1

Micro-ED

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

ADAPTIVE CONTROL FOR ROBOTIC MANIPULATORS

by

RAMESH SRINIVASAN

B. Eng., University of Poona, India, 1985.

A Thesis submitted to the
Faculty of Graduate Studies and Research,
In Partial Fulfilment of the Requirements for the Degree of
Master of Engineering,
in the
Department of Mechanical and Aeronautical Engineering,
Carleton University,
Ottawa, Ontario, Canada.

© copyright

December 1987, Ramesh Srinivasan

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-40632-1

ACCEPTANCE SHEET

The undersigned recommend to the Faculty of Graduate Studies and Research, acceptance of the thesis entitled "Adaptive Control for Robotic Manipulators", submitted by Ramesh Srinivasan in partial fulfilment of the requirements for the degree of Master of Engineering in Mechanical Engineering.

J. Z. Sasiadek,
Thesis Supervisor.

Chairman,
Dept. of Mech. and Aero. Eng.

ABSTRACT

Model Reference Adaptive Control methods have been gaining prominence in the control applications of robotic manipulators. This thesis investigates the effectiveness of MRAC techniques on several applications of robotic manipulator control.

The MRAC methods presented here are based on the theory of *partitioning control*, which makes them capable of compensating for the non-linear terms in the dynamic equations, as well as to decouple the dynamic interaction between the links. These features make MRAC ideal for manipulator control applications.

A computer program, written in 'C', is used to simulate the various manipulator applications, such as rigid manipulators, flexible manipulators and dual-arm manipulators. Simulation results indicate a very high degree of success in the control of rigid manipulators. Flexible and dual-arm manipulators exhibit certain oscillatory characteristics. The nature of this oscillatory behaviour has been investigated and suitable methods of related control have been proposed.

DEDICATION

To my Parents

ACKNOWLEDGEMENT

I would like to express my deepest appreciation and gratitude to Professor Jerzy Z. Sasiadek, my thesis advisor, for all his guidance, encouragement and friendship throughout the course of this work, as well as in the preparation of this thesis.

My thanks are also due to the J. N. Tata Endowment Trust, Bombay, India, for their initial financial support, and to the Department of Mechanical and Aeronautical Engineering, Carleton University for funding my research at this University.

My acknowledgement would be incomplete without a word of thanks to Raja Krishnan, for having proof-read the manuscript.

TABLE OF CONTENTS

	<u>Page</u>
ACCEPTANCE SHEET	ii
ABSTRACT	iii
DEDICATION	iv
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi
LIST OF TABLES	viii
LIST OF ILLUSTRATIONS	ix
LIST OF SYMBOLS	xii

CHAPTER I: Introduction

1.1 Preface	1
1.2 Literature Review of Adaptive Control	3
1.3 Review of Adaptive Manipulator Control	5
1.4 Objectives and Organization	7

CHAPTER II: Dynamics of Rigid Manipulators

2.1 Introduction	9
2.2 Lagrangian Dynamics for the Two-link Manipulator	10

CHAPTER III: Adaptive Control

3.1 Introduction	21
3.2 Types of Adaptive Control	21
3.3 Theory of Model Reference Adaptive Control	24

CHAPTER IV: MRAC Applications for Rigid Manipulators

4.1 Introduction	48
------------------	----

4.2	Computer Simulation of R2D2	49
4.3	Evaluation of another MRAC Scheme	62
4.4	MRAC for higher Degrees of freedom Manipulators	63
CHAPTER V: MRAC Applications for Flexible Manipulators		
5.1	Introduction	74
5.2	Modeling Link Flexibility	76
5.3	Theory of Deflection Compensation	79
5.4	MRAC with Deflection Compensation for Flexible Manipulators	86
5.5	Summary	94
CHAPTER VI: MRAC for Dual-arm Manipulators		
6.1	Introduction	97
6.2	Hierarchical Adaptive Control for a Planar dual-arm Manipulator	99
6.3	Hybrid Control of Dual-arm Manipulators	112
6.4	Summary	125
CHAPTER VII: Summary		
7.1	Conclusions	126
7.2	Suggestions for Future Research	128
APPENDIX 1: Lagrangian Dynamics for R2D3		
		130
APPENDIX 2: Lagrangian Dynamics for R3D3		
		134
APPENDIX 3: Simulation Program Code for Rigid Manipulators		
		140
APPENDIX 4: Simulation Program Code for Flexible Manipulators		
		150
APPENDIX 5: Simulation Program Code for Dual-arm Manipulators		
		163
REFERENCES		
		180

LIST OF TABLES

Table 5.1 Flexible Manipulator Parameters	87
Table 6.1 Dual-arm Parameters	103

LIST OF ILLUSTRATIONS

Fig. 2.1	Planar Two-link Manipulator	12
Fig. 2.2	Parameters of the Two-link Manipulator	13
Fig. 3.1	Essential Components of an Adaptive Control System	23
Fig. 3.2	Parameter-adaptive MRAC System	25
Fig. 3.3	Signal-Synthesis Adaptive MRAC System	26
Fig. 3.4	Self-tuning Regulator	27
Fig. 3.5	A Model-based Manipulator Control System	31
Fig. 3.6	MRAC Scheme of Horowitz	35
Fig. 3.7	Popov's Standard Feedback System	39
Fig. 3.8	MRAC Scheme of Asare-Wilson	42
Fig. 3.9	Control System for the Reference Model	44
Fig. 4.1	Step-response of R2D2, with MRAC and critical damping	54
Fig. 4.2	Step-response of R2D2, with MRAC (under-damped)	56
Fig. 4.3	Step-response of R2D2, without adaptation	58
Fig. 4.4	Angular Velocity Profile of R2D2, with MRAC	59
Fig. 4.5	Step-response of R2D2, with MRAC (payload = 3 Kg)	60
Fig. 4.6	Step-response of R2D2, with MRAC (payload = 5 Kg)	61
Fig. 4.7	Step-response of R2D2 (the second MRAC method)	64
Fig. 4.8	R2D3 Manipulator Configuration	66
Fig. 4.9	Step-response of R2D3	68
Fig. 4.10	R3D3 Manipulator Configuration	69
Fig. 4.11	Adaptive Step-response of R3D3 Manipulator	72
Fig. 4.12	Non-adaptive Step-response of R3D3 Manipulator	73

Fig. 5.1	Deflection Compensation, Type-1	80
Fig. 5.2	Deflection Compensation, Type-2	82
Fig. 5.3	Deflection Compensation, Type-3	83
Fig. 5.4	Deflection Compensation, Types 1 and 3	84
Fig. 5.5	End-effector Error Profiles for F2D2 and F2D3	88
Fig. 5.6	End-effector Error Profiles for F2D3 and F3D3	89
Fig. 5.7	End-effector Error Profiles for F2D2 and F3D3	90
Fig. 5.8	End-effector Error Profiles for F2D2, upto 10 seconds	92
Fig. 5.9	End-effector Error Profiles for F2D2, beyond 10 seconds	93
Fig. 5.10	End-effector Error Profiles for F2D2	95
Fig. 6.1	Trajectory for each arm of the Planar Dual Manipulator	101
Fig. 6.2	Tracking Error with Non-adaptive Control	104
Fig. 6.3	Tracking Error with MRAC	106
Fig. 6.4	Hierarchical Adaptive Control Structure	107
Fig. 6.5	Tracking Error with Hierarchical Adaptive Control	111
Fig. 6.6	Single-arm Configuration for Three-dimensional Dual Manipulator	113
Fig. 6.7	Error Profile without Supervisory Control, in a Gravity Field	117
Fig. 6.8	Error Profile with Supervisory Control, in a Gravity Field	118
Fig. 6.9	Error Profile with Supervisory Control, in a Weightless Environment	120
Fig. 6.10	Torque Profile without Supervisory Control, in a Gravity Field	121
Fig. 6.11	Torque Profile with Supervisory Control, in a Gravity Field	122

Fig. 6.12 Torque Profile with Supervisory Control, in a Weightless Environment

123

Fig. 6.13 Torque Profile without Force Control

124

LIST OF SYMBOLS

l_1, l_2, l_3	link lengths (m)
$\theta_1, \theta_2, \theta_3$	joint angles (rad)
m_1, m_2, m_3	masses of link-1, link-2 and (link-3 + payload) respectively (Kg)
L	Lagrangian of the system
K	kinetic energy of the system
P	potential energy of the system
K_1, K_2, K_3	kinetic energy of links 1, 2 and 3 respectively
P_1, P_2, P_3	potential energy of links 1, 2 and 3 respectively
τ_1, τ_2, τ_3	joint torques (N-m)
$\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3$	joint rates (rad/s)
$\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3$	joint accelerations (rad/s^2)
g	gravitational acceleration (m/s^2)
q_p	position vector of the manipulator
$\dot{q}_p$	velocity vector of the manipulator
$\ddot{q}_p$	acceleration vector of the manipulator
q_m	position vector of the model
$\dot{q}_m$	velocity vector of the model
$\ddot{q}_m$	acceleration vector of the model

$H(q_p)$	actual general inertia matrix of the manipulator
$c(q_p, \dot{q}_p)$	non-linear coupling vector (actual)
$c_{grav}(q_p)$	gravity vector of the manipulator
$f(q_p, \dot{q}_p)$	non-linear friction vector
$\tau(t)$	joint torque vector
q_{pd}	desired position vector
q_{vd}	desired velocity vector
$\dot{q}_{vd}$	desired acceleration vector
$e(t)$	position error vector
$\dot{e}(t)$	velocity error vector
$\hat{H}(q_p), \hat{c}(q_p, \dot{q}_p), \hat{c}_{grav}(q_p), \hat{f}(q_p, \dot{q}_p)$	adaptive estimates of $H(q_p), c(q_p, \dot{q}_p), c_{grav}(q_p), f(q_p, \dot{q}_p)$ respectively
$u(t)$	controlling input vector (commanded acceleration vector)
F_p, F_v	stability matrices
C_p, C_v	error gain matrices
α_{ij}, β_{ij}	adaptation gains
$v(t)$	compounded error vector
K_p, K_I, K_D	PID controller gain matrices
$r_p(t)$	desired position vector
E	Young's modulus of elasticity

I_n	moment of inertia of the beam cross-section about a neutral axis which passes through the centroid of the section
δ	link-tip or end-effector deviation
ρ	density of the material of the links
d	tracking error
ϵ	compensation vector
G, K	error compensation gains
F	normal end-effector force (N)
diff	angular position difference vector between two arms

CHAPTER I

INTRODUCTION

1.1 Preface

Robots and mechanical manipulators play an important and increasing role in today's industrial world. This is primarily due to their potential to integrate into and enhance the process of automation. This has led to an increasing application of robotic manipulators in production and manufacturing industries. Robotic manipulators are now an integral part of flexible manufacturing cells/systems. It is in fact thought that "factories of the future", in which human intervention would be occasional at best, and supervision remote, will not be long in the coming.

The last decade has seen robotic manipulators being applied in several special applications as well. They are now being actively used in space and undersea exploration, as well as in remote and hazardous environments, such as in the nuclear industry. In the field of biomechanics and biomedical engineering, robots have been envisaged to act as guides and "slaves", to aid the physically handicapped.

A robot, according to the Robot Institute of America, is a programmable multi-function manipulator, designed to move material, parts, tools or special devices through variable, programmed motions for the performance of a wide variety of tasks.

An industrial robot, according to Groover and Zimmers [1], is a general-purpose programmable machine, possessing certain anthropomorphic characteristics. The most typical human-like characteristic of the robot is its arm.

A robotic manipulator can be defined as an open-loop kinematic chain of links and joints, possessing several degrees of freedom. Actuators, which are responsible for the individual links are customarily located at the joints.

Notwithstanding the tremendous versatility of robotic manipulators, their practical exploitation has been fairly limited in scope. This is mainly due to the difficulty in controlling them. The dynamic characteristics of a manipulator are fairly complex and non-linear. They are also time-variant, since they depend on the exact configuration of the manipulator, and therefore change as the manipulator executes a task. Further, the mechanism of the manipulator is subject to the phenomenon of coupling, whereby the simultaneous movement of the links gives rise to coupling forces and torques. As a result, the motion of any link affects the behaviour of those of the other links in the kinematic chain. Finally, the designer

has to ensure that the control system of the manipulator will be capable of effecting optimum performance over a wide range of payload masses, which the manipulator might carry during its work-cycle.

These demanding requirements of manipulator control have led researchers to look beyond conventional linear control techniques. One of the most promising methods is adaptive control, which is rapidly gaining popularity among researchers.

Adaptive control, which comprises an important area of modern control, deals with the control of systems in the presence of uncertainties, structural perturbations and environmental variations.

1.2 Literature review of Adaptive Control

Adaptive control was first proposed by Draper and Li [2] as far back as 1951. Optimizing control, or 'peak-holding control' as they called it, dealt with a control system, which could optimize the performance of an internal combustion engine in the presence of uncertainties in the performance characteristics. The term "adaptive" was first used in control literature in 1954, when Tsien [3] described Ashby's [4] model of the human brain.

Model reference adaptive control (MRAC) techniques were first

introduced by Whitaker et al. [5] in 1958, when they considered adaptive aircraft flight control systems, employing a reference model to obtain error signals between the actual and desired behaviour. These error signals were used to modify the controller parameters to attain ideal behaviour, to the extent possible, in spite of uncertainties and varying system dynamics.

Further research in adaptive control led to the development of new techniques, such as self-oscillating adaptive control, by Li and Van der Velde [6] in 1960; variable structure control by Petrov et al. [7] in 1963; dual control by Bellman [8] and Fel'dbaum [9] in 1961 and partitioning adaptive control by Lainiotis [10] in 1976.

The next major step in the development of adaptive control was taken by Åström and Wittenmark [11] in their proposal of self-tuning regulators. Their work led to such an upsurge of interest throughout the world in this field, that self-tuning began to emerge as a new and distinct technique of control. It is now considered to belong to the class of adaptive control systems.

Another important breakthrough came in 1974 when Monopoli [12] introduced the augmented error approach to the design of stable MRAC systems. This led to a considerable amount of research in developing adaptive algorithms for globally asymptotically stable systems and resulted in 1980 in several important papers by Narendra et al. [13 - 15], Morse [16], Goodwin et al. [17] and Egardt

[18,19] on adaptive control systems.

Among the recent books on adaptive control are those by Saridis [20] who covers the important area of self-organizing control of stochastic systems, Egardt [21] dealing with stability of adaptive controllers, Landau [22] on model reference adaptive systems and the application of hyperstability concepts, and Chalam[23] on a comprehensive, integrated account of basic concepts, analytical tools, algorithms and a variety of application trends and techniques. Three publications edited by Narendra and Monopoli [24], Unbehauen [25] and Harris and Billings [26] are collections of interesting papers on various topics. Other related topics are covered by Ioannou and Kokotovic [27], Åström and Wittenmark [28] and Goodwin and Sin [29].

1.3 Review of Adaptive Manipulator Control

The application of adaptive control techniques to the control of manipulators has been a fairly popular subject of research. Dubowsky and Des Forges [30] have applied MRAC methods to robotic manipulators. They had used a linear, second-order, time-invariant differential equation as the reference model for each degree of freedom of the manipulator arm. The manipulator was controlled by adjusting the position and velocity feedback gains to follow the model. A steepest-descent method was used for updating

the feedback gains. Using a different approach [31], they present an adaptive control algorithm which minimizes the quadratic function of the error defined as the difference between the desired state vector and the manipulation process state vector.

Arimoto and Takegaki [32] have presented an adaptive control algorithm based on local parameter optimization for a robot described by a linear, time-varying model derived from linearization around the desired trajectory. This method assures the stability of the error system.

Koivo et al. [33] have used the Kalman filter in conjunction with an adaptive controller, to control the positions and velocities of the joints in an industrial manipulator. Their simulation results have demonstrated the feasibility of this technique. In a different paper, Koivo and Guo [34] have described an adaptive self-tuning controller for each joint of a robotic manipulator. They had used an autoregressive moving average (ARMA) model to fit the input-output data from the manipulator. However, they had assumed that the interaction forces among the joints are negligible.

Tomizuka and Horowitz [35] have applied MRAC systems for the dynamic control of mechanical manipulators, which are highly non-linear and complex. They have described both continuous-time and discrete-time adaptive schemes, which perform non-linearity compensation and decoupling control. Their simulation studies for a

three degrees of freedom manipulator indicated the definite advantages of MRAC methods in manipulator control.

Vukobratovic, et al. [36] report on the control synthesis for robotic systems with a variable payload, first in a non-adaptive manner. If such non-adaptive control cannot stabilize robotic manipulator trajectories, they suggest an adaptive control scheme, which includes an algorithm for on-line identification of variable payload and adaptation of local gains.

Lee and Chung [37] considered an adaptive control strategy for computer-based manipulators. In later papers [38, 39], they suggest an adaptive perturbation control which can track a time-based joint trajectory as closely as possible for all times and over a wide range of manipulator motion and payloads.

1.4 Objectives and organization

In the recent past, Model Reference Adaptive Control has come to be recognized as an effective method for robotic manipulator control applications. The objective of this thesis is to demonstrate and evaluate the efficacy of MRAC techniques for manipulator control, as well as to explore their usefulness in related robotic applications. Such related applications investigated here include dual-arm manipulators and flexible manipulators.

The remaining portion of this thesis is organized in the following manner. Chapter 2 deals with manipulator dynamics, and its derivation by the Lagrangian method for a two-link, rigid, planar manipulator. The theory of MRAC is presented in Chapter 3, while its application to a two-link, rigid planar manipulator is described in Chapter 4. In Chapter 5, MRAC applications to flexible manipulators and their results are discussed. Chapter 6 contains a description of the application of MRAC techniques to dual-arm manipulators along with simulation results. And the final conclusions of this thesis along with suggestions for future research comprise Chapter 7.

CHAPTER II

DYNAMICS OF RIGID MANIPULATORS

2.1 Introduction

Knowledge of the Dynamics, a major part of the Mechanics, of a robotic manipulator is essential for the design of its control system. Mechanics of a manipulator consists of its kinematics and its dynamics. Kinematics is concerned with the shape and geometry of the manipulator and its motion. The kinematic analysis of robotic manipulators is further subdivided into the forward or direct kinematic analysis and the inverse kinematic analysis.

The problem of determining the position and orientation of the end-effector, given the various joint-angles of the manipulator is commonly referred to as the forward or direct kinematic analysis. Here and henceforth, in this thesis, a joint angle is defined as the angle made by two adjacent links at that joint. Thus, joint angles refer to the orientation of a link with respect to its preceding link. The position and orientation of the end-effector however, are required to be specified in base coordinates - the coordinate system attached to the base of the manipulator. This is then the problem of direct kinematics.

Inverse kinematics, on the other hand, yields the joint angles of the manipulator, given the position and orientation of the end-effector. The inverse kinematic problem is not as simple as the forward kinematic one. Because the kinematic equations are non-linear, their solution is not always easy; or sometimes, even possible in a closed form. Also, the question of the existence of multiple solutions arises.

Dynamics of manipulators is a field devoted to the study of the forces required to cause motion. In this context, dynamics provides relationships between the torques to be applied by the actuators at the joints, and the motion of the manipulator thereby accomplished.

Similar to kinematics, dynamics too can be divided into direct and inverse dynamics. Inverse dynamics, which provides the actuator torques necessary for a particular motion, is used in manipulator control. The reverse process of computing motion from the actuator torques, is used in manipulator simulation. In this thesis, both direct and inverse dynamics have been used; the former in the design of the control system and the latter in the computer simulation. They are incorporated together in the computer program.

2.2 Lagrangian Dynamics for a Rigid, Two-link Manipulator

The two most popular methods for deriving the dynamic

equations of manipulators are the recursive Newton-Euler technique and the Lagrangian method. The Lagrangian method has been used in this thesis for the derivation of the dynamic equations of several manipulators.

In the remaining portion of this chapter, an example of the planar, two-link manipulator is used to illustrate the method of Lagrange for the dynamic equations of a manipulator.

Consider the planar, two-link manipulator of Fig. 2.1. The manipulator consists of two links jointed end-to-end, moving in a vertical plane. Actuators located at both joints cause rotation of the links in the vertical plane. The links are of lengths ' l_1 ' and ' l_2 ' respectively. The mass of link-1 is taken as ' m_1 ' and that of link-2 and the payload together as ' m_2 '. This clubbing of payload mass and the mass of link-2 is possible because of the assumption of point masses. These point masses ' m_1 ' and ' m_2 ' are further assumed to be acting at the distal end of each link, i.e. at joint-2 and the end-effector respectively (Fig. 2.2). In practice, the centre of mass of each link would lie somewhere within the body of the link, and between its two extreme ends. Consequently, the forces and torques to be applied by the actuators would be less than those computed for a point-mass model. Because it is the objective in this thesis to test the application of MRAC techniques to manipulators, it was deemed proper to adopt this conservative, or worst-case scenario. In the

Fig. 2.1 Planar Two-link Manipulator

Fig. 2.2 Parameters of the Two-link Manipulator

following analysis, masses and inertias of actuators, as well as frictional forces acting at the joints have been neglected.

The Lagrangian ' L ' of a system is defined as :

$$L = K - P \quad (2.1)$$

where;

K is the total kinetic energy of the system,
and P is the total potential energy of the system.
From Lagrange's relation for the forces, we have,

$$F_i = \frac{d}{dt} \cdot \frac{\partial L}{\partial \dot{x}_i} - \frac{\partial L}{\partial x_i} \quad (2.2)$$

Here, F_i is the force to be applied at the ' i^{th} ' joint and x_i is the generalized space coordinate of the ' i^{th} ' link, and

$$\dot{x}_i = \frac{d}{dt}(x_i) ; \quad 0 < i \leq n$$

where; n = number of degrees of freedom of the manipulator.

In the case under consideration, the manipulator has only revolute joints. The actuator force F_i can therefore be replaced by the actuator torque τ_i , and the generalized space coordinate x_i by the joint angle θ_i . We then have,

$$\tau_i = \frac{d}{dt} \cdot \frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} \quad (2.3)$$

Therefore, in order to obtain τ_i , we need the Lagrangian 'L' of the manipulator. Defining the kinetic energy of link-1 as ' K_1 ', that of link-2 and the payload as ' K_2 ', and their respective potential energies as ' P_1 ' and ' P_2 '; we can write:

$$K = K_1 + K_2 \quad \text{and} \quad P = P_1 + P_2 \quad (2.4)$$

For link-1, we have:

$$K_1 = 0.5m_1l_1^2\dot{\theta}_1^2 \quad (2.5)$$

$$P_1 = m_1gl_1S_1 \quad (2.6)$$

where; $S_1 = \sin(\theta_1)$ and $C_1 = \cos(\theta_1)$, etc.

and g is the acceleration due to gravity.

