MOTION DESIGN, CONTROL AND IMPLEMENTATION

IN ROBOT MANIPULATORS

GEOFFREY WILLIAM VERNON

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy of the Council for National Academic Awards

Mechanisms and Machines Group

Liverpool Polytechnic

April 1988

PUBLISHED PAPERS NOT COPIED AT THE REQUEST OF THE UNIVERSITY

ACKNOWLEDGEMENTS

I would like to thank Mr. John Rees Jones, my academic supervisor, for his guidance and assistance during the program. Without the initial financial assistance from his group, and later from the SERC under their ACME initiative, none of my work could have been carried out. Many a 'long hour' was spent discussing many aspects of the project and its development. Further encouragement came from Dr. George T. Rooney, who took time out to relieve me of a number of the chores arising from the work, also helping me with computational problems.

My thanks are also due to Dr. Michael J. Gilmartin who as second supervisor helped by commenting more generally on my work, and by constructively reviewing intermediate written works. Other members of the group aided my early analytic & computational work, namely Dr. Sedat Baysec, Mr. Patrick Fischer and Mr. Chen Hai Zong. I would like to express thanks to Steve Caulder, technician on the project who 'got on with the job' with minimal effort on my part.

Without my grandfather, William Wild, who taught me to enquire, and the inspiration and encouragement given by Mr.R.Stanley Wilkes, I would not have pursued a mechanical engineering career.

I would like to thank Dr. David Stoten of Liverpool University (now at Bristol) for the development of my interest in dynamics and control of manipulators.

Last but not least, my dear wife Debbie and daughter Elizabeth who were subjected to long hours of imposed quiet, little socialising, and were still able to encourage me. Debbie also proof read much of this text.

ABSTRACT

MOTION DESIGN, CONTROL AND IMPLEMENTATION
IN ROBOT MANIPULATORS

by

Geoffrey W. Vernon

The dynamic performance of robots, specifically the tracking accuracy and motion duration, is influenced by both the nominal motion profile and the feedback control method employed.

Three schemes are developed and experimentally tested to tackle the improvement of dynamic performance, in the absence of accurate dynamic models.

Model Referenced Adaptive Controller Schemes (MRACS) can be designed to facilitate the characterisation of otherwise complex system dynamics. In one scheme an MRACS is used to force the robot to behave as if it were linear and decoupled, enabling simple model based dynamic tuning methods to be applied to the motion laws. Its promise as a technique is demonstrated, but the controller performance is found to be degraded by practical limitations. It is applied to both joint and Cartesian based motion laws.

A computer controlled robot contains all the elements necessary for an autonomous self experimentation system. This feature is exploited in the derivation and implementation of two further schemes which are termed self learning. In these, the robot's trajectory is stored as a set of discrete data. Algorithms are developed for tuning this data subsequent to each run. Their use requires minimal knowledge of the dynamics, no additional transducers and little computation.

The first of the self learning schemes is used to cyclically reduce the tracking errors. Once complete, the updating process can be curtailed. Errors on completion are close to the transducer resolution.

The second of these schemes involves an incremental reduction in the duration of a given motion. Various parameters for detecting saturation are proposed and tested. A normalised ratio of peak to average velocity is found promising.

Combining these two schemes, tuning for speed to near saturation then tuning for accuracy, provides a method for obtaining a near minimum time trajectory, with maximum possible tracking accuracy, at low cost.

CONTENTS

$\underset{\sim}{A}$ - system dynamics coefficient matrix (state space)

$\underset{\sim}{B}$ - system input coefficient matrix (state space)

$\underset{\sim}{C}$ - system output coefficient matrix (state space)

$\underset{\sim}{D}$ - MRACS equivalent linear part gain matrix

F - force

$\underset{\sim}{F}$ - vector of forces

$\underset{\sim}{F}$ - general vector function, including controller function

G - transfer function

$\underset{\sim}{G}$ - vector of robot gravitation coefficients

$\underset{\sim}{H}$ - robot inertia matrix

$\underset{\sim}{I}$ - identity matrix

$\underset{\sim}{J}$ - link inertia matrix

$\underset{\sim}{K}$ - feedback gain matrix

KE - kinetic energy

L - Laplace operator

$\underset{\sim}{L}$ - learning correction weighting matrix

$\underset{\sim}{M_i}$ - matrices used in the solution of the Lyapunov equation

$\underset{\sim}{N_i}$ - total vector moment exerted on link i

P - prismatic joint

$\underset{\sim}{P}$ - system matrix; discretised response solution

$\underset{\sim}{P}$ - positive definite symmetric matrix solution of the Lyapunov equation

PE - potential energy

$\underset{\sim}{Q}$ - input matrix; discretised response solution

$\underset{\sim}{Q}$ - Lyapunov equation condition matrix

R - duration reduction factor

R - revolute joint

$\underset{\sim}{R}$ - rotation or direction cosine matrix. $\underset{\sim}{R}_{ij}$ defines the transformation from axis frame i to axis frame j

$\underset{\sim}{S}$ - matrix used in the solution of the Lyapunov equation

SP_i - saturation parameter

T - motion segment duration

$\underset{\sim}{T}$ - matrix of time boundary conditions, derivative coefficients and powers

$\underset{\sim}{U}$ - matrix relating inertias measured about one set of axes to another

$\underset{\sim}{V}$ - matrix of robot Coriolis and centripetal coefficients

X - general X displacement coordinate

$\underset{\sim}{X}$ - unit vector defining X axis

Y - general Y displacement coordinate

$\underset{\sim}{Y}$ - unit vector defining Y axis

Z - general Z displacement coordinate

$\underset{\sim}{Z}$ - unit vector defining Z axis

a - acceleration

a_i - polynomial coefficient

a_i - the common normal between the axis of the ith joint and the axis of joint i+1

b_i - polynomial coefficient

b - differential equation coefficient

$\underset{\sim}{b}$ - vector of motion segment boundary conditions

c_i - polynomial coefficient

c - cosine

$\underset{\sim}{c}$ - vector of polynomial coefficients

d - displacement

d_i - polynomial coefficient

d_i - prismatic joint variable; the distance between lines a_i and a_{i-1} measured along the axis of the ith joint

d.o.m - degrees of mobility

d.o.f - degrees of freedom

e - error

$\underset{\sim}{e}$ - error vector

$\underset{\sim}{f}_i$ - vector of forces exerted on link i by link i-1

g - acceleration due to gravity

h - change factor in time scaling

i - impulse (first time derivative of acceleration)

j - jerk (first time derivative of impulse)

k - stiffness coefficient

m - link mass

$\underset{\sim}{n}_i$ - vector of moments exerted on link i by link i-1

$\underset{\sim}{p}_i$ - position vector of the origin of link i coordinates

q - general joint axis variable

$\underset{\sim}{q}$ - vector of joint coordinates, specific to robot. values comprised of θ_i, d_i

$\underset{\sim}{r}$ - general motion command vector, corresponds to the demand version of $\underset{\sim}{q}$

$\underset{\sim}{r}_i$ - position vector of link i centre of gravity

s - Laplace variable

s - sine

$\underset{\sim}{s}_i$ - position vector of link i centre of gravity, from the origin of the links coordinate frame

s_{ij} - the jth element of vector $\underset{\sim}{s}_i$

t - time

u - inertial axis frame; relative axis displacements

$\underset{\sim}{u}$ - control input vector

v - velocity

$\underset{\sim}{v}$ - MRACS equivalent feedback system linear signal component

w - circular harmonic frequency

$\underset{\sim}{w}$ - MRACS adaptor non-linear time varying part; output signal

$\underset{\sim}{x}$ - state vector

$\underset{\sim}{x}$ - vector of world (Cartesian) coordinates

$\underset{\sim}{y}$ - system output (response) vector

z - z transform variable

$\underset{\sim}{z}$ - transfer matrix of the equivalent linear part of the MRACS adaptor

α_i - the angle between axis i+1 and axis i

β - rotation angle (in world space orientation matrix)

$\underset{\sim}{\Gamma}$ - MRACS gain matrices

π - Pi

π - series multiplication operator

Σ - series summation operator

σ - second order coefficients of MRACS model

μ - general purpose interpolation or weighting variable

τ - dummy time variable

$\underset{\sim}{\tau}$ - vector of joint torques

$\underset{\sim}{\Phi}$ - non-linear time varying MRACS adaptor functions

θ - revolute joint variable; the angle of rotation of the line a_i relative to the line a_{i-1} about the axis of the ith joint

$\underset{\sim}{\Omega}_i$ - vector of angular velocities of link i

∂ - partial derivative operator

δt - sample interval time

∞ - infinity

$\underset{\sim}{\phi}$ - non-linear time varying MRACS adaptor functions

ϵ - the natural number; 2.718.....

$\underset{\sim}{\Omega}$ - MRACS gain matrices

ζ - second order damping ratio

1 Introduction

1.1 Robotics

1.1.1 System Description

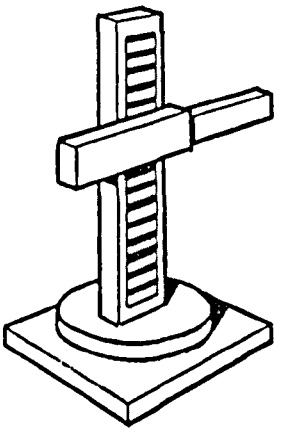
'Robot' as a label is loosely applied to single and multi-axis machines with axes driven between endstops either continuously or discretely. In the context of this work it is taken to mean programmable, multi-axis, servo-controlled machines. The industrial robot normally possesses up to 6 degrees of freedom. Six degrees of freedom satisfy the conditions required for arbitrary location and orientation of a body.

The links are most often in a serial chain, the joints being single R-revolute (rotary) or P-prismatic (linear slide) joints. Compound joints (eg. spherical or screw) are rarely used. The gross axis motions are provided by sets of P-P-P (gantry or Cartesian); R-P-P (cylindrical); R-R-P (polar); R-R-R (all revolute or SCARA) jointed links (figure 1.1), with the first of these axes connected to ground. The fine wrist motions are commonly R-R-R, although robots of the SCARA class are often just P or R-P.

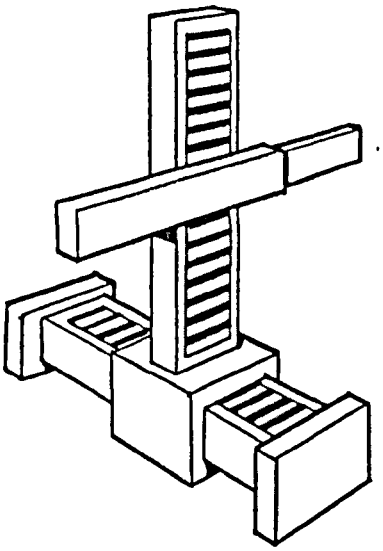
Actuation may be hydraulic or electric. Pneumatic systems are used but normally require brake systems in 'servo' applications to cope with gas compressibility. They are also limited to low capacities because of low operating pressures.

Control computer functions include feedback control sections and the motion programming and generation system. Manufacturers' feedback control techniques are generally error proportional with other simple compensation devices. These often provide stability at the expense of impaired manipulator performance.

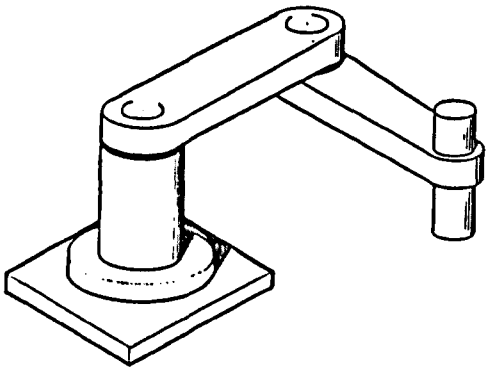
Motion programming comprises one or more of: teaching by leading through a task, input of data from a keyboard, or in some cases, off-line computer motion generation. The nominal trajectory is implemented as a series of spaced data sets. Operator control may be exercised over the path shape in space, but rarely over the time coordination.



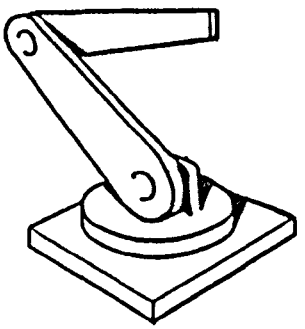
R-P-P Cylindrical



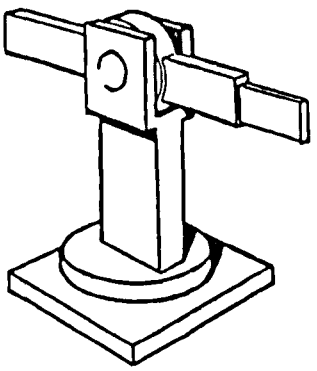
P-P-P Cartesian



R-R-R Revolute



R-R-R Revolute



R-R-P Polar

Figure 1.1 Gross Motion Axes of Robots

The manipulator used in the experimental work of this project, the Hall Automation 'Little Giant' has four axes R-R-P-R. The first three R-R-P are utilised in this project, being referred to as swing, vertical and horizontal, repectively. This machine is sketched in figure 1.2. It is expected that results obtained from this configuration can be extrapolated to give insight into applications on other manipulator configurations, with more degrees of freedom and more actuators.

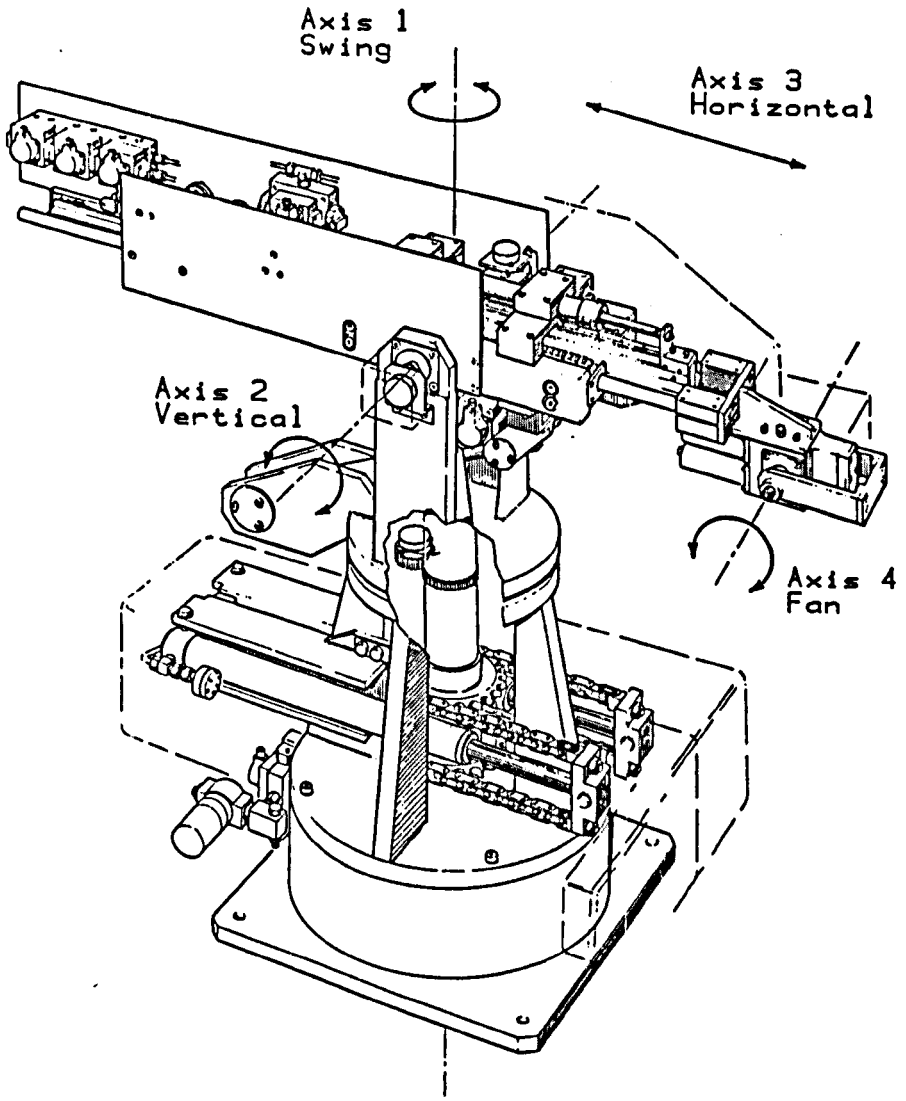


Figure 1.2 The Hall, Automation 'Little Giant'

The Little Giant transfer robot system includes a hydraulically driven articulated arm with associated hydraulic power pack. Its 'HAL system

90' control console is a dedicated point to point programmable controller. It is therefore not suitable for this project. In its place a far more versatile microcomputer and programmable interface has been developed for the purpose herein.

1.1.2 Robotics Research

Robotics is an exciting research area, as it develops and taxes engineering capabilities in many fields. The research fields include :

Robot Modelling and Simulation	Design of Robots Dynamics Control Performance Assessment Trajectory Generation
Workplace Modelling and Simulation	Design of Workplace Feasibility Assessment Collision Avoidance Solid Body Modelling Task Classification
Workplace Sensing	Vision: Design of Vision Systems Image Processing Parts Identification etc. Stereo Imaging Quality Control Tactile Sensors Seam Tracking
'Intelligent' Systems	Autonomous Robots High Level Languages Off Line Programming
Applications Development	Welding Polishing Industrial Trucks Materials Handling

	FMS
	Assembly
	Inspection
	Hazardous Environments
	Product Decoration
	Component Finishing
	Cutting
Robot Development	Internal Sensors
	Structure
	Actuators and Drives
	Performance Assessment
	Calibration
	Gripper/End Effector
	Reliability

Most of these are inter-related. Some of the headings and sub-headings can be interchanged. The table serves as an indicator of the size of the field of robotics. The work of this thesis lies within the dynamics, control and trajectory generation headings.

1.2 Manipulator Trajectories

1.2.1 Trajectory Types

It is assumed that motions may be determined by one of two desired operations, either :

(i) Process - The trajectory and often its time coordination are determined entirely by the task being undertaken. These may be pre-programmed or sensor driven motions. Process operations may also be classed as productive. Examples include component assembly, conveyor tracking, spray painting and arc welding. The nature of these operations means that the coordinate frame most generally suitable is Cartesian and located in the task itself, the end effector or some ground datum. Motions are defined either entirely by teach through learning or as Cartesian quantities defined as functions of time. The latter generally lead to straight line, circular or other regular shaped motions in space.

Process operations generally require higher tracking accuracy, so one objective of this work is to improve the robot's dynamic tracking accuracy, particularly when operating at high speed.

(ii) Move - The trajectory end conditions are defined, but between these end conditions, the type of motion and its time coordination is immaterial. The constraints placed on the motion are the avoidance of obstacles and the satisfactory execution of the move. Satisfactory execution will be assessed by the tracking accuracy and duration of the motion. There is therefore a greater degree of flexibility in a move operation which can be utilised in the improvement of motion generation schemes. Depending on the application, a large proportion of the total duration of a robot's work cycle may be made up of simply moving from one location to the next. This time adds to the cost of production, without adding anything to the intrinsic value of the product. Another objective of this project is therefore to improve the effective operating speeds of the robot.

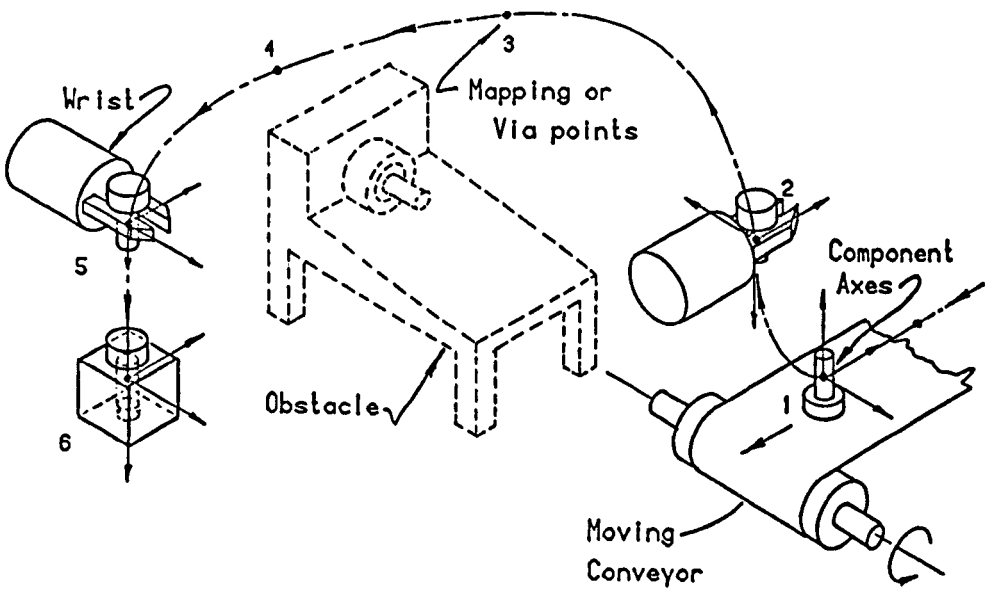


Figure 1.3 An Example Robot Cycle and Trajectory

Part of a robot cycle including both types of motion, is shown in figure 1.3. Process defined trajectories are typified by the regions 1-2 and 5-6. The move corresponds to the region 2-5, and can be carried out as fast as saturation and vibration limitations allow.

Both types of operation can be produced by robots having so-called Point To Point or Continuous Path controller units. The Point to Point does not have any tracking capability, it can only locate some prescribed configuration. The Point to Point motions are effectively continuous, otherwise the motions would be extremely violent, but the path and its time coordination cannot be operator controlled, outside of simple speed control. Methods used for generating the path between endpoints include interpolators and normalised tables. The Continuous Path control unit, more suited to seam welding, spray painting and the like, servos along either a taught path or a program defined path.

1.2.2 Trajectory Planning - Task Specification

Complete trajectory planning consists of many operations, including :

(i) Task Specification, defining motion constraints and boundary conditions :

- CAE System input; solid body models of task, tool, robot, obstacles
- teachbox, keyboard, lead through teaching
- sensor input; vision, tactile, sonar etc.
- synchronisation to external events
- communication with external devices

(ii) Motion Computation :

- motion law coefficient solutions
- motion law displacement data computation
- kinematic transformations
- scaling to transducers
- motion feasibility checking

(iii) Feedback Control :

- transducer input
- actuator output
- control algorithm computation
- safety checking and supervision

Complete robot supervisory and control functions include system supervision, giving communication with other computers;

synchronisation to other processes; diagnostics; management and operator information; low level sensor input; sequence control output; safety monitoring; trajectory specification etc. Of specific interest to this project is trajectory specification, computation and control. Trajectory and task specification is increasingly being carried out via computer languages.

There are some thirty plus, existing robot languages or computer languages for automation. Some of these require a compiler to convert the code as written by the programmer into an intermediate code which is then interpreted by the controller during run time. Others are completely interpreted during run time.

1.2.3 Levels of Motion Generating Languages

The four basic levels of such languages can be illustrated as :

(i) Objective Level - The ultimate level, in which a task and all the subsequent operations can be defined by simply stating the final objective. Other than for very simple objectives or dedicated tasks, these languages are not yet feasible.

(ii) Object Level - Here the task is specified, but it is limited to the description of the sequence of operations required to perform it. Whilst still at a relatively high level, it is now beginning to be feasible for a restricted set of tasks.

(iii) End Effector Level - Many available robot languages are at this level. It is little more than actuator level but defines the end effector motions, hence requiring the addition of a kinematic solution.

(iv) Actuator Level - At this level, individual axes are driven to specified positions. This is still a higher level than 'teach by lead through' or step by step teach programming. One line of a program, might read :

MOVE 39.65, 91.23, 173.88, 488.75, 177.23, 505.25

where the numbers represent the joint angles or displacements required to achieve a specified position in space. The starting

point is implicitly the set of joint values defining the point where the robot has been left from any earlier command. The motion used to get to the point defined by the above MOVE is coordinated.

1.2.4 Application Levels of Languages

The language requirements for automation go far beyond those of individual machines or robots. Integration of production systems requires the use of much more powerful languages, with hierarchical structures. The levels at which such languages might be applied, include :

Level	1	Single Robot Application
	2	Automation Cell or Group
	3	Combination of a number of Cells
	4	Combination of Automated Materials Feed & Storage with Automation Cells
	5	Integration of Overall Planning & Control Functions
	6	Integration of CAD Based Design Functions

Of the types of programming language used, for the purposes of this project, only the end effector and actuator levels are used, being applied to a 'level 1' Single Robot Application.

1.3 Increasing Trajectory Tracking Accuracy

As the operating speeds of robots increase, there is a need to put more effort into systems which maintain the required dynamic tracking accuracy, particularly in 'process' operations. The product of motion design is a nominal motion. How the robot system responds to it is another question. The modelling of robot dynamics indicates that the motions of the axes are defined by non-linear coupled functions of the torque or force input to the individual axes, including inertia, Coriolis, centripetal and gravitational terms. Many additional terms are required to completely model a robot, the above functions are simply the components of a classic rigid link robot model. It is easy to demonstrate that the duration and smoothness of a motion influence

the vibratory response of a manipulator. It is a far more difficult task to design motions which do not excite vibrations, using dynamic models.

The input to the axis servo drive is a function of both the demand displacement profile and the controller form. The output from the axis is a response displacement profile. There is scope to modify either the controller form or the demand profile, or both. (The controller can include both feedforward and feedback terms). The feedforward terms typically add derivatives of the demand profile to the input. Some cancellation or enhancement of the controlled system poles and zeros can be accomplished, nominally increasing performance. In the robot case, the complexity of the demand displacement profiles and the variation of system dynamics with time are such as to negate the advantages of fixed feedforward terms.

Modifying the demand profile can be carried out in an off line manner, reducing the real time computational load. Such modification can only compensate for the repeatable elements within the dynamics. These schemes cannot cater for random disturbances. It becomes clear that both feedback control and demand profile modification must be used together to achieve the optimum performance.

1.4 Increasing Trajectory Velocities

Production costs consist of two basic elements, so called fixed and variable costs. At the level of an automated cell, the fixed cost is the cost of machinery plus the constant overheads. The variable cost is that cost associated with the production of each element. The proportion of fixed costs transferred to each component is proportional to the time taken to make each component. If the fixed cost is a prominent part of the overall component costs, then reducing the cycle time will bring about an equivalent reduction in the component cost.

By design, handling devices such as robots need to be lightweight and dextrous, to achieve the large displacement ranges required and to move the workpiece into the required location. The moment arms and the loads will therefore be large. Most robot arms' link chains, (other

than the parallel linkage devices), are cantilevered out from their base links. They will be more compliant therefore, than devices such as machine tools.

The increased energy input to the system by virtue of the increased speeds in general results in an increased level of vibration in a compliant system. It is possible to diminish this effect by :

(i) increasing manipulator component stiffnesses
(ii) removing out of balance forces
(iii) reducing component masses
(iv) generating damping forces either actively or passively
(v) controlling the time history of the energy input

The first three entail design modifications to the manipulator and are outside the project scope. Structural vibrations, once excited in the robot, will occur in many different modes and degrees of freedom. The number of freedoms is very large, and due to the continual change in configuration, energy will be transferred from one mode to another. Sensor information is only available at the joints. Hence there are inadequate transducers and actuators to cope with the fourth option. The fifth option requires a comprehensive dynamic model of the robot and some form of energy control. Instead of considering energy directly, it is logical to utilise displacement control. The promise of carefully tailored displacement paths in minimising vibrations has been shown in numerous cams works in the past. Its application to the robot is found to be considerably more complex.

The approach presented in this project utilises the fact that any programmable industrial robot possesses the necessary elements of a self contained experimental system. It includes a strategy in which results from a previous run are processed and used to update the command trajectory. The objective of the work (introduced in chapter 8) is to reduce the non-productive time taken up by robot 'move' operations without incurring excessive dynamic disturbances or loss of co-ordination.

1.5 Thesis Structure

In chapter 2, kinematics of robots is developed. Initial derivations are relatively general, being for six degree of freedom robots. The robot used within the subsequent experimental work was a 'Hall Automation Little Giant'. The three gross motion axes were used and so the appropriate kinematic solutions are presented. The forward kinematics includes solutions for displacement, velocity and acceleration; for use with the dynamic modelling of chapter 3. Inverse solutions are derived for displacement and velocity; being required for use in the tuning method of chapter 7.

The dynamics and control of manipulators is discussed in chapter 3. For idealised motion control or tuning systems, full dynamic models would be used. In order to assess the feasibility of such methods, it is necessary to carry out a study of robot dynamic modelling. The robot dynamic modelling is used in the development of motion tuning in chapter 5, the self learning scheme of chapter 6 and the controller in chapter 7. On the basis of a control survey, an adaptive control method is selected, to be used later in chapter 7.

A study of motion generation is carried out in chapter 4, from which mathematical models of motion are specified. The elements required in the manual and automatic generation of motions are described. Schemes for achieving these elements are derived for use in chapters 6,7 and 8.

The concept of trajectory tuning, used to improve the manipulator trajectory tracking performance is pursued in chapter 5. Dynamic model based methods and self learning techniques for tuning are considered. Using dynamic models requires knowledge of the manipulators dynamic parameters. Some aspects of dynamic parameter measurement are therefore described. The complexity of the dynamic model necessitates development of simplified models. These simple models are then used in the tuning of basic motion laws. Self learning techniques are outlined, to be developed further in chapters 6 and 8. Selection of motion laws, particularly in the case of increased velocities is described.

The self learning scheme for increasing trajectory tracking accuracy is derived in chapter 6. As it is a semi-empirical technique, attention is focussed onto some of the practical considerations. It is then implemented on both single and multi axis systems.

Chapter 7 sees the application of a Model Referenced Adaptive Control Scheme to the tuning problem. Its inherent property of simplifying the robot dynamic behaviour is exploited in the tuning of motion laws.

A scheme for optimising the motion duration or trajectory velocity is developed in chapter 8. A self learning scheme capable of pushing the robot to near its saturation limits is defined and tested without recourse to dynamic modelling. Much of the effort is directed into locating a suitable saturation quantifying parameter.

2 Manipulator Kinematics

The position and orientation of a rigid body located arbitrarily in three dimensional space requires the specification of six coordinates. Conventionally these coordinates are referred to a right handed Cartesian axis frame, and are comprised of three linear and three angular displacements. A robot design must therefore also possess six degrees of freedom to attain arbitrary location of a rigid body.

The natural position coordinate set of a task and a robot are quite different. It is currently convenient to define a task in terms of Cartesian displacement and Euler angular coordinates. The datum of these coordinates may be taken as the task or some reference in the workspace or some component part of the robot. Robots are generally controlled at or close to the joints. The robot's natural coordinates are then a set of scaled transducer values which correspond to the joint linear or rotary displacements. Their datum is the robot base.

It is necessary to functionally relate some of these, interdependent coordinate sets. Generally speaking, a set of workspace or 'world' (Cartesian) displacements can be defined as :

$$\underset{\sim}{x} = \underset{\sim}{F}(\underset{\sim}{q}) \qquad \text{(forward kinematics)} \qquad (2.1)$$

the joint displacement :

$$\underset{\sim}{q} = \underset{\sim}{F}^{-1}(\underset{\sim}{x}) \qquad \text{(inverse kinematics)} \qquad (2.2)$$

where $\underset{\sim}{x}$ - vector of world coordinates

$\underset{\sim}{F}$ - some generalised vector function

$\underset{\sim}{q}$ - vector of joint coordinates

The forward kinematics is the fundamental one, and there is a one-one dependency. This is not the case with the inverse kinematics, where many solutions may exist. Serious numerical difficulties include this multiplicity of solutions, and the presence of singularities. Experience has shown that many of these problems disappear in the simpler configuration robots, particularly where no dimensional offsets occur, or if joint displacements are of relatively limited range.

2.1 Manipulator Specific Kinematics

A kinematic analysis of the manipulator is required in this project :

(i) to compute numerous kinematic variables which form the basis of the manipulator's dynamic model. Functions are required, relating the motions of the joints to the motions of the end effector, expressed in terms of a fixed Cartesian axis frame. Motion in terms of this frame can be referred to as being in 'world space'. Functions are also required to relate motions of the actuators to motions of the joints, or the 'joint space'.

(ii) for positioning, control and trajectory purposes. If a trajectory or point is computed in terms of the world space, then it must be kinematically transformed to joint coordinates or rather those of the transducer, in order to be used as a control command.

Efficiency of the kinematic solution is important. The computations can be lengthy and may determine the sampling interval of the discrete time controller, in turn degrading the performance of the feedback control.

General manipulator kinematic solutions for joint varibles given the world space variables are highly complex, Duffy 1980 [2.1]. Most manipulators are special cases and many of the common ones can be 'slotted' into a small number of variations on relatively simple configurations. The configuration chosen for analysis is assumed to possess three revolute wrist joints having co-intersecting axes. The inverse kinematic solution is greatly simplified by making this choice, Pieper 1968 [2.2]. Additionally this has the effect of giving the three joint motions nearest the base, domination of the major displacements. These three joints can therefore be termed the 'gross motion axes'. The wrist axes then control the orientation of the end effector and can nominally attain any desired orientation of the wrist.

Solutions for the joint variables are found to be conveniently split into three sections. These are solutions for the wrist point; the

gross motion axis variables; and finally determination of the wrist joint variables.

2.2 Manipulator System Kinematic Specification: General Preliminaries.

2.2.1 Degrees of Freedom

The maximum number of degrees of freedom (d.o.f.) a rigid body can possess is six. In the case of a manipulator, the number of degrees of freedom it possesses can be taken as the minimum number of coordinates required to specify uniquely the position and orientation of the end effector. Manipulator joints are made up of a pair of contact surfaces. These surfaces are usually thought of as lower pairs, although they invariably incorporate roller or ball bearings. A lower pair is comprised of two mating surfaces, mutually in contact and sliding with respect to one another. Of the six practical lower pairs, only two are normally used in manipulators. These are the revolute pair (R) - allowing rotation about its axis, and the prismatic pair (P) - enabling a translation along its axis. The number of independently driven joints can be termed the degree of mobility (d.o.m.) of the manipulator, to differentiate it from the number of d.o.f. of the end effector. Note that the d.o.f. ≤ 6, also the d.o.m. $\geq$ d.o.f.

2.2.2 World Coordinates (Grounded Set)

Fixed in relation to the ground, a right handed Cartesian axis frame defines the environment or 'world' set of coordinates, see figure 2.1. The Z_0 axis points vertically upwards. The complete world space vector will be comprised of three translational and three rotational elements, yet to be specified.

2.2.3 Joint Coordinates

The manipulator is primarily an open-chain mechanism comprised of m+1 rigid links. The ith joint lies between link i and link i-1. The base or ground link is link o, and the end effector, link m. Each link has embedded in it a right handed Cartesian axis frame. This is generally located at one of the link ends and has one of its axes (Z_i) co-axial

with that of the joint. The mass properties of each link are referenced to this axis frame.

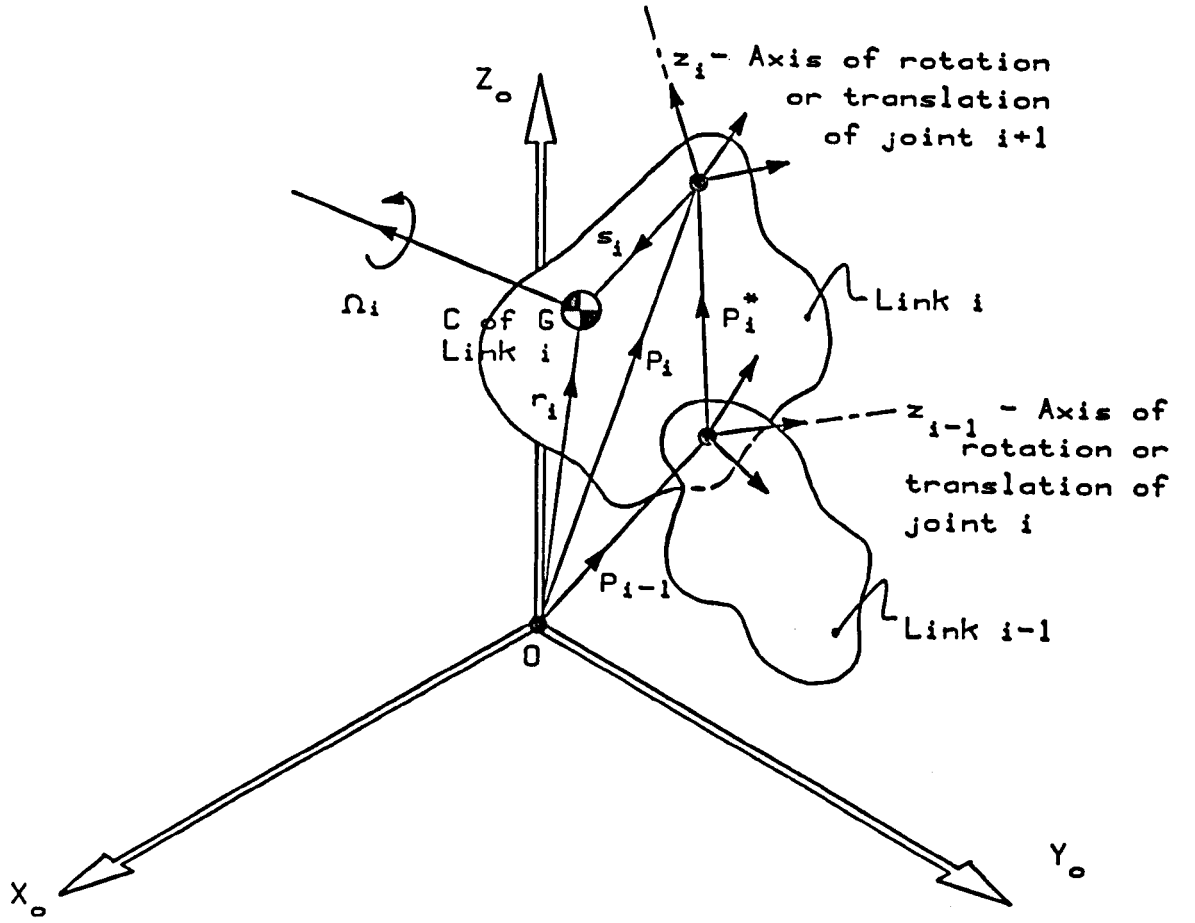


Figure 2.1 Link and Axis Kinematic Parameters

There is only one joint between each link and it possesses only a single degree of mobility, i.e. a pure rotation or translation. The magnitude of the mobility displacement from some datum position can therefore be specified as a single scalar quantity. The variable associated with this quantity becomes one of the elements of the joint space vector. The joint space vector will therefore possess the same number of elements as there are independent joint motions in the manipulator.

It is convenient to employ the widely used link arrangement, shown in figure 2.2 of Denavit and Hartenberg 1955 [2.3]. At most, two rotations and two translations are required to bring one link axis frame origin coincident with that of an adjacent frame. The notation used in the diagram is as follows :

θ_i - the angle of rotation of the line a_i relative to the line a_{i-1} about the axis of the ith joint

a_i - the length of the common normal between the axis of the ith joint and the axis of joint i+1

d_i - the distance between lines a_i and a_{i-1} measured along the axis of the ith joint

α_i - the angle between axis i+1 and axis i

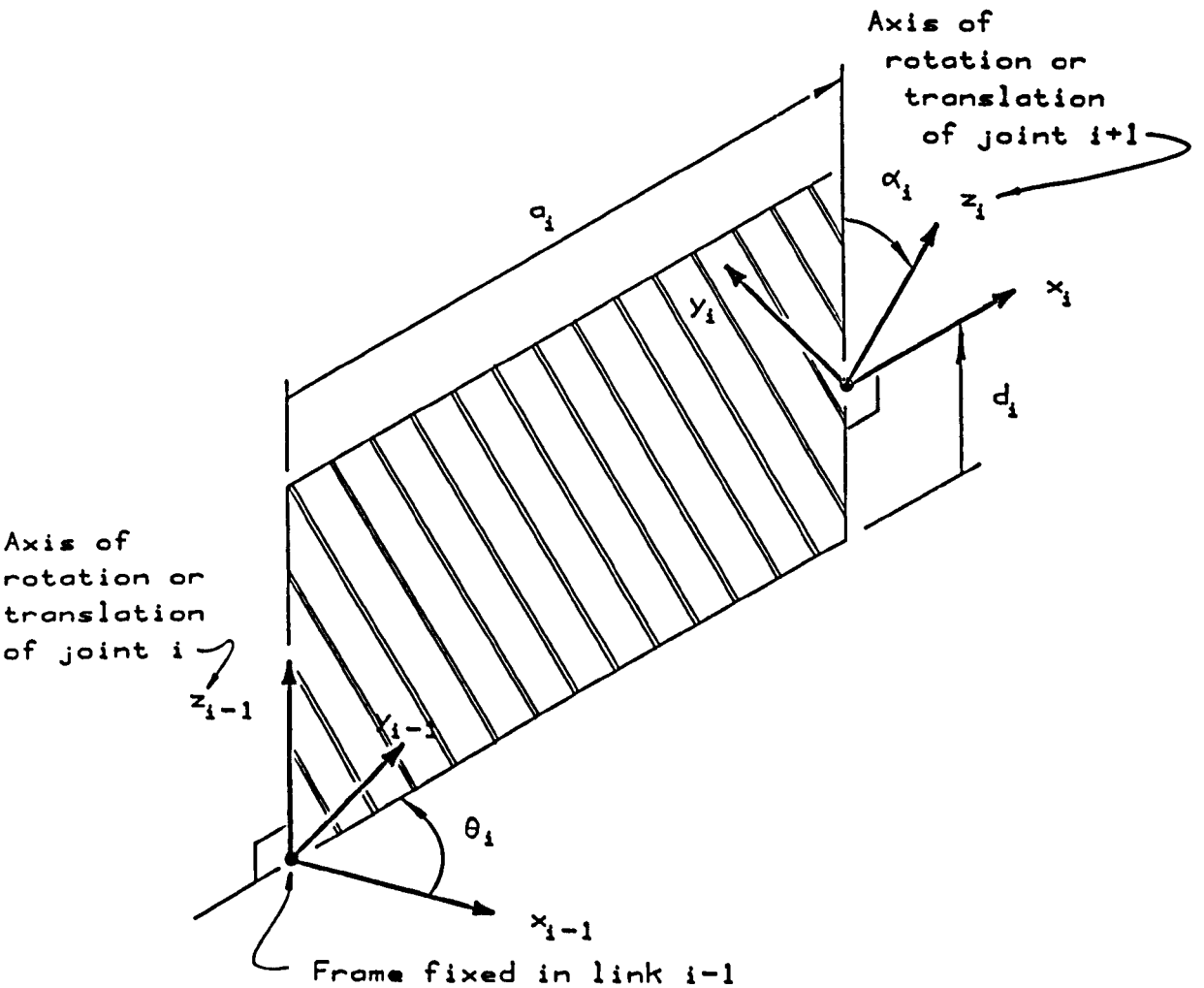


Figure 2.2 Denavit and Hartenberg Link Notation

For revolute joints θ_i is the variable, a_i, d_i and α_i are constants. For prismatic joints d_i is the variable, a_i, α_i and θ_i are constants. Depending on the manipulator configuration, some of the displacement elements which are constants; a_i or d_i may be zero and θ_i or α_i may be zero or $\pi/2$ for example. These conditions, which correspond to co-intersection or coincidence of axes, lead to simplification of the analysis.

