad (1.3.3.14)$$

Hence, the dynamical equation can be written as

$$\begin{aligned}\tau_j = -\tau_j^* &= \frac{d}{dt}(\frac{\partial T}{\partial \dot{\theta}_j}) - \frac{\partial T}{\partial \theta_j} + \frac{\partial W}{\partial \theta_j} \\ &= \sum_{i=1}^{n}[\frac{\partial \mathbf{r}_i}{\partial \theta_j} m_i \cdot \ddot{\mathbf{r}}_i + \mathbf{P}_{j-1}\mathbf{e} \cdot (\omega_i \times \mathbf{J}_i \omega_i + \mathbf{J}_i \dot{\omega}_i)] + \frac{\partial W}{\partial \theta_j} \qquad (1.3.3.15)\end{aligned}$$

In this equation, $\ddot{\mathbf{r}}_i$ can be derived as follows:

$$\begin{aligned}\ddot{\mathbf{r}}_i(\theta) &= \frac{\partial \mathbf{r}_i}{\partial \theta}\ddot{\theta} + [\frac{d}{dt}(\frac{\partial \mathbf{r}_i}{\partial \theta})]\dot{\theta} \\ &= \frac{\partial \mathbf{r}_i}{\partial \theta}\ddot{\theta} + \frac{\partial \dot{\mathbf{r}}_i}{\partial \theta}\dot{\theta} \qquad (1.3.3.16)\end{aligned}$$

where

$$\begin{aligned}\frac{d}{dt}(\frac{\partial \mathbf{r}_i}{\partial \theta})\dot{\theta} &= \frac{d}{dt}(\frac{\partial \mathbf{r}_i}{\partial \theta_1})\dot{\theta} + \dots + \frac{d}{dt}(\frac{\partial \mathbf{r}_i}{\partial \theta_i})\dot{\theta} \\ &= [\frac{\partial^2 \mathbf{r}_i}{\partial \theta_1^2}\dot{\theta}_1^2 + \frac{\partial^2 \mathbf{r}_i}{\partial \theta_1 \partial \theta_2}\dot{\theta}_1\dot{\theta}_2 + \dots + \frac{\partial^2 \mathbf{r}_i}{\partial \theta_1 \partial \theta_i}\dot{\theta}_1\dot{\theta}_i] + \\ &\quad + \dots + [\frac{\partial^2 \mathbf{r}_i}{\partial \theta_1 \partial \theta_2}\dot{\theta}_1\dot{\theta}_i + \dots + \frac{\partial^2 \mathbf{r}_i}{\partial \theta_i^2}\dot{\theta}_i^2] \\ &= \dot{\theta}^T \frac{\partial^2 \mathbf{r}_i}{\partial \theta^2}\dot{\theta} \qquad (1.3.3.17)\end{aligned}$$

From eq. (1.2.2.29) the following is derived:

$$\frac{\partial^2 \mathbf{r}_i}{\partial\theta_j \partial\theta_k} = \mathbf{P}_{k-1}\mathbf{e} \times \mathbf{Q}_k \ldots \mathbf{Q}_{j-1}\mathbf{e} \times [\mathbf{v}_j - \mathbf{Q}_j \ldots \mathbf{Q}_i [\mathbf{s}_{i+1}]_{i+1}] \qquad i > j > k \quad (1.3.3.18a)$$

and

$$\frac{\partial^2 \mathbf{r}_i}{\partial\theta_k \partial\theta_j} = \mathbf{P}_{j-1}\mathbf{e} \times \mathbf{Q}_j \ldots \mathbf{Q}_{k-1}\mathbf{e} \times [\mathbf{v}_k - \mathbf{Q}_k \ldots Q_i [\mathbf{s}_{i+1}]_{i+1}] \qquad i > k \geq j \quad (1.3.3.18b)$$

Up to now all the terms have been derived. Finally, we obtain the following dynamical equation of the robotic manipulator:

$$\begin{aligned}
\tau_j &= \sum_{i=1}^{n} [\frac{\partial \mathbf{r}_i}{\partial\theta_j} m_i \cdot (\frac{\partial \mathbf{r}_i}{\partial\theta})\ddot{\theta} + m_i \frac{\partial \mathbf{r}_i}{\partial\theta_j} \cdot \frac{\partial \dot{\mathbf{r}}_i}{\partial\theta}\dot{\theta} + P_{j-1}\mathbf{e} \cdot (\omega_i \times \mathbf{J}_i\omega_i + \mathbf{J}_i\dot{\mathbf{A}}_i\dot{\theta} + \mathbf{J}_i\mathbf{A}_i\ddot{\theta})] + \frac{\partial W}{\partial\theta_j} \\
&= \sum_{i=1}^{n} [(m_i \frac{\partial \mathbf{r}_i}{\partial\theta_j} \cdot \frac{\partial \mathbf{r}_i}{\partial\theta} + \mathbf{P}_{j-1}\mathbf{e} \cdot \mathbf{J}_i\mathbf{A}_i)\ddot{\theta} + \\
&+ (m_i \frac{\partial \mathbf{r}_i}{\partial\theta_j} \cdot \frac{\partial \dot{\mathbf{r}}_i}{\partial\theta} + \mathbf{P}_{j-1}\mathbf{e} \cdot \mathbf{J}_i\dot{\mathbf{A}}_i + \mathbf{P}_{j-1}\mathbf{e} \times \omega_i \cdot \mathbf{J}_i\mathbf{A}_i)\dot{\theta}] + \frac{\partial W}{\partial\theta_j} \\
&= \mathbf{I}_j(\theta)\ddot{\theta} + \mathbf{L}_j(\theta,\dot{\theta})\dot{\theta} + d_j(\theta) \qquad (1.3.3.19)
\end{aligned}$$

Therefore, τ, which is a torque vector can be expressed as:

$$\tau = \mathbf{I}(\theta)\ddot{\theta} + \mathbf{L}(\theta,\dot{\theta})\dot{\theta} + \mathbf{d}(\theta) \qquad (1.3.3.20)$$

In above two equations, $\mathbf{I}(\theta)$ is the $n \times n$ matrix of generalized inertia, $\mathbf{L}(\theta,\dot{\theta})$ is a $n \times n$ matrix and $\mathbf{d}(\theta)$ is a n-dimensional represents gravity forces. $\mathbf{I}_j(\theta)$ and $\mathbf{L}_j(\theta,\dot{\theta})$ are the j-th row of matrices $\mathbf{I}(\theta)$ and $\mathbf{L}(\theta,\dot{\theta})$, respectively, while $d_j(\theta)$ is the j-th component of vector $\mathbf{d}(\theta)$.

Now the last two terms of eq.(1.3.3.20), which are nonlinear in θ and $\dot{\theta}$, are written as a single vector, which is a function of θ and $\dot{\theta}$, $\mathbf{h}(\theta,\dot{\theta})$. Thus, we derive the dynamical model of the robotic manipulators as followings:

$$\tau = \mathbf{I}(\theta)\ddot{\theta} + \mathbf{h}(\theta,\dot{\theta}) \qquad (1.3.3.21)$$

where τ is the n-dimensional vector of generalized force grouping the motor torques and forces at the joints and $\mathbf{h}(\theta,\dot{\theta})$ is a n-dimensional vector describing the inertia terms that are quadratic in $\dot{\theta}$, as well as the viscous and gravity terms. Eq.(1.3.3.21) represents the general form of the coupled nonlinear differential equations of the robotic manipulator.

Chapter 2 Feedback Control of Robotic Manipulators Using Model Assignment

2.1 Introduction

There are two major manners in the system control, namely, feedback or closed-loop control and open-loop control. A typical control system has the feature that some output quantities are measured and compared with the desired output values, and the system's output is corrected by using the resulting errors from the comparison. Such a kind of system is called feedback control system. A block diagram of the feedback control system is shown in Figure 2.1.

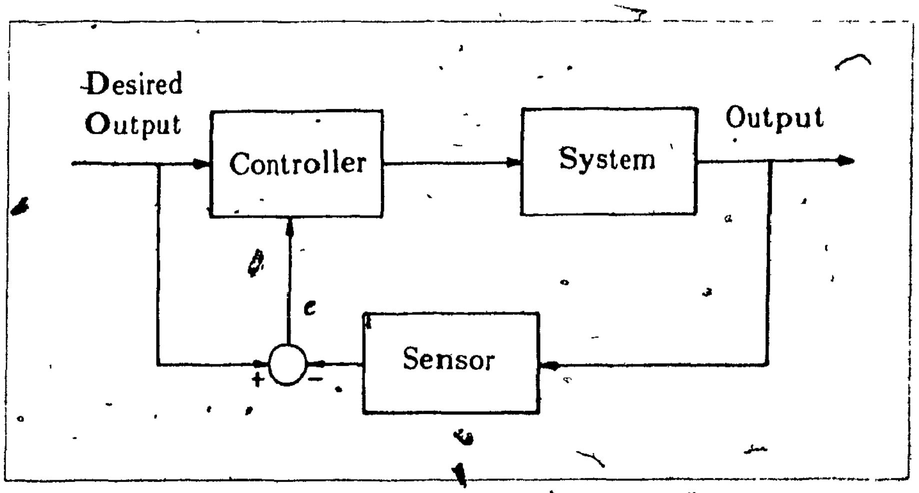

Figure 2.1 Feedback Control System

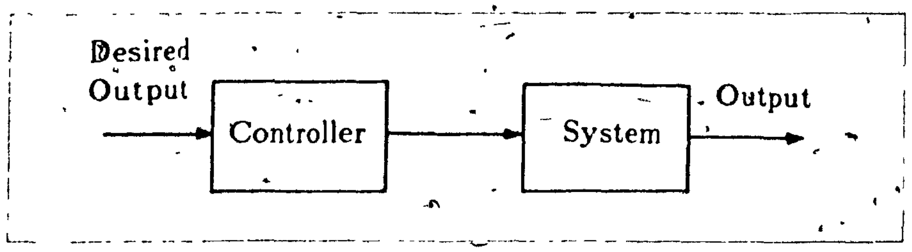

Figure 2.2 Open-loop Control System

In some cases, it is also possible to control the system in an open-loop manner. Such a kind of system is shown in Figure 2.2

Feedback control is widely used in robotic systems The reason is that, by using feedback, the designer of the robotic systems is often able to use inexpensive and inaccurate components, while the system is capable of achieving precise control in the presence of measuring errors and unpredictable disturbances.

Since robotic manipulator systems are highly nonlinear, either a complicated control scheme must be designed or linearization must be performed in order to determine the suitable control

Based on the linearized model, various feedback schemes have been proposed. Among these, pole-assignment is the most popular because it is simple to use and to implement. However, the problem of the pole-assignment scheme is that the solutions ~~for each~~ set of desired poles are infinitely many, and this scheme applies only to linear time-invariant systems. Complicated algorithms have to be derived to find a set of optimal gains for pole-assignment problem. In recent years, Bhattacharyya presented a series of papers on this topic using Sylvester's equation(1983) Comparatively, the forgoing schemes are rather complicated and time-consuming because they need too many computations to obtain the solutions at each sampling step.

In order to perform real-time control of robotic manipulators, it is important to reduce the computations of the control algorithm In this chapter, a method called *model assignment*, which requires less computations, is used instead of the pole-assignment scheme. The highly coupled nonlinear dynamical equations of the manipulator are linearized about a nominal trajectory first and then, the linearized system is rendered a prescribed second-order system using cancelling scheme. The idea of cancelling scheme is described in the next section.

2.2 Feedback Control of Dynamical System Using Cancelling Scheme

The dynamical equation of a nonlinear system can be expressed as:

$$m\ddot{x} + f(\dot{x}, x) = \tau \tag{2.2.1}$$

where $x, \dot{x}, \ddot{x}$ are the actual position, velocity and acceleration, respectively

By introducing a nonlinear control term, a control scheme called linearizing control law is used to cancel a nonlinearity in the controlled system. Such that the overall closed-loop system is linear.

The model-based portion of the control is:

$$\tau = a\tau' + b \tag{2.2.2}$$

where

$$\begin{aligned} a &= m \\ b &= f(x, \dot{x}) \end{aligned} \tag{2.2.3}$$

and the servo portion is

$$\tau' = \ddot{x}_d + k_v\dot{e} + k_p e \tag{2.2.4}$$

This leads to the system equation of motion written in error-space as:

$$\ddot{e} + \frac{k_v}{m}e + \frac{k_p}{m}e = 0 \tag{2.2.5}$$

where x_d is the desired position and e is the servo error, defined as $x_d - x$. The feedback gains k_v and k_p are adjusted to obtain a good control effort and a stable system performance.

Let $k'_v = k_v/m$ and $k'_p = k_p/m$. Then, eq (2.2.5) can be written as:

$$\ddot{e} + k'_v \dot{e} + k'_p e = 0 \tag{2.2.6}$$

The characteristic equation of eq (2.2.6) is:

$$s^2 + k_v s + k_p = 0 \tag{2.2.7}$$

which has the roots

$$s_1 = -\frac{k'_v}{2} + \frac{\sqrt{k'^2_v - 4k'_p}}{2} \tag{2.2.8}$$

$$s_2 = -\frac{k'_v}{2} - \frac{\sqrt{k'^2_v - 4k'_p}}{2} \tag{2.2.9}$$

These roots are called the poles of the closed-loop system By chosing the locations of the poles on the imaginary plane we can make the system stable and with good dynamical characteristics. The block diagram of such a system is shown in Figure 2 3

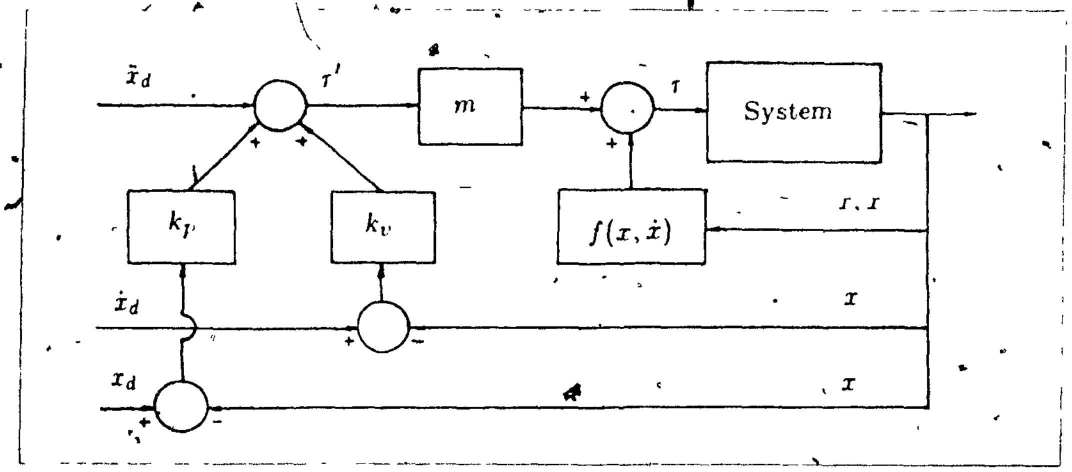

Figure 2.3 Control System Using Cancelling Scheme

2.3 Linearization of the Dynamical Model

In the last section, the cancelling scheme is performed by designing a nonlinear feedback. In order to introduce model-assignment scheme based on a linear feedback, the dynamical model of the robotic system is linearized first. Then, the so called model-assigment control is derived in the next section.

The general expression for the equation of motion of a robot manipulator is readily derived in Chapter 1 as

$$\mathbf{I}(\theta)\ddot{\theta} + \mathbf{h}(\theta, \dot{\theta}) = \tau \tag{2.3.1}$$

where $\dot{\theta}, \ddot{\theta}$ are, respectively, the first and second time derivatives of joint variable vector θ, $\mathbf{h}(\theta, \dot{\theta})$ being a n-dimentional vector that comprises the inertia terms that are quadratic in $\dot{\theta}$, as well as the dissipation and gravity terms. Eq (2.3.1) represents the general form of the coupled nonlinear differential equations of the robot manipulator.

The dynamical model of the robot manipulator is linearized about the desired trajectory followed. This is done by using the Taylor series expansion of both sides of eq (2.3.1). Let $\theta_d(t)$, $\dot{\theta}_d(t)$ and $\ddot{\theta}_d(t)$ denote the time histories of vector $\theta(t)$ along the desired trajectory, which are obtained from inverse kinematics. The higher-order terms are assumed to be negligible and the linearized perturbation equation is obtained as follows:

$$\mathbf{W}(\theta_d)\delta\ddot{\theta} + \mathbf{C}(\theta_d, \dot{\theta}_d)\delta\dot{\theta} + \mathbf{D}(\theta_d, \dot{\theta}_d, \ddot{\theta}_d)\delta\theta = \delta\tau(t) \tag{2.3.2}$$

where

$$\mathbf{W}(\theta_d) = \left.\frac{\partial\,\mathbf{I}(\theta)\ddot{\theta}}{\partial\ddot{\theta}}\right|_{\theta_d} = \mathbf{I}(\theta_d) \tag{2.3.2a}$$

$$\mathbf{C}(\theta_d, \dot{\theta}_d) = \left.\frac{\partial\mathbf{h}(\theta, \dot{\theta})}{\partial\dot{\theta}}\right|_{\theta_d, \dot{\theta}_d} \tag{2.3.2b}$$

$$\mathbf{D}(\theta_d, \dot{\theta}_d, \ddot{\theta}_d) = \left.\frac{\partial\mathbf{h}(\theta, \dot{\theta})}{\partial\theta}\right|_{\theta_d, \dot{\theta}_d} + \left.\frac{\partial\mathbf{I}(\theta)}{\partial\theta}\right|_{\theta_d}\ddot{\theta}_d \tag{2.3.2c}$$

Now the robot manipulator consists of feedforward control from trajectory planning, and feedback control from the actual trajectory. The feedforward torques can be obtained directly from the dynamical model of the robot manipulator. The input of this part of the system is the desired position, velocity and acceleration histories,

$\theta_d(t); \dot\theta_d(t), \ddot\theta_d(t)$, which are obtained from inverse kinematics, and the output is the desired torques $\tau_d(t)$

Thus, the control problem is reduced to determining a suitable $\delta\tau(t)$ which drives $\delta\theta(t)$ to zero. The design of the controller is performed in the next section.

2.4 Design of the Controller Using Model Assignment

In Section 2.3, the linearized perturbation equation is obtained. Since $\theta_d, \dot\theta_d, \ddot\theta_d$ are all functions of time, eq.(2.3.1) can be written as

$$\mathbf{I}(t)\delta\ddot\theta + \mathbf{C}(t)\delta\dot\theta + \mathbf{D}(t)\delta\theta = \delta\tau(t) \tag{2.4.1}$$

Now we want to find a control scheme that renders the linearized equation a prescribed second-order stationary system of the form:

$$\delta\ddot\theta + 2\mathbf{\Psi}^T\mathbf{\Omega_n}\mathbf{\Psi}\delta\dot\theta + \mathbf{\Omega_n}^2\delta\theta = 0 \tag{2.4.2}$$

where $\mathbf{\Psi}$ is the *nondimensional* $n \times n$ *damping matrix* and $\mathbf{\Omega_n}$ is the $n \times n$ *matrix of undamped natrual frequency* of the system. Notice that, in this formulation, rather than attempting to stablize the linearized model by pole assignment, its stabilization is attempted by *model assignment*. For a stable performance, $\mathbf{\Omega}_n$ will be chosen as positive definite. Moreover, for simplicity, $\mathbf{\Psi}$ and $\mathbf{\Omega}_n$ can be chosen as diagonal. Let

$$\delta\tau(t) = -\mathbf{K}_p(t)\delta\theta - \mathbf{K}_d(t)\delta\dot\theta \tag{2.4.3}$$

where $\mathbf{K}_p(t)$ and $\mathbf{K}_d(t)$ are two $n \times n$ time-varying feedback matrices.

Hence, eq.(2.4.1) can be written as

$$\delta\ddot\theta + \mathbf{I}(t)^{-1}[\mathbf{C}(t) + \mathbf{K}_d(t)]\delta\dot\theta + \mathbf{I}(t)^{-1}[\mathbf{D}(t) + \mathbf{K}_p(t)]\delta\theta = 0 \tag{2.4.4}$$

Comparing eq.(2.4.2) with eq.(2.4.4), the following is derived:

$$\mathbf{K}_d(t) = 2\mathbf{I}(t)\mathbf{\Psi}\mathbf{\Omega_n}\mathbf{\Psi} - \mathbf{C}(t) \tag{2.4.5}$$

$$\mathbf{K}_p(t) = \mathbf{I}(t)\mathbf{\Omega}_n^2 + \mathbf{D}(t) \tag{2.4.6}$$

Hence, the time-varying feedback matrices $\mathbf{K}_p(t)$ and $\mathbf{K}_d(t)$ are obtained for the desired control effort. Such a control scheme could be called *linearized cancelling control*, since it uses a time-varying control term to "cancel" the time-varying terms in the linearized controlled system such that the overall closed-loop system is time invariant. Compare with Craig's cancelling scheme(1986), this scheme has the advantage of simplicity in controller design.

However, both $\mathbf{K}_p(t)$and $\mathbf{K}_d(t)$ depend on the parameters of the system. When the precise parameters of the linearized equation are known, by using this method, we can control the system perfectly. In reality, the parameters of the system are not known perfectly, because of measurement and modelling errors and changes of the environment. An estimation scheme has to be derived in order to obtain accurate values of the parameters. A new approach of adaptive control using controller estimation is derived to solve this problem. The details are described in the next two chapters.

2.5 Example

In this example, a Cincinnati Milacron robot whose Hartenberg-Denavit parameters and inertia parameters are shown in Table 2.1 and 2.2, respectively, is used to illustrate the procedure.

The end effector of the manipulator is to move along a straight line in the Cartesian space as described by the following equations

$$\begin{aligned} x &= 1.33125\text{m}, & \dot{x} &= 0.0\text{ms}^{-1}, & \ddot{x} &= 0.0\text{ms}^{-2}, \\ y &= \beta\text{m}, & \dot{y} &= \dot{\beta}\text{ms}^{-1}, & \ddot{y} &= \ddot{\beta}\text{ms}^{-2}, \\ z &= 1.79875\text{m}, & \dot{z} &= 0.0\text{ms}^{-1}, & \ddot{z} &= 0.0\text{ms}^{-2}, \end{aligned} \tag{2.5.1}$$

where β is a prescribed function of time defined with cubic splines so as to meet the following end conditions:

$$\begin{aligned} \beta(0) &= 0, \quad \beta(T) = 1\mathrm{m} \\ \dot\beta(0) &= \dot\beta(T) = 0 \\ \ddot\beta(0) &= \ddot\beta(T) = 0 \\ T &= 1\mathrm{s} \end{aligned} \tag{2.5.2}$$

and T is the time it takes to traverse the trajectory. Details on how to construct such a function were derived by Angeles(1986) The parameter matrices of the linearized model of the manipulator at $t = 0.05$s are the following

$$\mathbf{W}(\theta_d) = \begin{pmatrix} 335.8 & 2.8 & 8.5 & 0.1 & 35.6 & -0.2 \\ 2.8 & 474.6 & 210.1 & 87.1 & 3.0 & 55.9 \\ 8.5 & 210.1 & 351.0 & 81.5 & 7.2 & 56.0 \\ 0.1 & 87.1 & 81.5 & 47.8 & 0.0 & 56.1 \\ 35.6 & 3.0 & 7.2 & 0.0 & 12.5 & 0.0 \\ 0.2 & 55.9 & 56.0 & 56.1 & 0.0 & 581.6 \end{pmatrix} \mathrm{Nms}^2 \tag{2.5.3}$$

$$\mathbf{C}(\theta_d, \dot\theta_d) = \begin{pmatrix} 437.6 & 452.9 & 1021.7 & 1307.1 & 518.1 & -257.1 \\ -137.0 & 287.5 & 347.3 & -807.9 & -329.6 & 2589.7 \\ 398.6 & 160.8 & 107.2 & -555.6 & -74.3 & 2213.2 \\ 48.1 & 56.5 & 62.1 & 214.2 & 54.6 & 745.3 \\ -60.8 & -46.7 & 111.7 & -138.0 & -59.1 & 15.4 \\ 36.4 & 37.1 & 40.4 & -166.9 & -37.6 & 584.6 \end{pmatrix} \mathrm{Nms} \tag{2.5.4}$$

$$\mathbf{D}(\theta_d, \dot\theta_d, \ddot\theta_d) = \begin{pmatrix} 51.2 & 33.5 & -14.5 & -179.5 & 992.7 & 40.9 \\ -43.0 & 3326.8 & 5808.1 & 6335.4 & 1983.7 & -3889.6 \\ 503.2 & 515.9 & -5072.2 & 10662.0 & 4625.7 & -1464.8 \\ 108.9 & 449.8 & -1468.2 & 2483.4 & 1271.8 & -1389.8 \\ 20.7 & 44.6 & 31.6 & -121.9 & -1978.5 & -2.8 \\ 107.5 & 273.6 & -963.4 & 1550.7 & 1158.67 & -1149.1 \end{pmatrix} \mathrm{Nm} \tag{2.5.5}$$

The prescribed second-order stationary system is designed by chosing matrices $\mathbf{\Psi}$ and $\mathbf{\Omega_n}$ to be diagonal Therefore, the prescribed linearized dynamical model for each joint can expressed as

$$\delta\ddot\theta_i + 2\zeta_i\omega_{ni}\delta\dot\theta_i + \omega_{ni}^2\delta\theta_i = 0, \qquad i = 1, \dots, 6 \tag{2.5.6}$$

where $\varsigma_i = \psi_i^2$. The correspending charateristic equation is:

$$s^2 + 2\varsigma_i \omega_{ni} s + \omega_{ni}^2 = 0, \qquad i = 1, ..., 6 \tag{2.5.7}$$

From eq.(2.5.7), we get the poles as

$$s = -\varsigma_i \omega_{ni} \pm j\omega_{ni}\sqrt{1 - \varsigma_i^2}, \qquad i = 1, ..., 6 \tag{2.5.8}$$

The poles for each equation are chosen to be located at $(-1, j)$ and $(-1, -j)$. Thus,

$$\omega_{ni} = \sqrt{2}, \qquad \varsigma_i = \frac{\sqrt{2}}{2} \tag{2.5.9}$$

Hence,

$$\Psi = \text{diag}(\frac{\sqrt[4]{8}}{2}, ..., \frac{\sqrt[4]{8}}{2}) \tag{2.5.10}$$

and

$$\mathbf{\Omega_n} = \text{diag}(\sqrt{2}, ..., \sqrt{2}) \tag{2.5.11}$$

From eqs.(2.5.4),(2.5.5),(2.5.6)and (2.5.10),(2.5.11), the feedback gains are obtained as:

$$\mathbf{K}_p(t) = \begin{pmatrix} -722.9 & -39.2 & -2.4 & 179.5 & 992.7 & 40.8 \\ 37.3 & -4275.9 & 5387.9 & 6509.7 & 1983.7 & 3889.6 \\ -520.2 & -936.1 & 4370.3 & 10825.0 & 4611.3 & 4352.8 \\ -108.7 & -634.1 & 1305.2 & 2479.2 & 1271.7 & 1277.6 \\ 91.8 & 50.6 & -17.2 & 122.0 & 172.9 & 27.7 \\ -107.1 & -385.5 & 851.4 & 1662.9 & 1158.6 & 867.5 \end{pmatrix} \text{Nms} \tag{2.5.12}$$

and

$$\mathbf{K}_d(t) = \begin{pmatrix} 1109.3 & -458.6 & 1007.8 & -1307.1 & -447.2 & 257.4 \\ 131.4 & -1236.6 & -767.2 & 633.6 & 323.6 & 2701.7 \\ -415.5 & -580.9 & 809.1 & 382.6 & 88.7 & 2325.2 \\ -47.9 & -230.8 & -225.1 & 118.4 & 54.7 & 857.6 \\ 132.0 & 40.7 & 97.4 & 138.1 & 34.2 & 15.4 \\ -36.0 & -149.0 & 152.4 & 54.6 & 37.5 & 866.2 \end{pmatrix} \text{Nm} \tag{2.5.13}$$

Table 2.1 Hartenberg-Denavit Parameters of the Cincinnati Milacron Robot

joint number	a_i (m)	b_i (m)	α_i (degrees)	initial θ_i (degrees)
1	0.00	1.50	90.00	0.00
2	1.02	0.00	0 00	90 00
3	~~1.02~~	~~0.00~~	~~0.00~~	-135.00
4	0.20	0.00	90.00	45.00
5	0.00	0.00	90.00	90 00
6	0.00	0.41	90.00	90 00

Table 2.2 Inertia Parameters of the Cincinnati Milacron Robot

item	joint 1	joint 2	joint 3	joint 4	joint 5	joint 6
$m_i\,(kg)$	680.0	360.0	180.0	55.0	36.0	68.0
$x_{ci}\,(m)$	0.0	-0.87	-0.64	-0.12	0.0	0.0
$y_{ci}\,(m)$	-0.33	0.0	0.04	0.04	-0.05	0 0
$z_{ci}\,(m)$	0.0	-0.13	0.0	0.0	-0.08	0.0
$I_{xxi}\,(kgm^2)$	0.0	11.0	1.10	0.44	0.47	0.44
$I_{xyi}\,(kgm^2)$	0 0	0.0	0.0	0.0	0.0	0 0
$I_{xzi}\,(kgm^2)$	0.0	0.0	0.0	0.0	0.0	0.0
$I_{yyi}\,(kgm^2)$	62 0	53.0	44.0	0 91	0.38	0.64
$I_{yzi}\,(kgm^2)$	0 0	0.0	0.0	0.0	0.0	0 0
$I_{zzi}\,(kgm^2)$	0 0	44.0	44.0	0.82	0.18	0.73

m_i	mass of the i-th link.
x_{ci}, y_{ci}, z_{ci}	coordinates of the mass center of the i-th link.
$I_{xxi}, I_{xyi}, I_{xzi}$	
$I_{yyi}, I_{yzi}, I_{zzi}$	entries of the inertia matrix of the i-th link

Chapter 3 Adaptive Control of Robotic Manipulators

3.1 Introduction

In recent years, adaptive control has been widely used for robotic systems. Koivo and Guo(1983) proposed an adaptive self-tuning controller using an autoregressive model to fit the input and output data of the robotic system Lee et al.(1984) proposed another adaptive control based on the linearized perturbation equation in the vicinity of a manipulator trajectory The main idea of this scheme will be described in this chapter. Many other methods have also been used for adaptive control. For example, Seraji(1986) derived a direct adaptive control scheme in Cartesian space using the Lyapunov method.