For link-2, we need to first write down the coordinates of the end of link-2 and then differentiate them with respect to time to obtain the kinetic energy. Denoting the Cartesian coordinates of the end of link-2 as (x_2, z_2) ;

$$x_2 = l_1 C_1 + l_2 C_{12}$$

(2.7)

$$z_2 = l_1 S_1 + l_2 S_{12}$$

where, $S_{12} = \sin(\theta_1 + \theta_2)$ and $C_{12} = \cos(\theta_1 + \theta_2)$, etc.

Differentiating eqn. (2.7) with respect to time, we get:

$$\dot{x}_2 = -l_1 \dot{\theta}_1 S_1 - l_2 \dot{\theta}_{12} S_{12}$$

(2.8)

$$\dot{z}_2 = l_1 \dot{\theta}_1 C_1 + l_2 \dot{\theta}_{12} C_{12}$$

where; $\dot{\theta}_{12} = \dot{\theta}_1 + \dot{\theta}_2$.

Squaring and adding $\dot{x}_2$ and $\dot{z}_2$;

$$\dot{x}_2^2 + \dot{z}_2^2 = l_1^2 \dot{\theta}_1^2 + l_2^2 (\dot{\theta}_{12})^2 + 2l_1 l_2 C_2 \dot{\theta}_1 \dot{\theta}_{12} \quad (2.9)$$

Also, $K_2 = 0.5 m_2 v_2^2$, where $v_2^2 = \dot{x}_2^2 + \dot{z}_2^2$

$$\therefore K_2 = 0.5 m_2 l_1^2 \dot{\theta}_1^2 + 0.5 m_2 l_2^2 (\dot{\theta}_{12})^2 + m_2 l_1 l_2 C_2 \dot{\theta}_1 \dot{\theta}_{12} \quad (2.10)$$

The potential energy term for link-2 becomes:

$$P_2 = m_2 g l_1 S_1 + m_2 g l_2 S_{12} \quad (2.11)$$

Now since $P = P_1 + P_2$,

$$P = m_{12}g l_1 S_1 + m_2 g l_2 S_{12} \quad (2.12)$$

and $K = K_1 + K_2$

$$K = 0.5 m_{12} l_1^2 \dot{\theta}_1^2 + 0.5 m_2 l_2^2 (\dot{\theta}_{12})^2 + m_2 l_1 l_2 C_2 \dot{\theta}_1 \dot{\theta}_{12} \quad (2.13)$$

where;

$m_{12} = m_1 + m_2$, in the usual notation.

The Lagrangian is then obtained as :

$$\begin{aligned} L &= K - P \\ &= 0.5 m_{12} l_1^2 \dot{\theta}_1^2 + 0.5 m_2 l_2^2 (\dot{\theta}_{12})^2 + m_2 l_1 l_2 C_2 \dot{\theta}_1 \dot{\theta}_{12} - \\ &\quad m_{12} g l_1 S_1 - m_2 g l_2 S_{12} \end{aligned} \quad (2.14)$$

From eqns. (2.3) and (2.14);

$$\begin{aligned} \tau_1 &= \ddot{\theta}_1 [m_{12} l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 C_2] \\ &\quad + \ddot{\theta}_2 [m_2 l_2^2 + m_2 l_1 l_2 C_2] \\ &\quad + 2\dot{\theta}_1 \dot{\theta}_2 [-m_2 l_1 l_2 S_2] \\ &\quad + \dot{\theta}_2^2 [-m_2 l_1 l_2 S_2] \\ &\quad + g [m_{12} l_1 C_1 + m_2 l_2 C_{12}] \end{aligned} \quad (2.15)$$

$$\begin{aligned}
\tau_2 = & \ddot{\theta}_1 [m_2 l_2^2 + m_2 l_1 l_2 C_2] \\
& + \ddot{\theta}_2 [m_2 l_2^2] \\
& + \dot{\theta}_1^2 [m_2 l_1 l_2 S_2] \\
& + g [m_2 l_2 C_{12}]
\end{aligned} \tag{2.16}$$

Rewriting eqns. (2.15) and (2.16);

$$\tau_1 = h_{11} \ddot{\theta}_1 + h_{12} \ddot{\theta}_2 + 2c_{1cor} \dot{\theta}_1 \dot{\theta}_2 + c_{1cent} \dot{\theta}_2^2 + c_{1grav} g \tag{2.17}$$

and

$$\tau_2 = h_{21} \ddot{\theta}_1 + h_{22} \ddot{\theta}_2 + c_{2cent} \dot{\theta}_1^2 + c_{2grav} g \tag{2.18}$$

where;

$$h_{11} = m_1 l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 C_2$$

$$h_{12} = m_2 l_2^2 + m_2 l_1 l_2 C_2$$

$$h_{21} = h_{12}$$

$$h_{22} = m_2 l_2^2$$

$$c_{1cor} = -m_2 l_1 l_2 S_2$$

$$c_{1cent} = -m_2 l_1 l_2 S_2$$

$$c_{2cent} = m_2 l_1 l_2 S_2$$

$$c_{1grav} = m_{12} l_1 C_1 + m_2 l_2 C_{12}$$

$$c_{2grav} = m_2 l_2 C_{12}$$

Expressed in matrix form, eqns. (2.17) and (2.18) become:

$$\begin{Bmatrix} \tau_1 \\ \tau_2 \end{Bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix} + \begin{Bmatrix} 2c_{1cor}\dot{\theta}_1\dot{\theta}_2 + c_{1cent}\dot{\theta}_2^2 \\ c_{2cent}\dot{\theta}_1^2 \end{Bmatrix} + \begin{Bmatrix} c_{1grav} \\ c_{2grav} \end{Bmatrix} \cdot g \quad \dots(2.19)$$

or

$$\tau(t) = H(q_p) \cdot \ddot{q}_p + c(q_p, \dot{q}_p) + G \cdot g \quad (2.20)$$

where;

$\tau(t)$ is the (2 x 1) torque vector

$H(q_p)$ is the (2 x 2) general inertia matrix (GIM)

$\ddot{q}_p$ is the (2 x 1) angular acceleration vector

$c(q_p, \dot{q}_p)$ is the (2 x 1) vector representing the non-linear coupling terms, such as Coriolis and centripetal acceleration

G is the (2 x 1) gravity vector

and g is gravitational acceleration.

In equation (2.20) and henceforth in this thesis, symbols written in bold type represent vectors or matrices, while scalars are written normally.

CHAPTER III

ADAPTIVE CONTROL

3.1 Introduction

In this chapter, the two principal types of adaptive control; *model reference adaptive control (MRAC)* and *self-tuning adaptive control (STAC)* are briefly described. The theory of partitioning control is presented with a view to illustrating how it naturally leads up to MRAC. Two different versions of MRAC are presented and discussed. The MRAC methods described in this chapter are used to control various manipulators, as will be seen in subsequent chapters.

3.2 Types of Adaptive Control

Landau [22] defines an adaptive system as one which measures a certain index of performance using the inputs, the states and the outputs of the adjustable system; and from a comparison of the measured performance-index values and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable system, or generates an auxiliary input in order to maintain the

performance-index (IP) values close to the set of given ones. This definition is illustrated in Fig. 3.1.

Although several other definitions for adaptive systems are available, many of them highlight the following basic functions, common to most of them:

1. Identification of unknown parameters, or measurement of a performance index;
2. Decision on the control strategy;
3. On-line modification of the parameters of the controller, or the input signal.

There are two principal approaches to adaptive control; namely, model reference adaptive control (MRAC) and self-tuning adaptive control (STAC), also known as self-tuning regulation. The MRAC controller aims to force the output of an unknown plant to asymptotically approach the output of a given reference model, which is part of the control system. MRAC is thus a special class of adaptive control systems, in which the given set of performance indexes (Fig. 3.1) is replaced by a set of dynamically generated performance indexes. To generate this reference set of performance indexes, a reference model is used, which is excited by the same input as the adjustable system. The performance index set can then be taken as the outputs of each system, which are compared to yield an error term. This error term is used by the adaptation mechanism to modify the controller parameters, so as to minimize this error.

Fig. 3.1 Essential Components of an Adaptive Control System

This is a *parameter-adaptive* system (Fig. 3.2). It is also possible to configure the adaptation mechanism to generate an auxiliary input signal, to minimize the difference between the output of the reference model and that of the adjustable system. Such a system is called a *signal-synthesis MRAC*, as illustrated in Fig. 3.3.

The main problem while designing MRAC systems is to determine the adaptation mechanism such that, not only is the error brought to zero, but also that the resulting system is stable. Stability is important because, if the system is not guaranteed to be stable, the system is of no practical utility.

Another important form of an adaptive system is the self-tuning regulator, originally introduced by Åström and Wittenmark in 1973 [11]. In self-tuning adaptive control, a design procedure for known plant parameters is first chosen. This is applied to the unknown plant, using recursively estimated values of the parameters. Such a system is shown in Fig. 3.4.

3.3 Theory of Model Reference Adaptive Control

MRAC, as mentioned earlier, lends itself well to the control of non-linear, time-varying and poorly known systems. A robotic manipulator is such a non-linear system, as can be seen from eqn. 3.1 (eqn. 2.20 rewritten).

Fig. 3.2 Model Reference Adaptive Control System:
Parameter Adaptation

Fig. 3.3 Model Reference Adaptive Control System:
Signal Synthesis Adaptation

Fig. 3.4 Self-tuning Regulator

$$\tau(t) = H(q_p) \cdot \ddot{q}_p + c(q_p, \dot{q}_p) + c_{grav} \cdot g \quad (3.1)$$

The non-linear character of the dynamic equations can be seen from the cyclic terms $\cos(\theta)$, $\sin(\theta)$, etc. in (2.15) and (2.16). These inherent non-linearities in manipulator dynamics arise from the following factors:

1. dynamic coupling between the joints;
2. Coriolis and centripetal terms;
3. damping forces due to friction at the joints;
4. modeling inaccuracies.

Frictional forces at the joints consist of forces due to viscous friction as well as Coulomb friction. While viscous friction is proportional to the joint velocity (linear or angular), Coulomb friction is more non-linear. Representing these forces together as $f(q_p, \dot{q}_p)$, a more accurate form of eqn. (3.1) would be:

$$\tau(t) = H(q_p) \cdot \ddot{q}_p + c(q_p, \dot{q}_p) + c_{grav} \cdot g + f(q_p, \dot{q}_p) \quad (3.2)$$

However, it should be noted that the frictional forces are not usually included in the modeling, primarily because it is difficult to predict them. In this thesis too, frictional forces have not been modelled in the computer simulation study. But in order to illustrate the method of non-linearity compensation and decoupling control, friction is included in this section.

Non-linearity compensation is achieved by the method of *control-law partitioning*. This technique requires the controller for the system to be decomposed into two parts. One part of the control law is model-based; i.e. it uses the dynamic equations of the particular system under control. The second part of the control law is the error-driven servo portion, which forms error signals by comparing desired and actual variables, and multiplies these errors by gains. The model-based portion of the control appears in a control law of the form:

$$\tau = a\tau' + b \quad (3.3)$$

where, a and b are constants or functions, and are chosen such that if τ' is taken as the new input to the system, the system now appears linear. Equating eqns. (3.2) and (3.3), we have:

$$H(q_p) \ddot{q}_p + c(q_p, \dot{q}_p) + c_{grav} \cdot g + f(q_p, \dot{q}_p) = a\tau' + b \quad (3.4)$$

In order to make the system appear to be linear, a judicious choice of a and b would be:

$$a = H(q_p)$$

and,

$$b = c(q_p, \dot{q}_p) + c_{grav} \cdot g + f(q_p, \dot{q}_p) \quad (3.5)$$

Finally, a suitable servo law is applied, such as:

$$\tau'(t) = \ddot{\mathbf{q}}_{pd} + \mathbf{K}_v \dot{\mathbf{e}}(t) + \mathbf{K}_p \mathbf{e}(t) \quad (3.6)$$

where, the servo error vector $\mathbf{e}(t)$ is given by :

$$\mathbf{e}(t) = \mathbf{q}_{pd} - \mathbf{q}_p \quad (3.7)$$

the difference between the desired and actual joint position vectors. The term ' $\dot{\mathbf{e}}(t)$ ' refers to the error vector between desired and actual angular velocity vectors, and $\ddot{\mathbf{q}}_{pd}$ is the desired angular acceleration vector of the joints. The resulting control system is shown in Fig. 3.5.

The major problem is that the values of the parameters in the model are often not known accurately. This is particularly true of friction effects. Also, manipulators are often employed for picking up various parts and tools. When a tool is grasped, the inertia matrix and the centre of mass of the last link of the manipulator change, and hence need to be updated to their new values. However, in many applications, the mass properties of objects that the manipulator picks up are not generally known, so that maintenance of an accurate dynamic model is very difficult.

Since in general, our model of the manipulator will never be perfectly accurate, we define the following notation: $\hat{\mathbf{H}}(\mathbf{q}_p)$ is our

Fig. 3.5 A Model-based Manipulator Control System

estimated model of the actual manipulator inertia matrix $H(q_p)$.

Likewise, $\hat{c}(q_p, \dot{q}_p)$, $\hat{c}_{grav}$ and $\hat{f}(q_p, \dot{q}_p)$ are our estimated models of the velocity terms, gravity terms and friction terms of the actual mechanism. Perfect knowledge of the model would mean:

$$\hat{H}(q_p) = H(q_p),$$

$$\hat{c}(q_p, \dot{q}_p) = c(q_p, \dot{q}_p),$$

$$\hat{c}_{grav} = c_{grav}$$

$$\hat{f}(q_p, \dot{q}_p) = f(q_p, \dot{q}_p) \quad (3.8)$$

Therefore, although the manipulator dynamics are given by eqn. (3.2), the new control law computes

$$\tau = a\tau' + b$$

$$a = H(\dot{q}_p) \quad (3.9)$$

$$b = c(q_p, \dot{q}_p) + c_{grav} \cdot g + f(q_p, \dot{q}_p)$$

Decoupling and linearizing will not therefore be perfectly

accomplished when the parameters are not exactly known. The mismatch between actual and modelled parameters will cause servo errors to be excited. A situation like this warrants the use of adaptive control, in order to satisfy eqn. (3.8).

In adaptive control, the servo errors are used to drive an adaptation scheme, which attempts to update the values of the model parameters until the errors disappear. Also, in the formulation of the control law (3.6, 3.7); use has been made of certain desired parameters such as θ_d , $\dot{\theta}_d$ and $\ddot{\theta}_d$. These can naturally be generated by a mathematically defined reference model, as in MRAC.

Like all adaptive systems, MRAC too has a 'learning' capability. This arises from the fact that not only do MRAC systems monitor the performance of the controlled plant, but they also monitor the performance of the control system itself.

Horowitz [42], in his doctoral dissertation, has proposed an MRAC scheme for mechanical manipulators. In it, he describes the application of this MRAC scheme to a rigid 3 degrees-of-freedom manipulator. Sutherland [43] also uses the same MRAC scheme in an application to a 2 degrees-of-freedom, rigid manipulator. He further investigates the applicability of MRAC methods to manipulators with flexible (elastic) links. This popular MRAC method was therefore selected to be applied to the control of a rigid 2-link, 2 degrees of

freedom manipulator (described in chapter 2). This manipulator will, henceforth in this thesis, be referred to by the acronym 'R2D2'. Here, the first two letters 'R2' stand for 2-link rigid manipulator, while the last two letters 'D2' refer to the two degrees-of-freedom of the manipulator.

A block diagram of the MRAC scheme proposed by Horowitz [42] is shown in Fig. 3.6. As can be seen from the figure, the manipulator and a mathematically defined reference model are operated in parallel. The performance of each is monitored in terms of joint angles and joint rates. Joint rate refers to the angular velocity of motion of one link with respect to the previous link, at that joint. Having thus obtained a means of assessing the performance of the plant and the model, the two performances are compared. This yields an error term, representing the difference in performances between the two. This error is used by the adaptation algorithm to modify or update the estimated values of $\hat{H}(q_p)$, $\hat{c}(q_p, \dot{q}_p)$ and $\hat{c}_{grav}(q_p)$. The adaptation algorithm therefore has to be designed to satisfy two conditions:

1. To reduce the performance error to a minimum
2. To guarantee overall stability of the entire system, at all times.

The dynamic equations for the manipulator 'R2D2', rewritten from eqn. (3.1) is:

$$\tau(t) = \hat{H}(q_p) \ddot{q}_p + \hat{c}(q_p, \dot{q}_p) + \hat{c}_{grav}(q_p) \quad (3.10)$$

Fig. 3.6 MRAC Scheme of Horowitz

and $\dot{\mathbf{q}}_p = \frac{d}{dt}(\mathbf{q}_p)$

The torque vector $\boldsymbol{\tau}(t)$ is calculated using the following control law (from eqn. (3.9)).

$$\boldsymbol{\tau}(t) = \hat{\mathbf{H}}(\mathbf{q}_p) \cdot \boldsymbol{\tau}' + \hat{\mathbf{c}}(\mathbf{q}_p, \dot{\mathbf{q}}_p) + \hat{\mathbf{c}}_{\text{grav}} \cdot \mathbf{g} \quad (3.11)$$

It is to be noted that the friction term has not been included in eqn. (3.11), as elaborated earlier. Denoting the new external controlling input $\boldsymbol{\tau}'$ by the input vector symbol $\mathbf{u}(t)$, eqn. (3.11) can be rewritten as :

$$\boldsymbol{\tau}(t) = \hat{\mathbf{H}}(\mathbf{q}_p) \cdot \mathbf{u}(t) + \hat{\mathbf{c}}(\mathbf{q}_p, \dot{\mathbf{q}}_p) + \hat{\mathbf{c}}_{\text{grav}} \cdot \mathbf{g} \quad (3.12)$$

Let us next specify the reference model. The model is to be an "ideal" one; in the sense that, given a controlling input $\mathbf{u}(t)$, it responds in a predictable manner. The response of this ideal system should not be affected by factors such as mass of the payload, configuration of the manipulator, etc. In other words, the output of the model is governed only by the controlling input and not by other disturbances. Such a reference model can be described by a double integrator, expressed in mathematical form as:

$$\dot{\mathbf{q}}_m(t) = \frac{d}{dt}(\mathbf{q}_m(t))$$

$$\mathbf{u}(t) = \frac{d}{dt} (\dot{\mathbf{q}}_m(t)) \quad (3.13)$$

where, $\mathbf{q}_m(t)$ and $\dot{\mathbf{q}}_m(t)$ are the angular position and velocity vectors of the reference model. The torque input is now given by:

$$\begin{aligned} \boldsymbol{\tau}(t) = & \hat{\mathbf{H}}(t) \cdot \mathbf{u}(t) + \hat{\mathbf{c}}(\mathbf{q}_p, \dot{\mathbf{q}}_p) + \hat{\mathbf{c}}_{\text{grav}} \cdot \mathbf{g} - \mathbf{F}_p[\mathbf{q}_p(t) - \mathbf{q}_m(t)] \\ & - \mathbf{F}_v[\dot{\mathbf{q}}_p(t) - \dot{\mathbf{q}}_m(t)] \end{aligned} \quad (3.14)$$

where $\hat{\mathbf{H}}(t)$ and $\hat{\mathbf{c}}(\mathbf{q}_p, \dot{\mathbf{q}}_p)$ are given by the adaptation algorithm listed below (for a 3 degrees of freedom manipulator).

$$\dot{\hat{h}}_{ii} = \alpha_{ii} v_i u_i \quad i = 1, 2, 3.$$

$$\dot{\hat{h}}_{ij} = \alpha_{ij} [v_i u_j + v_j u_i] \quad i = 1, 2 \quad j = i+1, \dots, 3$$

$$\dot{\hat{c}}_{ij}^{(1)} = \beta_{ij}^{(1)} [2\dot{q}_{pi} \dot{q}_{pj} v_i - \dot{q}_{pi}^2 v_j] \quad i = 1 \quad j = 2, 3.$$

$$\dot{\hat{c}}_{ii}^{(k)} = \beta_{ii}^{(k)} [\dot{q}_{pi}^2 v_k]$$

$$\dot{\hat{c}}_{23}^{(1)} = \beta_{23}^{(1)} [2\dot{q}_{p2} \dot{q}_{p3} v_1]$$

$$\dot{\hat{c}}_{13}^{(2)} = 2\beta_{13}^{(2)} [\dot{q}_{p1} \dot{q}_{p3} v_2 - \dot{q}_{p1} \dot{q}_{p3} v_3]$$

$$\hat{c}_{23}^{(2)} = 2\beta_{23}^{(2)} [\dot{q}_{p1} \dot{q}_{p2} v_2 - \dot{q}_{p2}^2 v_3] \quad (3.15)$$

where:

$\hat{h}_{ij}$ are elements of the estimated general inertia matrix $\hat{H}(t)$,

$\hat{c}_{ij}^{(k)}$ are the elements of the estimated cross-coupling term specifying the centrifugal and Coriolis effects,

u_i are the commanded accelerations,

and v_i are the generalized state errors.

The vector $v(t)$ is given by:

$$v(t) = C_p e(t) + C_v \dot{e}(t) \quad (3.16)$$

and

$$e(t) = q_p(t) - q_m(t) \quad (3.17)$$

$$\dot{e}(t) = \dot{q}_p(t) - \dot{q}_m(t)$$

The above adaptation algorithm is derived from principles of hyperstability theory and Popov's integral inequality. The hyperstability theory requires that the non-linear system under study be recast in a "standard" format. This format consists of splitting the system into two parts: a linear time-invariant feed-forward block and a non-linear, time-varying feedback block, as shown in Fig. 3.7.

Fig. 3.7 Standard Feedback System for Popov's Hyperstability Theory

Popov's Hyperstability theory can then be stated as follows:

"If the feedforward block is such that the feedback system is globally (asymptotically) stable for all feedback blocks satisfying the Popov integral inequality (PII), then the feedback system is said to be (asymptotically) hyperstable, and that the feedforward block is a hyperstable block."

Equation (3.18) states the PII.

$$\int_0^t \mathbf{w}^T \mathbf{v} \, d\tau \geq -\gamma_0^2 \quad \text{for all } t \geq 0 \quad (3.18)$$

where, $\mathbf{v}$ is the input vector of the feedback block, $\mathbf{w}$ is its output and γ_0^2 is a finite positive constant, which is independent of t . γ_0^2 depends only on the initial state of the system.

For a complete derivation of the adaptation algorithm, the reader is referred to Appendices H and I of [42]. Additional information on the hyperstability theory and its application in the design of MRAC systems can be found in [22] and [44].