2.3 Specification of Orientations and Translations

2.3.1 World Orientations

A method of specifying the orientation of the end effector and its intermediary links, relative to a grounded or datum set of axes is needed. The displacements of successive orthogonal rotations about sets of Cartesian axes are termed 'Euler angles' and are used to specify the orientation of a body. There are many possible sequences of rotations, but here attention is restricted to the ZXZ sequence of figure 2.3.

These result in rotation matrices relating the Cartesian components of a vector in one coordinate system to their resolved components in another. These are algebraically equivalent to

$$\underset{\sim}{Ro^6} = \begin{bmatrix} c\beta_3 & -s\beta_3 & 0 \\ s\beta_3 & c\beta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\beta_2 & -s\beta_2 \\ 0 & s\beta_2 & c\beta_2 \end{bmatrix} \begin{bmatrix} c\beta_1 & -s\beta_1 & 0 \\ s\beta_1 & c\beta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.3)$$

Rotn. of β_3 about the Z axis — Rotn. of β_2 about the new X axis — Rotn. of β_1 about the new Z axis

where

$\underset{\sim}{Ro^6}$ - the orientation transformation or direction cosine matrix of link 6 (the end effector) referenced to the grounded 'o' set of axes.

β_i - the rotation angle

$c\beta_i$, $s\beta_i$ - the sine and cosine of β_i

Multiplying out :

$$\underset{\sim}{Ro^6} = \begin{bmatrix} c\beta_1 c\beta_3 - s\beta_1 c\beta_2 s\beta_3 & -s\beta_1 c\beta_3 - c\beta_1 c\beta_2 s\beta_3 & s\beta_2 s\beta_3 \\ c\beta_1 s\beta_3 + s\beta_1 c\beta_2 c\beta_3 & c\beta_1 c\beta_2 c\beta_3 - s\beta_1 s\beta_3 & -s\beta_2 c\beta_3 \\ s\beta_1 s\beta_2 & c\beta_1 s\beta_2 & c\beta_2 \end{bmatrix} \qquad (2.4)$$

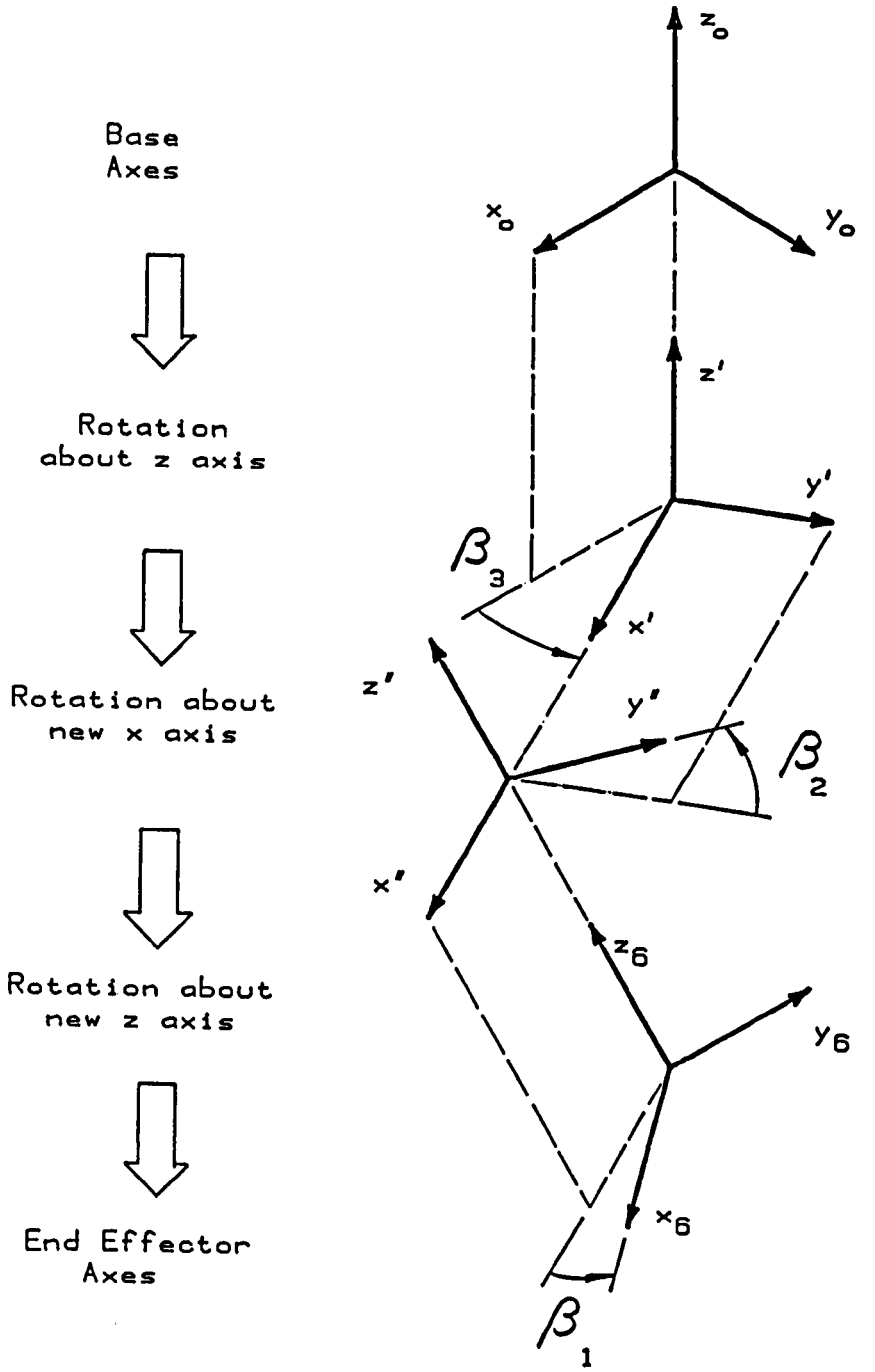


Figure 2.3 Successive Rotations Required to Specify the World Space Orientation of the End Effector
*(Note, all axes have the same origin and are shown separated for clarity only)

2.3.2 Link i-1 to Link i Orientation Transformations

Using the link axis frame definition of figure 2.2, relative link orientation transformation matrices may be obtained. These correspond

to rotations of :

(i) α_i about the $\underset{\sim}{X}_i$ axis

(ii) θ_i about the $\underset{\sim}{Z}_{i-1}$ axis

Resulting in :

$$\underset{\sim}{R}_{i-1}^{i} = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i \\ 0 & s\alpha_i & c\alpha_i \end{bmatrix} \qquad (2.5)$$

where

$\underset{\sim}{R}_{i-1}^{i}$ - the direction cosine matrix of link i referenced to link i-1. The components of a vector in one links frame can therefore be expressed in terms of another, by multiplying the vector by the appropriate matrix.

The elements of this matrix are termed the direction cosines of link i coordinates referenced to link i-1 coordinates, indicating link i orientation. Successive multiplication of these matrices can be used to reference any one link frame to any other, including of course the datum or grounded set of coordinate axes, e.g. :

$$\underset{\sim}{R}_{o}^{i} = \underset{\sim}{R}_{o}^{1}\ \underset{\sim}{R}_{1}^{2} \ \ldots\ \underset{\sim}{R}_{i-1}^{i} \qquad (2.6)$$

Also because the coordinate systems are ortho-normal :

$$\underset{\sim}{R}_{i}^{o} = [\underset{\sim}{R}_{o}^{i}]^{-1} = [\underset{\sim}{R}_{o}^{i}]^{T} \qquad (2.7)$$

2.3.3 World to Joint Axis Origin Locations

In terms of the grounded (world) set of axes, three linear displacement elements can be used to specify the location of the origin of any of the link axis frames. From figure 2.1, the vector defining the location of the origin of link i coordinate axes referenced to the ground frame is :

$$\underset{\sim}{p}_{i}^{o} = \begin{bmatrix} X_i^o \\ Y_i^o \\ Z_i^o \end{bmatrix} \qquad (2.8)$$

2.3.4 Link i-1 to Link i Axis Origin Locations

Again, using the previously defined link axis frames and their inter-relationships, a vector can be used to describe the location of the link i origin relative to the origin of link i-1, referenced to the link i coordinates :

$$\underset{\sim}{p}_{i}^{i*} = \begin{bmatrix} a_i \\ d_i s\alpha_i \\ d_i c\alpha_i \end{bmatrix} \qquad (2.9)$$

Note in using the notation of figure 2.1, there is the following

relationship between the vectors :

$$\underset{\sim}{}^{o}p_i = \underset{\sim}{}^{o}p_{i-1} + \underset{\sim}{}^{o}p_i^* \qquad (2.10)$$

where

$$\underset{\sim}{}^{o}p_i^* = \underset{\sim}{R}_o^i \; \underset{\sim}{}^{i}p_i^* \qquad (2.11)$$

2.3.5 Homogeneous Transformations

Denavit and Hartenberg noted that both the orientational and translational components can be integrated into a single 4 x 4 matrix. Successive multiplication of these matrices relates both these components from one axis to any other. The form of these partitioned matrices is as follows :

$$\left[\begin{array}{c|c} \underset{\sim}{R}_{i-1}^{i} & \underset{\sim}{}^{i}p_i^* \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right] \qquad (2.12)$$

This denotes the complete rotation and translation transformation between coordinate frames i-1 and i. Notation appears simplified but in the general case, successive multiplications generate superfluous algebra. Much of this algebra can be eliminated with a careful analysis of any specific manipulator. At the end of the day the loop equations can only be solved numerically, unless all the algebra is produced, which defeats the object of such a notation. Homogeneous transformations will therefore not be used in the sequel.

2.4 Spatial Motion - Link Kinematics

2.4.1 The General Case

Spatial motion requires more complex analysis than that of planar motion, because the direction of the angular velocity vector varies. Referring to figures 2.1 and 2.2, the motion of the coordinate frame relative to the base or ground, also its motion relative to the frame of the supporting link, are of interest. The analysis of the link centre of mass proves beneficial for later use in the dynamics. The linear position, velocity and acceleration relationships of the axis frames can be shown to be :

$$\underset{\sim}{p}_i = \underset{\sim}{p}_{i-1} + \underset{\sim}{p}_i^* \qquad (2.13)$$

$$\dot{\underset{\sim}{p}}_i = \dot{\underset{\sim}{p}}_{i-1} + \dot{\underset{\sim}{p}}_i^* + \underset{\sim}{\Omega}_i \times \underset{\sim}{p}_i^* \qquad (2.14)$$

$$\ddot{\underset{\sim}{p}}_i = \ddot{\underset{\sim}{p}}_{i-1} + \ddot{\underset{\sim}{p}}_i^* + \dot{\underset{\sim}{\Omega}}_i \times \underset{\sim}{p}_i^* + 2\,\underset{\sim}{\Omega}_i \times \dot{\underset{\sim}{p}}_i^* + \underset{\sim}{\Omega}_i \times (\underset{\sim}{\Omega}_i \times \underset{\sim}{p}_i^*) \qquad (2.15)$$

Similarly, the angular velocity relationships are :

$$\underset{\sim}{\Omega}_i = \underset{\sim}{\Omega}_{i-1} + \underset{\sim}{\Omega}_i^* \qquad (2.16)$$

$$\dot{\underset{\sim}{\Omega}}_i = \dot{\underset{\sim}{\Omega}}_{i-1} + \dot{\underset{\sim}{\Omega}}_i^* + \underset{\sim}{\Omega}_{i-1} \times \underset{\sim}{\Omega}_i^* \qquad (2.17)$$

And the centre of gravity relationships are :

$$\underset{\sim}{r}_i = \underset{\sim}{p}_i + \underset{\sim}{s}_i \qquad (2.18)$$

$$\dot{\underset{\sim}{r}}_i = \dot{\underset{\sim}{p}}_i + \dot{\underset{\sim}{s}}_i + \underset{\sim}{\Omega}_i \times \underset{\sim}{s}_i \qquad (2.19)$$

$$\ddot{\underset{\sim}{r}}_i = \ddot{\underset{\sim}{p}}_i + \ddot{\underset{\sim}{s}}_i + \dot{\underset{\sim}{\Omega}}_i \times \underset{\sim}{s}_i + 2\,\underset{\sim}{\Omega}_i \times \dot{\underset{\sim}{s}}_i + \underset{\sim}{\Omega}_i \times (\underset{\sim}{\Omega}_i \times \underset{\sim}{s}_i) \qquad (2.20)$$

where $\underset{\sim}{p}_i$ - position vector of the origin of link i coordinate frame

$\underset{\sim}{\Omega}_i$ - angular velocity of link i

$\underset{\sim}{r}_i$ - position vector of link centre of mass relative to the ground frame origin

$\underset{\sim}{s}_i$ - position vector of link centre of mass relative to the link frame origin

In the above relationships, $\dot{\underset{\sim}{p}}_i^*$, $\ddot{\underset{\sim}{p}}_i^*$, $\dot{\underset{\sim}{s}}_i$, $\ddot{\underset{\sim}{s}}_i$, $\underset{\sim}{\Omega}_i^*$ and $\dot{\underset{\sim}{\Omega}}_i^*$ are the pure velocity and acceleration components in the direction of the original vector. In order to compute these equations numerically, the vectors in these relationships must all be referenced to the same coordinate axis frame. Using the orientation transformation matrices, relations can be obtained as follows, for example :

$$^i\underset{\sim}{p}_i^* = [\underset{\sim}{R}_o^i]^T\, {}^o\underset{\sim}{p}_i^* = \underset{\sim}{R}_i^o\, {}^o\underset{\sim}{p}_i^* = \begin{bmatrix} a_i \\ d_i s\alpha_i \\ d_i c\alpha_i \end{bmatrix} \qquad (2.21)$$

where the i and o prefixes indicate the axis frame into which the elements of the vector have been resolved. These equations are simplified for particular types of joint. Also, for a rigid manipulator :

$$^i\dot{\underset{\sim}{s}}_i = {}^i\ddot{\underset{\sim}{s}}_i = 0 \qquad (2.22)$$

2.4.2 Rotational or Revolute Axis

a_i, α_i and d_i are constant and θ_i is the joint variable. If the axis

is revolute and the link is rigid :

$$ {}^{i}\dot{\underset{\sim}{p}}_{i}^{*} = {}^{i}\ddot{\underset{\sim}{p}}_{i} = 0 \qquad (2.23) $$

Also the joint variable θ_i is a rotation about the $\underset{\sim}{Z}_{i-1}$ axis, hence :

$$ {}^{i-1}\underset{\sim}{\Omega}_{i}^{*} = {}^{i-1}\underset{\sim}{Z}_{i-1}\,\dot{\theta}_i \qquad (2.24) $$

similarly

$$ {}^{i-1}\dot{\underset{\sim}{\Omega}}_{i}^{*} = {}^{i-1}\underset{\sim}{Z}_{i-1}\,\ddot{\theta}_i \qquad (2.25) $$

Therefore equations 2.13 to 2.20 become :

$$ \underset{\sim}{p}_i = \underset{\sim}{p}_{i-1} + \underset{\sim}{p}_i^{*} \qquad (2.26) $$

$$ \dot{\underset{\sim}{p}}_i = \dot{\underset{\sim}{p}}_{i-1} + \underset{\sim}{\Omega}_i \times \underset{\sim}{p}_i^{*} \qquad (2.27) $$

$$ \ddot{\underset{\sim}{p}}_i = \ddot{\underset{\sim}{p}}_{i-1} + \dot{\underset{\sim}{\Omega}}_i \times \underset{\sim}{p}_i^{*} + \underset{\sim}{\Omega}_i \times (\underset{\sim}{\Omega}_i \times \underset{\sim}{p}_i^{*}) \qquad (2.28) $$

$$ \underset{\sim}{\Omega}_i = \underset{\sim}{\Omega}_{i-1} + \underset{\sim}{Z}_{i-1}\,\dot{\theta}_i \qquad (2.29) $$

$$ \dot{\underset{\sim}{\Omega}}_i = \dot{\underset{\sim}{\Omega}}_{i-1} + \underset{\sim}{Z}_{i-1}\,\ddot{\theta}_i + \underset{\sim}{\Omega}_{i-1} \times \underset{\sim}{Z}_{i-1}\,\dot{\theta}_i \qquad (2.30) $$

$$ \underset{\sim}{r}_i = \underset{\sim}{p}_i + \underset{\sim}{s}_i \qquad (2.31) $$

$$ \dot{\underset{\sim}{r}}_i = \dot{\underset{\sim}{p}}_i + \underset{\sim}{\Omega}_i \times \underset{\sim}{s}_i \qquad (2.32) $$

$$ \ddot{\underset{\sim}{r}}_i = \ddot{\underset{\sim}{p}}_i + \dot{\underset{\sim}{\Omega}}_i \times \underset{\sim}{s}_i + \underset{\sim}{\Omega}_i \times (\underset{\sim}{\Omega}_i \times \underset{\sim}{s}_i) \qquad (2.33) $$

Which can be referenced to any coordinate frame. The frame used must of course be consistent, for the equations to be compatible.

2.4.3 Translational or Prismatic Axis

Here, a_i, α_i, θ_i are constant and d_i is the joint variable, hence :

$$ {}^{i}\dot{\underset{\sim}{p}}_i \neq 0 \quad \text{and} \quad {}^{i}\ddot{\underset{\sim}{p}}_{i}^{*} \neq 0 \qquad (2.34) $$

Again, the joint variable is associated with the $\underset{\sim}{Z}_{i-1}$ axis hence :

$$ {}^{i}\dot{\underset{\sim}{p}}_{i}^{*} = \underset{\sim}{Z}_{i-1}\,\dot{d}_i \qquad (2.35) $$

and $$ {}^{i}\ddot{\underset{\sim}{p}}_{i}^{*} = \underset{\sim}{Z}_{i-1}\,\ddot{d}_i \qquad (2.36) $$

also because both α_i and θ_i are constants :

$$ \underset{\sim}{\Omega}_{i}^{*} = \dot{\underset{\sim}{\Omega}}_{i}^{*} = 0 \qquad (2.37) $$

In this case, equations 2.13 to 2.20 result in :

$$\underset{\sim}{p}_i = \underset{\sim}{p}_{i-1} + \underset{\sim}{p}_i^* \qquad (2.38)$$

$$\dot{\underset{\sim}{p}}_i = \dot{\underset{\sim}{p}}_{i-1} + \underset{\sim}{Z}_{i-1}\,\dot{d}_i + \underset{\sim}{\Omega}_i \times \underset{\sim}{p}_i^* \qquad (2.39)$$

$$\ddot{\underset{\sim}{p}}_i = \ddot{\underset{\sim}{p}}_{i-1} + \underset{\sim}{Z}_{i-1}\,\ddot{d}_i + \dot{\underset{\sim}{\Omega}}_i \times \underset{\sim}{p}_i^* + 2\,\underset{\sim}{\Omega}_i \times (\underset{\sim}{Z}_{i-1}\,\dot{d}_i) + \underset{\sim}{\Omega}_i \times (\underset{\sim}{\Omega}_i \times \underset{\sim}{p}_i^*) \qquad (2.40)$$

$$\underset{\sim}{\Omega}_i = \underset{\sim}{\Omega}_{i-1} \qquad (2.41)$$

$$\dot{\underset{\sim}{\Omega}}_i = \dot{\underset{\sim}{\Omega}}_{i-1} \qquad (2.42)$$

$$\underset{\sim}{r}_i = \underset{\sim}{p}_i + \underset{\sim}{s}_i \qquad (2.43)$$

$$\dot{\underset{\sim}{r}}_i = \dot{\underset{\sim}{p}}_i + \underset{\sim}{\Omega}_i \times \underset{\sim}{s}_i \qquad (2.44)$$

$$\ddot{\underset{\sim}{r}}_i = \ddot{\underset{\sim}{p}}_i + \dot{\underset{\sim}{\Omega}}_i \times \underset{\sim}{s}_i + \underset{\sim}{\Omega}_i \times (\underset{\sim}{\Omega}_i \times \underset{\sim}{s}_i) \qquad (2.45)$$

2.5 Six Degree of Freedom Manipulator System Kinematics

2.5.1 Manipulator Link Displacement Solutions

Two problems exist here. The first is often termed 'forward kinematics' and consists of obtaining the position and orientation of the manipulator end effector (the world space vector) given the values of the joint variables (the joint coordinate vector). This problem is numerically well behaved and gives a unique solution. The second problem is to find the joint coordinate vector given the world space vector - so called 'inverse kinematics'. In general a closed form solution is not available. Various numerical difficulties arise, necessitating the use of alternative, prescribed solutions at the associated problematical points.

2.5.2 Forward Kinematics

The position of the end effector in terms of the joint variables may be obtained by summing the link vectors ${}^0\underset{\sim}{p}_i^*$. From equations 2.10 and 2.11 :

$${}^0\underset{\sim}{p}_6 = \sum_{i=0}^{6} {}^0\underset{\sim}{p}_i^* \equiv \sum_{i=0}^{6} \underset{\sim}{R}_0^i \; {}^i\underset{\sim}{p}_i^* \qquad (2.46)$$

Improved efficiency of evaluation can be achieved by nesting, reducing the complexity of successive $\underset{\sim}{R}_0^i$ terms. At each step of the computation, the individual link axis frame is expressed in terms of

the base coordinates. The world space orientation matrix 2.4 may be equated to the joint space orientation matrix 2.6, evaluated over all the joint transformations. This procedure provides orientation equations in apparent excess. Selecting the 1,3; 3,2; and 3,3 elements for example gives three linearly independent equations which may be used to solve for the world space orientations. The system of equations are :

$$ {}^{o}\underset{\sim}{p}_{6} = \begin{bmatrix} {}^{o}X_6 \\ {}^{o}Y_6 \\ {}^{o}Z_6 \end{bmatrix} = \begin{bmatrix} {}^{o}X_6\,(q_1\,q_2\,q_3\,q_4\,q_5\,q_6) \\ {}^{o}Y_6\,(q_1\,q_2\,q_3\,q_4\,q_5\,q_6) \\ {}^{o}Z_6\,(q_1\,q_2\,q_3\,q_4\,q_5\,q_6) \end{bmatrix} \qquad (2.47) $$

$$ \begin{bmatrix} {}^{o}\beta_1 \\ {}^{o}\beta_2 \\ {}^{o}\beta_3 \end{bmatrix} = \begin{bmatrix} {}^{o}\beta_1\,(q_1\,q_2\,q_3\,q_4\,q_5\,q_6) \\ {}^{o}\beta_2\,(q_1\,q_2\,q_3\,q_4\,q_5\,q_6) \\ {}^{o}\beta_3\,(q_1\,q_2\,q_3\,q_4\,q_5\,q_6) \end{bmatrix} \qquad (2.48) $$

where the q_i are joint variables, either θ_i or d_i. These equations could be used as they stand for evaluation, but some simplification can be attained. Commonly, the three wrist axes are coincident, hence

$$ {}^{o}\underset{\sim}{p}_{5} = {}^{o}\underset{\sim}{p}_{3} \qquad (2.49) $$

where from equation 2.10 :

$$ {}^{o}\underset{\sim}{p}_{6} = {}^{o}\underset{\sim}{p}_{5} + {}^{o}\underset{\sim}{p}_{6}^{*} \qquad (2.50) $$

and equation 2.46 gives :

$$ {}^{o}\underset{\sim}{p}_{6}^{*} = \underset{\sim}{R}{}^{o}_{6}\ {}^{6}\underset{\sim}{p}_{6}^{*} \qquad (2.51) $$

If equations 2.48 are evaluated for β_1, β_2 and β_3 then $\underset{\sim}{R}{}^{o}_{6}$ can be expressed in terms of the β_i resulting in :

$$ {}^{o}\underset{\sim}{p}_{6} = {}^{o}\underset{\sim}{p}_{3} + \underset{\sim}{R}{}^{o}_{6}(\beta_1,\beta_2,\beta_3)\ {}^{6}\underset{\sim}{p}_{6}^{*} \qquad (2.52) $$

which gives the remaining three elements of the world space vector. This form is sometimes referred to as 'wrist partitioned kinematics'. Under certain circumstances, the solutions for the β_i break down, in which case it may be preferable to return to the full set of equations, selecting another set i.e. switching solutions at the breakdown point only. When equating terms of the two $\underset{\sim}{R}{}^{o}_{6}$ matrices, careful selection of terms can result in much simpler solutions, eliminating some of the q_i terms. Further improvements can be made by noting common terms in the expressions and pre-computing them.

2.5.3 Inverse Kinematics

Many real solutions are obtained, corresponding to the multiple closures of the robot mechanism, for the same end effector position and orientation. They may be numerically ill-conditioned, produce singularities, redundant equations and degeneracy. Some of these problems are eliminated by mechanical and other limitations on the manipulators reachable workspace. Limitations in themselves may create further problems in that the equations become discontinuous.

All of the equations are trigonometric functions of joint variables. A substitution is therefore often made for tangent half angles. In general, these turn equations once manipulated for solution, into high degree polynomials, beyond practical methods of solution, Pieper 1968 [2.2]. Because of this, Pieper concluded that 'the complete solution is not at this time technically feasible'. Duffy 1980 [2.1] using his 'Unified Theory' was able to obtain solutions. Solutions for such general cases are found to be extremely inefficient when most industrial robots are analysed. The configurations available, usually lead to appreciable simplification of the equations. As many of the relative joint offsets are zero, and wrist axes co-intersect, the wrist partitioned kinematics can be reversed, thus from 2.52 :

$$ {}^{0}\underset{\sim}{p}_{3} = {}^{0}\underset{\sim}{p}_{6} - \underset{\sim}{R}_{0}^{6}(\beta_1,\beta_2,\beta_3)\ {}^{6}\underset{\sim}{p}_{6}^{*} \qquad (2.53) $$

and from ${}^{0}\underset{\sim}{p}_{3}$, solutions may be obtained for q_1, q_2 and q_3 as equation 2.46 gives, in the nested form :

$$ {}^{0}\underset{\sim}{p}_{3} = \underset{\sim}{R}_{0}^{1}\ ({}^{1}\underset{\sim}{p}_{1}^{*} + \underset{\sim}{R}_{1}^{2}\ ({}^{2}\underset{\sim}{p}_{2}^{*} + \underset{\sim}{R}_{2}^{3}\ {}^{3}\underset{\sim}{p}_{3}^{*})\ ) \qquad (2.54) $$

where the ${}^{i}\underset{\sim}{p}_{i}^{*}$ and $\underset{\sim}{R}_{i-1}^{i}$ terms are given in equations 2.9 and 2.5 respectively. The set of sub-equations from 2.54 relate the relative joint displacement parameters θ_i or d_i (for i = 1,2,3) to the wrist point solution from equation 2.53. In the most general case with this form, the solution of a fourth degree polynomial is the most that is required. In the chosen example, these are further reduced to quadratics, by the simpler geometry. Finally, solutions for the wrist joint relative displacements, are found by equating terms from the two $\underset{\sim}{R}_{0}^{6}$ matrices. The 1,3; 3,2; and 3,3 elements are found to give a sufficient number of linearly independent equations. These may be solved for θ_4, θ_5 and θ_6.

2.5.4 Kinematic Inverse - Special Solution Cases

There are a number of special cases which as described, cause problems in the solution. Five examples are outlined here.

Degeneracy - This occurs when the number of d.o.f. of the end effector is less than the number of joints. It may be a design feature of some manipulators, to improve dexterity. In the 3R wrist typified by the Puma 560, it occurs when the θ_4 and θ_6 rotations are co-axial or when the θ_2 and θ_5 axes are parallel. In these circumstances, the manipulator possesses five or fewer d.o.f. even though it still has 6 d.o.m. Mathematically, these effects cause two of the six equations to be linear combinations of the others. If there are more than six joints, or combinations of joints which provide no extra d.o.f. then extra constraints are required for a unique solution.

Numeric Ill Conditioning - Close to deadpoints or singularities, digital arithmetic truncation may create large errors in the solutions. At the deadpoints, special solutions are therefore needed.

Finite Number of Multiple Solutions - These correspond to the multiple mechanism closures which may be possible in achieving a given end effector world space vector. For example, the Puma can typically attain the same endpoint conditions through two, four or eight closures.

No Real Solutions - Imaginary solutions arise when a world space vector is specified which lies outside of the manipulator's workspace.

Mechanical Limitations - Even if equation solutions exist, rotational or translational displacement demands may exceed the capabilities of the individual axes.

2.6 Consideration of a Particular Configuration

2.6.1 The 2R-P $d_1 d_2 d_3 a_3$

General solutions as stated are complex. To reinforce the validity of any particular analyses, a survey of some 40 manufacturers available manipulator configurations has been carried out. In conclusion, just two configurations are adequately representative of many manipulators, these are the 6R and 2R-P-3R, one of the two at least, being produced by 31 of the 40 companies.

The kinematics of the 6R sub-classified as 2R-R-3R $d_1 d_2 a_2 d_4 d_6$ and that of the 2R-P-3R $d_1 d_2 d_3 a_3 d_6$ were studied and found to need very similar solution techniques. With various special case a_i and d_i values (e.g. zero), they encompass many configurations. Here, attention will be restricted to the reduced configuration, 2R-P $d_1 d_2 d_3 a_3$, which corresponds specifically to the first three 'gross', spatial axes of the Little Giant. Given this configuration, the forward kinematic solutions are straightforward. Obviously no independent control may be exercised over the wrist orientation, but any arbitrary value of end point location may be achieved within the feasible workspace.

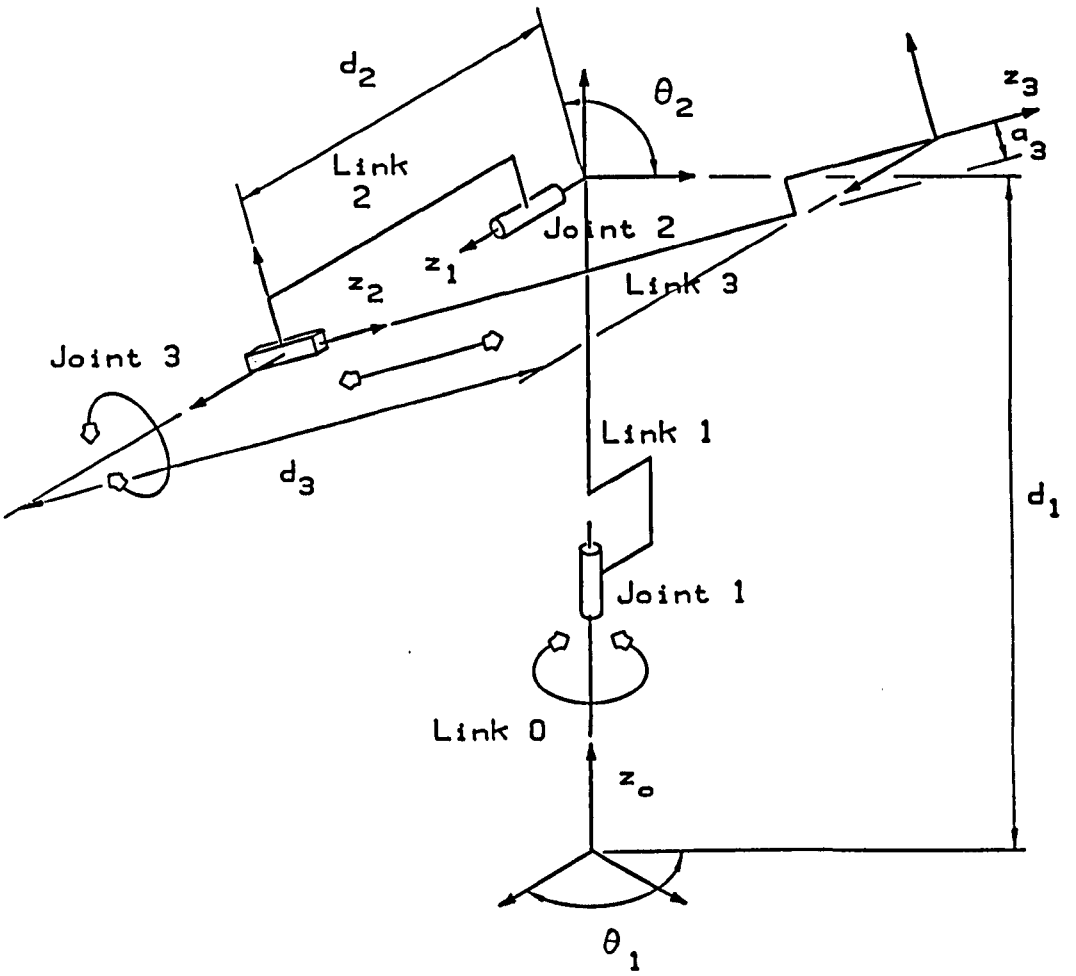


Figure 2.4 The 2R-P $d_1 d_2 d_3 a_3$ Manipulator Kinematic Parameters

2.6.2 2R-P $d_1 d_2 d_3 a_3$ Forward Kinematics

Working sequentially through equations 2.23 to 2.45 as appropriate, the following relationships are obtained (see figure 2.4) :

Link 1 - Revolute Joint

$$ {}^{0}\underset{\sim}{\Omega}_1 = {}^{0}\underset{\sim}{\Omega}_1{}^{*} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \tag{2.55} $$

$$ {}^{0}\dot{\underset{\sim}{\Omega}}_1 = {}^{0}\dot{\underset{\sim}{\Omega}}_1{}^{*} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} \tag{2.56} $$

$$ {}^{0}\underset{\sim}{p}_1 = \underset{\sim}{R}{}^{0}_{1}\, {}^{1}\underset{\sim}{p}_1{}^{*} = \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \tag{2.57} $$

$$ {}^{0}\dot{\underset{\sim}{p}}_1 = {}^{0}\underset{\sim}{\Omega}_1 \times {}^{0}\underset{\sim}{p}_1{}^{*} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.58} $$

$$ {}^{0}\ddot{\underset{\sim}{p}}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.59} $$

$$ {}^{0}\underset{\sim}{r}_1 = {}^{0}\underset{\sim}{p}_1 + \underset{\sim}{R}{}^{0}_{1}\, {}^{1}\underset{\sim}{s}_1 = \begin{bmatrix} c_1 s_{11} + s_1 s_{13} \\ s_1 s_{11} - c_1 s_{13} \\ d_1 + s_{12} \end{bmatrix} \tag{2.60} $$

$$ {}^{0}\dot{\underset{\sim}{r}}_1 = {}^{0}\dot{\underset{\sim}{p}}_1 + {}^{0}\underset{\sim}{\Omega}_1 \times {}^{0}\underset{\sim}{s}_1 = \begin{bmatrix} \dot{\theta}_1 (c_1 s_{13} - s_1 s_{11}) \\ \dot{\theta}_1 (c_1 s_{11} + s_1 s_{13}) \\ 0 \end{bmatrix} \tag{2.61} $$

$$ {}^{0}\ddot{\underset{\sim}{r}}_1 = {}^{0}\ddot{\underset{\sim}{p}}_1 + {}^{0}\dot{\underset{\sim}{\Omega}}_1 \times {}^{0}\underset{\sim}{s}_1 + {}^{0}\underset{\sim}{\Omega}_1 \times ({}^{0}\underset{\sim}{\Omega}_1 \times {}^{0}\underset{\sim}{s}_1) $$

$$ = \begin{bmatrix} \ddot{\theta}_1 (c_1 s_{13} - s_1 s_{11}) - \dot{\theta}_1{}^2 (c_1 s_{11} + s_1 s_{13}) \\ \ddot{\theta}_1 (c_1 s_{11} + s_1 s_{13}) + \dot{\theta}_1{}^2 (c_1 s_{13} - s_1 s_{11}) \\ 0 \end{bmatrix} \tag{2.62} $$

Link 2 - Revolute Joint

$$ {}^{0}\underset{\sim}{\Omega}_2 = {}^{0}\underset{\sim}{\Omega}_1 + \underset{\sim}{R}{}^{0}_{1}\, {}^{1}\underset{\sim}{\Omega}_2{}^{*} = \begin{bmatrix} s_1 \dot{\theta}_2 \\ -c_1 \dot{\theta}_2 \\ \dot{\theta}_1 \end{bmatrix} \tag{2.63} $$

$$ {}^{0}\dot{\underset{\sim}{\Omega}}_2 = {}^{0}\dot{\underset{\sim}{\Omega}}_1 + {}^{0}\dot{\underset{\sim}{\Omega}}_2 + {}^{0}\underset{\sim}{\Omega}_1 \times {}^{0}\underset{\sim}{\Omega}_2{}^{*} $$

$$ = \begin{bmatrix} \ddot{\theta}_2 s_1 + \dot{\theta}_1 \dot{\theta}_2 c_1 \\ -\ddot{\theta}_2 c_1 + \dot{\theta}_1 \dot{\theta}_2 s_1 \\ \ddot{\theta}_1 \end{bmatrix} \tag{2.64} $$

$$ {}^{o}\underset{\sim}{p}_2 = {}^{o}\underset{\sim}{p}_1 + \underset{\sim}{R}_{o}^{2}\ {}^{2}\underset{\sim}{p}_2^{*} = \begin{bmatrix} s_1 d_2 \\ -c_1 d_2 \\ d_1 \end{bmatrix} \tag{2.65} $$

$$ {}^{o}\dot{\underset{\sim}{p}}_2 = {}^{o}\underset{\sim}{\Omega}_2 \times {}^{o}\underset{\sim}{p}_2^{*} = \begin{bmatrix} \dot{\theta}_1\ c_1 d_2 \\ \dot{\theta}_1\ s_1 d_2 \\ 0 \end{bmatrix} \tag{2.66} $$

$$ {}^{o}\ddot{\underset{\sim}{p}}_2 = {}^{o}\ddot{\underset{\sim}{p}}_1 + {}^{o}\dot{\underset{\sim}{\Omega}}_2 \times {}^{o}\underset{\sim}{p}_2^{*} + {}^{o}\underset{\sim}{\Omega}_2 \times ({}^{o}\underset{\sim}{\Omega}_2 \times {}^{o}\underset{\sim}{p}_2^{*}) $$

$$ = \begin{bmatrix} \ddot{\theta}_1\ c_1 d_2 - \dot{\theta}_1^2\ s_1 d_2 \\ \ddot{\theta}_1\ s_1 d_2 - \dot{\theta}_1^2\ c_1 d_2 \\ 0 \end{bmatrix} \tag{2.67} $$

$$ {}^{o}\underset{\sim}{r}_2 = {}^{o}\underset{\sim}{p}_2 + \underset{\sim}{R}_{o}^{2}\ {}^{2}\underset{\sim}{s}_2 = \begin{bmatrix} s_1\ (d_2 + s_{22}) + c_1\ (c_2 s_{21} + s_2 s_{23}) \\ -c_1\ (d_2 + s_{22}) + s_1\ (c_2 s_{21} + s_2 s_{23}) \\ d_1 + s_2 s_{21} - c_2 s_{23} \end{bmatrix} \tag{2.68} $$

$$ {}^{o}\dot{\underset{\sim}{r}}_2 = {}^{o}\dot{\underset{\sim}{p}}_2 + {}^{o}\underset{\sim}{\Omega}_2 \times {}^{o}\underset{\sim}{s}_2 = $$

$$ \begin{bmatrix} \dot{\theta}_1\ [c_1\ (d_2 + s_{22}) - s_1\ (c_2 s_{21} + s_2 s_{23})] - \dot{\theta}_2\ [c_1\ (s_2 s_{21} - c_2 s_{23})] \\ \dot{\theta}_1\ [s_1\ (d_2 + s_{22}) + c_1\ (c_2 s_{21} + s_2 s_{23})] - \dot{\theta}_2\ [s_1\ (s_2 s_{21} - c_2 s_{23})] \\ \dot{\theta}_2\ (c_2 s_{21} + s_2 s_{23}) \end{bmatrix} \tag{2.69} $$

$$ {}^{o}\ddot{\underset{\sim}{r}}_2 = {}^{o}\ddot{\underset{\sim}{p}}_2 + {}^{o}\dot{\underset{\sim}{\Omega}}_2 \times {}^{o}\underset{\sim}{s}_2 + {}^{o}\underset{\sim}{\Omega}_2 \times ({}^{o}\underset{\sim}{\Omega}_2 \times {}^{o}\underset{\sim}{s}_2) $$

$$ = \begin{bmatrix} \ddot{\theta}_1\ [c_1\ (d_2 + s_{22}) - s_1\ (c_2 s_{21} + s_2 s_{23})] \\ -\dot{\theta}_1^2\ [s_1\ (d_2 + s_{22}) + c_1\ (c_2 s_{21} + s_2 s_{23})] \\ +\dot{\theta}_1 \dot{\theta}_2\ [2\ s_1\ (s_2 s_{21} - c_2 s_{23})] \\ -\dot{\theta}_2^2\ [c_1\ (c_2 s_{21} + s_2 s_{23})] \\ -\ddot{\theta}_2\ [c_1\ (s_2 s_{21} - c_2 s_{23})] \\ \\ \ddot{\theta}_1\ [s_1\ (d_2 + s_{22}) + c_1\ (c_2 s_{21} + s_2 s_{23})] \\ +\dot{\theta}_1^2\ [c_1\ (d_2 + s_{22}) - s_1\ (c_2 s_{21} + s_2 s_{23})] \\ -\dot{\theta}_1 \dot{\theta}_2\ [2\ c_1\ (s_2 s_{21} - c_2 s_{23})] \\ -\dot{\theta}_2^2\ [s_1\ (c_2 s_{21} + s_2 s_{23})] \\ -\ddot{\theta}_2\ [s_1\ (s_2 s_{21} - c_2 s_{23})] \\ \\ -\dot{\theta}_2^2\ [s_2 s_{21} - c_2 s_{23}] \\ +\ddot{\theta}_2\ [c_2 s_{21} + s_2 s_{23}] \end{bmatrix} \tag{2.70} $$