An adaptive control system is required when a convetional controller cannot work. In our case, this happens because of

(a) Simplification introduced in the dynamical modelling of the robotic manipulators

(b) Disturbances to which the system is subject

Obviously when any of the above is non-negligible, conventional feedback control is no longer indifferent to variations and the robot will produce unacceptable errors in tracking the trajectory either in Cartesian or in joint space Early attemps to solve these problems involved the design of nonlinear control systems, which in the simplest case

case only program a deliberate adjustment of some system quantity, usually loop gain, to compensate for a change of system parameters caused by changes of the environment. The simple structure of such a system is shown in Figure 3.1. In this system the difference between the actual and the desired responses is used to supplement the input signal to achieve a better result. Such kind of systems have often been successful over a restricted range of operating conditions. Outside this range the system must be adapted to fit the new conditions.

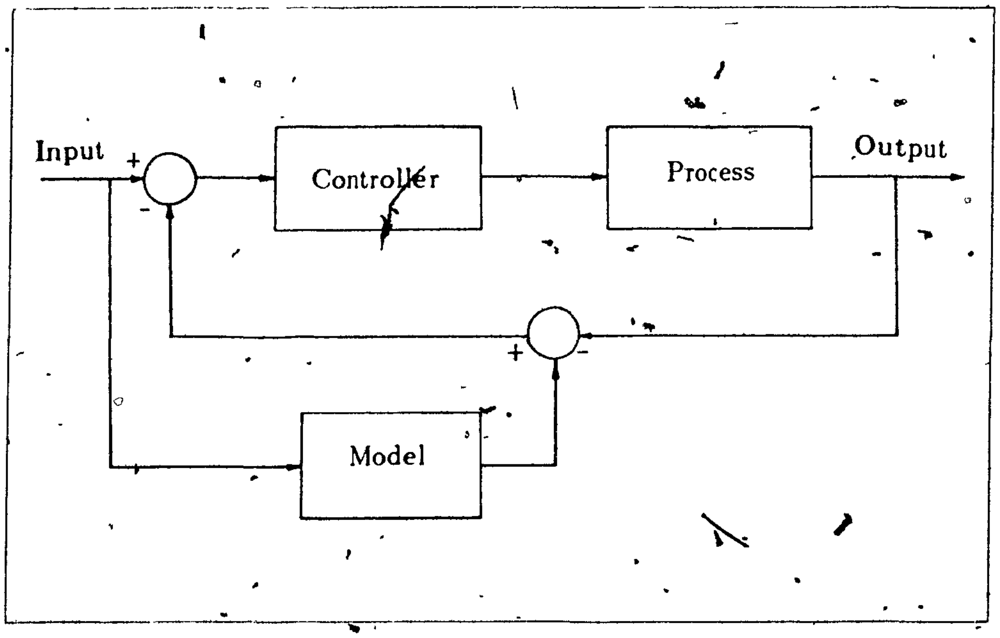

Figure 3.1 Nonlinear Control

In these cases, the system is required to change its own compensation and this leads to a requirement of adaptive control The block diagram of an adaptive control system is shown in Figure 3 2 In this system, the control can be thought of as being the combination of two closed loops The inner loop consists of the process and an ordinary linear feedback regulator The parameters of the system are adjusted by the outer loop, which is composed of a recursive parameter-estimation and a design calculation. The design calculation box in Figure 3 2 represents an on-line solution to a design problem for a system with known parameters.

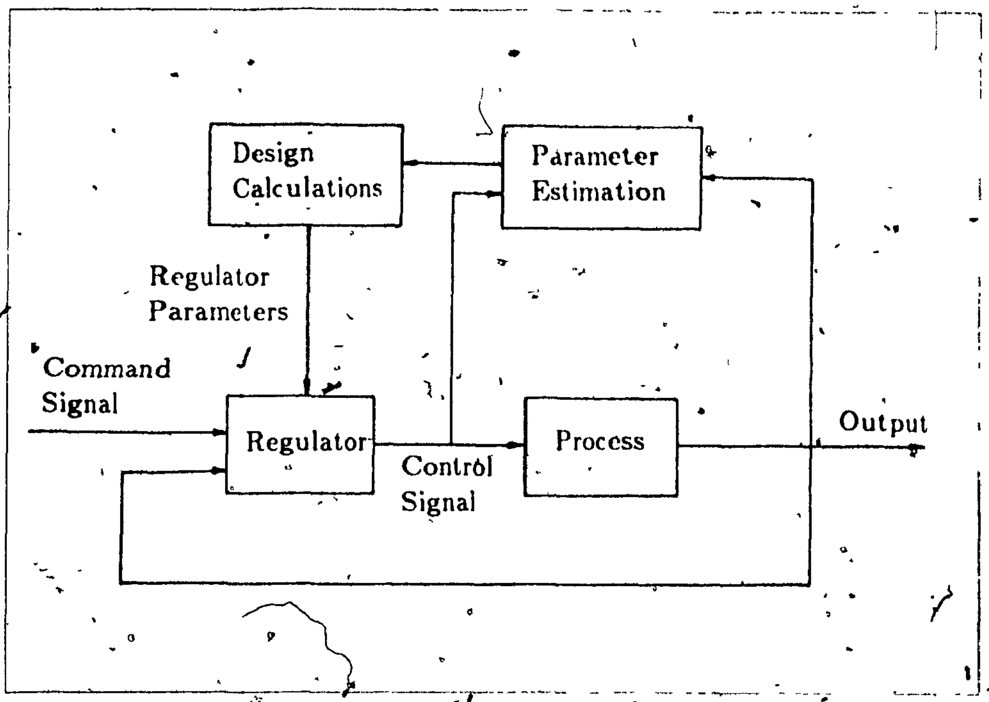

Figure 3.2 Adaptive Control System

There are always at least two elements that appear in some form in adaptive control systems, namely, identification and actuation Identification refers to the measurement of the dynamic parameters of the process to be controlled, whereas, actuation is the generation of an appropriate actuation signal

The identification problem is the most important aspect of adaptive control and becomes the central element of such a system because a good adaptability of a system means a frequent and rapid solution of the identification system.

Since the identification problem is so important to the adaptive control system, it will be discussed in more detail in the next section.

3.2 System Identification

The task of system-identification is usually referred to as the determination of a mathematical model for a system or a process by observing the relationships of the input and the output of the system.

The system model is a set of mathematical equations which relates the input and the output of the system. By investigating the system's responses under the condition of a variety of inputs, we can derive such a model.

Generally speaking, we can divide the system-identification problem into two categories:

(a) The complete identification problem: This means that we do not know anything about the basic properties of the system, such as whether it is linear or nonlinear and so on. Obviously, this is an extremely difficulty problem to solve. For such a kind of a problem, some assumptions are usually made before a meaningful solution can be attempted.

(b) Partial identification problem: This means that we know some of the basic characteristics of the system. However, we may not know some or all of the parameters of the system. In this case, it is easier for us to deal with such a kind of problem than the first one.

In robotics systems, we know a good deal about the structure of the system, so that it is possible for us to derive a specific mathematical model of the robot dynamic system. That is exactly what has been done in the Chapter 1. But, as mentioned in Chapter 2, in reality, because of measurement and modelling errors and changes of the environment, the characteristics of the system are not known perfectly. So system-identification has to be introduced in order to solve the problem.

Fortunately, in our case, only a set of parameters in the model are left to be determined because we know the structure of the system, the system-identification problem is reduced to that of parameter-identification.

Since a majority of system-identification problems can be reduced to the aforementioned parameter-identification problem, the treatment of the parameter-estimation scheme is considered to be of greatest importance. The block diagram of a typical

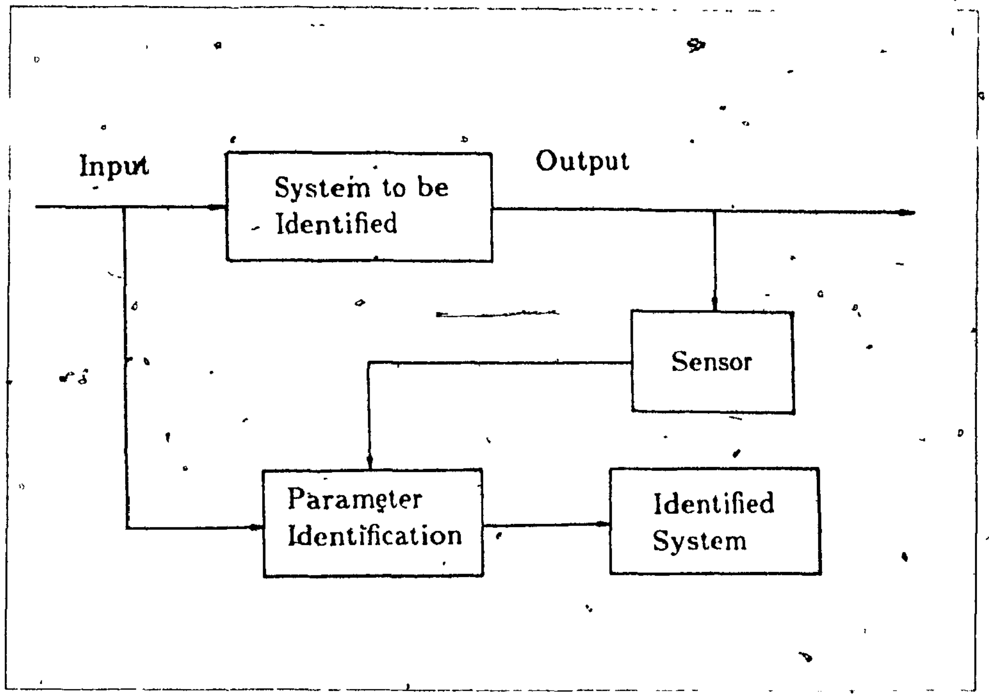

Figure 3.3 Parameter-Identification

parameter-identification system is shown in Figure 3.3

The input and the output of the system are measured by sensors. These measurement are used as the information of the parameter-identification scheme. Then, the parameter-identification is performed to select the model in the specified class that best fits the statistical data.

The study of an adaptive control scheme which is based on the linearized pertubation equation in the vicinity of a nominal trajectory will be disscused in detail in the next section. In this scheme, the parameters and the feedback gains of the linearized system are updated and adjusted at each sampling period to obtain the satisfied control effort. In this way, the total torques applied to the actuators of the robot joints consists of the nominal torques and the variational torques. The nominal torques are computed from the theoretical model which has been derived in Chapter 1 and the variational torques are computed from the one-step optimal control law of the linearized system.

3.3 Parameter Identification of the Perturbation Equation of Robotic Manipulators

The general dynamical equations of the motion of a n-link manipulator are a set of second-order, highly coupled nonlinear differential equations. These can be written as

$$\mathbf{I}(\theta)\ddot{\theta} + \mathbf{h}(\theta,\dot{\theta}) = \tau(t) \tag{3.3.1}$$

By defining a n-dimensional state vector for the system as:

$$\begin{aligned} \mathbf{x}^T(t) &= (x_1, x_2, ..., x_{2n}) \\ &= (\theta_1, \theta_2, ..., \theta_n, \dot{\theta}_1, \dot{\theta}_2, ..., \dot{\theta}_n) \end{aligned} \tag{3.3.2}$$

and a n-dimensional input vector as:

$$\tau^T(t) = (\tau_1, \tau_2, ..., \tau_n) \tag{3.3.3}$$

Eq.(3.3.1) can be written in the state space as:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \tau(t)) \tag{3.3.4}$$

The right hand side of eq.(3.3.4) is a nonlinear $2n$-dimensional vector-valued function.

Then, the linearized perturbation equation can be obtained by introducing the Taylor series expansion of eq.(3.3.4). It is assumed that the higher order terms are negligible. Then the associated perturbation equations for this control system can be written as:

$$\begin{aligned} \delta\dot{\mathbf{x}}(t) &= \nabla_x f|_d \delta\mathbf{x}(t) + \nabla_\tau f|_d \delta\tau(t) \\ &= \mathbf{X}(t)\delta\mathbf{x}(t) + \mathbf{Y}(t)\delta\tau(t) \end{aligned} \tag{3.3.5}$$

where $\nabla_x f|_d$ and $\nabla_\tau f|_d$ are the Jacobian matrices of $f(\mathbf{x}(t))$ with respect to x and τ, respectively, evaluated at desired state $\mathbf{x}_d$ and input τ_d. $\delta\mathbf{x}(t)$ and $\delta\tau(t)$ are the state error and servo input, respectively, i.e.,

$$\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_d(t) \tag{3.3.6}$$

$$\delta\tau(t) = \tau(t) - \tau_d(t) \tag{3.3.7}$$

The parameters of the perturbation equation, eq.(3.3.5), $\mathbf{X}(t)$ and $\mathbf{Y}(t)$, are time varying. They depend on the position and the velocity of the manipulator at the current instant. In order to get the ideal controller to render the system a prescribed system, matrices $\mathbf{X}(t)$ and $\mathbf{Y}(t)$ must be known all the time along the trajectory. The problem arises because of the complexity of the system, noises introduced to the system during the operation and errors in the measurements and calculations. In fact, the actual parameters of the system are unknown. It is necessary to find an estimation technique to identify the unknown parameters of the system. One method consists of estimating the parameters of this perturbation equation. The diagram of such a control scheme is shown in Figure 3.4.

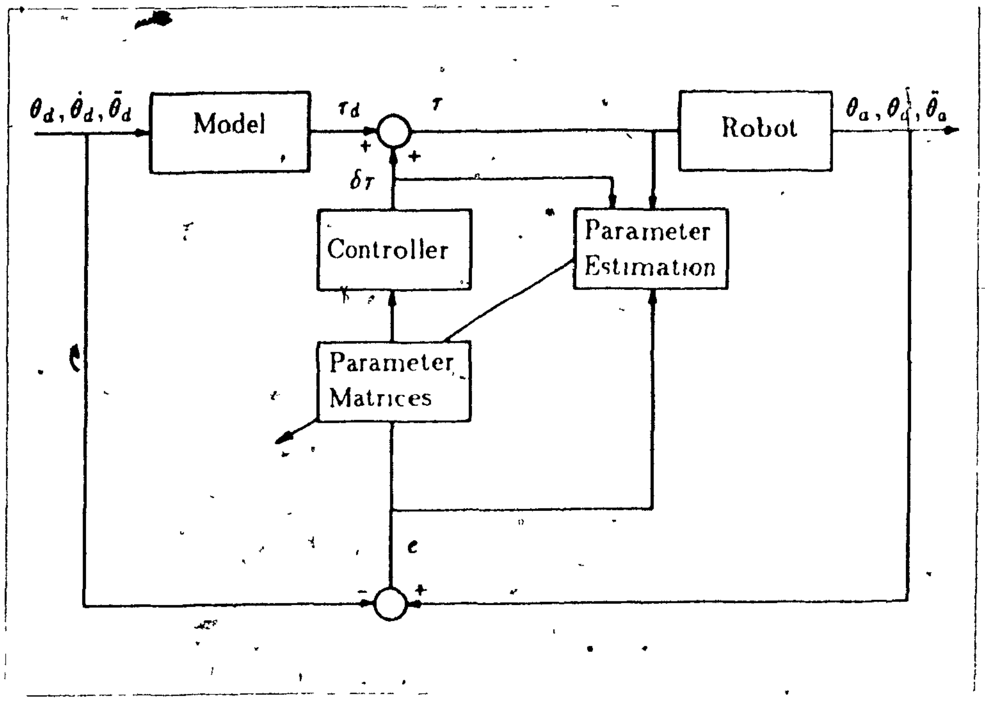

Figure 3.4 Adaptive Control with Model Estimation

In this scheme, the overall control problem is reduced to finding suitable variational torques $\delta\tau$ which drive the errors of the system $\delta\mathbf{x}(t)$ to zero.

The input of the system, namely, the torque for each joint now consists of two components, one is the feedforward component obtained from inverse dynamics, namely, the nominal torques obtained from the theoritical model of the system, and another one is the feedback component which is calculated from the feedback control law

In order to estimate the unknown parameters of the perturbation equations, the discretized equation corresponding to eq (3.3.5) is written as

$$\mathbf{x}(k+1) = \mathbf{R}(k)\mathbf{x}(k) + \mathbf{S}(k)\tau(k) \qquad k = 0, 1, \ldots\ldots \tag{3.3.8}$$

where k means at the k-th sampling step, $\mathbf{x}(k)$ is a $2n$-dimensional state vector given by:

$$\mathbf{x}(k) = \mathbf{\Gamma}(k, t_o)\mathbf{x}(t_0) + \int_{t_o}^{k} \mathbf{\Gamma}(k,t)\mathbf{Y}(t)\tau(t)dt \tag{3.3.9}$$

where $\tau(t)$ is a constant control input vector between k and $(k+1)$-st step and $\mathbf{\Gamma}(k, t_0)$ is the state-transition matrix of the system. Matrices $\mathbf{R}(k)$ and $\mathbf{S}(k)$ are next defined as:

$$\mathbf{R}(k) = \mathbf{\Gamma}(k+1, k) \tag{3.3.10}$$

and

$$\mathbf{S}(k)\tau(k) = \int_{k}^{k+1} \mathbf{\Gamma}(k+1, t)\mathbf{Y}(t)\tau(t)dt \tag{3.3.11}$$

Now, the matrices to be estimated here are $\mathbf{R}(k)$ and $\mathbf{S}(k)$. For its simplicity and ease of application, a recursive least-square parameter-identification scheme is used to perform the estimation of $\mathbf{P}(k)$ and $\mathbf{S}(k)$.

Before this least-square scheme can be used, the system equations, eq (3.3.8), have to be rearranged as

$$\mathbf{x}_i(k+1) = \phi^T(k)\mathbf{z}_i(k) \qquad i = 1, 2, \ldots, 2n \tag{3.3.12}$$

where $\mathbf{z}_i(k)$ is a vector of the i-th row of the unknown parameter matrices, namely

$$\mathbf{z}_i(k) = (r_{i1}(k), \ldots, r_{i2n}(k), s_{i1}(k), \ldots, s_{in}(k)) \qquad i = 1, \ldots, 2n \tag{3.3.13}$$

and vector $\phi(k)$ contains the states and the inputs of the system, i.e.,

$$\phi^T(k) = (x_1(k), \ldots, x_{2n}(k), u_1(k), \ldots, u_n(k)) \tag{3.3.11}$$

Based on eq (3.3.12), a recursive least-square parameter-identification scheme can be found that minimizes an error criterion, which is quadratic in the error. The unknown parameters can be estimated by using the following equations

$$\tilde{\mathbf{z}}_i(k+1) = \tilde{\mathbf{z}}_i(k) + r(k+1)\mathbf{P}(k)\phi(k+1)[\mathbf{x}_i(k+1) - \phi^T(k+1)\tilde{\mathbf{z}}_i(k)] \tag{3.3.13}$$

$$\mathbf{P}(k+1) = \mathbf{P}(k) - r(k+1)\mathbf{P}(k)\phi(k+1)\phi^T(k+1)\mathbf{P}(k) \tag{3.3.14}$$

$$r(k+1) = [\phi^T(k+1)\mathbf{P}(k)\phi(k+1) + \rho]^{-1} \tag{3.3.15}$$

where ρ is the "forgetting factor", which is defined between 0 and 1. ρ is usually chosen to be $0.90 \leq \rho \leq 0.98$.

After the parameters in $\mathbf{R}(k)$ and $\mathbf{S}(k)$ are obtained, there are many ways of finding the proper control law. For instance, Riccati equation is widely used.

The main idea of above adaptive control scheme was derived by Lee et al (1984). However, the computations of this scheme are quite large. As we know, it is very important to reduce the computations in the robot control scheme. So, we have to keep on finding new adaptive algorithms which need less computations. A new approach of adaptive control using controller estimation is proposed. The details of least-square estimation, discretization of dynamical systems and controller estimation scheme are described in the next chapter.

Chapter 4 Controller Estimation—A New Approach of Robotic Adaptive Control

4.1 Introduction

Many approaches to the control of robotic manipulators are based on linear time-varying compensation schemes In order to maintain good performance over a wide range of motions and payloads, adaptive control methods are used As described in Chapter 3, the aim of adaptive control techniques for robotic systems is to estimate the parameters of the models recursively, and then use the estimates to calculate the control law by a suitable design method

In this chapter, a new method of obtaining the controllers for the adaptive control system of robotic manipulators is proposed Based on the linearized model, a linear time-varying controller has been designed to render the linearized system a stationary closed-loop second order system of a prescribed form in Chapter 2. Now, a recursive least-square identification scheme is used to estimate the parameters in the controller. The feedback gains of the controller are updated and adjusted in each sampling step to obtain the desired control effort

Instead of estimating the manipulator parameters of its dynamic model, the parameters of the controller are estimated directly. Therefore, the control algorithm is much simpler than other approaches and the computations of the recursive estimation are reduced.

4.2 Least-square Estimation

The least-square technique provides us with a mathematical procedure by which a model can achieve a best fit to experimental data. This procedure has been widely used in the adaptive control systems.

The principle of least-square is that *the unknown parameters of a model should be chosen in such a way that the sum of the squares of the differences between the actually observed and computed values multiplied by numbers that measure the degree of precision is a minimum.*

In order to derive an analytic solution to the least-square problem, the computed values are required to be linear functions of the unknown parameters.

4.2.1 General Description of Least-square Problem

In formulating an identification problem, the measure of how well a model fits the actual system input and output data is given by introducing a criterion. The criterion is a function of the input error, output error, or generalized error. The function is usually chosen to be quadratic, although it is sometimes of other forms. So, the criterion for a least-square procedure is expressed as

$$\Delta(\nu) = \frac{1}{2}\sum_{i=1}^{N} \varepsilon^T \varepsilon \qquad (4.2.1.1)$$

where ν is a vector of unknown parameters, N is the number of observations and ε is the error vector.

The first one who gave the formulation, solution and application of such an identification problem was Gauss(1809). He formulated this problem and solved it based on the minimization of the sumation of the squares of the error.

The least-square method has the advantage of simplicity and ease to understand and use Althouth it sometimes gives estimates with the wrong mean values, this approach is still widely used in solving the identification problem One item that should be taken into account when we use this method is that the least-square method is restricted to model structures that are linear in the unknown parameters

In the general least-square problem, it is assumed that there is a variable, y, which is related linearly to a set of known functions and the estimated unknown parameters as follows

$$y = \nu_1\phi_1 + \nu_2\phi_2 + \ldots + \nu_n\phi_n \qquad (4\ 2\ 1\ 2)$$

That is, y is computed by the known function and the estimated unknown parameters. Where $\phi_1, \ldots \phi_n$ are functions of the states of the system and $\nu_1, \ldots, \nu_n$ are unknown parameters

Assume that a sequence of k observations on both ϕ and y have been made at times $t_1, \ldots, t_k$ Then the observed data can be denoted as $y(i)$ and $\phi_1(i), \phi_2(i), \ldots, \phi_k(i), i = 1, 2, \ldots, k$. In this way, the following set of k linear equations can be used to relate these data

$$y(i) = \nu_1(i)\phi_1(i) + \nu_2\phi_2(i) + \ldots + \nu_n(i), \qquad i = 1, 2, \ldots, k \qquad (4.2.1\ 3)$$

To simplify the expressions, eq (4 2.1.3) can be rearranged into a simple matrix form as follows

$$\mathbf{y} = \Phi\nu \qquad (4.2\ 1.4)$$

where

$$\mathbf{y} = \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(k) \end{pmatrix}, \qquad \nu = \begin{pmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_n \end{pmatrix} \qquad (4.2.1\ 5)$$

and

$$\Phi = \begin{pmatrix} \phi_1(1) & \dots & \phi_n(1) \\ \phi_1(2) & \dots & \phi_n(2) \\ \vdots & \ddots & \vdots \\ \phi_1(k) & \dots & \phi_n(k) \end{pmatrix} \qquad (4.2.1.6)$$

In order to solve this equation, matrix Φ has to be inverted. When $k = n$, ν can be uniquely solved from eq.(4.2.1.4) as

$$\tilde{\nu} = \Phi^{-1}\mathbf{y} \qquad (4.2.1.7)$$

where $\tilde{\nu}$ is the estimated value of ν

However, when $k > n$, it is impossible to get ν in this way. The alternative is to determine ν using the the least-square theorem

The error vector is defined as

$$\varepsilon = \mathbf{y} - \Phi\nu \qquad (4.2.1.8)$$

From the criterion expressed in eq(4.2.1.1) we can get:

$$\begin{aligned} 2J &= (\mathbf{y} - \Phi\nu)^T(\mathbf{y} - \Phi\nu) \\ &= \mathbf{y}^T\mathbf{y} - \nu^T\Phi^T\mathbf{y} - \mathbf{y}^T\Phi\nu + \nu^T + \nu^T\Phi^T\Phi\nu \end{aligned} \qquad (4.2.1.9)$$

J reaches minimum when the differentiation of it equals to zero

$$\frac{\partial J}{\partial \nu}\Big|_{\nu=\tilde{\nu}} = -\Phi^T\mathbf{y} + \Phi^T\Phi\tilde{\nu} = 0 \qquad (4.2.1.10)$$

Therefore, $\tilde{\nu}$ can be solved as

$$\begin{aligned} \tilde{\nu} &= (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y} \\ &= \Phi^I\mathbf{y} \end{aligned} \qquad (4.2.1.11)$$

where Φ^I is called Moore-Penrose generalized inverse of Φ and the result $\tilde{\nu}$ is the least-square estimation of ν.

4.2.2 Recursive Least-square Estimation

In the case of observations that are obtained sequentially, it is waste to solve the problem from the beginning when the least-square problem has been solved for the k observations and the $(k+1)$-st observation just came in. Therefore, we should find a way to get the estimates for $k+1$ observations on the base of the results obtained for k observations. In this case, our parameter estimates can be improved by making use of the new data and the estimated parameters can be updated step by step without repeating the computation of eq (4.2.1.11). Such a kind of least-square estimation scheme is called *recursive least-square estimation*.