The matrices $\mathbf{C}_p$ and $\mathbf{C}_v$ (eqn. 3.16) as well as $\mathbf{F}_p$ and $\mathbf{F}_v$ (eqn. 3.14) are square matrices selected by the designer. Horowitz suggests a method for selecting them as:

$$[\mathbf{C}_p \quad \mathbf{C}_v] = [\sigma_p \mathbf{I} \quad \sigma_v \mathbf{I}] \quad (3.19)$$

$$[F_p \quad F_v] = [\rho_p I \quad \rho_v I]$$

where, I is the identity matrix, and the four scalar variables σ_p , σ_v , ρ_p and ρ_v are all positive and satisfy:

$$\sigma_v > \sigma_p,$$

$$\sigma_v \rho_v I - \sigma_p H > 0$$

and

(3.20)

$$[\sigma_v \rho_p + \sigma_p \rho_v] I - \sigma_p H > 0$$

Asare and Wilson [45] have proposed an alternate MRAC scheme for robotic manipulators, based on a slightly different control law, the block diagram of which is depicted in Fig. 3.8.

The adaptation algorithm used by them is very similar to the one used by Horowitz. The control law is illustrated in Fig. 3.9, as applied to the reference model (pure double integrator)

There are two main differences between this control structure and Horowitz's method, described earlier:

1. Horowitz's method has two distinct feedback loops; an outer position feedback to provide the difference between desired and actual position. This difference is acted upon by the integral portion of the PID controller, before feedback values of position and velocity

are subtracted from it. The position and velocity feedback constitute the inner loop. Hence, the proportional and derivative portions of the PID controller act on the negative feedback segments.

In the Asare-Wilson method, a PD controller is used. The proportional gain acts on the position difference in the forward loop, while only velocity is fed back using the derivative gain.

2. Horowitz uses the output of the PID controller as input to both the plant and the model. Asare and Wilson apply the desired position directly as input to the model. This means that the model is further isolated from the plant, in the sense that feedback values of the plant variables are not used to process the input to the model. The model input is processed from its own output variables by the PID controller. This should make for a model which is more "ideal", as it is completely unaffected by the plant performance.

We can write the control law from Fig. 3.9 as:

$$\ddot{\mathbf{q}}_m + K_D \dot{\mathbf{q}}_m + K_p \mathbf{q}_m = K_p \mathbf{r} \quad (3.21)$$

where; $\mathbf{q}_m$, $\dot{\mathbf{q}}_m$ and $\ddot{\mathbf{q}}_m$ denote the position, velocity and acceleration vectors of the model,
 $\mathbf{r}$ is the desired position vector,
 and K_p and K_D are the PD controller gains.

Fig. 3.9 Control System for the Reference Model

The transfer function of eqn. (3.21) becomes:

$$T(s) = \frac{K_P}{s^2 + K_D s + K_P} \quad (3.22)$$

The poles of this second order system are:

$$s_{1,2} = \frac{-K_D \pm \sqrt{K_D^2 - 4 K_P}}{2} \quad (3.23)$$

For stability, we need $K_D > 0$

$$\text{and } K_P \geq 0.25 K_D^2 \quad (3.24)$$

Since each joint is now represented by a decoupled double integrator, the matrices K_P and K_D are diagonal matrices. It should however be noted that the terms K_P and K_D in eqns. (3.22) to (3.24) refer not to the matrix as a whole, but rather to its individual non-zero diagonal elements.

While it is adequate to satisfy eqn. (3.24) in order to be assured of a stable system, very often, it becomes necessary to specify the exact response of the system (e.g. underdamped with a fast initial rise, overdamped with no overshoot, or critically damped). Therefore, a relationship between the controller gains and the response characteristics is needed. Equation (3.21) can be rewritten as:

$$\frac{1}{K_P} \ddot{q}_m + \frac{K_D}{K_P} \dot{q}_m + q_m = r \quad (3.25)$$

This is similar to the form of a standard second-order system:

$$a \ddot{x} + b \dot{x} + x = r \quad (3.26)$$

By comparison:

$$a = 1 / K_p \quad \text{and} \quad b = K_D / K_p$$

where:

$$a = 1 / \omega_n^2, \quad b = 2\zeta / \omega_n.$$

ω_n is the natural frequency of the system and ζ the damping coefficient.

$$\text{Therefore, } K_p = \frac{1}{4\zeta^2} \cdot K_D^2 \quad (3.27)$$

For example, for the case of critical damping, $\zeta = 0.707$,

$$\text{thus, } K_p = 0.5 K_D^2 \quad (3.28)$$

While PD control is often adequate, addition of an integral term to the servo law would enable the system to reject disturbances caused by modeling errors or inaccuracies. The system would then have no steady-state error in the presence of biases of one kind or another. But with this control law, the system becomes a third order system. The control law then becomes:

$$\ddot{q}_m + K_D \dot{q}_m + K_p q_m + K_I \int q_m dt = K_p r + K_I \int r dt \quad (3.29)$$

Its characteristic equation is:

$$s^3 + K_D s^2 + K_P s + K_I = 0 \quad (3.30)$$

Solution of eqn. (3.30) gives guidelines for the choice of the PID controller gains

$$\left. \begin{aligned} s_1 s_2 + s_2 s_3 + s_3 s_1 &= K_P \\ -s_1 s_2 s_3 &= K_I \\ -(s_1 + s_2 + s_3) &= K_D \end{aligned} \right\} \quad (3.31)$$

CHAPTER IV

APPLICATION OF MRAC TO RIGID MANIPULATORS

4.1 Introduction

In Chapter 2, the dynamics of rigid manipulators was presented, based on an example of Lagrangian dynamics for the two-link rigid manipulator with two degrees of freedom (R2D2). In Chapter 3, adaptive control theory has been discussed with a focus on two specific MRAC methods.

In this chapter, the application of the MRAC method to the two-link manipulator is first described. A computer simulation is used to study the effectiveness of manipulator control. Simulations are also used to investigate the application of a slightly modified MRAC scheme, shown by Asare and Wilson, to various rigid manipulators. The performance of a two-link, rigid manipulator with three degrees of freedom (R2D3), as well as that of a three-link, rigid manipulator with three degrees of freedom (R3D3) have been investigated. Simulations results are presented and discussed.

4.2 Computer Simulation of R2D2

To investigate the dynamics of manipulators and their control systems, a computer simulation method has been used. This method has both advantages and disadvantages.

One of the primary advantages of a computer simulation method is its short development time, when compared to the time required to build, commission and test an experimental test-rig. Secondly, the cost involved in a simulation study is usually much less than that for an experimental set-up. And finally, the flexibility of a simulation study makes it extremely attractive. Major changes in design can be implemented easily and quickly.

One of the chief drawbacks of the simulation method is the fact that the authenticity of its results depend to a large extent on the accuracy of the modeling, and the assumptions made about true-life conditions.

Early in the course of this research, a feasibility study was conducted, to determine if MRAC could be used for commercial industrial manipulators. The Rhino XR-II series instructional robot available in the Control Systems laboratory at Carleton University, and the Unimation PUMA-560 industrial robot in the Department of Electrical Engineering at the National Research Council laboratories in Ottawa were studied. The difficulty of bypassing the controllers on

these robots, in order to implement other control strategies, and the time involved, were factors that influenced the author's decision to use the computer simulation method.

The Apollo work-station computers at Carleton University were used to implement the computer simulations. The simulations themselves were written in the 'C' programming language. The Domain Aegis and Unix operating systems on these machines made 'C' a logical choice.

The two-link rigid manipulator is assumed to be at rest in its initial horizontal position (both links extended horizontally along the X-direction): At this position, the state vectors of both manipulator and the reference model are initialized to zero.

$$\mathbf{q}_p^T = \{ \theta_{p1} \quad \theta_{p2} \} = \{ 0 \quad 0 \} \text{ rad}$$

$$\dot{\mathbf{q}}_p^T = \{ \dot{\theta}_{p1} \quad \dot{\theta}_{p2} \} = \{ 0 \quad 0 \} \text{ rad/s}$$

$$\ddot{\mathbf{q}}_p^T = \{ \ddot{\theta}_{p1} \quad \ddot{\theta}_{p2} \} = \{ 0 \quad 0 \} \text{ rad/s}^2 \quad (4.1)$$

$$\mathbf{q}_m^T = \{ \theta_{m1} \quad \theta_{m2} \} = \{ 0 \quad 0 \} \text{ rad}$$

$$\dot{\mathbf{q}}_m^T = \{ \dot{\theta}_{m1} \quad \dot{\theta}_{m2} \} = \{ 0 \quad 0 \} \text{ rad/s}$$

$$\ddot{\mathbf{q}}_m^T = \{ \ddot{\theta}_{m1} \quad \ddot{\theta}_{m2} \} = \{ 0 \quad 0 \} \text{ rad/s}^2 \quad (4.2)$$

In order to obtain the response of a particular system, the system is excited by impulse inputs, step inputs, ramp inputs or higher order inputs. Accordingly, a step input is used here to obtain the response of the manipulator. The step input is specified in terms of the desired final position vector $\mathbf{r}_p$. For example, $\mathbf{r}_p^T = \{0.5, 1.0\}$ rad requires the robot to move to, and come to a final stop at a position in which its first joint angle is 0.5 radian and its second 1.0 radian.

In fact, there is a routine "CART2POL" in the program, which computes the inverse kinematics of the manipulator. This routine takes the user-specified Cartesian coordinates of the final end-effector position as its input, performs the inverse kinematics and provides the main program with the step input vector $\mathbf{r}_p$.

In the simulation, there are several instances requiring numerical integration; e.g. the integral portion of the PID controller, and the double integrators ($1/s^2$) representing both model and plant. Sutherland [43] reasons that the Runge-Kutta fourth order method is best suited for numerical integration, from a stability point of view. Hence, this method is used for numerical integration, with a time-step of 1 millisecond.

The parameters used in the simulation were as follows:

Coordinates of the final end-effector position,

$$(x, z) = (948.32, 1476.92) \quad \text{mm}$$

Resulting final position joint angles,

$$\theta_{f1} = 0.5 \text{ rad} \quad \text{and} \quad \theta_{f2} = 1.0 \text{ rad}$$

Gain matrices for the error vectors,

$$C_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C_v = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

Stability matrices,

$$F_p = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

$$F_v = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

Gains for the adaptation algorithm,

$$\alpha = \begin{bmatrix} 5 & 0.01 \\ 0.01 & 5 \end{bmatrix}$$

$$\beta = \begin{bmatrix} 10^{-3} & 2 \\ 10^{-3} & 0.01 \end{bmatrix}$$

Link and payload masses,

$$m_1 = 10 \text{ Kg}$$

$$m_2 = 5 \text{ Kg}$$

$$\text{payload} = 0 \text{ Kg}$$

Link lengths,

$$l_1 = l_2 = 1 \text{ m}$$

Link cross-sections,

$$\text{Diameter of Link-1} = 4.04 \text{ cm}$$

$$\text{Diameter of Link-2} = 2.86 \text{ cm}$$

The step response of the two-link manipulator is shown in Fig. 4.1. The PID controller gains affect the response characteristics of the manipulator. After various trials, they were set as follows for the critical damping case (Fig. 4.1)

Fig. 4.1 Step-response of R2D2, with MRAC
(critical damping)

$$\mathbf{K}_p = \begin{bmatrix} 80 & 0 \\ 0 & 80 \end{bmatrix}$$

$$\mathbf{K}_I = \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}$$

$$\mathbf{K}_D = \begin{bmatrix} 30 & 0 \\ 0 & 30 \end{bmatrix}$$

In these trials, the following guidelines were used to arrive at the values shown in the above section:

- The proportional gain term $\mathbf{K}_p$ determines the initial-rise time required for the response to attain the desired value for the first time; higher the gain, lower the response time.

- The integral term $\mathbf{K}_I$ is required to reduce the steady-state error to a minimum. Its choice also affects the oscillations in the response and the system stability.

- The derivative term $\mathbf{K}_D$ largely affects the nature of the response characteristic, in terms of underdamped (Fig. 4.2), overdamped or critically damped (Fig. 4.1). It also makes the control system sensitive to the rate of change of the controlled variable.

The decoupling of the dynamic equations of the manipulator, achieved by MRAC, can be seen from a comparison of Fig. 4.1 and Fig.

Fig. 4.2 Step-response of R2D2, with MRAC (underdamped)

4.3. In the latter, the simulation has been run without the adaptation algorithm, given in eqns. (3.15). The response now is no longer as smooth as in Fig. 4.1. The kink in the position profile is due to the coupling between the links, which is quite marked at this instant, when the motion of the links changes from acceleration to deceleration. The velocity profile, shown in Fig. 4.4, is very smooth in nature, also confirming the success of the non-linearity compensation and decoupling control of MRAC.

To investigate the insensitivity of MRAC to payload variations, the simulations are run with different payloads. The responses (Figs. 4.5 - 4.6) indicate that MRAC indeed succeeds in maintaining an optimum and unvarying response to a given input, even when the payload varies. This important property - payload insensitivity - which is unique to adaptive control, is what renders MRAC so attractive for manipulator control applications.

The adaptation algorithm used by Horowitz [42] gives the first derivatives of the adaptation gains with respect to time. Sutherland [43] however, while using the same adaptation algorithm, implements it differently in his computer simulation. He uses the 'delta' operator to add subsequent values of the gains to an already established 'gain-memory'.

In this thesis, the adaptation algorithm is implemented to provide the time-derivatives of the gains (as in [42]). This therefore

Fig. 4.3 Step-response of R2D2, without Adaptation (critical damping)

Fig. 4.4 Angular Velocity Response of R2D2, with MRAC

Fig. 4.5 Step Response of R2D2, with MRAC (payload = 3 Kg)

Fig. 4.6 Step Response of R2D2, with MRAC
(payload = 5 Kg)

requires a real-time solution by the process of numerical integration.

As mentioned earlier, all the numerical integration was performed using the Runge-Kutta fourth order method. Subsequently, various lower order Runge-Kutta methods as well as the Euler integration method were used. No discernible differences were observed, either in stability or in accuracy of the results, for all the above methods. The reason for this is the very small step-size (1 millisecond) used for numerical integration. Therefore, the Euler method was adopted in subsequent simulations. This change greatly simplified the integration procedure, in addition to making for a more compact computer code.

4.3 Evaluation of another MRAC Scheme

In the last section, the MRAC scheme proposed by Horowitz [42] was used to control a two degrees-of-freedom manipulator. In this section, a slightly different version of MRAC (Fig. 3.8) is used to control the same manipulator. As will be seen, the success of this implementation led to its use in the control of higher degrees-of-freedom manipulators (R2D3 and R3D3).

Asare and Wilson [45] have proposed an alternate configuration of an MRAC scheme. This scheme differs from the one used by Horowitz, in that it dispenses with the need for the stability matrices

F_p and F_v by defining the control law differently.

Asare and Wilson have also concluded that an MRAC scheme in which, the parameters of the system (H and c) are continuously modified is more stable and performs better trajectory tracking than MRAC schemes in which the PID controller gains are modified. This is because, in the latter method, simplification and linearization of the non-linear system is necessary, in order to apply optimizing algorithms to the controller. However, this simplification ignores certain non-linear terms, giving rise to modeling inaccuracies and eventual system instability.

The step response of R2D2 manipulator is shown in Fig. 4.7, using a simulation based on this MRAC scheme. It was observed that this MRAC scheme provides results comparable to those of the previous section . However, the implementation in the computer simulation is simpler and shorter. Therefore, it was decided to use this MRAC scheme for the control of higher degrees-of-freedom manipulators.

4.4 MRAC for Higher Degrees-of-Freedom Manipulators

The manipulator R2D2 is a simple manipulator with only two links and a planar movement capability. In this section, more complex manipulators are simulated. Adding a third degree of

Fig. 4.7 Step Response of R2D2
(the second MRAC method)

freedom to R2D2 would render it more versatile by making it three-dimensional. Such a two-link, rigid manipulator with three degrees-of-freedom (R2D3) is shown in Fig. 4.8.

The Lagrangian approach, as described in Chapter 2 is used once again for the derivation of the dynamic equations of this manipulator. In this case, joint-1 possesses two degrees of freedom; a 'yaw' (θ_1) in the horizontal XY plane, and a 'pitch' (θ_2) in the vertical XZ plane; while joint-2 is capable of only the 'pitch' degree of freedom (θ_3). The complete dynamic equations are derived in Appendix-1 of this thesis. The final torque equations are:

$$\begin{aligned} \tau_1 = & \ddot{\theta}_1 [m_{12}l_1^2 C_2^2 + m_2 l_2^2 C_{23}^2 + 2m_2 l_1 l_2 C_2 C_{23}] \\ & - 2\dot{\theta}_1 \dot{\theta}_2 [m_{12}l_1^2 C_2 S_2 + m_2 l_2^2 C_{23} S_{23} + m_2 l_1 l_2 \sin(2\theta_2 + \theta_3)] \\ & - 2\dot{\theta}_1 \dot{\theta}_3 [m_2 l_2^2 C_{23} \widehat{S_{23}} + m_2 l_1 l_2 C_2 S_{23}] \end{aligned} \quad (4.3)$$

$$\begin{aligned} \tau_2 = & \ddot{\theta}_2 [m_{12}l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 C_3] \\ & + \ddot{\theta}_3 [m_2 l_2^2 + m_2 l_1 l_2 C_3] \\ & + \dot{\theta}_1^2 [m_{12}l_1^2 C_2 S_2 + m_2 l_2^2 C_{23} S_{23} + m_2 l_1 l_2 \sin(2\theta_2 + \theta_3)] \\ & - \dot{\theta}_3^2 [m_2 l_1 l_2 S_3] \\ & - 2\dot{\theta}_2 \dot{\theta}_3 [m_2 l_1 l_2 S_3] \\ & + g [m_{12}l_1 C_2 + m_2 l_2 C_{23}] \end{aligned} \quad (4.4)$$

Fig. 4.8 R2D3 Manipulator Configuration.

$$\begin{aligned}
\tau_3 = & \ddot{\theta}_2 [m_2 l_2^2 + m_2 l_1 l_2 C_3] \\
& + \ddot{\theta}_3 [m_2 l_2^2] \\
& + \dot{\theta}_1^2 [m_2 l_2^2 C_{23} S_{23} + m_2 l_1 l_2 C_2 S_{23}] \\
& + \dot{\theta}_2^2 [m_2 l_1 l_2 S_3] \\
& + g [m_2 l_2 C_{23}]
\end{aligned} \tag{4.5}$$

where, the same notation has been used as in Chapter 2.

The step response characteristics of R2D3 are shown in Fig. 4.9. Once again, the figures show the effectiveness of MRAC control for three-dimensional manipulators. Further, the addition of an orthogonal degree-of-freedom to the two degrees-of-freedom manipulator does not in any way make a difference to its performance under MRAC.

Next, the three-link planar manipulator (Fig. 4.10) with three degrees-of-freedom was considered for control and simulation. The derivation of the dynamic equations is shown in Appendix-2.

$$\begin{aligned}
\tau_1 = & \ddot{\theta}_1 [m_1 l_2^2 + m_2 l_2^2 + m_3 l_3^2 + 2m_2 l_1 l_2 C_2 + \\
& 2m_3 l_1 l_3 C_{23} + 2m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_2 [m_2 l_2^2 + m_3 l_3^2 + m_2 l_1 l_2 C_2 + m_3 l_1 l_3 C_{23} + \\
& 2m_3 l_2 l_3 C_3]
\end{aligned}$$

Fig. 4.9 Adaptive Step Response of R2D3 Manipulator

Fig. 4.10 R3D3 Manipulator Configuration.

$$\begin{aligned}
& + \ddot{\theta}_3 [m_3 l_3^2 + m_3 l_1 l_3 C_{23} + m_3 l_2 l_3 C_3] \\
& - 2\dot{\theta}_1 \dot{\theta}_2 [m_{23} l_1 l_2 S_2 + m_3 l_1 l_3 S_{23}] \\
& - 2\dot{\theta}_1 \dot{\theta}_3 [m_3 l_2 l_3 S_3 + m_3 l_1 l_3 S_{23}] \\
& - 2\dot{\theta}_2 \dot{\theta}_3 [m_3 l_2 l_3 S_3 + m_3 l_1 l_3 S_{23}] \\
& - \dot{\theta}_2^2 [m_{23} l_1 l_2 S_2 + m_3 l_1 l_3 S_{23}] \\
& - \dot{\theta}_3^2 [m_3 l_1 l_3 S_{23} + m_3 l_2 l_3 S_3] \\
& + g [m_{123} l_1 C_1 + m_{23} l_2 C_{12} + m_3 l_3 C_{123}] \quad (4.6)
\end{aligned}$$

$$\begin{aligned}
\tau_2 = & \ddot{\theta}_1 [m_{23} l_2^2 + m_3 l_3^2 + m_{23} l_1 l_2 C_2 + m_3 l_1 l_3 C_{23} + 2m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_2 [m_{23} l_2^2 + m_3 l_3^2 + 2m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_3 [m_3 l_3^2 + m_3 l_2 l_3 C_3] \\
& - 2\dot{\theta}_1 \dot{\theta}_3 [m_3 l_2 l_3 S_3] \\
& - 2\dot{\theta}_2 \dot{\theta}_3 [m_3 l_2 l_3 S_3] \\
& + \dot{\theta}_1^2 [m_{23} l_1 l_2 S_2 + m_3 l_1 l_3 S_{23}] \\
& - \dot{\theta}_3^2 [m_3 l_2 l_3 S_3] \\
& + g [m_{23} l_2 C_{12} + m_3 l_3 C_{123}] \quad (4.7)
\end{aligned}$$

$$\begin{aligned}
\tau_3 = & \ddot{\theta}_1 [m_3 l_3^2 + m_3 l_1 l_3 C_{23} + m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_2 [m_3 l_3^2 + m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_3 [m_3 l_3^2]
\end{aligned}$$

$$\begin{aligned}
& + 2\dot{\theta}_1 \dot{\theta}_2 [m_3 l_2 l_3 S_3] \\
& + \dot{\theta}_1^2 [m_3 l_2 l_3 S_3 + m_3 l_1 l_3 S_{23}] \\
& + \dot{\theta}_2^2 [m_3 l_2 l_3 S_3] \\
& + g [m_3 l_3 C_{123}]
\end{aligned} \tag{4.8}$$

The results of the simulation for R3D3 are shown in Figs. 4.11 and 4.12.