Link 3 - Prismatic Joint

$$ {}^{0}\underset{\sim}{\Omega}_3 = {}^{0}\underset{\sim}{\Omega}_2 \qquad (2.71) $$

$$ {}^{0}\dot{\underset{\sim}{\Omega}}_3 = {}^{0}\dot{\underset{\sim}{\Omega}}_2 \qquad (2.72) $$

$$ {}^{0}\underset{\sim}{p}_3 = {}^{0}\underset{\sim}{p}_2 + \underset{\sim}{R}_0^3 \, {}^{3}\underset{\sim}{p}_3^* = \begin{bmatrix} s_1 d_2 + c_1 \ (c_2 a_3 + s_2 d_3) \\ -c_1 d_2 + s_1 \ (c_2 a_3 + s_2 d_3) \\ d_1 + (s_2 a_3 - c_2 d_3) \end{bmatrix} \qquad (2.73) $$

$$ {}^{0}\dot{\underset{\sim}{p}}_3 = {}^{0}\dot{\underset{\sim}{p}}_2 + \underset{\sim}{R}_0^3 \, {}^{3}\dot{\underset{\sim}{p}}_3^* + {}^{0}\underset{\sim}{\Omega}_3 \times {}^{0}\underset{\sim}{p}_3^* = $$

$$ \begin{bmatrix} \dot{\theta}_1 \ [c_1 d_2 - s_1 \ (c_2 a_3 + s_2 d_3)] - \dot{\theta}_2 \ [c_1 \ (s_2 a_3 - c_2 d_3)] + \dot{d}_3 \ (c_1 s_2) \\ \dot{\theta}_1 \ [s_1 d_2 + c_1 \ (c_2 a_3 + s_2 d_3)] - \dot{\theta}_2 \ [s_1 \ (s_2 a_3 - c_2 d_3)] + \dot{d}_3 \ (s_1 s_2) \\ \dot{\theta}_2 \ (c_2 a_3 + s_2 d_3) - \dot{d}_3 c_2 \end{bmatrix} \qquad (2.74) $$

$$ {}^{0}\ddot{\underset{\sim}{p}}_3 = {}^{0}\ddot{\underset{\sim}{p}}_2 + {}^{0}\ddot{\underset{\sim}{p}}_3^* + {}^{0}\dot{\underset{\sim}{\Omega}}_3 \times {}^{0}\underset{\sim}{p}_3^* + 2 \, {}^{0}\underset{\sim}{\Omega}_3 \times {}^{0}\dot{\underset{\sim}{p}}_3^* + {}^{0}\underset{\sim}{\Omega}_3 \times ({}^{0}\underset{\sim}{\Omega}_3 \times {}^{0}\underset{\sim}{p}_3^*) = $$

$$ \begin{bmatrix} \ddot{\theta}_1 \ [c_1 d_2 - s_1 \ (c_2 a_3 + s_2 d_3)] - \dot{\theta}_1^2 \ [s_1 d_2 + c_1 \ (c_2 a_3 + s_2 d_3)] \\ +\dot{\theta}_1 \dot{\theta}_2 \ [2 s_1 \ (s_2 a_3 - c_2 d_3)] - \dot{\theta}_1 \dot{d}_3 \ [2 s_1 s_2] + \dot{\theta}_2 \dot{d}_3 \ [2 c_1 c_2] \\ -\dot{\theta}_2^2 \ [c_1 \ (c_2 a_3 + s_2 d_3)] - \ddot{\theta}_2 \ [c_1 \ (s_2 a_3 - c_2 d_3)] + d_3 \ [c_1 s_2] \\ \\ \ddot{\theta}_1 \ [s_1 d_2 + c_1 \ (c_2 a_3 + s_2 d_3)] + \dot{\theta}_1^2 \ [c_1 d_2 - s_1 \ (c_2 a_3 + s_2 d_3)] \\ -\dot{\theta}_1 \dot{\theta}_2 \ [2 c_1 \ (s_2 a_3 - c_2 d_3)] + \dot{\theta}_1 \dot{d}_3 \ [2 c_1 s_2] + \dot{\theta}_2 \dot{d}_3 \ [2 s_1 c_2] \\ -\dot{\theta}_2^2 \ [s_1 \ (c_2 a_3 + s_2 d_3)] - \ddot{\theta}_2 \ [s_1 \ (s_2 a_3 - c_2 d_3)] + \ddot{d}_3 \ [s_1 s_2] \\ \\ \ddot{\theta}_2 \dot{d}_3 \ [2 s_2] - \dot{\theta}_2^2 \ [s_2 a_3 - c_2 d_3] + \ddot{\theta}_2 \ [c_2 a_3 + s_2 d_3] - \ddot{d}_3 \ [c_2] \end{bmatrix} \qquad (2.75) $$

$$ {}^{0}\underset{\sim}{r}_3 = {}^{0}\underset{\sim}{p}_3 + \underset{\sim}{R}_0^3 \, {}^{3}\underset{\sim}{s}_3 $$

$$ = \begin{bmatrix} s_1 \ (d_2 + s_{32}) + c_1 \ [(c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33})] \\ -c_1 \ (d_2 + s_{32}) + s_1 \ [(c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33})] \\ d_1 + (s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33}) \end{bmatrix} \qquad (2.76) $$

$$ {}^{0}\dot{\underset{\sim}{r}}_3 = {}^{0}\dot{\underset{\sim}{p}}_3 + {}^{0}\underset{\sim}{\Omega}_3 \times {}^{0}\underset{\sim}{s}_3 = $$

$$ \begin{bmatrix} \dot{\theta}_1 \ [c_1 \ (d_2 + s_{32}) - s_1 \ ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ -\dot{\theta}_2 \ [c_1 \ ((s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33}))] + \dot{d}_3 \ [c_1 s_2] \\ \\ \dot{\theta}_1 \ [s_1 \ (d_2 + s_{32}) + c_1 \ ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ -\dot{\theta}_2 \ [s_1 \ ((s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33}))] + \dot{d}_3 \ [s_1 s_2] \\ \\ \dot{\theta}_2 \ [(c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33})] + \dot{d}_3 \ [c_2] \end{bmatrix} \qquad (2.77) $$

and finally

$$ {}^{o}\ddot{\underset{\sim}{r}}_3 = {}^{o}\ddot{\underset{\sim}{p}}_3 + {}^{o}\dot{\underset{\sim}{\Omega}}_3 \times {}^{o}\underset{\sim}{s}_3 + {}^{o}\underset{\sim}{\Omega}_3 \times ({}^{o}\underset{\sim}{\Omega}_3 \times {}^{o}\underset{\sim}{s}_3) $$

$$ = \begin{bmatrix} \ddot{\theta}_1 [c_1 (d_2 + s_{32}) - s_1 ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ -\dot{\theta}_1^2 [s_1 (d_2 + s_{32}) + c_1 ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ +\dot{\theta}_1 \dot{\theta}_2 [2 s_1 ((s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33}))] \\ -\dot{\theta}_1 \dot{d}_3 [2 s_1 s_2] + \dot{\theta}_2 \dot{d}_3 [2 c_1 c_2] \\ -\dot{\theta}_2^2 [c_1 ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ -\ddot{\theta}_2 [c_1 ((s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33}))] + \ddot{d}_3 [c_1 s_2] \\ \\ \ddot{\theta}_1 [s_1 (d_2 + s_{32}) + c_1 ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ +\dot{\theta}_1^2 [c_1 (d_2 + s_{32}) - s_1 ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ -\dot{\theta}_1 \dot{\theta}_2 [2 c_1 ((s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33}))] \\ +\dot{\theta}_1 \dot{d}_3 [2 c_1] + \dot{\theta}_2 \dot{d}_3 [2 s_1 c_2] \\ -\dot{\theta}_2^2 [s_1 ((c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33}))] \\ -\ddot{\theta}_2 [s_1 ((s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33}))] + \ddot{d}_3 [s_1 s_2] \\ \\ -\dot{\theta}_1 \dot{\theta}_2 [2 s_{32}] + \dot{\theta}_2 \dot{d}_3 [2 s_2] \\ -\dot{\theta}_2^2 [(s_2 a_3 - c_2 d_3) + (s_2 s_{31} - c_2 s_{33})] \\ +\ddot{\theta}_2 [(c_2 a_3 + s_2 d_3) + (c_2 s_{31} + s_2 s_{33})] - \ddot{d}_3 [c_2] \end{bmatrix} \qquad (2.78) $$

2.6.3 Displacement Kinematic Inverse

Based on equation 2.73, and dropping the prescript ^{o} and subscript $_3$, on the X,Y and Z components :

$$ {}^{o}\underset{\sim}{p}_3 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} s_1 d_2 + c_1 (c_2 a_3 + s_2 d_3) \\ -c_1 d_2 + s_1 (c_2 a_3 + s_2 d_3) \\ d_1 + s_2 a_3 - c_2 d_3 \end{bmatrix} \qquad (2.79) $$

Note that the respective axis frame locations have been chosen with hindsight, enabling substantial cancellation and simpler solutions. Substituting the X and Y terms :

$$ X s_1 - Y c_1 = d_2 \qquad (2.80) $$

hence

$$ \theta_1 = 2 \, \mathrm{Tan}^{-1} \left[\frac{X \pm \sqrt{(X^2 + Y^2 - d_2^2)}}{d_2 - Y} \right] \qquad (2.81) $$

noting that if $d_2 = 0$ then :

$$ \theta_1 = \mathrm{Tan}^{-1} (Y/X) \text{ for } Y \neq 0 \qquad (2.82) $$

where the negative root is the only physical possibility. Solutions can exceed 2π but the mechanical capabilities again mean this is easily solved :

$$\text{if } \theta_1 > 2\pi \text{ then } \theta_1 = \theta_1 - 2\pi \qquad (2.83)$$

Substituting the Z and X terms :

$$s_2\,[c_1\,(d_1 - Z)] + c_2\,(s_1 d_2 - X) = -c_1 a_3 \qquad (2.84)$$

hence similarly :

$$\theta_2 = 2\,\text{Tan}^{-1}\left[\frac{c_1\,(d_1 - Z) \pm \sqrt{(\,[c_1\,(d_1 - Z)]^2 + (s_1 d_2 - X)^2 - (c_1 a_3)^2}}{s_1 d_2 - X - c_1 a_3}\right] \qquad (2.85)$$

Again the negative root is appropriate for the mechanical capabilities of $60^\circ \le \theta_2 \le 120^\circ$. And finally :

$$d_3 = (d_1 + s_2 a_3 - Z) / c_2 \qquad (2.86)$$

Now close to cos $\theta_2 = 0$ the solution breaks down. The 12 bit resolution on the axis displacement measurements means the minimum magnitude cosine value other than 0 is around :

$$|\cos\theta_2| \approx 2\times10^{-4} \qquad (2.87)$$

At zero the set of forward kinematic solutions reduce to :

$$X = c_1 d_3 + d_2 s_1 \qquad (2.88)$$

$$Y = s_1 d_3 - d_2 c_1 \qquad (2.89)$$

$$Z = d_1 + a_3 \qquad (2.90)$$

hence

$$X^2 + Y^2 = d_3^2 + d_2^2 \qquad (2.91)$$

or

$$d_3 = +\sqrt{(X^2 + Y^2 - d_2^2)} \qquad (2.92)$$

the + root giving d3 > 0 (again mechanically constrained to be so.)

Actual values are $d_1 = 1.111$ m, $d_2 = 0$ and $a_3 = -0.085$ m. Both forward and inverse solutions were verified against each other for a range of values.

2.6.4 Velocity Kinematic Inverse

These solutions are required in the case where velocity boundary conditions are imposed on Cartesian defined motions. Additionally, the nominal joint velocities resulting from a Cartesian based motion are required as an input to a model used in chapter 7.

From equations 2.74 :

$$ {}^{o}\dot{\underset{\sim}{p}}_{3} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = $$

$$ \begin{bmatrix} \dot{\theta}_1 [c_1 d_2 - s_1 (c_2 a_3 + s_2 d_3)] - \dot{\theta}_2 [c_1 (s_2 a_3 - c_2 d_3)] + \dot{d}_3 (c_1 s_2) \\ \dot{\theta}_1 [s_1 d_2 + c_1 (c_2 a_3 + s_2 d_3)] - \dot{\theta}_2 [s_1 (s_2 a_3 - c_2 d_3)] + \dot{d}_3 (s_1 s_2) \\ \dot{\theta}_2 (c_2 a_3 + s_2 d_3) - \dot{d}_3 c_2 \end{bmatrix} \qquad (2.93) $$

Dropping the pre-script and post-script from rows 1 and 2 :

$$ \dot{\theta}_1 = (c_1 \dot{Y} - s_1 \dot{X}) / (c_2 a_3 + s_2 d_3) \qquad (2.94) $$

and row 3 results in

$$ \dot{\theta}_2 = (\dot{Z} + c_2 \dot{d}_3) / (c_3 a_3 + s_2 d_3) \qquad (2.95) $$

which on back substitution into row 1 provides :

$$ \dot{d}_3 = \frac{(c_2 a_3 + s_2 d_3)[\dot{X} - \dot{\theta}_1 (c_1 d_2 - s_1 (c_2 a_3 + s_2 d_3))] + \dot{Z} c_1 (s_2 a_3 - c_2 d_3)}{c_1 d_3} \qquad (2.96) $$

These three equations break down for zero denominators. The conditions are :

(i) $c_2 a_3 + s_2 d_3 = 0$

This causes the solutions for X and Y to appear as :

$X = s_1 d_2$ and $Y = -c_1 d_2$

hence $\text{Tan}\ \theta_1 = -X / Y$. This solution is outside the mechanical range of the manipulator, so can be neglected.

(ii) The denominator of equation 2.96 can be zero for $d_3 = 0$ or $c_1 = 0$. $d_3 = 0$ is again outside the manipulator's capability.

$c_1 = 0$ occurs at two locations within the θ_1 range.

Re-writing row 2 for $c_1 = 0$, and substituting for $\dot{\theta}_2$ gives the alternative form :

$$ \dot{d}_3 = \frac{(c_2 a_3 + s_2 d_3)[\dot{Y} - \dot{\theta}_1 s_1 d_2] + \dot{Z} s_1 (s_2 a_3 - c_2 d_3)}{s_1 d_3} \qquad (2.97) $$

In summary, practical kinematic solutions for the 2R-P $d_1 d_2 d_3 a_3$ manipulator have been obtained for :

(i) the world motions (displacement, velocity and acceleration) of the link axis origins and the link centres of gravity, as functions of the joint displacement, velocity and acceleration values.

(ii) the joint motions (displacement and velocity) as functions of the world displacement and velocity.

3 Manipulator Dynamics and Control

3.1 Dynamics Specific to Robotics

Dynamic models can be used within robotics for numerous purposes, e.g. motion design and optimisation, feedback control design and simulation. Considering these individually :

(i) Motion Design and Optimisation - Motion can be designed using dynamic models, to minimise tracking errors with respect to some given nominal motion. Once generated, a nominal motion is distorted in its execution by virtue of the robot's dynamics. Machine designers have attempted in the past to reduce this distortion by explicitly building the system dynamics into the motion. If a system dynamic model is known to sufficient accuracy, then the dynamic coefficients can be built into the motion law definition, producing a so called 'tuned' motion law. Driving the dynamic system with such a tuned law, results in reduced tracking errors. Stoddart 1953 [3.1] uses very simple models of cam follower systems to design so called 'Polydyne' motion profiles. These of course are implemented in metal and so are dedicated to a given task and operating speed. The dynamic qualities are sensitive to the operating speed. More practicable motion 'tuning' schemes will be devised in the sequel.

If these ideas are applied to programmable multi-axis machines such as robots, there is far more flexibility, as the motion shape definition (equivalent to the cam profile) can be modified with ease. To perform such model based tuning requires precise modelling, the feasibility of the process or otherwise, is discussed further in chapter 5.

(ii) Feedback Control - Industrial robot controller designs are usually empirical but more advanced designs are needed for the higher speeds envisaged and now available in some new robots. Controller structure design, gain optimisation, stability and performance assessment would be very difficult without dynamic models. Certain controllers require a full dynamic model, or at least its form, built into them. Feedback control is employed on

all but the most basic of manipulators. The objectives being to attain good tracking accuracy to a time coordinated motion command. An undesirable by-product of control is the interaction of the controller and the robot dynamics, potentially inducing instability. Industrial robot feedback controllers often comprise independent and crude controllers for each link. This type of design neglects the form of the robot's dynamics and almost certainly degrades performance to something below the machine's capability. So a dynamic model would ideally be used both in the design of the form of controller, and its stability analysis.

(iii) Simulation - Improvements can be made in the design of the manipulator and actuator systems through improved understanding of manipulator dynamics. Motion feasibility and predicted system performance assessments may also be made using dynamic simulations. Feasibility of a motion can be assessed using dynamic models i.e. computing the actuator torque or force demand from the desired motion (and its derivatives) and ensuring that it is within the actuator capabilities. This applies to motions having a predefined path and time coordination. Some motions may have no constraints on the path or its timing. In these cases, dynamic models may provide extra constraints given say a minimum time requirement. This application of simulation is clearly demonstrated in Shiller and Dubowsky 1985 [3.2]. Completely automated motion generation is extremely complex but schemes such as this, could form a useful part of an overall strategy.

3.2 Level of Modelling Detail

Some problems can be classed as lying within a subset of dynamics (e.g. planar dynamics), leading to a general simplification. This is not the case with robot modelling, which includes both multi-body and spatial dynamics. Referring back to the introduction, certain features common to most manipulators automatically simplify models and are often taken for granted. For example :

(i) Use of single freedom joints (revolute or prismatic) as opposed to combinations; one joint per link which has its axis

orthogonal or parallel to those of the adjacent links.

(ii) The basic structure is an open chain, serial sequence of links.

(iii) The three axes nearest the 'ground' generate the major displacements and so dominate the gross motion response.

There are many other factors which can be considered in respect of robot dynamic models. Of course it is desirable to omit the features which are not essential. The practical factors must be individually assessed to justify this approach.

As a starting point, it is of value to consider some of the commonly referred to works which apply robot dynamics.

In order to produce time optimal and sub-optimal controls, Kahn and Roth 1971 [3.3] derived the equations of motion of a 3 axis, all revolute manipulator, using Lagranges' equations. The optimal solution is found to be specific to particular initial and final conditions, making it necessary to recompute the optimal trajectory for every set of boundary conditions. Linearising the model, they produce a more useful but sub-optimal control, also a switching control (of bang-bang type). The dynamic model assumes perfect actuators (unity transfer function); rigid links; coordinate axes defining joints are parallel to the principal axes of inertia for the mass centre of the link; the mass centres of links lie on the straight line between successive link coordinate frame origins; and two of the four offsets (by the Denavit and Hartenberg scheme) between joints are zero on all links. Kahn and Roth also notice the advantages in carefully selecting the locations of each link coordinate frame.

Paul 1972 [3.4] also uses a Lagrangian formulation. The dynamic equations are again used for control purposes. He claims that the velocity dependent terms are only significant at high speed and are small for the arm being used. The varying inertial and gravitational components are computed and added into an otherwise conventional PD controller. Other assumptions made are similar to those of Kahn and Roth.

More efficient dynamic algorithms are used in Luh, Walker and Paul

1980 [3.5] which is a logical extension to Paul's earlier work. In this is developed a so called computed torque control.

Issues which are apparent in the development of dynamic models are :

(i) Ease of derivation /formulation
(ii) Algebra verification
(iii) Level of understanding/insight/information given
(iv) Ease of conversion to a computer algorithm
(v) Computational load
(vi) Accuracy of modelling - Gross response
- Fine motion response
(perturbation about gross response)
(vii) Experimental effort required to evaluate model parameters
(viii) Suitability for forward or reverse computation

Factors which influence these issues include :

(i) General or specific dynamic model
(e.g. exploitation of manipulator specific properties)
(ii) Explicit or Recursive equation forms

(iii) Approximations to be considered
-Inclusion of kinematic loops
-Locations of C of G's
-Lumped or distributed parameters
-Link internal drive components
-Inclusion of compliance, backlash, frictions
-Inclusion of Inertial, Coriolis,
Centripetal and Gravitational terms
-Level of actuator modelling

It is not practical to carry out motion generation specifically for each individual robot. It is concluded that the best compromise for a dynamic model is one of the usual rigid link schemes, neglecting detailed issues. It is known that the insight given and the gross responses are broadly speaking, accurate.

The full rigid scheme, i.e. including Inertial, Centripetal, Coriolis

and Gravitational terms results in a set of nonlinear second order, second degree, coupled differential equations. Actuator and joint effects are neglected. This conclusion is frustrating, but it is unlikely that any significant progress in generic terms can be made at this stage with dynamic motion design using highly detailed models.

This lack of fine accuracy is catered for in differing ways by the motion design schemes of chapters 6 and 7. In chapter 6, the response itself is used as the modification to the motion law. The robot's dynamics are embedded in the response, and so there is no requirement for an exact dynamic model. The MRACS scheme of chapter 7 attempts to force the robot to behave as if its dynamics are simple and well defined.

3.3 Dynamic Model Derivation Method

Dynamic study yields mathematical models, analagous to the relationships between forces and resulting motion. The methods of obtaining these models are many. Of those applied to manipulators, schemes include :

Newton's Laws
Euler's Equations
Free Body Analysis
d'Alembert's Formalisms

] these are closely related in their use.

Lagrange's Equations

Hamilton's Canonical Equations

and less well known schemes include :

Gibb's Appel Equations

Bond Graphs

Influence Coefficients

Articulated Body Inertias

Boltzmann-Hamel's Equations

The most commonly used are Newton's Laws with Euler's Equations (free body), alternatively Lagrange's Equations.

The equations of spatial manipulators become unwieldy without an efficient, precise and clear notation. The kinematics notation is adopted and expanded with appropriate symbols.

Studies have been made of the computational efficiency of different dynamic models. Hollerbach 1980 [3.6] for example compares the numbers of multiplications and additions required in the evaluation of six schemes for the evaluation of the dynamics. The most efficient in these terms is found to be the so-called Newton-Euler Recursive Method. This relatively general method can be significantly improved by tailoring for a specific manipulator. Horak 1984 [3.7] describes these tailored algorithms running five times faster. He takes advantage of the relatively simple geometry of real manipulators, also the fact that most manipulators possess only 4 to 7 links. Lagrange's equations are used for the first three links, the remaining 1 to 4 links being defined by Newton-Euler recursive equations.

The Newton-Euler theme was pursued further by Walker and Orin 1982 [3.8]. In this, four different methods of computing and solving the sets of equations are contrasted. In the most efficient (method 3), link inertias, masses and centres of mass of links above the joint being computed are grouped to be considered as a composite unit. Although efficient, minimal insight into the dynamics is derived from it. Method 2 is preferred here, it is only a little less computationally efficient, but utilises a set of equations which are meaningful and easier to derive.

System dynamics and kinematics may be related from one moving or fixed coordinate frame of reference, to another. Given a multiple link manipulator, each link has a frame of reference. The relationships between the frames of one link and another are generally far simpler than those between the link and the ground. By exploiting this characteristic and recursively computing the dynamics of each link in turn, they attain a major simplification.

In this recursive form, the velocities and accelerations are computed from the ground up, the torques and forces from the end effector back to ground. At each stage, numerical values are passed from one link's set of equations to the next. Walker and Orin's 'Method 2' also

exploits the symmetry of the robot's inertia matrix, but not in its inversion. Closed form equations of motion can be systematically but tediously derived using the method, by successively substituting for inter link forces and torques.

The complete method is less viable when analysing mechanisms containing closed kinematic loops. The equations can be set up for solution, but some of them need to be simultaneously solved using a numerical method. These closed loops arise through the presence of actuator sub-systems or stabilising links for extra stiffness. In appropriate circumstances the loop can be neglected, mass properties being distributed into the links on either side of the joint. Actuator force or torque can then be resolved to appear as if it acted directly at the joint.

Lagrangian mechanics provides a systematic derivation of the manipulator dynamics. This again may be presented in a closed or recursive form. It is less computationally efficient, Hollerbach 1980 [3.6] and in the closed form does not provide inter-link force data. One advantage is that in the closed form, a matrix representation yields insight into the total manipulator's dynamics. The level of coupling between joint coordinates, relative complexities and significance of the inertia, velocity product and gravitational terms is more apparent than with the sets of independent Newton-Euler equations.

Lagrangian based derivations for spatial manipulators with three or more links are notoriously lengthy and prone to algebraic errors. This is especially true if link alignments and centres of mass are not conveniently arranged. During the lengthy but systematic derivation, an apparent symmetry appears in the terms, many of which simplify and cancel. These features suggest improvements could be made to the algebraic processes involved in deriving open chain mechanism dynamics. Other valuable work in this area would be a text containing a collection of verified and representative rigid dynamic manipulator models.

In the Lagrangian formulation, closed kinematic loops again require special considerations. They give rise to a set of non-holonomic

constraints, appearing in the equations as Lagrange multipliers, which also need solution.

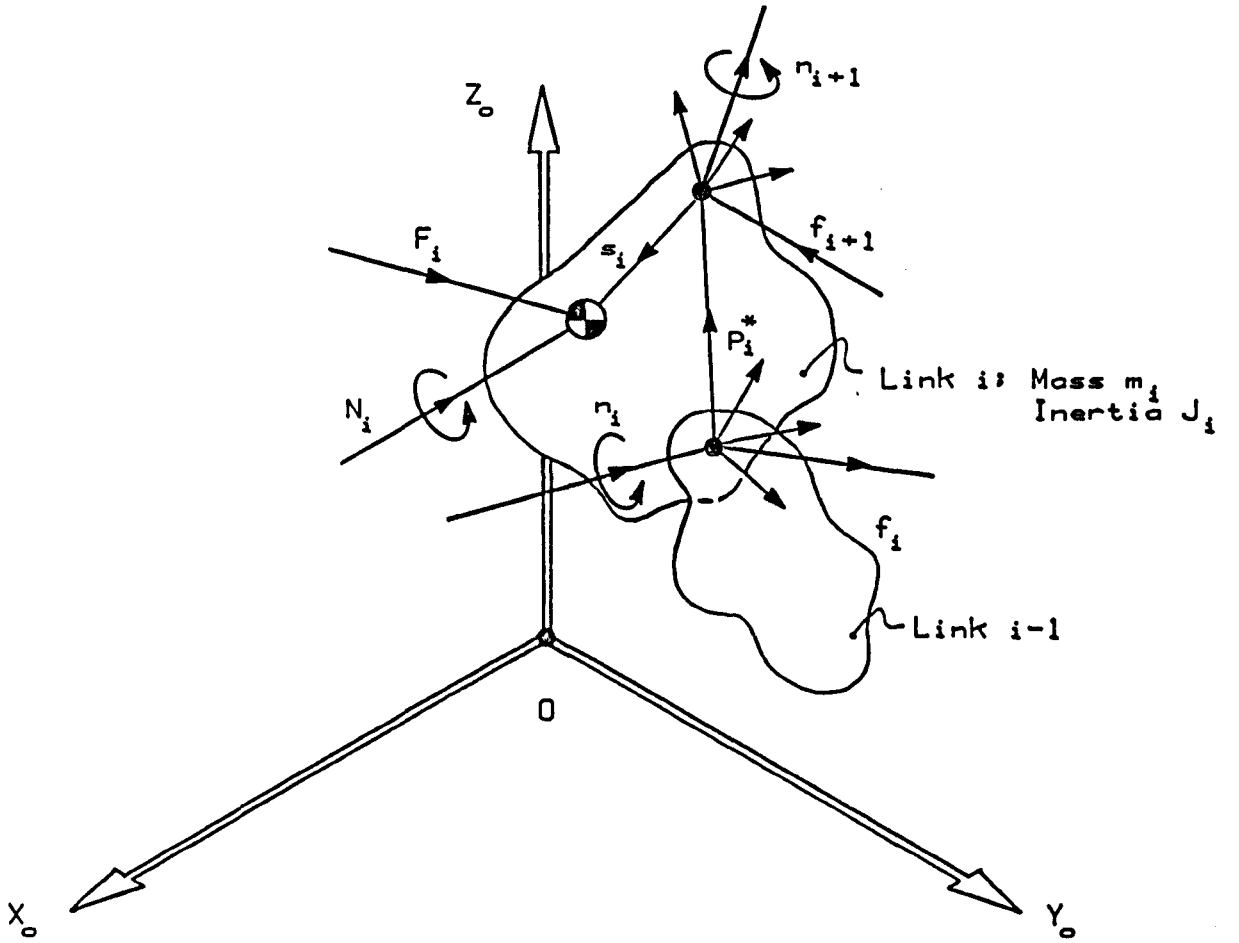


Figure 3.1 Manipulator Link Dynamic Parameters

3.3.1 Spatial Equations of Motion : Newton-Euler

For the ith link (figures 3.1 & 2.1), the dynamic equations of motion may be stated as :

$$\underset{\sim}{F}_i = m_i \ddot{\underset{\sim}{r}}_i \qquad \text{Newtons Second Law} \qquad (3.1)$$

$$\underset{\sim}{N}_i = \underset{\sim}{J}_i \dot{\underset{\sim}{\Omega}}_i + \underset{\sim}{\Omega}_i \times (\underset{\sim}{J}_i . \underset{\sim}{\Omega}_i) \qquad \text{Eulers Equation} \qquad (3.2)$$

$$\underset{\sim}{f}_i = \underset{\sim}{F}_i + \underset{\sim}{f}_{i+1} \qquad (3.3)$$

$$\underset{\sim}{n}_i = \underset{\sim}{n}_{i+1} + \underset{\sim}{N}_i + (\underset{\sim}{p}_i^* + \underset{\sim}{s}_i) \times \underset{\sim}{F}_i + \underset{\sim}{p}_i \times \underset{\sim}{f}_{i+1} \qquad (3.4)$$

where $\underset{\sim}{F}_i$ - total vector of forces applied to link i

m_i - link i mass

$\underset{\sim}{N}_i$ - total external vector moment exerted on link i

$\underset{\sim}{J}_i$ - link inertia matrix

$\underset{\sim}{f}_i$ - force exerted on link i by previous link

$\underset{\sim}{n}_i$ - moment exerted on link i by previous link

The first two equations consider the motion effects of the net forces and moments on the link. The latter two give dynamic equilibrium conditions for the link as a whole. To obtain an explicit matrix equation for the whole robot in the form :

$$\underset{\sim}{H}(\underset{\sim}{q})\ \ddot{\underset{\sim}{q}} + \underset{\sim}{V}(\underset{\sim}{q},\dot{\underset{\sim}{q}})\ \dot{\underset{\sim}{q}} + \underset{\sim}{G}(\underset{\sim}{q}) = \underset{\sim}{\tau} \qquad (3.5)$$

where $\underset{\sim}{H}$ - robot inertia matrix

$\underset{\sim}{q}$ - vector of joint coordinates

$\underset{\sim}{V}$ - matrix of robot Coriolis and centripetal coefficients

$\underset{\sim}{G}$ - vector of robot gravitation coefficients

$\underset{\sim}{\tau}$ - vector of joint torques

Equation 3.5 requires all quantities in the equation to be referred to the same datum, then successive substitutions need to be made for all the kinematic parameters and the inter link forces and moments. This entirely negates the apparent simplicity of the Newton Euler scheme. If on the other hand, all that is required is a numerical solution e.g. for simulation, these equations provide this very efficiently. Quantities are referred only to their local coordinate frames making them very simple. The sets of equations may be set up in subroutines or procedures to cater for fairly general open chain link sets.

3.3.2 Spatial Equations of Motion : Lagrange

A normal Lagrangian formulation automatically eliminates the inter link forces and moments, except those which appear in the generalised forces. The ith link's dynamics are described by :

$$\frac{d}{dt}\left[\frac{\partial(KE)}{\partial \dot{q}_i}\right] - \frac{\partial(KE)}{\partial q_i} + \frac{\partial(PE)}{\partial q_i} = F_i \qquad (3.6)$$

This seemingly innocuous expression again demands substantial effort to obtain the equations of motion, because of the mechanism's kinematic complexity. The kinetic and potential energy expressions are for an 'n' axis robot :

$$KE = \frac{1}{2}\sum_{i=1}^{n}\left(m_i\,\dot{\underset{\sim}{r}}_i \cdot \dot{\underset{\sim}{r}}_i + \underset{\sim}{\Omega}_i \cdot \underset{\sim}{J}_i \cdot \underset{\sim}{\Omega}_i\right) \qquad (3.7)$$

$$PE = g \sum_{i=1}^{n} [0,\ 0,\ m_i]^T . \underset{\sim}{r}_i \qquad (3.8)$$

The repeated multiplication and cancellations which occur cause a cascading of any errors present, throughout the derivation, necessitating great care. Final equations do possess some 'pattern' properties which aid checking. For example, the inertia matrix must be symmetric* giving a check to the off diagonal terms; Coriolis and centripetal terms tend to exhibit forms of symmetry; the gravitational term is usually simple in form; the presence or omission of coupling terms can to an extent be qualitatively checked etc. The centre of gravity accelerations can be used to check the remaining inertia matrix leading diagonal elements. If both revolute and prismatic joints are present in the same robot, the effectiveness of some of these methods is lost as the patterns in the velocity product terms are not as clear.

* If motion is computed relative to some fixed position in space, then the symmetry of the inertia matrix can be deduced from Maxwell's reciprocity theorem.

3.4 2R-P $d_1 d_2 d_3 a_3$ Equations of Motion by Lagrange

The various terms in the energy expressions require evaluation for the chosen robot. The $\underset{\sim}{r}$, $\dot{\underset{\sim}{r}}$ and $\underset{\sim}{\Omega}$ terms have been computed in the kinematics sub-section. The inertia terms are given by :

$$^0\underset{\sim}{J}_i = \underset{\sim}{R}^0_i \ {}^i\underset{\sim}{J}_i \ \underset{\sim}{R}_i^0 \qquad (3.9)$$

and it is assumed that both the inertia matrix and the centre of gravity location vectors contain no zero elements when referenced to their own link coordinate frame. Note that link rotational kinetic energy can alternatively be expressed as :

$$\frac{1}{2} (\underset{\sim}{R}_i^0 \ {}^0\underset{\sim}{\Omega}_i)^T . {}^i\underset{\sim}{J}_i . (\underset{\sim}{R}_i^0 \ {}^0\underset{\sim}{\Omega}_i) \qquad (3.10)$$

or

$$\frac{1}{2} {}^0\underset{\sim}{\Omega}_i^T . (\underset{\sim}{R}^0_i \ {}^i\underset{\sim}{J}_i \ \underset{\sim}{R}_i^0) . {}^0\underset{\sim}{\Omega}_i \qquad (3.11)$$

and because the final result is a scalar (of the same value), they are equal. In hindsight the former is much simpler to compute than the latter. On multiplying out the rotational inertia terms, the forms are simple, thus :

$$^0\underset{\sim}{\Omega}_1^T . {}^0\underset{\sim}{J}_1 . {}^0\underset{\sim}{\Omega}_1 = \dot{\theta}_1^2 \ J_{1yy} \qquad (3.12)$$

$$ {}^{o}\underset{\sim}{\Omega}_2 . {}^{o}\underset{\sim}{J}_2 . {}^{o}\underset{\sim}{\Omega}_2 + {}^{o}\underset{\sim}{\Omega}_3 . {}^{o}\underset{\sim}{J}_3 . {}^{o}\underset{\sim}{\Omega}_3 = {}^{o}\underset{\sim}{\Omega}_2 . ({}^{o}\underset{\sim}{J}_2 + {}^{o}\underset{\sim}{J}_3) . {}^{o}\underset{\sim}{\Omega}_2 \qquad (3.13) $$

because $ {}^{o}\underset{\sim}{\Omega}_3 = {}^{o}\underset{\sim}{\Omega}_2 $ and $ \underset{\sim}{Ro}^3 = \underset{\sim}{Ro}^2 $

hence

$$ {}^{o}\underset{\sim}{\Omega}_2 . ({}^{o}\underset{\sim}{J}_2 + {}^{o}\underset{\sim}{J}_3) . {}^{o}\underset{\sim}{\Omega}_2 = $$

$$ \dot{\theta}_2^2 [J_{2yy} + J_{3yy}] + \dot{\theta}_1 \dot{\theta}_2 [2s_2 (J_{2xy} + J_{3xy}) - 2c_2 (J_{2yz} + J_{3yz})] $$

$$ + \dot{\theta}_1 [s_2^2 (J_{2xx} + J_{3xx}) - 2s_2 c_2 (J_{2xz} + J_{3xz}) + c_2^2 (J_{2zz} + J_{3zz})] \qquad (3.14) $$

The robot's complete kinetic energy term may now be computed :

$$ KE = \frac{1}{2}\Big[\dot{\theta}_1^2 [m_1 (s_{11}^2 + s_{13}^2) + m_2 (d_2 + s_{22})^2 + m_3 (d_2 + s_{32})^2 + J_{1yy} $$

$$ + s_2^2 (m_2 s_{23}^2 J_{2xx} + J_{3xx} + m_3 (d_3 + s_{33})^2) $$

$$ + s_2 c_2 (2 s_{21} s_{23} - 2 (J_{2xz} + J_{3xz}) + 2m_3 (d_3 + s_{33})(a_3 + s_{31})) $$

$$ + c_2^2 (m_2 s_{21}^2 + m_3 (a_3 + s_{31})^2 + J_{2zz} + J_{3zz})] $$

$$ + \dot{\theta}_1 \dot{\theta}_2 [c_2 (2m_2 s_{23} (d_2 + s_{22}) - 2 (J_{2yz} J_{3yz}) + 2m_3 (d_3 + s_{32})) $$

$$ + s_2 (-2m_2 s_{21} (d_2 + s_{22}) - 2m_3 (a_3 + s_{31})(d_2 + s_{32}) + 2(J_{2xy} + J_{3xy}))] $$

$$ + \dot{\theta}_2^2 [m_2 (s_{23}^2 + s_{21}^2) + m_3 (a_3 + s_{31})^2 + J_{2yy} + J_{3yy} + m_3 (d_3 + s_{33})^2] $$

$$ + \dot{\theta}_2 \dot{d}_3 [-2m_3 (a_3 + s_{31})] + \dot{d}_3^2 [m_3] + \dot{d}_3 \dot{\theta}_1 [2m_3 s_2 (d_2 + s_{32})] \Big] $$

(3.15)

and the potential energy term is :

$$ PE = g [m_1 (s_{12} + d_1) + m_2 (s_{21} s_2 - s_{23} c_2 + d_1) $$

$$ + m_3 (s_2 (a_3 + s_{31}) - c_2 (d_3 + s_{33}) + d_1)] \qquad (3.16) $$

The elements of the Lagrange equations may now be computed and substituted into the overall expressions. Hence in the form :

$$ \underset{\sim}{H}(\underset{\sim}{q}) \ddot{\underset{\sim}{q}} + \underset{\sim}{V}(\underset{\sim}{q}, \dot{\underset{\sim}{q}}) \dot{\underset{\sim}{q}} + \underset{\sim}{G}(\underset{\sim}{q}) = \underset{\sim}{\tau} $$

$$ \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{d}_3 \end{bmatrix} + \begin{bmatrix} 0 & V_{12} & V_{13} \\ V_{21} & 0 & V_{23} \\ V_{31} & V_{32} & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \\ G_3 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \qquad (3.17) $$

where

$$ H_{11} = m_1 (s_{11}^2 + s_{13}^2) + m_2 [(d_2 + s_{22})^2 + (s_2 s_{23} + c_2 s_{21})^2] $$

$$ + m_3 [(d_3 + s_{32})^2 + (s_2 (d_3 + s_{33}) + c_2 (a_3 + s_{31}))^2] $$

$$ + J_{1yy} + s_2^2 (J_{2xx} + J_{3xx}) - 2s_2 c_2 (J_{2xz} + J_{3xz}) + c_2^2 (J_{2zz} + J_{3zz}) $$

$$ H_{12} = m_2 (c_2 s_{23} - s_2 s_{21}) (d_2 + s_{22}) $$

$$ + m_3 (d_2 + s_{32}) (c_2 (d_3 + s_{33}) - s_2 (a_3 + s_{31})) $$

$$ - c_2 (J_{2yz} + J_{3yz}) + s_2 (J_{2xy} + J_{3xy}) $$

$$ H_{13} = m_3 s_2 (d_2 + s_{32}) $$

$$ H_{21} = H_{12} $$

$$ H_{22} = m_2 (s_{33}^2 + s_{21}^2) + m_3 [(d_3 + s_{33})^2 + (a_3 + s_{31})^2] + J_{2yy} + J_{3yy} $$

$$ H_{23} = -m_3 (a_3 + s_{31}) $$

$$ H_{31} = H_{13} $$

$$H_{32} = H_{23}$$

$$H_{33} = m_3$$

$$V_{12} = \dot{\theta}_1 .2\ [m_2\ (c_2 s_{21} + s_2 s_{23})\ (c_2 s_{23} - s_2 s_{21})$$
$$+ m_3\ (c_2\ (a_3 + s_{31}) + s_2\ (d_3 + s_{33}))\ (c_2\ (d_3 + s_{33}) - s_2\ (a_3 + s_{31}))$$

$$+ s_2^2\ (J_{2xz} + J_{3xz}) - s_2 c_2\ ((J_{2zz} + J_{3zz}) - (J_{2xx} + J_{3xx}))$$
$$- c_2^2\ (J_{2xz} + J_{3xz})]$$
$$+ \dot{\theta}_2\ [-m_2\ (c_2 s_{21} + s_2 s_{23})\ (d_2 + s_{22})$$
$$- m_3\ (c_2\ (a_3 + s_{31}) + s_2\ (d_3 + s_{33}))\ (d_2 + s_{32})$$
$$+ s_2\ (J_{2yz} + J_{3yz}) + c_2\ (J_{2xy} + J_{3xy})]$$
$$+ \dot{d}_3\ [2 m_3 c_2\ (d_2 + s_{32})]$$

$$V_{13} = \dot{\theta}_1 .2\ [m_3\ (s_2^2\ (d_3 + s_{33}) + s_2 c_2\ (a_3 + s_{31}))]$$

$$V_{21} = \dot{\theta}_1 .(-1)\ [m_2\ (c_2 s_{21} + s_2 s_{23})\ (c_2 s_{23} - s_2 s_{21})$$
$$+ m_3\ (c_2\ (a_3 + s_{31}) + s_2\ (d_3 + s_{33}))\ (c_2\ (d_3 + s_{33}) - s_2\ (a_3 + s_{31}))$$

$$+ s_2^2\ (J_{2xz} + J_{3xz}) - s_2 c_2\ ((J_{2zz} + J_{3zz}) - (J_{2xx} + J_{3xx}))$$
$$- c_2^2\ (J_{2xz} + J_{3xz})]$$

$$V_{23} = \dot{\theta}_2 .2\ [m_3\ (d_3 + s_{33})]$$

$$V_{31} = \dot{\theta}_1\ [-m_3\ (s_2^2\ (d_3 + s_{33}) + s_2 c_2\ (a_3 + s_{31}))]$$
$$(\ = -c_{13}/2\ )$$

$$V_{32} = \dot{\theta}_2\ [-m_3\ (d_3 + s_{33})]$$
$$(\ = -c_{23}/2\ )$$

$$G_1 = 0$$

$$G_2 = g\ [m_2\ (c_2 s_{21} + s_2 s_{23}) + m_3\ (c_2\ (a_3 + s_{31}) + s_2\ (d_3 + s_{33}))]$$

$$G_3 = g\ [\ -m_3 c_2\ ]$$

These are the coefficients of three, non-linear second order coupled differential equations.

3.5 Manipulator Feedback Control

Industrial robot feedback controller design has received a considerable amount of attention, yet it would appear that a selection of one particular approach above others is difficult to make. What is fundamentally needed is precise time coordinated trajectory tracking which implies a means of processing various error signals, and in turn minimising them. Although several organisations are striving to standardise manipulator performance tests, there is still no straightforward means of comparison between error results from one manipulator to another.

The controller requirement here is an important one. Without an effective controller, there is little point in enhancing the qualities of a robot motion. An attempt could be made to supply trajectory commands to the manipulator without any form of feedback. This works well in the ideal world of simulations, because all the parameters are precisely defined, and probably invariant. This is not found to be the case with a real robot.

A closed loop position control system is therefore a necessity. The actual output from the system is continually monitored and compared to the desired output. The corresponding error signals are generated and then used to reduce the errors. In general, a controller's performance can potentially be increased given an augmented knowledge of the plant states. How much information is available will usually depend on a cost / performance compromise. In the case of the 'Little Giant' robot, displacement potentiometers and differential pressure feedback transducers are fitted as standard. The latter are analogous to acceleration feedback, but non-linear because of the robot's dynamics.