From eq (4.2.1.11), the least-square estimates can be computed on the base of k observations as follows:

$$\tilde{\nu}(k) = [\Phi^T(k)\Phi(k)]^{-1}\Phi^T(k)\mathbf{y}(k) \tag{4.2.2.1}$$

Based on the $(k+1)$-st observation we can obtain a new equation as:

$$y(k+1) = \nu_1\phi_1(k+1) + \nu_2\phi_2(k+1) + \ldots + \nu_n\phi_n(k+1) \tag{4.2.2.2}$$

With the $(k+1)$-st observation, the solution of the system is expressed as:

$$\tilde{\nu}(k+1) = [\Phi^T(k+1)\Phi(k+1)]^{-1}\Phi^T(k+1)\mathbf{y}(k+1) \tag{4.2.2.3}$$

where

$$\Phi(k+1) = \begin{pmatrix} \phi_1(1) & & \phi_n(1) \\ \phi_1(2) & \cdot & \phi_n(2) \\ & \vdots & \vdots \\ \phi_1(k) & & \phi_n(k) \\ --- & --- & --- \\ \phi_1(k+1) & & \phi_n(k+1) \end{pmatrix} = \begin{pmatrix} \Phi(k) \\ ---- \\ \phi^T(k+1) \end{pmatrix} \tag{4.2.2.4}$$

and

$$\mathbf{y}(k+1) = \begin{pmatrix} y(1) \\ \vdots \\ y(k) \\ ---- \\ y(k+1) \end{pmatrix} = \begin{pmatrix} \mathbf{y}(k) \\ ---- \\ y(k+1) \end{pmatrix} \tag{4.2.2.5}$$

Then eq.(4.2.2.3) can be written as:

$$\tilde{\nu}(k+1) = [\Phi^T(k)\Phi(k) + \phi^T(k+1)\phi(k+1)]^{-1}[\Phi^T(k)\mathbf{y}(k) + \phi^T(k+1)y(k+1)] \quad (4.2.2.6)$$

Now we recall the folowing:

LEMMA(Matrix inversion) Let $\mathbf{A}, \mathbf{C}$ and $\mathbf{A} + \mathbf{BCD}$ be non-singular square matrices; then

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1} \quad (4.2.2.7)$$

The proof of this LEMMA is shown in Appendix A. Let

$$\mathbf{P}(k) = (\Phi^T(k)\Phi(k))^{-1} \quad (4.2.2.8)$$

Hence,

$$\mathbf{P}(k+1) = (\Phi^T(k+1)\Phi(k+1))^{-1} \quad (4.2.2.9)$$

By applying eqs.(4.2.2.4), (4.2.2.5) and the matrix-inversion lemma, $\mathbf{P}(k+1)$ can be written in the following form:

$$\begin{aligned}\mathbf{P}(k+1) &= [\mathbf{P}(k)^{-1} + \phi(k+1)\phi^T(k+1)]^{-1} \\ &= \mathbf{P}(k) - \mathbf{P}(k)\phi(k+1)[1 + \phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1} \\ &\quad \cdot \phi^T(k+1)\mathbf{P}(k)\end{aligned} \quad (4.2.2.10)$$

From eqs.(4.2.2.6) and (4.2.2.10), the following is derived:

$$\begin{aligned}\tilde{\nu}(k+1) &= \mathbf{P}(k+1)[\Phi^T(k)\mathbf{y}(k) + \phi(k+1)y(k+1)] \\ &= \mathbf{P}(k)\Phi^T(k)\mathbf{y}(k) - \mathbf{P}(k)\phi(k+1)[1 + \phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1} \\ &\quad \cdot \phi^T(k+1)\mathbf{P}(k)\Phi^T(k)\mathbf{y}(k) + \mathbf{P}(k)\phi(k+1)y(k+1) - \mathbf{P}(k)\phi(k+1) \\ &\quad [1 + \phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1}\phi^T(k+1)\mathbf{P}(k)\phi(k+1)y(k+1)\end{aligned} \quad (4.2.2.11)$$

The last two terms of eq (4.2.2.11) can be rewritten as

$$\begin{aligned}&\mathbf{P}(k)\phi(k+1)[1 + \phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1}[1 + \phi^T(k+1)\mathbf{P}(k)\phi(k+1) \\ &- \phi^T(k+1)\mathbf{P}(k)\phi(k+1)]y(k+1) \\ &= \mathbf{P}(k)\phi(k+1)[1 + \phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1}y(k+1)\end{aligned} \quad (4.2.2.12)$$

From eq.(4.2.2.1) and eq.(4.2.2.8), we can derive the relationship between $\tilde{\nu}$ and $\mathbf{P}$ as:

$$\tilde{\nu}(k) = \mathbf{P}(k)\mathbf{\Phi}^T(k)\mathbf{y}(k) \tag{4.2.2.13}$$

Therefore, eq.(4.2.2.11) is finally simplified as:

$$\begin{aligned}\tilde{\nu}(k+1) =& \tilde{\nu}(k) + \mathbf{P}(k)\phi(k+1)[1+\phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1}\\ &\cdot[y(k+1) - \phi^T(k+1)\tilde{\nu}(k)\end{aligned} \tag{4.2.2.14)]$$

From eq.(4.2.2.14) we can see obviously that the estimated value $\tilde{\nu}(k+1)$ is obtained from previous estimates that have been obtained based on k observations plus a correction term. The correction term is proportional to $y(k+1) - \phi^T(k+1)\nu(k)$ and $\mathbf{P}(k)$ in the correction term is updated by the recursive formulation at each step. In eq.(4.2.2 14), vector $\mathbf{P}(k)\phi(k+1)[1+\phi^T(k+1)\mathbf{P}(k)\phi(k)]^{-1}$ determines how the error is weighted, namely, it tells how the correction and the previous estimates should be combined.

By rearranging all above equations, the *recursive least-square* estimation algorithm is finally written as:

$$\tilde{\nu}(k+1) = \tilde{\nu}(k) + r(k-1)\mathbf{P}(k)\phi(k+1)[y(k+1) - \phi^T(k+1)\tilde{\nu}(k)] \tag{4.2.2.15a}$$

$$\mathbf{P}(k+1) = \mathbf{P}(k) - r(k+1)\mathbf{P}(k)\phi(k+1)\phi^T(k+1)\mathbf{P}(k) \tag{4.2.2.15b}$$

$$r(k+1) = [1+\phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1} \tag{4.2.2.15c}$$

In the above formulation one can find that all the observations are given the same weight, which is suitable for time-invariant systems. But the robotic manipulator under disscussion is a time-varying system, and hence it is necessary to eliminate the influence of old data. This can be done by using a loss function with exponential weighting, i.e.,

$$\Delta(\nu) = \sum_{k=1}^{n} \rho^{n-k}[y(k+1) - \phi^T(k)\nu(k)]^2, \qquad 0<\rho<1 \tag{4.2.2.16}$$

where the "forgetting factor", ρ, is a measure of how fast old data are forgotten.

The least-square estimation, when using the loss function of eq.(4.2.2.16), is given by

$$\tilde{\nu}(k+1) = \tilde{\nu}(k) + r(k+1)\mathbf{P}(k)\phi(k+1)[y(k+1) - \phi^T(k+1)\tilde{\nu}(k)] \quad (4.2.2.17a)$$

$$\mathbf{P}(k+1) = [\mathbf{P}(k) - r(k+1)\mathbf{P}(k)\phi(k+1)\phi^T(k+1)\mathbf{P}(k)] \quad (4.2.2.17b)$$

$$r(k+1) = [\rho + \phi^T(k+1)\mathbf{P}(k)\phi(k)]^{-1} \quad (4.2.2.17c)$$

These three equations are used to estimate the parameters at the k-th step. By using eqs.(4.2.2.17a–c) recursively, the unknown vectors $\nu(k)$ can be estimated at each sampling period.

4.2.3 Selection of the Initial Value of Matrix P

When we start the parameter estimation using *recursive least-square* scheme, the initial value of matrix **P** should be chosen first.

Substitute eq.(4.2.2.17c) into eq.(4.2.2.17b), we get:

$$\mathbf{P}(k+1) = [\mathbf{P}(k) + \phi^T(k+1)\phi(k+1)]^{-1} \quad (4.2.3.1)$$

Starting from $\mathbf{P}(0)$ and iterating k times using eq.(4.2.3.1), $\mathbf{P}(k)$ can be written as:

$$\mathbf{P}(k) = [\mathbf{P}(0)^{-1} + \phi^T(k)\phi(k)]^{-1} \quad (4.2.3.2)$$

Notice that, eq.(4.2.3.2) has to agree with eq.(4.2.2.8). The simple way is to force $\mathbf{P}(0)^{-1}$ to zero. This can be done by letting $\mathbf{P} = \alpha\mathbf{1}$ and $\alpha \to \infty$, namely,

$$\lim_{\alpha\to\infty} \mathbf{P}(0)^{-1} = \lim_{\alpha\to\infty} \frac{1}{\alpha}\mathbf{1} = 0 \quad (4.2.3.3)$$

where $\mathbf{1}$ is the $n \times n$ identity matrix.

4.3 Controller Estimation

The method presented here is aimed at estimating the controller directly. In this way, we can make the control scheme much simpler and reduce the computations involved The discretized equations of the system are derived first, then the details about the controller estimation will be described.

4.3.1 Discretization of the System

In order to provide an apropriate form for applying the *model assignment* control scheme derived in Chapter 2, the linearized equation of the system, eq.(2.4.1), is expressed in the state space. Let

$$\mathbf{x}_1(t) = \delta\theta \qquad (4.3.1.1a)$$

$$\mathbf{x}_2(t) = \delta\dot{\theta} \qquad (4.3.1.1b)$$

$$\mathbf{u}(t) = \delta\tau \qquad (4.3.1.1c)$$

where $\mathbf{x}_1(t), \mathbf{x}_2(t)$ and $\mathbf{u}(t)$ are all n-dimensional vectors, the first two denoting the states and the third one the control variable.

Then, eq.(2.4.1) can be written in the state space as:

$$\dot{\mathbf{x}}_1(t) = \mathbf{x}_2(t) \qquad (4.3.1.2a)$$

$$\dot{\mathbf{x}}_2(t) = \mathbf{I}(t)^{-1}[\mathbf{u}(t) - \mathbf{C}(t)\mathbf{x}_2(t) - \mathbf{D}(t)\mathbf{x}_1(t)] \qquad (4.3.1.2b)$$

Let

$$\mathbf{V}(t) = \left(\begin{array}{c|c} \mathbf{0} & \mathbf{1} \\ \hline -\mathbf{I}(t)^{-1}\mathbf{D}(t) & -\mathbf{I}(t)^{-1}\mathbf{C}(t) \end{array}\right) \qquad (4.3.1.3)$$

and

$$\mathbf{U}(t) = \left(\begin{array}{c} \mathbf{0} \\ \hline \mathbf{I}^{-1}(t) \end{array}\right), \qquad \mathbf{x} = \left(\begin{array}{c} \mathbf{x}_1 \\ \hline \mathbf{x}_2 \end{array}\right) \qquad (4.3.1.4)$$

where $\mathbf{0}$ is a $n \times n$ zero matrix.

Eqs.(4.3.1.2a) and (4.3.1.2b) can be rewritten as:

$$\dot{\mathbf{x}}(t) = \mathbf{V}(t)\mathbf{x}(t) + \mathbf{U}(t)\mathbf{u}(t) \tag{4.3.1.5}$$

For least-square parameter estimation, the appropriate discrete linear equations should be derived on the base of eq.(4.3.1.5).

It is assumed that the discrete events of interest can be indexed by a sequence of real numbers $t_1, t_2, \ldots, t_k, \ldots$, which are thought of as discrete-time points.

The relationship between the system variables at the sampling steps is derived. The state at the next sampling time t_{k+1} can be obtained by the given state at the sampling time t_k. Since robotic systems are slow time-varying, when t_{k+1} is close to t_k, matrices $\mathbf{V}$ and $\mathbf{U}$ and system input $\mathbf{u}$ can be assumed to be constant.

Therefore, the state at the next sampling step t_{k+1} is given by:

$$\mathbf{x}(t_{k+1}) = e^{\mathbf{V}(t_k)(t_{k+1}-t_k)}\mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} e^{\mathbf{V}(t_k)(t_{k+1}-s)} ds\mathbf{U}(t_k)\mathbf{u}(t_k) \tag{4.3.1.6}$$

Let

$$\mathbf{M}(t_{k+1}, t_k) = e^{\mathbf{V}(t_k)(t_{k+1}-t_k)} \tag{4.3.1.7}$$

$$\mathbf{N}(t_{k+1}, t_k) = \int_0^{t_{k+1}-t_k} e^{\mathbf{V}(t_k)s'} ds'\mathbf{U} \tag{4.3.1.8}$$

The system equation of the sampling system is then written as:

$$\mathbf{x}(t_{k+1}) = \mathbf{M}(t_{k+1}, t_k)\mathbf{x}(t_k) + \mathbf{N}(t_{k+1}, t_k)\mathbf{u}(t_k) \tag{4.3.1.9}$$

For periodic sampling with period t_s, the model of eq.(4.3.1.9) is simplified to the following system:

$$\mathbf{x}(kt_s + t_s) = \mathbf{M}(kt_s + t_s)\mathbf{x}(kt_s) + \mathbf{N}(kt_s + t_s)\mathbf{u}(kt_s) \tag{4.3.1.10}$$

Now, the problem left is how to compute matrices $\mathbf{M}$ and $\mathbf{N}$. One way to simplify the computation is to calculate

$$\mathbf{B} = \int_0^{t_s} e^{\mathbf{V}s} ds = \mathbf{1}t_s + \frac{\mathbf{V}t_s^2}{2!} + \frac{\mathbf{V}^2 t_s^3}{3!} + \ldots + \frac{\mathbf{V}^i t_s^{i+1}}{(i+1)!} + \ldots \tag{4.3.1.11}$$

Therefore, matrices **M** and **N** are given as

$$\mathbf{M} = \mathbf{1} + \mathbf{V}\mathbf{B} \qquad (4.3.1.12a)$$

$$\mathbf{N} = \mathbf{B}\mathbf{U} \qquad (4.3.1.12b)$$

where **1** is an $n \times n$ identity matrix.

When t_s is chosen to be very small, the higher-order term in eq. (4.3.1.11) can be eliminated to obtain:

$$\mathbf{M} = \mathbf{1} + \mathbf{V}\mathbf{1}t_s \qquad (4.3.1.13a)$$

$$\mathbf{N} = \mathbf{1}t_s\mathbf{U} \qquad (4.3.1.13b)$$

The discrete-time system at hand is obtained by substituting eqs.(4.3.1.13a) and (4.3.1.13b) into eq.(4.3.1.9).

$$\mathbf{x}(kt_s + t_s) = (\mathbf{1} + \mathbf{V}(kt_s)\mathbf{1}t_s)\mathbf{x}(k) + \mathbf{1}t_s\mathbf{U}(kt_s)\mathbf{u}(kt_s) \qquad (4.3.1.14)$$

This equation can be expressed in a simple way as follows:

$$\mathbf{x}(k+1) = (\mathbf{1} + \mathbf{V}(k))\mathbf{x}(k) + t_s\mathbf{U}(k)\mathbf{u}(k) \qquad (4.3.1.15)$$

Thus, the discrete-time system of the robot linearized model expressed in eqs.(4.3.1.2a) and (4.3.1.2b) is finally derived as:

$$\mathbf{x}_1(k+1) = \mathbf{x}_1(k) + t_s\mathbf{x}_2(k) \qquad (4.3.1.16a)$$

$$\mathbf{x}_2(k+1) = t_s\mathbf{I}^{-1}(k)[\mathbf{u}(k) - \mathbf{C}(k)\mathbf{x}_2(k) - \mathbf{D}(k)\mathbf{x}_1(k)] + \mathbf{x}_2(k) \qquad (4.3.1.16b)$$

A new approach of adaptive control scheme called controller estimation, based on model assignment, is derived. Instead of estimating the parameters in the above discrete-time system, namely, matrices **I**, **C** and **D**, the parameters in the feedback gains are estimated. Details are described in the next sub-section.

4.3.2 Controller Estimation

In Chapter 2, a *model-assignment* control scheme has been derived. The position and velocity feedback are used to render the system a prescribed second order stationary system. In this sub-section, a control estimation scheme is designed to perform the parameter estimation for the feedback gains obtained in *model assignment*. The structure of such a system is shown in Figure 4.1.

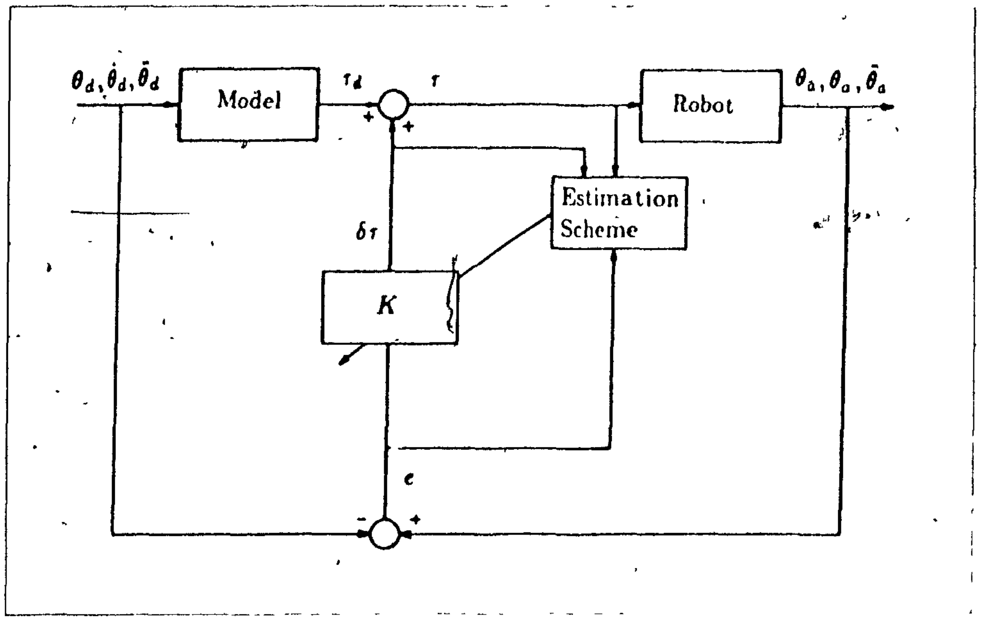
$\theta_d, \dot{\theta}_d, \ddot{\theta}_d$
Model
τ_d
τ
Robot
$\theta_a, \dot{\theta}_a, \ddot{\theta}_a$
Estimation Scheme
$\delta\tau$
K
e

Figure 4.1 Adaptive Control with Controller Estimation

The following relationship already introduced in eqs (2.4.5) and (2.4.6) is recalled

$$\mathbf{K}_p(t) \quad \mathbf{I}(t)\Omega_n^2 \quad \mathbf{D}(t) \tag{4.3.2.1a}$$

$$\mathbf{K}_d(t) \quad 2\mathbf{I}(t)\Psi^T\Omega_n\Psi \quad \mathbf{C}(t) \tag{4.3.2.1b}$$

In the discrete-time system, the forgoing relationship can be expressed as

$$\mathbf{D}(k) \quad \mathbf{I}(k)\Omega_n^2 \quad \mathbf{K}_p(k) \tag{4.3.2.2a}$$

$$\mathbf{C}(k) - 2\mathbf{I}(k)\Psi^T\Omega_n\Psi \quad \mathbf{K}_d(k) \tag{4.3.2.2b}$$

Substitution of eqs (4.3.2.2a) and (4.3.2.2b) into eq.(4.3.1.16b) and rearrangement of the arising equations yields

$$\mathbf{I}(k)\mathbf{x}_2(k+1) = t_s[\mathbf{u}(k) - \mathbf{I}(k)\mathbf{\Omega}_n^2\mathbf{x}_1(k) - 2\mathbf{I}(k)\mathbf{\Psi}^T\mathbf{\Omega}_n\mathbf{\Psi}\mathbf{x}_2(k)]$$
$$+ \mathbf{I}(k)\mathbf{x}_2(k) + t_s[\mathbf{K}_p(k)\mathbf{x}_1(k) + \mathbf{K}_d(k)\mathbf{x}_2(k)] \qquad (4.3.2.3)$$

where t_s is the sampling period. In eq (4.3.2.3), the unknown matrices are $\mathbf{K}_p, \mathbf{K}_d$ and $\mathbf{I}$. $\mathbf{K}_p$ and $\mathbf{K}_d$ are the feedback gain matrices which will be estimated by using least-square identification scheme. $\mathbf{I}$ is the inertia matrix of the system which we want to avoid to estimate. This problem can be solved as follows

Let the theoretical linearized model of the robot manipulator be represented as:

$$\mathbf{I}^*(t)\delta\ddot{\theta} + \mathbf{C}^*(t)\delta\dot{\theta} + \mathbf{D}^*(t)\delta\theta = \mathbf{u}(t) \qquad (4.3.2.4)$$

where $\mathbf{I}^*(t), \mathbf{C}^*(t)$ and $\mathbf{D}^*(t)$ are the theoretical parameter matrices of the robot manipulator model.

Then, $\mathbf{I}^*(k)$, which denotes the theoretical inertia matrix of the robot manipulator at the k-th step, is used instead of the actual $\mathbf{I}(k)$ in eq (4.3.2.3) and an arbitrary error vector $\mu(k)$ is added to the right-hand side of eq.(4.3.2.3) to obtain:

$$\mathbf{I}^*(k)\mathbf{x}_2(k+1) = t_s[\mathbf{u}(k) - \mathbf{I}^*(k)\mathbf{\Omega}_n^2\mathbf{x}_1(k) - 2\mathbf{I}^*(k)\mathbf{\Psi}^T\mathbf{\Omega}_n\mathbf{\Psi}\mathbf{x}_2(k)] + \mathbf{I}^*(k)\mathbf{x}_2(k)$$
$$+ t_s[\mathbf{K}_p(k)\mathbf{x}_1(k) + \mathbf{K}_d(k)\mathbf{x}_2(k)] + \mu(k) \qquad (4.3.2.5)$$

Let

$$\mathbf{y}(k+1) = \mathbf{I}^*(k)\mathbf{x}_2(k+1) \qquad (4.3.2.6a)$$
$$\mathbf{b}(k) = t_s[\mathbf{u}(k) - \mathbf{I}^*(k)\mathbf{\Omega}_n^2\mathbf{x}_1(k) - 2\mathbf{I}^*(k)\mathbf{\Psi}^T\mathbf{\Omega}_n\mathbf{\Psi}\mathbf{x}_2(k)] + \mathbf{I}^*(k)\mathbf{x}_2(k) \qquad (4.3.2.6b)$$

Eq.(4.3.2.5) can be written as:

$$\mathbf{y}(k+1) = \mathbf{K}_p(k)\mathbf{x}_1(k)t_s + \mathbf{K}_d(k)\mathbf{x}_2(k)t_s + \mathbf{b}(k) + \mu(k) \qquad (4.3.2.7)$$

Let

$$\mathbf{x}_1(k)t_s = [\phi_1(k) \cdots \phi_n(k)]^T \tag{4.3.2.8}$$

$$\mathbf{x}_2(k)t_s = [\phi_{n+1}(k) \cdots \phi_{2n}(k)]^T \tag{4.3.2.9}$$

$$\mathbf{e}(k)\alpha = \mathbf{b}(k) + \mu(k) \tag{4.3.2.10}$$

where $\mathbf{e}(k)$ is a n-dimensional unknown vector and α is an estimation weighting factor which can be adjusted during the recursive identification procedure.

Finally, eq.(4.3.2.5) can be written in the following form

$$\mathbf{y}(k+1) = \mathbf{K}(k)\phi(k) \tag{4.3.2.11}$$

where

$$\mathbf{y}(k+1) = \begin{pmatrix} y_1(k+1) \\ \vdots \\ y_i(k+1) \\ \vdots \\ y_n(k+1) \end{pmatrix}, \qquad \phi(k) = \begin{pmatrix} \phi_1(k) \\ \vdots \\ \phi_i(k) \\ \vdots \\ \phi_{2n}(k) \\ \alpha \end{pmatrix} \tag{4.3.2.12}$$

and

$$\mathbf{K}(k) = \begin{pmatrix} k_p(k)_{11} & \dots & k_p(k)_{1n} & k_d(k)_{11} & \dots & k_d(k)_{1n} & e_1(k) \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ k_p(k)_{i1} & \dots & k_p(k)_{in} & k_d(k)_{i1} & \dots & k_d(k)_{in} & e_i(k) \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ k_p(k)_{n1} & \dots & k_p(k)_{nn} & k_d(k)_{1n} & \dots & k_d(k)_{nn} & e_n(k) \end{pmatrix} \tag{4.3.2.13}$$

Now we define the i-th row of the unknown parameters in matrix $\mathbf{K}(k)$ as the following $(2n+1)$-dimensional vector $\mathbf{z}_i(k)$, where $i = 1, 2, \dots, n$

$$\mathbf{z}_i(k) = [k_p(k)_{i1}, \dots, k_p(k)_{in}, k_d(k)_{i1}, \dots, k_d(k)_{in}, e_i(k)]^T \tag{4.3.2.14}$$

Hence, eq.(4.3.2.10) can be written as

$$y_i(k+1) = \phi^T(k)\mathbf{z}_i(k); \qquad i = 1, 2, \dots, n \tag{4.3.2.15}$$

In the framework of the above formulation, the model output is linear in the unknown parameters to be identified. Hence, based on eq (4.3.2.15), the *recursive least-square* estimation scheme, which has been derived in the last section can be introduced as:

$$\begin{aligned}
\tilde{\mathbf{z}}_i(k+1) &= \tilde{\mathbf{z}}_i(k) + r(k+1)\mathbf{P}(k)\phi(k+1)[y_i(k+1) - \phi^T(k+1)\tilde{\mathbf{z}}_i(k)] \\
\mathbf{P}(k+1) &= \mathbf{P}(k) - r(k+1)\mathbf{P}(k)\phi(k+1)\phi^T(k+1)\mathbf{P}(k) \qquad (4.3.2.17) \\
r(k+1) &= [\rho + \phi^T(k+1)\mathbf{P}(k)\phi(k+1)]^{-1}
\end{aligned}$$

Eq.(4.3.2.17) is used to estimate the parameters in the feedback gain matrix at the k-th step. $\phi^T(k+1)\tilde{\mathbf{z}}_i(k)$ is the predicted value of $y_i(k+1)$ based on the estimated value $\tilde{\mathbf{z}}_i(k)$. The corrections are performed by $r(k+1)\mathbf{P}(k)\phi(k+1)[y_i(k+1) - \phi^T(k+1)\tilde{\mathbf{z}}_i(k)]$, where $r(k+1)\mathbf{P}(k)\phi(k+1)$ is the weighting factor which indicates how the corrections and the previous estimates are weighted to obtain the new estimated $\tilde{\mathbf{z}}_i(k+1)$, i.e., $\tilde{\mathbf{z}}_i(k+1)$ is a combination of the estimate value at the k-th step, $\tilde{\mathbf{z}}_i(k)$, and the corrections. The "forgetting factor" is usually chosen to be $0.91 \leq \rho \leq 0.99$. By using eq (4.3.2.17) recursively, the unknown vectors $\mathbf{z}_i(k)$ can be estimated at each sampling period, thereby estimating the feedback gain matrix.

Chapter 5 Dynamic Modelling and Adaptive Control Simulations

5.1 Introduction

In order to test the dynamic modelling algorithm and the adaptive control algorithm, simulations of the two algorithms were performed A two-link manipulator was used in the simulation The reason for using this manipulator is that we can get the closed form dynamic model of it This closed-form model can be used to check the dynamic modelling package derived by using numerical method The simulations were also performed on the Cincinnati Milacron and PUMA 600 Robot An IMSL subroutine that implements the fifth-/sixth-order Runge-Kutta method was used in the simulation

Satisfactory results of the simulation were odtained

5.2 Simulations of Dynamical Package

The dynamic modelling package was implemented on a VAX-780 computer(McRCIM). Two examples were performed, one on a two-link manipulator and another one on the Cincinnati Milacron Robot.

5.2.1 Simulation on the Two-link Manipulator

In this example, a two-link manipulator is used to perform the simulation. The architecture of the robot is shown in Figure 5.1, the input parameters are listed in Table 5.1 Both links are assumed to be the slender, uniform and rigid bars The masses and lengths of link 1 and link 2 are m_1, m_2 and l_1, l_2 respectively The closed-form dynamical equations of this manipulator is derived as

$$\begin{pmatrix} m_{11} & 0 \\ 0 & m_{22} \end{pmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} + \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} \tau_{d1} \\ \tau_{d2} \end{pmatrix} \qquad (5.2.1.1)$$

where

$$m_{11} = \frac{1}{3} m_1 l_1^2 + m_2 (l_1^2 + l_1 l_2 \cos\theta_2 + \frac{1}{3} l_2^2 \cos^2\theta_2) \qquad (5.2.1.1a)$$

$$m_{22} = \frac{1}{3} m_2 l_2^2 \qquad (5\ 2.1\ 1b)$$

$$h_1 = -m_2 l_2 \sin\theta_1 (l_1 + \frac{2}{3} l_2 \cos\theta_2) \dot{\theta}_1 \dot{\theta}_2 + b_1 \dot{\theta}_1 \qquad (5.2\ 1.1c)$$

$$h_2 = m_2 (\frac{1}{2} l_1 l_2 + \frac{1}{3} l_2^2 \cos^2\theta_2) \dot{\theta}_2^2 \sin\theta_2 + \frac{1}{2} g m_2 l_2 \cos\theta_2 + b_2 \dot{\theta}_2 \qquad (5.2.1\ 1d)$$

The joint trajectories are given as

$$\theta_1 = 15 + 20t^2 - \frac{40}{9} t^3 \qquad (deg)$$

$$\theta_2 = 30t^2 - \frac{20}{3} t^3 \qquad (deg) \qquad (5.2\ 1\ 2)$$

and the total simulation time is 3(s) Simulations are performed for both closed form dynamical model and the dynamic modelling package using the same trajectories as shown in eq (5 2,1 2)

We can find that the results obtained from numerical modelling package and closed-form model are almost the same, namely, the errors between them are very very small. The maximum error between torques from the two packages is only 0.0001(Nm). The outputs are shown in Table 5 2 and 5.3.

5.2.2 Simulation on the Cincinnati Milacron Robot

In this example, the dynamic modelling package is tested on the Cincinnati Milacron Robot. The end effector of the robot moves along a straight line in the Cartesian space described by following equations

$$
\begin{array}{lll}
x = 1.33125\text{m}, & \dot{x} = 0.0\text{ms}^{-1}, & \ddot{x} = 0.0\text{ms}^{-2} \\
y = \beta\text{m}, & \dot{y} = \dot{\beta}\text{ms}^{-1}, & \ddot{y} = \ddot{\beta}\text{ms}^{-2} \\
z = 1.79875\text{m}, & \dot{z} = 0.0\text{ms}^{-1}, & \ddot{z} = 0.0\text{ms}^{-2}
\end{array}
\tag{5.2.2.1}
$$

where β is a prescribed function of time defined with cubic splines so as to meet the following end conditions

$$
\begin{array}{l}
\beta(0) = 0, \quad \beta(T) = 1\text{m} \\
\dot{\beta}(0) = \dot{\beta}(T) = 0 \\
\ddot{\beta}(0) = \ddot{\beta}(T) = 0 \\
T = 1\text{s}
\end{array}
\tag{5.2.2.2}
$$

where T is the total simulation time. Details about the splines used to create the trajectories were derived by Angeles (1986).

The architecture of the manipulator is shown in Figure 5.2, and the Hartengerg-Denavit parameters and the inertia parameters of the robot are shown in Table 5.4 and 5.5, respectively. The trajectories of the six joint obtained from inverse kinematics are shown in Figure 5.3 - 5.8. The outputs torques are shown in Figure 5.9 - 5.14. These results are the same as those from Ma Ou's dynamic modelling package(1987) by using the same trajectories for the same robot.