Fig. 4.11 Adaptive Step Response of R3D3 Manipulator

Fig. 4.12 Non-adaptive Step Response of R3D3 Manipulator

CHAPTER V

APPLICATION OF MRAC TO FLEXIBLE MANIPULATORS

5.1 Introduction

Today's industrial robots usually have several degrees of freedom (from 4 to 6), and are supported by sophisticated controllers and several sensors. The precise positioning of the end-effector, or the tip of the last link is vital for the performance of a number of tasks. This positioning of the end-effector of the manipulator is normally achieved by controlling the torques and forces applied by the various actuators located at the joints.

This method of joint control translates into the control of the end-effector, only when the link geometry (Kinematics) remains constant. This would mean that the shape and geometry of the structural links of the robot need to remain unchanged, even when the robot carries a heavy payload and executes a fast movement. In other words, the structural links of a manipulator would have to be perfectly rigid.

The positional accuracy of a robot can be degraded considerably, if this exacting condition on rigidity is not maintained.

However, in order to construct the links of the manipulator for rigidity, one must pay the price in terms of mass. Heavy structural links require large actuators to move them. Large actuators will in turn consume more power. This will affect the payload capacity as well as the speed of operation adversely. Besides, the resulting robot will make it very difficult to transport it from one location to another. This excessive weight is a problem in certain applications, such as Space.

All these problems, which stem from the heavy weight of the links, can be overcome by relaxing the rigidity constraint. This would enable the robot designer to construct the links of non-conventional robot structurals, such as aluminum alloys and composite materials. The resulting slender, light-weight links would be strong in compression, but comparatively weaker in flexure (bending). Effective control of such a flexible manipulator would render it invaluable in space and undersea exploration, as well as remote or hazardous applications, such as the nuclear industry.

In this chapter, MRAC methods are applied to the control of manipulators with flexible links. In order to do that, flexibility of the links needs to be modelled first, so that their effect on the dynamics of the manipulator will be known. Link flexibility, thus modelled would aid in the prediction of link deflection. In order to maintain tracking performance of the manipulator end-effector, the deflection of the links has to be compensated for in some manner. The

performance of flexible manipulators with MRAC and an algorithm for deflection compensation has been investigated in this chapter.

5.2. Modeling Link Flexibility

Several researchers have studied the problem of controlling flexible manipulators. In his paper, Sasiadek [46] investigates the performance of a generic two-link manipulator in a gravity field. He uses a model based on Lagrangian Dynamics coupled with a finite-element deflection approach.

Schafer and Holzach [47] describe their experimental research on the active vibration control of a flexible cantilever beam. They report a good agreement between their experimental and simulated results. Meldrum and Balas [48] have studied the effectiveness of MRAC control on a single-link, flexible remote manipulator. They conclude that MRAC shows promise in the position control of a single-link, flexible manipulator with a collocated sensor and actuator at its tip.

Usoro et al. [49], [50], [51] describe a finite element / Lagrange approach to modeling light-weight, flexible manipulators. Another work in which experimental results are compared with a simulated model of a flexible single-link manipulator is by Hastings and Book

[52]. Ozguner and Yurkovich [53] also describe their experiments on the active vibration control of a flexible beam.

In modeling the forces and torques acting on the different parts of a robotic manipulator, use can once again be made of the dynamic equations. In these equations, the various forces and torques (inertial, Coriolis, centripetal, coupling, gravitational and friction, if modelled) acting on the manipulator are represented. The left-hand side of the dynamic equations τ_i then represents the resultant torque at each joint due to all the above forces and torques.

This concept is very useful in the modeling of flexible manipulators ([54], [55]). Each link- i of a flexible manipulator can be considered to be a cantilever beam, fixed(clamped) at its proximal joint- i and free at its distal joint($i + 1$).

A simplifying, albeit conservative assumption made here is that the weight of each link (and payload, if any) is assumed to be acting through the tip of that link. From classical beam theory for cantilever beams acted upon by moment loads at the ends, we have the following equations for deflection and slope at each link-tip.

$$\delta_{\text{(link-tip)}} = - Tl^2 / 2E I_n \quad (5.1)$$

$$\text{slope}_{\text{(link-tip)}} = - Tl / E I_n \quad (5.2)$$

where:

T is the moment load in N-m acting on the link, equal to τ_i in the dynamic equations;

l is the link length in metre;

E is the modulus of elasticity in N / m^2 ;

and I_n is the Moment of Inertia of the beam cross-section about the neutral axis, which passes through the centroid of the section in m^4 ,

$= \pi r^4 / 4$; r being the radius of the link cross-section, in m.

The validity of equations (5.1) and (5.2) is based on the following assumptions.

1. The elastic properties of the material of the manipulator links is governed by Hooke's law.
2. The material of the links is isotropic and homogeneous.
3. The beam is subjected to pure bending.
4. The beam is straight, with a constant cross-section.
5. The properties of the beam are such that it will fail by bending.
6. The beam has an axis of symmetry in the plane of bending.
7. Cross-sections of the beam remain plane during bending.

Another important criterion which must be true for the above bending theory to be applicable is the condition of small-scale deflection. Large-scale deflection of beams requires a totally different and more complex analysis. Although it is difficult to

precisely separate small-scale and large-scale bending, several criteria are currently in use to do this. Hopkins [56] has it that the theory of small-scale bending can be applied, as long as the ratio of vertical deflection to beam length does not exceed 0.2.

5.3 Theory of Deflection Compensation

The bending of the flexible links of a manipulator affect the positional accuracy of its end-effector in many ways. In order to incorporate deflection compensation in the design of a controller, the mechanism of deflection must be well understood. The various types of link deflection and the manner in which they can be compensated for is explained by means of an example.

Consider the two-link manipulator with two degrees of freedom (R2D2). If the links of this manipulator are made flexible by proper choice of material and geometric proportions, it can now be called a two-link *flexible* manipulator with two degrees of freedom (F2D2).

The process of deflection compensation for F2D2 consists of the following three steps:

1. Correcting the angular position of link-1 (Fig. 5.1) to restore joint-2 to a location that would have been occupied by an equivalent rigid link.
2. Correcting the angular position of link-2 to overcome the

Fig. 5.1 Deflection Compensation, Type 1

deviation caused by a change in the flexural slope at the end of link-1 (i.e. at joint-2); as shown in Fig. 5.2.

3. Correcting the angular position of link-2 to restore the end-effector to its target location (Fig. 5.3).

Steps 1 and 3, which are similar in nature, are achieved as illustrated in Fig. 5.4. The original MRAC simulation generates values for the output of the manipulator, viz. joint angles, joint rates and joint accelerations. These values are used to compute joint torques, or the moments acting on each link of the manipulator. These moments in turn are needed to compute deflection and slope at the tip of each link.

Since the simulation step-size is as small as 1 millisecond, it is safe to assume that δ and hence $\delta\theta$ (in Fig. 5.4) are small. Therefore,

$$\delta\theta = \delta / l \quad (5.3)$$

Also, because ' δ ' is very small, it can be assumed that when the angular position of the link is $(\theta + \delta\theta)$, the deflection would almost equal δ , again. Therefore, in order to return the link-tip to the undeflected position, one of two steps must be taken:

1. The current link position can be updated by $(-\delta\theta)$; i.e. as though an equivalent rigid link were positioned at $(\theta - \delta\theta)$. This is

2

MicroD

Fig. 5.2 Deflection Compensation, Type 2

Fig. 5.3 Deflection Compensation, Type 3

Fig. 5.4 Deflection Compensation, Types 1 and 3

illustrated by the discontinuous line in Fig. 5.4. The control system then drives the link further until the end-effector reaches the target position.

or

2. The desired final position of the link, represented by r can be updated by $(+\delta\theta)$ to $(r + \delta\theta)$. Similar to the first method, this too has an identical effect on the flexible manipulator. The control system uses the new value of r to drive the end-effector on target.

For step 2 of the deflection compensation procedure, the following analysis is used. Since link-2 is positioned relative to link-1, any change in the orientation of link-1 would affect the positioning of link-2 in space. In the deflected state, the end of link-1 makes a different angle with the coordinate frame than the portion of the link closest to joint-1. Denoting this difference by Φ (Fig. 5.2), and the angular orientations of the two links by θ_1 and θ_2 , we can denote the position of the end of link-1 as $(\theta_1 - \Phi)$. The global position of link-2 then becomes $(\theta_1 + \theta_2 - \Phi)$ instead of the desired $(\theta_1 + \theta_2)$. To correct for this deviation, θ_2 needs to be updated to $(\theta_2 + \Phi)$. All the above procedures are executed in an iterative manner in the computer simulation.

The deflection compensation theory described above acts as an

"interface" between the dynamics of the flexible manipulator and the MRAC controller. Such an interface is necessary because the MRAC controller is designed for rigid manipulators, and as such cannot be used directly to control a flexible manipulator. In the following section, results of the simulations of flexible manipulators with MRAC control are presented.

5.4 MRAC with Deflection Compensation for Flexible Manipulators

A computer simulation was developed first for F2D2, and subsequently modified for F2D3 and F3D3 (the acronyms F2D2, etc. represent flexible versions of their rigid equivalents, R2D2, etc.). The deflection compensation for F2D3 and F3D3 are similar to that described for F2D2, except more elaborate.

The parameters of the flexible manipulators have to be chosen carefully. While on the one hand, discernible link deflections are desirable for purposes of investigation, the deflections should be limited to the small-scale domain. With the above considerations in mind, the following parameters were chosen for the flexible manipulators. The links are cylindrical with a solid circular cross-section of the following dimensions:

Manipulator	Link-1		Link-2		Link-3	
	Dia.(cm)	Mass(Kg)	Dia.(cm)	Mass(Kg)	Dia.(cm)	Mass(Kg)
F2D2 & F2D3	6.86	10	4.86	5		
F3D3	6.86	10	4.86	5	4.34	4.0

Table 5.1 Flexible Manipulator Parameters

Material: Aluminum with Young's modulus $E = 69.1 \text{ kN/mm}^2$ and density $\rho = 2700 \text{ Kg/m}^3$.

In the following simulations, the results are presented in the form of plots of end-effector error versus time. The error refers to the positioning error between the end-effector of the flexible manipulator and its rigid equivalent. In other words, the position of an equivalent rigid manipulator is taken as the reference coordinate, against which the position of the flexible manipulator is compared. These comparisons are shown in Figs. 5.5, 5.6 and 5.7.

The purpose of standardizing the link masses of all three simulations (Table 5.1) is to eliminate any effect of change in the

Fig. 5.5 End-effector Error Profiles for F2D2 and F2D3

Fig. 5.6 End-effector Error Profiles for F2D3 and F3D3

Fig. 5.7 End-effector Error Profiles for F2D2 and F3D3

link-mass on the comparisons. The error responses are now governed solely by the number of degrees-of-freedom. It should be noted that the forcing function used here is the same one as described by equations 6.1 and 6.9. This forcing function varies from 0 to 5 seconds and then remains constant beyond 5 seconds.

In that period (0 - 3.5 sec), the manipulator 'F2D2' has the greatest error. Whereas, the manipulator 'F3D3' displays a highly oscillatory behavior around the transition zone (3.5 - 6.5 sec). Once again, the manipulator 'F2D2' is subject to heavy oscillations beyond 6 seconds. On the contrary, the manipulator 'F2D3' seems to have the lowest error (< 5 mm), throughout. These results indicate a strong dependence of the positional accuracy of the end-effector on the number of degrees of freedom of that manipulator. They also indicate that adding an orthogonal degree of freedom to a planar manipulator to make it three-dimensional, seems to improve the positioning accuracy tremendously. This suggests that the addition of such an orthogonal degree of freedom improves the damping characteristics of the flexible manipulator.

Since the error response of the flexible manipulator 'F2D2' is found to be oscillatory, an attempt was made to modify the deflection compensation algorithm suitably. Once the forcing function reaches a constant value ($t > 5$ second), the deflection compensation routine takes into account only the deflections due to gravity (weight). This modification produces better results (Fig. 5.8 and 5.9).

Fig. 5.8 End-effector Error Profiles for F2D2, upto 10 sec

Fig. 5.9 End-effector Error Profile for F2D2, beyond 10 sec

Nevertheless, a mild oscillation persists along with a steady-state error.

To examine if this unsteady behavior is mass-dependant, another simulation of the manipulator 'F2D2' is performed, with link masses 4 and 1 Kg. respectively. The resulting plot (Fig. 5.10) shows a steady error-profile with the error converging to zero. This indicates that the mass-distribution of the links is also a critical factor for the deflection control of flexible manipulators.

5.5 Summary

In this chapter, deflection of the links of a flexible manipulator are first modelled. A suitable deflection compensation algorithm is developed, which is then used in conjunction with MRAC methods in an attempt to investigate the effectiveness of flexible manipulator control.

Simulation results lead to the following conclusions:

- A three-dimensional flexible manipulator (having atleast one orthogonal d.o.f.) performs better trajectory tracking than a planar (two-dimensional) flexible manipulator, under MRAC.
- Tracking accuracy of the end-effector is largely dependant on the distribution of link-masses (proportion of the masses of the two links) of a flexible manipulator.

Fig. 5.10 End-effector Error Profile for F2D2

- It is the author's opinion that a method based on a combination of MRAC and active vibration control (low-authority control, [57]) would achieve better positional and vibration control of a flexible manipulator, than MRAC alone.

CHAPTER VI

MRAC FOR DUAL-ARM MANIPULATORS

6.1 Introduction

This chapter describes the application of Model Reference Adaptive Control (MRAC) to dual-arm manipulators. MRAC has been successfully applied to the control of single, rigid manipulator arms, as described in Chapter 4. The ability of MRAC for non-linearity compensation and decoupling control, as well as its capacity to be unaffected by variations in the payload mass, it was felt, would make it an ideal tool for experimentation with dual arms [58].

A dual-arm manipulator is a set of two identical manipulator arms, working together in a coordinated manner, to accomplish complicated tasks. The need for such dual-arm manipulators is felt where large, heavy or unwieldy objects make it difficult for a single, large arm to handle them effectively. Such requirements arise in the nuclear industry, and in undersea and space exploration.

One of the most noticeable and significant dual-arm applications is the one related to Space. Assembly of a space-station.

could be done with the use of robots. Parts of a space-station may not be very heavy (in fact, they would be weightless in Space), but they may be large and difficult to handle. Large and irregular shapes require large and heavy manipulators to transport them. These in turn, require big rockets to lift them up into Space. The costs of such operations may be prohibitive. The expenses could be lowered by placing two lighter robots into orbit, rather than a heavy one. Two manipulators, working in coordination are capable of performing more complex operations successfully. However, they require a sophisticated control system to coordinate tasks and operations of the two arms working simultaneously. Also, the position and force control system of two coordinated arms is more difficult to design, than that of a single arm [59 - 63].

Tasks, assigned to dual-arm manipulators can be classified as:

1. Tasks being done by the two arms together, where each arm has an equal contribution.
2. Tasks being done by the two arms together, where one arm serves as the main (leading) arm, and the second performs an auxiliary role.
3. Tasks being done independently by each arm.

While using dual-arms, there is usually a requirement for the end-effectors of both arms to track prescribed trajectories. However for instance, when the dual-arm is used to transport an irregular object, the payloads associated with each arm may be different.

Even in such a situation, the tracking performance of the manipulators should not be adversely affected. Hierarchical adaptive control is used here to achieve this end.

6.2 Hierarchical Adaptive Control for a Planar Dual-Arm Manipulator

The dual-arm manipulator selected for the simulation study consists of two rigid, double-link manipulator arms. The model of 'R2D2' used in earlier chapters is again used here for each arm. The two arms of the dual-arm manipulator are identical to each other in all respects.

The following assumptions were made to simplify the modeling:

1. Each arm is constrained to move within its own vertical plane; i.e. the vertical plane passing through its base.
2. The dynamics of the actuators are neglected for modeling purposes.
3. The links are assumed to be perfectly rigid.
4. The joints are considered to be frictionless.
5. Each arm starts its motion from rest at the horizontal position.

The bases of the two arms were set up with a fixed distance between them. And since each arm moves within its own vertical plane only, the horizontal distance between the two end-effectors remains constant as well.

This in effect, simplifies the problem to one of coordination of the two arms in two dimensions. The assumption here is that the two arms, having been stationed with a desired distance between them, have to follow identical trajectories within their own planes of operation. Such a trajectory should be followed by both end-effectors with respect to space and time; that is to say that all three motion parameters (position, velocity and acceleration) of the two arms must be identical. This situation can be visualized in an application where the two arms have to transport a large and irregular object together. The two arms are initially positioned with the required distance between them. They are then required to transport the object so that it does not tilt or change its orientation in space, despite the different payloads acting on the two arms.

The planar trajectory for each arm, shown in Fig. 6.1 was specified as follows:

$$\theta_{d1}(i) = [1 - \cos(\psi(i))] / 2$$

(6.1)

$$\theta_{d2}(i) = \theta_{d1}(i) / 2$$

Fig. 6.1 Trajectory for each arm of Planar Dual Manipulator -

where:

θ_{d1} and θ_{d2} are the elements of the desired position vector r_p ;

$$\psi(i) = i \times \pi / 5000;$$

and $i = 0, 1, 2, \dots, 5000$ is the iteration number.

Beyond 5000 iterations, r_p remains constant.

The payload being carried by the two arms being an irregular object, the payload to be carried by each arm would be different. This payload difference between the two arms was set arbitrarily at 1 Kg. This difference would cause the two trajectories to be different, when non-adaptive control is used. This difference is sought to be minimized; indeed eliminated by means of adaptive and hierarchical control.

The simulation for the two-link manipulator, described earlier, is used as a basis for the simulation of the dual-arm manipulator. The essential difference is that there are now two arms being simultaneously simulated in parallel. Parameters of the dual manipulator are as follows:

<u>SIMULATION DATA</u>				
<u>PARAMETER</u>	<u>ARM-1</u>		<u>ARM-2</u>	
	<u>Link-1</u>	<u>Link-2</u>	<u>Link-1</u>	<u>Link-2</u>
1. Link length (m)	1	1	1	1
2. Cross-sectional dia (mm)	40.4	38.3	40.4	38.3
3. Link mass (Kg)	10	9	10	9
4. Payload (Kg)	0		1	

Table 6.1 Dual-arm parameters

The simulation of the dual-arm manipulator is first run without the adaptation algorithm. The tracking error between the arms, 'd' is plotted against time, as shown in Fig. 6.2. As can be seen, the error increases rapidly to a maximum of approximately 44 mm, at 2.2 seconds. In this acceleration phase, the arm with the heavier payload is slower to respond, resulting in increasing error. However, after about two seconds, this heavier arm begins to catch up with the lighter one, thus narrowing the error down to a minimum at 3.6 seconds. Subsequently, the deceleration phase commences, and once again, the response of the heavier arm is slower. This results in a cross-over of trajectories and a consequent increase in the error. Finally, beyond about 5 seconds, the tracking error begins its final convergence to zero. This corresponds to the

Fig. 6.2 Tracking Error with non-adaptive Control

moment in time from when on the exciting signal r_p remains constant. Beyond 9 seconds tracking error of the end-effector is less than 5 mm.

Next, the MRAC simulation of the dual-arm manipulator is run. This time, the adaptation algorithm is included in the simulation. The results are illustrated in Fig. 6.3, in the form of a plot of tracking error versus time.

The most prominent difference between figures 6.2 and 6.3 is the drastic reduction in the maximum error, from 44 mm to 2.5 mm. Further, the second maximum in 'd', which occurs during the deceleration phase, is much smaller than the first. Additionally, the error too converges more rapidly to zero. Once again, these results prove the ability of MRAC for non-linearity compensation and decoupling control, as well as its insensitivity to payload variations.

Thus far, the two arms are being controlled independently by their MRAC controllers. There is no mechanism to coordinate their movement. It was felt that the addition of such a coordinating mechanism between the two arms would further improve the tracking integrity between the two arms. This then was the motivation for Hierarchical Adaptive Control.

The Hierarchical Control proposed in this thesis consists of two levels of control, as shown in Fig. 6.4. The upper level is the

Fig. 6.3 Tracking Error with MRAC

Fig. 6.4 Hierarchical Adaptive Control Structure

supervisory control. The second level comprises the two MRAC schemes that control the two individual arms. This level is split; i.e. MRAC for arm-A is slightly higher in hierarchy than that for arm-B. This means that the trajectory of arm-A is taken as a reference, to be followed by arm-B. In the event that the two arms track identical trajectories, the lower level of control remains autonomous from the supervisory level; i.e. each arm is controlled entirely by its own MRAC controller alone. But when the two trajectories begin to differ, the supervisory controller adapts the MRAC controller of arm-B to force it follow the reference trajectory of arm-A.

The performances of both arms are compared, and any difference is fed to the supervisory controller for the necessary action. The supervisory control algorithm computes the cartesian coordinates of both the end-effectors, using the Kinematics of the arm, as follows:

$$\begin{aligned} x &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ z &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{aligned} \quad (6.2)$$

Denoting the coordinates of arm-A as (x_A, z_A) and those of arm-B (x_B, z_B) , the vertical distance 'd' or the tracking error between them is given by:

$$d = [(x_A - x_B)^2 + (z_A - z_B)^2]^{0.5} \quad (6.3)$$

Supervisory control must drive this distance to zero; i.e.

$$\text{as } t \rightarrow \infty, d \rightarrow 0 \quad (6.4)$$

To do this, the first time derivative of d , ' $\dot{d}$ ' is defined:

$$\dot{d}_i = (d_i - d_{i-1}) / (\text{time-step}) \quad (6.5)$$

After many attempts, the following algorithm to modify the controller of arm-B was chosen:

$$\theta_{dB}[j] = \theta_{dA}[j] + \epsilon[j]; \quad j = 1, 2 \quad (6.6)$$

where:

$\theta_{dA}[1]$ and $\theta_{dA}[2]$ denote the desired joint angles for arm-A,

$\theta_{dB}[1]$ and $\theta_{dB}[2]$ those for arm-B,

and $\epsilon[j]$ represents an input compensation term.

Thus when arm-B tracks the trajectory of arm-A exactly,

$\theta_{dB}[j] = \theta_{dA}[j]$, $j = 1, 2$, and $\epsilon[j] = 0$. But when a finite error exists between the two end-effectors, the controlling input to arm-B,

$\theta_{dB}[j]$ is modified by the addition of the term $\epsilon[j]$, to make the error converge to zero. The term $\epsilon[j]$ is computed as follows:

$$\epsilon[1] = \epsilon[2] = G * d * \dot{d} \quad (6.7)$$

Equation (6.7) is based on the popular 'PD' type of control action. The proportional term $(G * d)$ is responsible for driving the error to zero; while the derivative term ' $\dot{d}$ ' makes the law sensitive to the rate of increase of the error. Now, gain 'G' itself is modified to 1.5 times its previous value, after the minimum value of d is reached. This is necessary to account for the rapid increase of d during the acceleration phase of the arms, and the subsequent slow reduction during the decelerating phase. The value of 'G', as well as its modification were set on the basis of experiments.