Feedback introduces a new problem, that of stability. Closing of the feedback loop can result in a continual 'hunting' about a static operating point - termed marginal stability. More serious, complete instability manifests itself as oscillations of axes, with increasing amplitude, or servoing away from the set point, until saturation occurs. Stability considerations thus form a large part of control system design.

There is a conflicting requirement to that of stability. Control system dynamics must enable the manipulator to respond quickly to any given command. This implies high gains in the controller, whereas stability generally requires low gains. For the fastest response with no overshoot, an equivalent to critical damping is needed. Because the dynamics of a robot vary, this cannot be attained without the controller likewise changing.

The standard feedback control system used by the 'Little Giant' is outlined in figure 3.2. This is an error proportional system with a stabilising differential pressure feedback. The gains are fixed and so chosen to be stable for the worst manipulator configuration and load.

The effect of this strategy is that the robot lags more than it needs to in any other configuration. Drives incorporating nominally flow proportional valves result in an integral term being added to the controller, in principle excluding steady state errors. The servo-valve's integral action works on the total control signal, so an independent integral term is not used. Hydraulic cylinder and servo-valve dynamics, as shown in chapter 5, are considerably more complex than D.C. drives and amplifiers.

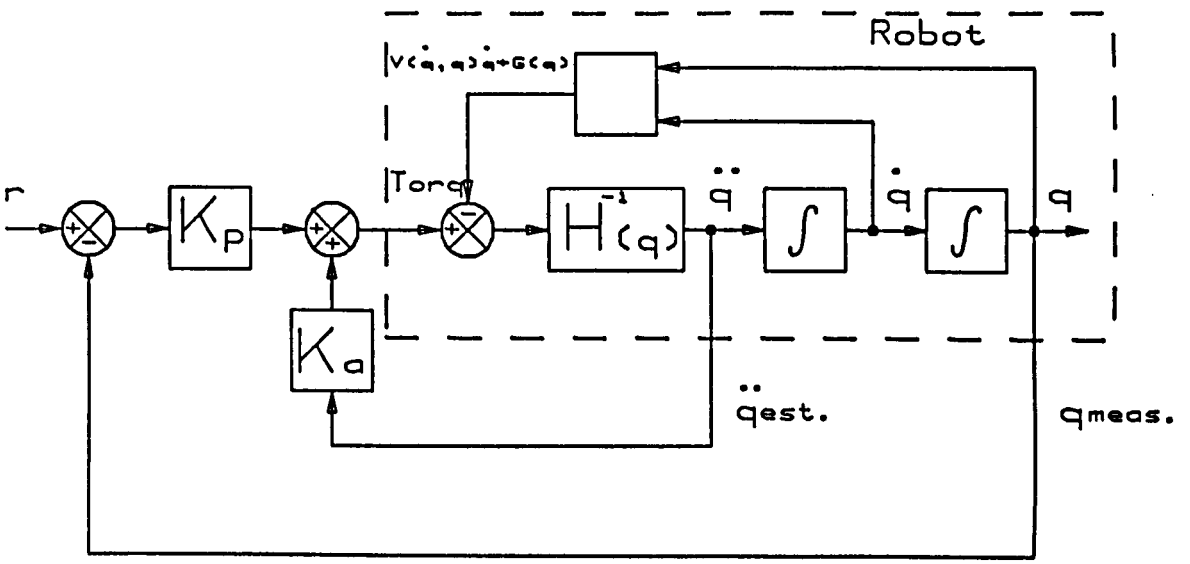


Figure 3.2 The Little Giant - Nominal Feedback Controller

It is a matter for debate as to whether trajectory synthesis is part of, or separate from, feedback control. In this project, feedback control is considered separately from trajectories, these being treated in chapter 4. Several of the well known robot 'control' works are entirely related to trajectory generation or kinematic solutions. Such a classical work is the Resolved Motion Rate Control of Whitney 1969 [3.9]. In this, the relationship between joint and world velocities is shown to be linear. Although this has implications in feedback control, it is really a kinematic observation.

Certain classes of trajectory command cause an oscillatory response. Controllers can be used to moderate such undesirable commands. Many workers have looked at the problem of control of compliant systems, but solutions appear of academic interest only. Simple mechanisms have been considered, the solutions are complex, and need precision

modelling as most of the compliant effects are outside of the control loop.

The problems to contend with in the control of manipulators, include :

(i) The complex nature of the dynamic system. This can be seen

in the equations of motion, section 3.4, which are non-linear with large changes in inertia, velocity product and gravity moments. Link cross coupling further complicates the system.

(ii) Parameter changes which are usually unknown, such as variations in the mass of manipulated objects; wear of manipulator components or tools used; other non-linearities such as backlash; static and dynamic bearing friction; torque outputs due to fluctuations in available power (hydraulic system flow) etc.

(iii) Disturbances e.g. external loads, tool forces, signal noise and vibration due to structural compliance.

(iv) Restrictions imposed on the controller by computational speed may influence the complexity or performance of the desired control algorithm.

3.6 Manipulator Control Survey

An exhaustive survey would be both impractical and impossible. The sheer volume of work precludes this. The objective here is to be aware of the controllers used in manipulators and the issues involved, so as to consider possible control strategies.

3.6.1 Linear State Feedback

Modified linear controllers have been developed to compensate for specific, modelled components of the equations of motion. Added to these compensation terms are typically conventional PID type control terms. Linear state feedback controllers can be used (figure 3.3), but because of continual parameter changes in the linearised model, the plant matrix and input map must be continually re-computed or

estimated. The schemes will be sensitive to disturbances. Specific robot designs can simplify these feedback control systems. As already outlined, invariant inertia matrices, zero gravitational vectors (balancing), and small centrifugal and Coriolis components can be obtained by design, thus simplifying the control.

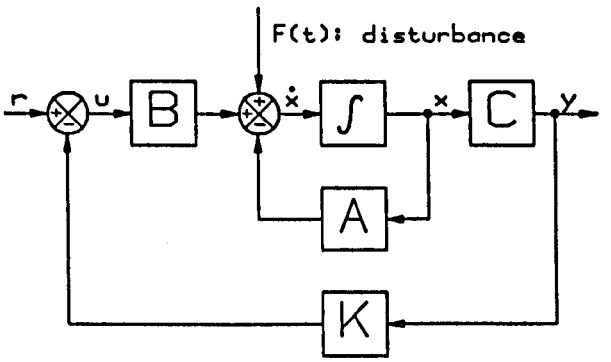


Figure 3.3 State Feedback Control System

From the dynamic equations of motion, it appears clear that coupling between links is important, yet some studies suggest that there is little if anything, gained in the use of multi-variable or 'centralised' control as opposed to de-centralised or single joint control, e.g. Golla, Garg and Hughes 1981 [3.10]. Assuming a linearised model is available, classic linear state feedback can be employed. The system or plant (subscript p) is modelled as the set of first order state equations :

$$\dot{\underset{\sim}{x}}_p = \underset{\sim}{A}_p \, \underset{\sim}{x}_p + \underset{\sim}{B}_p \, \underset{\sim}{u}_p \qquad (3.18)$$

$$\underset{\sim}{y}_p = \underset{\sim}{C}_p \, \underset{\sim}{x}_p \qquad (3.19)$$

and the feedback element may be :

$$\underset{\sim}{u}_p = \underset{\sim}{K} \, (\underset{\sim}{r} - \underset{\sim}{y}_p) \qquad (3.20)$$

hence

$$\dot{\underset{\sim}{x}}_p = (\underset{\sim}{A}_p - \underset{\sim}{B}_p \, \underset{\sim}{K} \, \underset{\sim}{C}_p) \, \underset{\sim}{x}_p + \underset{\sim}{B}_p \, \underset{\sim}{K} \, \underset{\sim}{r} \qquad (3.21)$$

where $\underset{\sim}{x}_p$ - state vector

$\underset{\sim}{A}_p$ - system coefficient matrix

$\underset{\sim}{B}_p$ - system input coefficient matrix

$\underset{\sim}{u}_p$ - control input vector

$\underset{\sim}{y}_p$ - system output (response) vector

$\underset{\sim}{C}_p$ - system output coefficient matrix

$\underset{\sim}{K}$ - feedback gain matrix

$\underset{\sim}{r}$ - motion input vector function

and so this closed loop equation may be analysed in a similar manner to the original state equation. If the $\underset{\sim}{A}_p$ and $\underset{\sim}{B}_p$ matrices are known to vary excessively then their elements may be continually estimated. These systems are described in control texts. Control systems developed using these forms of equations are described in this project.

3.6.2 Optimal Control

Optimal controllers have been developed for robots, maximising some performance criteria. An effective choice of weighting factors in these criteria is difficult to make. Optimisation of the trajectory for task time or energy consumption (including the rigid dynamics) has been performed, a classic scheme being that of Kahn and Roth 1971 [3.3]. Minimum time controls are furnished using Pontryagin's Minimum Principle and indicate optimal bang-bang trajectories. They point out that disturbances are not accounted for. A sub-optimal control is developed yielding similar results. Neither are of use here as they are effectively open loop and the proposed commands are seriously discontinuous. Optimal minimisation of errors is less common, probably because of the extra complexity introduced by the controller.

One scheme which avoids this problem is Goor 1984 [3.11], who provides 'a new approach to minimum time robot control'. In this he contends that robot dynamics is significantly affected, if not dominated by motor dynamics. The effect of his analysis is to reduce the joint error dynamics to third order linear, with constant coefficients. This enables a precise and simple feedforward control to be computed. His approximations appear to hold in the case of the Puma robot. Many have commented on the high reflected motor inertias of this robot in comparison to the links, due to gear ratios of around 100 to 1. To assess whether this feature can be employed on other robots, would require a substantial and specific experimental analysis.

3.6.3 Active Force Control

Hewit and Burdess 1981 [3.12] reduce the manipulator to a set of decoupled second order systems by measuring the torque applied at each joint and the resultant link accelerations. Knowing the mass/inertia

matrix, the non-linear velocity product terms are computed using the difference between the torque and the mass acceleration product. These terms may then be used for accurate compensation. This is much simpler computationally than trying to evaluate all these terms separately. The scheme must also cope well with disturbances from any source (other than measurement noise) by virtue of the measurement process. This feature is verified in a later experimental investigation, Galatis, Hadzistylis and Hewit 1986 [3.13]. One complication in the earlier scheme is that computations are performed in Cartesian coordinates. This is related to the problem of acceleration measurement. Simplification may be attained through the use of relative link acceleration transducers as opposed to seismic type absolute acceleration devices.

3.6.4 Computed Torque Control

Complete actuator torque or force computation seems a popular method in the US. Luh, Walker and Paul 1980 [3.5] use this. As opposed to the earlier work of Paul's, it is pointed out that except for friction, the terms which make up the joint forces, (namely inertia, centripetal, Coriolis and gravitation) give more or less equally important contributions in typical trajectories. A powerful mini-computer is required to carry out the calculations at a sufficient rate, although dedicated processors are now used. To reduce the volume of calculations, pre-evaluated functional relationships are stored in table look up form, Bejczy 1974 [3.14]. For a six degree of freedom manipulator, the storage space demanded is excessive and further, the results can be highly innacurate in certain regions. Another weakness of computed torque methods is that they still require a good feedback control function, to maintain the performance in the presence of disturbances, model errors etc.

3.6.5 Model Referenced Adaptive Controllers

Model following controllers have specific advantages in this project. This is especially true if the model form is much simpler than the real system dynamics. If good model following is possible, motion commands may easily be pre-processed to result in the desired responses. This characteristic is utilised in chapter 7.

Liegeois, Fournier and Aldon 1980 [3.15] present a Model Reference controller, which they compare to a conventional P-D controller. Errors produced are reduced tenfold to those of the P-D system alone. The manipulator used operates at up to 2 m/s tip velocity. Although the controller form is given, it is in outline only and requires clarification.

Dubowsky and Desforges 1979 [3.16] employ another Model Referenced Adaptive Control System (MRACS), justifying its use on the basis of the following :

(i) Manipulator dynamics are time and position dependent.
(ii) A wide range of command inputs are to be expected.
(iii) Substantial uncertainty in the system characteristics is introduced by unknown payload mass properties.
(iv) The reference model can be chosen to give desired closed loop characteristics.
(v) The control computer works with the simple model, rather than the complex non-linear equations of the actual system.

The method used to develop the system is the pioneering work of Donalson and Leondes 1963 [3.17] and does not incorporate any of the refinements of the design procedure presented by Landau 1974 [3.18]. A high standard of performance is attained all the same.

Another MRACS is proposed and simulated by Horowitz and Tomizuka 1980 [3.19]. Results imply the system to be insensitive to payload changes. As in many of these systems, simulation ensures that all the system's parameters are known precisely, thus design of equally precise controllers is straightforward. Insensitivity to payload changes though, does serve as a guide to potential performance in a practical system.

The forms of MRACS vary, but Landau's design procedure is systematic, assures good model following and stability. Stoten 1980 [3.20] applies it to manipulators in a simulation, producing similar results to Horowitz and Tomizuka. A version of Landau's scheme is shown in figure 3.4. The complete scheme applied in practice to the 'Little Giant' is described in chapter 7.

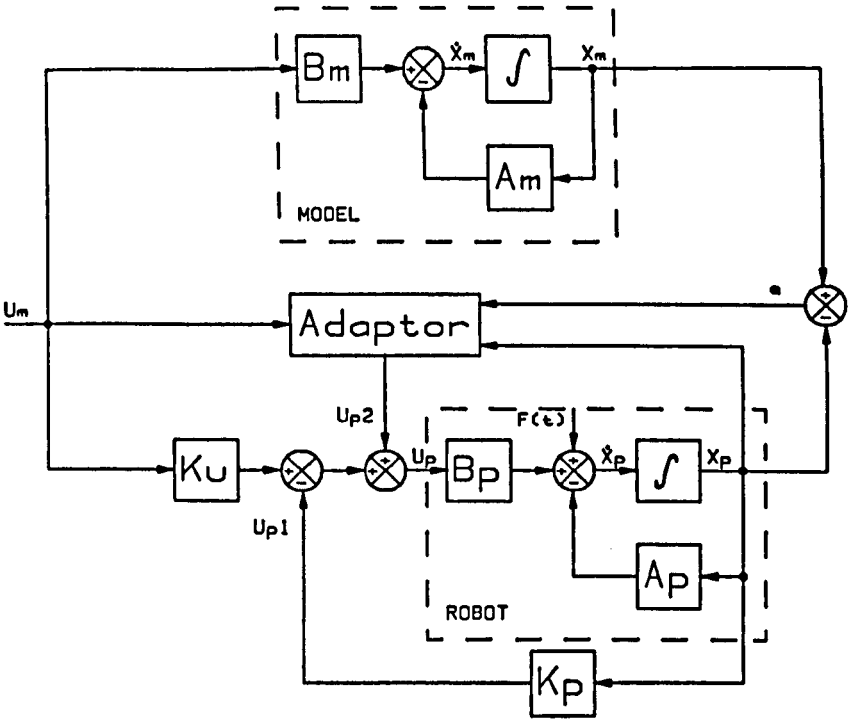


Figure 3.4 A Model Referenced Adaptive Control Scheme

Other forms of controller include the addition of self learning systems, which exploit the cyclic nature of most robot applications. Past response information may be processed efficiently and used to modify the trajectory displacement command. This in turn modifies the control input and so the response. These motion tuning systems are described in more detail in chapters 5 and 6.

4 Manipulator Trajectory Generation

A trajectory is defined here as the path taken by some manipulator end effector datum through its workspace. It is idealised to a set of mathematical functions of time which describe motions between displacement endpoints. These motions may actually be defined at the joints, the actuators or the end effector. The functions used are explicit, the independent variable being time and the dependent variable, some coordinate value. They are differentiable with respect to time and so the command velocity, acceleration etc. may be evaluated explicitly if required. The trajectories studied will be made up of concatenated segments with boundary conditions being specified at the segment ends, comprising displacements, velocities etc.

4.1 Trajectories - Literature Review

The majority of work on manipulator trajectories has been carried out under the following headings :

(i) Tracking and Cartesian Path Motions
(ii) Time or Energy Trajectory Optimisation
(iii) Obstacle Avoidance

Obstacle avoidance is only considered in this project, in that it is assumed that 'VIA' points have been supplied by the trajectory planner.

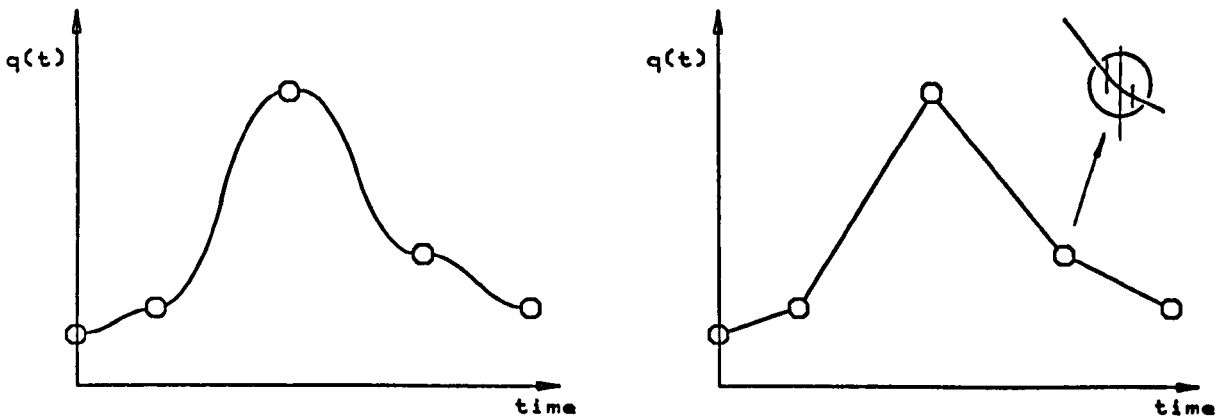
The field of cams provides a rich source of information on both motion laws and their vibratory responses. Little work exists on the effect of manipulator trajectories and motion laws, on the manipulator's response. Brady 1982 [4.1] had also noticed this void, pointing out 'Incorporating dynamics into trajectory planning is an open research area'.

4.1.1 Tracking and Cartesian Path Motions

Brady covers many aspects of trajectory planning, and serves as a useful reviewer.

Whitney 1969 [3.9] used the inverse Jacobian matrix to control motion along world coordinates, noting the linear relationship between joint and world velocities.

Taylor 1979 [4.2] describes two methods for the execution of straight line trajectories in Cartesian space. The first is a refinement of Paul's method below. Trajectory coordinates are generated by interpolation between endpoints in Cartesian space with solution of the inverse kinematics at every coordinate. Taylor's second method, of more interest here, is to precompute a few intermediate points between the endpoints in Cartesian space but then carry out the interpolation in joint coordinates. This method will generate a 'ripple' error from the idealised path but the solution points are constrained to lie within pre-specified bounds of the idealised linear Cartesian path. This constraint is used to dictate the number of intermediate points.



(a) Segmented motion with stop/start (b) Piecewise linear approximation

Figure 4.1 Paul's Motion Generation Strategies

Paul 1981 [4.3] replaces continuous segmented motion with a piecewise linear approximation (figure 4.1). To eliminate velocity and acceleration discontinuities he then rounds off the corners by fitting a quartic between equispaced points in time either side of the corner. The associated coefficients are then solved to produce the motion law. Because of the transition symmetry the 6 boundary conditions result in only a quartic polynomial. He specifies the solution evaluation frequency as ten times the structural natural frequency of the manipulator. Another restriction is the required tracking accuracy. Simplified linear interpolation routines are used to decrease the sampling interval to satisfy these constraints.

4.1.2 Cams

The synthesis of tuned cams, working backwards from the desired motion through the system dynamics to the cam profile, is covered in several texts. Stoddard 1953 [3.1] presents a complete design procedure for so called 'Polydyne' cam laws. The motion laws are polynomial based, the dynamic systems considered are simple linear mass spring models.

The optimisation of cam profiles to minimise residual vibration is developed by Kwakernaak and Smit 1968 [4.4]. Cam laws with finite jerk are shown to give transient responses which are primarily functions of the period ratio.

4.1.3 Trajectory Optimisation

Minimum time trajectories can be shown to be bang-bang through the use of Pontryagin's Minimum Principle. A trajectory in which maximum force is applied to accelerate and then decelerate is termed bang-bang. A survey of trajectory models carried out by Mutjaba 1977 [4.5] proves of interest. For normalised parameters the fifth degree polynomial, a cosine and a sine added to a linear ramp were shown to be 10-20% slower than a bang-bang law.

Unfortunately the bang-bang cannot be used as it excites excessive vibrations and will result in early component failure. Bang-bang trajectories were also obtained in the classic work of Kahn and Roth 1971 [3.3] which included up to three instantaneous torque reversals which would be even less desirable. It should be noted that they, along with Vukobratovic and Kircanski 1982 [4.6] who presented near bang-bang optimal energy trajectories used rigid dynamic models for their responses. The undesirable dynamic properties of the motion laws were therefore conveniently not manifest.

The design and effects of tuning the multi-harmonic law to reduce vibrations is treated by Rees Jones 1977 [4.7]. The predecessor project to the work of this thesis is described by Rees Jones, Fischer and Rooney 1984 [4.8]. In this work, hardware and software for a single d.o.f. hydraulic rig, driving a closed loop free oscillator under stop-go-stop conditions was developed. The ability of the tuned

approach combined with an adaptive cyclic modification algorithm to reduce vibrations was demonstrated. The driver system was considered as if it were a flexible cam, with time as the independant variable. The specification of the work was to study the transport of compliant loads rather than compliant driver systems. Dedicated motion generation facilities were expanded to drive the three planar axes of the 'Little Giant' robot.

4.2 Mathematical Trajectory Models

If our general trajectory coordinate set is $\underset{\sim}{q}$, an explicit function of time :

$$\underset{\sim}{q} = \underset{\sim}{q}(t) \qquad \underset{\sim}{q}(t) = [\, q_1(t),\ q_2(t), \ldots . q_n(t)\,]^T \quad (4.1)$$

for a multi-degree of freedom system, there are many functions of time which could be used to describe the trajectory. The needs of a trajectory model must be studied further to select any one in particular. These include :

(i) Solution of the function coefficients given boundary conditions of varying order, should be both flexible and simple.

(ii) The trajectory function should be mathematically simple to evaluate, enabling real time computation if necessary.

(iii) The solution should be able to take advantage of the massive data compaction obtained by storing function coefficients, rather than the entire trajectory as a look up table.

(iv) The trajectory must be an explicit function of time.

(v) The function and its low order derivatives at least, must be continuous and not possess unpredictable features such as meandering between end points.

(vi) Extreme values of functions should be capable of easy evaluation for the purposes of motion feasibility assessment.

The widest source of motion laws is probably the field of cams. Searching through several cams texts; all the laws can be classified within one of three groups :

(i) Polynomial Laws

(ii) Harmonic Laws

(iii) Combinations of (i) and (ii)

Other functions which could be used as trajectory descriptions include other trigonometric, hyperbolic and exponential functions and their inverses. Such functions will be neglected.

4.2.1 Harmonic Series

These are encompassed by the general form :

$$q(t) = \sum_{i=0}^{m} a_{1i} \cos a_{2i} t + b_{1i} \sin b_{2i} t \qquad (4.2)$$

where the a and b are constant coefficients defining the law. Studying individually the criteria from the list of trajectory requirements, (iii), (iv) and (v) are all satisfied. Solution for the function coefficient values (i), given boundary conditions is simple for low order series, but becomes complex for more than around 4 boundary conditions. Although the trajectory is mathematically simple (ii), it is computationally lengthy, requiring some 8500 x (m+1) machine execution cycles for evaluation of a single coordinates trajectory data point. The expression m+1 defines the number of cosine and sine pairs in the motion law. Extreme function values (vi) are difficult to compute and suffer from an infinite number of solutions.

4.2.2 Polynomial Series

The general form is :

$$q(t) = \sum_{i=0}^{n} c_i \, t^i \qquad (4.3)$$

where the c_i are the polynomial coefficients defining the law. Criteria (iii), (iv) and (v) are again broadly satisfied by the function. Solution for coefficients (i) is relatively simple and an algorithm is readily set up for nth degree polynomials. Initial condition specification for t=0 gives half the coefficient values directly as all other terms disappear on derivative evaluation. If the law is evaluated using floating point power routines, trajectory points will also require around 8500 x (n+1) cycles. A commonly used procedure is to nest the terms, thus :

$$q(t) = c_0 + t\,(c_1 + t\,(c_2 + \ldots + t\,(c_{n-1} + c_n\, t)\ldots) \qquad (4.4)$$

and the number of execution cycles drops dramatically to 600 x (n+1). Extreme function values (vi) suffer from similar problems to those of harmonic series. Possibly an answer in both cases is to evaluate the trajectory model with a simple searching pattern.

The multi harmonic possesses one particular characteristic not present in the polynomial, that is the frequency components are known and finite in number. This has implications in the design of trajectories for vibration avoidance. Sine/cosine look up tables could be used as a means of reducing the load of harmonic series evaluation, but these suffer from numerical problems.

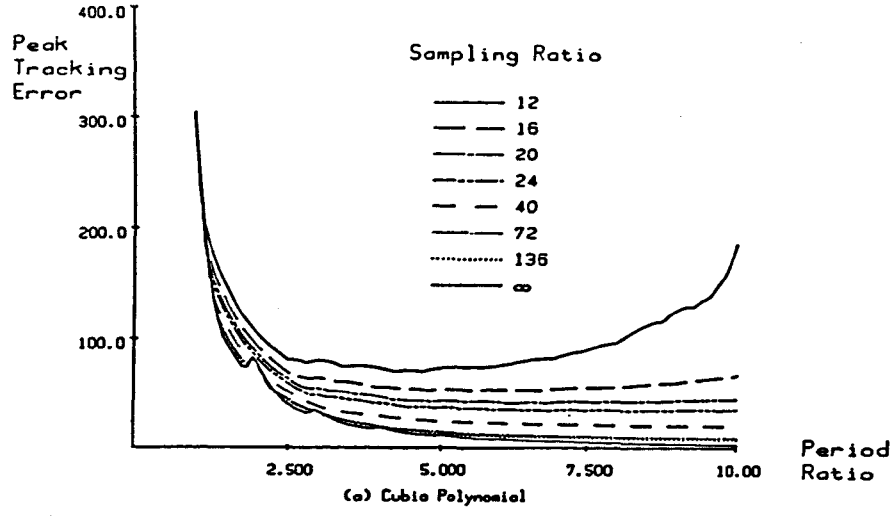
This qualitative assessment of the two general forms results in the following chart, an asterisk meaning an acceptable level of criteria satisfaction.

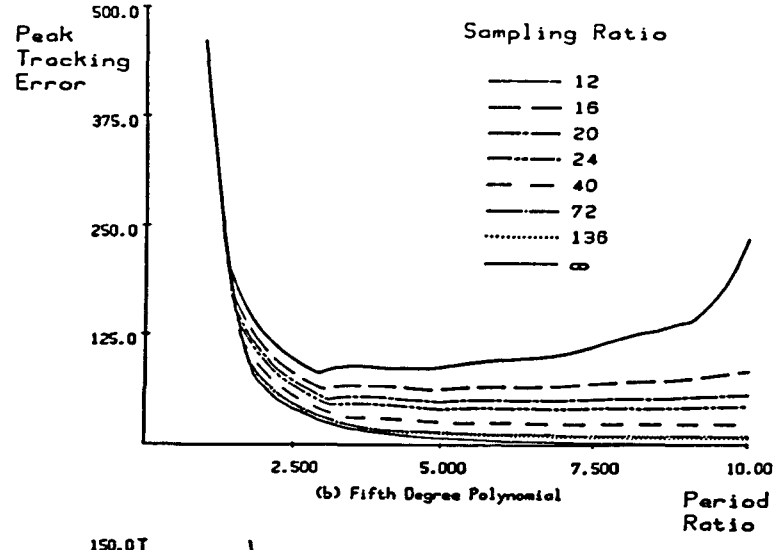
Criteria	Harmonic	Polynomial
(i) Boundary Condition Specification	-	*
(ii) Computational Simplicity	-	*
(iii) Data Compaction Facility	*	*
(iv) Explicit Function	*	*
(v) Continuous and Predictable	*	*
(vi) Extrema Evaluation	-	-

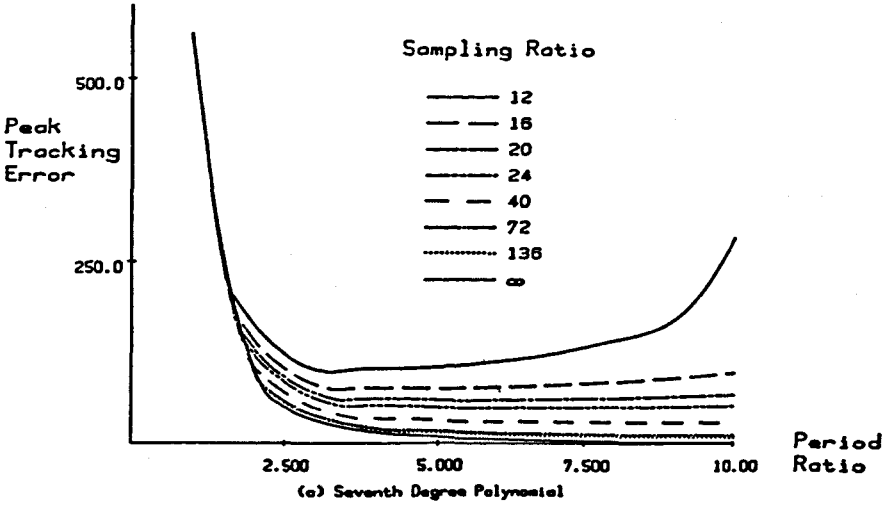
On this basis, polynomials are selected as the most suitable.

4.2.3 Polynomial Degree Versus Sampling Interval

In principle it is preferable to derive the required data sampling rate from various dynamic factors. The sampling rate should be related to system fundamental frequencies, for stability and adequate control. There is also little point in computing a motion law with a high order of smoothness, if the corresponding output is coarse because of low sample rates. In flight error will increase as a result. The effects on peak tracking error of various combinations of sample rate, motion duration, and system natural frequency are shown in figure 4.2. These are based on the second order undamped model response to normalised polynomial laws of degree 3, 5 and 7. The effect of period ratio is







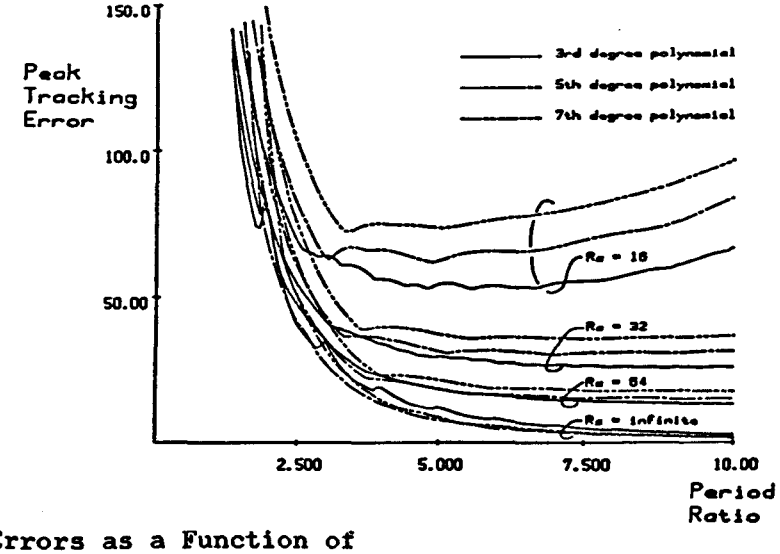


Figure 4.2 Peak Tracking Errors as a Function of Sampling Frequency and Period Ratio

Sample Ratio = $T/\delta t$ Period Ratio = $\omega_n T/(2\pi)$

well known, but variations due to sampling rate and polynomial degree are not so predictable. Apparent discontinuities are due to the individual curves being made up of the concatenated peak regions of many continuous curves.

4.3 Polynomial Motion Laws

4.3.1 Selection of Boundary Conditions

The number of possible boundary conditions which could be imposed on polynomials is infinite, but the following constraints limit the range :

(i) Zeroth order time derivatives must be specified at both ends of the motion segment.
(ii) The time derivative orders must be in an integer arithmetic series of step size one.
(iii) The highest integer specified in the series is 4. This corresponds to the fourth time derivative, impulse.

The only possible boundary condition combinations which satisfy these constraints are set out in the table 4.3, below. These will produce polynomials from the first to the ninth degree, and to abbreviate the boundary conditions :

d - displacement
v - velocity
a - acceleration
j - jerk
i - impulse

The combinations most likely to be used are those with symmetric boundary conditions, marked with an '*'. The horizontal dash is the divisor between initial and final conditions. There is nothing implied in the relative positions of the sets of conditions, so for example the upper conditions may be initial or final ones.

Polynomial Degree	1	2	3 (i)	3 (ii)	4 (i)	4 (ii)	5 (i)	5 (ii)	5 (iii)	6 (i)	6 (ii)	7 (i)	7 (ii)	8	9
	*			*					*				*		*
	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d
	d	v	v	v	v	v	v	v	v	v	v	v	v	v	v
		d	a	d	a	a	a	a	a	a	a	a	a	a	a
			d	v	j	d	j	d	d	j	j	j	j	j	j
					d	v	i	v	v	i	d	i	d	i	i
							d		a	d	v	v	a	d	d
										v	a	a	j	a	v
														j	a
															j
															i

Table 4.3 - Practical Boundary Condition Combinations

With the coefficients extracted, the relevant time derivatives of up to 9th degree polynomial terms are :

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
d	1	t	t^2	t^3	t^4	t^5	t^6	t^7	t^8	t^9
v	0	1	2t	$3t^2$	$4t^3$	$5t^4$	$6t^5$	$7t^6$	$8t^7$	$9t^8$
a	0	0	2	6t	$12t^2$	$20t^3$	$30t^4$	$42t^5$	$56t^6$	$72t^7$
j	0	0	0	6	24t	$60t^2$	$120t^3$	$210t^4$	$336t^5$	$504t^6$
i	0	0	0	0	24	120t	$360t^2$	$840t^3$	$1680t^4$	$3024t^5$

Table 4.4 Polynomial Time Derivatives

Note that due to differentiation, the numeric coefficients vary widely from 1 to $\approx 10^3$, in tandem with the powers of t going from 1 to 9. These terms are used as part of the solution of a system of equations, so care must be taken to avoid ill conditioning.

4.3.2 Polynomial Coefficient Solutions

Given boundary conditions as shown in the table, along with the corresponding time values, solutions can be found for the coefficients either numerically or algebraically. Algebraic solutions are not really feasible in a general solution, but given the restricted range and form of solutions required provide useful benefits. These include a high speed of solution (important for real time control) and the ability to set up solutions to minimise the ill conditioning problem. Some patterns are obvious in the derivatives and can, by induction, be described by the following expressions. If the polynomial motion for each coordinate is given by :

$$q(t) = \sum_{i=0}^{m} c_i t^i \qquad (4.5)$$

then a general derivative can be described as :

$$\frac{d^j}{dt^j} q(t) = \sum_{i=j}^{m} \frac{i!}{(i-j)!} c_i t^{i-j} \qquad (4.6)$$

or alternatively :

$$\frac{d^j}{dt^j} q(t) = \sum_{i=j}^{m} c_i \left[\prod_{k=i-j+1}^{i} k \right] t^{i-j} \qquad (4.7)$$

These are useful in evaluating derivatives which are not specified as boundary conditions. They are also found useful in the motion tuning of chapter 5. Algebraically the latter form appears more complex, but it is attractive for computational evaluation. Boundary conditions each comprise a value of :

$$\frac{d^j}{dt^j} q(t)$$

for a specified value of t. The derivative order j is an integer 0 to 4, and if the polynomial degree is m then the total number of boundary conditions is m+1. So it is easy to show :

$$\frac{d^j}{dt^j} q(t) = 0 \quad \text{for } j > m \qquad (4.8)$$

which again assists in polynomial evaluation at high order derivatives. If the segment of motion starts at time T_1 and ends at T_2, these can be initial or final conditions. This reduces the number of solutions required from 25 to 15, but for computation the reverse boundary condition laws must be evaluated with 'backwards time'. All the components are now available to write the whole system of equations in the form :

$$[\underset{\sim}{b}\ (d_1, d_2, v_1, \ldots)] = [\underset{\sim}{T}\ (T_1, T_2, T_1^2, T_2^2, \ldots)]\ [\underset{\sim}{c}\ (c_0, c_1, \ldots c_n)]^T$$

Displacement and derivative boundary conditions	Time boundary cond-itions, derivative coefficients and powers	Polynomial coefficients

(4.9)

which can be written in the form :

$$\underset{\sim}{b} = \underset{\sim}{T}\ \underset{\sim}{c} \qquad (4.10)$$

therefore :

$$\underset{\sim}{c} = \underset{\sim}{T}^{-1}\ \underset{\sim}{b} \qquad (4.11)$$

The key to the problem is therefore the inverse $\underset{\sim}{T}$ matrix. The computation of this is now considered in more detail. For wider flexibility in boundary condition specification, all the terms in the $\underset{\sim}{T}$ matrix could be numerically evaluated to suit. It would then be inverted for that specific case. This approach is enlarged upon by Rees Jones et al 1984 [4.8]. Given the simple constraints on the boundary conditions mentioned above, the number of permutations is limited, so for the restricted cases, pre-computed matrix inverses are generated. Final coefficient solutions are presented here, based on 'MATHLAB' solutions. 'MATHLAB' is an algebraic manipulation package written in LISP, available on Liverpool Polytechnic's DEC 2065

mainframe. If solutions are required for general time boundary conditions T_1 and T_2, then practical limitations of the package limit solutions to 6 x 6 matrices. Algebraic complexity of the $\underset{\sim}{T}$ matrix is greatly reduced if the solution is based on time boundary conditions 0 and T. This stipulation is found not to reduce the effectiveness of the solutions at all, all that is needed is an extra 'dummy' time variable. The maximum 10 x 10 matrix solutions could now be found. Substantial manual simplification was required because of MATHLAB's limited simplification ability. The 15 matrix equation solutions are set out in appendix A. Note that the numerical ill-conditioning problem mentioned above does not arise, because the solutions are explicit. These solutions are built into several of the programs developed in the sequel. POLY.PAS (appendix B) includes the full set.

4.3.3 Manual Motion Generation

The scheme comprising POLY.PAS, GENMO.PAS and RUN.PAS provides a complete environment for the generation of multi-axis, multi-segment, motions utilising polynomials of degree 1 to 9. All the logical possibilities of boundary condition combinations are available. POLY.PAS provides motion selection, boundary condition specification and polynomial coefficient solution facilities. The data file generated by it can then be further processed by GENMO.PAS. For compatibility, any command data (from the above command options) can be selected as options, whilst GENMO.PAS is run. The data file output from GENMO.PAS is now suitable for loading into memory using the RUN.PAS program. This interprets the command and motion data in real time, providing all the computation for real time control and self learning purposes.

4.4 Semi-Automatic Motion Generation

4.4.1 A Basic Motion Automation Language

Fully automated motion generation was not required for this work. For motion generation purposes, languages to date possess a range of features :

Synchronisation of motions (implicit in coordinated motion)
Interruption of motion (adapt to changing circumstances)
Sequencing with external events
Transformations to other coordinate sets
Varying trajectory paths (linear, circular)
Varying trajectory types (joint interpolated, polynomial)
Intermediate points (obstacle avoidance)
Peak velocity specification
Time dependency
Motion boundary condition specification

The last two items are rarely included in any robot programming system, but are pre-requisites of any dynamic motion development work. The work of chapters 6,7 and 8 also requires the ability to differentiate between a normal joint motion, a motion tuned for accuracy, and a motion tuned for speed. The requirement, as became clear from earlier chapters, was for a motion generation system which was versatile and easy to use. To this end, a motion generation system was developed, comprising motion specification, motion calculation and a real time controller (POLY.PAS,GENMO.PAS,RUN.PAS).

Secondly, some of the concepts developed could become difficult to use in any practical robot system, unless they were constructed within a real robot controller environment, and its inbuilt constraints. To keep sight of this fact, a simple, semi-automatic motion generating scheme was developed (TEACH.PAS,GAL.PAS,RUN.PAS). This did not need to be a complete system, simply those parts which would normally be associated with the motion generation side.

All these programs are included in appendix B. The complete set of programs was designed to be compatible as appropriate, at the various

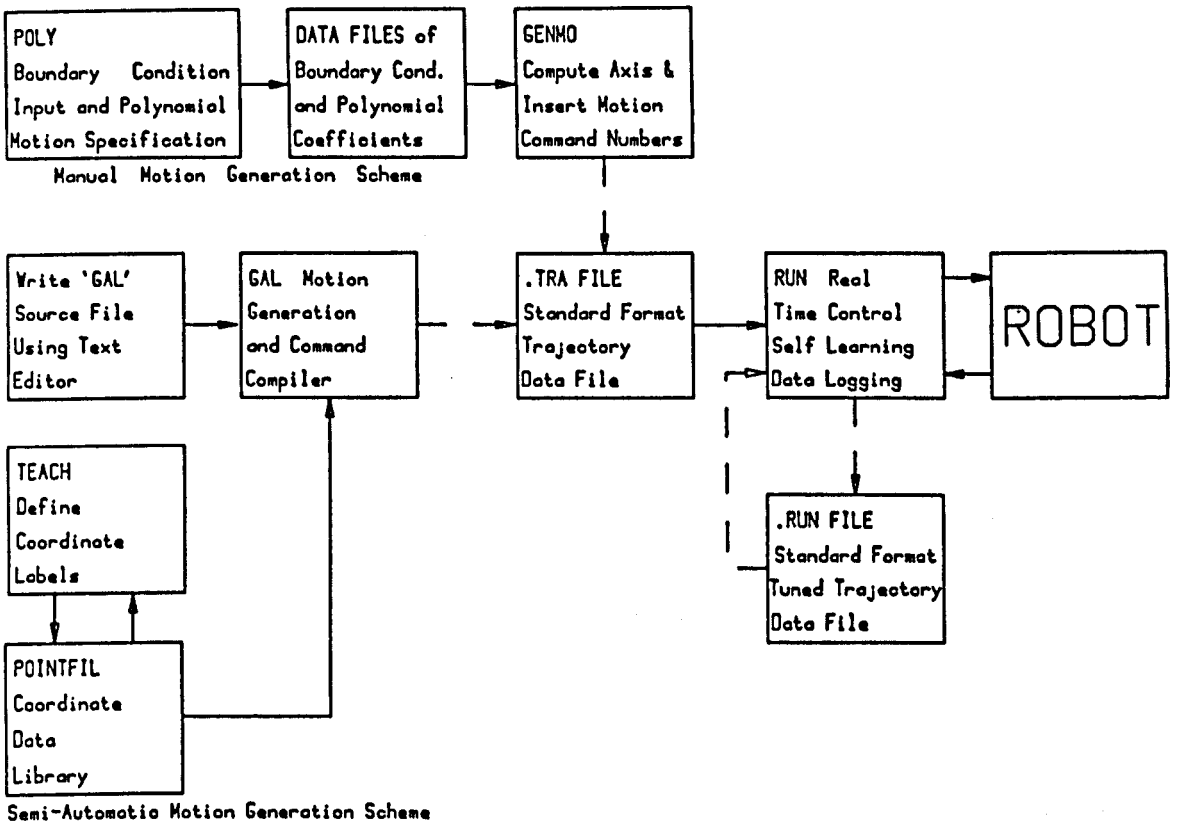


Figure 4.5 The Motion Generation and Control System Software

possible levels of interface. Figure 4.5 shows these inter-relationships. All data can be output to the graphics program which was developed to suit them, GGRAF.PAS.