Table 5.1 Input Parameters of the Two-link Manipulator

m_1(kg)	m_2(kg)	l_1(m)	l_2(m)
10.00	10.00	1.00	1.00

Table 5.2 Output of Numerical Solution

$t(s)$	$\tau_1(\mathrm{Nm})$	$\tau_2(\mathrm{Nm})$
0.0	1.8617	52.5407
0.3	14 8434	51.8081
0.6	10 1574	50.4711
0.9	5 0767	47.8179
1.2	-0.2436	42.8631
1.5	-4.6461	35.2343
1.8	-7.2514	25.4771
2.1	-8.1306	15.0168
2.4	-8.1368	5.6608
2.7	-8.2923	-0.9521
3.0	-9 1319	-3.3957

Table 5.3 Output of Closed Form Solution

$t(s)$	$\tau_1(\mathrm{Nm})$	$\tau_2(\mathrm{Nm})$
0.0	1.8617	52.5407
0.3	14 8454	51.8081
0.6	10 1575	50.4710
0.9	5.0767	47.8178
1 2	-0.2436	42.8632
1.5	-4.6460	35.2343
1.8	-7.2514	25.4772
2.1	-8.1305	15.0168
2.4	-8.1369	5 6608
2.7	-8.2923	-0.9522
3.0	-9.1319	-3.3957

Table 5.4 Hartenberg-Denavit Parameters of the Cincinnati Milacron Robot

joint number	a_i(m)	b_i(m)	α_i(degrees)	initial θ_i(degrees)
1	0.00	1.50	90.00	0.00
2	1 02	0.00	0.00	90 00
3	1.02	0.00	0.00	-135.00
4	0.20	0.00	90.00	45 00
5	0.00	0.00	90.00	90.00
6	0.00	0.41	90.00	90.00

Table 5.5 Inertia Parameters of the Cincinnati Milacron Robot

item	joint 1	joint 2	joint 3	joint 4	joint 5	joint 6
$m_i(kg)$	680.0	360.0	180.0	55.0	36.0	68.0
$x_{ci}(m)$	0.0	-0.87	-0.64	-0.12	0.0	0.0
$y_{ci}(m)$	-0.33	0.0	0.04	0.04	-0.05	0.0
$z_{ci}(m)$	0.0	-0.13	0.0	0 0	-0.08	0.0
$I_{xxi}(kgm^2)$	0 0	11.0	1.10	0.44	0.47	0.44
$I_{xyi}(kgm^2)$	0.0	0.0	0.0	0.0	0 0	0.0
$I_{xzi}(kgm^2)$	0 0	0.0	0.0	0.0	0 0	0 0
$I_{yyi}(kgm^2)$	62.0	53.0	44 0	0 91	0.38	0.64
$I_{yzi}(kgm^2)$	0.0	0.0	0 0	0 0	0 0	0 0
$I_{zzi}(kgm^2)$	0.0	44.0	44 0	0 82	0 18	0.73

m_i	mass of the i-th link
x_{ci}, y_{ci}, z_{ci}	coordinates of the mass center of the i-th link
$I_{xxi}, I_{xyi}, I_{xzi}$	
$I_{yyi}, I_{yzi}, I_{zzi}$	entries of the inertia matrix of the i-th link

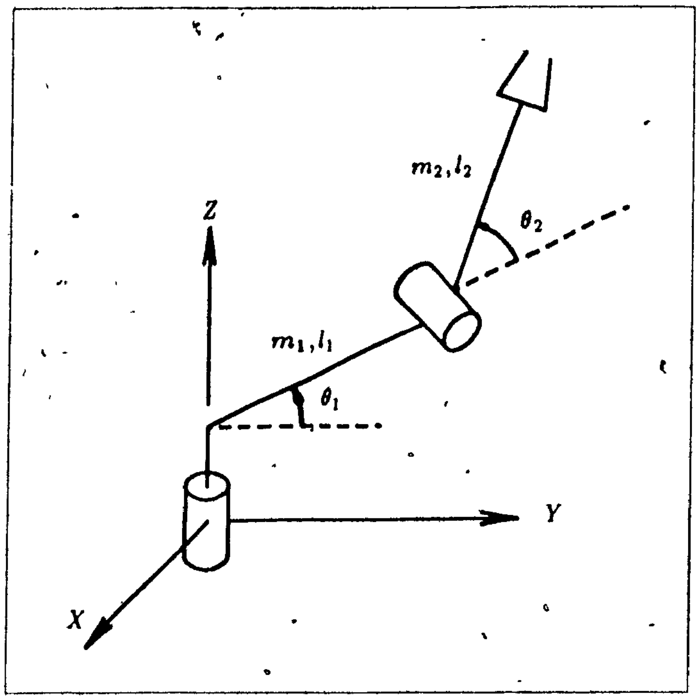

Figure 5.1 The Architecture of the Two-link Manipulator

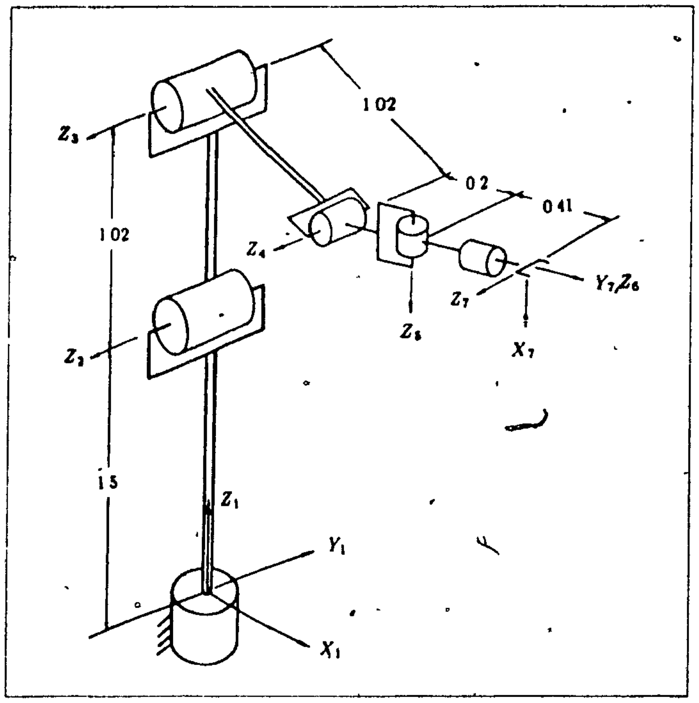

Figure 5.2 The Architectuie of the Cicinnati Milacron Robot

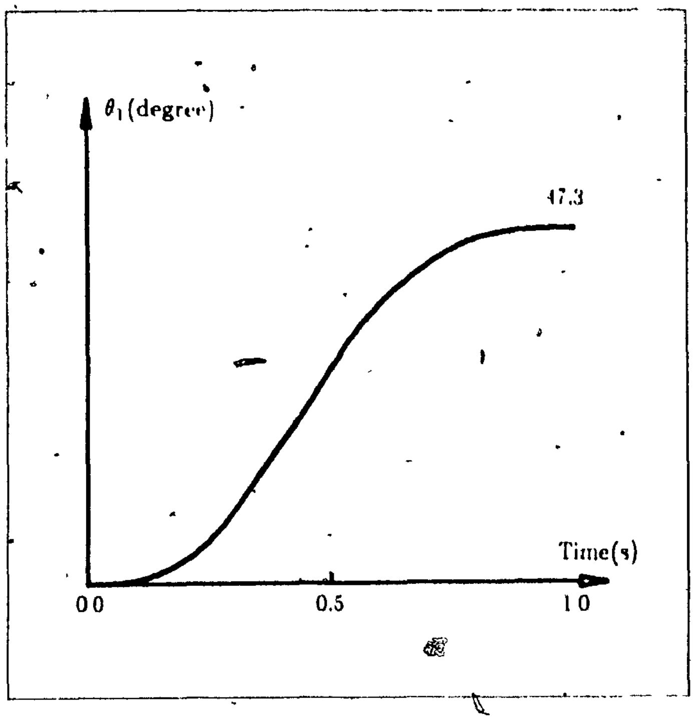

Figure 5.3 Trajectory of joint 1

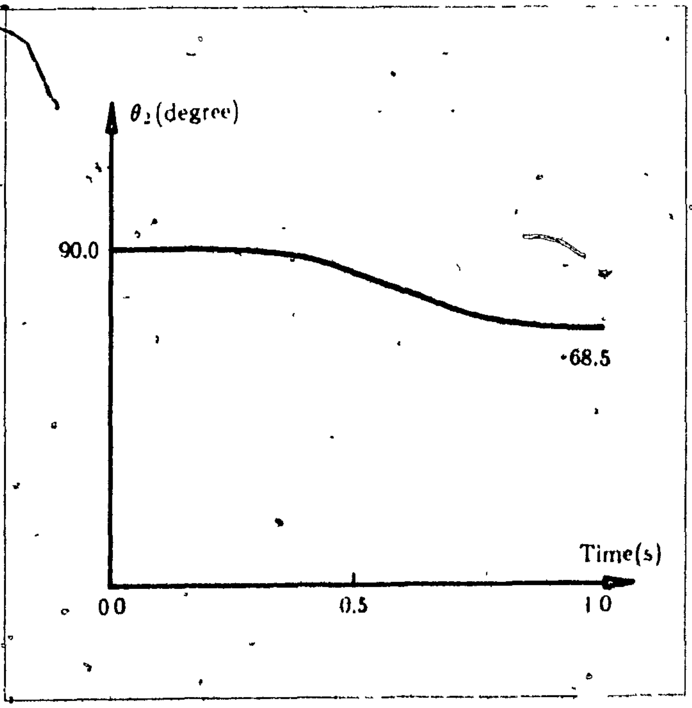

Figure 5.4 Trajectory of joint 2

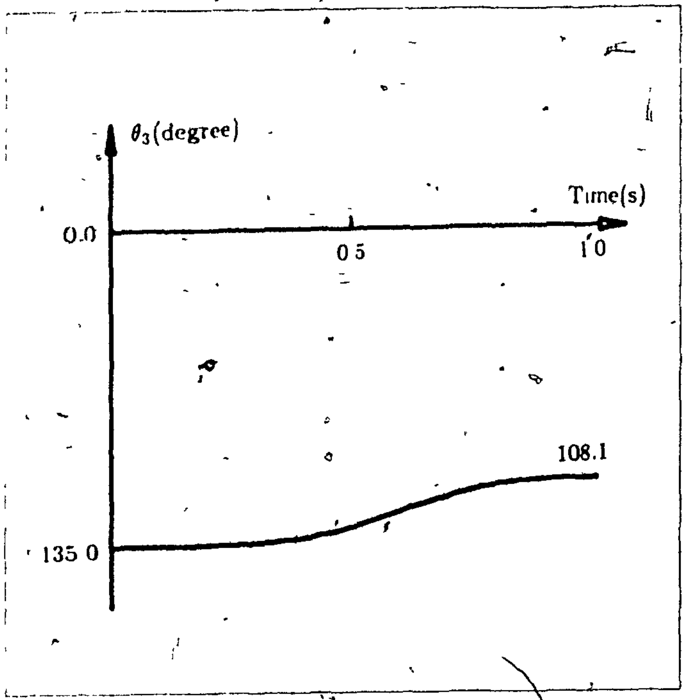

Figure 5.5 Trajectory of joint 3

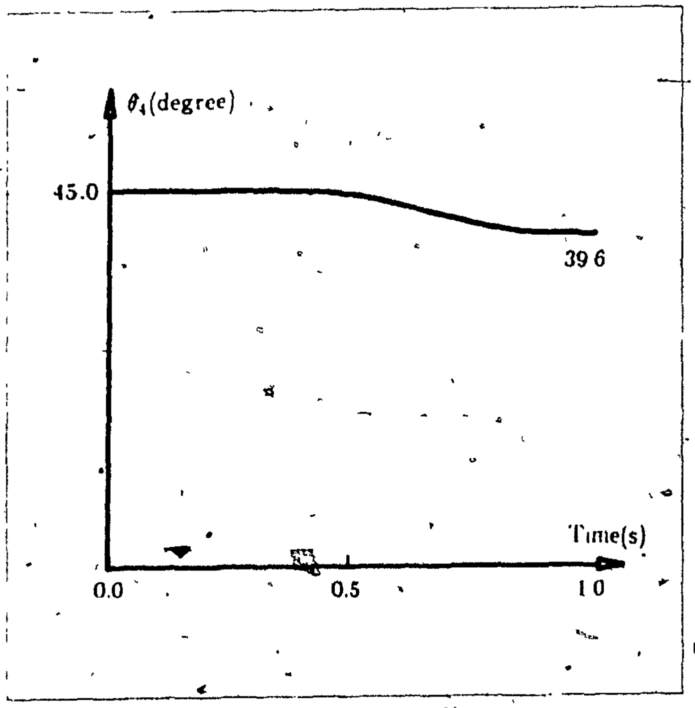

Figure 5.6 Trajectory of joint 4

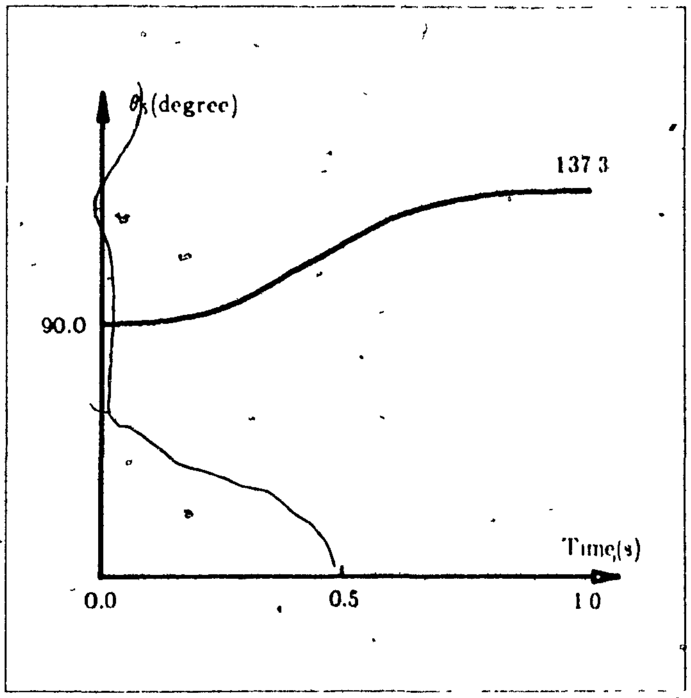

Figure 5.7 Trajectory of joint 5

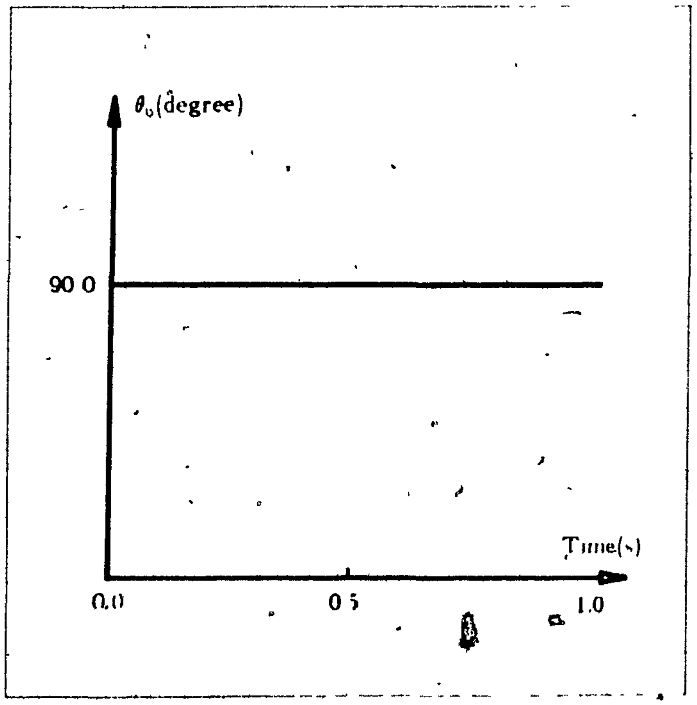

Figure 5.8 Trajectory of joint 6

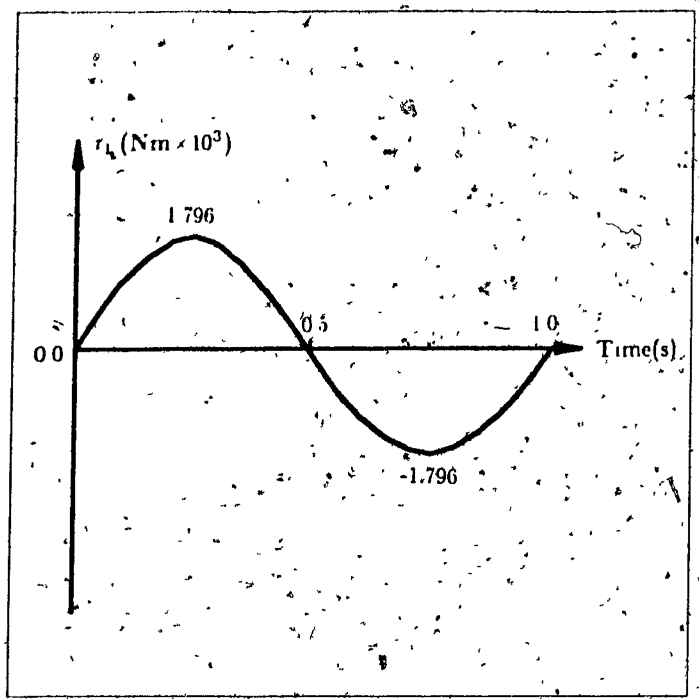

Figure 5.9 Torque on joint 1

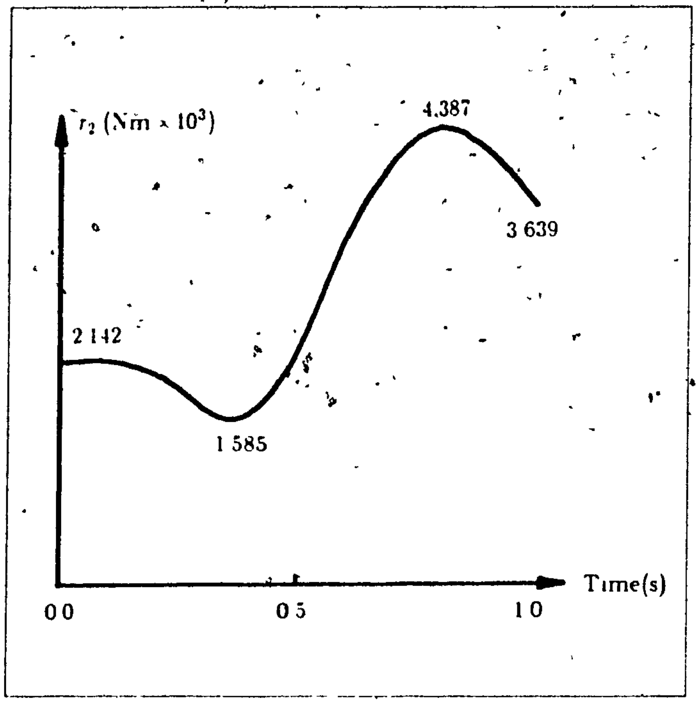

Figure 5.10 Torque on joint 2

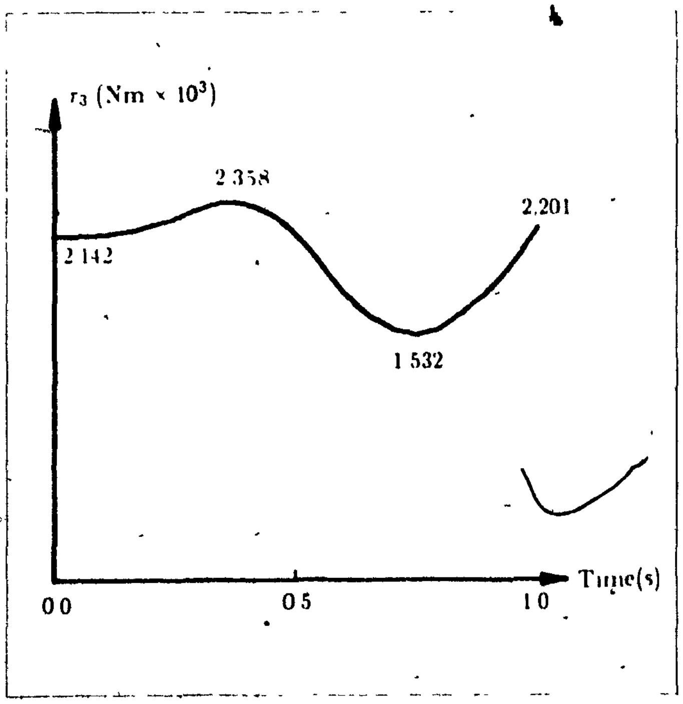

Figure 5.11 Torque on joint 3

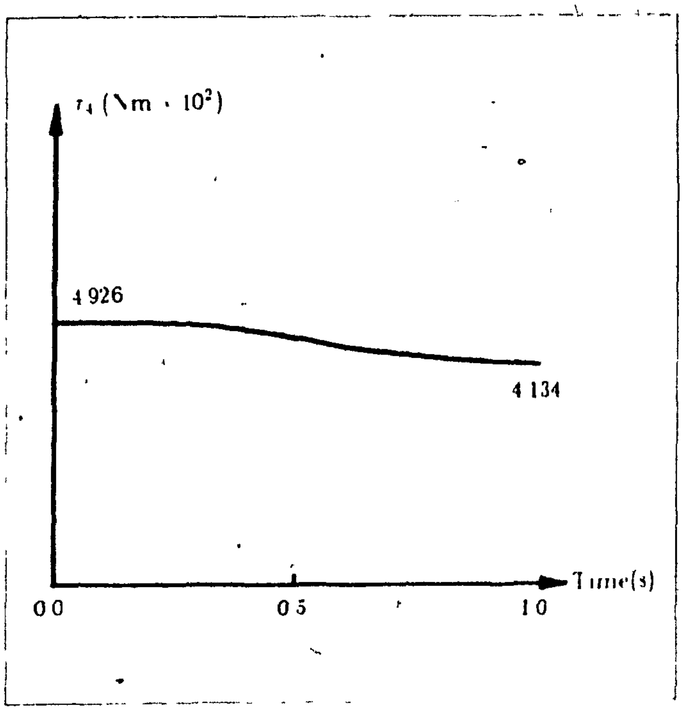

Figure 5.12 Torque on joint 4

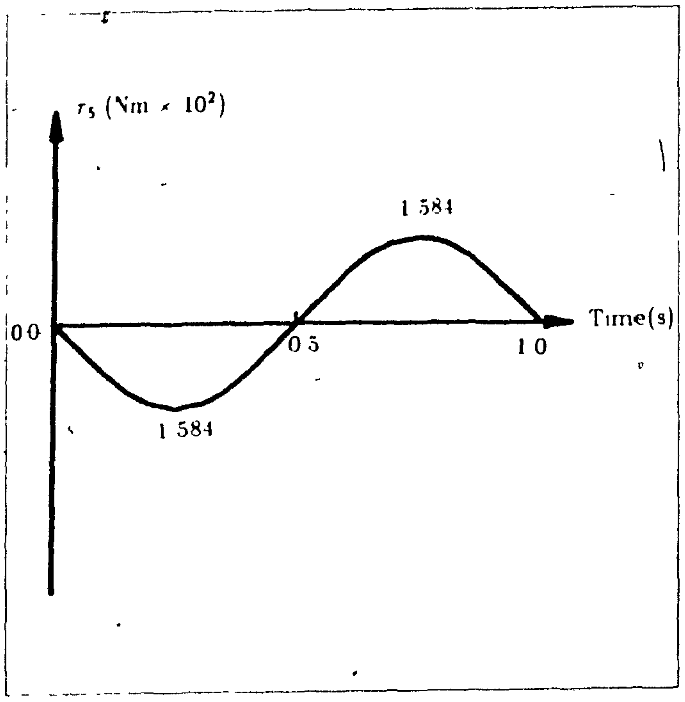

Figure 5.13 Torque on joint 5

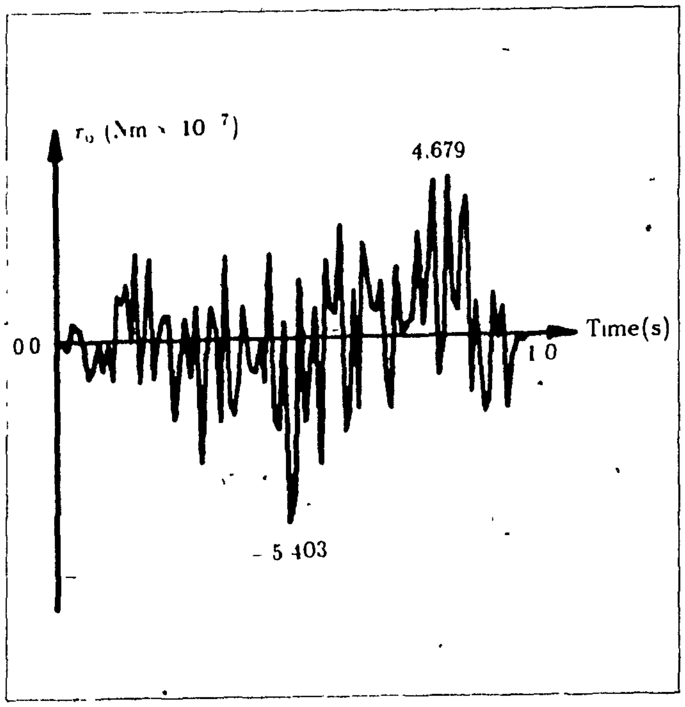

Figure 5.14 Torque on joint 6

5.3 Simulations of Adaptive Control Package

Two examples were performed for the simulation of adaptive control package. This package was also implemented on a VAX-780(McRCIM)

5.3.1 Simulation on the Two-link Manipulator

The simulation of the control algorithm on a two-link robot manipulator is performed. The architecture of this manipulator is shown in Fig 5.1 The dynamical model of this manipulator is expressed in eq (5 2 1 1) Based on this model the linearized perturbation equation is obtained as follows

$$\begin{pmatrix} m_{11} & 0 \\ 0 & m_{22} \end{pmatrix}\begin{pmatrix} \delta\ddot{\theta}_1 \\ \delta\ddot{\theta}_2 \end{pmatrix} + \begin{pmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{pmatrix}\begin{pmatrix} \delta\dot{\theta}_1 \\ \delta\dot{\theta}_2 \end{pmatrix} + \begin{pmatrix} 0 & d_{12} \\ 0 & d_{22} \end{pmatrix}\begin{pmatrix} \delta\theta_1 \\ \delta\theta_2 \end{pmatrix} \quad \begin{pmatrix} \delta\tau_1 \\ \delta\tau_2 \end{pmatrix} \qquad (5.3.1.1)$$

where

$$c_{11} = -m_2 l_2(l_1 - \frac{1}{3}l_2\cos\theta_{d2})\theta_{d2}\sin\theta_{d2} \qquad (5.3.1.1a)$$

$$c_{12} = -m_2 l_1(l_1 + \frac{1}{3}l_2\cos\theta_{d2})\theta_{d1}\sin\theta_{d2} \qquad (5.3.1.1b)$$

$$c_{21} = m_2 l_2(l_1 + \frac{2}{3}l_2\cos\theta_{d2})\theta_{d1}\sin\theta_{d2} \qquad (5.3.1.1c)$$

$$c_{22} = 0 \qquad (5.3.1.1d)$$

$$\begin{aligned} d_{12} = & -m_2 l_2\{[l_1 cos\theta_{d2} + \frac{1}{3}l_2(\cos^2\theta_{d2} \quad \sin^2\theta_{d2})]\dot{\theta}_{d1}\dot{\theta}_{d2} \\ & + \sin\theta_{d2}(l_1 + \frac{2}{3}l_2\cos\theta_{d2})\ddot{\theta}_{d1}\} \end{aligned} \qquad (5.3.1.1e)$$

$$d_{22} = m_2 l_2\{[\frac{1}{2}l_1\cos\theta_{d2} + \frac{1}{3}l_2(\cos^2\theta_{d2} \quad \sin^2\theta_{d2})]\theta_{d1}^2 \quad \frac{1}{2}g\sin\theta_{d2}\} \qquad (5.3.1.1f)$$

In the simulation, the manipulator moves along a desired trajectory from an initial configuration described by the joint coordinates θ_i $(15^\circ, 0^\circ)$, to a final configuration described by the joint coordinates θ_f $(75^\circ, 90^\circ)$ in 4 seconds A cubic polynomial of the form

$$\theta_i(t) = a_{i0} + a_{i1}t + a_{i2}t^2 + a_{i3}t^3 \qquad i \quad 1,2 \qquad (5.3.1.2)$$

is used for trajectory planning (Craig 1986). In order to test the adaptibility of the control scheme a 10% error is added to the model of the robot manipulator The simulation results are shown in Figure 5 15 - 5 18 The position errors are found to be very small. The maximum position error of θ_1 is below 0 00032 rad and that of θ_2 is below 0 00016 rad Even when there is a 10% error in the model, the position errors of the two joints are very small.