The plot of ' $\dot{d}$ ' versus time, for this simulation is shown in Fig. 6.5. A further small improvement in the performance is evident from the figure. This time, the maximum error is only 1.8 mm.

Thus, even with an unbalanced payload distribution on the two arms, their tracking performance has been maintained with an error of less than 2 mm, with the use of *Hierarchical Adaptive Control*.

Fig. 6.5 Tracking Error with Hierarchical Adaptive Control

6.3 Hybrid Control of Dual-Arm Manipulators

In this section, the theory of hierarchical adaptive control is extended to a higher degree of freedom dual-arm manipulator. As well, an attempt is made to apply hybrid force/position control to this manipulator. The design of a hybrid force/position controller is complicated by the additional requirements of force control. The controller should be capable of performing position and force control simultaneously, as well achieve effective coordination between the two arms [64 - 67].

The dual-arm manipulator used here is similar to the one described in the previous section, except that each arm has three degrees-of-freedom (similar to R2D3). This is achieved by the addition of a horizontal degree-of-actuation to joint-1 of each arm. This arrangement confers a three-dimensional capability on each arm (Fig. 6.6). However, it is to be noted that the rotation of the two arms about their vertical axes is limited to a total swing of 30° each. In addition, the two arms can be mounted on a rotary base, thus complementing their limited movement in the third dimension. Therefore, any movement of the arms in the third dimension would be achieved by a combination of movements, of the rotary table itself and rotations of the arms about their vertical axes. The major contribution to this movement would however be due to the rotary table.

Fig. 6.6 Single-arm Configuration for 3-Dimensional Dual Manipulator

Hybrid control, or the simultaneous control of forces and position requires careful consideration. This is because position and force controls are strongly inter-related, so that any change in position of the end-effector affects the forces it can apply on the object and vice-versa (changes in applied forces adversely affect the positioning).

Let 'F' and '-F' be the normal forces, that are to be applied on the object by the two arms. It is known from the configuration of the arm that these forces are to be exerted at the end-effector by means of torques applied at the base (joint-1) in the horizontal plane, of each arm.

The value of this torque ' T_0 ' is given by:

$$T_0 = F d \cos(\Phi) \quad (6.8)$$

where;

$$d^2 = l_1^2 + l_2^2 - 2l_1l_2\cos(\pi - \theta_3)$$

is obtained from the geometry of the configuration,

and $\Phi = \theta_1 + 0.5\theta_2$, for $l_1 = l_2$, as in this case.

Since the two arms are identical, the value of T_0 for each arm would be equal in magnitude, but opposite in direction, as long as the arms are tracking each other perfectly. These base-torques applied by the base motors do not cause any motion of the arms in

the horizontal plane, but are merely responsible for applying the normal forces on the object.

The trajectory is specified as before (eqn. 6.1), except that the motion in the third dimension is specified as :

$$\theta_{d3}(i) = \theta_{d1}(i) / 3 \quad (6.9)$$

This time, the payload difference between the two arms is set at 5 Kg. Also, a (3×1) vector 'diff' is defined to represent the angular position differences between the arms:

$$\text{diff} = q_{pA} - q_{pB} \quad (6.10)$$

where, q_{pA} and q_{pB} are the (3×1) position vectors of arms A, and B respectively.

The supervisory law used is:

$$r_B(i+1) = r_B(i) + e(i) \quad (6.11)$$

where:-

r_B is the commanded input vector representing the desired position of arm-B;

i is the iteration number;

$$e(i) = k * \text{diff}[i];$$

k is the differential error gain factor, which depends on the characteristics of the dual manipulator.

The MRAC algorithm yields the torques necessary for the position control of both arms. In order to include force control as well, the base-torques obtained from the force control algorithm need to be superposed on to the torques obtained from the MRAC algorithm. This essentially gives a hybrid force/position control. The motor torques for position and force control of each arm are calculated separately and combined at the end of each iteration. Then the dynamic equations are solved to obtain arm positions, velocities, accelerations and the end-effector forces.

The initial simulation of this dual-arm manipulator includes gravitational forces. This corresponds to a terrestrial environment. Figure 6.7 shows a plot of the tracking error without supervisory control. Each arm is under the exclusive control of its own MRAC controller. A maximum error of 2 mm is obtained. On the introduction of hierarchical control, maximum error is reduced to 0.4 mm (Fig. 6.8). However, the control effort seems to give rise to oscillations in the error-profile, during the better portion of motion of the arms. The differential error gain ' k ' was set equal to 8, by experimentation.

Fig. 6.7 Error Profile without Supervisory Control, in a Gravity Field

Fig. 6.8 Error Profile with Supervisory Control, in a Gravity Field

A third case is studied; in which a weightless environment, such as outer space is considered. In the absence of gravity terms, inertial and coupling forces become predominant. In Fig. 6.9, the maximum error is once again limited to 0.4 mm. But two interesting observations can be made.

Firstly, a very large differential error gain ($K = 800$), is required to keep the error minimal. This is not surprising, considering the fact that the absence of any damping effect, due to gravity, must be compensated for by a stronger control effort. Secondly, the oscillations in the error-profile are of a uniformly small magnitude, superimposed over a curve, very similar to the one of Fig. 6.7. As well, the error is close to zero upto about 1.5 second, corresponding to the point in time halfway in the acceleration phase.

In order now to examine the force control of the hybrid control system, the torques required to be applied by the base-actuators of the two arms are plotted (Fig. 6.10, 6.11, 6.12). In these three figures, the torques for arm-1 remain fairly uniform, while those for arm-2 vary considerably. This is because the motion of the object is caused by the torques exerted by the base-actuators of arm-2; whereas arm-1 merely exerts the necessary reactionary torque, essential to maintain the prescribed normal forces on the object. In Fig. 6.13, base-torques have been plotted for a simulation without force-control. Here, the torques exerted by both arms are

Fig. 6.9 Error Profile with Supervisory Control, in a Weightless Environment

Fig. 6.10 Torque Profiles without Supervisory Control, in a Gravity Field

Fig. 6.11 Torque Profiles with Supervisory Control, in a Gravity Field

6.12 Torque Profiles with Supervisory Control, in a weightless Environment

Fig. 6.13 Torque Profiles without Force Control

identical in nature, though different in magnitude, due to the differing payloads handled by each.

6.4 Summary

To summarize this chapter, hierarchical adaptive control is successfully applied to the problem of trajectory tracking and coordination between the two arms of a dual manipulator. Hybrid force/position control has also been attempted. The controller parameters in weightless and gravitational environments must be different for effective control. The oscillatory nature of the tracking error, in the case of hybrid position/force control needs further investigation.

CHAPTER VII

SUMMARY

7.1 Conclusions

The objective of this thesis was to demonstrate and evaluate the usefulness of Model Reference Adaptive Control techniques in manipulator control, as also to investigate their effectiveness in related robotic applications. Both rigid and flexible robotic manipulators, with several degrees of freedom and possessing different configurations were studied. The feasibility of controlling a dual-arm manipulator with MRAC methods was also investigated. A computer simulation method was used in all the above studies.

In order to mathematically model a manipulator, a description of its dynamics must be known. The derivation of the dynamic equations by the method of the Lagrangian was presented. This method though cumbersome, proved to be quite simple and straight-forward, and was hence used throughout this thesis.

The dynamic equations illustrate the high degree of non-linearity inherent in manipulator systems, and the strong dynamic interaction between the joints. Additionally, the

manipulator dynamics are affected by its configuration and the mass of the payload, which is usually not known. These two factors also change as the manipulator goes through its motions.

An MRAC method was presented to enable positioning control of manipulators. The method was based on the theory of partitioning control, and was capable of compensating for the non-linearities in the dynamic equations and decouple the link interaction. This is evident from the simulation results. Further, the MRAC scheme ensures a manipulator performance, which is insensitive to payload variations; i.e. a consistent performance is guaranteed, despite variations in the payload mass.

The generality of the above method was borne out by simulations of manipulators with higher degrees of freedom, both two and three dimensional. The results indicated a very high degree of reliability of MRAC methods for the positioning control of rigid manipulators.

In order to control a flexible manipulator, the deflections of the links need to be predicted and subsequently compensated for by the control system. The deflection prediction of the manipulator end-effector was achieved by modeling the links as flexible beams. Deflection compensation was attempted by updating the "knowledge" of the end-effector position, that the control system possesses. The resulting modification in the control effort (actuator torque) provided

the necessary compensation. This method resulted in accurate final positioning of the end-effector. But the oscillations produced by a whipping motion of the links, as the manipulator executed its motion could not be completely eliminated. This would require active vibration control. It was observed though that a three-dimensional manipulator was much less subject to this oscillatory behaviour than a two-dimensional one with the same physical configuration. The addition of an orthogonal degree of freedom to a planar manipulator helps to damp out much of the oscillation.

Attempts were also made to achieve positional coordination between the two arms of a dual-arm manipulator, using MRAC. Good results are obtained for both planar and three-dimensional dual-arm manipulators. The control system ensures good tracking coordination of the two arms, despite the differing payloads carried by each. Hybrid force/position control was tested for the three-dimensional dual-arm manipulator. Results indicate the feasibility of using MRAC to achieve hybrid force/position control, as well as coordination of the two arms of a dual-arm manipulator.

7.2 Recommendations

The implementation of MRAC methods for the real-time control of manipulators has proven to be a challenging task. This means that further simplification of MRAC schemes is called for. Also, the

development of newer computer architectures such as parallel processing is required to speed up the computational portion of the control schemes.

Before flexible manipulators can become more practicable, it must be possible to control them completely. This implies that the control system should be able to overcome the problems of oscillation and vibration by some form of active vibration control. Such a system could include the dynamic modification of the local stiffnesses of different parts of the manipulator structure.

An Expert System could be developed to replace or augment the capabilities of a hierarchical adaptive control structure. Such a system would enhance the dexterity and manoeuvrability of dual arm manipulators. Another area of interest is the cooperation among multiple-arm manipulators.

APPENDIX 1

LAGRANGIAN DYNAMICS FOR R2D3

Refer to Fig. 4.8 for the physical configuration of the manipulator R2D3.

The Lagrangian 'L' of a system is defined as:

$$L = K - P \quad (A1.1)$$

where:

K is the total kinetic energy of the system,
and P is the total potential energy of the system.

Defining the individual kinetic and potential energies of each link as K_1 , K_2 , P_1 and P_2 ;

$$K = K_1 + K_2 \quad \text{and} \quad P = P_1 + P_2 \quad (A1.2)$$

Let (x_1, y_1, z_1) be the coordinates of Joint-2. Then,

$$x_1 = l_1 C_1 C_2$$

$$y_1 = l_1 S_1 C_2$$

$$z_1 = l_1 S_2$$

(A1.3)

Differentiating eqn.(A1.3) with respect to time, we have

$$\begin{aligned}\dot{x}_1 &= -l_1[C_1\dot{S}_2\dot{\theta}_2 + C_2S_1\dot{\theta}_1] \\ \dot{y}_1 &= l_1[C_1C_2\dot{\theta}_1 - S_1S_2\dot{\theta}_2] \\ \dot{z}_1 &= l_1C_2\dot{\theta}_2\end{aligned}\quad (A1.4)$$

Squaring and adding eqn. (A1.4)

$$\begin{aligned}v_1^2 &= \dot{x}_1^2 + \dot{y}_1^2 + \dot{z}_1^2 \\ &= l_1^2[\dot{\theta}_2^2 + C_2^2\dot{\theta}_1^2]\end{aligned}\quad (A1.5)$$

The kinetic energy of link-1 K_1 then becomes

$$\begin{aligned}K_1 &= 0.5m_1v_1^2 \\ &= 0.5m_1l_1^2[\dot{\theta}_2^2 + C_2^2\dot{\theta}_1^2]\end{aligned}\quad (A1.6)$$

and its potential energy P_1

$$P_1 = m_1gz_1 = m_1gl_1S_2\quad (A1.7)$$

Similarly, let (x_2, y_2, z_2) be the coordinates of the end-effector.

Then,

$$x_2 = [l_1 C_2 + l_2 C_{23}] C_1$$

$$y_2 = [l_1 C_2 + l_2 C_{23}] S_1 \quad (A1.8)$$

$$z_2 = l_1 S_2 + l_2 S_{23}$$

The time derivatives of eqn. (A1.8) are

$$\dot{x}_2 = \dot{x}_1 - l_2 C_{23} S_1 \dot{\theta}_1 - l_2 C_1 S_{23} \dot{\theta}_{23}$$

$$\dot{y}_2 = \dot{y}_1 + l_2 C_{23} C_1 \dot{\theta}_1 - l_2 S_1 S_{23} \dot{\theta}_{23} \quad (A1.9)$$

$$\dot{z}_2 = \dot{z}_1 + l_2 C_{23} \dot{\theta}_{23}$$

and

$$v_2^2 = v_1^2 + l_2^2 [\dot{\theta}_{23}^2 + C_{23}^2 \dot{\theta}_1^2] \quad (A1.10)$$

The kinetic energy of link-2 is

$$K_2 = 0.5 m_2 v_2^2 \quad (A1.11)$$

and its potential energy is

$$P_2 = m_2 g z_2 \quad (A1.12)$$

From eqns. (A1.1, A1.2, A1.6, A1.7, A1.11 and A1.12), the Lagrangian 'L' of the system is obtained, which is required to calculate the torques as:

$$\tau_i = \frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} \quad (A1.13)$$

From eqn. (A1.13), the joint torques are obtained as follows:

$$\begin{aligned} \tau_1 = & \ddot{\theta}_1 [m_{12} l_1^2 C_2^2 + m_2 l_2^2 C_{23}^2 + 2m_2 l_1 l_2 C_2 C_{23}] \\ & - 2\dot{\theta}_1 \dot{\theta}_2 [m_{12} l_1^2 C_2 S_2 + m_2 l_2^2 C_{23} S_{23} + m_2 l_1 l_2 \sin(2\theta_2 + \theta_3)] \\ & - 2\dot{\theta}_1 \dot{\theta}_3 [m_2 l_2^2 C_{23} S_{23} + m_2 l_1 l_2 C_2 S_{23}] \end{aligned} \quad (A1.14)$$

$$\begin{aligned} \tau_2 = & \ddot{\theta}_2 [m_{12} l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 C_3] \\ & + \ddot{\theta}_3 [m_2 l_2^2 + m_2 l_1 l_2 C_3] \\ & + \dot{\theta}_1^2 [m_{12} l_1^2 C_2 S_2 + m_2 l_2^2 C_{23} S_{23} + m_2 l_1 l_2 \sin(2\theta_2 + \theta_3)] \\ & - \dot{\theta}_3^2 [m_2 l_1 l_2 S_3] \\ & - 2\dot{\theta}_2 \dot{\theta}_3 [m_2 l_1 l_2 S_3] \\ & + g [m_{12} l_1 C_2 + m_2 l_2 C_{23}] \end{aligned} \quad (A1.15)$$

$$\begin{aligned} \tau_3 = & \ddot{\theta}_2 [m_2 l_2^2 + m_2 l_1 l_2 C_3] \\ & + \ddot{\theta}_3 [m_2 l_2^2] \\ & + \dot{\theta}_1^2 [m_2 l_2^2 C_{23} S_{23} + m_2 l_1 l_2 C_2 S_{23}] \\ & + \dot{\theta}_2^2 [m_2 l_1 l_2 S_3] \\ & + g [m_2 l_2 C_{23}] \end{aligned} \quad (A1.16)$$

where, the same notation has been used as in Chapter 2.

APPENDIX 2

LAGRANGIAN DYNAMICS FOR R3D3

Refer to Fig. 4.10 for the physical configuration of the manipulator R3D3.

The Lagrangian 'L' of a system is defined as:

$$L = K - P \quad (A2.1)$$

where:

K is the total kinetic energy of the system,
and P is the total potential energy of the system.

Defining the individual kinetic and potential energies of each link as K_1, K_2, K_3 , and P_1, P_2, P_3 respectively;

$$K = K_1 + K_2 + K_3 \quad \text{and} \quad P = P_1 + P_2 + P_3 \quad (A2.2)$$

Let (x_1, z_1) be the coordinates of Joint-2. Then, the kinetic and potential energies of link-1 are:

$$K_1 = 0.5m_1 \bar{l}_1^2 \dot{\theta}_1^2 \quad (A2.3)$$

$$P_1 = m_1 g l_1 S_1 \quad (A2.4)$$

Let (x_2, z_2) be the coordinates of Joint-3. Then,

$$x_2 = l_1 C_1 + l_2 C_{12}$$

and

(A2.5)

$$z_2 = l_1 S_1 + l_2 S_{12}$$

Differentiating eqn.(A2.5) with respect to time, we have

$$\dot{x}_2 = -l_1 S_1 \dot{\theta}_1 - l_2 S_{12} \dot{\theta}_{12}$$

and

(A2.6)

$$\dot{z}_2 = l_1 C_1 \dot{\theta}_1 + l_2 C_{12} \dot{\theta}_{12}$$

Squaring and adding eqn. (A2.6)

$$\begin{aligned} v_2^2 &= \dot{x}_2^2 + \dot{z}_2^2 \\ &= l_1^2 \dot{\theta}_1^2 + l_2^2 \dot{\theta}_{12}^2 + 2l_1 l_2 \dot{\theta}_1 \dot{\theta}_{12} C_2 \end{aligned} \quad (A2.7)$$

The kinetic energy of link-2 K_2 then becomes

$$\begin{aligned} K_2 &= 0.5m_2 v_2^2 \\ &= 0.5m_2 l_1^2 \dot{\theta}_1^2 + 0.5m_2 l_2^2 \dot{\theta}_{12}^2 + m_2 l_1 l_2 C_2 \dot{\theta}_1 \dot{\theta}_{12} \end{aligned} \quad (A2.8)$$

and its potential energy P_2

$$P_2 = m_2 g z_2 = m_2 g [l_1 S_1 + l_2 S_{12}] \quad (A2.9)$$

Similarly, let (x_3, z_3) be the coordinates of the end-effector. Then,

$$\begin{aligned} x_3 &= x_2 + l_3 C_{123} \\ z_3 &= z_2 + l_3 S_{123} \end{aligned} \quad (A2.10)$$

The time derivatives of eqn. (A2.10) are

$$\begin{aligned} \dot{x}_3 &= \dot{x}_2 - l_3 S_{123} \dot{\theta}_{123} \\ \dot{z}_3 &= \dot{z}_2 + l_3 C_{123} \dot{\theta}_{123} \end{aligned} \quad (A2.11)$$

and

$$v_3^2 = \dot{x}_3^2 + \dot{z}_3^2 \quad (A2.12)$$

The kinetic energy of link-3 is

$$K_3 = 0.5 m_3 v_3^2 \quad (A2.13)$$

and its potential energy is

$$P_3 = m_3 g z_3 \quad (A2.14)$$

From eqns. (A2.1, A2.2, A2.3, A2.4, A2.8, A2.9, A2.13 and A2.14), the Lagrangian 'L' of the system is obtained, which is required to calculate the torques as:

$$\tau_i = \frac{d}{dt} \cdot \frac{\partial L}{\partial \dot{\theta}_i} - \frac{\partial L}{\partial \theta_i} \quad (A2.15)$$

From eqn. (A2.15), the joint torques are obtained as follows:

$$\begin{aligned} \tau_1 = & \ddot{\theta}_1 [m_{123} l_1^2 + m_{23} l_2^2 + m_3 l_3^2 + 2m_{23} l_1 l_2 C_2 + \\ & 2m_3 l_1 l_3 C_{23} + 2m_3 l_2 l_3 C_3] \\ & + \ddot{\theta}_2 [m_{23} l_2^2 + m_3 l_3^2 + m_{23} l_1 l_2 C_2 + m_3 l_1 l_3 C_{23} + \\ & 2m_3 l_2 l_3 C_3] \\ & + \ddot{\theta}_3 [m_3 l_3^2 + m_3 l_1 l_3 C_{23} + m_3 l_2 l_3 C_3] \\ & - 2\dot{\theta}_1 \dot{\theta}_2 [m_{23} l_1 l_2 S_2 + m_3 l_1 l_3 S_{23}] \\ & - 2\dot{\theta}_1 \dot{\theta}_3 [m_3 l_2 l_3 S_3 + m_3 l_1 l_3 S_{23}] \\ & - 2\dot{\theta}_2 \dot{\theta}_3 [m_3 l_2 l_3 S_3 + m_3 l_1 l_3 S_{23}] \\ & - \dot{\theta}_2^2 [m_{23} l_1 l_2 S_2 + m_3 l_1 l_3 S_{23}] \\ & - \dot{\theta}_3^2 [m_3 l_1 l_3 S_{23} + m_3 l_2 l_3 S_3] \\ & + g [m_{123} l_1 C_1 + m_{23} l_2 C_{12} + m_3 l_3 C_{123}] \end{aligned} \quad (A2.16)$$

$$\tau_2 = \ddot{\theta}_1 [m_{23} l_2^2 + m_3 l_3^2 + m_{23} l_1 l_2 C_2 + m_3 l_1 l_3 C_{23} +$$

$$\begin{aligned}
& 2m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_2 [m_{23} l_2^2 + m_3 l_3^2 + 2m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_3 [m_3 l_3^2 + m_3 l_2 l_3 C_3] \\
& - 2\dot{\theta}_1 \dot{\theta}_3 [m_3 l_2 l_3 S_3] \\
& - 2\dot{\theta}_2 \dot{\theta}_3 [m_3 l_2 l_3 S_3] \\
& + \dot{\theta}_1^2 [m_{23} l_1 l_2 S_2 + m_3 l_1 l_3 S_{23}] \\
& - \dot{\theta}_3^2 [m_3 l_2 l_3 S_3] \\
& + g [m_{23} l_2 C_{12} + m_3 l_3 C_{123}]
\end{aligned} \tag{A2.17}$$

$$\begin{aligned}
\tau_3 = & \ddot{\theta}_1 [m_3 l_3^2 + m_3 l_1 l_3 C_{23} + m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_2 [m_3 l_3^2 + m_3 l_2 l_3 C_3] \\
& + \ddot{\theta}_3 [m_3 l_3^2] \\
& + 2\dot{\theta}_1 \dot{\theta}_2 [m_3 l_2 l_3 S_3] \\
& + \dot{\theta}_1^2 [m_3 l_2 l_3 S_3 + m_3 l_1 l_3 S_{23}] \\
& + \dot{\theta}_2^2 [m_3 l_2 l_3 S_3] \\
& + g [m_3 l_3 C_{123}]
\end{aligned} \tag{A2.18}$$

For Page Sequence Only

APPENDIX 3

SIMULATION PROGRAM FOR R2D2 MANIPULATOR

/*#####

Model Reference Adaptive Control for a DOUBLE-LINK RIGID MANIPULATOR.