To make the semi-automatic motion generating scheme versatile enough to cope with the wide range of tests likely to be carried out, it was considered that it should possess the following features :

(i) Motion durations should be defined, subject to them being a multiple of the sampling interval.
(ii) Various motion types must be permitted.
(iii) A means of building a library of coordinate data into a motion generator should be provided. This should include editing facilities. Data should be capable of being specified in world, joint and transducer coordinates.
(iv) A delay of arbitrary duration is required.

(v) Trajectory coefficients and data should be pre-computed to minimise the real time computation load.

(vi) Un-specified motion parameters should be estimated or defined automatically.

(vii) The default motion law used for segments should be a fifth degree polynomial.

(iix) It should be possible to specify intermediate (or 'VIA') points in a motion.

(ix) The robot interface and calibration must be in-built.

(x) A repeat loop would be a desirable feature for testing purposes.

(xi) The complete set of motion and response data should be recorded as unsigned integer data to minimise storage and processing requirements.

(xii) The scheme should be designed where possible for 'n' joints and numbers of motion segments, etc rather than a fixed number.

The schemes/programs which more or less satisfy the above requirements are defined in the following sub-sections.

4.4.2 TEACH.pas : A Coordinate Programming System

This provides an environment in which a library of coordinate sets with their associated identifiers, may be accumulated and edited. The library exists as a permanent file POINTFIL.SA, retained on hard disk. The coordinate data may be taught or entered off-line, in world, joint or transducer values. Executing TEACH produces the prompt '#' which allows entry of four commands; HERE, WHERE, REMOVE and EXIT.

HERE <point identifier>[,option] - The point identifier can be constructed from up to eight alphanumeric characters. These can be any of the full ASCII character set other than codes 0 to 32 and 128 to 160. The options mentioned above which can be used are :

,0 or nothing stated - Reads the current displacement transducer values, converts them to joint coordinates, checks they are in the allowable region, converts them to world and joint values, writes the whole set to the screen and files them in the library (assuming the identifier does not already exist in the library). The robot is moved

around its workspace by some other means than the computer keyboard.

,W - Prompts for off-line entry of the set of world coordinates defining the required location. The corresponding transducer and joint values are computed and displayed and filed under the point identifier.

,J - Prompts for the off-line entry of the set of joint coordinates defining the location. Corresponding world and transducer coordinates are computed, stored in the library and displayed on screen.

,T - Prompts for the set of transducer coordinates values defining the location. The other coordinate data is computed and filed.

The units used for coordinate data are metres, radians and transducer count (12 bit unsigned integer in this case).

WHERE <point identifier> - Tests to see if the specified point exists. If it does, the complete set of world, joint and transducer coordinate data defining it is displayed.

REMOVE <point identifier> - Checks to see if the point exists in the library. If it does, it is removed/deleted from the file.

EXIT - exits back to the computer systems operating system.

The library of coordinate identifiers and values is accessible to other programs. A typical entry in the library (POINTFIL.SA) is :

	TO_FIXTURE		
(world X, Y, Z)	1.5594007E-04	1.21954083E+00	1.51169777E+00
(joint θ_1, θ_2, d_3)	1.57066846E+00	1.95451307E+00	1.14486444E+00
(transducer)	3583	3548	3530

An additional feature of the TEACH environment is that it can be used in a 'calculator' mode. Coordinate data can be entered, and kinematic solutions for the remaining unknown coordinates is obtained directly. The robot kinematic solutions have been written into a separate module. They constitute the only part of the program which is robot specific.

4.4.3 GAL.pas : A Motion Programming System

GAL is the compiler, which converts a sequence of robot and motion commands into an intermediate code. This code is defined in the following sections and comprises a series of commands and coordinate data.

INITIALISE - initialises the interfacing software and hardware, performs basic checks, then moves the robot to the point called 'HOME', where it remains until there is activity at the keyboard. In the intermediate code it is defined as :

0 <Joint 1 value> <Joint 2 value>....<Joint n value> <CR>

Joint 'values' are the 12 bit transducer values described earlier. The next integer following will be a command number. <CR> represents the ASCII carriage return character, (ASCII code 13).

PAUSE[,option] - Holds the manipulator at the last specified point, or current position until there is activity at the keyboard. The options are :

,K - await keyboard activity

,P - await input line going high or true

If no option is specified, the default is ,K. The intermediate code is

1 <Joint 1 value> <Joint 2 value>.....<Joint n value> [,K option]

3 <M> <Joint 1 value> <Joint 2 value>..<Joint n value> [,P option]

DELAY <value> - the robot dwells wherever it is located at the time of the command. <value> defines the length of the delay in milliseconds. The actual delay will be the nearest multiple of the sample interval to the requested delay. The value is integer4, (allowing up to three and a half weeks' delay). Intermediate code is again simple, comprising :

2 <N> <Joint 1 value> <Joint 2 value>....<Joint n value> <CR>

where N represents the number of samples delay.

MOVE <point identifier>
[VIA <point identifier> (options)
.....
VIA <point identifier>]

Moves the robot from the current position (nominally that resulting

from the last command) to the first point identifier, passing through the sequence of via points which follow the command. The default is direct motion without any via points. The robot does not stop at via points, it simply passes through them. The motion coordination with time is fixed internally, MOVE implies track the default command, using the estimated time. The robot axes should possess zero velocity at the start and finish of a MOVE. The VIA command must start in the second or fourth columns. The intermediate code representation is :

4 <N> <M> <Joint 1 value> <Joint 2 value>....<Joint n value> <CR>
<Joint 1 value> .....<Joint n value> <CR>

etc., where the N value defines the number of samples tracked in total. The M value makes the command compatible with other commands. Its value is immaterial to this command.

FOR 1,<maximum loop counter value>
{series of statements}
ENDFOR

This gives a simple loop facility for repeating the series of statements enclosed, an integer number of times. The maximum loop counter value is integer4 ($> 2x10^9$). Caution needs to be taken in its use in two respects. (i) The manipulator commanded position at the start of the loop must be the same as that at the end of the loop. (ii) The total actual time taken for a single loop should be small, if there are any MOVE type commands which require data logging within the loop. This prevents excessive demands being made on system memory. The corresponding intermediate code is :

5 <N> <M> <.....commands or coordinate sets>

where N defines the number of sample intervals enclosed by the loop and M is the number of loops to be made. The whole command will be made up of N+3 integers in the loop.

SPEED <speed value> - An estimate of the minimum duration of any MOVE operation is made within the system. There is therefore a corresponding average speed for this particular motion. The integer2 value specified must lie within 1 to 100 and defines the percentage of this average speed at which the robot is to be operated. If the SPEED is not set, then a default of 50 is assumed. There is no intermediate code for it, as it is implicit in the motion data stored within the intermediate code, being used by MOVE etc.

MOVACC <point identifier>

[VIA <point identifier> (options)

.....

VIA <point identifier>]

This is the same as the MOVE command on its first run, but on subsequent runs, uses the past response data to redefine the motion command for improved tracking accuracy of the nominal motion, as described in chapter 6. The intermediate code is defined similarly to MOVE :

6 <N> <M> <Joint 1 value> <Joint 2 value>....<Joint n value> <CR>

.....

<Joint 1 value> <Joint 2 value>....<Joint n value> <CR>

M specifies the weighting, currently M = 0 defines MOVACC tuning for accuracy.

MOVFAS <point identifier>

[VIA <point identifier> (options)

.....

VIA <point identifier>]

As the MOVE command on its first run, but on subsequent runs, uses the past response data to redefine the motion command for reduced duration of the nominal motion, as described in chapter 8. The intermediate code is defined similarly to MOVE :

6 <N> <M> <Joint 1 value> <Joint 2 value>....<Joint n value> <CR>

.....

<Joint 1 value> <Joint 2 value>....<Joint n value> <CR>

M specifies the weighting, currently M = 100 defines MOVFAS tuning for speed.

END - Moves the robot to the position termed 'HOME' in the library where it remains until there is activity at the keyboard, indicating the robot power supply has been switched off. With power on, control is retained to hold the robot at the point specified. The intermediate code is :

7 <Joint 1 value> <Joint 2 value>....<Joint n value> <CR>

The program then allows various options for such as logging response and command data to disk files etc.

The requirement for several of these commands has been established in

hindsight. The command set is by no means complete for a robot language.

4.4.4 Computer Aided Program Generation

After experimentation using the manual motion generation scheme, it becomes evident that a number of the processes can be further automated. A substantial part of the process is the specification and structuring of the motion and associated commands. Entering all the coordinate and higher order boundary condition data is a tedious business. The coordinate and other data values can be replaced with identifiers which define locations in space. Continuity of displacement, velocity and acceleration can be automatically assured by cascading values through a motion. The order of motions and the types of motion used (MOVE,MOVFAS,MOVACC etc) can be changed rapidly. A brief analysis suggested the minimum requirements of a computer aided scheme (abridged robot language) are :

Text program entry
Basic syntax checking
String & file handling
Generation of real time motion data
Coordinate labels
Automatic duration estimation
Smooth, monotonic motions
Extrema checking
Basic loop facilities
Automatic boundary condition estimation

The complete compiler program giving these facilities is listed in appendix B. The limited facilities available are illustrated in the examples shown following the program, test1 through to test12 (appendix B). Test12 is shown in figure 4.6. One example of intermediate code is shown. This code is again entirely compatible with the same real time operating program; RUN.PAS. For historic reasons, the two systems were developed on two separate computers, an Apricot PC and a Motorola VME10. Serial communications were therefore developed between the two machines. In this way intermediate code could be generated on the VME10 using GAL.PAS and transferred to the

Apricot, where RUN.PAS resided. Differences in the two machine's versions of Pascal meant that it was difficult to transfer complete Pascal programs from one machine to the other. The conversion required added nothing to the project so was not carried out. Particular features of the computer aided motion generation system require more detailed consideration, specifically motion duration estimation, boundary condition estimation, extrema elimination (which would otherwise give rise to meandering of motion laws.)

```
      PROGRAM TEST12              Geoff Vernon; Dec 3rd 1985
INITIALISE                  set up the hardware & software
                            before run
PAUSE,K                     wait until activity at keyboard
                            before starting to move
DELAY 5000                  delay 5000 milliseconds
SPEED 20                    set the speed to 20% of the maximum
MOVE TO_PICKUP_POINT        track command to
                            location TO_PICKUP_POINT
SPEED 100                   set speed to maximum
PAUSE,P                     pause until line 1 of parallel i/o high
FOR 1,8                     loop the following statements 8 times
MOVFAS TO_ASSEMBLY          track, self learning, emphasis on speed,
   VIA AVOID_TOP_LATHE      to location TO_ASSEMBLY via the
   VIA AVOID_CEILING        following points in the list

MOVE TO_PICKUP_POINT        track command to location specified
ENDFOR                      end of the FOR loop

DELAY 1500                  delay 1500 milliseconds

SPEED 80                    slow down a little for the ACC move
MOVACC TO_BRACKET           move accurately, by learning,
                            to TO_BRACKET

FOR1,50                     loop 50 times, the following statements
MOVACC TO_ASSEMBLY          move accurately, by learning, to
   VIA AVOID_TOP_LATHE      TO_ASSEMBLY via locations
   VIA AVOID_CEILING        AVOID_TOP_LATHE & AVOID_CEILING

MOVACC TO_BRACKET           move accurately by learning to TO_BRACKET
ENDFOR                      end of FOR loop

PAUSE,P                     pause until line 1 of parallel i/o high

SPEED 100                   full speed
MOVE TO_FIXTURE             track command to TO_FIXTURE
MOVE TO_CONVEYOR_LOAD_POINT   track command to
                                  TO_CONVEYOR_LOAD_POINT

SPEED 10                    slow right down and
MOVE HOME                   move to pre-defined HOME position

END                         end of program
```

Figure 4.6 An Example 'GAL' Program

4.4.5 Basic Motion Duration Estimation

Motion duration control is not normally directly available, on existing robots. The programmer interacts with the robot system during

programming, adjusting the speeds of different parts of a task in order to achieve the required duration. An improved situation would permit the programmer to specify the motion or task duration, and the system would assess the feasibility of the selected value. For the system to propose an optimum value, dynamic models and iterative equation solutions etc would be required. An efficient method might be to use Hollerbach's time scaling. The dynamics would be solved for the motion just once. The peak torque/force values would be compared to the actuator capabilities. The new motion duration could then be computed directly, using the simple, scalar time scaling equations.

For the automated system, a conservative estimate of motion duration was crudely estimated by using a basic knowledge of the robot's behaviour.

4.4.6 Boundary Condition Estimation Based on Extrema Elimination

In single segment stop-go-stop motions, the boundary conditions are fixed, i.e. for axis displacement $q(t)$ for $0 \leq t \leq T$:

$$q(0) = d_1 \qquad q(T) = d_2 \tag{4.12}$$

$$\dot{q}(0) = v_1 = 0 \qquad \dot{q}(T) = v_2 = 0 \tag{4.13}$$

$$\ddot{q}(0) = a_1 = 0 \qquad \ddot{q}(T) = a_2 = 0 \tag{4.14}$$

In this simple case, the lack of additional constraints results in a symmetric polynomial, containing a single inflexion. There are no maxima or minima within the region of interest, the function is monotonic. These are in all desirable properties for a motion law. In a more general case, where the higher derivative values are non zero, the motion profile can meander and contain maxima and minima at various derivative orders. In multi-segment and multi-axis motions it is often desirable to use the velocity boundary conditions as a means of controlling the direction of motion. It is also established that matching velocities and accelerations in the transition from one segment to another will reduce transient vibrations. A complete motion generation scheme would be able to avoid the undesirable features in an automatic fashion. The basic reasons for avoiding extrema are the need to :

(i) Remain within the workspace limits

(ii) Avoid superfluous displacements, hence unnecessary accelerations

(iii) Avoid meandering and resultant undue vibration

In hindsight, an easier, more practical solution would be to use entirely data based motion shapes. These could be defined using a set of simple rules such as linear interpolation at various derivative orders (giving controlled monotonicity for motion segments), fixed peak accelerations etc.

Motion segments can be classified into four groups. These are :

(i) Complete or self contained stop-go-stop motion

(ii) Initial or starting motion segment

(iii) Intermediate motion segment

(iv) End motion segment

These are illustrated in figure 4.7. Given sufficient time, a complete system for the elimination of extrema could be accomplished, but was considered of low priority within this work. For all classes of polynomial it is clearly a large task. To understand the principal problems, just one case is considered in the sequel.

4.4.7 Polynomial Motion Extrema

The motion, or a segment of it is represented by the function :

$$q(t) = \sum_{i=0}^{m} c_i \, t^i \qquad 0 \le t \le T \qquad (4.15)$$

where T is the motion duration. A single point of inflexion may exist in a single segment motion, or the intermediate segment of a multi segment motion. In total, the number of inflexion points in the whole of any uni-directional piece of motion must not exceed one. Considering the fifth degree polynomial case :

$$q(t) = c_0+c_1 t+c_2 t^2+c_3 t^3+c_4 t^4+c_5 t^5 \qquad (4.16)$$

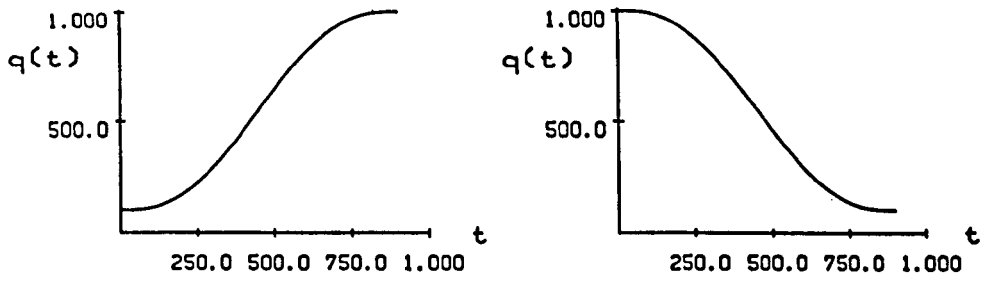
$$\dot{q}(t) = c_1+2c_2 t+3c_3 t^2+4c_4 t^3+5c_5 t^4 \qquad (4.17)$$

$$\ddot{q}(t) = 2c_2+6c_3 t+12c_4 t^2+20c_5 t^3 \qquad (4.18)$$

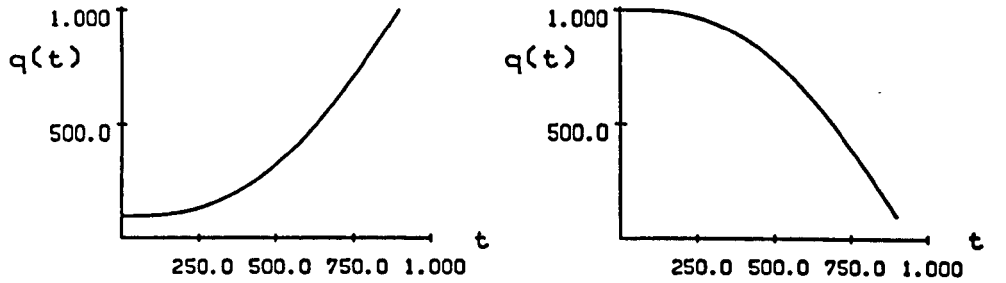
Conditions for extrema are :

$$\dot{q}(t_o) = 0 \begin{cases} \ddot{q}(t_o) < 0 & \text{for a maximum} \\ \ddot{q}(t_o) > 0 & \text{for a minimum} \\ \ddot{q}(t_o) = 0 & \begin{cases} q(t-\delta t) < q(t) > q(t+\delta t) & \text{maximum} \\ q(t-\delta t) > q(t) < q(t+\delta t) & \text{minimum} \\ q(t-\delta t) > q(t) > q(t+\delta t) & \text{inflexion} \\ q(t-\delta t) < q(t) < q(t+\delta t) & \text{inflexion} \end{cases} \end{cases} \qquad (4.19)$$

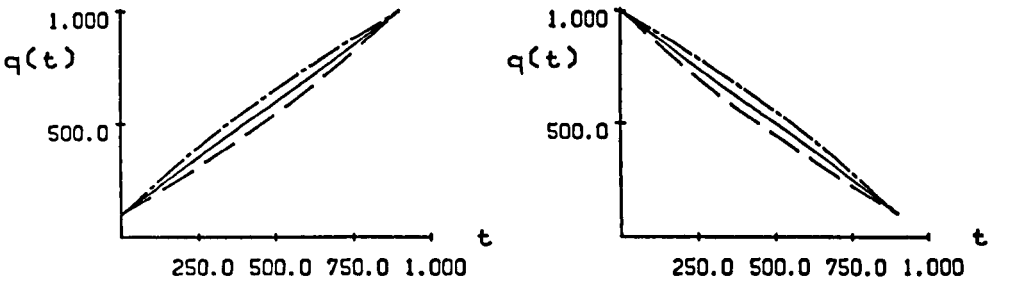
*(Note here that δt is being used as a small time interval, rather than the prescribed symbol for the sampling interval)



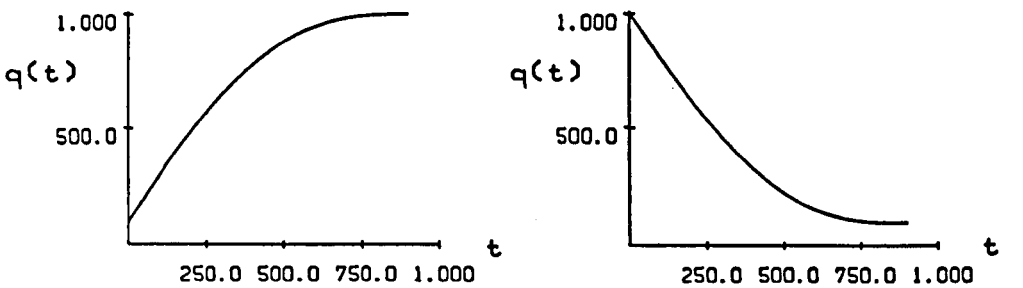
(i) Complete Stop-Go-Stop Motion



(ii) Initial Motion Segment



(iii) Intermediate Motion Segment



(iv) End Motion Segment

Figure 4.7 Motion Segment Classification

In this latter case where both the first and second derivatives vanish, the conditions are insufficient for an easy, mechanised solution. A strict assessment could use a Taylor series expansion. This enables the nature of a critical point to be determined when the first n derivatives vanish at such a point. In our case, the polynomials used generally exhibit 'nice' behaviour. A point of inflexion also occurs for :

$$\dot{q}(t) \neq 0 \text{ and } \ddot{q}(t) = 0 \quad \begin{cases} \ddot{q}(t-\delta t) < 0 & \ddot{q}(t+\delta t) > 0 \\ \text{or..} & \\ \ddot{q}(t-\delta t) > 0 & \ddot{q}(t+\delta t) < 0 \end{cases} \tag{4.20}$$

i.e. there is a change of sign in the second derivative. This inflexion is the only one which occurs in the case where all the maxima and minima are eliminated from the region $0 < t < T$. Extrema may exist within the entire function range. The requirement for a motion generation system is to avoid their presence within the region being used to represent the motion.

Considering the limited case of :

(i) Initial acceleration zero (hence $c_2 = 0$)
(ii) Final acceleration zero
(iii) Inflexions are permitted at the segment boundaries
(iv) One, of the (initial or final) velocities is zero

Given these conditions, there is a tractable, explicit root solution to enable the location of turning points.

In the case where $v_1 = a_1 = 0$, the turning points are at the solutions to :

$$\dot{q}(t_o) = (3c_3+4c_4t+5c_5t^2)t^2 = 0 \tag{4.21}$$

hence the non trivial case gives :

$$t_o = -\left[\frac{2c_4}{5c_5}\right] \pm \left[\left[\frac{2c_4}{5c_5}\right]^2 - \left[\frac{3c_3}{5c_5}\right]^2\right]^{\frac{1}{2}} \tag{4.22}$$

plus two roots at $t = 0$. For inflexions :

$$\ddot{q}(t_o) = (6c_3+12c_4t+20c_5t^2)t = 0 \tag{4.23}$$

hence

$$t_o = -\left[\frac{3c_4}{10c_5}\right] \pm \left[\left[\frac{3c_4}{10c_5}\right]^2 - \left[\frac{3c_3}{10c_5}\right]^2\right]^{\frac{1}{2}} \tag{4.24}$$

plus one root at $t = 0$.

For an initial segment represented by the fifth degree polynomial :

$$\begin{array}{lll} d_1 < d_2 & & \\ v_1 = 0 & v_2 > 0 & \\ a_1 = 0 & a_2 = 0 & \end{array} \qquad (4.25)$$

the average velocity is $v_{av} = \frac{(d_2-d_1)}{T}$ (4.26)

The coefficient solutions are (from appendix A) :

$$\begin{aligned} c_0 &= d_1 \\ c_1 &= 0 \\ c_2 &= 0 \\ c_3 &= \frac{20(d_2-d_1)-8v_2T}{2T^3} \\ c_4 &= \frac{-30(d_2-d_1)+14v_2T}{2T^4} \\ c_5 &= \frac{12(d_2-d_1)-6v_2T}{2T^5} \end{aligned} \qquad (4.27)$$

To be monotonic increasing :

If for all $q_1, q_2 \in R$ such that $q_1 < q_2$, we have $f(q_1) \leq f(q_2)$

or in the region of interest

$$\lim_{\delta t \to 0} \quad q(t+\delta t)-q(t) \leq 0 \text{ for } d_1 \leq q(t) \leq d_2$$

One way in which to assure this would be to set c_3, c_4 and c_5 all greater than zero. Setting this as a constraint defines the bounds on the segment final velocity as :

$$\begin{aligned} v_2 &< 2.5\frac{(d_2-d_1)}{T} && \text{for the } c_3 \text{ condition} \\ v_2 &> (15/7)\frac{(d_2-d_1)}{T} && \text{for the } c_4 \text{ condition} \\ v_2 &< 2\frac{(d_2-d_1)}{T} && \text{for the } c_5 \text{ condition} \end{aligned} \qquad (4.28)$$

These conditions are conflicing and cannot all be satisfied. A set of displacement curves are shown in figure 4.8. All three cases above are depicted, indicating that the resultant motions are clearly usable. For the specific example :

Example 1

$$\begin{array}{lll} d_1 = 1.145 & d_2 = 1.225 & T = 1.55 \\ v_1 = 0 & v_2 = 0.168 & v_{av} = 0.0516129 \\ a_1 = 0 & a_2 = 0 & \end{array}$$

hence

$$\begin{aligned} c_3 &= -0.0648786 \\ c_4 &= 0.1079 \\ c_5 &= -0.0336663 \end{aligned}$$

roots occur at $t_o = 0.58396, 1.9800$

$$\ddot{q}(t) = 2t(10c_5t^2+6c_4t+3c_3)$$

which for the root in range which is $\ddot{q}(t_o) = 0.0801365$ (i.e. a min.)

Example 2

$d_1 = 1.45$ $d_2 = 0.1$ $T = 1.45$
$v_1 = 0$ $v_2 = -3.25862$ $v_{av} = -0.93103$
$a_1 = 0$ $a_2 = 0$

hence

$c_3 = 10.62774$
$c_4 = -12.06309$
$c_5 = 3.47518$

roots occur at $t_o = 0.7326$, 1.7163 and $\dddot{q}(t_o) < 0$ (i.e. a max).

Solutions for both the possible inflexion points are :

$t_o = 0.1910$, 1.6456

For the conditions of example 2, with varying the final velocity, the resultant displacement curves are shown in figure 4.8. In the region :

$$2v_{av} \leq v_2 \leq 2.5v_{av}$$

there appear to be desirable characteristics. The coefficients are :

for $v_2 = 2v_{av}$ $c_3 = \frac{2(d_2-d_1)}{T^3}$ $c_4 = \frac{-(d_2-d_1)}{T^4}$ $c_5 = 0$

for $v_2 = 15v_{av}/7$ $c_3 = \frac{10(d_2-d_1)}{7T^3}$ $c_4 = 0$ $c_5 = \frac{-3(d_2-d_1)}{7T^5}$

for $v_2 = 2.5v_{av}$ $c_3 = 0$ $c_4 = \frac{5(d_2-d_1)}{2T^4}$ $c_5 = \frac{-3(d_2-d_1)}{2T^5}$

If polynomials are retained, it appears certain there is much to be gained from further study of these conditions. In the more general case, root solutions are unlikely to be explicit, requiring the use of numerical root searching routines.

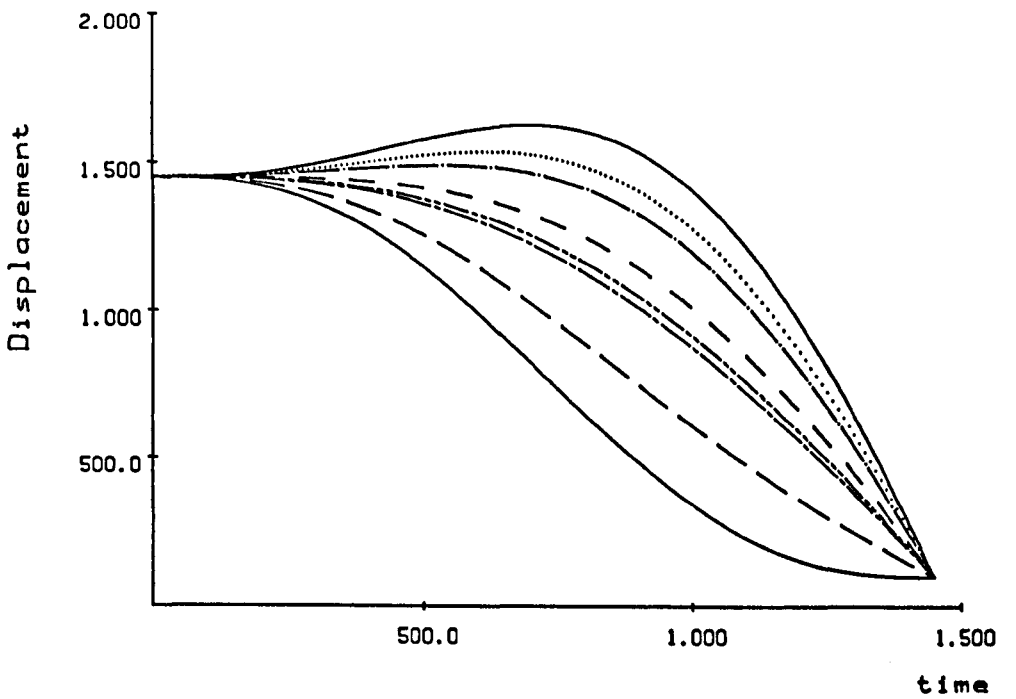


Figure 4.8 The Effect of Boundary Conditions on the Turning Points of Polynomials

Data Corresponding to Figure 4.8 :

Curve		Final Velocity	Roots 1	Roots 2	Inflexions 1	Inflexions 2	Note
————	1	0	2.9725	-0.0725	1.3659	0.8091	$v_2 = 0$
— —	2	-0.93103	3.54	-0.4467	1.6558	0.6642	$v_2 = v_{av}$
—·—	3	-1.86207	584125.9	-945419.9	233325.7	-504296.2	$v_2 = 2v_{av}$
—··—	4	-1.99507	4.2051	-4.2052	2.1026	-2.1026	$v_2 = 15v_{av}/7$
— —	5	-2.32759	1.9333	0	1.4500	0	$v_2 = 2.5v_{av}$
—·—	6	-2.97910	1.8696	0.5148	1.6417	0.1466	$v_2 = 3v_{av}$
.....	7	-3.25862	1.7163	0.7326	1.6456	0.1910	$v_2 = 3.5v_{av}$
————	8	-3.72414	1.3670	1.1463	1.6428	0.2422	$v_2 = 4v_{av}$

In this chapter, it is assumed that the control function $\underset{\sim}{F}$ is fixed and the motion command $\underset{\sim}{r}$ is to be varied to attain the desired response, $\underset{\sim}{y}$. This can be termed 'tuning' the trajectory. Tuning can be thought of as an off-line (i.e. not computed in real time); feedforward compensation technique. The performance criteria of interest are the tracking error and motion duration.

Conventional feedback control produces adequate tracking accuracy at low speed. It can also give the highest static accuracy possible, for the given system. At high speeds, compensation terms which work well at one configuration or speed may accentuate errors at another. More powerful controllers, which can adapt to the changing state, give improved performance by varying apparent stiffnesses with additional control input.

5.1 Dynamic Model Based Tuning Methods

Referring back to the dynamic model of equation 3.17, with the addition of the simplest controller and actuator models, now repeated as :

$$\underset{\sim}{H}(\underset{\sim}{q})\ddot{\underset{\sim}{q}} + \underset{\sim}{V}(\underset{\sim}{q},\dot{\underset{\sim}{q}})\dot{\underset{\sim}{q}} + \underset{\sim}{G}(\underset{\sim}{q}) = \underset{\sim}{K}[\bar{\underset{\sim}{r}} - \underset{\sim}{q}] \qquad (5.1)$$

where $\underset{\sim}{q}(t)$ is the joint response and $\bar{\underset{\sim}{r}}$ is the 'fixed' or reference command. Under normal circumstances the command is the desired response. Defining some 'tuned' command simply as $\underset{\sim}{r}$, then the perfect response will be attained if a solution to $\underset{\sim}{r}$ is obtained as :

$$\underset{\sim}{r} = \bar{\underset{\sim}{r}} + \underset{\sim}{K}^{-1}\ [\underset{\sim}{H}(\bar{\underset{\sim}{r}})\ddot{\bar{\underset{\sim}{r}}} + \underset{\sim}{V}(\bar{\underset{\sim}{r}},\dot{\bar{\underset{\sim}{r}}})\dot{\bar{\underset{\sim}{r}}} + \underset{\sim}{G}(\bar{\underset{\sim}{r}})] \qquad (5.2)$$

This is directly analogous to Paul's computed torque method; [3.5], except that the required displacement trajectory is computed, not the torque. The outline of such a scheme is shown in figure 5.1. If the model form and its parameters are known exactly, then the response would be perfect. In practice the form and values of all these quantities are not accurately known. With even small errors the discrepancies rapidly accumulate in the time history of the response, preventing precision tracking. The manpower required to obtain such parameters and model forms accurately is large.

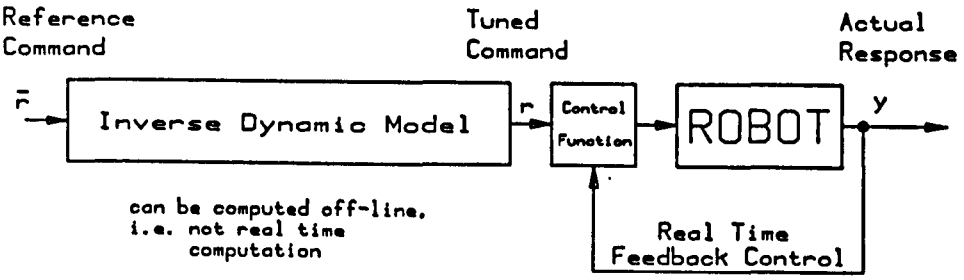


Figure 5.1 Analytic or Model Based Motion Tuning

Note that the nominal trajectory velocity and acceleration is also required in the solution for the tuned trajectory. The nominal trajectory must therefore be differentiable, and inverse kinematic solutions must be available (if appropriate).

Justifiable simplifications would aid model based tuning. Some works for example suggest that if velocities are low, velocity product terms are negligible. This is inaccurate as it is the relative magnitudes of velocity and acceleration that matter, not the absolute values. For general motions these ratios vary infinitely, so it is not possible to neglect the velocity products. Hollerbach 1984 [5.1] shows that the velocity product terms remain just as significant as the acceleration dynamics for all speeds of movement.

5.1.1 Simplification of The Dynamic Equations

Specific manipulator designs are possible which possess simplified inertia properties. Invariant inertia matrices are obtained e.g. Asada 1986 [5.2] by using specific configurations which satisfy a table of conditions. Such decoupled robot inertia matrices are usually only possible in manipulators possessing a maximum of two links. Asada outlines two basic orthogonal axis configurations which satisfy the conditions for decoupled inertias.

Gravitational terms can also be modified by careful design. The R-R-R-P 'SCARA' class only has a gravitational term in the final axis. The three revolute axes are all perpendicular to the horizontal plane

of operation, the final prismatic axis operating vertically. The inertia and coupling terms are also simple because of the configuration. These features probably explain the high performances attained by some manufacturers e.g. Adept $9ms^{-1}$ controlled velocity and 5g acceleration. The disadvantage is the limitation placed on the range of tasks these robots may undertake by virtue of the dominantly planar operation.

There is little work verifying even some of the simpler dynamic models. The MIT direct drive arm has been carefully designed with very high performance, relatively easy to model drives and transducers; it is one of Asada's 3 axis configurations possessing simple dynamics; it has a powerful control computer giving high sample rates; there is no load carried and yet its dynamic model experimental verification still indicates significant discrepancies, An, Atkeson, Griffiths and Hollerbach 1986 [5.3]. Simulation verification difficulties are much more evident when considering real manipulators. Featherstone 1985 [5.4] compares carefully prepared model results with the corresponding responses from a PUMA 560. Substantial errors are present in many of the results. In another experimental model verification a strong emphasis on actuator sub-system modelling yielded relatively high accuracy in the planar hydraulic work of Baysec 1983 [5.5].

These works serve to illustrate the difficulties in both producing and 'honing' such basic dynamic models.

5.1.2 Actuator/Joint/Drive Train Modelling

Generating accurate actuator system models is time consuming. Actuators are therefore often represented as simple linear algebraic relations with torque or force proportional to input. The dynamic elements are neglected. For a given manipulator and drive bandwidth this should be reviewed. Including actuator dynamics implies simultaneous solution with the robot's dynamics, increasing complexity and the likelihood of numerical difficulties. Actuator systems of certain robots can be more dominant than the link dynamics and so cannot be neglected. Motor armature effective inertia when driving through a gear ratio in excess of say 100:1 becomes highly significant.

5.2 Dynamic Model Parameter Measurement

If a dynamic model was to be used in the tuning process, some consideration must be given to the measurement of the various coefficients in the model.

5.2.1 The Robot Dynamic Parameters

The minimum number and types of robot dynamic parameters for an n link manipulator may be estimated as follows :

link relative joint locations	2n	
link relative joint orientations	2n	
*link centre of gravity locations	3n	
link masses	1n	
*link inertia tensors (in general)	6n	by symmetry
drive transfer functions (simplest linear)	1n	
control transfer functions (simplest e.g.P+V)	2n	
total	17n	parameters

*(Most robots contain other components e.g. gears, cable tracks, sub-links, shafts etc. which move relative to the link itself causing these quantities to vary.

5.2.2 Displacement Measurements

Location and orientation datums as used within the kinematic model, are not normally accesible for easy measurement. In fact, most exist inside solid matter and so must be estimated. Protruding, precision datum cylinders were manufactured and fitted to the 'Little Giant' to assist displacement measurement. Determination of the link masses can be carried out using standard industrial scales. Some of the 'Little Giant' links weigh as much as 75kg and include various brackets, cables and hoses which need to be removed before disassembly. Data supplied by the manufacturer was found to be seriously lacking in both quantity and quality. It is clear that the mass properties are a consequence, not a part of, the design process.

5.2.3 Inertia Measurements

The moment of inertia of robot links may be estimated or measured. Several methods are compared in Atkeson An and Hollerbach 1985 [5.6]. If the components are pre-defined on a CAD system, certain packages can compute the inertia tensor. It is understood that both IBM's 'CADAM' and Ferranti-Cetecs' 'CAMX' products can do this. The method considers the component 'finite' element by element. Practical features prevent these estimates being exact. Alternatively, periodic time measurements may be made with trifilar, bifilar or compound pendulum methods. In order to derive the full inertia tensor, a relationship must be established between these measured period values and the required elements of the inertia tensor. The methods for measurements about a single axis are well established, but this is not so for the complete inertia tensor. It is necessary to select sets of axes about which measurements are to be taken. These must be independent, result in low errors due to matrix ill conditioning and need only simple analysis for ease of checking.

To find the complete inertia tensor, problems arise because of the disposition of the axes about which measurements need to be taken. As many manipulator components are long and slender, the two axes perpendicular to the components longitudinal axis will present little problem. The remaining axes though will probably require brackets fitting to the link, (of simple geometry).

5.2.4 Inertia Axis Transformations

Assuming the appropriate method (bifilar or compound) has been selected, the set of axes about which measurements are taken needs to be selected. Referring to figure 5.2, this choice is best made by considering the expression relating to an inertia value about a single axis, to the constituents in the required axis frame :

$$J_{ii} = u_{ix}^2 J_{oxx} + u_{iy}^2 J_{oyy} + u_{iz}^2 J_{ozz} - 2.u_{ix}u_{iy}J_{oxy} - 2.u_{iy}u_{iz}J_{oyz} - 2.u_{iz}u_{ix}J_{ozx} \quad (5.3)$$

From this equation, given six arbitrary axes, the following matrix equation can be written :

$$\begin{bmatrix} J_{11} \\ J_{22} \\ J_{33} \\ J_{44} \\ J_{55} \\ J_{66} \end{bmatrix} = \begin{bmatrix} u_{1x}^2 & u_{1y}^2 & u_{1z}^2 & -2u_{1x}u_{1y} & -2u_{1y}u_{1z} & -2u_{1z}u_{1x} \\ u_{2x}^2 & u_{2y}^2 & u_{2z}^2 & -2u_{2x}u_{2y} & -2u_{2y}u_{2z} & -2u_{2z}u_{2x} \\ u_{3x}^2 & u_{3y}^2 & u_{3z}^2 & -2u_{3x}u_{3y} & -2u_{3y}u_{3z} & -2u_{3z}u_{3x} \\ u_{4x}^2 & u_{4y}^2 & u_{4z}^2 & -2u_{4x}u_{4y} & -2u_{4y}u_{4z} & -2u_{4z}u_{4x} \\ u_{5x}^2 & u_{5y}^2 & u_{5z}^2 & -2u_{5x}u_{5y} & -2u_{5y}u_{5z} & -2u_{5z}u_{5x} \\ u_{6x}^2 & u_{6y}^2 & u_{6z}^2 & -2u_{6x}u_{6y} & -2u_{6y}u_{6z} & -2u_{6z}u_{6x} \end{bmatrix} \begin{bmatrix} J_{0xx} \\ J_{0yy} \\ J_{0zz} \\ J_{0xy} \\ J_{0yz} \\ J_{0zx} \end{bmatrix} \tag{5.4}$$

A question at this stage is what conditions must be satisfied for the rows to be linearly independent? An algebraic analysis of the determinant would be tedious. It is also unlikely to yield simple conditions for singularity. An intuitive selection of six axes is

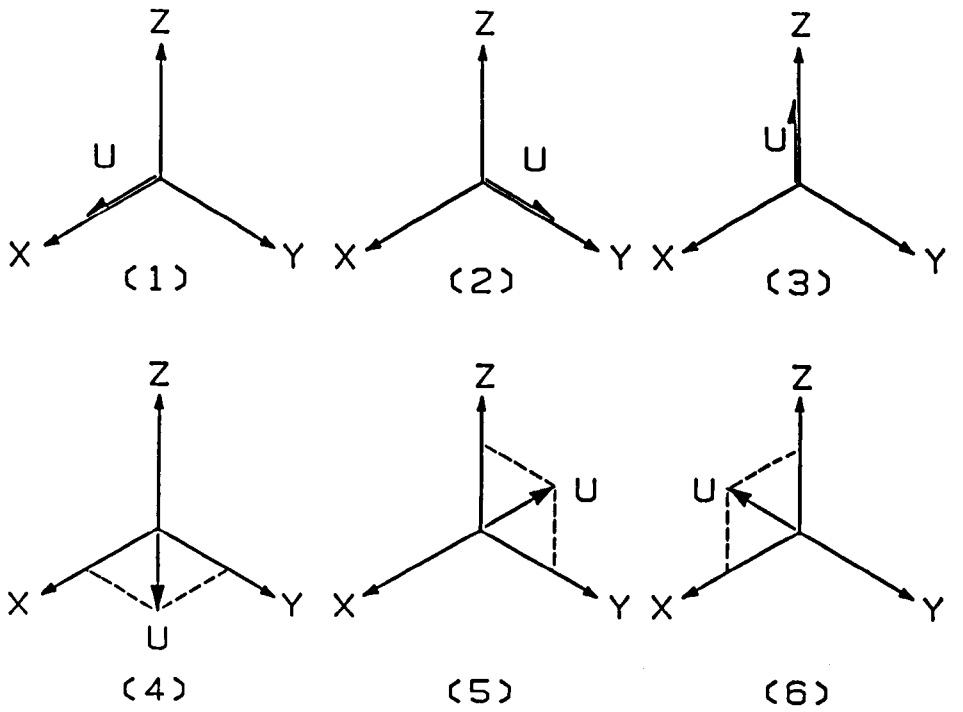


Figure 5.2 A Set of Six Axes for Inertia Tensor Evaluation

*(In each case u is a unit vector defining the axis about which a single scalar inertia value is to be measured)

found to yield results. From knowledge of triangularly decomposed matrices and their determinants, the main diagonal elements must all be non-zero for the matrix to be non-singular. The order of axes has therefore been selected appropriately. If the above matrix equation is represented as :

$$\underset{\sim}{J_{nn}} = \underset{\sim}{U}\ \underset{\sim}{J_{0\alpha\beta}} \tag{5.5}$$

where $\underset{\sim}{J}_{nn}$ - inertias in link system frame

$\underset{\sim}{U}$ - matrix relating inertias measured about one set of axes to another set

$\underset{\sim}{J}_{o\alpha\beta}$ - inertias in measurement system frame

then given the sets of axes, figure 5.2, the frames being represented as unit vectors are :

$$\underset{\sim}{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & -1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & -1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & -1 \end{bmatrix} \qquad (5.6)$$

and it can be shown that the determinant :

$$|\underset{\sim}{U}| = -1 \qquad (5.7)$$

and

$$\underset{\sim}{U}^{-1} = \underset{\sim}{U} \qquad (5.8)$$

The physical significance of this last equation is not entirely clear. The inherent symmetry of the axis directions chosen and the inertia tensor presumably influence these properties which it is easy to show, are not general. For example, if the 6th axis is selected to have a negative X component $-\sqrt{2}$, this results in :

$$\underset{\sim}{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & -1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & -1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 1 \end{bmatrix} \qquad (5.9)$$

hence

$$|\underset{\sim}{U}| = 1 \quad \text{but} \quad \underset{\sim}{U}^{-1} \neq \underset{\sim}{U} \quad \text{as} \qquad (5.10)$$

$$\underset{\sim}{U}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & -1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & -1 & 0 \\ -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 & 1 \end{bmatrix} \qquad (5.11)$$

5.2.5 Frequency Analysis Models of Drives and Controllers

There are numerous other parameters required to quantify the actuator sub-system, probably combined with the control components. In the past measurements were obtained by monitoring the response to various standard, time coordinated inputs. This process is still performed, but the application of digital technology has enhanced the methods. Digital signal analysers simultaneously sample both the input and

response of a servo system. The input to the servo is usually an harmonic, or random or pseudo random function. From this process, logged data can be derived quantities such as gain and phase, plotted against frequency. These curves may be termed transfer functions, but to actually obtain the form and content details requires curve fitting. As this feature was not available on the machine used, a computer aided curve fit package was developed, based on a Tektronix 4051 microcomputer.