5.3.2 Simulation on PUMA 600 Robot

The adaptive control package was also tested on PUMA 600 Robot. The architecture of the robot are shown in Figure 5 19, while the Hartenberg-Denavit parameters and the dynamical parameters of the robot are shown in Table 5 6 and Table 5.7 , respectively. The joint trajectories for the six joints are of the following form

$$\theta_i \quad \theta_{iT}[t - \frac{1}{2\pi}\sin(\frac{2\pi}{T}t)], \qquad i = 1,\ldots,6 \qquad (5.3.2.1)$$

where T, which is the time period of the whole trajectory, equals to $1(s)$ and θ_{iT} is the final value of each joint variable given by the following equation.

$$\begin{aligned} \theta_{iT} &= \frac{\pi}{3}, & i &= 1,2,4,5,6 \\ &= -\frac{\pi}{6}, & i &= 3 \end{aligned} \qquad (5.3.2.2)$$

In the simulation, the "forgeting factor" ρ and the estimation weighting factor α were chosen to be 0 95 and 0 0001, respectively The simulation sampling period t_s was chosen to be $0.01(s)$. The adaptibility of the control algorithm was tested by introducing 5% and 10% of parameter errors in the model This was done by varying the parameters in the direct dynamic part, namely, the Robot box (sea Figure 4 1), by 5% and 10%, respectively The position and velocity errors were found to be convergent and of very small magnitude, in both cases In the first case, the maximum position and velocity errors are below $0.001(rad)$ and $0.002(rad./s)$, respectively. In the second case, the maximum position and velocity errors are below 0 0012($rad.$) and 0 0035(rad/s), respectively The position errors e_i and velocity errors $\dot{e}_i$ are shown in Figure 5 20 5 43

Table 5.6 Hartenberg-Denavit Parameters of PUMA 600 Robot

joint number	a_i(m)	b_i(m)	α_i(degrees)	initial θ_i(degrees)
1	0 00	0.00	0 00	0 00
2	0.00	-0 149	-90 00	0 00
3	0.432	0 00	0.00	0 00
4	0.02	-0.432	90.00	0 00
5	0.00	0 00	90.00	0.00
6	0.00	-0.056	90.00	0 00

Table 5.7 Inertia Parameters of PUMA 600 Robot

item	joint 1	joint 2	joint 3	joint 4	joint 5	joint 6
$m_i(kg)$	10.521	15 781	8 767	1 052	1.052	0 351
$x_{ci}(m)$	0.0	0.14	0 0	0 0	0 0	0 0
$y_{ci}(m)$	-0 054	0.0	-0.197	0 0	-0.007	0.0
$z_{ci}(m)$	0.0	0.0	0.0	-0.057	-0.007	-0.019
$I_{xxi}(kgm^2)$	1.612	0.4898	3 3768	0 1810	0 0735	0 0071
$I_{xyi}(kgm^2)$	0.0	0.0	0.0	0.0	0.0	0.0
$I_{xzi}(kgm^2)$	0 0	0.0	0.0	0.0	0.0	0.0
$I_{yyi}(kgm^2)$	0.5091	8.0783	0.3009	0.1810	0 0735	0.0071
$I_{yzi}(kgm^2)$	0 0	0.0	0.0	0.0	0 0	0 0
$I_{zzi}(kgm^2)$	1 612	8.2872	3.3768	0 1273	0.1273	0.0141

m_i ....mass of the i-th link

x_{ci}, y_{ci}, z_{ci} ....coordinates of the mass center of the i-th link

$I_{xxi}, I_{xyi}, I_{xzi}$.

$I_{yyi}, I_{yzi}, I_{zzi}$............entries of the inertia matrix of the i-th link

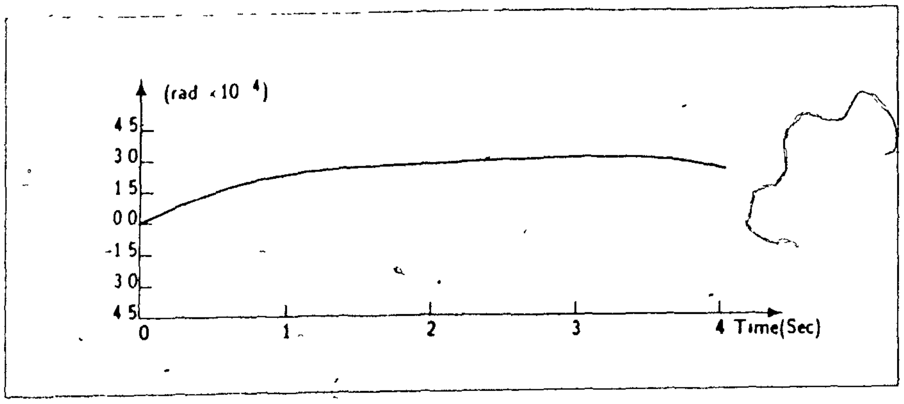

Figure 5.15 Position Error in Joint 1 .

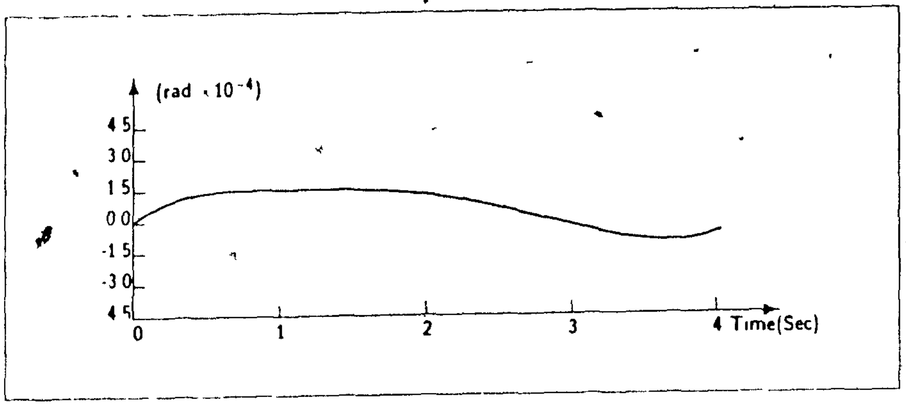

Figure 5.16 Position Error in Joint 2

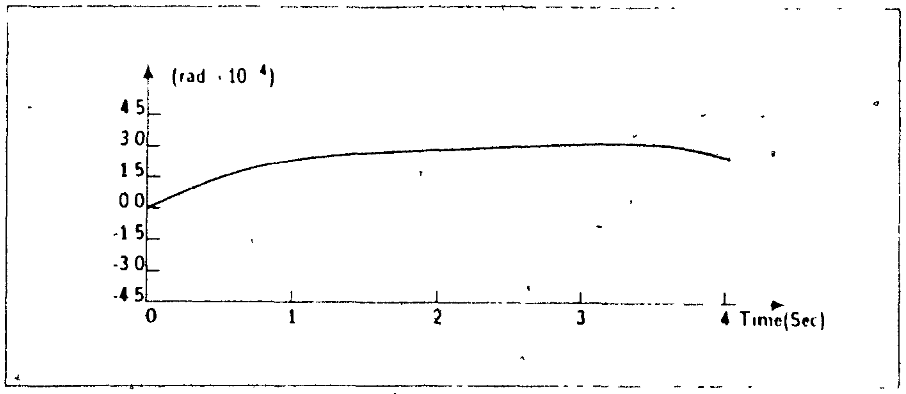

Figure 5.17 Position Error in Joint 1(with 10% error in the model)

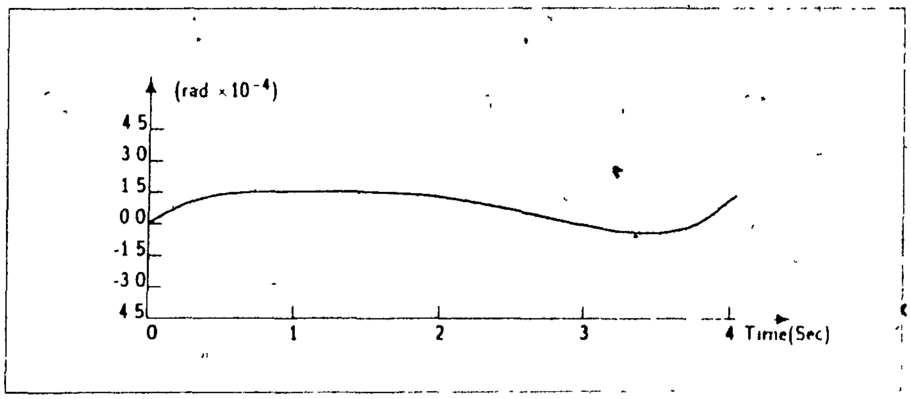

Figure 5.18 Position error in Joint 2(with 10% error in the model)

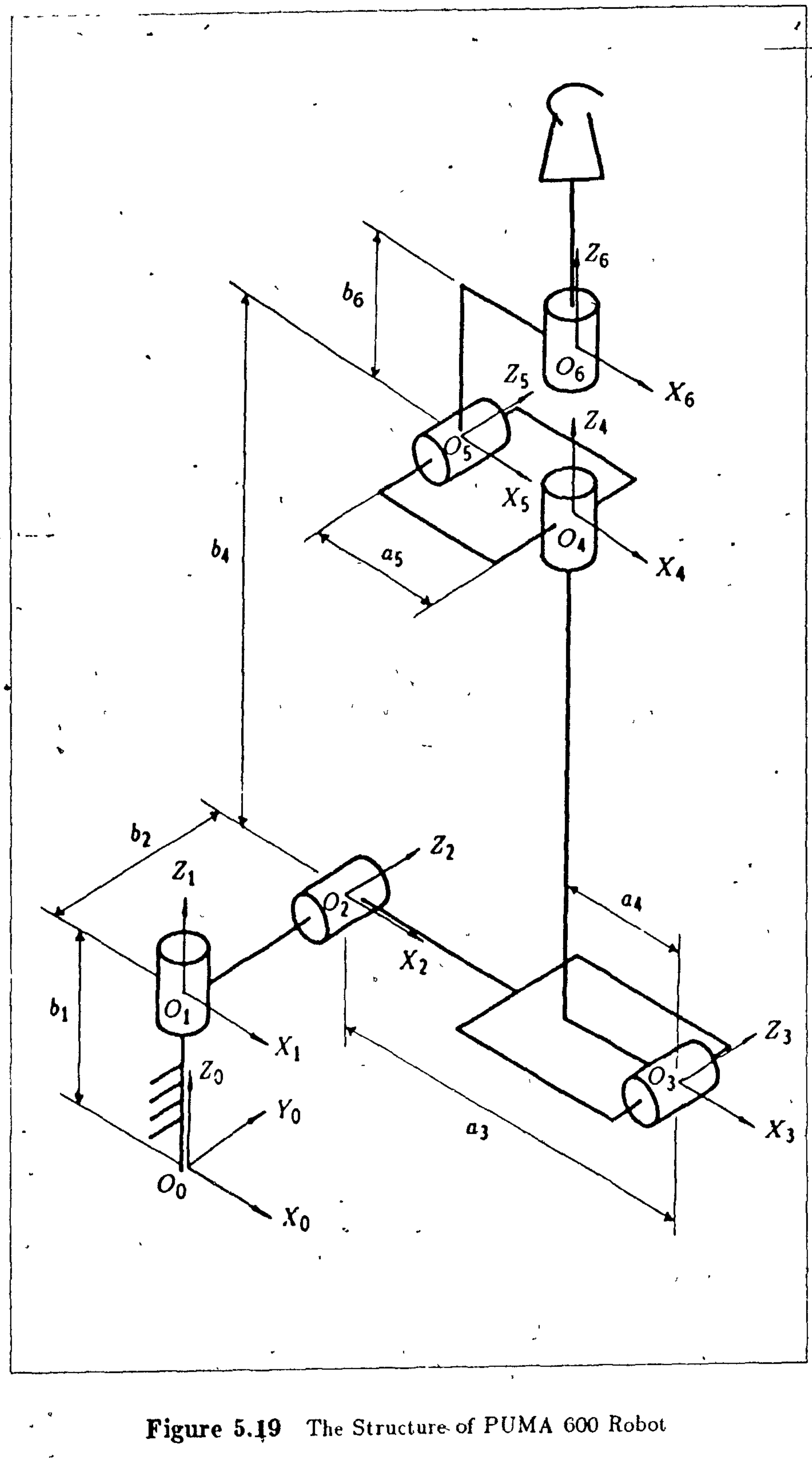

Figure 5.19 The Structure of PUMA 600 Robot

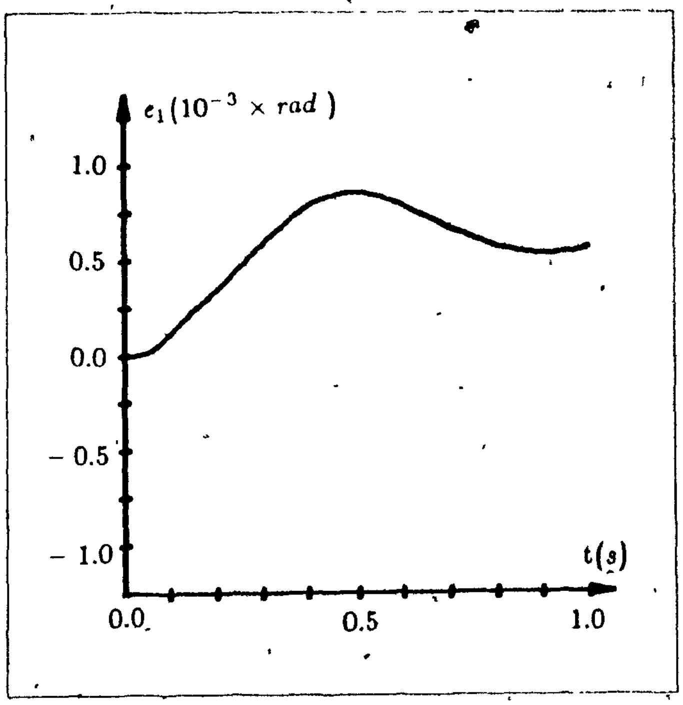

Figure 5.20 Position Error in Joint 1(with 5% error in the model)

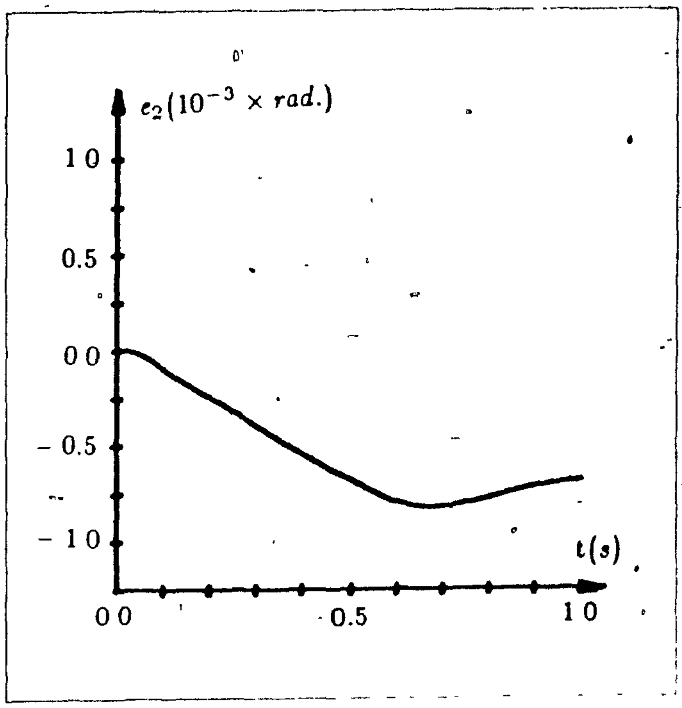

Figure 5.21 Position Error in Joint 2(with 5% error in the model)

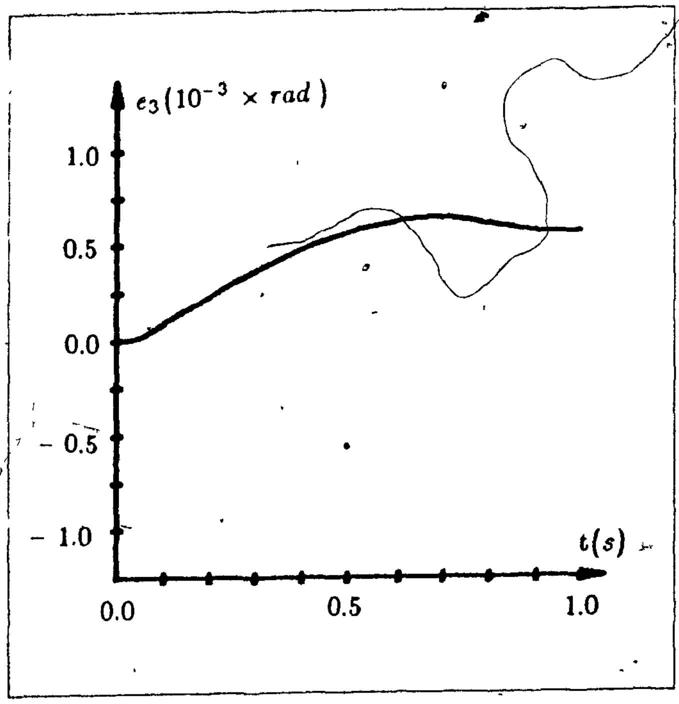

Figure 5.22 Position Error in Joint 3(with 5% error in the model)

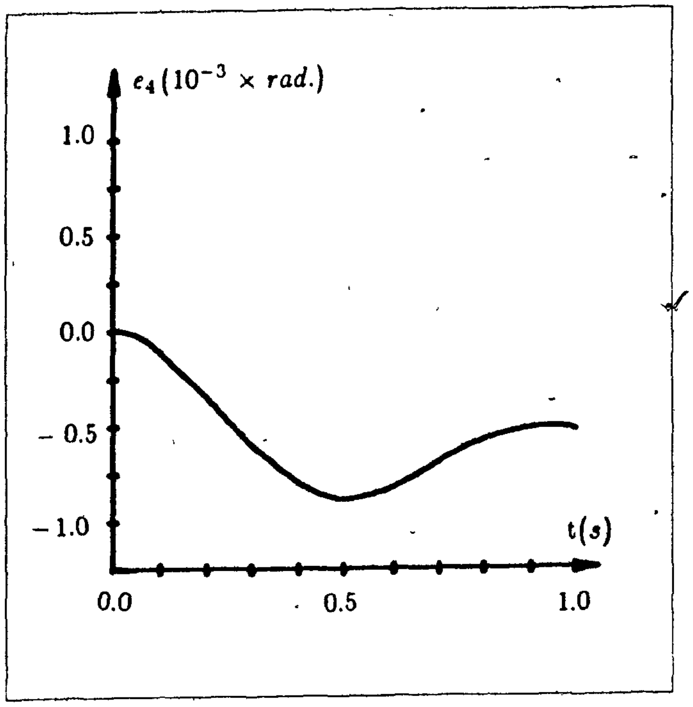

Figure 5.23 Position Error in Joint 4(with 5% error in the model)

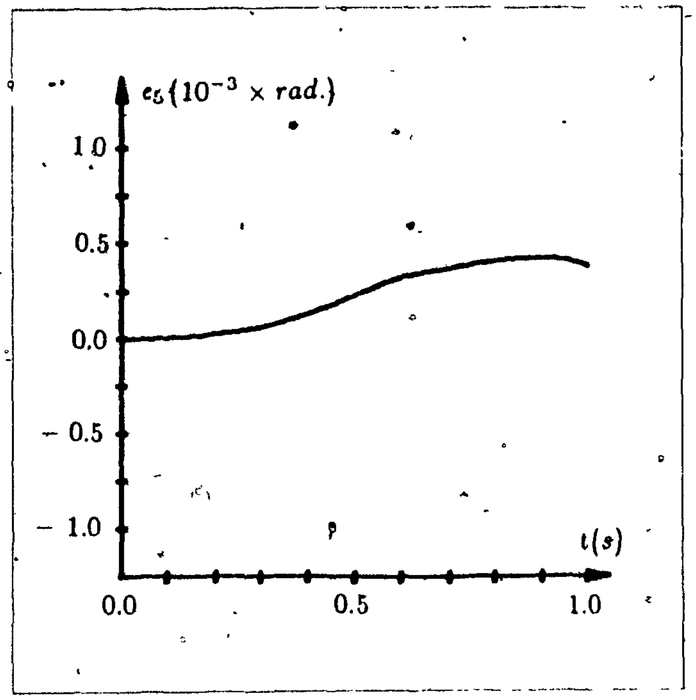

Figure 5.24 Position Error in Joint 5(with 5% error in the model)

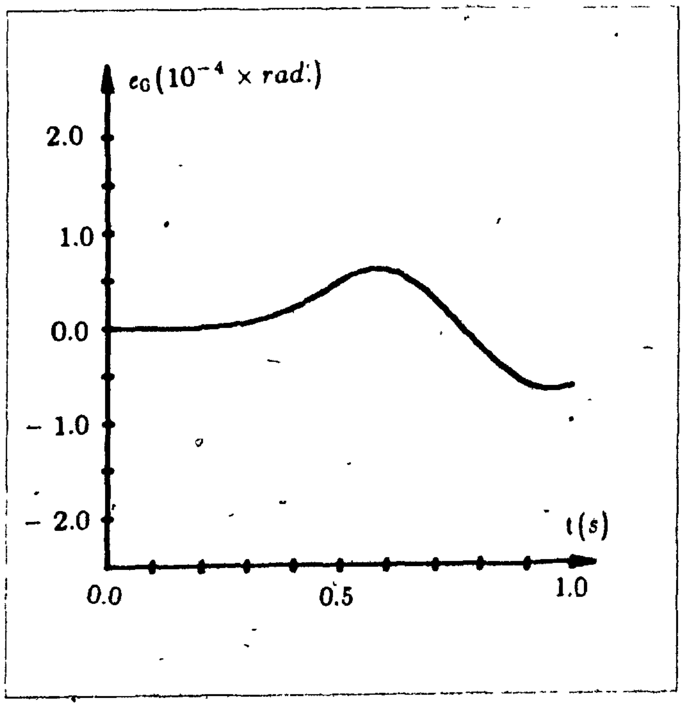

Figure 5.25 Position Error in Joint 6(with 5% error in the model)

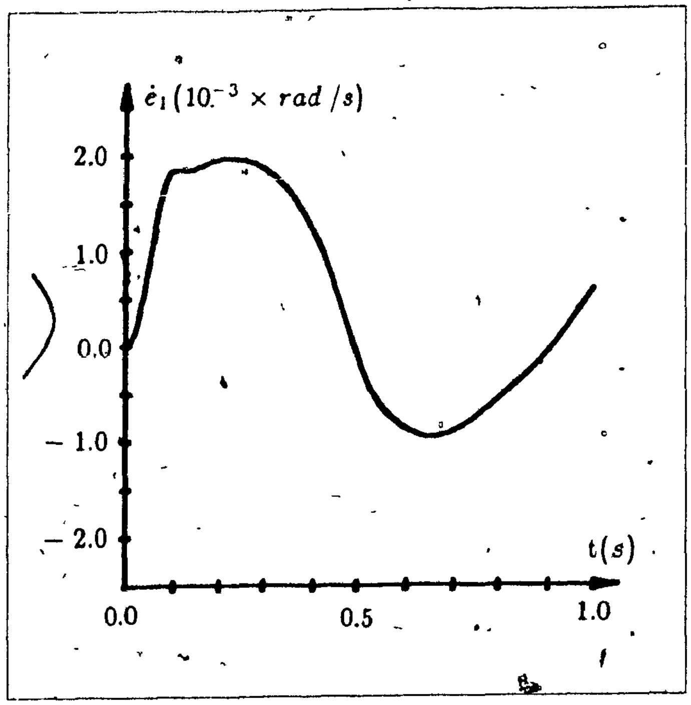

Figure 5.26 Velocity Error in Joint 1(with 5% error in the model)

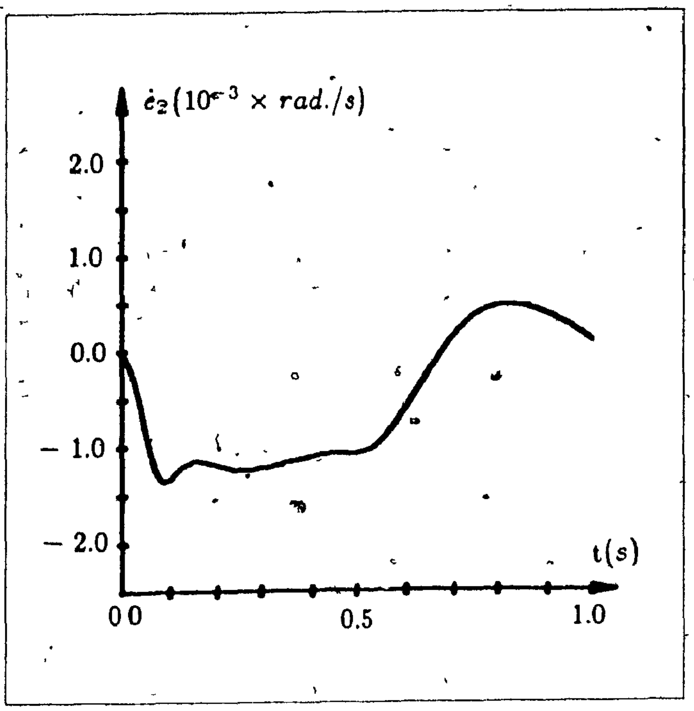

Figure 5.27 Velocity Error in Joint 2(with 5% error in the model)

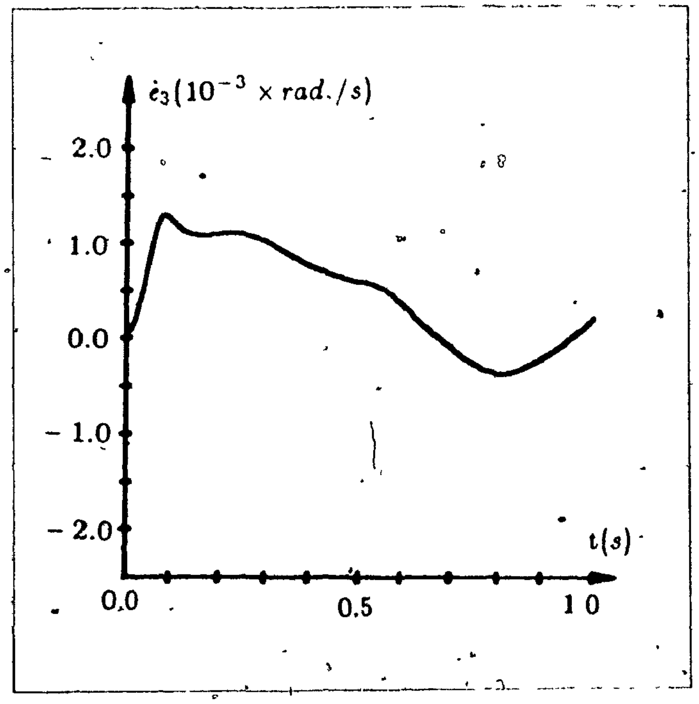

Figure 5.28 Velocity Error in Joint 3(with 5% error in the model)

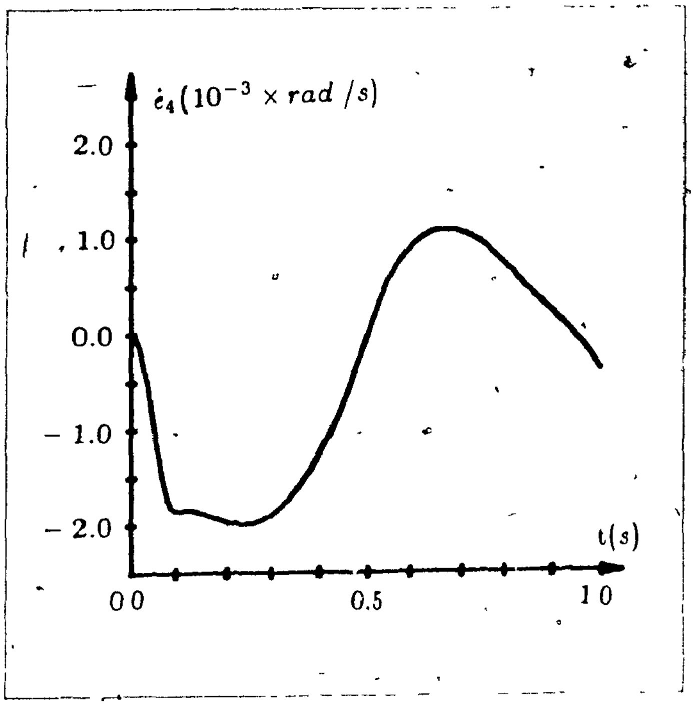

Figure 5.29 Velocity Error in Joint 4(with 5% error in the model)

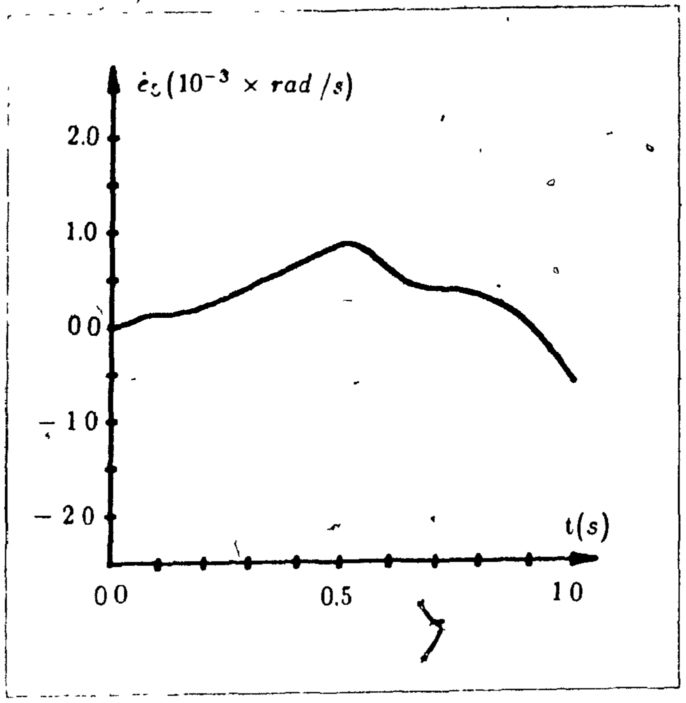

Figure 5.30 Velocity Error in Joint 5(with 5% error in the model)