STEP RESPONSE

#####

INPUT - link lengths and masses;

cartesian coordinates of the final END-EFFECTOR position.

INITIAL - determine optimum theta-1, theta-2 < CART2POL > ;

PROCESS

LOOP - apply the MRAC algorithm;

OUTPUT - required angular position of the links;

iterated values of: POSITION;

VELOCITY;

ACCELERATION;

final coordinates of the end-effector and error.

****/

```
#include <stdio.h>
#include <math.h>
#define sq(n) n*n
#define cub(m) m*m*m
#define pow4(o) o*o*o*o
FILE *fptr, *fopen();
```

```
/* CART2POL : CARTESIAN TO POLAR CONVERSION ROUTINE
```

```
*****
```

```
cart2pol (l1, l2, refx2, refy2, rp)
```

```
float l1, l2, refx2, refy2, *rp;
```

```
{
```

```
    int ij, j, k, l;
```

```
    float ct2, theta2, a, b, m, n, ct1, theta1, refx1, refy1, pi, pi2, dx1, dy1,
```

```
          delta1, delta2, x2, y2, alpha1, nx1, ny1, phi, x2, y2, nx2, ny2, dx2,
```

```
          dy2, alpha2, dev1, dev2;
```

```
    ij = 1;
```

```
    j = 1;
```

```
    k = 1;
```

```
    l = 1;
```

```
    pi = 3.1415926;
```

```
    pi2 = pi * 2.0;
```

```
    ct2 = ((sq(refx2) + sq(refy2)) - (sq(l1) + sq(l2))) / (2.0 * l1 * l2);
```

```
    theta2 = acos(ct2);
```

```
    if (refy2 < 0.0)
```

```
        ij = -1;
```

```
    theta2 += ij;
```

```
    a = l1 + (l2 * cos(theta2));
```

```
    b = l2 * sin(theta2);
```

```
    m = refx2;
```

```
    n = refy2;
```

```
    ct1 = ((b * n) + (a * m)) / (sq(a) + sq(b));
```

```
    theta1 = acos(ct1);
```

```
    theta1 += ij;
```

```

    refx1 = (l1 * cos(theta1));
    refy1 = (l1 * sin(theta1));
    *rp = theta1;
    *(rp + 1) = theta2;
}

```

```

/* SUB-ROUTINES FOR THE MRAC STRATEGY.

```

```

*****/

```

```

crossmul(mat, vec, x)

```

```

float mat[2][2], vec[2], *x;

```

```

{
    *x = mat[0][0] * vec[0] + mat[0][1] * vec[1];
    *(x + 1) = mat[1][0] * vec[0] + mat[1][1] * vec[1];
}

```

```

/*

```

```

    END OF SUB-ROUTINES

```

```

    -----

```

```

    */

```

```

/*

```

```

    THE MAIN PROGRAM

```

```

*****/

```

```

main()

```

```

{

```

```

    /*

```

```

        DECLARATIONS

```

```

        -----

```

```

    */

```

```

    float x[2], temp[2], tempi[2], tempv[2], modelacc[2], accel[2],

```

```

        manipint[2], modelint[2];

```

```

    float mmod[2][2], nmod[2][2], manip[2][2], nmanip[2][2], minv[2][2],

```

```

        adj[2][2], kn[2][2], kn[2][2], fp[2][2], fv[2][2], cp[2][2],

```

```

        cv[2][2], gp[2][2], gi[2][2], gd[2][2], delnmod[2][2],

```

```

        delnmod[2][2];

```



```

float xmp[2],xmv[2],u[2],xp[2],xv[2],exp[2],erv[2],y[2],rp[2],
      mass[2],l[2],condp[2],condv[2],interu[2],clxp[2],clxv[2],
      clxmp[2],clxmv[2];
float pi,g,H,det,refx2,refy2,error,x2,y2,meter;
int i,j,k,nn,divide10,flag,cqunter;

```

/*

INITIALIZATIONS

*/

```

fptr = fopen("r2d2.in","r");
g = 9.81;
pi = 3.1415926;
fscanf(fptr, "%f %f ", &mass[0], &mass[1]);
printf("\nMASS OF FIRST LINK = %f Kg\nMASS OF SECOND LINK = %f
      Kg\n", mass[0],mass[1]);
fscanf(fptr, "%f ", &h);
printf("\nTIME STEP FOR THE INTEGRATION ROUTINES = %f second\n",
      h);
fscanf(fptr, "%d ", &nn);
nn = 10000;
printf("\nTOTAL NUMBER OF ITERATIONS: %d \n", nn);
fscanf(fptr, "%f %f ", &l[0], &l[1]);
printf("\nLENGTH OF FIRST LINK = %f m\nLENGTH OF SECOND LINK = %f
      m\n", l[0],l[1]);
fscanf(fptr, "%f %f %f ", &gp[0][0], &gi[0][0], &gd[0][0]);
printf("\nPID CONTROLLER GAINS ARE: P = %f I = %f D = %f\n",
      gp[0][0],gi[0][0],gd[0][0]);
fscanf(fptr, "%f %f ", &cp[0][0], &cv[0][0]);
printf("\nCP = %f CV = %f \n", cp[0][0], cv[0][0]);
fscanf(fptr, "%f %f %f ", &km[0][0], &km[0][1], &km[1][1]);
printf("\nKM[1][1] = %f KM[1][2] = %f KM[2][2] = %f \n",
      km[0][0],km[0][1],km[1][1]);
fscanf(fptr, "%f ", &kn[0][1]);
printf("\nKN = %f \n", kn[0][1]);
kn[0][0] = 0;
kn[1][0] = 0;

```

```

km[1][1] = km[0][1];

for (j = 0; j < 2; j++) {
    for (k = 0; k < 2; k++) {
        fv[j][k] = 0;
        fp[j][k] = fv[j][k];
    }
}

```

```

cv[0][1] = 0;
cv[1][0] = 0;
cv[1][1] = cv[0][0];
cp[0][1] = 0;
cp[1][0] = 0;
cp[1][1] = cp[0][0];

```

```

i = 1;
flag = 0;
counter = 0;
divide10 = 1;
gp[0][1] = 0;
gp[1][0] = 0;
gp[1][1] = gp[0][0];
gi[0][1] = 0;
gi[1][0] = 0;
gi[1][1] = gi[0][0];
gd[0][1] = 0;
gd[1][0] = 0;
gd[1][1] = gd[0][0];

```

/*

CARTESIAN TO POLAR CONVERSION

*/

```

for (j = 0; j < 2; j++) {
    xp[j] = 0;
    xv[j] = 0;
}

```

```

    xmp[j] = xp[j];
    xmv[j] = xv[j];
    interu[j] = 0;
    accel[j] = 0;
}

```

```

meter = 0;
while (i <= nn) {
    rp[0] = 0.5;
    rp[1] = 1.0;

```

``` /* MRAC ALGORITHM ----- */ ```

```

/* DETERMINE ERROR BETWEEN COMMANDED POSITION AND ACTUAL POSITION */

```

```

for(j = 0; j < 2; j++) {
    clxp[j] = rp[j] - xp[j];
    manipint[j] += clxp[j] * h;
    clxmp[j] = rp[j] - xmp[j];
    modelint[j] += clxmp[j] * h;
}

```

```

/*      APPLY THE PID CONTROLLER TO THE MANIPULATOR      */

```

```

crossmul(gi, manipint, tempi);
crossmul(gp, clxp, tempv);
crossmul(gd, xv, tempv);

```

```

/*FIND ERRORS IN POSITION AND VELOCITY BETWEEN MODEL AND MANIPULATOR*/

```

```

for (j = 0; j < 2; j++) {
    u[j] = tempv[j] + tempi[j] - tempv[j];
    exp[j] = xmp[j] - xp[j];
    erv[j] = xmv[j] - xv[j];
}

```

}

/* APPLY ERROR VECTOR GAINS */

```

crossmul(cp,exp,condp);
crossmul(cv,erv,condv);
for(j = 0; j < 2; j++) {
    y[j] = condv[j] + condp[j];
}

```

/* PROVIDE AN INITIAL GUESS FOR THE ESTIMATED INERTIA AND

COUPLING MATRICES */

```

if (i == 1) {
    nmod[0][0] = 1.0;
    nmod[0][1] = 0;
    nmod[1][0] = 0;
    nmod[1][1] = 1.0;
    for (j = 0; j < 2; j++) {
        for (k = 0; k < 2; k++)
            nmod[j][k] = 0;
    }
}

```

/* THE ADAPTATION ALGORITHM FOR ESTIMATED INERTIA AND COUPLING

MATRICES */

```

if (i > 1) {
    delnmod[0][0] = km[0][0] * y[0] * u[0];
    delnmod[0][1] = km[0][1] * (y[0] * u[1] + y[1] * u[0]);
    delnmod[1][0] = delnmod[0][1];
    delnmod[1][1] = km[1][1] * y[1] * u[1];
    delnmod[0][1] = km[0][1] * ((2 * y[0] * xv[0] + xv[1]) -
        y[1] * sq(xv[0]));
    delnmod[1][1] = km[1][1] * y[0] * sq(xv[1]);
    for (j = 0; j < 2; j++) {

```

```

    }
    for (k = 0; k < 2; k++) {
        nmod[j][k] += delnmod[j][k] * h;
        nmod[j][k] += delnmod[j][k] * h;
    }
}

```

/* DUE TO THE ABSENCE OF AN EXPERIMENTAL ROBOT, THE ROBOT MOTION AND DYNAMICS HAVE TO BE SIMULATED THEREFORE, NEED INERTIA AND COUPLING MATRICES OF THE ROBOT ITSELF */

```

mmanip[0][0] = (mass[0] + mass[1]) * sq(l[0]) + (mass[1] * sq(l[1]))
              + (2.0 * mass[1] * l[0] * l[1] * cos(xp[1]));
mmanip[1][1] = mass[1] * sq(l[1]);
mmanip[0][1] = mmanip[1][1] + mass[1] * l[0] * l[1] * cos(xp[1]);
mmanip[1][0] = mmanip[0][1];
fp[0][1] = -1.0 * mass[1] * l[0] * l[1] * sin(xp[1]);
fp[0][0] = 2.0 * fp[0][1];
fp[1][0] = -1.0 * fp[0][1];
fv[0][0] = (mass[0] + mass[1]) * g * l[0] * cos(xp[0]);
fv[1][1] = mass[1] * g * l[1] * cos(xp[0] + xp[1]);
fv[0][1] = fv[1][1];
mmanip[0][1] = fp[0][0] * xv[0] + xv[1] + fp[0][1] * sq(xv[1])
              + fv[0][0] + fv[0][1];
mmanip[1][1] = fp[1][0] * sq(xv[0]) + fv[1][1];

```

/* FIND THE INVERSE OF THE INERTIA MATRIX OF THE MANIPULATOR */

```

det = (mmanip[0][0] * mmanip[1][1]) - (mmanip[0][1] * mmanip[1][0]);
adj[0][0] = mmanip[1][1];
adj[1][1] = mmanip[0][0];
adj[0][1] = -mmanip[0][1];
adj[1][0] = -mmanip[1][0];
for (j = 0; j < 2; j++) {
    for (k = 0; k < 2; k++)

```

```

        minv[j][k] = adj[j][k]/det;
    }

/*      OBTAIN THE ACCELERATION VECTOR OF THE MANIPULATOR      */

    crossmul(mmod,u,x);
    for (j = 0; j < 2; j++)
        interu[j] = x[j] + mmod[j][1] - mmanip[j][1];
    crossmul(minv,interu,accel);

/* INTEGRATE TWICE TO OBTAIN THE VELOCITY AND ACCELERATION OF THE
                                           MANIPULATOR */

    for (j = 0; j < 2; j++) {
        xv[j] += accel[j] * h;
        xp[j] += xv[j] * h;
    }

/* APPLY THE PID CONTROLLER TO THE MODEL TO OBTAIN ITS ACCELERATION */

    crossmul(g1,modelint,temp1);
    crossmul(gp,clxmp,temp2);
    crossmul(gd,xmv,tempv);

/* INTEGRATE TWICE TO OBTAIN THE VELOCITY AND ACCELERATION OF THE
MODEL */

    for (j = 0; j < 2; j++) {
        modelacc[j] = temp2[j] - tempv[j];
        xmv[j] += modelacc[j] * h;
        xmp[j] += xmv[j] * h;
    }
    x2 = l[0]*cos(xp[0]) + l[1]*cos(xp[0]+xp[1]);
    y2 = l[0]*sin(xp[0]) + l[1]*sin(xp[0]+xp[1]);

```

OUTPUT

*/

```

    if (i <= 100) {
        if (divide10 == 10)
            counter = 1;
    }
    if (i > 100) {
        if (divide10 == 100)
            counter = 1;
    }
    if (counter == 1) {
printf("ITERATION NO.           %d \n",i);
printf("POSITION (rad):         %f      %f\n", xp[0], xp[1]);
printf("VELOCITY (rad/s):         %f      %f\n", xv[0], xv[1]);
printf("ACCELERATION (rad/sq.s):  %f      %f\n", accel[0],
            accel[1]);
printf("*****\n");
        divide10 = 0;
        counter = 0;
    }
    i += 1;
    divide10 += 1;
}

```

```
/* DETERMINE AND PRINT THE ERROR IN FINAL END-EFFECTOR POSITIONING */
```

```

x2 = l[0] * cos(xp[0]) + l[1] * cos(xp[0] + xp[1]);
y2 = l[0] * sin(xp[0]) + l[1] * sin(xp[0] + xp[1]);
error = hypot((refx2 - x2), (refy2 - y2));
/*   printf("ERROR = %f mm \n", 1000.0 * error); */
}

```

```

/*                                     */
/*****

```

APPENDIX 4

SIMULATION PROGRAM FOR F2D2 MANIPULATOR

```
/*%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
Model Reference Adaptive Control for a DOUBLE-LINK FLEXIBLEMANIPULATOR  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
INPUT   - link lengths and masses;  
          cartesian coordinates of the final END-EFFECTOR position.  
INITIAL - determines optimum theta-1, theta-2 < CART2POL > ;  
PROCESS  
  LOOP   - apply the MRAC algorithm;  
  OUTPUT - required angular position of the links;  
          iterated values of : POSITION;  
                               VELOCITY;  
                               ACCELERATION;  
                               TORQUE;  
          final coordinates of the end-effector and error.  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%/  
  
#include <stdio.h>
```



```

#include <math.h>

#define sq(n) n*n
#define cub(m) m*m*m
#define pow4(o) o*o*o*o
FILE *fptr, *fopen();

/* CART2POL : CARTESIAN TO POLAR CONVERSION ROUTINE
*****/

cart2pol (l1, l2, refx2, refy2, rp)
float l1, l2, refx2, refy2, *rp;
{
    int ij, j, k, l;
    float ct2, theta2, a, b, m, n, ct1, theta1, refx1, refy1, pi, pi2, dx1, dy1,
        delta1, delta2, x1, y1, alpha1, nx1, ny1, phi, x2, y2, nx2, ny2, dx2,
        dy2, alpha2, dev1, dev2;

    ij = 1;
    j = 1;
    k = 1;
    l = 1;
    pi = 3.1415926;
    pi2 = pi * 2.0;
    ct2 = ((sq(refx2) + sq(refy2)) - (sq(l1) + sq(l2)))/(2.0*l1*l2);
    theta2 = acos(ct2);
    if (refy2 < 0.0)
        ij = -1;
    theta2 += ij;
    a = l1 + (l2 * cos(theta2));
    b = l2 * sin(theta2);
    m = refx2;
    n = refy2;
    ct1 = ((b * n) + (a * m)) / (sq(a) + sq(b));
    theta1 = acos(ct1);
    theta1 += ij;
    refx1 = (l1 * cos(theta1));
    refy1 = (l1 * sin(theta1));
}

```

```

    *rp = theta1;
    *(rp + 1) = theta2;
}

/* ELASTIX : THE ROUTINE THAT USES THE DYNAMIC EQUATIONS (ref. PAUL)
   TO DETERMINE THE BENDING OF THE LINKS.
   *****/

elastix(m,d,l,e,theta,omega,alpha,s,delta,flag)
int flag;
float m[2],d[2],l[2],e[2],theta[2],omega[2],alpha[2],*s,*delta;
{
    int j;
    double pi,g,slope[2],del[2],i[2],ip[2],p[2],r[2],mc[2];
    pi = 3.1415926;
    g = 9.81;
    for (j=0; j<2; j++) {
        r[j] = d[j] / 2.0;
        i[j] = pi * pow4(r[j]) / 4.0;
        ip[j] = m[j] * (sq(r[j]) / 4.0 + sq(l[j]) / 3.0);
    }
    if (flag == 1) {
        for (j=0; j<2; j++) {
            omega[j] = 0;
            alpha[j] = 0;
        }
    }
    mc[0] = (((m[0] + m[1]) * sq(l[0])) + (m[1] * sq(l[1])) +
        (2.0 * m[1] * l[0] * l[1] * cos(theta[1]))) * alpha[0] +
        (((m[1] * sq(l[1])) + (m[1] * l[0] * l[1] * cos(theta[1])))
        * alpha[1] - (2.0 * m[1] * l[0] * l[1] * sin(theta[1]) *
        omega[0] + omega[1]) - (m[1] * l[0] * l[1] * sin(theta[1])
        * sq(omega[1])) + (m[0] + m[1]) * g * l[0] * cos(theta[0])
        + (m[1] * g * l[1] * cos(theta[0] + theta[1])));
    mc[1] = ((m[1] * sq(l[1])) + (m[1] * l[0] * l[1] * cos(theta[1])))

```

```

        * alpha[0] + (m[1] * sq(l[1]) * alpha[1]) + (m[1] * l[0]
        * l[1] * sin(theta[1]) * sq(omega[0]))
        + (m[1] * g * l[1] * cos(theta[0] + theta[1]));
    for (j=0; j<2; j++) {
        del[j] = ((mc[j] * sq(l[j])) / (2.0 * e[j] * i[j]));
        slope[j] = ((mc[j] * l[j]) / (e[j] * i[j]));
    }
    *delta = del[0];
    *(delta + 1) = del[1];
    *s = slope[0];
    *(s + 1) = slope[1];
}

```

```

/* FLEX : THE COMPENSATION ROUTINE THAT OVERCOMES THE LINK DEFLECTIONS

```

```

*****

```

```

flex(s,delta,xp,l,tt,rp,f,rx2,ry2,)

```

```

int f;

```

```

float s[2],delta[2],xp[2],l[2],tt[2],rx2,ry2,*rp;

```

```

{

```

```

    float pi,pi2,dx1,dy1,x1,y1,refx1,refy1,alpha[2],x2,y2,gamma[2],

```

```

        dx2,dy2,refx2,refy2,error;

```

```

    pi = 3.1415926;

```

```

    pi2 = pi * 2.0;

```

```

    alpha[0] = delta[0]/l[0];

```

```

    alpha[1] = delta[1]/l[1];

```

```

    *rp = tt[0] + alpha[0];

```

```

    *(rp + 1) = tt[1] + alpha[1] + s[0];

```

```

}

```

```

/* SUB-ROUTINES FOR THE NRAC STRATEGY.

```

```

*****

```

```

crossmul(mat, vec, x)

```

```

float mat[2][2],vec[2],*x;

```

```

{

```

```

*x = mat[0][0] * vec[0] + mat[0][1] * vec[1];
*(x + 1) = mat[1][0] * vec[0] + mat[1][1] * vec[1];
}

```

```

/*                      END OF SUB-ROUTINES
-----*/

```

```

/*                      THE MAIN PROGRAM
-----*/

```

```
main()
```

```

{
/*                      DECLARATIONS
-----*/

```

```

float x[2], temp[2], tempi[2], tempv[2], accel[2], manipint[2],
      modelint[2];
float mmod[2][2], nmod[2][2], manip[2][2], nmanip[2][2], minv[2][2],
      adj[2][2], km[2][2], kn[2][2], fp[2][2], fv[2][2], cp[2][2],
      ,cv[2][2], gp[2][2], gi[2][2], gd[2][2], delmmod[2][2],
      delnmod[2][2], ce[2][2];
float xmp[2], xmv[2], u[2], xp[2], xv[2], exp[2], erv[2], err[2], y[2],
      rp[2], mass[2], l[2], jerk[2], facc[2], bacc[2], tart[2],
      dia[2], ymodofe[2], s[2], delta[2], phi[2], modelacc[2],
      condp[2], condv[2], conde[2], interu[2], clxp[2], clxv[2],
      clxmp[2], clxmv[2];
float meter, g, h, det, refx2, refy2, error, x2, y2, pi, omega;
int i, j, k, nn, divide10, flag, counter;

```

```

/*                      INITIALIZATIONS
-----*/

```

```
g = 9.81;
```

```

pi = 3.1415926;
fptr = fopen("f2d2.in","r");
fscanf(fptr, " %f %f ", &mass[0], &mass[1]);
printf("\nMASS OF FIRST LINK = %f Kg\nMASS OF SECOND LINK = %f
      Kg\n",mass[0],mass[1]);
fscanf(fptr, " %f ", &h);
printf("\nTIME STEP FOR THE INTEGRATION ROUTINES = %fsecond\n",h);
fscanf(fptr, " %d ", &nn);
printf("\nTOTAL NUMBER OF ITERATIONS: %d \n", nn);
fscanf(fptr, " %f %f ", &l[0], &l[1]);
printf("\nLENGTH OF FIRST LINK = %f m\nLENGTH OF SECOND LINK
      = %f m\n",l[0],l[1]);
fscanf(fptr, " %f %f %f ", &gp[0][0], &gi[0][0], &gd[0][0]);
printf("\nPID CONTROLLER GAINS ARE: P = %f I = %f D = %f\n",
      gp[0][0],gi[0][0],gd[0][0]);
fscanf(fptr, " %f ", &ymodofe[0]);
printf("\nYOUNG'S MODULUS OF ELASTICITY: %f \n", ymodofe[0]);
fscanf(fptr, " %f %f ", &dia[0], &dia[1]);
printf("\nDIAMETER OF FIRST LINK = %f cm\nDIAMETER OF SECOND LINK
      = %f cm\n",
      100.0 * dia[0], 100.0 * dia[1]);
fscanf(fptr, " %f %f ", &cp[0][0], &cv[0][0]);
printf("\nCP = %f CV = %f \n", cp[0][0], cv[0][0]);
fscanf(fptr, " %f %f %f ", &kcm[0][0], &kcm[0][1], &kcm[1][1]);
printf("\nKN[1][1] = %f KN[1][2] = %f KN[2][2] = %f \n",
      kcm[0][0],kcm[0][1],kcm[1][1]);
fscanf(fptr, " %f ", &kcn[0][1]);
printf("\nKN = %f \n", kcn[0][1]);
kcn[0][0] = 0;
kcn[1][0] = 0;
kcn[1][1] = kcn[0][1];
for (j = 0; j < 2; j++) {
    cnde[j] = 0;
    for (k = 0; k < 2; k++)
        fp[j][k] = 0;
}

```