System non-linearities limit the validity of the method. The open loop transfer function appears the simplest to obtain. In practice, several difficulties arise. The actuator drifts, soon reaching either the mechanical or transducer endstops. The analysis validity is often doubtful away from a setpoint. In the open loop, the dynamic range of measurements is probably wider, accentuating unwanted noise. Operating with a closed loop, some of these problems are resolved.

5.2.6 Actuator Measurements and Results

Using the aforementioned scheme, dynamic models of two robot actuator subsystems were obtained. The first, a high performance dc servo amplifier, motor and transducer, figures 5.3 to 5.5. The second was a servo current amplifier, hydraulic servo-valve, cylinder and transducer. The equipment layout and results are shown in figures 5.6 to 5.8. It should be noted that these results can be obtained very rapidly, and fairly accurately to around ±½ dB. The validity range depends on many parameters, but if measurements are closed loop, they should be reasonable.

The Hewlett Packard digital signal analyser used, applies the discrete Fourier transform. As a result the spacing of data points is linear in frequency interval. The Bode plot is logarithmic in form, so that the results have widely spaced data points at the low frequency end, with tightly packed data at the high frequencies. The first one or two points on the transfer function curves as measured are often spurious, this being a function of the algorithms within the machine. Non-linearities and signal noise tend to be accentuated in the higher frequencies, seen in figures 5.5(a) and 5.8(a). The simulation model of the pre-amplifier, power amplifier, DC servo-motor its load and

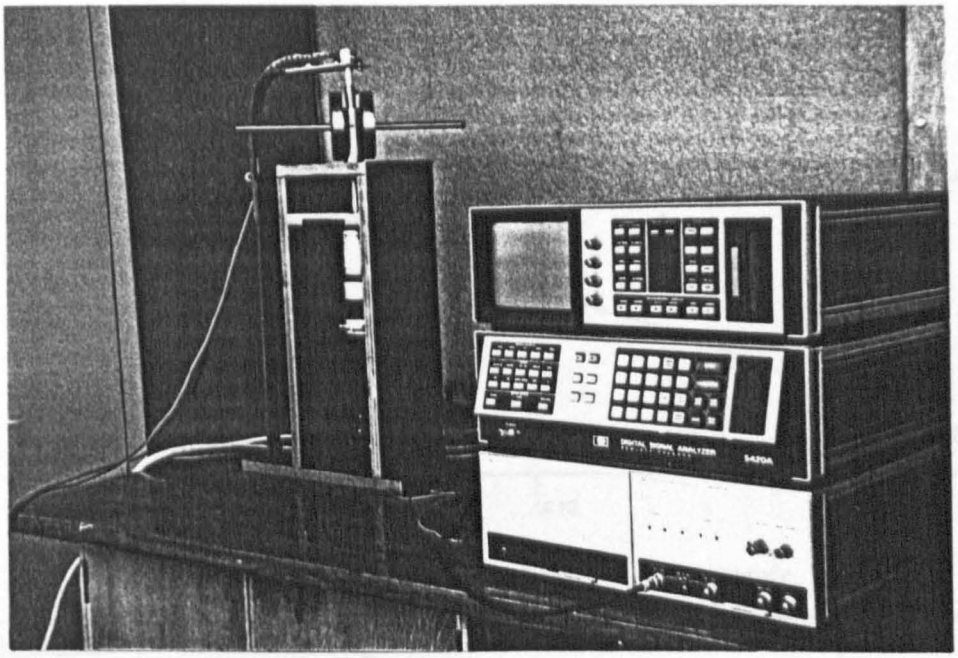


Figure 5.3 DC Servo Motor and Load, with HP5420 Signal Analyser in the Foreground

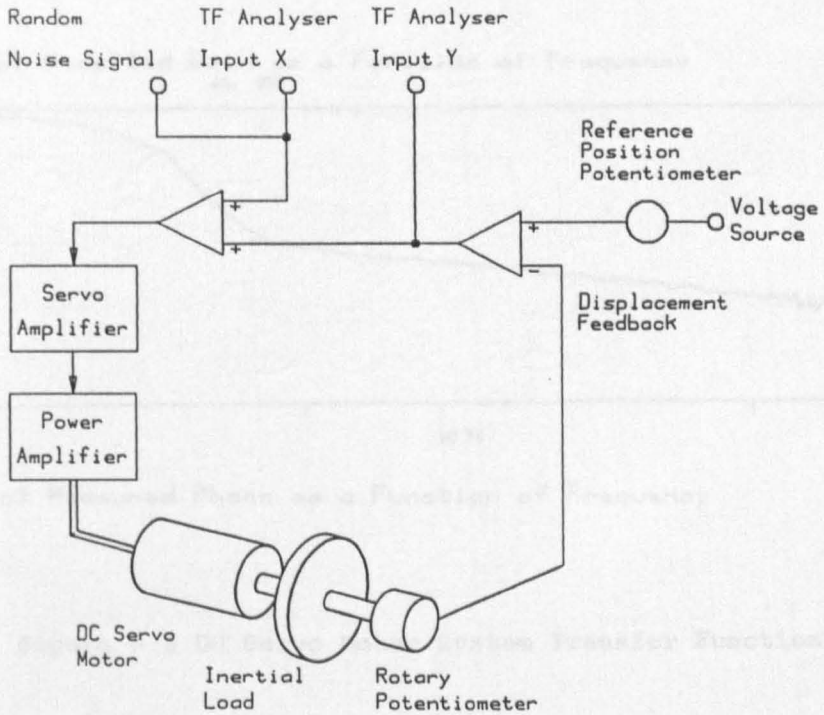
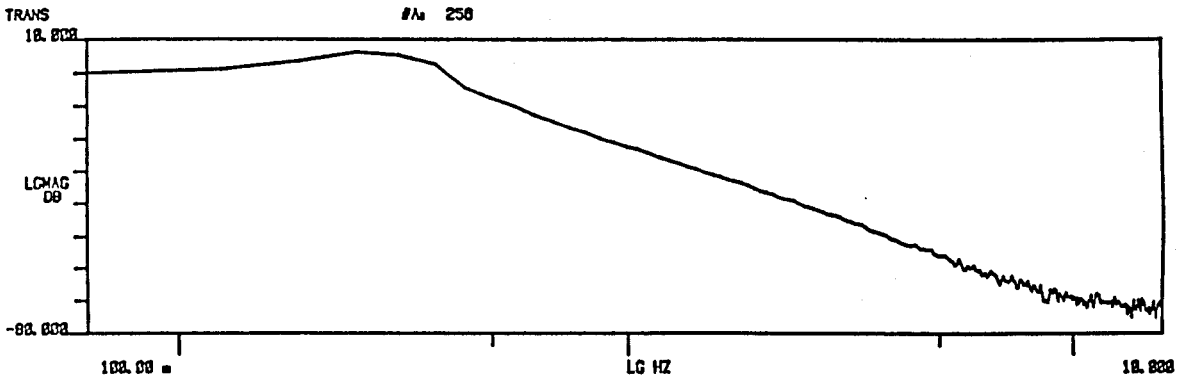
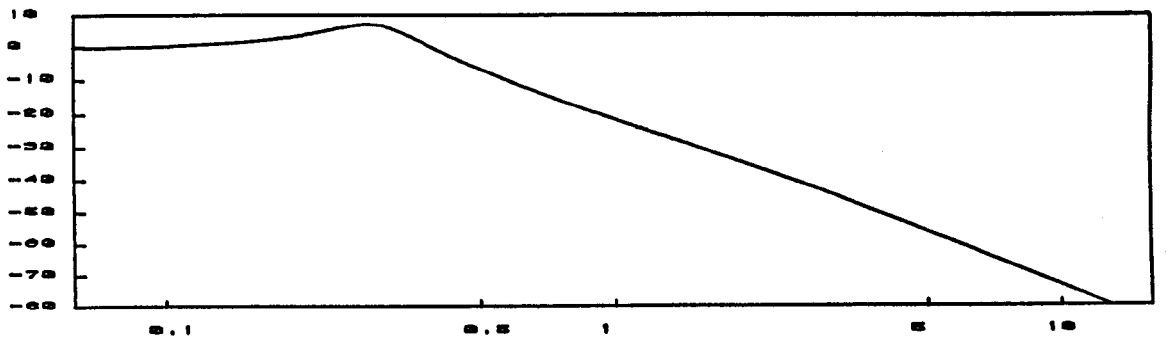


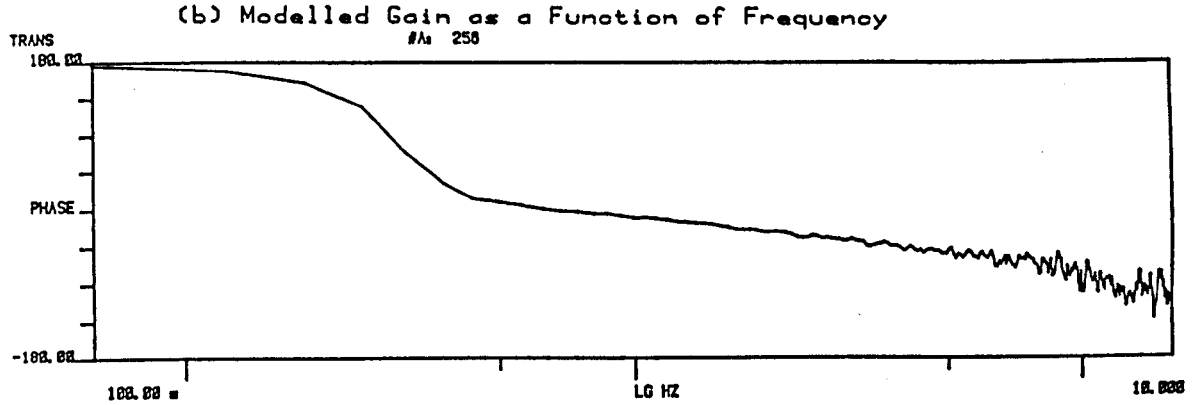
Figure 5.4 Schematic of DC Servo Motor Test Circuit



(a) Measured Gain as a Function of Frequency



(b) Modelled Gain as a Function of Frequency



(c) Measured Phase as a Function of Frequency

Figure 5.5 DC Servo Motor System Transfer Function

transducers was found to be :

$$y(s)/u(s) = \frac{K}{(1+\tau s)(s^2+2\zeta w_n s+w_n^2)} \qquad (5.12)$$

where $K = 0.9224$
$\tau = 0.05689$
$\zeta = 0.2010$
$w_n = 0.2936$

Peak gain error squared at the given data points = 0.3674
Sum of squared gain errors at the six data points = 1.216
The number of iterations to achieve this result = 30

The equivalent simulation model in the case of the pre-amplifier, hydraulic servo-valve, cylinder, load and transducer was found to be :

$$y(s)/u(s) = \frac{(s^2+2\zeta_3 w_{n3} s+w_{n3}^2)}{(1+\tau_1 s)(1+\tau_2)(s^2+2\zeta_1 w_{n1} s+w_{n1}^2)(s^2+2\zeta_2 w_{n2} s+w_{n2}^2)} \qquad (5.13)$$

where $\tau_1 = 0.2738$ $\zeta_1 = 0.05121$ $w_{n1} = 5.266$
$\tau_2 = 0.005076$ $\zeta_2 = 34.56$ $w_{n2} = 0.002280$
$\zeta_3 = 5.356$ $w_{n3} = 0.04811$

Peak gain error squared at the given data points = 1.683
Sum of squared gain errors at the six data points = 4.881
The number of iterations to achieve this result = 101

The solution method used was a univariate searching algorithm, using least squared errors as the performance criterion. The program which gives the transfer function and parameter values is listed in appendix B. It can cope with any number of each of the types of pole and zero, and returns the values in a ready factorised form.

The analyser was also found extremely useful in the empirical optimisation of gains. For example, progressively adjusting system gains on the DC servo-motor, with the analyser in a real time mode enabled a maximum closed loop bandwidth to be obtained as 33.78 Hz (figure 5.9(a)). For greater clarity, the vertical scale was expanded to precisely align to the 3 dB points, figure 5.9(b). It is possible to estimate the optimum harmonic bandwidth of such a motor based on :

$$w_2 - w_1 \approx 1.2539 \sqrt{\{N_{max}/(A.J)\}} \qquad (5.14)$$

where $w_2 - w_1$ - harmonic bandwidth (rads.s^{-1})
N_{max} - maximum torque of motor (N.m)
A - rotor harmonic amplitude (rads)
J - total rotor inertia (kg.m^2)

The assumptions made in the derivation of equation 5.14 are an inertial load, viscous friction, proportional error control, negative velocity feedback, damping ratio $1/\sqrt{2}$. The motor/load tests had the following approximate characteristics :

$N_{max} = 2.52$ N.m
A = 0.5 rads
J = 2 x10^{-4} kg.m^2

which indicate a nominal optimum bandwidth of around 32 Hz; suggesting that as a rough estimate, the method may be more widely useful.

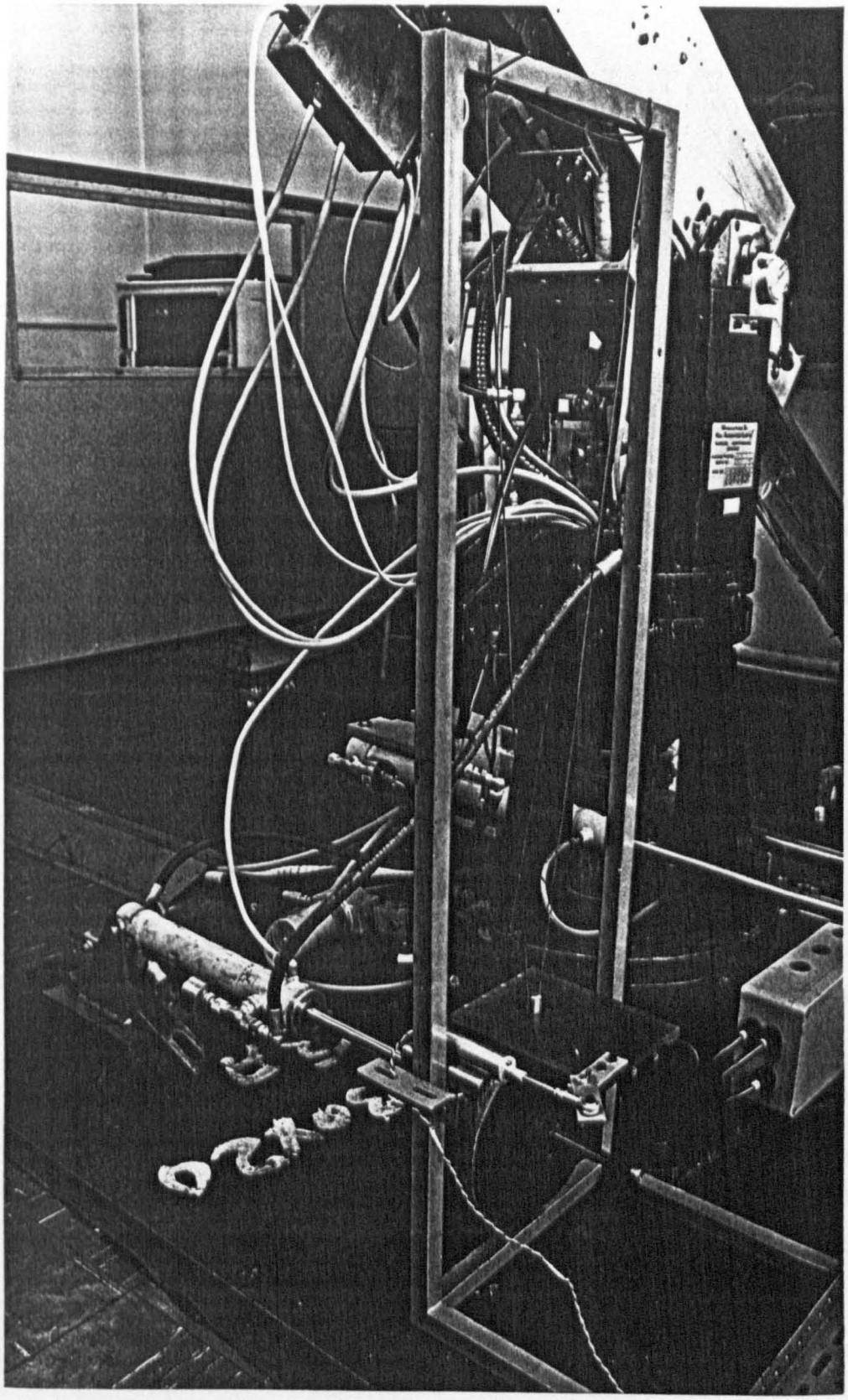


Figure 5.6 Hydraulic Cylinder and Load Test

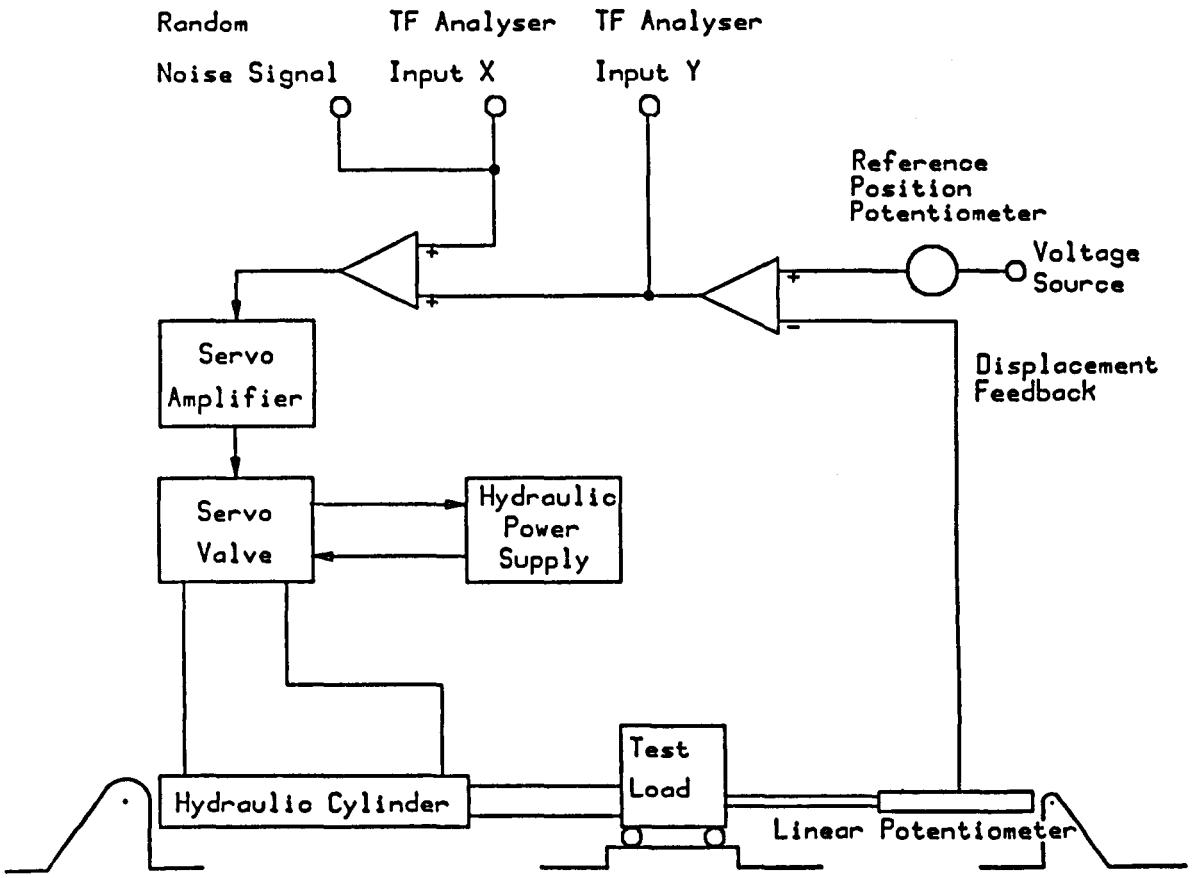
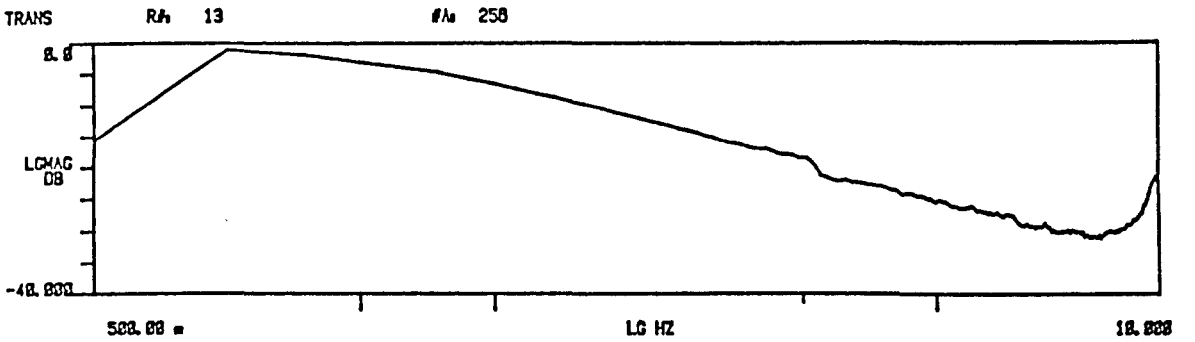
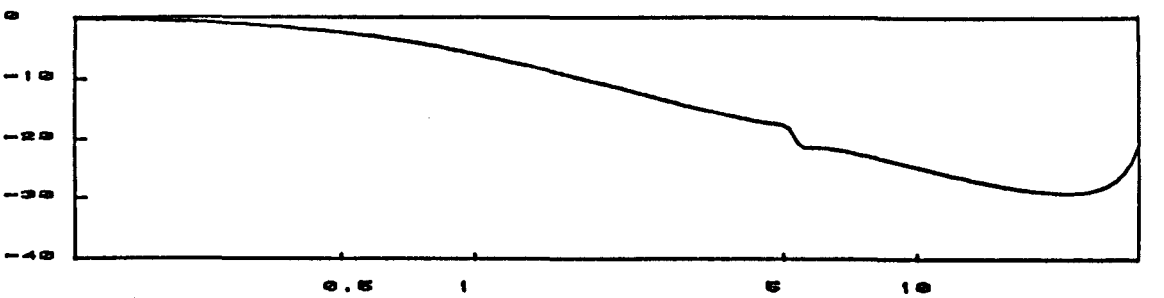


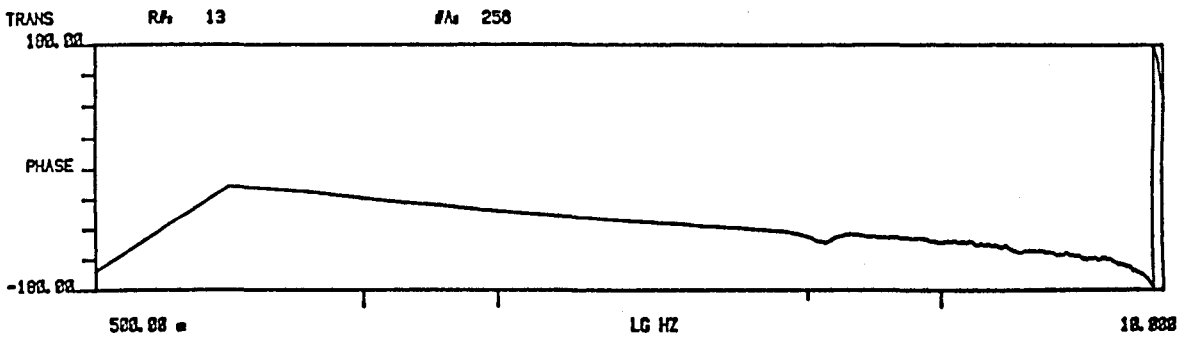
Figure 5.7 Schematic of the Hydraulic Cylinder and Load Test Rig



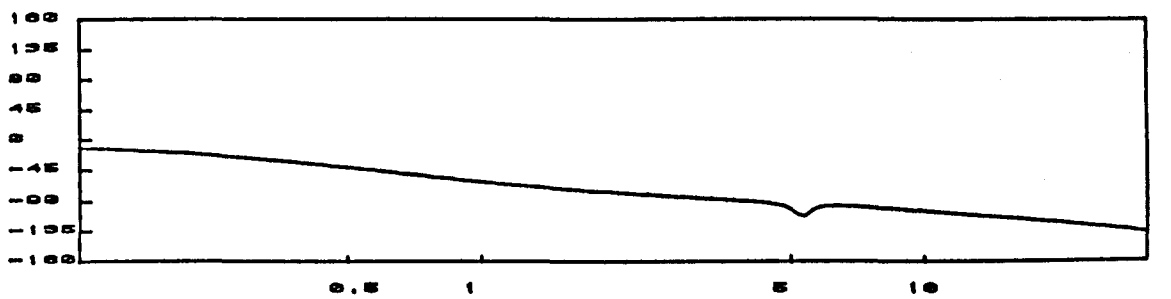
(a) Measured Gain as a Function of Frequency



(b) Modelled Gain as a Function of Frequency

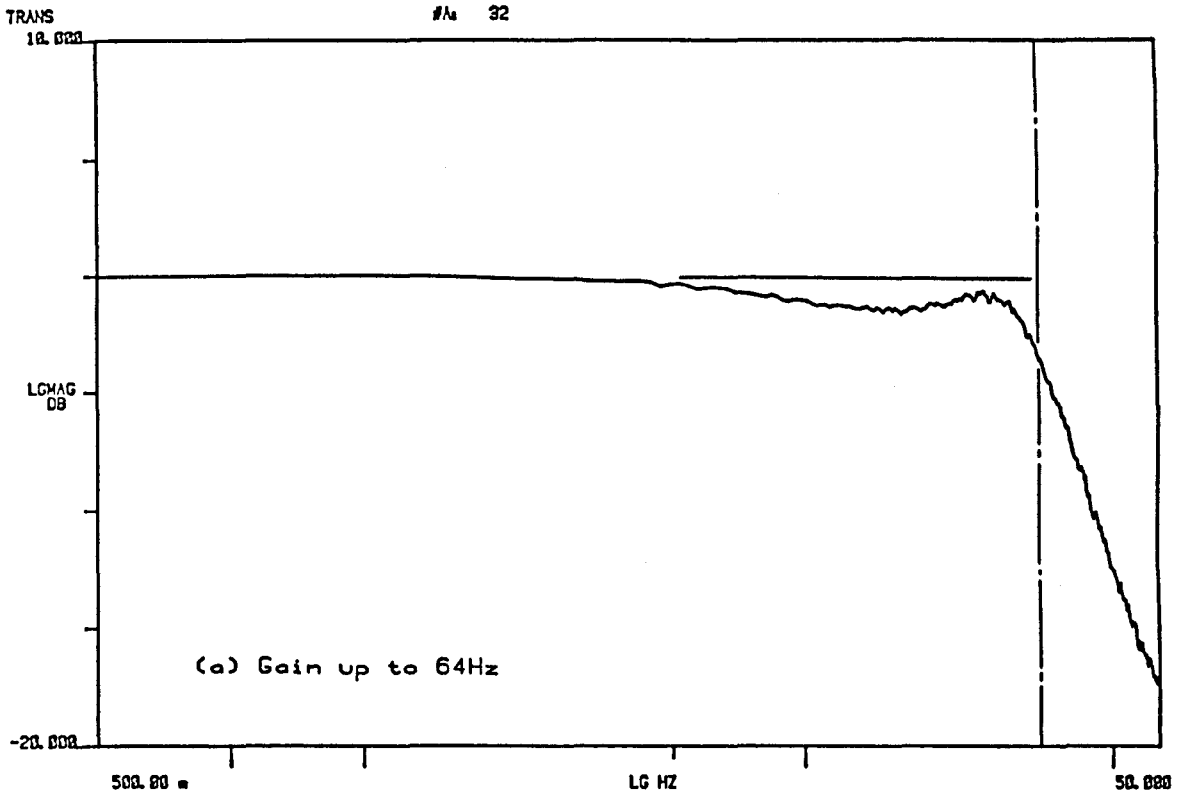


(c) Measured Phase as a Function of Frequency



(d) Modelled Phase as a Function of Frequency

Figure 5.8 Hydraulic Servo System Transfer Functions



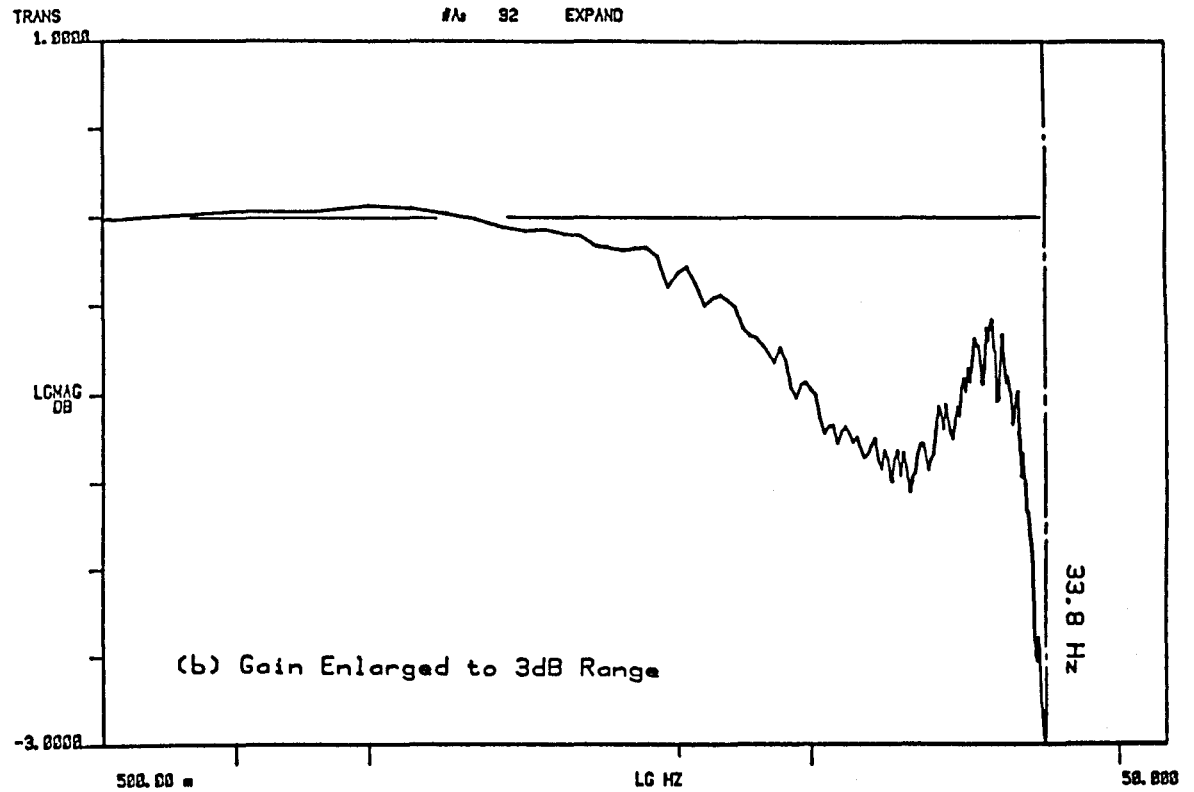


Figure 5.9 Optimised DC Servo Motor Displacement Loop Bandwidth

The measurement and modelling of the simpler components within a detailed dynamic model, is clearly a significant amount of work. It is therefore considered that detailed modelling of a robot not feasible within the timescales of this project. Further, it appears likely that a trade-off exists; as the complexity of the model increases, the apparent accuracy increases but the actual numerical accuracy decreases with the accumulation of parameter measurement errors.

Attempting to tune motions based on these more complex models is only practical, if high parameter measurement accuracy is achieved. There may be other alternatives; such as to couple a motion tuning system to an on-line system identification algorithm. There are other problems, some of these effects have widely varying periods or associated displacement components in a response. By setting up the equations appropriately, the required command could be computed by iteratively integrating and varying the command at each step, until the required 'response' was achieved. Numerical integration routines will tend to drift on the gross response because the computation points must be selected to pick up the higher frequency elements or locate the switching point for a discontinuity. These switched solutions, required for example in Coulomb friction, are necessarily iterative.

In summary, it is probably impossible to gain any benefit from the more complex models, when tuning motions. It is also certain that the tuning process could only be numerical; no tuned coefficient functions would be available, for computing the tuned motion. Simplification of the dynamic model therefore warrants some attention.

5.3 Methods Based on Reduced Dynamic Models

Simplified models can also be used for motion tuning. The dynamic equations 3.17 can be linearised for example. Tuning could then be carried out using these linearised equations, re-computing the dynamic coefficients within them at suitable intervals. Alternatively, certain forms of advanced feedback control permit otherwise complex and disturbed systems to respond as if their dynamics could be characterised by simple models. One such scheme, already mentioned in section 3.6.5 is a model referenced adaptive control scheme. Tuning motions for this scheme are described in chapter 7.

5.3.1 Linearised Equations of Motion

For small perturbations about an operating point, the equations may be linearised. Taking the complete set of partial derivatives for each equation and approximating $d\underset{\sim}{q}_j$ etc. to be of small but finite value, the equations can be written in a second order linear form. With some manipulation this can be expressed in state space form :

$$\dot{\underset{\sim}{x}}(d\dot{\underset{\sim}{q}}, d\underset{\sim}{q}) = \underset{\sim}{A}\ \underset{\sim}{x}(d\dot{\underset{\sim}{q}}, d\underset{\sim}{q}) + \underset{\sim}{B}\ \underset{\sim}{u}(d\underset{\sim}{\tau}) \qquad (5.15)$$

Note that these coordinates and torques are perturbation values and so must be added to the state point values to obtain absolute numerical values. This form also requires the algebraic inversion of the linearised inertia matrix. Expressed in the form :

$$\bar{\underset{\sim}{H}}\ d\ddot{\underset{\sim}{q}} + \bar{\underset{\sim}{V}}\ d\dot{\underset{\sim}{q}} + \bar{\underset{\sim}{G}}\ d\underset{\sim}{q} = d\underset{\sim}{\tau} \qquad (5.16)$$

the 2R-P $d_1\ d_2\ d_3\ a_3$ linearised equations are :

$$\begin{bmatrix} \bar{H}_{11} & \bar{H}_{12} & \bar{H}_{13} \\ \bar{H}_{21} & \bar{H}_{22} & \bar{H}_{23} \\ \bar{H}_{31} & \bar{H}_{32} & \bar{H}_{33} \end{bmatrix} \begin{bmatrix} d\ddot{\theta}_1 \\ d\ddot{\theta}_2 \\ d\ddot{d}_3 \end{bmatrix} + \begin{bmatrix} \bar{V}_{11} & \bar{V}_{12} & \bar{V}_{13} \\ \bar{V}_{21} & \bar{V}_{22} & \bar{V}_{23} \\ \bar{V}_{31} & \bar{V}_{32} & \bar{V}_{33} \end{bmatrix} \begin{bmatrix} d\dot{\theta}_1 \\ d\dot{\theta}_2 \\ d\dot{d}_3 \end{bmatrix} + \begin{bmatrix} 0 & \bar{G}_{12} & \bar{G}_{13} \\ 0 & \bar{G}_{22} & \bar{G}_{23} \\ 0 & \bar{G}_{32} & \bar{G}_{33} \end{bmatrix} \begin{bmatrix} d\theta_1 \\ d\theta_2 \\ dd_3 \end{bmatrix} = \begin{bmatrix} d\tau_1 \\ d\tau_2 \\ d\tau_3 \end{bmatrix} \qquad (5.17)$$

where the $\bar{\underset{\sim}{H}}$ matrix is identical to the $\underset{\sim}{H}$ matrix of equation 3.17 (and so remains symmetric). The $\underset{\sim}{V}$ and $\underset{\sim}{G}$ matrices though are quite different. The state point is made up of a set of constant numerical values θ_1, θ_2 and d_3 and their corresponding velocities and accelerations. The equations can only be valid therefore for small values around the state point. Evaluating the terms from this equation, then re-arranging to state space form, results in the following components :

$$\underset{\sim}{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ A_{61} & A_{62} & A_{63} & A_{64} & A_{65} & A_{66} \end{bmatrix} \qquad (5.18)$$

$$x = \begin{bmatrix} d\theta_1 \\ \dot{d\theta_1} \\ d\theta_2 \\ \dot{d\theta_2} \\ dd_3 \\ \dot{dd_3} \end{bmatrix} \quad (5.19) \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ B_{21} & B_{22} & B_{23} \\ 0 & 0 & 0 \\ B_{41} & B_{42} & B_{43} \\ 0 & 0 & 0 \\ B_{61} & B_{62} & B_{63} \end{bmatrix} \quad (5.20) \qquad u = \begin{bmatrix} d\tau_1 \\ d\tau_2 \\ d\tau_3 \end{bmatrix} \quad (5.21)$$

These elements are complex, for example :

$$A_{42} = \Big[(H_{22}H_{33} - H_{23}^{*}) \left[\frac{\partial H_{11}}{\partial \theta_2}\ddot{\theta}_1 + \frac{\partial H_{12}}{\partial \theta_2}\ddot{\theta}_2 + \frac{\partial H_{13}}{\partial \theta_2}\ddot{d}_3 + \frac{\partial V_{12}}{\partial \theta_2}\dot{\theta}_2 + \frac{\partial V_{13}}{\partial \theta_2}\dot{d}_3\right]$$
$$+ (H_{13}H_{23} - H_{12}H_{33}) \left[\frac{\partial H_{21}}{\partial \theta_2}\ddot{\theta}_1 + \frac{\partial V_{21}}{\partial \theta_2}\dot{\theta}_1 + \frac{\partial G_2}{\partial \theta_2}\right]$$
$$+ (H_{12}H_{23} - H_{13}H_{22}) \left[\frac{\partial H_{31}}{\partial \theta_2}\ddot{\theta}_1 + \frac{\partial_{31}}{\partial \theta_2}\dot{\theta}_1 + \frac{\partial G_3}{\partial \theta_2}\right] \Big] \div$$
$$[H_{11}(H_{22}H_{33} - H_{23}^{*}) + H_{12}(H_{13}H_{23} - H_{12}H_{33}) + H_{13}(H_{12}H_{23} - H_{13}H_{22})] \quad (5.22)$$

and can be evaluated at the set point, for constant values of :

$$\theta_1, \dot{\theta}_1, \ddot{\theta}_1 \quad \theta_2, \dot{\theta}_2, \ddot{\theta}_2 \quad d_3, \dot{d}_3, \ddot{d}_3.$$

The embedded sub-group terms are extremely complex so it is probably easier to estimate the A_{ij} values in some other way.

5.3.2 Linearised Equation Solutions in Tuning Motion Laws

The solutions of ordinary, linear differential equations representing continuous systems can be used in the tuning process. Using the discrete solutions has definite advantages though, as the response is not dependent on the complete input function, just its instantaneous value. The discrete solution may be written as :

$$\underset{\sim}{x}_{i+1} = \underset{\sim}{P}\,\underset{\sim}{x}_i + \underset{\sim}{Q}\,\underset{\sim}{r}_i \quad (5.23)$$

$$\underset{\sim}{y}_i = \underset{\sim}{C}\,\underset{\sim}{x}_i \quad (5.24)$$

The direct solution method, using exponentiation of a matrix gives :

$$\underset{\sim}{P} = \epsilon^{A\delta t} \quad (5.25) \qquad \underset{\sim}{Q} = \int_0^t \epsilon^{A\tau} B.d\tau \quad (5.26)$$

where δt - the sample period or interval

$\underset{\sim}{P}$ - discretised response solution; system matrix

$\underset{\sim}{Q}$ - discretised response solution; input matrix

The simpler solution is offered by the Laplace method :

$$\underset{\sim}{P} = L^{-1}\{(s\underset{\sim}{I} - \underset{\sim}{A})^{-1}\}_{t=\delta t} \quad (5.27)$$

$$\underset{\sim}{Q} = L^{-1}\left\{\frac{1}{s}(s\underset{\sim}{I} - \underset{\sim}{A})^{-1}\underset{\sim}{B}\right\}_{t=\delta t} \quad (5.28)$$

In either case, the matrices for representative problems are at least of order 6, hence the inversion problems are severe for other than fixed numerical cases. The characteristic equation :

$$| s\underset{\sim}{I} - \underset{\sim}{A} | = \underset{\sim}{O}$$

in our case is found to be :

$$\begin{aligned} & s^6 - [A_{22} + A_{44} + A_{66}]s^5 \\ & + [(A_{44} + A_{66})A_{22} - A_{42}A_{24} - -A_{62}A_{26} + A_{66}A_{44} - A_{64}A_{46} - A_{65}]s^4 \\ & + [(-A_{66}A_{44} + A_{64}A_{46} + A_{65})A_{22} + (A_{66}A_{42} - A_{62}A_{46})A_{24} \\ & \quad -A_{62}A_{25} + (-A_{64}A_{42} + A_{62}A_{44})A_{26} + A_{65}A_{44} - A_{64}A_{45}]s^3 \\ & + [(-A_{65}A_{44} - A_{64}A_{45})A_{22} + (A_{65}A_{42} - A_{62}A_{45})A_{24} \\ & \quad + (-A_{64}A_{42} + A_{62}A_{44})A_{25}]s^2 \\ & = 0 \end{aligned} \qquad (5.30)$$

Substitution of the A_{ij} elements in full appears unlikely to yield any generic conclusions. The above expression only caters for one specific three axis robot. The prospect of algebraically inverting $(s\underset{\sim}{I} - \underset{\sim}{A})$ is not attractive. It is probably easier to use the equations in their original form. Some other approach should be considered.