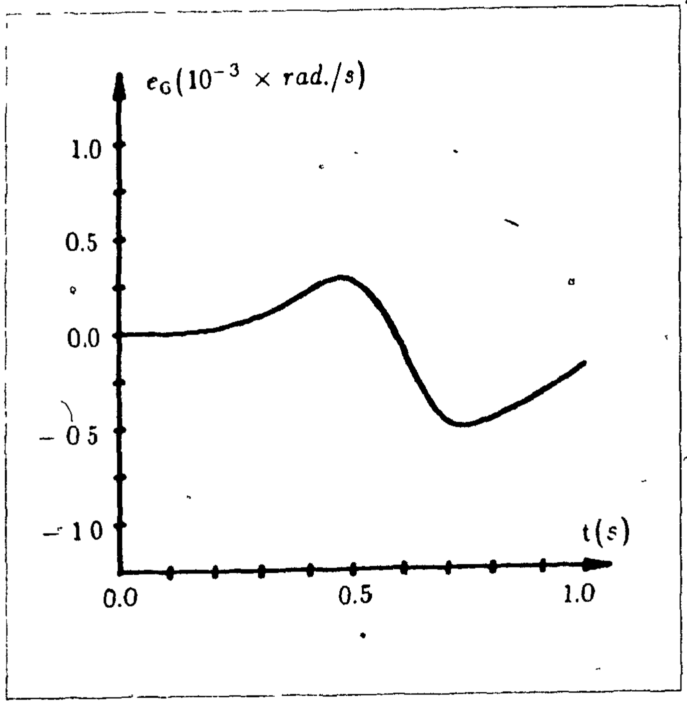

Figure 5.31 Velocity Error in Joint 6(with 5% error in the model)

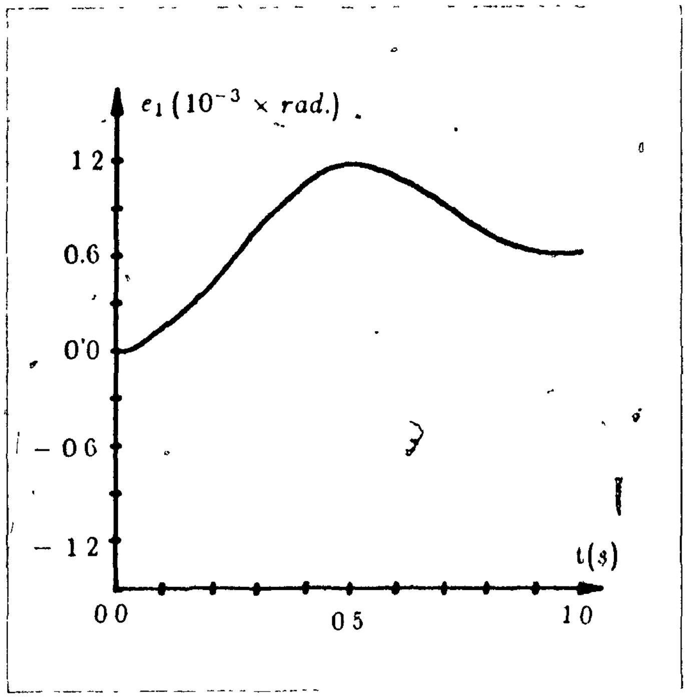

Figure 5.32 Position Error in Joint 1(with 10% error in the model)

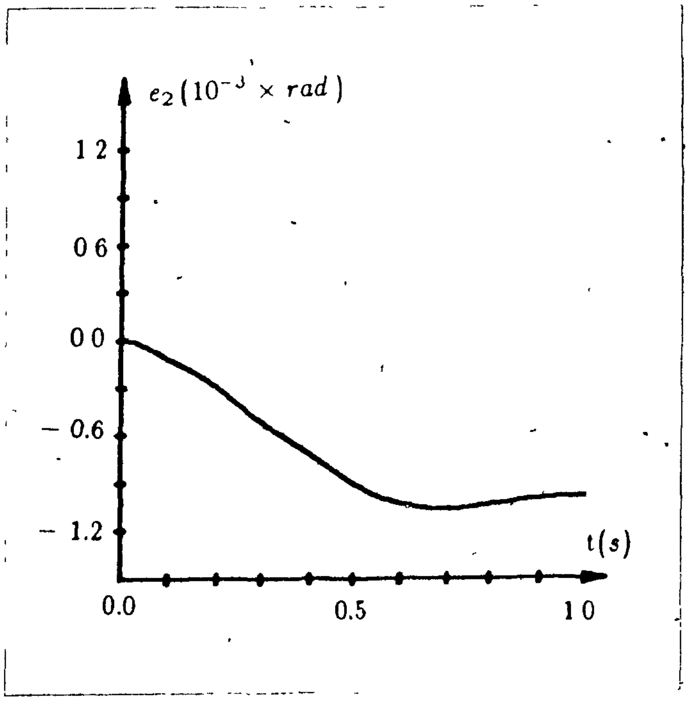

Figure 5.33 Position Error in Joint 2(with 10% error in the model)

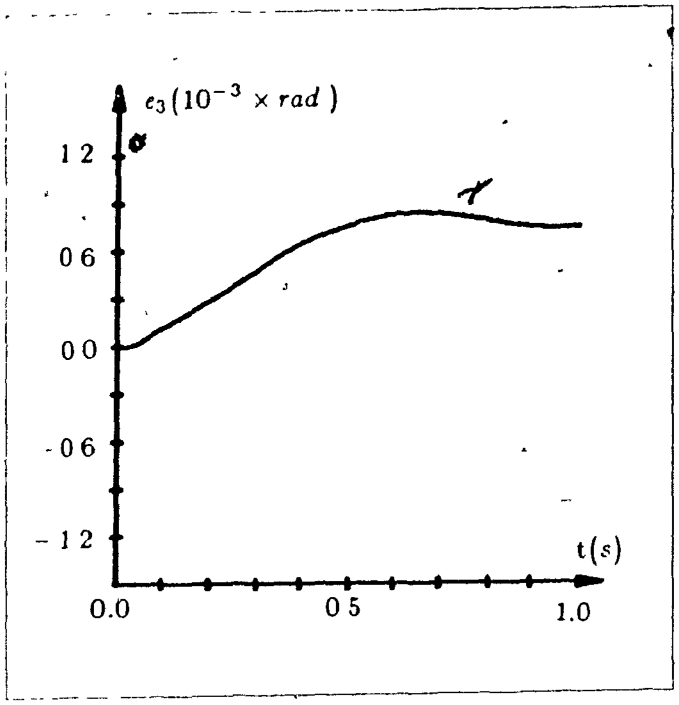

Figure 5.34 Position Error in Joint 3(with 10% error in the model)

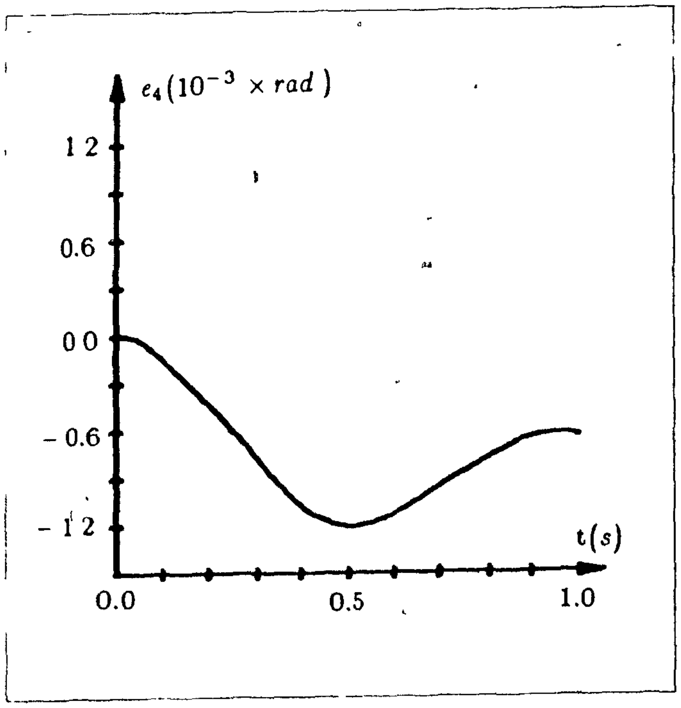

Figure 5.35 Position Error in Joint 4(with 10% error in the model)

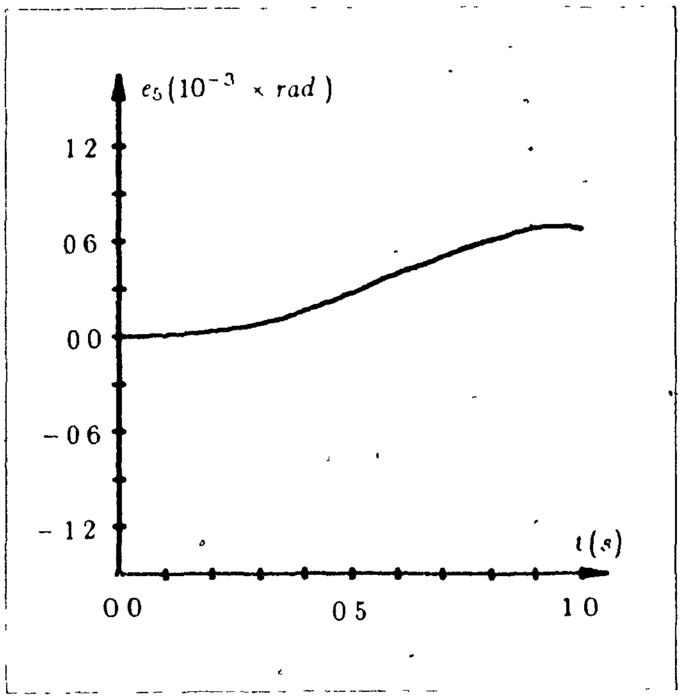

Figure 5.36 Position Error in Joint 5(with 10% error in the model)

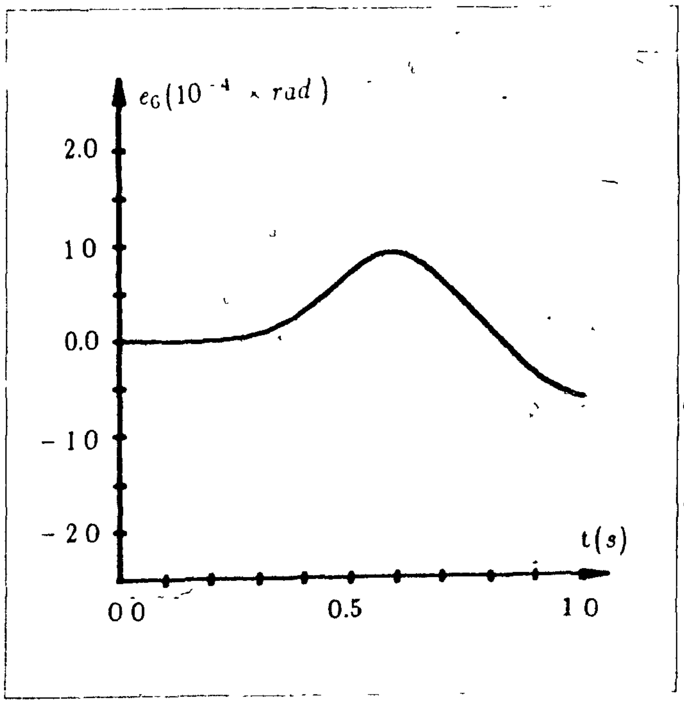

Figure 5.37 Position Error in Joint 6(with 10% error in the model)

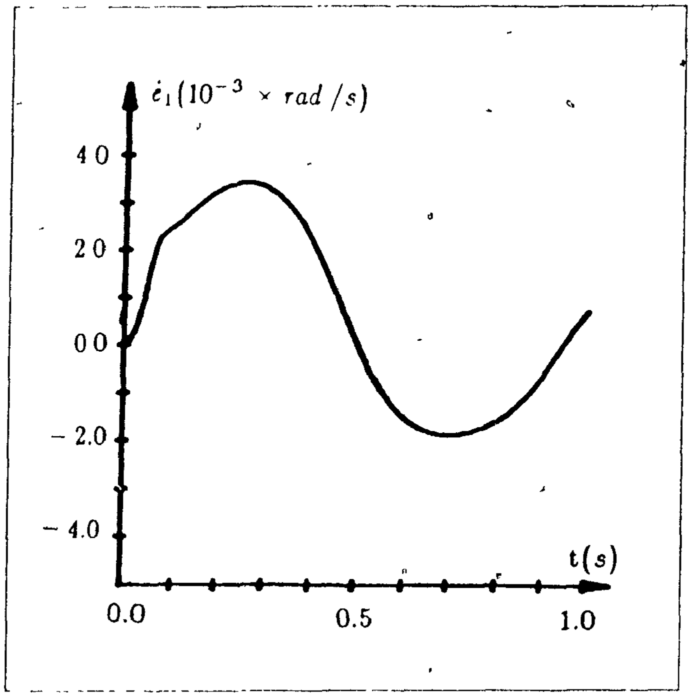

Figure 5.38 Velocity Error in Joint 1(with 10% error in the model)

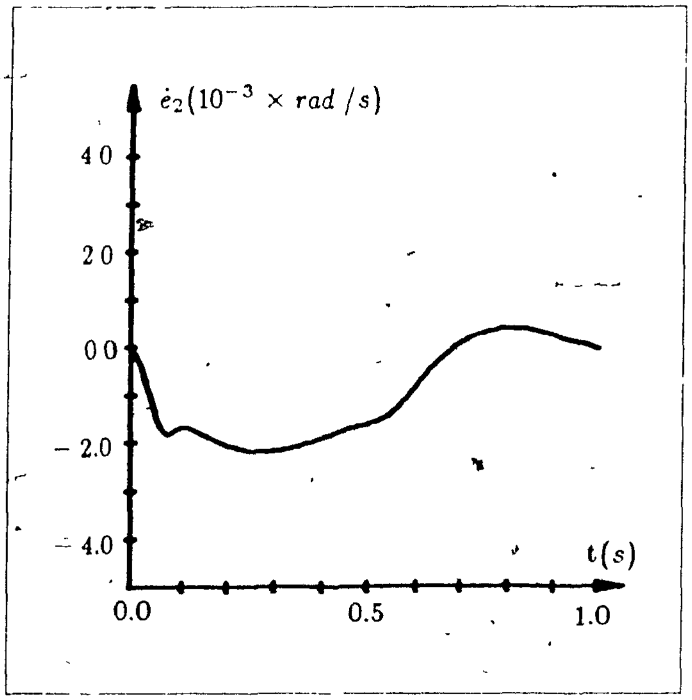

Figure 5.39 Velocity Error in Joint 2(with 10% error in the model)

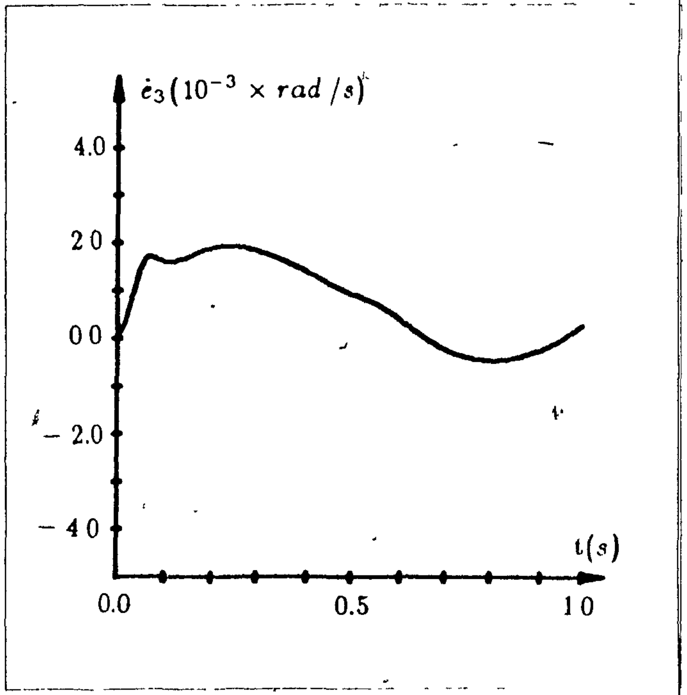

Figure 5.40 Velocity Error in Joint 3(with 10% error in the model)

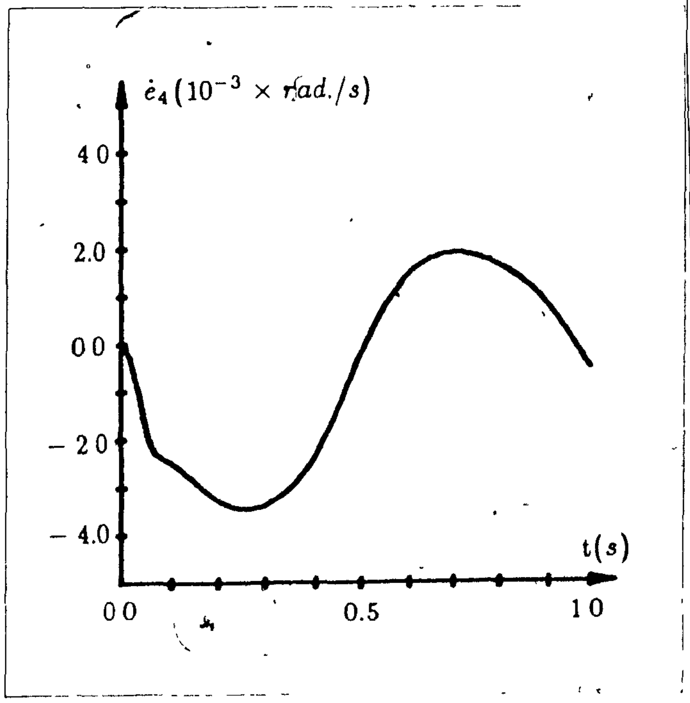

Figure 5.41 Velocity Error in Joint 4(with 10% error in the model)

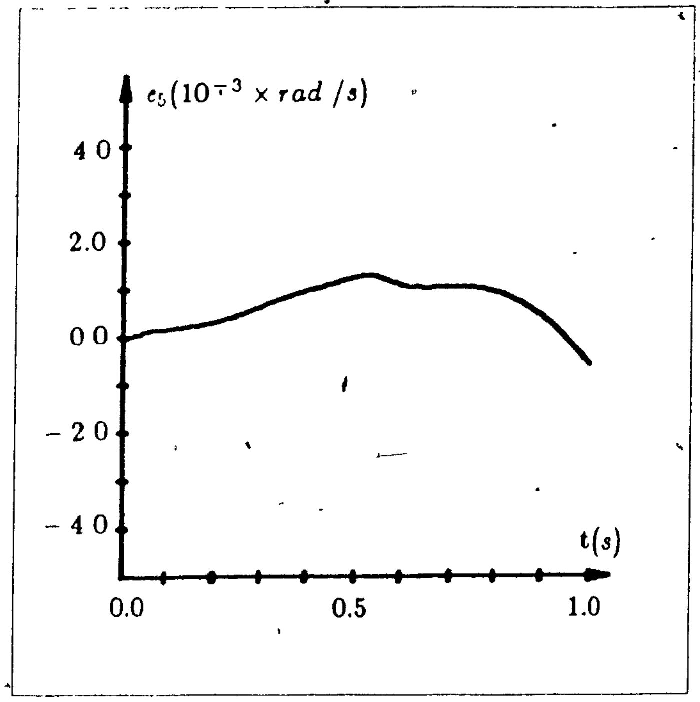

Figure 5.42 Velocity Error in Joint 5(with 10% error in the model)

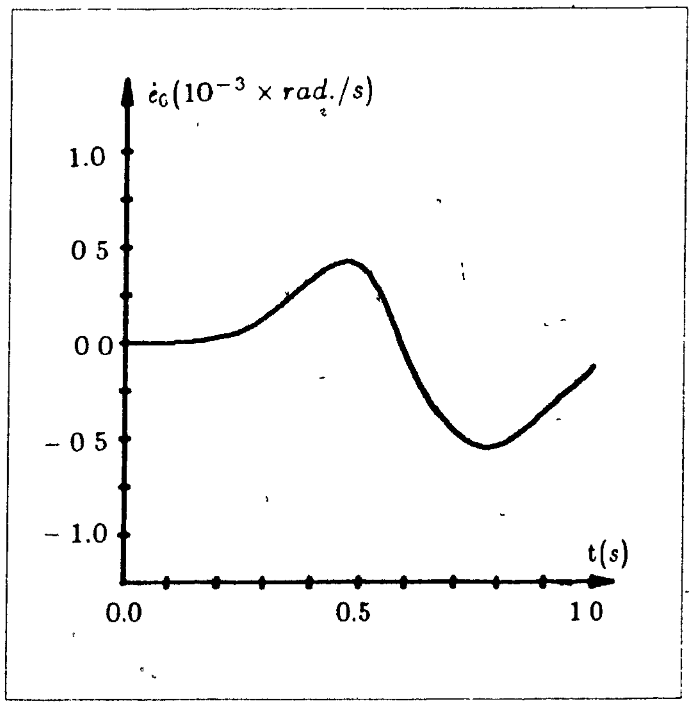

Figure 5.43 Velocity Error in Joint 6(with 10% error in the model)

Chapter 6 Conclusions and Recommendations for Further Research

6.1 Dynamic Modelling Package

In order to gain some insight into the control problem, a dynamic modelling problem was solved and the modelling package was implemented on the VAX-780 (McRCIM). The associated formulae were derived using Kane's dynamical equations in the following form:

$$\tau_i^* + \tau_i = 0 \qquad i = 1, .., n \tag{6.1.1}$$

where τ_i^* and τ_i are the generalized inertia force and the generalized active force, respectively, associated with the i-th joint. The latter is calculated as:

$$\tau_i = \frac{\partial T}{\partial \theta_i} - \frac{d}{dt}\frac{\partial T}{\partial \dot{\theta}_i} + \frac{\partial W}{\partial \theta_i} \qquad i = 1, \ldots, n \tag{6.1.2}$$

where T is the kinetic energy of the manipulator, W is its potential energy, θ_i is the i-th joint variable and n is the number of joints. Each term of eq (6.1.2) is computed numerically. The dynamical model of the robotic manipulator is finally obtained as

$$\tau = \mathbf{I}(\theta)\ddot{\theta} + \mathbf{h}(\theta, \dot{\theta}) \tag{6.1.3}$$

By giving the trajectory for each joint of the manipulator, the generalized inertia matrix $\mathbf{I}(\theta)$, and the vector which describes the inertia, viscous and gravity terms, $\mathbf{h}(\theta, \dot{\theta})$, can be calculated at any given time.

The modelling package was implemented in Fortran-77. Two examples were performed for the testing of the package.

In the first example, a two-link manipulator with revolute joints was used. The results from the numerical method, namely, the method based on the above algorithm, and the closed-form dynamical model were compared with each other. The errors were found to be very small. The maximum error is only 0.0001(Nm).

As to the second example, the test was performed on the Cincinnati Milacron Robot, which has six revolute joints. The results are in full agreement with those obtained using Ma Ou's modelling package(1987), which was developed independently.

6.2 Model-Assignment

A feedback control scheme, called model-assignment control, is derived for linearized models of robot manipulators This method is based on the linearized model of the manipulator as follows:

$$\mathbf{I}(t)\delta\ddot{\theta} + \mathbf{C}(t)\delta\dot{\theta} + \mathbf{D}(t)\delta\theta = \delta\tau(t) \tag{6.2.1}$$

Then, the position and velocity feedback controller is introduced as:

$$\delta\tau(t) = -\mathbf{K}_p(t)\delta\theta - \mathbf{K}_d(t)\delta\theta \tag{6.2.2}$$

The feedback gains are designed to render the system a prescribed second-order stationary system of the following form:

$$\delta\ddot{\theta} + 2\mathbf{\Psi}^T\mathbf{\Omega}_n\mathbf{\Psi}\delta\dot{\theta} + \mathbf{\Omega}_n^2\delta\theta = 0 \tag{2.4.2}$$

where $\mathbf{\Psi}$ is the *nondimensional* $n \times n$ *damping matrix* and $\mathbf{\Omega}_n$ is the $n \times n$ *matrix of undamped natrual frequency* of the system.

In this case, the time-varying terms in the linearized robot model are cancelled by a corresponding time-varying control term.

The feedback gains are derived then as

$$\mathbf{K}_d(t) = 2\mathbf{I}(t)\mathbf{\Psi}^T\mathbf{\Omega}_n\mathbf{\Psi} - \mathbf{C}(t) \tag{6.2.3}$$

$$\mathbf{K}_p(t) = \mathbf{I}(t)\mathbf{\Omega}_n^2 - \mathbf{D}(t) \tag{6.2.4}$$

Compared with other approaches, like pole-assignment, this method has the advantage of leading to a simpler algorithm and needs less computations. Moreover, this scheme is not limited to stationary systems, whereas pole-assignment is. The results of the numerical example show that the gains obtained with this procedure are of a reasonable magnitude.

6.3 Controller Estimation for the Adaptive Control of Robotic Manipulators

A new method of adaptive control of robot manipulators is derived. The suitable controller is designed by model assignment. The estimation scheme is introduced to identify the unknown gains in the controller.

The controller estimation is based on the following discrete-time equation.

$$\mathbf{y}(k+1) = \mathbf{K}(k)\phi(k) \tag{6.3.1}$$

In this equation, matrix $\mathbf{K}$ contains the feedback gains to be estimated and ϕ contains the input and output states of the system. A recursive least-squares estimation scheme was applied to eq (6.3.1). Since the robotic system is a time-varying system, an estimation scheme which is suitable for the time-varying systems is derived as follows, to perform the parameter estimation of the feedback gains

$$\tilde{\mathbf{z}}_i(k+1) = \tilde{\mathbf{z}}_i(k) + r(k+1)\mathbf{P}(k)\phi(k+1)[y_i(k+1) - \phi^T(k+1)\tilde{\mathbf{z}}_i(k)] \tag{6.3.2a}$$

$$\mathbf{P}(k+1) = \mathbf{P}(k) - r(k+1)\mathbf{P}(k)\phi(k+1)\phi^T(k+1)\mathbf{P}(k) \tag{6.3.2b}$$

$$r(k+1) = [\rho + \phi^T(k+1)\mathbf{P}(k+1)\phi(k+1)]^{-1} \tag{6.3.2c}$$

where $\phi^T(k+1)\tilde{\mathbf{z}}_i(k+1)$ is the predicted value of $y_i(k+1)$, based on the estimated value $\tilde{\mathbf{z}}_i(k)$. The corrections are performed by $r(k+1)\mathbf{P}(k+1)\phi(k+1)[y_i(k+1)-\phi^T(k+1)\tilde{\mathbf{z}}_i(k)]$ and $\phi(k)$ contains the unknown parameters of the feedback gains, which are to be estimated.

Compare with Lee's parameter estimation method(1984), this method has the advantage of reducing the computations and simplifying the control algorithm.

By using the controller estimation scheme, the number of the unknown parameters, which are estimated, is reduced to $2n^2+n$, while the number of the estimated parameters in Lee's method was $6n^2$ For a six-link manipulator, the number of unknown parameters is reduced from 216 to 78. Because the controller is obtained from the estimation directly, it is not necessary to design the controller at each sampling period. Therefore, the control algrithm is much simpler.

The simulations of a two-link robot manipulator and the PUMA 600 robot were performed on a VAX-780 computer Satisfactory results were obtained. The adaptibility of the controller estimation scheme has been demonstrated by the convergent and small-error results in the presence of errors in robot model during the simulation.

6.4 Recommendations for the Further Research

(1) The controller estimation method can be extended. The controller estimation scheme can be applied not only for the model-assignment scheme but for other optimum control algorithms.

(2)The estimation weighting factor α in the state vector ϕ expressed in eq.(4.3.2.12) might be obtained by using an on-line identification scheme This can probably done by introducing some stochastic theorem.

References

Angeles J., Alivizatos A. and Zsombor-Murray P.J. "The Synthesis of Smooth Trajectories for Pick-and-Place Operations", Proceeding of the 1986 IEEE International Conference on System, Man, and Cybernetics,pp666 - 670

Angeles J., " The computer implementation of manipulator Euler-Lagrange equations from a Newton-Euler formulation", *RMSL-McRCIM Technical report, McGill University*, 1986

Angeles J., "On the Numerical Solution of the Inverse Kinematic Problem", *The International Journal of Robotics Research* Vol. 4 No. 2, Summer 1985, pp. 21 37

Åstrom K J and Wittenmark B., *Computer -Controlled Systems:Theory and Design* Pretice-Hall, Inc Englewood Cliffs, N J 1984

Åstrom K J. and Wittenmark B , *Computer -Controlled Systems:Theory and Design* Pretice-Hall, Inc Englewood Cliffs, N J 1984

Bhattacharyya S P and E de Souza, "Pole Assignment Via Sylvester's Equation", System and Control Letters, Vol 1 No 4, pp261-263, 1983

Brady M , Hollerbach J M , Johnson T L , Pérez T. L. and Mason M T., *Robot Motion: Planning and Control* The MIT Press 1982.