```

        fv[j][k] = 0;
        ce[j][k] = 0;
    }
    cv[0][1] = 0;
    cv[1][0] = 0;
    cv[1][1] = cv[0][0];
    cp[0][1] = 0;
    cp[1][0] = 0;
    cp[1][1] = cp[0][0];
    ymodofe[1] = ymodofe[0];
    i = 1;
    flag = 0;
    counter = 0;
    divide10 = 1;
    gp[0][1] = 0;
    gp[1][0] = 0;
    gp[1][1] = gp[0][0];
    gi[0][1] = 0;
    gi[1][0] = 0;
    gi[1][1] = gi[0][0];
    gd[0][1] = 0;
    gd[1][0] = 0;
    gd[1][1] = gd[0][0];

```

/*

CARTESIAN TO POLAR CONVERSION

*/

```

for (j = 0; j < 2; j++) {
    xp[j] = 0;
    xv[j] = 0;
    xmp[j] = xp[j];
    xmv[j] = xv[j];
    interu[j] = 0;
    accel[j] = 0;
    modelacc[j] = 0;
}

```

```

    tart[j] = rp[j];
    jerk[j] = 0;
    facc[j] = 0;
    bacc[j] = 0;
    omega = 0;
}

meter = 0;
while (i <= nn) {
    meter += pi/5000;
    if (meter > pi) {
        meter = pi;
        flag = 1;
    }
    rp[0] = 1.5 * (1 - cos(meter));
    rp[1] = 0.75 * (1 - cos(meter));
    for (j = 0; j < 2; j++) {
        tart[j] = rp[j];
        jerk[j] = (accel[j] - bacc[j]) / h;
        facc[j] = accel[j] + (jerk[j] * h);
    }
    /*
        ELASTIX
        -----
*/
    if (flag == 0) {
        elastix(mass,dia,l,ymodofe,xp,xv,accel,s,delta,flag);

    /*
        FLEX
        ----
*/
        flex(s,delta,xp,l,tart,rp,flag,refx2,refy2);
    }
    flag = 0;
    /*
        RESET bacc[j]
*/
    for (j = 0; j < 2; j++)
        bacc[j] = accel[j];

```

```
/*                      WITHOUT THE FLEX COMPENSATION                      */
```

```
for (j = 0; j < 2; j++) rp[j] = tart[j];
```

```
/*                      MRAC ALGORITHM                      */
```

```
-----
```

```
/* DETERMINE ERROR BETWEEN COMMANDED POSITION AND ACTUAL POSITION */
```

```
for(j = 0; j < 2; j++) {
    clxp[j] = rp[j] - xp[j];
    manipint[j] += clxp[j] * h;
    clxmp[j] = tart[j] - xmp[j];
    modelint[j] += clxmp[j] * h;
}
```

```
/*                      APPLY THE PID CONTROLLER TO THE MANIPULATOR      */
```

```
crossmul(gi, manipint, tempi);
crossmul(gp, clxp, temp);
crossmul(gd, xv, tempv);
```

```
/*FIND ERRORS IN POSITION AND VELOCITY BETWEEN MODEL AND MANIPULATOR*/
```

```
for (j = 0; j < 2; j++) {
    u[j] = temp[j] + tempi[j] - tempv[j];
    erp[j] = xmp[j] - xp[j];
    erv[j] = xmv[j] - xv[j];
}
```

```
/*                      APPLY ERROR VECTOR GAINS                          */
```

```
crossmul(cp, erp, condp);
crossmul(cv, erv, condv);
crossmul(ce, err, conde);
```



```

for(j = 0; j < 2; j++) {
    y[j] = condv[j] + condp[j] ;
}

```

```

/* PROVIDE AN INITIAL GUESS FOR THE ESTIMATED INERTIA AND COUPLING
MATRICES */

```

```

if (i == 1) {
    nmod[0][0] = 8;
    nmod[0][1] = 2;
    nmod[1][0] = 2;
    nmod[1][1] = 1;
    nmod[0][1] = 0;
    nmod[1][1] = 0;
}

```

```

/* ADAPTATION ALGORITHM FOR ESTIMATED INERTIA AND COUPLING MATRICES*/

```

```

if (i > 1) {
    delnmod[0][0] = kn[0][0] * y[0] * u[0];
    delnmod[0][1] = kn[0][1] * (y[0] * u[1] + y[1] * u[0]);
    delnmod[1][0] = delnmod[0][1];
    delnmod[1][1] = kn[1][1] * y[1] * u[1];
    delnmod[0][1] = kn[0][1] * ((2 * y[0] * xv[0] * xv[1]) - y[1] * sq(xv[0]));
    delnmod[1][1] = kn[1][1] * y[0] * sq(xv[1]);
    for (j = 0; j < 2; j++) {
        for (k = 0; k < 2; k++) {
            nmod[j][k] += delnmod[j][k] * h;
            nmod[j][k] += delnmod[j][k] * h;
        }
    }
}

```

```

/*DUE TO THE ABSENCE OF AN EXPERIMENTAL ROBOT, THE ROBOT MOTION AND
DYNAMICS HAVE TO BE SIMULATED; THEREFORE, NEED INERTIA AND COUPLING
MATRICES OF THE ROBOT ITSELF. */

```

```

mmanip[0][0] = (mass[0]+mass[1])*sq(l[0]) + (mass[1]*sq(l[1]))
              + (2.0 * mass[1] * l[0] * l[1] * cos(xp[1]));
mmanip[1][1] = mass[1] * sq(l[1]);
mmanip[0][1] = mmanip[1][1]+mass[1]*l[0]*l[1]*cos(xp[1]);
mmanip[1][0] = mmanip[0][1];
fp[0][1] = -1.0 * mass[1] * l[0] * l[1] * sin(xp[1]);
fp[0][0] = 2.0 * fp[0][1];
fp[1][0] = -1.0 * fp[0][1];
fv[0][0] = (mass[0] + mass[1]) * g * l[0] * cos(xp[0]);
fv[1][1] = mass[1] * g * l[1] * cos(xp[0] + xp[1]);
fv[0][1] = fv[1][1];
mmanip[0][1] = fp[0][0]*xv[0]*xv[1]+fp[0][1]*sq(xv[1])
              +fv[0][0]+fv[0][1];
mmanip[1][1] = fp[1][0] * sq(xv[0]) + fv[1][1];

/*      FIND THE INVERSE OF THE INERTIA MATRIX OF THE MANIPULATOR      */

det = (mmanip[0][0]*mmanip[1][1])-(mmanip[0][1]*mmanip[1][0]);
adj[0][0] = mmanip[1][1];
adj[1][1] = mmanip[0][0];
adj[0][1] = -mmanip[0][1];
adj[1][0] = -mmanip[1][0];
for (j = 0; j < 2; j++) {
    for (k = 0; k < 2; k++)
        minv[j][k] = adj[j][k]/det;
}

/*      OBTAIN THE ACCELERATION VECTOR OF THE MANIPULATOR      */

crossmul(mmod,u,x);
for (j = 0; j < 2; j++)
    interu[j] = x[j] + mmod[j][1] - mmanip[j][1];
crossmul(minv,interu,accel);

/*INTEGRATE TWICE TO OBTAIN VELOCITY AND ACCELERATION OF MANIPULATOR*/

```

```

for (j = 0; j < 2; j++) {
    xv[j] += accel[j] * h;
    xp[j] += xv[j] * h;
}

```

/* APPLY THE PID CONTROLLER TO THE MODEL TO OBTAIN ITS ACCELERATION */

```

crossmul(gi,modelint,temp1);
crossmul(gp,clxmp,tempp);
crossmul(gd,xmv,tempv);

```

/* INTEGRATE TWICE TO OBTAIN THE VELOCITY AND ACCELERATION OF MODEL */

```

for (j = 0; j < 2; j++) {
    modelacc[j] = tempp[j] - tempv[j];
    xmv[j] += modelacc[j] * h;
    xmp[j] += xmv[j] * h;
}

```

```

phi[0] = delta[0]/l[0];
phi[1] = delta[1]/l[1];
x2 = l[0]*cos(xp[0]-phi[0]) + l[1]*cos(xp[0]-phi[0] +
    xp[1]-phi[1]-s[0]);
y2 = l[0]*sin(xp[0]-phi[0]) + l[1]*sin(xp[0]-phi[0] +
    xp[1]-phi[1]-s[0]);

```

/*

OUTPUT

*/

```

if (i <= 100) {
    if (divide10 == 10)
        counter = 1;
}

```

```

if (i > 100) {
    if (divide10 == 100)
        counter = 1;
}

if (counter == 1) {
    fscanf(fp, " %f %f ", &refx2, &refy2);
    error = hypot((refx2-x2), (refy2-y2));
    error *= 1000.0;
    printf("ITERATION NO.: %d ", i);
    printf("          ERROR = %f mm \n", error);
    printf("REFERENCE : %f %f \n", refx2, refy2);
    printf("COORDINATES: %f %f \n", x2, y2);
    printf("POSITION = %f %f \n", xp[0], xp[1]);
    printf("IDEAL = %f %f \n", rp[0], rp[1]);
    printf("VELOCITY = %f %f \n", xv[0], xv[1]);
    printf("ACCELERATION = %f %f \n", accel[0], accel[1]);
    printf("TORQUE = %f %f \n", interu[0], interu[1]);
    printf("*****\n \n");
    divide10 = 0;
    counter = 0;
}

i += 1;
divide10 += 1;
}

```

/* DETERMINE AND PRINT THE ERROR IN FINAL END-EFFECTOR POSITIONING */

```

x2 = l[0] * cos(xp[0]) + l[1] * cos(xp[0] + xp[1]);
y2 = l[0] * sin(xp[0]) + l[1] * sin(xp[0] + xp[1]);
error = hypot((refx2 - x2), (refy2 - y2));
printf("ERROR = %f mm \n", 1000.0 * error);
}

```

```

/*
/*****

```

APPENDIX 5

SIMULATION PROGRAM FOR DUAL-ARM MANIPULATOR

```
/******  
*  
* Model Reference Adaptive Control for a Dual-arm Manipulator *  
* Each arm has 2 LINKS WITH 3 DOF *  
*  
* HIERARCHICAL ADAPTIVE CONTROL *  
*  
*****  
* INPUT - link lengths and masses; *  
* cartesian coordinates of the final END-EFFECTOR position. *  
* INITIAL - determine optimum theta-1, theta-2 < CART2POL > ; *  
* PROCESS *  
* LOOP - apply the MRAC algorithm; *  
* OUTPUT - required angular position of the links; *  
* iterated values of : POSITION; *  
* VELOCITY; *  
* ACCELERATION; *  
* TORQUE; *  
* final coordinates of the end-effector and error. *
```

```

*
*****/

```

```

#include <stdio.h>
#include <math.h>
#define sq(n) n*n
#define cub(m) m*m*m
#define pow4(o) o*o*o*o
FILE *fptr, *fopen();

```

```

/*****
*
*           SUB-ROUTINES FOR THE MRAC STRATEGY.
*
*****/

```

```

crossmul(mat, vec, x)
float mat[2][3][3], vec[2][3], *x;
{
    int j, k;
    for (j = 0; j < 2; j++) {
        for (k = 0; k < 3; k++) {
            *(x+3*j+k) = mat[j][k][0]*vec[j][0]+mat[j][k][1]*vec[j][1]
                        +mat[j][k][2]*vec[j][2];
        }
    }
}

```

```

/*****
*
*           END OF SUB-ROUTINES
*
*****/

```

```

/*****

```



```

pi = 3.1415926;
km[0][0][0] = 5.0;
km[0][0][1] = 2.0;
km[0][0][2] = 1.0;
printf("\n %f %f %f\n", km[0][0][0], km[0][0][1], km[0][0][2]);

km[0][1][0] = 2.0;
km[0][1][1] = 5.0;
km[0][1][2] = 2.0;
printf("\n %f %f %f\n", km[0][1][0], km[0][1][1], km[0][1][2]);

km[0][2][0] = 1.0;
km[0][2][1] = 2.0;
km[0][2][2] = 5.0;
printf("\n %f %f %f\n", km[0][2][0], km[0][2][1], km[0][2][2]);

km[0][0][0] = 0;
km[0][0][1] = 1.0;
km[0][0][2] = 2.0;
printf("\n %f %f %f\n", km[0][0][0], km[0][0][1], km[0][0][2]);

km[0][1][0] = 0;
km[0][1][1] = 0;
km[0][1][2] = 1.0;
printf("\n %f %f %f\n", km[0][1][0], km[0][1][1], km[0][1][2]);

km[0][2][0] = 0;
km[0][2][1] = 0;
km[0][2][2] = 0;
printf("\n %f %f %f\n", km[0][2][0], km[0][2][1], km[0][2][2]);

for (j = 0; j < 3; j++) {
    for (k = 0; k < 3; k++) {
        km[1][j][k] = km[0][j][k];
        km[1][j][k] = km[0][j][k];
    }
}

```



```

    }
}

for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
        for (k = 0; k < 3; k++) {
            fv[i][j][k] = 0;
            fp[i][j][k] = fv[i][j][k];
        }
    }
}

```

```

cv[0][0][0] = 20.0;
cp[0][0][0] = 10.0;
printf("\n %f      %f\n", cv[0][0][0], cp[0][0][0]);
for (i = 0; i < 2; i++) {
    cv[i][0][0] = cv[0][0][0];
    cv[i][0][1] = 0;
    cv[i][0][2] = 0;
    cv[i][1][0] = 0;
    cv[i][1][1] = cv[0][0][0];
    cv[i][1][2] = 0;
    cv[i][2][0] = 0;
    cv[i][2][1] = 0;
    cv[i][2][2] = cv[0][0][0];
    cp[i][0][0] = cp[0][0][0];
    cp[i][0][1] = 0;
    cp[i][0][2] = 0;
    cp[i][1][0] = 0;
    cp[i][1][1] = cp[0][0][0];
    cp[i][1][2] = 0;
    cp[i][2][0] = 0;
    cp[i][2][1] = 0;
    cp[i][2][2] = cp[0][0][0];
}

```

}

l[0][0] = 1.0;

l[0][1] = 1.0;

printf("\nLENGTHS ARE: l1 = %f l2 = %f \n", l[0][0], l[0][1]);

fscanf(fptr, " %f %f ", &mass[0][0], &mass[0][1]);

printf("\n 1st MASSES ARE: m1 = %f m2 = %f \n", mass[0][0],
mass[0][1]);

fscanf(fptr, " %f %f ", &mass[1][0], &mass[1][1]);

printf("\n 2nd MASSES ARE: m1 = %f m2 = %f \n", mass[1][0],
mass[1][1]);

for (k = 0; k < 2; k++) {

l[1][k] = l[0][k];

}

it = 1;

flag = 0;

counter = 0;

nn = 10000;

divide10 = 1;

gp[0][0][0] = 160.0;

gi[0][0][0] = 50.0;

gd[0][0][0] = 160.0;

printf("\nPID CONTROLLER GAINS ARE: P = %f I = %f D = %f \n",

gp[0][0][0], gi[0][0][0], gd[0][0][0]);

for (i = 0; i < 2; i++) {

gp[1][0][0] = gp[0][0][0];

gp[1][0][1] = 0;

gp[1][0][2] = 0;

gp[1][1][0] = 0;

gp[1][1][1] = gp[0][0][0];

gp[1][1][2] = 0;

```

    gp[1][2][0] = 0;
    gp[1][2][1] = 0;
    gp[1][2][2] = gp[0][0][0];
    gi[1][0][0] = gi[0][0][0];
    gi[1][0][1] = 0;
    gi[1][0][2] = 0;
    gi[1][1][0] = 0;
    gi[1][1][1] = gi[0][0][0];
    gi[1][1][2] = 0;
    gi[1][2][0] = 0;
    gi[1][2][1] = 0;
    gi[1][2][2] = gi[0][0][0];
    gd[1][0][0] = gd[0][0][0];
    gd[1][0][1] = 0;
    gd[1][0][2] = 0;
    gd[1][1][0] = 0;
    gd[1][1][1] = gd[0][0][0];
    gd[1][1][2] = 0;
    gd[1][2][0] = 0;
    gd[1][2][1] = 0;
    gd[1][2][2] = gd[0][0][0];
}

force = 2.0;

printf("NORMAL END-EFFECTOR FORCES:  %f N  %f N \n",force,-force);

h = 0.001;

meter = 0;

for (i = 0; i < 3; i++) {
    difference[i] = 0;
    prediff[i] = 0;
    diffdot[i] = 0;
}

for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
        xp[i][j] = 0;
        xv[i][j] = 0;
    }
}

```

```

    xmp[i][j] = xp[i][j];
    xmv[i][j] = xv[i][j];
    interu[i][j] = 0;
    accel[i][j] = 0;
    acc[i][j] = 0;
    clxp[i][j] = 0;
    clxmp[i][j] = 0;
    clxv[i][j] = 0;
    clxmv[i][j] = 0;
}
}

fscanf(fp, " %f %f %f ", &factor, &gain, &swit);
printf(" FACTOR = %f GAIN = %f SWITCH = %f \n", factor, gain, swit);

/*****
*
*                      NRAC  ALGORITHM
*
*****/

while (it <= nn) {
    meter += pi/5000;
    if (meter > pi)
        meter = pi;
    rp[0][0] = 0.5 * (1 - cos(meter));
    rp[0][1] = rp[0][0]/2.0;
    rp[0][2] = rp[0][0]/3.0;

    rp[1][0] = rp[0][0];
    rp[1][1] = rp[0][1];
    rp[1][2] = rp[0][2];
    if (swit == 1) {
        rp[1][0] = rp[0][0] + correction[0];
        rp[1][1] = rp[0][1] + correction[1];
        rp[1][2] = rp[0][2] + correction[2];
    }
}

```

```

/*****
* DETERMINE THE ERROR BETWEEN COMMANDED POSITION AND ACTUAL POSITION *
*****/

```

```

    for (i = 0; i < 2; i++) {
        for(j = 0; j < 3; j++) {
            clxp[i][j] = rp[i][j] - xp[i][j];
            manipint[i][j] += clxp[i][j] * h;
            clxmp[i][j] = rp[i][j] - xmp[i][j];
            modelint[i][j] += clxmp[i][j] * h;
        }
    }

```

```

/*****
*                               APPLY THE PID CONTROLLER TO THE MANIPULATOR                               *
*****/

```

```

    crossmul(gi, manipint, tempi);
    crossmul(gp, clxp, temp);
    crossmul(gd, xv, tempv);

```

```

/*****
* FIND ERRORS IN POSITION AND VELOCITY BETWEEN MODEL AND MANIPULATOR *
*****/

```

```

    for (i = 0; i < 2; i++) {
        for (j = 0; j < 3; j++) {
            u[i][j] = temp[i][j] + tempi[i][j] - tempv[i][j];
            exp[i][j] = xmp[i][j] - xp[i][j];
            erv[i][j] = xmv[i][j] - xv[i][j];
        }
    }

```

```

/*****
*                               APPLY ERROR VECTOR GAINS                               *
*****/

```

```

*****/

    crossmul(cp,exp,condp);
    crossmul(cv,erv,condv);
    for (i = 0; i < 2; i++) {
        for(j = 0; j < 3; j++) {
            y[i][j] = condv[i][j] + condp[i][j];
        }
    }

/*****
* PROVIDE AN INITIAL GUESS FOR THE ESTIMATED INERTIA AND COUPLING . *
*          MATRICES          *
*****/

    if (it == 1) {
        for (i = 0; i < 2; i++) {
            mmod[i][0][0] = 1.0;
            mmod[i][0][1] = 0;
            mmod[i][0][2] = 0;
            mmod[i][1][0] = 0;
            mmod[i][1][1] = 1.0;
            mmod[i][1][2] = 0;
            mmod[i][2][0] = 0;
            mmod[i][2][1] = 0;
            mmod[i][2][2] = 1.0;
            for (j = 0; j < 3; j++) {
                for (k = 0; k < 3; k++) mmod[i][j][k] = 0;
            }
        }
    }

/*****
* ADAPTATION ALGORITHM FOR ESTIMATED INERTIA AND COUPLING MATRICES *
*****/

```

```

if (it > 1) {
  for (i = 0; i < 2; i++) {
    delnmod[i][0][0] = kn[i][0][0]*y[i][0]+u[i][0];
    delnmod[i][0][1] = kn[i][0][1]*(y[i][0]+u[i][1]+y[i][1]+u[i][0]);
    delnmod[i][0][2] = kn[i][0][2]*(y[i][0]+u[i][2]+y[i][2]+u[i][0]);
    delnmod[i][1][0] = delnmod[i][0][1];
    delnmod[i][1][1] = kn[i][1][1]*y[i][1]+u[i][1];
    delnmod[i][1][2] = kn[i][1][2]*(y[i][1]+u[i][2]+y[i][2]+u[i][1]);
    delnmod[i][2][0] = delnmod[i][0][2];
    delnmod[i][2][1] = delnmod[i][1][2];
    delnmod[i][2][2] = kn[i][2][2] * y[i][2] * u[i][2];
    delnmod[i][0][2] = kn[i][0][2] * ((2 * y[i][0] * xv[i][0] *
      xv[i][2]) - y[i][2] * sq(xv[i][0]));
    delnmod[i][1][2] = kn[i][1][2] * ((2 * y[i][1] * xv[i][1] *
      xv[i][2]) - y[i][2] * sq(xv[i][1]));
    delnmod[i][2][2] = kn[i][2][2] * y[i][2] * sq(xv[i][2]);
    for (j = 0; j < 3; j++) {
      for (k = 0; k < 3; k++) {
        mmod[i][j][k] += delnmod[i][j][k] * h;
        mmod[i][j][k] += delnmod[i][j][k] * h;
      }
    }
  }
}

```

```

/*****
* DUE TO THE ABSENCE OF AN EXPERIMENTAL ROBOT, THE ROBOT MOTION AND
* DYNAMICS HAVE TO BE SIMULATED. THEREFORE, NEED INERTIA AND COUPLING*
* MATRICES OF THE ROBOT ITSELF
*****/

```