Alternatively, we can use a model which is crude but correct in a gross sense, i.e. it possesses inertia, viscous friction and stiffness (gravity). This basic model may be capable of tuning the motion and improving the quality of response.

Referring back to section 3.6.5, Landau's MRACS scheme is in principle capable of making the complex robot dynamics appear as if they were simply decoupled second order linear systems. If the axis responses could be constrained to behave in such a manner, then tuning the motions becomes much simpler, and the above method becomes feasible.

The selection of the model coefficients (3 per axis) can be made by characterising the 'slowest' response of the manipulator in terms of second order time constants. This will prevent optimum performance being attained in any other configuration, but automatically builds in some actuator saturation limits. A more advanced scheme might use sets of simple models for differing configuration zones. These simple models could be obtained experimentally by using a digital signal analyser, but the interaction between axes would be difficult to quantify.

With linear models implemented digitally, and a fixed sample interval, the response can be computed without knowledge of the form of the input. This has advantages in model based tuning in that the tuning algorithm becomes simpler still. Two schemes are implemented in chapter7, which cater for :

(i) Joint based motions.

(ii) World (Cartesian) based motions.

Using the simplified dynamic models, it becomes practical to derive explicit relations for the tuned motion law coefficients; as follows :

5.3.3 Tuned Joint Polynomial Laws

Tuned polynomial motion laws are obtained by substituting the desired motion law and its appropriate derivatives into a linearised dynamic system model, in place of the response. This appears to neglect the complementary function component of the response. This is found to be reasonable, given that its elimination can be achieved by ensuring the initial conditions of the system match those of the command motion.

The coefficients of the tuned polynomial motion law which is required to be input can then be computed. Patterns are noted in the solution and can be exploited in the derivation of algebraic algorithms. The polynomial input may be generated as a force or a displacement function of time. The tuned motion driven law is easily derived from the force driven law, and so the latter is computed initially. In the case of cam driven systems, the motion generated can usually be considered as single-input, single-output, this case being considered first. The driven system in the robot case is multi-degree of freedom. It is found to be practically impossible to tune motion laws in this case based on 'true' dynamic models. The solution is to use a controller, capable of effectively decoupling the degrees of freedom, as already described.

5.3.4 Single Degree of Freedom - Force Driven

A linear system, characterised by the nth order differential equation

$$\sum_{j=0}^{n} b_j \frac{d^j}{dt^j} q(t) = F(t) \qquad (5.31)$$

is required to exhibit a polynomial response, of the mth degree :

$$q(t) = \sum_{i=0}^{m} c_i \; t^i \qquad (5.32)$$

It can be shown by induction, that the jth derivative of q(t) is given by :

$$\frac{d^j}{dt^j} q(t) = \sum_{i=j}^{m} \{ \prod_{k=i-j+1}^{i} k \} \; c_i \; t^{i-j} \qquad (5.33)$$

and of course :

$$\frac{d^j}{dt^j} q(t) = q(t) \quad \text{for } j = 0 \qquad (5.34)$$

hence

$$F(t) = b_0 \; q(t) + \sum_{j=1}^{n} b_j \; [\sum_{i=j}^{m} \{ \prod_{k=i-j+1}^{i} k \} \; c_i \; t^{i-j}] \qquad (5.35)$$

which is another polynomial of the form :

$$F(t) = \sum_{i=0}^{m} a_i \; t^i \qquad (5.36)$$

and with some manipulation, the a_i coefficients can be expressed as :

$$a_i = b_0 \; c_i + \sum_{j=1}^{m-i} b_j \; c_{i+j} \{ \prod_{k=i+1}^{i+j} k \} \qquad (5.37)$$

for i = 0,1,2,.....,m-1

and $a_m = b_0 \; c_m$ for i = m (5.38)

This last element a_m is not a function of n; as :

$$\frac{d^j}{dt^j} q(t) = 0 \qquad \text{for all } j > m \qquad (5.39)$$

also note $b_j = 0$ for $j > n$ (5.40)

5.3.5 Single Degree of Freedom - Motion Driven

If a linear system is driven by the time coordinated displacement function r(t), then the nth order differential equation whose coordinate q(t) is coupled to r(t) up to the term of order 1 is :

$$\sum_{j=0}^{n} b_j \frac{d^j}{dt^j} q(t) = \sum_{j=0}^{1} b_j \frac{d^j}{dt^j} r(t) \qquad (5.41)$$

The system is required to exhibit the polynomial response :

$$q(t) = \sum_{i=0}^{m} c_i \; t^i \qquad (5.42)$$

It is assumed that a solution can be found for r(t) as a polynomial :

$$r(t) = \sum_{i=0}^{m} d_i \; t^i \qquad (5.43)$$

hence the problem is to find the d_i coefficients. If the right hand side of equation 5.41 is assigned thus :

$$\sum_{j=0}^{1} b_j \frac{d^j}{dt^j} r(t) = F(t) \qquad (5.44)$$

This being exactly the same form as equation 5.31. If F(t) is also a polynomial :

$$F(t) = \sum_{i=0}^{m} a_i \; t^i \qquad (5.45)$$

Then it can be seen that :

$$\sum_{j=0}^{n} b_j \frac{d^j}{dt^j} q(t) = F(t) \qquad (5.46)$$

and so the a_i coefficients may be obtained from equations 5.37 and 5.38. Next, referring to equation 5.41, this is solved (neglecting the complementary function). The solutions, with the notation appropriately modified are :

$$d_m = a_m/b_o \qquad \text{for } i = m \qquad (5.47)$$

$$d_i = \frac{1}{b_o} \left[a_i - \sum_{j=1}^{m-i} b_j \, d_{i+j} \left\{ \prod_{k=i+1}^{i+j} k \right\} \right] \qquad (5.48)$$

$$\text{for } i = m-1, m-2, \ldots\ldots, 0$$

These evaluations must be carried out in the set order to avoid unnecessary algebraic complexity. The evaluation orders for a_i and d_i are not compatible for explicit solutions for the d_i terms. Explicit solutions can be found if required, reducing the above to a one step process.

It should be noted that in the case where damping is present, (coupled with low order polynomial laws) steps arise in the tuned displacement law. This is required in order to generate a static force which can overcome the viscous damping term arising out of a nominally instantaneous change in velocity.

5.4 Self Learning Based Methods

A computer controlled robot contains all the elements necessary for an autonomous self-experimentation system, whilst in normal use. This feature may be taken advantage of in the reduction of tracking errors and motion duration.

5.4.1 Command Tuning Using Past Response Data

Whatever form of control is employed, at the end of the day the inputs to a servo system are manipulated in response to measurements made by transducers on the system. These manipulations can be made using feedback control in real time, or after a run cycle and hence off-line, reducing computational load.

Other options for command tuning include the use of past response data

from previous samples or run cycles (assuming repetitive operations). Data from previous samples may be used in linear parameter estimation techniques. Data from previous completed run cycles rather than samples has rarely been used for control purposes and will be explored here. Because of the potential, this approach will be termed, (albeit primitive) 'self learning'. It is implied that some form of performance improvement is achieved by using previous cycle response data, which can be thought of as past experience. The element of the principle used here is that however complex the system's dynamics, they do not change substantially from one repeated operational cycle to another.

Static accuracies of industrial robots can be very high, typically 0.05% of the axis displacement range, but dynamically they may exceed 10%. The tracking accuracy of the robot can be improved by tuning the shape of the command profile.

In the robot and controller we have a self contained experimental and data logging system. It is capable of estimating its own dynamic parameters but substantial effort is required to use these in tuning the command for an arbitrary trajectory. The robot dynamics are inherently contained in the response, so it is appropriate to use this information in a self learning strategy, figure 5.10. The objective of the strategy is to progressively increase the trajectory tracking accuracy until some machine limitations are reached.

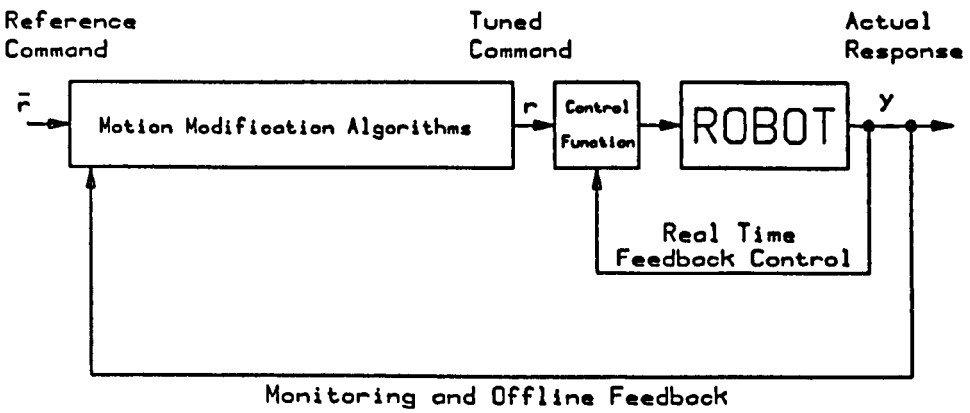


Figure 5.10 Structure of a Self Learning Motion Tuning Scheme

5.4.2 Relation to Previous work

Monitoring response information from a trajectory run has been widely studied in the estimation of inertial components. Most of this work has been simulation, exceptions being those of Atkeson, An and Hollerbach 1985 [5.6] and An, Atkeson, Griffiths and Hollerbach 1986 [5.3]. These highlighted the problems of using transducer measurements as opposed to high resolution and accuracy, variable values from a noise free numerical simulation. Craig 1984 [5.7] produces a simulation of a learning scheme which does utilise robot repeatability, but in constructing a torque function. It relies on substantial dynamic modelling and the use of a complex control law.

Arimoto 1984 [5.8] proposed a 'Betterment Process' in which arbitrarily weighted error derivatives from a previous run are added into the trajectory. The 'simplest structure' proposed here is overlooked, because convergence is not assured in general. In his implementation, Arimoto 1985 [5.9] uses an up-dating algorithm :

$$\underset{\sim}{r}_{k+1} = \underset{\sim}{r}_k + \underset{\sim}{L}\frac{d}{dt}(\dot{\underset{\sim}{r}} - \dot{\underset{\sim}{y}}_k) \qquad (5.49)$$

where $\underset{\sim}{L}$ - learning gain matrix (square)

The error is computed in terms of velocity, so the nominal command must be available expressed as a velocity. It is then numerically differentiated and added into the previous command. After 28 runs, the displacement errors are reduced to around 10% of those at the start of the process. The slow convergence is attributed to the low magnitudes of the $\underset{\sim}{L}$ matrix elements, which is also chosen to be diagonal. Because $\underset{\sim}{y}_k$ is measured, the above subtraction will amplify the signal or truncation noise. Numerically differencing this quantity will further exacerbate the problem.

Two self learning schemes are developed in this project; the trajectory tracking accuracy is cyclically improved in the first scheme, of chapter 6. The motion speed is progressively increased in the scheme of chapter 8. It remains to select a suitable motion law.

5.5 Motion Law Selection for Increased Trajectory Velocities

Motion law selection becomes more critical at higher speeds, as it can

influence the trajectory tracking accuracy. This happens because the law itself interacts with the system dynamics, in turn making the task of the controller more difficult.

Designing displacement trajectories for compliant spatial dynamic structures in order to excite minimal vibrations is an extremely difficult topic. It is, though, possible to carry out the inverse of this, demonstrating a relationship between the vibration amplitudes and the shape of the associated displacement laws and their derivatives.

5.5.1 Minimum Vibration Polynomial Motion Laws

A motion was used for comparative tests, comprising a diamond shaped path traced in a vertical plane, figure 5.11. Acceleration results taken from three displacement laws applied to an end effector motion are shown in figures 5.12 and 5.13. The motion boundary conditions are shown in table 5.14. The three planar actuators were required to carry this motion out: being the vertical, horizontal and fan axes.

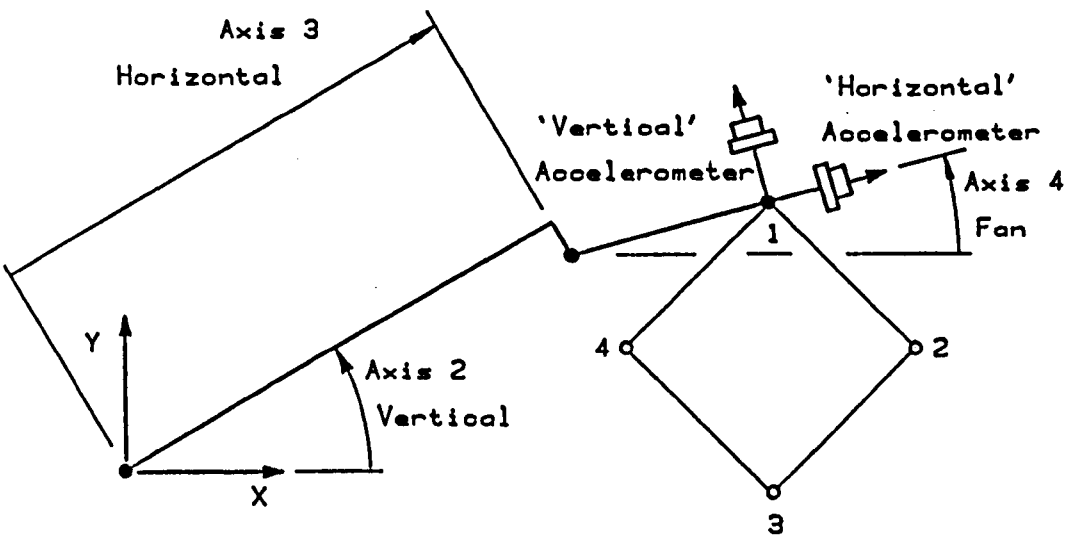


Figure 5.11 Planar Diamond Test Path Executed by Little Giant

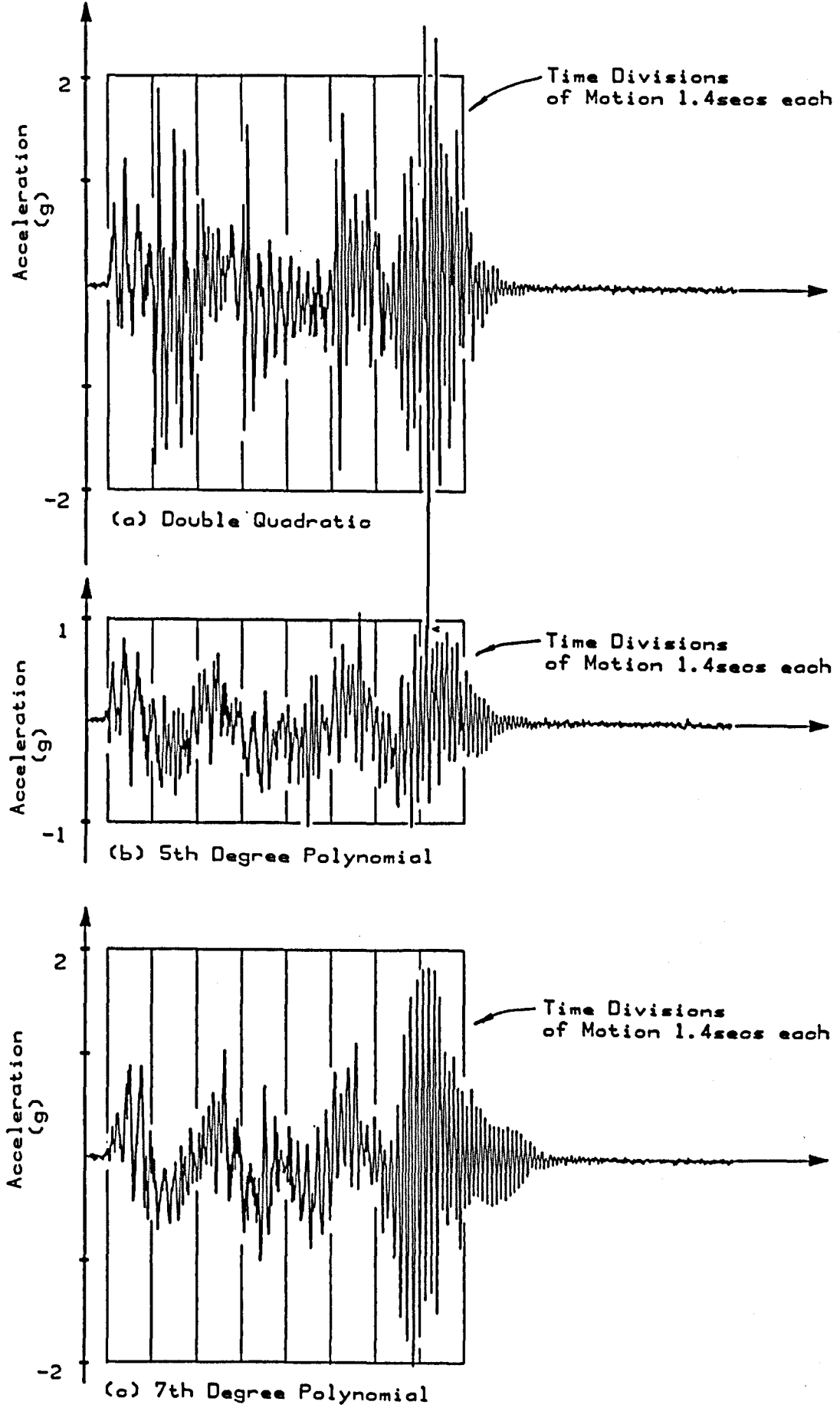


Figure 5.12 Vertical Acceleration Responses from Planar Diamond

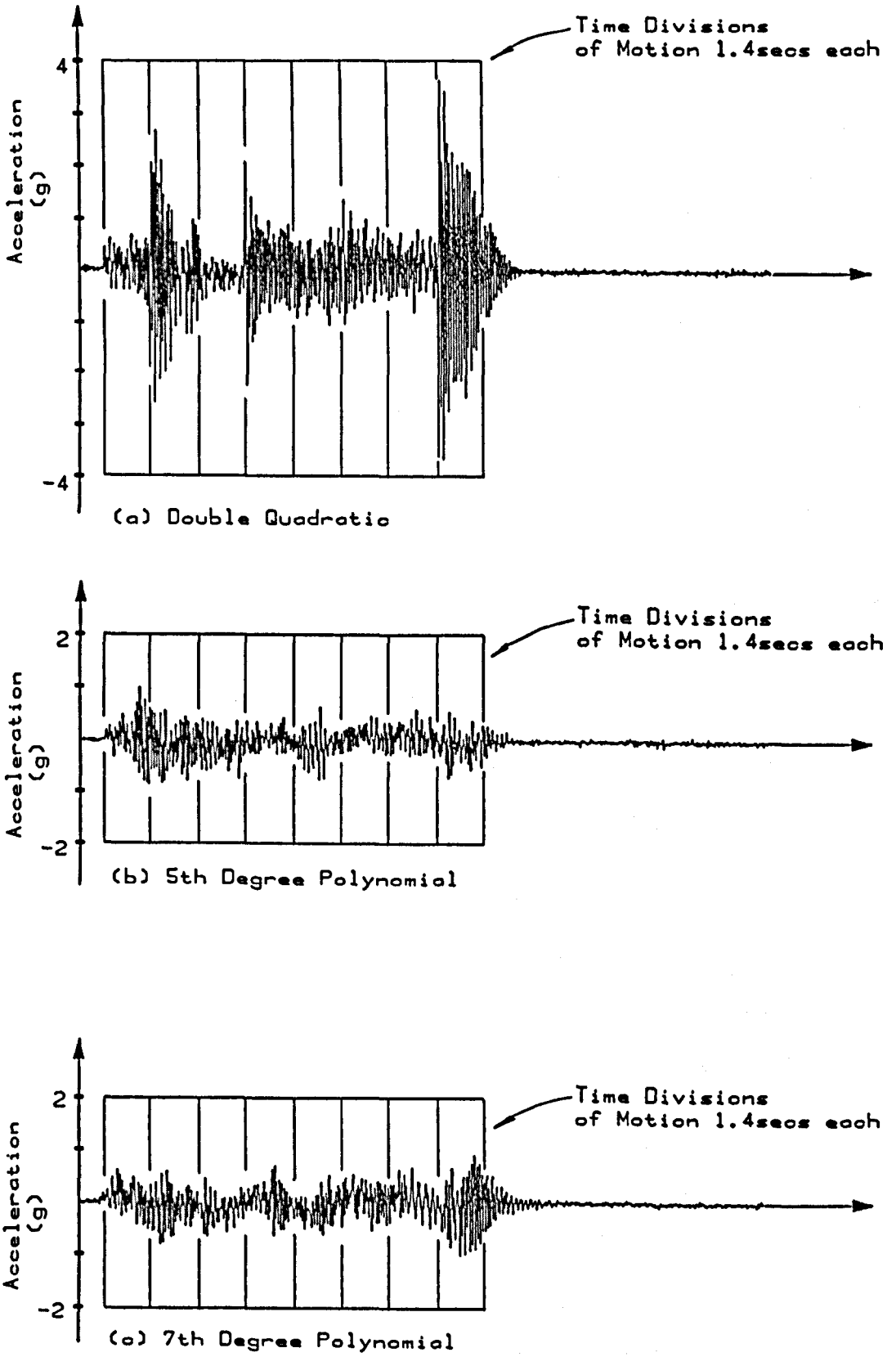


Figure 5.13 Horizontal Acceleration Responses from Planar Diamond

Two strain gauge type accelerometers were used, one aligned to the vertical, the other the horizontal. Their alignment was maintained approximately by maintaining the wrist fan axis horizontal, whilst the motion was performed. The vertical axis of the robot (which is visibly more oscillatory) generated higher vibration amplitudes than the horizontal axis. The laws tested were :

Double Quadratic	(Two successive quadratics,
(Bang-Bang Acceleration)	continuous in velocity value)
Fifth Degree Displacement Law	(Continuous in acceleration value)
Seventh Degree Displacement Law	(Continuous in jerk value)

From these albeit simple tests, a decrease in vibration amplitude is evident up to and including the fifth degree law. There is a subsequent increase in vibration amplitude with the seventh degree results. All the laws contained nominally instantaneous stops at the corners of the displacement paths. Boundary conditions at the corners included both zero velocity and acceleration. The polynomials were specified in terms of the Cartesian motions in a plane, then kinematically transformed to the joint displacement parameters, both angular and linear.

Displacement Coordinate	X (mm)	Y (mm)	t (sec)
d_1	1000	200	0
d_2	1200	0	1.4
d_3	1000	-200	2.8
d_4	800	0	4.2
d_5	1000	200	5.6

Table 5.14 Cartesian and Time Boundary Conditions

(for the Planar Diamond Test Path, figure 5.11, axis 4 servoed to hold an horizontal position, all other derivative boundary conditions zero)

The change of vibration amplitudes with the order of the laws is expected on the basis of other, usually single degree of freedom works. Clues as to why the amplitudes of vibration reach such a minima

arise, if the shapes of the laws are considered in the context of a single degree of freedom system (second order, undamped). As is seen in tables 5.15 and 5.16, increasing the order of the law, for the same zero derivative boundary conditions requires that the peak velocity increases. If the law is generated using a digital system with a constant sample interval, then increasing velocity implies increased step sizes in the discretised output. The displacement command 'impulse' is proportional to the degree of the law.

5.5.2 Polynomial Law Properties

The reason for the reduction in vibration magnitude for the initial increase in degree of polynomial is the reduction of acceleration discontinuities.

Normalised Symmetric Law	Peak Velocity	Peak Acceln.	RMS Acceln.	Acceln. Step	Peak Jerk
Double Quadratic	2	4	2	8	∞
Cubic Polynomial	1.5	6	3.464	6	12
Fifth Degree	1.875	5.774	4.140	0	60
Seventh Degree	2.188	7.500	5.045	0	42

Table 5.15 Symmetric Motion Law Parameters

If the values are further normalised to the first useable value in each column :

Normalised Law	Peak Velocity	Peak Accel	RMS Accel	Accel Step	Total Value (without jerk)
Double Quadratic	1	1	1	1	4
Cubic Polynomial.	0.75	1.5	1.732	0.75	4.732
Fifth Deg. Poly.	0.9375	1.44	2.070	0	4.448
Seventh Deg. Poly.	1.0938	1.875	2.523	0	5.492

Table 5.16 Normalised, Symmetric Motion Law Parameters

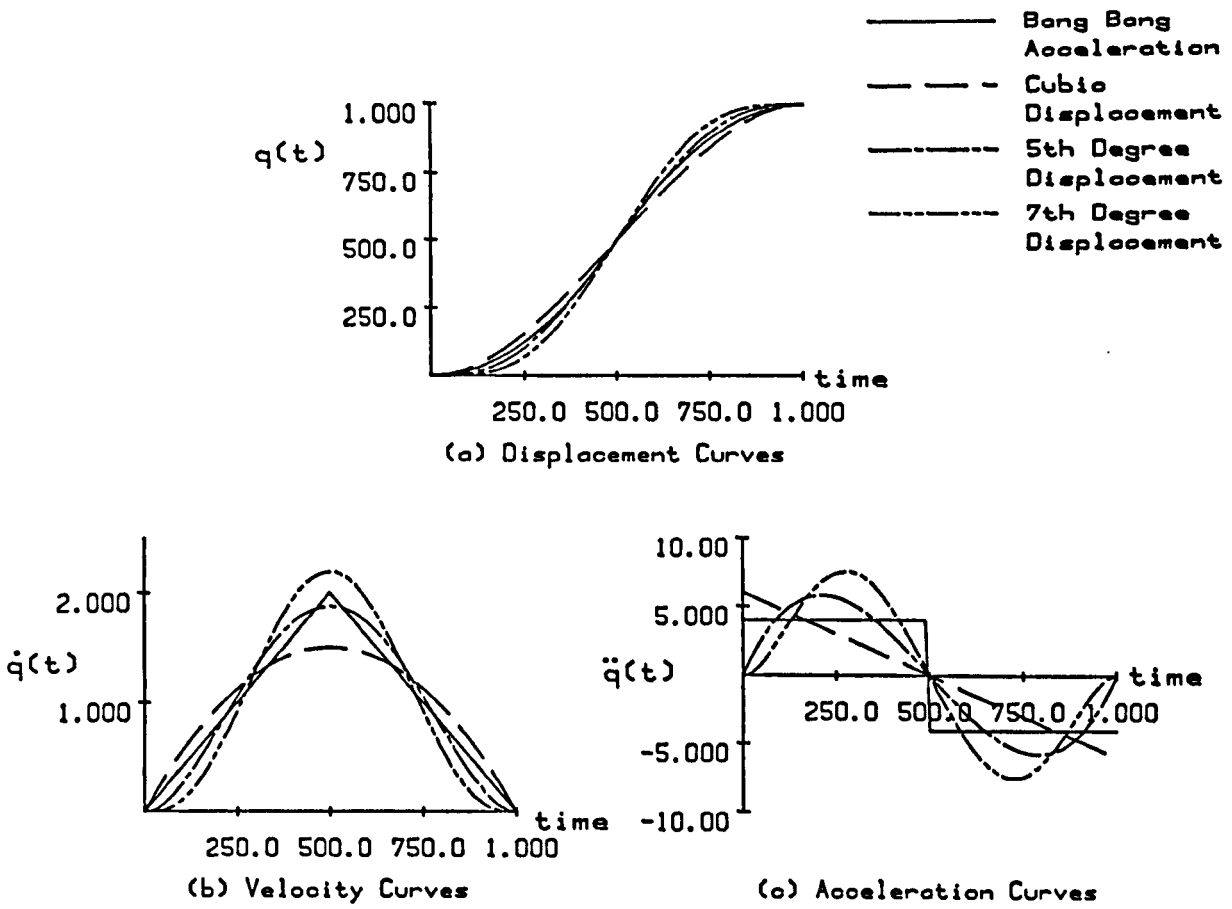


Figure 5.17 Polynomial Motion Laws

Referring to figure 5.17, the first two low order laws possess discontinuities in their acceleration curves. The double quadratic (bang - bang acceleration) contains three step changes in acceleration, hence demand force. The centre step, termed the cross-over, is the largest and creates most problems. It is particularly troublesome in systems containing backlash. These of course include robots. The cubic contains only two steps in acceleration, at the start and at the end of the motion. The fifth degree law contains no steps in acceleration, though two appear in the jerk curve. The seventh degree law has no steps in it up to the jerk curve, but the peak velocity is the highest. This is significant as it dictates the largest step sizes which appear in the digitised displacement command.

From table 5.16 the fifth degree law looks the more interesting. as there are no step changes in acceleration, and the peak velocity and accelerations are similar in magnitude to the double quadratic

'optimum' motion law. (Optimality in this context is expressed as the minimum RMS acceleration for the law). The robot inertias change significantly, as a function of joint displacement and so the optimal laws for robots cannot simply be defined in terms of the RMS acceleration. As a general selection guide, though, it is clear that a law's RMS acceleration is important.

When considering second order systems, the most important items as far as the initial oscillatory component goes are the initial conditions of displacement and velocity, relative to the command. The direct link between the acceleration command and response, although clear in the results is difficult to establish in an analysis. The oscillatory component viewed at acceleration is particularly pronounced. (An harmonic function, twice differentiated, is increased in amplitude by the circular frequency squared). The oscillatory response tends to track the same shape as the acceleration curve. The significance of jerk in the command is even more difficult to establish.

5.5.3 Polynomial Law Selection - The Influence of Sample Interval

The location of the vibration amplitude minima then depends on the relative values of system natural frequency, system sampling interval and the motion duration.

In a discrete sampled system, the size of the step change in command is related to the sampling rate. Classically the effect is assumed to be negligible so long as the sample rate is of the order of 2x, 2πx or 10x the most significant or highest distinguishable natural frequency. (In practice there are many factors influencing selection of sample interval). In order to put these hypotheses to the test a simple but computationally time consuming simulation was carried out.

There are interactions between sampling interval, motion duration and system natural frequencies. A second order mass spring system was simulated, being driven using normalised third, fifth and seventh degree laws. The system dynamic equations are :

$$m\ddot{y}(t) + k(y(t) - r(t)) = 0 \qquad (5.50)$$

or

$$m\ddot{y}(t) + ky(t) = kr(t) \qquad (5.51)$$

where

$$r(t) = \sum_{i=0}^{n} c_i t^i \qquad (5.52)$$

$$r(0) = 0 \qquad \dot{r}(0) = 0 \qquad \ddot{r}(0) = 0$$

$$r(T) = 1 \qquad \dot{r}(T) = 0 \qquad \ddot{r}(T) = 0 \qquad \text{for } 0 \leq t \leq T \qquad (5.53)$$

The peak absolute value of error resulting from the command polynomial is plotted in figure 4.2 for values of period ratio up to ten, and sampling ratios from 16 to infinity.

It is clear that to gain benefits from the higher order laws, high sample rates must be used. Sample rate itself is very important. Depending on configuration and load, the robot used in the tests has natural frequencies in the 8-16 Hz region. Motion durations for low, medium and high speed are therefore around 1, 0.5, and 0.25 seconds respectively.

Based on figure 4.2, the required sample rates to hold tracking errors down near the equivalent continuous system (ie Rs $\geq$ 64) are in the regions 100, 200 and 500 Hz respectively.

Note that the cubic motion law appears the best until very high sampling rates are achieved. There are both gross and transient components in the peak error magnitude curves. Wear rate increases with peak and RMS force, hence is related to peak and RMS accelerations. Power consumption in the context of motion law selection is related to the joint torque (or force) and velocity. Peak and RMS accelerations and velocities, therefore influence power consumption and hence motion law selection.

The fifth degree polynomial law is selected on the basis of the compromise between the requirements for zero step changes in acceleration, low RMS acceleration, low peak velocity and low peak acceleration. The vibration response (figures 5.12 & 5.13) is shown to be the best of the three laws tested, although this is not entirely supported by the simple undamped model responses of figure 4.2.

6 Self Learning for Tracking Accuracy Improvement

As described in chapter 5, a robot has all the pre-requisites for an autonomous self experimentation system. Based on such a system, algorithms are to be developed, enabling the system to increase its own performance, cycle after cycle.

6.1 Development of a Self Learning Tuning Algorithm

6.1.1 Consideration of Simple Parameter Estimation Techniques

Considering just two successive state vectors $\underset{\sim}{x}_{i+1}$ and $\underset{\sim}{x}_i$, under any circumstances there will exist constant matrices, $\underset{\sim}{P}$ and $\underset{\sim}{Q}$ valid over that one sample interval such that :

$$\underset{\sim}{x}_{i+1} = \underset{\sim}{P}\,\underset{\sim}{x}_i + \underset{\sim}{Q}\,\underset{\sim}{r}_i \qquad (6.1)$$

given $\underset{\sim}{r}_i$. Techniques are available for estimating the system parameters $\underset{\sim}{P}$ and $\underset{\sim}{Q}$ from the response, using 'least squares' and other methods. In the cases where the parameters are continually changing, once a solution has been found it can be tracked as the parameters change. If significant noise is present and the system parameters are changing rapidly, the accuracy of the estimate will suffer. Sahirad, Ristic and Besant 1986 [6.1] present such a scheme. A simple two link planar robot was modelled, but no results are presented.

For improved tracking accuracy, these linear parameter estimation based control schemes may offer a solution, although they would be more suitable given long and invariant system time constants.

An analysis of parameter estimation based systems leads to other possibilities. The linear approximation above may be re-arranged in order to compute $\underset{\sim}{r}_i$ to give some desired response :

$$\underset{\sim}{r}_i = (\underset{\sim}{C}\,\underset{\sim}{Q})^{-1}(\underset{\sim}{y}_{i+1} - \underset{\sim}{C}\,\underset{\sim}{P}\,\underset{\sim}{x}_i) \qquad (6.2)$$

where
- i - sample number
- t - time, $t = (i-1).\delta t$
- δt - sample time
- $\underset{\sim}{x}$ - state vector (n x 1)
- $\underset{\sim}{r}$ - input vector (m x 1)
- $\underset{\sim}{P}, \underset{\sim}{Q}$ - response solution matrices (n x n) and (n x m)

$\underset{\sim}{C}$ - output matrix (m x n)

Hence given $\underset{\sim}{x}_{i+1}$, $\underset{\sim}{x}_i$ and $\underset{\sim}{r}_i$ there are n equations with a maximum of n(n x m) unknowns. To evaluate all the unknowns, a minimum of n + m samples are required. If the system time constants are much shorter than the period (n + m).δt then solutions for the parameters are hopelessly out of date. In practice, truncation and noise would also add to the number of samples required to give solutions.

Utilising the fact that most robot tasks are repetitive and the robots themselves are repeatable, data from previous run cycles may be used to increase the accuracy of a parameter estimation process. If the number of the run cycle is denoted by k, then :

$$\underset{\sim}{y}_{i+1} = \underset{\sim}{C}\,\underset{\sim}{P}\,\underset{\sim}{x}_{i,k} + \underset{\sim}{C}\,\underset{\sim}{Q}\,\underset{\sim}{r}_{i,k} \qquad (6.3)$$

so if a change is made to the command in the demand $\underset{\sim}{r}$ at or after the ith sample, then :

$$\underset{\sim}{y}_{i+1,k+1} - \underset{\sim}{y}_{i+1,k} = \underset{\sim}{C}\,\underset{\sim}{P}\,(\underset{\sim}{x}_{i,k+1} - \underset{\sim}{x}_{i,k}) + \underset{\sim}{C}\,\underset{\sim}{Q}\,(\underset{\sim}{r}_{i,k+1} - \underset{\sim}{r}_{i,k})$$

or (6.4)

$$\delta\underset{\sim}{y}_{i+1,k} = \underset{\sim}{C}\,\underset{\sim}{P}\,\delta\underset{\sim}{x}_{i,k} + \underset{\sim}{C}\,\underset{\sim}{Q}\,\delta\underset{\sim}{r}_{i,k} \qquad (6.5)$$

where $\delta\underset{\sim}{x}$ - change in $\underset{\sim}{x}$ between successive runs

$\delta\underset{\sim}{y}$ - change in $\underset{\sim}{y}$ between successive runs

$\delta\underset{\sim}{r}$ - change in $\underset{\sim}{r}$ between successive runs

If the system is repeatable :

$$\delta\underset{\sim}{x}_{i,k} \approx \underset{\sim}{0} \qquad (6.6)$$

hence

$$\delta\underset{\sim}{y}_{i+1,k} = \underset{\sim}{C}\,\underset{\sim}{Q}\,\delta\underset{\sim}{r}_{i,k} \qquad (6.7)$$

giving a further m equations in m x m unknowns. Equation 6.3 gives m equations with an additional m x n unknowns, hence only half the number of samples are required to estimate the system parameters. By including data from runs k-1, k-2 etc. the process could be used to overcome some of the problems.

6.1.2 Expansion to a Self Learning Algorithm

At this juncture, it is as well to recall the purpose of attempting a parameter estimation. That is, the knowledge of the parameters enables prediction of the input required now, to give a desired response at some point later. Assuming that this process has been successfully achieved up to and including the current sample i by some as yet unspecified process, then again :

$$\underset{\sim}{x}_{i,k+1} \approx \underset{\sim}{x}_{i,k} \qquad (6.8)$$

so equation 6.7 still holds. The error vector $\underset{\sim}{e}$; at the next sample is

$$\underset{\sim}{e}_{i+1,k} = \bar{\underset{\sim}{r}}_{i+1} - \underset{\sim}{y}_{i+1,k} \qquad (6.9)$$

where $\bar{\underset{\sim}{r}}$ denotes the nominal or reference command, which is constant, irrespective of the run number, k. The condition for zero error at the next sample (i + 1) is satisfied if the command at the previous sample is changed as follows :

$$\delta\underset{\sim}{r}_{i,k} = (\underset{\sim}{C}\ \underset{\sim}{Q})^{-1}\ \underset{\sim}{e}_{i,k} \qquad (6.10)$$

This requires knowledge of the elements of $\underset{\sim}{Q}$, in turn necessitating analysis and parameter estimation for the particular scheme. In hindsight, experimental results show that some very crude approximations hold adequately in at least one case. The extension of these approximations to more general manipulators requires justification.

The change in response resulting from any change in command between cycles k and k+1 can be expressed as :

$$\underset{\sim}{y}_{i+1,k+1} - \underset{\sim}{y}_{i+1,k} = \underset{\sim}{C}\ \underset{\sim}{Q}\ (\underset{\sim}{r}_{i,k+1} - \underset{\sim}{r}_{i,k}) \qquad (6.11)$$

and for reasonably accurate feedback control, tracking gives :

$$\underset{\sim}{y}_{i} \approx \underset{\sim}{r}_{i} \qquad (6.12)$$

therefore

$$\underset{\sim}{C}\ \underset{\sim}{Q} \approx \underset{\sim}{I} \quad \text{(the identity matrix)} \qquad (6.13)$$

hence

$$\delta\underset{\sim}{r}_{i,k} \approx \underset{\sim}{I}\ (\bar{\underset{\sim}{r}}_{i+1} - \underset{\sim}{y}_{i+1,k}) \qquad (6.14)$$

Now this corresponds to the response after one cycle only. Because it is only an approximation and using the maximum correction is equivalent to marginal stability, some conservatism is wise. Hence the algorithm becomes :

$$\underset{\sim}{r}_{i,k+1} = \underset{\sim}{r}_{i,k} + \underset{\sim}{L}\ (\bar{\underset{\sim}{r}}_{i+1} - \underset{\sim}{y}_{i+1,k}) \qquad (6.15)$$

where

$$\underset{\sim}{L} = \begin{bmatrix} L_{11} & 0 & \cdots & 0 \\ 0 & L_{22} & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots\cdots & 0 & L_{mm} \end{bmatrix} \qquad (6.16)$$

and $\quad 0 < L_{jj} < 1 \quad$ for $j = 1,2,\ldots,m$

The scheme is shown in figure 6.1. Its function is to add in a proportion of the previous run's error to the command used on the next run. The main points are that the error is automatically scaled to the command; It has resulted from the actual response of the system

including an 'exact' model; over a small piece of motion there is a linear relation between the command and response.

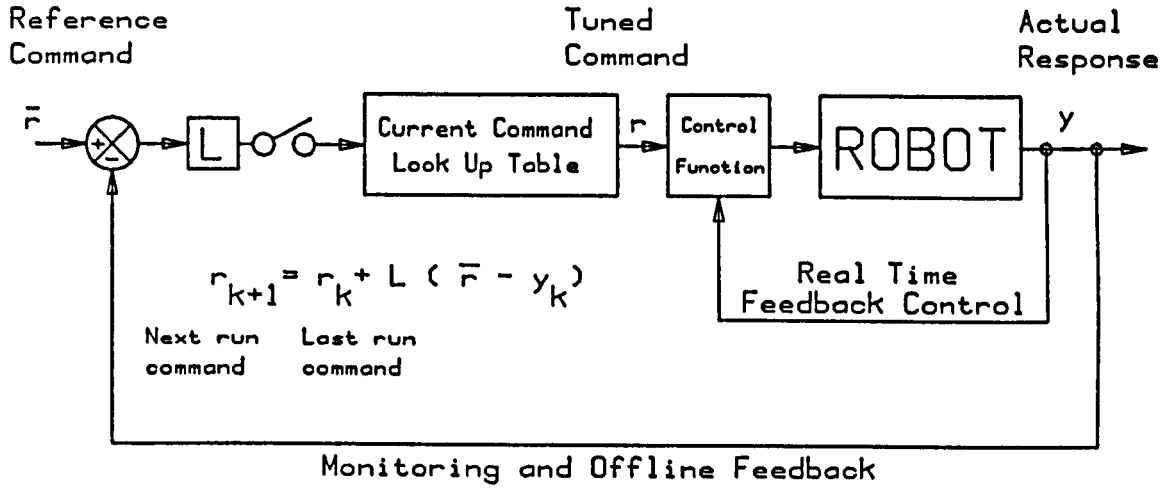


Figure 6.1 The Self Learning Motion Tuning Scheme

In specific cases, the validity of the $\underset{\sim}{y} \approx \underset{\sim}{r}$ assumption may be verified by plotting the input and response against each other. For other than saturated trajectories or very poor feedback control, the curve has close to unity slope and its intercept passes through the origin.

6.2 Practical Considerations

6.2.1 Modifying More Than One Sample

The scheme as it stands requires that only one sample per run be modified. This produces two problems; one is that the command acquires a large step in it, between the sample being tuned and the rest of the command; the other is that the number of runs required to fully tune a motion becomes very large.

Broad, simplistic assumptions enable the development of an empirical method for increasing the practicality of the scheme.