Cavin R K. and Bhattacharyya S P , "Robust and Well-Conditioned Eigenstructure Assignment Via Sylvester's Equation", Optimal Control Applications & Methods, Vol. 4, pp205—212, 1983

Craig J J., *Introduction to Robotics Mechanics & Control* Addison-Wesley Publishing Company, Reading, Massachusetts, 1986

Davidson E J ,"On Pole Assignment in Linear Systems with Incomplete State Feedback", IEEE Trans on Automatic Control Junio, pp348 351 1970

Dubowsky S., Kornbluh R., "On the development of high performance adaptive control algorithms for robotics manipulator"

Eykhoff P., *System identification :Parameter and state Estimation* Wiley-Interscience, New York, 1974

Guo L. and Angeles J., "Controller Estimation for Linearized Models of Robot Manipulators", Proceedings of the Eighteenth Annual Pittsburgh Conference on Modelling and Simulation, Pittsburgh 1987

Hewit R.J. and Burdess J.S., " Fast Dynamic Decoupled Control for Robotics Using Active Force Control",*Mechanism and Machine Theory*, Vol.16 No.5, pp.535 542,1981

Hollerbach J. M. 1980, 'A recusive lagrangian formulation of manipulator dynamics and a comparitive study of dynamics formulation complexity', IEEE Trans on System, Man and Cybernetics, Vol SMC-10 pp 730—736.

Hsia T.C. *System Identification*, Lexington Books, 1977.

Kane T. R. and Levinson D. A., 1985, Dynamics: Theory and applications, McGraw-Hill Book Company.

Keel L.H., Fleming J.A. and Bhattacharyya S.P.," Minimum Norm Pole Assignment Via Sylvester's Equation", Contemporary Mathematics, Vol. 47, pp265 - 272, 1985

Koivo A. J. and Guo T H., "Adaptive linear controller for robotic manipulators", *IEEE Trans. Aut Control* ,AC28(2), pp.162-170,1983

Lee C.S.G., Gonzalez R.C. and Fu K.S., 1983,'Tutorial on Robotics' IEEE Computer Society.

Lee C.S.G. and Chung M J. (1984) "An adaptive control stratgy for computer-based manipulators", *IEEE Trans.Aut. Control* AC-29 N.9 1984

Lee C.S.G., Chung M.J.and Lee B.H.. "Adaptive control for robot manipulators in joint and Cartesian coordinates", *IEEE 1984 International Conference On Robotics*, pp. 530-540

Lin C. F., "Adaptive Controller Design for Robot Arms", *IEEE Trans. and Aut. Control*, Vol.AC-29,No.4, pp.350-353,1984

Luh J.Y.S., Walker M.W. and Paul R.P., 1980, "On-line Computational Scheme Mechanical Manipulators", Journal of Dynamic System, Measurement and Control, Trans. ASME, vol 102, pp 69–76.

Ma O., "Dynamics of Serial Robot Manipulators", Master Thesis, McGill University, 1987.

Reid J.G., *Linear Systems Fundamentals*, McGraw-Hill Book Company, 1983

Rojas A. A., Dinámica de Sistemas Articulados de Cuerpos Rigidos, Ph.D. Thesis, National Autonomous University of Mexico, Mexico City, June 1987

Seraji H., "Direct Adaptive Control of Manipulator in Cartesian Space" *Jet Propulsion Laboratory, Engineering Memorandum* 1986

Seraji H., "Adaptive Control of Robotic Manipulator" *Jet Propulsion Laboratory, Engineering Memorandum* 347–182, January 1986

Slotine J.J.E., "The Robust Control of Robot Manipulators" *The International Journal of Robotics Research*, Vol.4, No.2, Summer 1985, pp.49–64

Uicker J.J. 1969 , 'Dynamic Behavior of Spatial Linkages', Trans. ASME, Journal Eng. for Industry, serie B, No. 91 251—265

Wonham W.M., "On Pole Assignment in Multi-Input Controllable Linear Systems", IEEE Trans. on Automatic Control, vol. AC-12, pp 660—665, 1967

2 *Robotics Research (Haunfusa and Inone, editors)*, pp.119-126 1985

Appendix A

LEMMA Let **A**,**C**, and **A**+**BCD** be nonsigular square matrices, then the matrix identify

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1} \qquad (A.1)$$

holds.

Proof: Premultiply both sides of eq.(A.1) by **A** + **BCD**,

$$\mathbf{I} = (\mathbf{A} + \mathbf{BCD})[\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1} \qquad (A.2)$$

The objective now is to prove the identity eq.(A.2). In other words, we wish to show that the right hand side of eq.(A.1) can be reduced to an identity matrix. By direct manipulation, we get

$$\begin{aligned}
&[\mathbf{A} + \mathbf{BCD}][\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1}] \\
&\quad = \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1} \\
&\quad - \mathbf{BCDA}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1} \\
&\quad = \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{B}[\mathbf{I} + \mathbf{CDA}^{-1}\mathbf{B}][\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]\mathbf{DA}^{-1} \\
&\quad = \mathbf{IBCDA}^{-1} - \mathbf{BC}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}][\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\mathbf{DA}^{-1} \\
&\quad = \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{BCDA}^{-1} \\
&\quad = \mathbf{I}
\end{aligned}$$

Appendix B

Adaptive Control Package

```
C************ADAPTIVE PACKAGE FOR 6-LINK MANIPULATORS************
C                (USING CONTROLLER ESTIMATION SCHEME)
C
C        IN THIS PACKAGE, A NEW APPRAOCH OF ADAPTIVE CONTROL IS
C        IMPLEMENTED  THE CONTROLLER OF THE ROBOT MANIPULATOR
C        IS ESTIMATED BY USING LEAST-SQUARE SCHEME. THIS PACKAGE
C        IS IMPLEMENTED BY GUO LIN IN MAY 1987.
C   USER'S GUIDE;
C        (I) CREAT A INDIVIDUAL DATA FILE WHICH CONTAINS H-D
C            PARAMETERS AND DYNAMIC PARAMETERS OF THE ROBOT
C            MANIPULATOR WHICH IS TO BE TESTED.
C        (II) WRITE A SUBROUTINE NAMED 'TRAJ' TO CALCULATE
C             THE CHOSEN TRAJECTORY IN THE FOLLOWING FORM.
C                SUBROUITNE JOTRAJ(T,THETAD,THDOTD,THDDOD)
C                 ....
C                RETURN
C                END
C             WHERE THETAD,THDOTD,THDDOD ARE THE N-DIMENSIONAL
C             VECTORS OF THE DESIRED POSITION,VELOCITY AND
C             ACCELERATION OF THE ROBOT IN THE JOINT SPACE.
C        (III) INPUT THE SAMPLING PERIOD, SIMULATION TIME AND
C              NUMBER OF THE LINKAGE  (N=1,  ..6) ON THE KEYBOARD
C
C        (IV)THE DYNAMICAL SIMULATOR USDE IN THE SIMULATION OF
C            THIS CONTROL PACKAGE IS IMPLEMNETED BY OU MA
C            (FOR DETAILS SEE MA'MASTER THESIS 1987)
C
C*****************************************************************


C------------------------MAIN PROGRAM----------------------------
C
C   INPUT:
C        TT---------SAMPLING PERIOD
C        TZ---------TATOL SIMULATION TIME
C        N----------LINKAGE NUMBER
C
C   OUTPUT:
C        E----------POSITION ERROR OF EACH JOINT
C        ED---------VELOCITY ERROR OF EACH JOINT
C
C

         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
         DOUBLE PRECISION IMO(3,3,6),MO(6)
         DIMENSION THETA(6),THDOT(6),THDDOT(6),THETAD(6),THDOTD(6),
     *             THDDOD(6),DTAU(6),AO(6),BO(6),SAO(6),CAO(6),
     *             GO(3),RHO(3,6),IPO(6),E(6),ED(6),EDD(6)
         COMMON /CON/DTAU,TT,TZ
         OPEN(20,FILE='PSE DAT',STATUS='UNKNOWN')
         OPEN(30,FILE='VLE.DAT',STATUS='UNKNOWN')
C
C INPUT SIMULATION PARAMETERS ON THE KEYBOARD AND SET INITIAL VALUES
C
         WRITE(*,*) 'INPUT SAMPLING PERIOD AND SIMULATION TIME'
         READ(*,*) TT,TZ
         WRITE(*,*) 'PLEASE INPUT THE NUMBER OF THE LINKAGE'
         READ(*,100) N
         WRITE(*,*) 'PLEASE INPUT ALPHA'
         READ(*,*) AL
100      FORMAT(I3)
```

```
        WRITE(*,*) 'SAMPLING PERIOD   TT=',TT
        WRITE(*,*) 'SIMULATION TIME   TZ=',TZ
        WRITE(*,110) 'NUMBER OF LINKAGE',N
110     FORMAT(A17,I3)
        IJ=1
        IN=0
        T1=0.D00
        T2=0.D00
        IND=1
        TOL=0.0000001D00
        DO 30 I=1,N
          DTAU(I)=0.D00
30      CONTINUE
C
C READ IN H-D PARAMETERS AND DYNAMICAL PARAMETERS
C
        CALL READAT('puma.input',IJ,N,IPO,AO,BO,SAO,CAO,RHO,GO,MO,IMO)
        print *,'Just before calling READAT: T1, T2 =',t1,t2
C
C START SIMULATION
C
        IS=1
        IND=1
        TOL=0.000001D00
        nn=0
200     T1=T2
        T2=T2+TT
c       print *,'Just before calling JOTRAJ: T1, T2 =',t1,t2
        CALL JOTRAJ(T2,THETAD,THDOTD,THDDOD)
C       CALL TRAJ(T2,TT,TZ,THETAD,THDOTD,THDDOD)
c       print *,'Just before calling DSIMU: T1, T2 =',t1,t2
        CALL DSIMU(T1,T2,N,THETA,THDOT,THDDOT,TOL,IND,IS)
        print *,'Just after calling DSIMU',n
        NI=6
        CALL CONT(AL,NI,IN,THETA,THDOT,THDDOT,THETAD,THDOTD,THDDOD,
       *E,ED,EDD,MO,IMO,AO,BO,SAO,CAO,IPO,RHO)
C
C  OUTPUT RESULTS
C
        write(*,*) 't=',t2
        write(*,*) 'position error'
        WRITE(20,121) t2,E,t2
        write(30,121) t2,ed,t2
121     format(f4.2,1x,f10.7,1x,1x,f10.7,1x,f10.7,1x,f10.7,1x,
       *f10.7,1x,f10.7,1x,f4.2)
120     FORMAT(6f12.9)
        IF(T2.GE.TZ) THEN
        GOTO 300
        ELSE
        IN=IN+1
        GOTO 200
        END IF
300     STOP
        END


        SUBROUTINE JOTRAJ(T,THETA,THDOT,THDDOT)
C**********************************************************************
C    Compute the positions, velocities, and accelerations in joint space
C    with arbitrary given time t
C**********************************************************************
        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
```

```
C
C   Arrays used to hold joint trajectory data
        DIMENSION ATRAJ(6),THETA(6),THDOT(6),THDDOT(6)
C
C  Ma Ou's H-D notation and \theta notation
c            DATA  THETAO/  0.D00,   0.D00,    0.D00,    0.D00,    0.D00,    0.D00/
c            DATA  THT/    60 D00,  -30.D00,   60.D00,   60.D00,   60.D00,   60.D00/
C
C
C
        PI2=DASIN(1 D00)
        DD=PI2/90.D00
C
C   TT stands for time period of the whole motion procedure
C
        TT=1.D00
        ATRAJ(1)=60 D00*DD/TT
        ATRAJ(2)=60.D00*DD/TT
        BTRAJ=4*PI2/TT
C
C
C   Compute the tensor of inertia
C
C
C
C
C   In the following computation, T denotes the time t
C
        BT=BTRAJ*T
        SBT=DSIN(BT)
        DO 30   I=1, 2
           THETA(I)=ATRAJ(I)*(T-SBT/BTRAJ)
           THDOT(I)=ATRAJ(I)*(1.D00-DCOS(BT))
 30        THDDOT(I)=ATRAJ(I)*BTRAJ*SBT
        DO 40   I=3, 6
           THETA(I)=THETA(1)
           THDOT(I)=THDOT(1)
 40        THDDOT(I)=THDDOT(1)
c       WRITE(*,303) NO,T,THETA,THDOT,THDDOT
 303    FORMAT(//2X,'Point No. =',I3,5X,'Time :  t =',F8.4/
     *           2X,'THETA (1, 2, . . , 6) =',6F9.5/
     *           2X,'THDOT (1, 2, ..., 6) =',6F9.5/
     *           2X,'THDDOT (1, 2, ..., 6) =',6F9.5/)
        RETURN
        END
```

```
        SUBROUTINE TORQUE(T,TAU,N)
C*********************************************************************
C       THIS SUBROUTINE IS USED TO CALCULATE THE TORQUE FOR EACH JOINT
C
C   INPUT
C       T------------GIVEN TIME
C       N------------NUMBER OF THE JOINTS
C   OUTPUT
C       TAU----------N-DIMENSIONAL VECTOR CONTAINS TORQUES(FORCES)FOR
C                    THE JOINTS
C
C*********************************************************************
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION IM,M,ML,NL
      DIMENSION TAU(6),IP(6),A(6),B(6),SA(6),CA(6),RO(3,6),G(3),
     *          Q(3,3,6),AA(3,6),M(6),IM(6),ML(6),NL(6),DTAU(6),
     *          THETAD(6),THDOTD(6),THDDOD(6),a1(6)
      COMMON /HDDATA/IP,A,B,SA,CA,Q,AA,NT
      COMMON /INDATA/RO,G,M,IM,ML,NL
      COMMON /CON/DTAU,TT,TZ
C     CALL TRAJ(T,TT,TZ,THETAD,THDOTD,THDDOD)
      CALL JOTRAJ(T,THETAD,THDOTD,THDDOD)
C
C  INTRODUCE ERRORE IN THE PARAMETERS OF THE SYSTEMS
C
      do 200 i=1,6
        a1(i)=a(i)*1.10d00
200   continue
      CALL DYNAK1(N,IP,A1,B,SA,CA,RO,G,THETAD,THDOTD,THDDOD,
     *            M,IM,TAU)
      DO 100 I=1,6
        TAU(I)=TAU(I)+DTAU(I)
100   CONTINUE
      RETURN
      END


      SUBROUTINE CONT(AL,N,IN,THETA,THDOT,THDDOT,THETAD,THDOTD,THDDOD,
     *                E,ED,EDD,MO,IMO,AO,BO,SAO,CAO,IPO,RHO)
C--------------------CONTROL ROUTINE USING LEAST-SQUARE METHOD---------
C-----------------------------TO ESTIMATE FEEDBACK GAINS---------------------
C************************************************************************
C
C   INPUT:
C      N,IN,THETA,THDOT,THDDOT,THETAD,THDOTD,THDDOD,TT,PS,BOM
C   OUTPUT:
C      E-----------POSITION ERROR OF EACH JOINT
C      ED----------VELOCITY ERROR OF EACH JOINT
C      EDD---------ACCERATION EEROR OF EACH JOINT
C      DTAU--------DELTA TORQUE ON EACH JOINT FORM THE FEEDBACK CONTROLLER
C
C************************************************************************
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION THETA(6),THDOT(6),THDDOT(6),THETAD(6),
     *          THDOTD(6),THDDOD(6),
     *          AO(6),BO(6),SAO(6),CAO(6),RHO(3,6),
     *          E(6),ED(6),EDD(6),PS(6,6),
     *          BOM(6,6),IPO(6),DTAU(6),GI(6,6)
      DOUBLE PRECISION IMO(3,3,6),MO(6),KK(6,13)
      COMMON /CON/DTAU,TT,TZ
      CALL IDEN(PS,6)
      CALL IDEN(BOM,6)
      S=DSQRT(2.D00)
      CALL MULT3(S,PS,PS,6,6)
      CALL MULT3(S,BOM,BOM,6,6)
      CALL EVALUI(N,IPO,AO,BO,SAO,CAO,THETAD,RHO,IMO,MO,GI)
      CALL ERROR(N,IN,THETA,THDOT,THDDOT,THETAD,THDOTD,THDDOD,
     *           E,ED,EDD)
      CALL LSE(GI,PS,BOM,E,ED,EDD,KK,IN,N,AL)
      CALL FEEDB(KK,E,ED,N)
      RETURN
      END
      SUBROUTINE ERROR(N,IN,THETA,THDOT,THDDOT,THETAD,THDOTD,THDDOD,
     *                 E,ED,EDD)
C         -------------THE CALCULATION OF ERROR BETWEEN------------
C         -------DESIRED TRAJECTORY AND ACTUAL TRAJECTORY-------
C*************************************************************************
C
C         THIS SUBROUTINE PERFORM THE CALCULATION OF ERROR BETWEEN THE
C         DESIRED TRAJECTORY AND ACTUAL TRAJECTORY. THE POSITION, VELOCITY
C         AND ACCELERATION ERROR ARE STORED IN 'E','ED'AND 'EDD', RESPECTIVELY
C
C     INPUT:
C         N-----------NUMBER OF THE LINKAGES
C         IN----------RECURSIVE FLAG
C         THETAD------DESIRED POSITION
C         THDOTD------DESIRED VELOCITY
C         THDDOD-----DESIRED ACCELERATION
C         THETA-------ACTUAL POSITION
C         THDOT-------ACTUAL VELOCITY
C         THDDOT------ACTUAL ACCELERATION
C
C    OUTPUT:
C         E-----------POSITION ERROR
C         ED----------VELOCITY ERROR
C         EDD---------ACCELERATION ERROR
C
C*************************************************************************
          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
          DIMENSION THETA(N),THDOT(N),THDDOT(N),THETAD(N),THDOTD(N),
     *              THDDOD(N),E(N),ED(N),EDD(N)
          DO 110 I=1,N
                E(I)=THETA(I)-THETAD(I)
                ED(I)=THDOT(I)-THDOTD(I)
                EDD(I)=THDDOT(I)-THDDOD(I)
110          CONTINUE
          RETURN
          END



          SUBROUTINE LSE(GI,PS,BOM,E,ED,EDD,KK,IN,N,AL)
C        ---------------LEAST-SQUARE ESTIMATION-----------------.
C*******************************************************************
C
C         THIS SUBROUTINE IS USED TO PERFORM LEAST SQUARE ESTIMATION
C         OF THE FEED BACK GAIN "K"
C
C     INPUT:
C         GI--------INERTIA MATRIX OF DYNAMIC MODEL OF R M.
C         PS--------DAMPING MATRIX OF THE SYSTEM
C         BOM-------NATRUAL FREQUENCY OF THE SYSTEM
C         E---------POSITION ERROR
C         ED--------VELOCITY ERROR
C         EDD-------ACCELERATION ERROR
C         TT--------SAMPLING PERIOD
C         IN--------RECURSIVE FLAG
C         N---------NUMBER OF THE LINKAGES
C
C    OUTPUT:
C         KK--------ESTIMATED MATRIX
C                   (INCLUDING FEEDBACK GAIN AND ERROR VECTOR)
C
C*******************************************************************
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION KK,KKP1,KKD1,KKP,KKD
      DIMENSION GI(6,6),PS(6,6),BOM(6,6),E(6),
     *          ED(6),EDD(6),B1(6),B2(6),B3(6),
     *          P(13,13),PHE(13),PP(13),PFR(13,13),DTAU(6),
     *          PT(13,13),PE(13),Y(6),YY(6),YT(6),YT1(6),
     *          PF(13,13),PP1(13,13),PHET(13,13),KK(6,13),
     *          KKP1(6,6),KKD1(6,6),KKP(6,6),KKD(6,6)
      COMMON /CON/DTAU,TT,TZ
C
C  SET INTIAL VALUES
C
      IF(IN.EQ.0) THEN
         DO 30  I=1,N
            DO 30  J=1,N+N+1
               KK(I,J)=0.D00
30       CONTINUE
         CALL MULT(GI,BOM,KKP1,6,6,6)
         CALL MULT(KKP1,BOM,KKP,6,6,6)
         CALL MULT(BOM,GI,KKD1,6,6,6)
         CALL MULT(KKD1,PS,KKD,6,6,6)
         DO 40  I=1,6
            DO 40  J=1,6
            KK(I,J)=KKP(I,J)
            KK(I,J+6)=KKD(I,J)*2'D00
40       CONTINUE
      ELSE
         END IF
         CALL MULT1(KKP,E,B1,N,N)
         CALL MULT1(KKD,ED,B2,N,N)
         CALL MULT1(GI,ED,B3,N,N)
      IF(IN.EQ.0) THEN
         DO 50  I=1,6
            KK(I,13)=-(B1(I)+2.D00*B2(I))*TT+B3(I)
50       CONTINUE
      ELSE
         END IF
         DO 55  I=1,6
            YT1(I)=-(B1(I)+2.D00*B2(I))*TT+B3(I)
55       CONTINUE
         S=10.D25
         DO 60  I=1,N+N+1
            P(I,I)=P(I,I)*S
60       CONTINUE
C
C   CALCULATE ERROR BETWEEN DESIRED  'Y' AND THE ACTUAL 'Y'
C
      DO 70 I=1,N
         PHE(I)=E(I)*TT
         PHE(I+6)=ED(I)*TT
70    CONTINUE
      PHE(13)=AL
      DO 90 I=1,N
         Y(I)=0.D00
         DO 90  J=1,N
            Y(I)=(EDD(J)*TT+ED(J))*GI(I,J)+Y(I)
90    CONTINUE
      DO 100 I=1,N
         YT(I)=0.D00
         DO 110  J=1,N+N+1
            YT(I)=YT(I)+PHE(J)*KK(I,J)
110      CONTINUE
         YY(I)=Y(I)-YT(I)
100   CONTINUE
      RU=9.5D-1
C
```

```
C    CALCULATE SCALER 'RU'
C
        CALL MULT2(PHE,P,PP,N+N+1,N+N+1)
        V=0.D00
        DO 120 I=1,N+N+1
           V=V+PP(I)*PHE(I)
120     CONTINUE
        R=1.D00/(V+1.D00)
C
C    CALCULATE MATRIX 'P'
C
        DO 130 I=1,N+N+1
           DO 130 J=1,N+N+1
              PHET(I,J)=PHE(I)*PHE(J)
130     CONTINUE
        CALL MULT(P,PHET,PT,N+N+1,N+N+1,N+N+1)
        CALL MULT(PT,P,PF,N+N+1,N+N+1,N+N+1)
        CALL MULT3(R,PF,PFR,N+N+1,N+N+1)
        DO 140 I=1,N+N+1
           DO 140 J=1,N+N+1
              P(I,J)=(P(I,J)-PFR(I,J))/RU
140     CONTINUE
        CALL MULT3(R,P,PP1,13,13)
C
C    CALCULATE MATRIX 'KK'
C
        CALL MULT1(PP1,PHE,PE,N+N+1,N+N+1)
        DO 150 I=1,N
           DO 150 J=1,N+N+1
        KK(I,J)=PE(J)*YY(I)+KK(I,J)
150     CONTINUE
600     format(6f12.7)
        RETURN
        END



        SUBROUTINE FEEDB(KK,E,ED,N)
C
C       ------CALCULATE MULTIPLY FORCES------
C*****************************************************************
C
C       THIS SUBROUTINE IS USED TO CALCULATE THE MULTIPLY FORCES
C       FROM THE FEEDBACK GAINS
C
C    INPUT:
C       KK---------ESTIMATED MATRIX(INCLUDING FEEDBACK GAINS)
C       E----------POSITION ERROR
C       ED---------VELOCITY ERROR
C       EDD--------ACCELERATION ERROR
C       N----------NUMBER OF THE LINKAGES
C
C   OUTPUT:
C       DATU-----MULTIPLY FORCES
C
C*****************************************************************
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DOUBLE PRECISION KK,KP,KD
        DIMENSION KK(6,13),KP(6,6),KD(6,6),E(6),ED(6),
                  DTAU(6),DTT1(6),DTT2(6)
        COMMON /CON/DTAU,TT,TZ
        DO 10 I=1,N
           DO 10 J=1,N
              KP(I,J)=KK(I,J)
```

```
            KD(I,J)=KK(I,J+N)
 10      CONTINUE
         CALL MULT1(KP,E,DTT1,N,N)
         CALL MULT1(KD,ED,DTT2,N,N)
         DO 20 I=1,N
            DTAU(I)=-DTT1(I)-DTT2(I)
 20      CONTINUE
         RETURN
         END


         SUBROUTINE MULT(A1,A2,A,N1,N2,N3)
C
C        THIS SUBROUTINE IS USED TO PERFORM THE MULTIPLICATION BETWEEN
C        TWO MATRICES
C
         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
         DIMENSION A1(N1,N2),A2(N2,N3),A(N1,N3)
         DO 10 I=1,N1
            DO 10 J=1,N3
               A(I,J)=0.D00
               DO 20 K=1,N2
                  A(I,J)=A(I,J)+A1(I,K)*A2(I,K)
 20            CONTINUE
 10      CONTINUE
         RETURN
         END


         SUBROUTINE MULT1(G,D,D1,N1,N2)
C
C        THIS SUBROUTINE IS USED TO PERFORM THE MULTIPLICATION BETWEEN
C        ONE MATRIX AND ONE VECTOR
C
         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
         DIMENSION G(N1,N2),D(N2),D1(N1)
         DO 10 I=1,N1
            D1(I)=0.D00
            DO 10 J=1,N2
               D1(I)=D1(I)+G(I,J)*D(J)
 10      CONTINUE
         RETURN
         END


         SUBROUTINE MULT2(D,G,D1,N1,N2)
C
C        THIS SUBROUTINE IS USED TO PERFORM THE MUTIPLICATION BETWEEN
C        ONE TRANSPOSED VECTOR AND ONE MATRIX
C
         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
         DIMENSION D(N1),G(N1,N2),D1(N2)
         DO 10 I=1,N2
            D1(I)=0.D00
            DO 10 J=1,N1
               D1(I)=D1(I)+D(J)*G(J,I)
 10      CONTINUE
         RETURN
         END
```

```
      SUBROUTINE MULT3(S,A,B,N1,N2)
C
C     THIS SUBROUTINE IS USED TO PERFORM THE MULTIPLICATION BETWEEN
C     ONE SCALAR AND ONE MATRIX
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N1,N2),B(N1,N2)
      DO 10 I=1,N1
         DO 10 J=1,N2
            B(I,J)=S*A(I,J)
10    CONTINUE
      RETURN
      END



      SUBROUTINE IDEN(IDE,N1)
C
C     THIS SUBROUTINE IS USED TO GET AN IDENTITY MATRIX
C
      DOUBLE PRECISION IDE(N1,N1)
      DO 10 I=1,N1
         DO 10 J=1,N1
            IF(I.EQ.J) THEN
               IDE(I,J)=1.D00
            ELSE
               IDE(I,J)=0.D00
            END IF
10    CONTINUE
      RETURN
      END
```