```

for (i = 0; i < 2; i++) {
  mmanip[i][0][0] = (mass[i][0]+mass[i][1])*sq(l[i][0])*
    sq(cos(xp[i][1])) + (mass[i][1])*sq(l[i][1])

```

```

*sq(cos(xp[i][1]+xp[i][2])) + (2.0*mass[i][1]*l[i][0]*
l[i][1]*cos(xp[i][1])*cos(xp[i][1]+xp[i][2]));
mmanip[i][0][1] = 0;
mmanip[i][0][2] = 0;
mmanip[i][1][0] = mmanip[i][0][1];
mmanip[i][1][1] = (mass[i][0]+mass[i][1])*sq(l[i][0]) +
mass[i][1]*sq(l[i][1]) + 2.0*mass[i][1]*l[i][0]*l[i][1]
*cos(xp[i][2]);
mmanip[i][1][2] = mass[i][1]*sq(l[i][1]) + mass[i][1]*l[i][0]*
l[i][1]*cos(xp[i][2]);
mmanip[i][2][0] = mmanip[i][0][2];
mmanip[i][2][1] = mmanip[i][1][2];
mmanip[i][2][2] = mass[i][1] * sq(l[i][1]);

fp[i][0][2] = (mass[i][0]+mass[i][1])*l[i][0]*l[i][0]*
cos(xp[i][1])*sin(xp[i][1]) + mass[i][1]*l[i][1]*
l[i][1]*sin(xp[i][1]+xp[i][2])*cos(xp[i][1]+xp[i][2]) +
mass[i][1]*l[i][0]*l[i][1]*sin(2.0*xp[i][1] + xp[i][2]);
fp[i][1][2] = mass[i][1]*l[i][1]*l[i][1]*sin(xp[i][1]
+xp[i][2])*cos(xp[i][1]+xp[i][2]) + mass[i][1]*
l[i][0]*l[i][1]*cos(xp[i][1])*sin(xp[i][1]+xp[i][2]);
fp[i][2][2] = mass[i][1]*l[i][0]*l[i][1]*sin(xp[i][2]);

fv[i][1][1] = (mass[i][0]+mass[i][1])*l[i][0]*cos(xp[i][1]) +
mass[i][1]*l[i][1]*cos(xp[i][1]+xp[i][2]);
fv[i][2][2] = mass[i][1]*l[i][1]*cos(xp[i][1]+xp[i][2]);

mmanip[i][0][1] = (-2.0)*fp[i][0][2]*xv[i][0]*xv[i][1] -
2.0*fp[i][1][2]*xv[i][0]*xv[i][2];
mmanip[i][1][1] = fp[i][0][2]*sq(xv[i][0]) -
fp[i][2][2]*sq(xv[i][2]) - 2.0*fp[i][2][2]*xv[i][1]*xv[i][2]
+ fv[i][1][1]*g;

mmanip[i][2][1] = fp[i][1][2]*sq(xv[i][0]) +
fp[i][2][2]*sq(xv[i][1]) + fv[i][2][2]*g;

```



```

}

/*****
* FIND THE INVERSE OF THE INERTIA MATRIX OF THE MANIPULATOR
*****/

```

```

for (i = 0; i < 2; i++) {
    det = mmanip[i][0][0] * (mmanip[i][1][1]*mmanip[i][2][2] -
                             mmanip[i][1][2] * mmanip[i][2][1])
        + mmanip[i][0][1] * (mmanip[i][1][2]*mmanip[i][2][0] -
                             mmanip[i][1][0]*mmanip[i][2][2])
        + mmanip[i][0][2] * (mmanip[i][1][0]*mmanip[i][2][1] -
                             mmanip[i][1][1]*mmanip[i][2][0]);

    adj[i][0][0] = mmanip[i][1][1]*mmanip[i][2][2] -
                  mmanip[i][1][2]*mmanip[i][2][1];
    adj[i][0][1] = mmanip[i][0][2]*mmanip[i][2][1] -
                  mmanip[i][0][1]*mmanip[i][2][2];
    adj[i][0][2] = mmanip[i][0][1]*mmanip[i][1][2] -
                  mmanip[i][1][1]*mmanip[i][0][2];
    adj[i][1][0] = mmanip[i][2][0]*mmanip[i][1][2] -
                  mmanip[i][1][0]*mmanip[i][2][2];
    adj[i][1][1] = mmanip[i][0][0]*mmanip[i][2][2] -
                  mmanip[i][0][2]*mmanip[i][2][0];
    adj[i][1][2] = mmanip[i][1][0]*mmanip[i][0][2] -
                  mmanip[i][0][0]*mmanip[i][1][2];
    adj[i][2][0] = mmanip[i][1][0]*mmanip[i][2][1] -
                  mmanip[i][2][0]*mmanip[i][1][1];
    adj[i][2][1] = mmanip[i][2][0]*mmanip[i][0][1] -
                  mmanip[i][0][0]*mmanip[i][2][1];
    adj[i][2][2] = mmanip[i][0][0]*mmanip[i][1][1] -
                  mmanip[i][1][0]*mmanip[i][0][1];

    for (j = 0; j < 3; j++) {
        for (k = 0; k < 3; k++) minv[i][j][k] = adj[i][j][k]/det;
    }
}

```

```
/* OBTAIN THE ACCELERATION VECTOR OF THE MANIPULATOR */
```

```
crossmul(nmod,u,x);
for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
        intertork[i][j] = x[i][j] + nmod[i][j][1];
        interu[i][j] = intertork[i][j] - manip[i][j][1];
    }
}
crossmul(minv,interu,accel);
```

```
/* INTEGRATE TWICE TO OBTAIN THE VELOCITY AND POSITION OF THE
MANIPULATOR */
```

```
for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
        xv[i][j] += accel[i][j] * h;
        xp[i][j] += xv[i][j] * h;
    }
}

arm1sqr = sq(1[0][0]) + sq(1[0][1]) - 2.0*1[0][0]*1[0][1]*
cos(pi - xp[0][1]);
arm2sqr = sq(1[1][0]) + sq(1[1][1]) - 2.0*1[1][0]*1[1][1]*
cos(pi - xp[1][1]);

if (arm1sqr >= 0) {
    arm[0] = sqrt(arm1sqr);
}

if (arm2sqr >= 0) {
    arm[1] = sqrt(arm2sqr);
}

for (j = 0; j < 2; j++) moment[j] = force * arm[j];
motortork[0] = moment[0]*cos(xp[0][0] + xp[0][1]/2.0);
motortork[1] = moment[1]*cos(xp[1][0] + xp[1][1]/2.0);
motortork[1] += intertork[1][0];
motortork[1] += -1.0;
```

```
/* APPLY THE PID CONTROLLER TO THE MODEL TO OBTAIN ITS ACCELERATION */
```

```
crossmul(gp, clxmp, temp);
```

```
crossmul(gd, xmv, tempv);
```

```
/* INTEGRATE TWICE TO OBTAIN VELOCITY AND ACCELERATION OF THE MODEL */
```

```
for (i = 0; i < 2; i++) {
    for (j = 0; j < 3; j++) {
        acc[i][j] = temp[i][j] - tempv[i][j];
        xmv[i][j] += acc[i][j] * h;
        xmp[i][j] += xmv[i][j] * h;
    }
}
```

```
/* DETERMINE AND PRINT THE ERROR IN FINAL END-EFFECTOR POSITIONING */
```

```
for (i = 0; i < 2; i++) {
    actualx[i] = l[i][0]*cos(xp[i][0])*cos(xp[i][1])+l[i][1]*
        cos(xp[i][0])*cos(xp[i][1]+xp[i][2]);
    actualy[i] = l[i][0]*sin(xp[i][0])*cos(xp[i][1])+l[i][1]*
        sin(xp[i][0])*cos(xp[i][1]+xp[i][2]);
    actualz[i] = l[i][0]*sin(xp[i][1])+l[i][1]*sin(xp[i][1]+
        xp[i][2]);
    refx[i] = l[i][0]*cos(rp[i][0])*cos(rp[i][1])+l[i][1]*
        cos(rp[i][0])*cos(rp[i][1]+rp[i][2]);
    refy[i] = l[i][0]*sin(rp[i][0])*cos(rp[i][1])+l[i][1]*
        sin(rp[i][0])*cos(rp[i][1]+rp[i][2]);
    refz[i] = l[i][0]*sin(rp[i][1])+l[i][1]*sin(rp[i][1]+rp[i][2]);
}
error[0]=1000*hypot((actualx[0]-actualx[1]),
    (actualy[0]-actualy[1]), (actualz[0]-actualz[1]));
```

```

for (i = 0; i < 3; i++) {
    difference[i] = xp[0][i] - xp[1][i];
    diffdot[i] = (difference[i] - prediff[i]) / h;
    prediff[i] = difference[i];
}

if (it == 1000) gain *= factor;
for (j = 0; j < 3; j++) correction[j] = (gain)*difference[j];

```

```

/*                                OUTPUT
   -----
*/

```

```

if (it <= 100) {
    if (divide10 == 10) counter = 1;
}
if (it > 100) {
    if (divide10 == 100) counter = 1;
}

if (counter == 1) {
    printf("ITERATION NO.: %d ERROR = %f == ", it, error[0]);
    printf("MOTOR TORQUES: %f N-m %f N-m \n",
            intertork[0][0], intertork[1][0]);
    printf("TARGET (rad) = %f %f %f \n", rp[0][0], rp[0][1],
            rp[0][2]);
    printf("POSITION1 (rad) = %f %f %f \n", xp[0][0], xp[0][1],
            xp[0][2]);
    printf("POSITION2 (rad) = %f %f %f \n", xp[1][0], xp[1][1],
            xp[1][2]);
    printf("DIFFERENCE (rad) = %f %f %f \n", difference[0],
            difference[1], difference[2]);
    printf("VELOCITY 1 = %f %f %f \n", xv[0][0], xv[0][1],
            xv[0][2]);
    printf("VELOCITY 2 = %f %f %f \n", xv[1][0], xv[1][1],
            xv[1][2]);
    printf("ACCELERATION1 = %f %f %f \n", accel[0][0],
            accel[0][1], accel[0][2]);
}

```

3 of/de 3

MicroD

```

printf("ACCELERATION2  = %f %f %f \n", accel[1][0],
      accel[1][1], accel[1][2]);
if (swit == 1) printf("CORRECTIONS: %f %f \n",
      correction[0], correction[1]);
printf("\n*****\n"); /*
divide10 = 0;
counter = 0;
}
it+= 1;
divide10 += 1;
}

/*
/*****

```

BIBLIOGRAPHY

- [1] GROOVER M. P. and ZIMMERS E. W., CAD/CAM, Prentice-Hall, 1984.
- [2] DRAPER C. S. and LI Y. T., Principles of Optimizing Control Systems and an Application to the Internal Combustion Engine, ASME Publication, 1951.
- [3] TSIEN H. S., Engineering Cybernetics, McGraw-Hill, New York, 1954.
- [4] ASHBY W. R., Design for a Brain, John Wiley, New York, 1960.
- [5] WHITAKER H. P., YAMRON J., and KEZER A., Design of Model Reference Adaptive Control Systems for Aircraft, Report no. ~~R-164~~, Instrumentation Lab., MIT, 1958.
- [6] LI Y. T., and VAN DER VELDE-W. E., The Philosophy of non-linear Adaptive Control, Proc. IFAC Congress, Moscow, 1960.
- [7] PETROV B. N., ULANOV G. M., and EMEL'YANOV S. V., Invariantnost i optimizatsia sistem avtomaticheskogo regulirovaniia s zvestoko i peremennoi stukturoi, IFAC Congress, Basel, 1963.
- [8] BELLMAN R., Adaptive Control Processes: A Guided Tour, Princeton University Press, Princeton, N.J., 1961.
- [9] FEL'DBAUM A. A., Dual Control Theory I - IV, Auto. Rem. Control, 1960 - 1961.

- [10] LAINIOTIS D. G., Partitioning: A unifying framework for adaptive systems, Proc. IEEE, 1976.
- [11] ÅSTRÖM K. J., and WITTENMARK B., On Self-tuning Regulators, Automatica, 1973.
- [12] MONOPOLI R. V., Model Reference Adaptive Control with an Augmented Error Signal, IEEE Trans. Auto. Control, AC-19, pp. 474 - 484, 1974.
- [13] NARENDRA K. S., LIN Y. H., and VALAVANI L. S., Stable Adaptive Controller Design, IEEE Trans. Auto. Control, AC-25, pp. 440 - 448, 1980.
- [14] NARENDRA K. S., and LIN Y. H., Stable Discrete Adaptive Control, IEEE Trans. Auto. Control, AC-25, pp. 456 - 461, 1980.
- [15] LIN Y. H., and NARENDRA K. S., A New Error Model for Adaptive Systems, IEEE Trans. Auto. Control, AC-25, pp. 585 - 587, 1980.
- [16] MORSE A. S., Global Stability of Parameter-adaptive Control Systems, IEEE Trans. Auto. Control, AC-25, pp. 433 - 439, 1980.
- [17] GOODWIN G. C., RAMADGE P. J., and CAINES P. E., Discrete-time Multivariable Adaptive Control, IEEE Trans. Auto. Control, AC-25, pp. 449 - 456, 1980.
- [18] EGARDT B., Stability Analysis of Continuous-time Adaptive Control Systems, SIAM J. Control Optimiz., pp. 540 - 558, 1980.
- [19] EGARDT B., Stability Analysis of Discrete-data Adaptive Control Schemes, IEEE Trans. Auto. Control, AC-25, pp. 710 - 716, 1980.
- [20] SARIDIS G. N., Self-Organizing Control of Stochastic Systems, Marcel - Dekker, New York, 1977.
- [21] EGARDT B., Stability of Adaptive Controllers, Springer-Verlag,

Berlin, 1979.

- [22] LANDAU I. D., Adaptive Control: The Model Reference Approach, Marcel - Dekker, New York, 1979.
- [23] CHALAM V. V., Adaptive Control Systems: Techniques and Applications, Marcel - Dekker, New York, 1987.
- [24] NARENDRA K. S. and MONOPOLI R. V. (eds.), Applications of Adaptive Control, Academic Press, New York, 1980.
- [25] UNBEHAUEN H., (ed.), Methods and Applications in Adaptive Control, Springer-Verlag, Berlin, 1980.
- [26] HARRIS C. J., and BILLINGS S. A., (eds.), Self-tuning and Adaptive Control: Theory and Applications, Peter Peregrinus, London, 1981.
- [27] IOANNOU P. A., and KOKOTOVIC P. V., Adaptive Systems with Reduced Models, Springer-Verlag, Berlin, 1983.
- [28] ÅSTRÖM K. J., and WITTENMARK B., Computer - Controlled Systems: Theory and Design, Prentice-Hall, Englewood Cliffs, N. J., 1984.
- [29] GOODWIN G. C., and SIN K. S., Adaptive Filtering Prediction and Control, Prentice-Hall, Englewood Cliffs, N. J., 1984.
- [30] DUBOWSKY S., and DES FORGES D. T., The Application of MRAC to Robotic Manipulators, ASME Trans., J. Dyn. Syst. Meas. Contr., 1979.
- [31] DUBOWSKY S., and DES FORGES D. T., Robotic Manipulator Control Systems with Invariant Dynamic Characteristics, Proc. 5th World Congress on Machines and Mechanisms (IFIOMM), Montreal, 1979.

- [32] ARIMOTO S., and TAKEGAKI M., An Adaptive Method for Trajectory Control of Manipulators, Proc. 8th IFAC World Congress, Kyoto, pp. 1921 - 1926, 1981.
- [33] KOIVO A. J., ZHANG Q., and GUO T. -H., Industrial Manipulator Control using Kalman Filter and Adaptive Controller, Proc. IFAC Symp. Identification and System Parameter Estimation, Washington D. C., pp. 265 - 270, 1982.
- [34] KOIVO A. J., and GUO T. -H., Adaptive Linear Controller for Robotic Manipulators, IEEE Trans. Auto. Contr. AC-28, pp. 162 - 171, 1983.
- [35] TOMIZUKA M., and HOROWITZ R., Model Reference Adaptive Control of Mechanical Manipulators, Proc. IFAC Workshop on Adaptive Systems in Control and Signal Processing, San Francisco, pp. 27-32, 1983.
- [36] VUKOBRATOVIC M., STOKIC D., and KIRCANSKI N., Towards Non- adaptive and Adaptive Control of Manipulation Robots, IEEE Trans. Auto. Contr. AC-29, pp. 841 - 844, 1984.
- [37] LEE C. S. G., and CHUNG M. J., An Adaptive Control Strategy for Computer-based Manipulators, Proc. IEEE Conf. Decision and Control, Orlando, Fla., pp. 95 - 100, 1982.
- [38] LEE C. S. G., and CHUNG M. J., An Adaptive Control Strategy for Mechanical Manipulators, IEEE Trans. Auto. Contr. AC-29, pp. 837 - 840, 1984.
- [39] LEE C. S. G., and CHUNG M. J., Adaptive Perturbation Control with Feedforward Compensation for Robot Manipulators, Simulation, 44: pp. 127 - 136, 1985.

- [40] LANDAU I. D., A Survey of Model Reference Adaptive Techniques (Theory and Applications), Automatica, 10: pp. 353 - 379, 1974.
- [41] CRAIG J. J., Introduction to Robotics : Mechanics and Control, Addison -Wesley Publishing Company, 1986.
- [42] HOROWITZ R., Model Reference Adaptive Control of Mechanical Manipulators, Ph.D. Dissertation, University of California, Berkeley, 1983.
- [43] SUTHERLAND J. L., Model Reference Adaptive Control of a Two-link Manipulator, M. Eng. Thesis, Carleton University, Ottawa, Ontario, 1987.
- [44] POPOV V. M., Hyperstability of Control Systems, Springer-Verlag, Berlin, 1973.
- [45] ASARE H. R., and WILSON D. G., Evaluation of Three Model Reference Adaptive Control Algorithms for Robotic Manipulators, Proc. IEEE Int'l Conf. Robotics and Automation, April 1987, pp. 1531 - 1542.
- [46] SASIADEK J. Z., Two-link Lightweight Flexible Manipulator, Proc. IASTED Int'l Conf. Advances in Robotics, Santa Barbara, CA, May 1985, Vol. 1, pp. 152 - 155.
- [47] SCHAFER B. E., and HOLZACH H., Experimental Research on Flexible Beam Modal Control, Proc. AIAA Dynamics Specialists Conf., Palm Springs, CA, May 1984.
- [48] MELDRUM D. R., and BALAS M. J., Application of Model Reference Adaptive Control to a Flexible Remote Manipulator Arm, Proc. 1986 ACC, Seattle, WA, June 1986, Vol. 2, pp. 825 -

832.

- [49] USORO P. B., and NADIRA R., Analysis of Light-weight Flexible Manipulator Dynamics, ASME Publication: Computers in Engineering, 1984, pp. 167 - 174.
- [50] USORO P. B., NADIRA R., and MAHIL S. S., Control of Light-weight Flexible Manipulators: A Feasibility Study, IEEE Proc. 1984 ACC, San Diego, California, June 1984, Vol. 3, pp. 1209 - 1216.
- [51] USORO P. B., NADIRA R., and MAHIL S. S., A Finite - Element / Lagrange Approach to Modeling Light-weight Flexible Manipulators, ASME J. Dyn. Syst. Meas. and Contr., September 1986, pp. 198 - 204.
- [52] HASTINGS G. G., and BOOK W. J., Verification of a Linear Dynamic Model for Flexible Robotic Manipulators, Proc. 1986 IEEE Int'l Conf. Robotics and Automation, Vol. 3, pp. 1024 - 1029.
- [53] OZGUNER U., and YURKOVICH S., Active Vibration Control on the OSU Flexible Beam, Proc. 1987 ACC, Minneapolis, MN, June 1987, Vol. 3, pp. 970 - 975.
- [54] SASIADEK J. Z., and SRINIVASAN R., Model Reference Adaptive Control for a Flexible Two-link Manipulator Arm, Proc. IFAC Int'l Symposium Theory of Robots, Vienna, Austria, Dec. 1986.
- [55] SASIADEK J. Z., and SRINIVASAN R., Adaptive Control for Flexible Manipulators, Proc. 3rd Canadian Universities Conf. CAD/CAM, Ottawa, Ontario, July 1987, pp. 150 - 159.
- [56] HOPKINS R. B., Design Analysis of Shafts and Beams, McGraw-Hill, New York, 1970.

- [57] BURKE S. E., and HUBBARD J. E. Jr., Active Vibration Control of a Simply Supported Beam using a Spatially Distributed Actuator, 1987 IEEE Control Systems Magazine.
- [58] SASIADEK J. Z., and SRINIVASAN R., Adaptive Control of Dual Arms, Proc. 1987 IEEE Int'l Conf. System, Man and Cybernetics, Oct. 1987, Alexandria, Va., pp. 690 - 694.
- [59] OZGUNER U., et al., Decentralized Variable Structure Control of a Two-arm Robotic System, Proc. 1987 IEEE Int'l Conf. on Robotics and Automation, Raleigh, N. Carolina.
- [60] TARN T. J., YUAN X., and BEJCZY A. K., Coordinated Control of Two Robot Arms, Proc. 1986 IEEE Int'l Conf. on Robotics and Automation, San Francisco, California.
- [61] ZHENG Y. F., and LUH J. Y. S., Control of Two Coordinated Robots in Motion, Proc. 24th IEEE Int'l Conf. on Decision and Control, Fort Lauderdale, 1985.
- [62] LIM J., and CHYUNG D. H., On a Control Scheme for Two Coordinating Robot Arms, Proc. 24th IEEE Int'l Conf. on Decision and Control, Fort Lauderdale, 1985.
- [63] ALFORD C., and BELYEU S., Coordinated Control of Two Arms, Proc. 1984 IEEE Int'l Conf. on Robotics and Automation, Atlanta, Georgia.
- [64] UCHIYAMA H., et al., Hybrid Position/Force Control for Coordination of a Two-arm Robot, Proc. 1987 IEEE Int'l Conf. on Robotics and Automation, Raleigh, N. Carolina.
- [65] KOIVO A. J., Adaptive Position-Velocity-Force Control of Two Manipulators, Proc. 24th IEEE Int'l Conf. on Decision and Control,

Fort Lauderdale, Florida, 1985.

- [66] HAYATI S., Hybrid Position/Force Control of Multi-arm Cooperating Robots, Proc. 1986 IEEE Int'l Conf. on Robotics and Automation, San Francisco, California.
- [67] ZAPATA R., FOURNIER A., and DAUCHEZ P., True Cooperation of Robots in Multi-arm Tasks, Proc. 1987 IEEE Int'l Conf. on Robotics and Automation, Raleigh, N. Carolina.

END

08-08-88

FIN