Appendix C

Dynamic Modelling Package

```
C*****************DYNAMIC MODELLING PACKAGE*****************************
C
C       THE PACKAGE IS IMPLEMENTED ON THE BASE OF KANE'S  DYNAMICAL
C       EQUATIONS. BY USING THIS PACKAGE ONE CAN GET THE DYNAMIC
C       MODEL OF THE ROBOT WITH ANY NUMBER OF LINKAGES (UP TO 6).
C       IT WAS TESTED ON A VAX-780 (McRCIM) IN DEC. 1986.
C
C INPUT:
C
C                                     ; M= NUMBER OF THE EQUATIONS OF MOTION.
C                                     ; N= NUMBER OF LINKAGES
C
C    1.     P(I),I=1,18               ; PARAMETERS OF DENAVIT&HARTENBERG :
C                                          I=1,6...........A(I) ,
C                                          I=7,12.........B(I)
C                                          I=13,18........ALPHA(I)
C    2.     PD(I),I=1,60              ; DYNAMICS PARAMETERS OF EACH LINKAGE:
C                                          I=1 MASS
C                                          I=2,4 CENTROID COORDINATES
C                                          I=5,10 MOMENT OF INERTIA.
C
C OUTPUT:
C           TF(I)                     ; TORQUES ACTING ON THE LINKAGES
C
C**********************************************************************
       REAL  THETA(6),DF(7,6),P(480),V(6),A(6),PD(60),TF(6),
     -       XA(21),YA(21),AX(21),BX(21),CX(21),PA(24)
       CHARACTER*64 FNAME
       DATA TOL,MAX,N,M/0.000001,30,6,7/
       WRITE(*,1000)
1000   FORMAT(2X,'NAME OF THE INPUT DATA FILE : ')
       READ(*,1010)FNAME
1010   FORMAT(A5)
       OPEN(6,FILE=FNAME,STATUS='UNKNOWN')
       OPEN(7,FILE='FORCE.DAT',STATUS='UNKNOWN')
       OPEN(8,FILE='LIN DAT',STATUS='UNKNOWN')
C THE OUTPUT IS STORED IN THE FILE NAMED 'FORCE.DAT'
       READ(5,100)(P(I),I=1,18)
       READ(5,100)(PD(I),I=1,60)
       READ(5,201) MN
C203     FORMAT(F10.5,I3)
       WRITE(6,102)
       T=0.0
       DTA=3./MN
       PII=3.14159265359
       RAD=PII/180.
       DO 10 I=1,N
           P(I+12) = RAD * P(I+12)
          P(I+18) = SIN(P(I+12))
          P(I+12) = COS(P(I+12))
10     CONTINUE
       DO 1011 I=1,MN
       CALL TRAJ(T,THETA,V,A)
       CALL FUN(THETA,P,M,N)
       CALL DFDX(THETA,DF,P,M,N)
       CALL VELAC(M,N,DF,P,V)
       CALL FZA(PD,P,N,V,A,TF)
       WRITE(7,170) T,TF(1),TF(2)
       T=DTA+T
100    FORMAT(6F10.5)
```

```
101     FORMAT(5F10.5,I3)
102     FORMAT(/5X,'PARAMETERS OF MECHANISM AND INITIAL VALUES'//11X,
     -          'A',9X,'B',7X,'ALFA',6X,'THETA'//)
103     FORMAT(2X,I4,4F10.5)
104     FORMAT(/2X,'1.- MASS'/2X,'2  - 4. CENTROID OF LINKAGE,',
     -          ' REFRRED TO SYSTEM I+1'/2X,'5.- IXX, 6.- IXY,'
     -          ' 7.- IXZ, 8 - IYY, 9 - IYZ, 10.- IZZ'/)
105     FORMAT(I4,6F10.5)
C106      FORMAT(3X,'VALUES OF POSICTION, VELOCITY AND ACEL. IN JOINT SPACE')
107     FORMAT(3E21.13)
108     FORMAT(//2X,'TIME= ',F10.5)
200     FORMAT(2F10.5)
201     FORMAT(I4)
300     FORMAT(6F12.7)
1011    CONTINUE
        STOP
        END



        SUBROUTINE FUN(THETA,P,M,N)
        REAL  THETA(6),P(480),TCS(12)
C*************************************************************************
C   THE ROTATION AND DISPLACEMENT EQUATIOS VANISH(=0). HEREBY, CALCULATE
C   THE JOINT VARIABLES FOR ANY PARTICULAR CONFIGULRATION
C   OF THE ROBOT END EFFECTOR.
C*************************************************************************
C
C  LEFT-HAND SIDES OF DISPLACEMENT EQUATIONS ARE COMPUTED
C
        DO 2 I=1,N
           TCS(I) =  COS(THETA(I))
           TCS(I+N) =  SIN(THETA(I))
    2   CONTINUE
        CALL PROD(TCS,P)
        CALL VECX(TCS,P)
        RETURN
        END



        SUBROUTINE PROD(TCS,P)
        REAL P(480),TCS(12)
C*************************************************************************
C   IN THIS SUBROUTINE FOLLOWING PRODUCTS ARE COMPUTED :
C   Q1 IN P(33-41)
C   Q1*Q2 IN P(42-50)
C   Q1*Q2*Q3 IN P(51-59)
C   Q1*Q2*Q3*Q4  IN P(60-68)
C   Q1*Q2*Q3*Q4*Q5 IN P(69-77)
C   Q1*Q2*Q3*Q4*Q5*Q6 IN P(78-86)
C*************************************************************************
        P(33) = TCS(1)
        P(34) = TCS(7)
        P(35) = 0.0
        P(36) =-TCS(7)  * P(13)
        P(37) = TCS(1)  * P(13)
        P(38) = P(19)
```

```
      P(39) = TCS(7) * P(19)
      P(40) = -TCS(1) * P(19)
      P(41) = P(13)
      IP = 32
      DO 10 I=2,6
         IA = IP
         IP = IP + 9
         A = TCS(I+6) * P(I+12)
         B = TCS(I) * P(I+12)
         C = TCS(I+6) * P(I+18)
         D = TCS(I) * P(I+18)
         DO 10 J=1,3
            IPJ = IP + J
            P(IP+J) = P(IA+J)*TCS(I)+P(IA+J+3)*TCS(I+6)
            P(IP+J+3) = -P(IA+J)*A+P(IA+J+3)*B+P(IA+J+6)*P(I+18)
            P(IP+J+6) = P(IA+J)*C-P(IA+J+3)*D+P(IA+J+6)*P(I+12)
   10 CONTINUE
      RETURN
      END


      SUBROUTINE VECX(TCS,P)
      REAL TCS(12),P(480)
C***********************************************************************
C  SUBROUTINE VECX.  CALCULATES THE CARTESIAN COORDINATES OF THE
C                    END EFFECTOR WITH RESPECT TO THE BASE
C                    COORDINATE SYSTEM
C***********************************************************************
      P(102) = P(6) * TCS(6)
      P(103) = P(6) * TCS(12)
      P(104) = P(12)
      DO 10 I=1,5
         K=6-I
         KK = 83+K+K+K
         KKP3 = KK + 3
         PX1 = P(K) + P(KKP3+1)
         PX2 = P(K+12)*P(KKP3+2) - P(K+18)*P(KKP3+3)
         P(KK+1) = TCS(K)*PX1 - TCS(K+6)*PX2
         P(KK+2) = TCS(K+6)*PX1 + TCS(K)*PX2
         P(KK+3) = P(K+6)+ P(K+18)*P(KKP3+2)+ P(K+12)*P(KKP3+3)
   10 CONTINUE
      RETURN
      END


      SUBROUTINE DFDX(THETA,DF,P,M,N)
      REAL THETA(N),DF(M,N),P(480)
C***********************************************************************
C  SUBROUTINE DFDX   CALCULATES JACOBIAN MATRIX
C***********************************************************************
      L=23
      LX = 87
      LP = 104
      DF(1,1) = -P(84)
      DF(2,1) = -P(85)
      DF(3,1) =  P(78)+P(82)
      DF(4,1) =  P(81)-P(79)
      DF(5,1) = -P(88)
      DF(6,1) =  P(87)
      DF(7,1) =  0.0
      DO 10 I=2,N
         L = L + 9
         LX = LX+3
         LP = LP+3
         J = L + 7
         DF(1,I) = P(J)*(P(82)+P(86))-P(J+1)*P(81)-P(J+2)*P(84)
         DF(2,I) =-P(J)*P(79)+P(J+1)*(P(78)+P(86))-P(J+2)*P(85)
         DF(3,I) =-P(J)*P(80)-P(J+1)*P(83)+P(J+2)*(P(78)+P(82))
            TEMP = P(J)*(P(85)-P(83))+ P(J+1)*(P(80)-P(84))
         DF(4,I) = TEMP+ P(J+2)*(P(81)-P(79))
         DO 4 K=1,3
            DF(K+4,I)= -P(L+K)*P(LX+1)+ P(L+3+K)*P(LX)
            P(LP+K)=DF(K+4,I)
    4    CONTINUE
   10 CONTINUE
      P(105) = DF(5,1)
      P(106) = DF(6,1)
      P(107) = DF(7,1)
      RETURN
      END



      SUBROUTINE VELAC(M,N,DF,P,V)
      REAL DF(7,6),P(480),V(6),A(6)
      COMMON DDT(6,6,3)
C***********************************************************************
C
C   IN THIS SUBROUTINE WE CALCULATE THE VELOCITY AND ACCELERATION
C   OF 7 LINKAGE KINEMATIC CHAIN
C     V(I) - VECTOR OF VELOCITY: ANGULAR (1-3) AND END EFFECTOR (5-7)
C     A(I) - VECTOR OF ACCELERATION ANGULAR(1-3) AND END EFFECTOR(5-7)
C
C   WE OBTAIN :
C     THE VELOCITY AND ACCELERATION IN  V(I) AND A(I).
C
C***********************************************************************
C   CALCULATE THE 2ND DERIVATIVE OF ANGULAR ACCELERATION WITH RESPECT TO
C   THETA DOT.
      NM1=N-1
      P(123)=-P(40)*V(1)
      P(124)= P(39)*V(1)
      P(125)= 0.0
      VEC1= 0.0
      VEC2= 0.0
      VEC3= V(1)
      IP= 153
      IW= 123
      ID1= 30
      DO 2 I=2,NM1
         ID1=ID1+9
         IW=IW+3
         IP=IP+3
         VEC1=VEC1+P(ID1)*V(I)
         VEC2=VEC2+P(ID1+1)*V(I)
         VEC3=VEC3+P(ID1+2)*V(I)
         P(IP)=VEC1
         P(IP+1)=VEC2
         P(IP+2)=VEC3
         ID2=ID1+9
         P(IW)=VEC2*P(ID2+2)-VEC3*P(ID2+1)
         P(IW+1)=VEC3*P(ID2)-VEC1*P(ID2+2)
         P(IW+2)=VEC1*P(ID2+1)-VEC2*P(ID2)
2     CONTINUE
      P(168)=P(75)*V(6)+P(165)
      P(169)=P(76)*V(6)+P(166)
      P(170)=P(77)*V(6)+P(167)
C
C   CALCULATE THE 2ND DERIVATIVE OF DISPLACEMENT WITH RESPECT TO THETA
C
      DDT(1,1,1)=-P(106)
      DDT(1,1,2)= P(105)
      DDT(1,1,3)= 0 0
      DO 8 I=2,N
         ID3=102+3*I
         DDT(1,I,1)=-P(ID3+1)
         DDT(1,I,2)= P(ID3)
         DDT(1,I,3)= 0.0
         DDT(I,1,1)=DDT(1,I,1)
         DDT(I,1,2)=DDT(1,I,2)
         DDT(I,1,3)=DDT(1,I,3)
         ID4=21+9*I
         DO 7 J=I,N
            ID5=102+3*J
            DDT(I,J,1)=P(ID4+1)*P(ID5+2)-P(ID4+2)*P(ID5+1)
            DDT(I,J,2)=P(ID4+2)*P(ID5)-P(ID4)*P(ID5+2)
            DDT(I,J,3)=P(ID4)*P(ID5+1)-P(ID4+1)*P(ID5)
            IF(I EQ J) GO TO 7
            DDT(J,I,1)=DDT(I,J,1)
            DDT(J,I,2)=DDT(I,J,2)
            DDT(J,I,3)=DDT(I,J,3)
7        CONTINUE
8     CONTINUE
      IX1=134
      DO 10 I=1,N
         IX1=IX1+3
         S1=0.0
         S2=0.0
         S3=0.0
         DO 9 J=1,N
            S1=S1+DDT(J,I,1)*V(J)
            S2=S2+DDT(J,I,2)*V(J)
            S3=S3+DDT(J,I,3)*V(J)
9        CONTINUE
         P(IX1+1)=S1
         P(IX1+2)=S2
         P(IX1+3)=S3
10       CONTINUE
      RETURN
      END



      SUBROUTINE CROSS(X,Y,Z)
      REAL X(3),Y(3),Z(3)
C*************************************************************************
C
C   THIS SUBROUTINE CALCULATE THE CROSS MULTIPLICATION OF TWO VECTORS
C   X AND Y. THE RESULT IS STORED IN VECTOR Z.
C
C*************************************************************************
      Z(1)=X(2)*Y(3)-Y(2)*X(3)
      Z(2)=X(3)*Y(1)-Y(3)*X(1)
      Z(3)=X(1)*Y(2)-Y(1)*X(2)
      RETURN
      END


      SUBROUTINE DRDT(PD,P,NM1)
      REAL PD(60),P(480),VEC1(3),VEC2(3),VEC3(3)
C**********************************************************************
C
C   THIS SUBROUTINE CALCULATE THE DERIVATIVE OF R(I)
C   WITH RESPECT TO TIME AND STORE THEM.
C
C     DR(1) IN P(171-173)
C     DR(2) IN P(174-179)
C     DR(3) IN P(180-188)
C     DR(4) IN P(189-200)
C     DR(5) IN P(201-215)
C     DR(6) IN P(105-122)
C
C   FIRST CALCULATE X(I+1)-RHO(I) REFERRED TO THE FIXED SYSTEM,
C   SECONDLY CALCULATE THE DERIVATIVE OF R(I).
C
C**********************************************************************
      IP=33
      IX=89
      IPD=0
      IS=171
      A1=P(90)-PD(2)
      B1=P(91)-PD(3)
      C1=P(92)-PD(4)
      AA=P(33)*A1+P(36)*B1+P(39)*C1
      BB=P(34)*A1+P(37)*B1+P(40)*C1
      CC=P(35)*A1+P(38)*B1+P(41)*C1
      P(171)=P(105)+BB
      P(172)=P(106)-AA
      P(173)=P(107)
      DO 10 I=2,NM1
         IP=IP+9
         IX=IX+3
         IPD=IPD+10
         IS=IS+3
         A1=P(IX+1)-PD(IPD+2)
         B1=P(IX+2)-PD(IPD+3)
         C1=P(IX+3)-PD(IPD+4)
         VEC2(1)=P(IP)*A1+P(IP+3)*B1+P(IP+6)*C1
         VEC2(2)=P(IP+1)*A1+P(IP+4)*B1+P(IP+7)*C1
         VEC2(3)=P(IP+2)*A1+P(IP+5)*B1+P(IP+8)*C1
         P(IS)=P(105)+VEC2(2)
         P(IS+1)=P(106)-VEC2(1)
         P(IS+2)=P(107)
         IDX=105
         IQ=30
         DO 5 J=2,I
            IQ=IQ+9
            IDX=IDX+3
            IS=IS+3
            VEC1(1)=P(IQ)
            VEC1(2)=P(IQ+1)
            VEC1(3)=P(IQ+2)
            CALL CROSS(VEC1,VEC2,VEC3)
            P(IS)=P(IDX)-VEC3(1)
            P(IS+1)=P(IDX+1)-VEC3(2)
            P(IS+2)=P(IDX+2)-VEC3(3)
5        CONTINUE
10    CONTINUE
```

```
      RETURN
      END


      SUBROUTINE PIPTA(PD,P,B,N)
      REAL PD(60),P(480),B(3,21)
      C13=PD(5)*P(35)+PD(6)*P(38)+PD(7)*P(41)
      C23=PD(6)*P(35)+PD(8)*P(38)+PD(9)*P(41)
      C33=PD(7)*P(35)+PD(9)*P(38)+PD(10)*P(41)
      B(1,1)=C13*P(33)+C23*P(36)+C33*P(39)
      B(2,1)=C13*P(34)+C23*P(37)+C33*P(40)
      B(3,1)=C13*P(35)+C23*P(38)+C33*P(41)
      IB=1
      DO 2 I=2,N
         IPD=5+10*(I-1)
         IP=24+9*I
         IB=IB+1
         C11=PD(IPD)*P(IP)+PD(IPD+1)*P(IP+3)+PD(IPD+2)*P(IP+6)
         C21=PD(IPD+1)*P(IP)+PD(IPD+3)*P(IP+3)+PD(IPD+4)*P(IP+6)
         C31=PD(IPD+2)*P(IP)+PD(IPD+4)*P(IP+3)+PD(IPD+5)*P(IP+6)
         C12=PD(IPD)*P(IP+1)+PD(IPD+1)*P(IP+4)+PD(IPD+2)*P(IP+7)
         C22=PD(IPD+1)*P(IP+1)+PD(IPD+3)*P(IP+4)+PD(IPD+4)*P(IP+7)
         C32=PD(IPD+2)*P(IP+1)+PD(IPD+4)*P(IP+4)+PD(IPD+5)*P(IP+7)
         C13=PD(IPD)*P(IP+2)+PD(IPD+1)*P(IP+5)+PD(IPD+2)*P(IP+8)
         C23=PD(IPD+1)*P(IP+2)+PD(IPD+3)*P(IP+5)+PD(IPD+4)*P(IP+8)
         C33=PD(IPD+2)*P(IP+2)+PD(IPD+4)*P(IP+5)+PD(IPD+5)*P(IP+8)
         B11=C11*P(IP)+C21*P(IP+3)+C31*P(IP+6)
         B12=C12*P(IP)+C22*P(IP+3)+C32*P(IP+6)
         B13=C13*P(IP)+C23*P(IP+3)+C33*P(IP+6)
         B22=C12*P(IP+1)+C22*P(IP+4)+C32*P(IP+7)
         B23=C13*P(IP+1)+C23*P(IP+4)+C33*P(IP+7)
         B33=C13*P(IP+2)+C23*P(IP+5)+C33*P(IR+8)
         B(1,IB)=B13
         B(2,IB)=B23
         B(3,IB)=B33
         IQ=30
         DO 1 J=2,I
            IQ=IQ+9
            IB=IB+1
            B(1,IB)=B11*P(IQ)+B12*P(IQ+1)+B13*P(IQ+2)
            B(2,IB)=B12*P(IQ)+B22*P(IQ+1)+B23*P(IQ+2)
            B(3,IB)=B13*P(IQ)+B23*P(IQ+1)+B33*P(IQ+2)
1        CONTINUE
2     CONTINUE
      RETURN
      END


         SUBROUTINE DRTDT(P,V,NM1)
         REAL P(480),V(6),DR(5,5,3)
C*************************************************************************
C
C CALCULATE THE DERIVATIVE OF PARTIAL R(I) OVER PARTIAL THETA
C WITH RESPECT TO TIME.
C
C*************************************************************************
      IS=171
      IPS=314
      IDS=213
      DO 10 I=2,NM1
```

```
               IS=IS+3
               DR(1,1,1)=-P(IS+1)
               DR(1,1,2)=.P(IS)
               DR(1,1,3)= 0.0
               DO 4 J=2,I
                 IS=IS+3
                 DR(1,J,1)=-P(IS+1)
                 DR(1,J,2)= P(IS)
                 DR(1,J,3)= 0 0
                 DR(J,1,1)= DR(1,J,1)
                 DR(J,1,2)= DR(1,J,2)
                 DR(J,1,3)= DR(1,J,3)
                 ISA=IS
                 IP=21+9*J
                 DO 2 K=J,I
                   DR(J,K,1)= P(IP+1)*P(ISA+2)-P(IP+2)*P(ISA+1)
                   DR(J,K,2)= P(IP+2)*P(ISA)-P(IP)*P(ISA+2)
                   DR(J,K,3)= P(IP)*P(ISA+1)-P(IP+1)*P(ISA)
                   ISA=ISA+3
                   IF(K.EQ.J) GO TO 2
                   DR(K,J,1)= DR(J,K,1)
                   DR(K,J,2)= DR(J,K,2)
                   DR(K,J,3)= DR(J,K,3)
2                CONTINUE
4              CONTINUE
               DO 8 J=1,I
                 IDS=IDS+3
                 P(IDS) = 0.0
                 P(IDS+1)=0 0
                 P(IDS+2)=0 0
                 DO 6 K=1,I
                   IPS=IPS+3
                   P(IPS+1)=DR(K,J,1)
                   P(IPS+2)=DR(K,J,2)
                   P(IPS+3)=DR(K,J,3)
                   P(IDS)=P(IDS)+DR(K,J,1)*V(K)
                   P(IDS+1)=P(IDS+1)+DR(K,J,2)*V(K)
                   P(IDS+2)=P(IDS+2)+DR(K,J,3)*V(K)
6                CONTINUE
8              CONTINUE
10          CONTINUE
            RETURN
            END




              SUBROUTINE GINE(PD,P,B,GI,N)
              REAL PD(60),P(480),B(3,21),GI(6,6),VEC1(3),VEC2(3),VEC3(3)
C****************************************************************************
C
C  THE GENERALIZED INERTIAL MATRIX IS CALCULATD IN THIS SUBROUTINE
C
C****************************************************************************
         DO 133 I=1,N
          DO 134 J=1,N
         GI(I,J)=0 0
134        CONTINUE
133       CONTINUE
         ID=1
         NM1=N-1
         IDR=171
         SP=P(171)*P(171)+P(172)*P(172)+P(173)*P(173)
         GI(1,1)=SP*PD(ID)
         DO 6 I=2,NM1
            IDR=IDR+3
            ID=ID+10
            VEC1(1)=P(IDR)
            VEC1(2)=P(IDR+1)
            VEC1(3)=P(IDR+2)
            SP=VEC1(1)*VEC1(1)+VEC1(2)*VEC1(2)+VEC1(3)*VEC1(3)
            GI(1,1)=GI(1,1)+SP*PD(ID)
            DO 4 J=2,I
               IDA=IDR
               IDR=IDR+3
               VEC2(1)=P(IDR)
               VEC2(2)=P(IDR+1)
               VEC2(3)=P(IDR+2)
               SP=VEC1(1)*VEC2(1)+VEC1(2)*VEC2(2)+VEC1(3)*VEC2(3)
               GI(1,J)=GI(1,J)+SP*PD(ID)
               DO 2 K=J,I
                  IDA=IDA+3
                  VEC3(1)=P(IDA)
                  VEC3(2)=P(IDA+1)
                  VEC3(3)=P(IDA+2)
                  SP=VEC2(1)*VEC3(1)+VEC2(2)*VEC3(2)+VEC2(3)*VEC3(3)
                  GI(J,K)=GI(J,K)+SP*PD(ID)
 2             CONTINUE
 4          CONTINUE
 6       CONTINUE
         IDR=105
         SP=P(105)*P(105)+P(106)*P(106)+P(107)*P(107)
         GI(1,1)=GI(1,1)+SP*PD(51)+B(3,1)
         DO 10 I=2,N
            IDA=IDR
            IDR=IDR+3
            VEC1(1)=P(IDR)
            VEC1(2)=P(IDR+1)
            VEC1(3)=P(IDR+2)
            SP=P(105)*VEC1(1)+P(106)*VEC1(2)+P(107)*VEC1(3)
            GI(1,I)=GI(1,I)+SP*PD(51)
            DO 8 J=I,N
               IDA=IDA+3
               VEC2(1)=P(IDA)
               VEC2(2)=P(IDA+1)
               VEC2(3)=P(IDA+2)
               SP=VEC1(1)*VEC2(1)+VEC1(2)*VEC2(2)+VEC1(3)*VEC2(3)
               GI(I,J)=GI(I,J)+SP*PD(51)
 8          CONTINUE
 10      CONTINUE
         IB=1
         DO 16 I=2,N
            IB=IB+1
            GI(1,1)=GI(1,1)+B(3,IB)
            IQ=30
            DO 14 J=2,I
               IBA=IB
               IB=IB+1
               GI(1,J)=GI(1,J)+B(3,IB)
               IQ=IQ+9
               DO 12 K=J,I
                  IBA=IBA+1
                  SP=B(1,IBA)*P(IQ)+B(2,IBA)*P(IQ+1)+B(3,IBA)*P(IQ+2)
                  GI(J,K)=GI(J,K)+SP
 12            CONTINUE
 14         CONTINUE
 16      CONTINUE
         DO 18 I=1,N
```

```
              DO 18  J=1,N
18        GI(J,I)=GI(I,J)
          RETURN
          END




            SUBROUTINE DGINE(PD,P,B,DGI,DGIAP,DGIW,N)
            REAL PD(60),P(480),B(3,21),DGI(6,6),VEC1(3),VEC2(3),VEC3(3),
          - DGIAP(6,6),DGIW(6,6)
C*************************************************************************
C
C  DERIVE THE GENERAL INERTIA MATRIX AND SIMPLICITY.
C  PERFORM THE PRODUCT MATRIX OF PARTIAL R(I) WITH RESPECT TO THETA
C  AND STORE IN DGI(I,J).
C
C*************************************************************************
          DO 133 I=1,N
          DO 133 J=1,N
          DGI(I,J)=0.0
          DGIAP(I,J)=0.0
          DGIW(I,J)=0.0
133       CONTINUE
          ID=1
          NM1=N-1
          IDR=171
          IDD=213
          DO 10 I=2,NM1
             ID=ID+10
             IDA=IDD
             DO 8 J=1,I
                IDR=IDR+3
                V1X=P(IDR)
                V1Y=P(IDR+1)
                V1Z=P(IDR+2)
                IDA=IDD
                DO 6 K=1,I
                   IDA=IDA+3
                   V1DX=P(IDA)
                   V1DY=P(IDA+1)
                   V1DZ=P(IDA+2)
                   SP=V1X*V1DX+V1Y*V1DY+V1Z*V1DZ
                   DGI(J,K)=DGI(J,K)+SP*PD(ID)
6               CONTINUE
8            CONTINUE
             IDD=IDA
10        CONTINUE
          IDR=102
          IDD=135
          DO 14 I=1,N
             IDR=IDR+3
             V1X=P(IDR)
             V1Y=P(IDR+1)
             V1Z=P(IDR+2)
             IDA=IDD
             DO 12 J=1,N
                IDA=IDA+3
                V1DX=P(IDA)
                V1DY=P(IDA+1)
                V1DZ=P(IDA+2)
                SP=V1X*V1DX+V1Y*V1DY+V1Z*V1DZ
                DGI(I,J)=DGI(I,J)+SP*PD(51)
```

```
12          CONTINUE
14       CONTINUE
C
C CALCULATE THE PRODUCT OF E(I)XW(J)
C
         VEC2(1)=P(156)
         VEC2(2)=P(157)
         VEC2(3)=P(158)
         P(258)=-VEC2(2)
         P(259)= VEC2(1)
         P(260)= 0.0
         P(261)= P(40)*VEC2(3)-P(41)*VEC2(2)
         P(262)= P(41)*VEC2(1)-P(39)*VEC2(3)
         P(263)= P(39)*VEC2(2)-P(40)*VEC2(1)
         IR=261
         IW=156
         DO 4 I=3,N
            IR=IR+3
            IW=IW+3
            VEC2(1)=P(IW)
            VEC2(2)=P(IW+1)
            VEC2(3)=P(IW+2)
            P(IR) =-VEC2(2)
            P(IR+1)=VEC2(1)
            P(IR+2)=0.0
            DO 2 J=2,I
               IR=IR+3
               IPA=21+9*J
               VEC1(1)=P(IPA)
               VEC1(2)=P(IPA+1)
               VEC1(3)=P(IPA+2)
               CALL CROSS(VEC1,VEC2,VEC3)
               P(IR)=VEC3(1)
               P(IR+1)=VEC3(2)
               P(IR+2)=VEC3(3)
2           CONTINUE
4        CONTINUE
C
C CALCULATE THE PRODUCT OF AT(I)XW(I)*I(I)*A(I) AND STORE IN DGIW(J,K)
C THE CALCULATED VALUES OF DA(I)T*(I(I)*A(I)) ARE STORED IN DGIAP(J,K)
C
         IB=1
         IDI=255
         DO 22 I=2,N
            IDW=120
            DO 20 J=1,I
               IDI=IDI+3
               V1X=P(IDI)
               V1Y=P(IDI+1)
               V1Z=P(IDI+2)
               IBA=IB
               IF(J.EQ.1) GO TO 16
               IDW=IDW+3
               E1X=P(IDW)
               E1Y=P(IDW+1)
               E1Z=P(IDW+2)
16             DO 18 K=1,I
                  IBA=IBA+1
                  SP=V1X*B(1,IBA)+V1Y*B(2,IBA)+V1Z*B(3,IBA)
                  DGIW(J,K)=DGIW(J,K)+SP
                  IF(J.EQ 1) GO TO 18
                  SP1=E1X*B(1,IBA)+E1Y*B(2,IBA)+E1Z*B(3,IBA)
                  DGIAP(J,K)=DGIAP(J,K)+SP1
18             CONTINUE
20          CONTINUE
            IB=IBA
```

```
22    CONTINUE
      RETURN
      END


        SUBROUTINE FZA(PD,P,N,V,A,TF)
        REAL PD(60),P(480),V(N),A(N),B(3,21),GI(6,6),DGI(6,6),
     -        DGIAP(6,6),DGIW(6,6),DV(6),TF(N),PA(24)
C*****************************************************************
C
C     GI(I,J) STORES THE ENTRIES OF GENERALIZED INERTIA MATRIX
C    DGI(I,J) STORES THE PRODUCTS OF PARCIAL R(I) WITH RESPECT
C             TO THETA AND TEH DERIVATIVE WITH RESPECT TO TIME
C DGIAP(I,J) STORE PRODUCT OF DA(I)T*(I,I)*A(I)).
C DGIW (I,J) STORE PRODUCT OF AT(I)XW(I)*(I(I)*A(I))
C
C*****************************************************************
      NM1=N-1
      GRAV=9.81
      CALL DRDT(PD,P,NM1)
      CALL PIPTA(PD,P,B,N)
      CALL DRTDT(P,V,NM1)
      CALL GINE(PD,P,B,GI,N)
      CALL DGINE(PD,P,B,DGI,DGIAP,DGIW,N)
      DV(1)=PD(1)*P(173)+PD(11)*P(176)+PD(21)*P(182)+PD(31)*P(191)
      DV(1)=DV(1)+PD(41)*P(203)+PD(51)*P(107)
      DV(2)=PD(11)*P(179)+PD(21)*P(185)+PD(31)*P(194)+PD(41)*P(206)
      DV(2)=DV(2)+PD(51)*P(110)
      DV(3)=PD(21)*P(188)+PD(31)*P(197)+PD(41)*P(209)+PD(51)*P(113)
      DV(4)=PD(31)*P(200)+PD(41)*P(212)+PD(51)*P(116)
      DV(5)=PD(41)*P(215)+PD(51)*P(119)
      DV(6)=PD(51)*P(122)
      WRITE(6,100)
100   FORMAT(/5X,'VELOCIDAD',5X,'ACELERACION',5X,'PAR GRAV ',5X,
     -        'PAR TOTAL'/)
110   FORMAT(2X,I2,4(E15.6,2X))
      DO 2 I=1,N
        DV(I)=-DV(I)*GRAV
        S=0.0
        DO 1 J=1,N
          T1=DGI(I,J)+DGIW(I,J)+DGIAP(J,I)
          S=S+GI(I,J)*A(J)+T1*V(J)
1       CONTINUE
        TF(I)=S+DV(I)
2     CONTINUE
      RETURN
      END


      SUBROUTINE TRAJ(T,THETA,V,A)
      REAL THETA(6),V(6),A(6)
      pi=acos(-1.)
      THETA(1)=(15.+20.*T**2-40./9*T**3)*pi/180.
      THETA(2)=(30.*T**2-20./3.*T**3)*pi/180.
      THETA(3)=0.0
      THETA(4)=0.0
      THETA(5)=0.0
      THETA(6)=0.0
      V(1)=(40.*T-40./3.*T**2)*pi/180.
      V(2)=(60.*T-20.*T**2)*pi/180.
      V(3)=0.0
      V(4)=0.0
      V(5)=0.0
```

```
V(6)=0.0
A(1)=(40.-80./3.*T)*pi/180.
A(2)=(60.-40.*T)*pi/180.
A(3)=0.0
A(4)=0.0
A(5)=0.0
A(6)=0.0
RETURN
END
```