Stable linear tracking system responses are constrained to lie within

an envelope which can be defined by sums of exponentially decaying terms, centred around the nominal command curve. By superposition, this will apply to the dynamic response arising from the changes made in the tuned command.

Rather than modify only one sample per cycle, all samples ahead of the current datum sample may be tuned in the same way as the datum sample, but using a progressively increasing attenuation. If this is also defined by an exponential decay, then the sum of the responses due to the sequence of 'impulses' arising from the sequence of changes made to the command must also decay, relative to the nominal command.

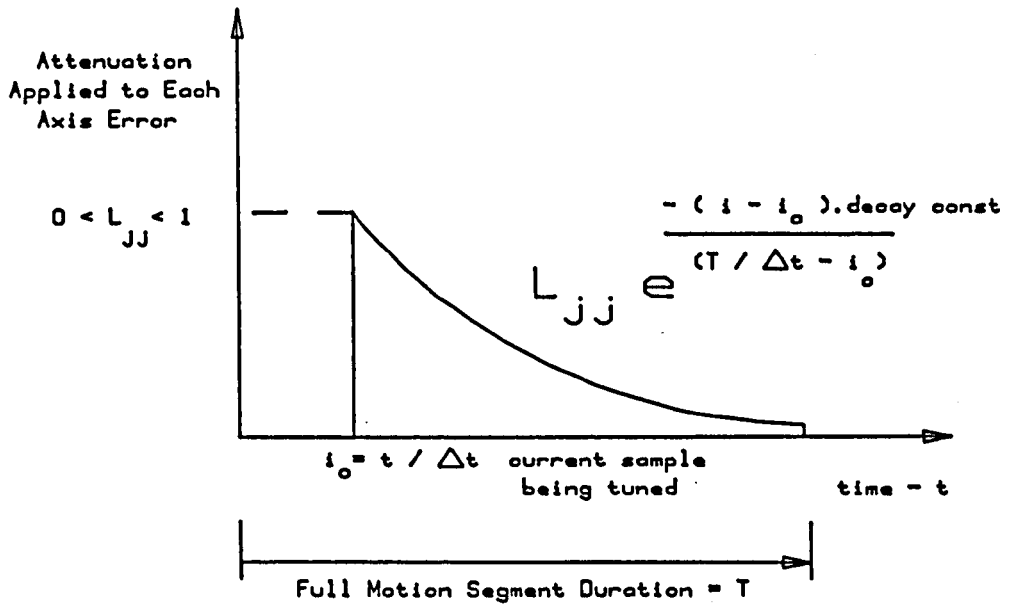


Figure 6.2 Window Function Employed to Speed up the Learning Process

6.2.2 The Decay Weighting Function

The form of the decay function applied to the Ljj factor is as shown in figure 6.2. Note that it has a non-zero value from the current datum sample all the way to the end of the motion segment undergoing tuning. The process is similar to the use of Hanning windows, etc, on time domain signals, hence the function is subsequently referred to as a 'window'. Computation times for exponential functions are lengthy, but the functions themselves are well behaved. In the actual scheme

therefore, a coarse look up table has been used with interpolation, to greatly reduce the computational load associated with the window.

There must be some finite value of decay at the tail end of the window function, by virtue of the exponential function itself. This is undesirable because the modification made to any one sample of the tuned command is the accumulation of the sum of each error times the appropriate value of attenuation. At the tail end, the tuning process will have been applied many more times to any given sample, so there will be an increased tendency to accumulate noise in the tuned command. A second reason zero modification is required at the tail end is that the tuned command must blend into the section of command following it. There will otherwise be a step change in displacement command at the end. The window function can be made to effectively taper away to zero by utilising the 12 bit (or whatever) resolution. (Rounding reduces the small gain values to zero.)

Of course a robot's dynamics are not linear, but assuming that we are operating close to a nominal command and the changes in the displacement response are comparatively small, then a linear approximation over any one sample (and probably a small region) can be justified.

6.2.3 Limits on Error Improvement

System noise and repeatability place a limit on the accuracy to which the response can be tuned. This limit is found in our case to be two or three units (of 12 bit integer resolution). It is therefore unrealistic to set zero as the target error. These 'satisfactory' error limits as set for each axis are found to differ. A small amount of experimentation is sufficient to establish the values. The noise spectrum is not entirely random, there are components due to torque or force ripple, and mains. There is also a dc component due to drift in various amplifier offsets. It is difficult to reduce the effect of noise, because of these properties. Without any consideration of noise, the accumulated result is an extremely rough looking tuned command signal.

Certain types of noise can still be a problem. Power supply spikes can

be added into the tuned command. With a high quality, well designed commercial robot, these problems would be less significant. A single pole digital filter acting on the change made to the tuned command reduces the effect of noise, although it increases the peak errors. All responses are shown without any filtering.

The error limits need to be a compromise. If they are too large, then tuning will only take place every (say) 5 samples, sufficient to knock the response within the bounds of tolerance on the reference command. The result is the tuned command adopts a 'saw tooth' like profile, and the robot response is oscillatory, but usually within the stated tolerance values. If the tolerance is set too tight then the robot would simply repeat the motion ad infinitum. To avoid this possibility in a case such as the robot being overloaded, (or the tolerance being set too tight), a limit is placed on the number of cycles applied per datum sample.

Once the current datum sample is tuned to the required accuracy level, then the system indexes on to a new value (sample number) in the motion segment which includes at least one axis error outside the bounds.

6.2.4 Phasing of the Applied Correction

The change in response, due to a change made in the command only one sample earlier, cannot of course be very large. Some experimentation was carried out to establish if there were any benefits in processing the command based on a phase shifted lead of the response. This cannot be performed in normal real time feedback control because of the stability implications. Alternatively, it would imply accurate prediction of the response ahead. As the tuning process is effectively offline, then this restriction does not exist in the same way. The phase shift was found to interact with the higher frequency components in the response. A small oscillation, given the appropriate phase shift could be reflected in the response ahead. A limit is therefore placed on the maximum phase shift allowed. For a sample interval of 10ms, phase shifts of 1 to 10 samples were tested. Processing the ith sample based on the measurements of error at the sample ahead; i+5 was found to significantly increase the speed of convergence.

Interestingly, although this implied a 0.05 second lead and there were oscillations in the response of the vertical axis of around 0.07 seconds period, there appeared no interaction.

6.2.5 Extra Hardware Requirements

The data processing required can be carried out during any available dwell period after or during each run of the self tuning motion. Memory requirements will vary depending on the actual implementation, but the calculations prove fairly simple. The Intel 8086 based microcomputer used, with Microsofts' MSDOS and MSPascal took less than 0.7ms computation time per sample, with three axes being processed. There are numerous strategies which could be employed to reduce the amount of memory required. In the experimental system, much more information was logged in memory than would be necessary for a commercial system. The user would only specify the self tuning process for those segments requiring it. Further, it may only be worthwhile on the major axes of the robot.

The memory requirements in the case where the tuning is carried out after each run are as follows. The response is logged for the whole segment. The tuned and reference command motions would also be stored in memory. (Memory for the reference command is not an additional requirement, it would be need to be stored anyway or possibly computed during the motion.) For six axes, 12 bit resolution and 100 Hz sample rate this implies a total memory requirement of :

6 axes x 100 Hz x 2 bytes x 2 trajectories x 60 secs / 1024
= 141 kbytes additional memory per minute of tuneable motion

If the tuning was computed during each sample interval, then the response would not need to be logged. Selecting only three axes for tuning :

3 axes x 100 Hz x 2 bytes x 1 trajectory x 60 secs / 1024
= 35 kbytes additional memory per minute of tuned motion

Depending on the quality of the hardware, system RAM costs have reduced dramatically in recent years. Typical figures for on board costs are in the 20 to 40p per kbyte region. Extra costs due to memory requirements for say ten minutes of tuneable motion are therefore around £105, i.e. as an optional extra probably less than 0.5% of the cost of a basic robot.

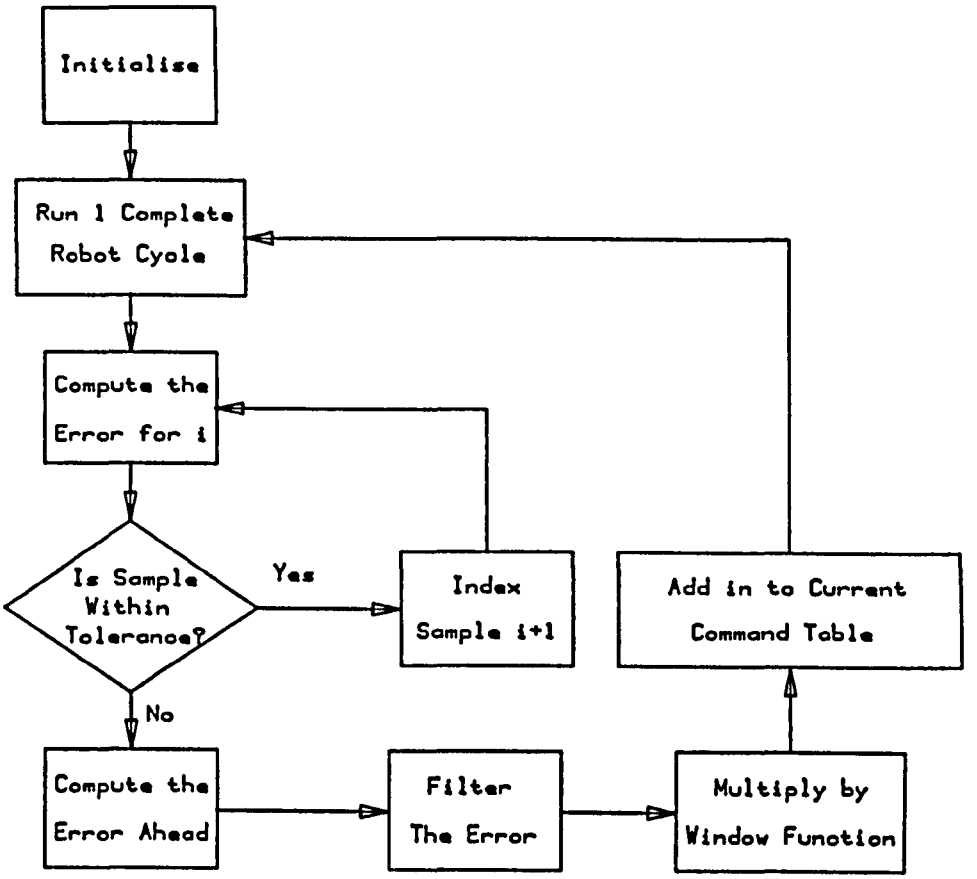


Figure 6.3 Flow Chart for Self Learning Scheme

Figure 6.3 outlines the basic steps in the logic of the software, these being one of a number of possible implementations. The actual routines are located in a module called MODMVACC.PAS which forms one of four modules making up the program RUN.PAS. This in turn is part of a suite of programs which comprise a manual robot motion generating system. It is manual in the sense that all the motion parameters need specifying through the keyboard. Various other parameters are set in data files which can easily be edited. In this way, none of the component programs making up this system are specific to any robot.

6.3 Development of the Implementation

6.3.1 Single Axis Simulation

A single axis dynamic simulation was used as a starting point. The

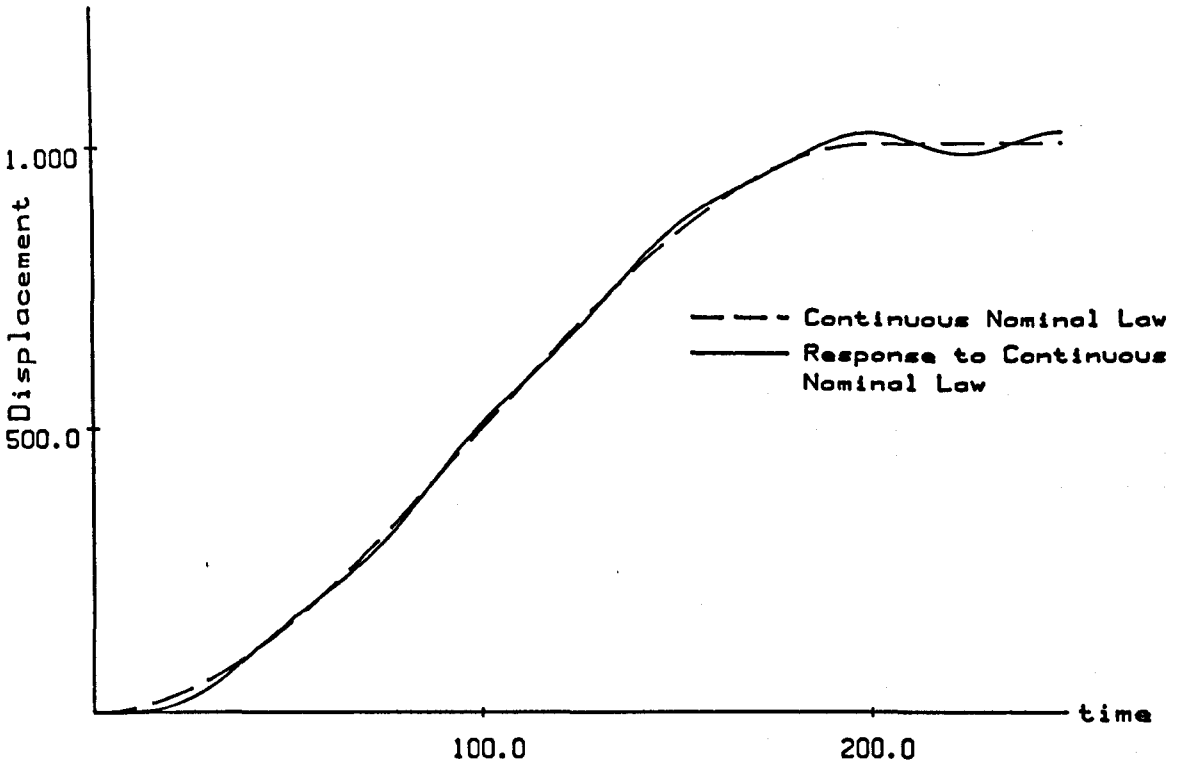
motion of a mass spring system was controlled by varying the input motion at the spring. The lack of damping simplified the problem. In the simulation self learning and analytic (model) based tuning could be compared. In the presence of damping, model based tuning results in discontinuous displacement commands. These are unattractive in real systems.

The motion period ratio was chosen as 4 to make the results more interesting. Hence, a system natural frequency of 20 Hz required a motion duration of 0.2 seconds. For simple boundary conditions; the displacement ranging from 0 to 1 unit; zero initial and final velocities, a cubic law was adequate. The sample frequency for the discretised input was 100 Hz. In all cases (figures 6.4 to 6.7), the responses plotted are pseudo continuous. The response data presented has a higher resolution than the sampling frequency would normally allow. (In the experimental results presented later, data is only available at the sampling points). On the displacement curves (figures 6.4 to 6.6) differences are difficult to resolve, so the displacement error curves (figure 6.7) are interpreted instead.

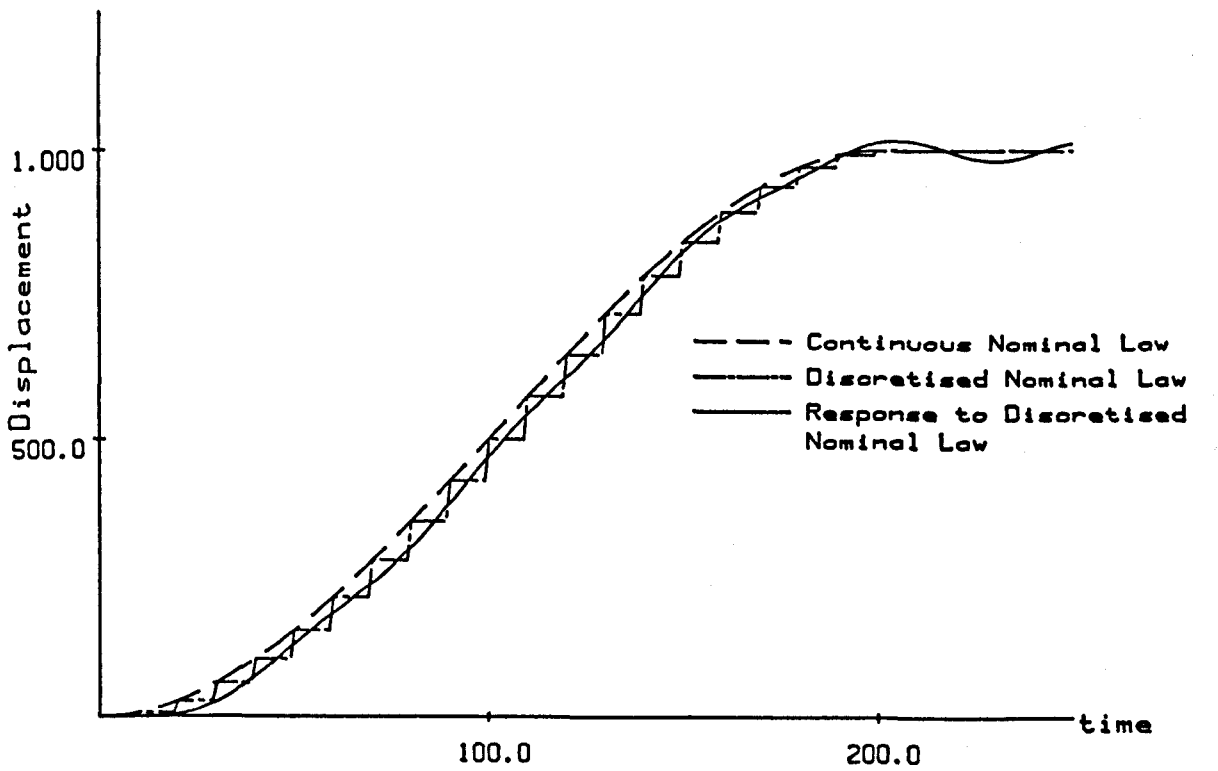
The error response of the system to the continuous cubic is a more or less constant frequency and amplitude oscillation. Its centre line is the same shape as the acceleration curve for the law, i.e. a straight line, skewed to the horizontal.

The response to a discretised cubic is not as predictable. Two additional components are manifest, making the system almost appear as if it was of a higher order. There is a gross lag arising from the sampling interval. Its peak magnitude approximates well to the vertical step size around the central region of the displacement command. The apparent 'second harmonic' response results from the discretisation process. The effects due to the sequence of steps are attenuated, as the sampling frequency in this case is 5 times higher than the system natural frequency.

Tuning the continuous cubic based on the model parameters results in a line of zero error, i.e. superimposed on the horizontal axis. If this tuned cubic is then discretised and used as the command, there is only a slight improvement in the response over the untuned cubic. No

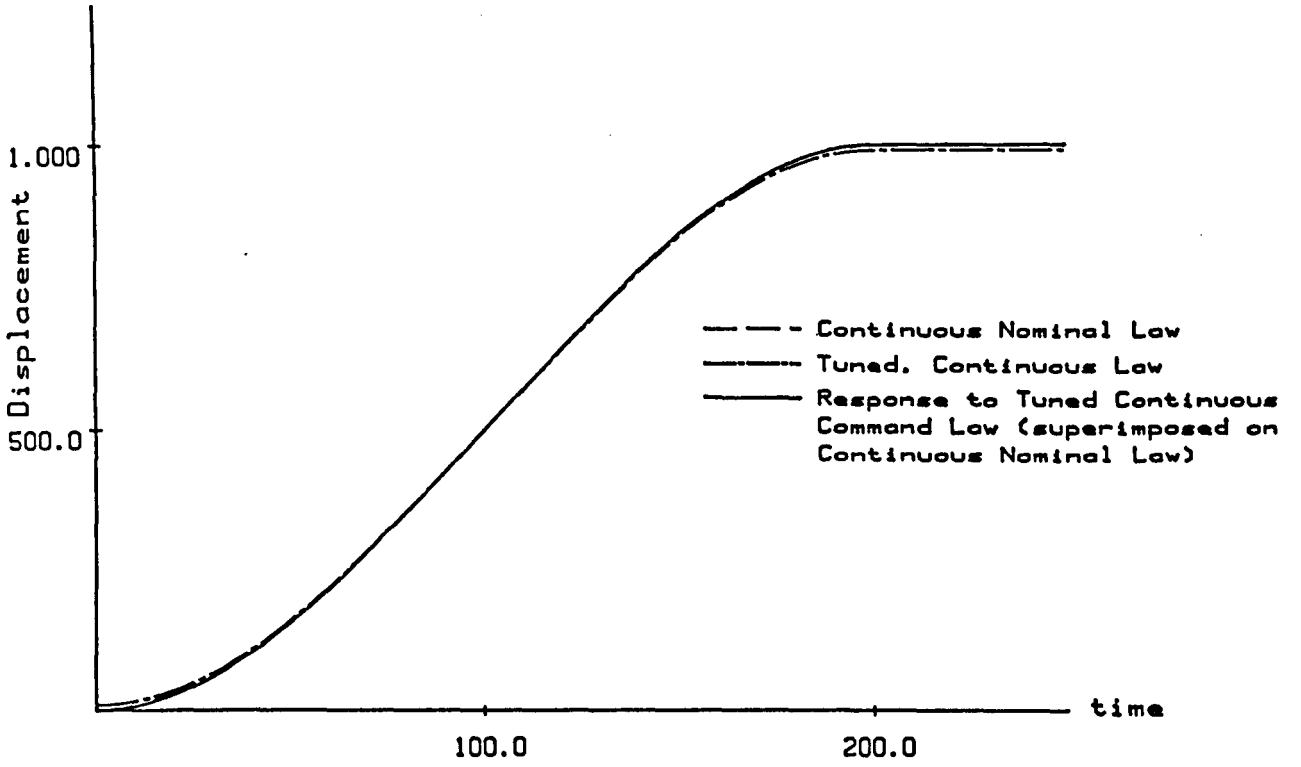


(a) Response to Continuous Law

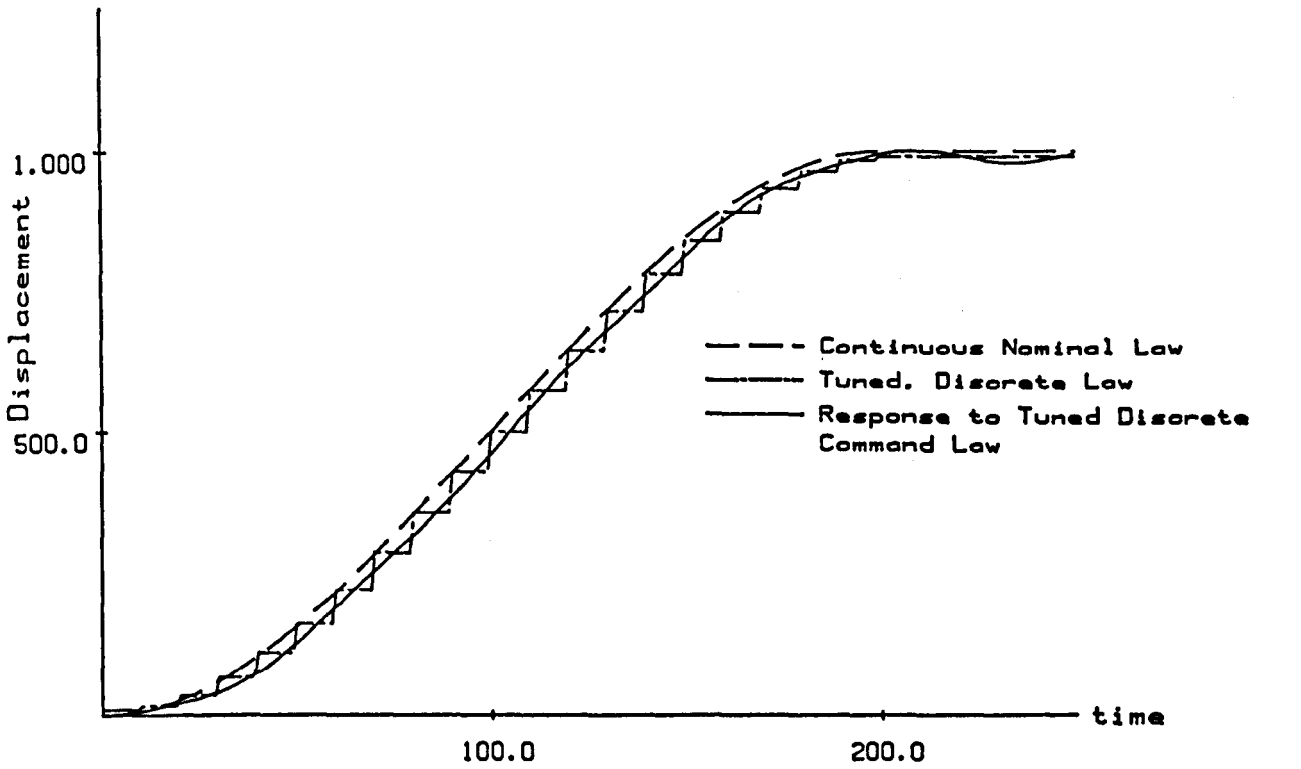


(b) Response to Discrete Law

Figure 6.4 Simulated Responses to Continuous and Discrete Cubic Motion Laws



(a) Response to Continuous Law



(b) Response to Discrete Law

Figure 6.5 Simulated Responses to Continuous Model Based Tuned Laws

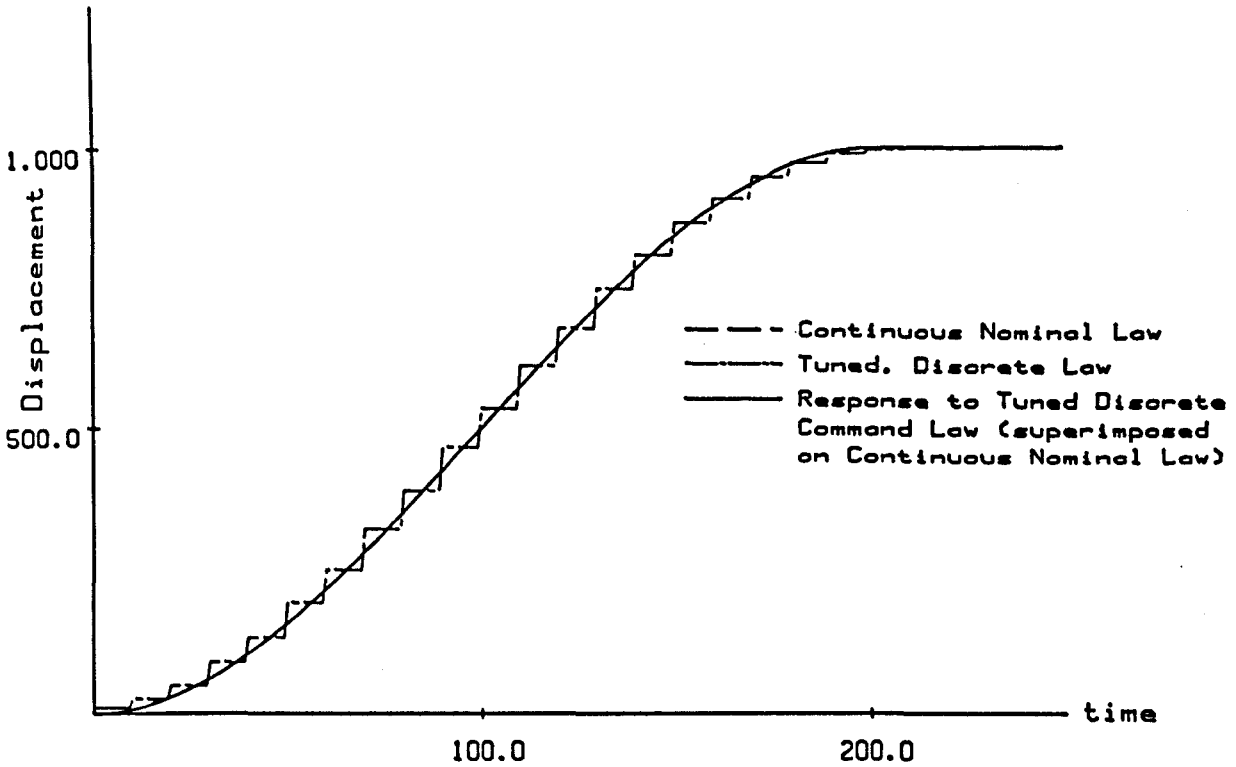


Figure 6.6 Simulated Response to a Self Tuned Law

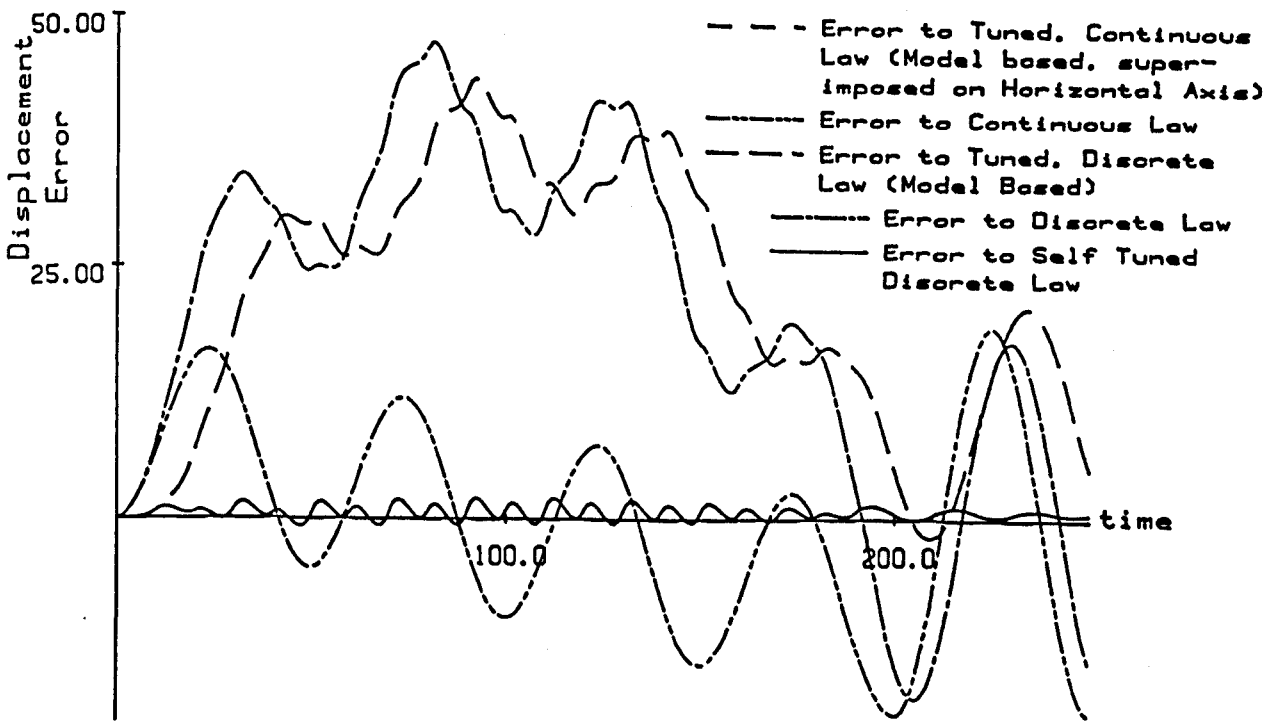


Figure 6.7 Error Curves for all 5 Cases

account has been taken in the tuning of the discretisation. There is however, around a 10% reduction in the peak error, and the significance of the 'first harmonic' has decreased. A perfect response can be obtained at the sample points, by using a discrete response solution and inverting it. Command velocity must be available, which is not always possible. This method is pursued further in section 7.3.

Applying the self learning process gave the achievement of at least the set tolerance at the sample points. In normal systems, the small size of steps due to discretisation combined with the relatively long time constants should cause the 'ripple' error to be small. The final response has negligible gross or first 'harmonic' components, just that of the 'ripple' from the sampling process. The lack of regularity in the response probably derives from the varying number of learning cycles per sample, due to the varying degree of difficulty with which the scheme attains the desired tolerance at each sample. Note at this point, the tuning process possessed none of the refinements such as the window function or filter etc. Another interesting feature of the simulation is that its computation can be carried out faster than the modelled time period. To observe the process effectively, required the insertion of delays into the routines.

6.3.2 Single Axis Implementation - Low Performance Drive

The first actual implementation used a 'Feedback' educational quality DC servo motor fitted with a 30:1 ratio worm gear box (figures 6.8 and 6.9). Servo control was implemented digitally (displacement error proportional with velocity feedback) and adjusted to give a maximum tracking error of around 10% of the motion displacement range. The displacement range of 1.57 radians on the gearbox output shaft took place in 0.8 seconds. The sampling frequency was again set at 100 Hz. The window function was not used at this point, so many cycles were needed to achieve the tuned response.

The progressive stages in the learning process are seen in figures 6.10(a) to 6.10(d), corresponding to : untuned, one quarter, half and the complete motion having been tuned. Without any filtering of the response error before it is used to augment the tuned command, sharp features arise. The presence of considerable backlash in the worm

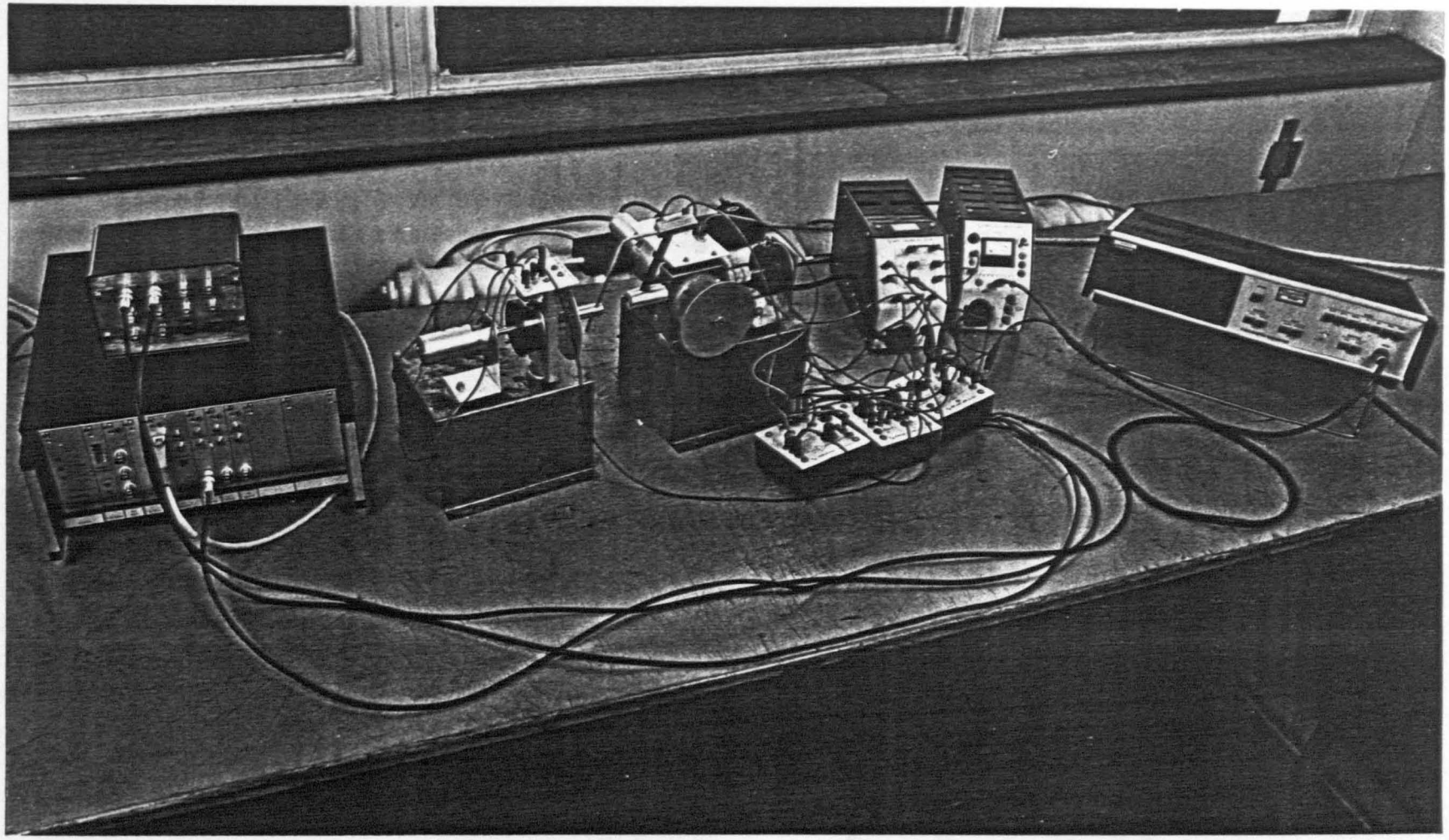


Figure 6.8 'Feedback' Servo System and Interface

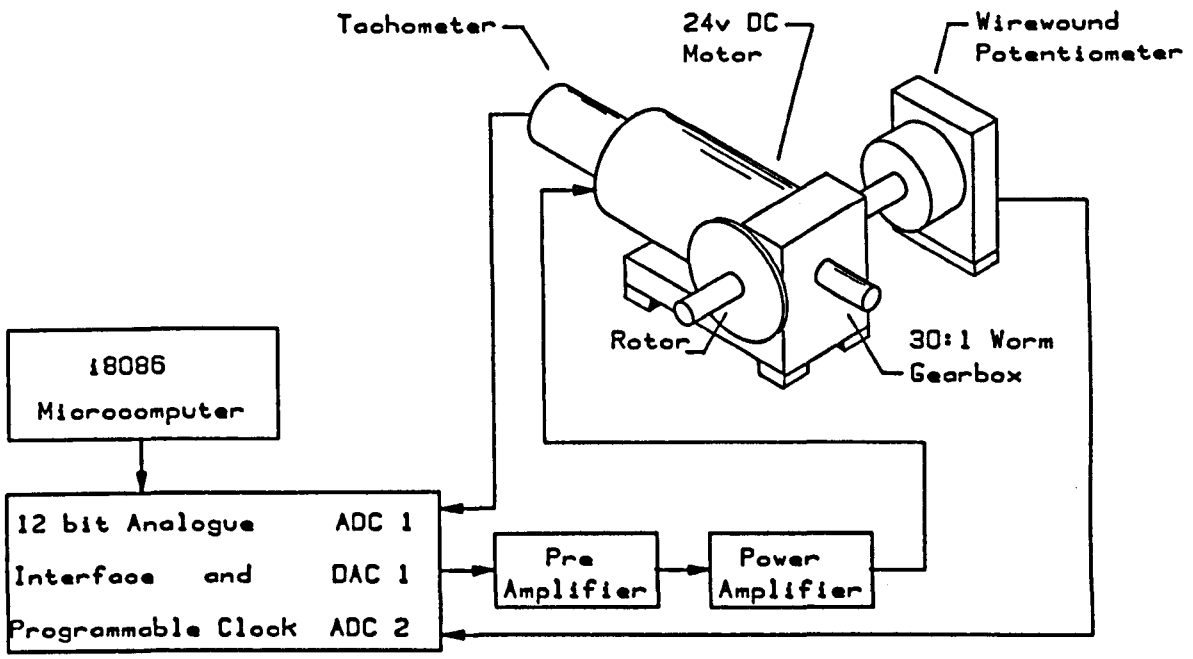
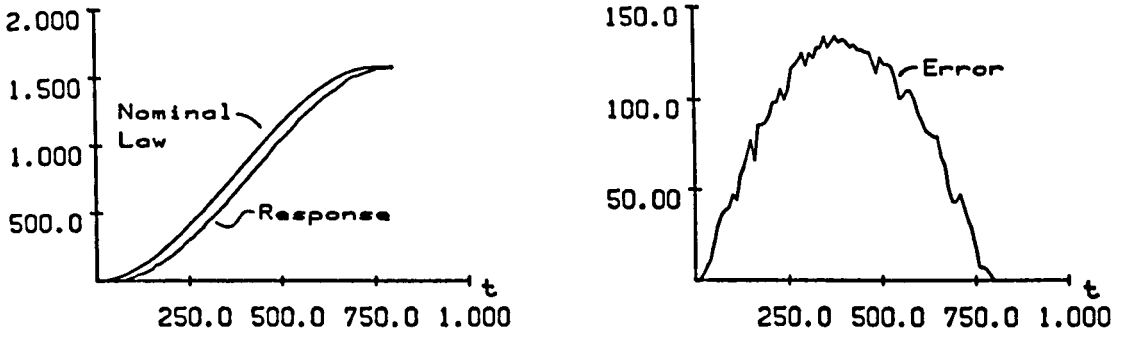
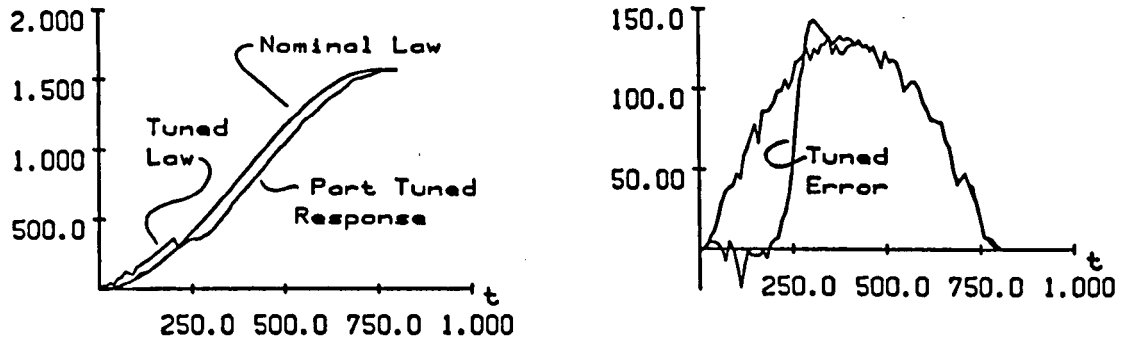


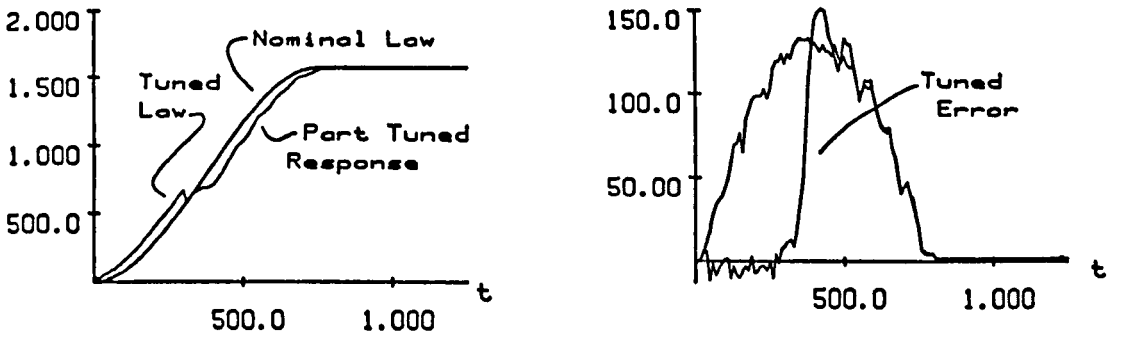
Figure 6.9 Schematic of 'Feedback' Servo System



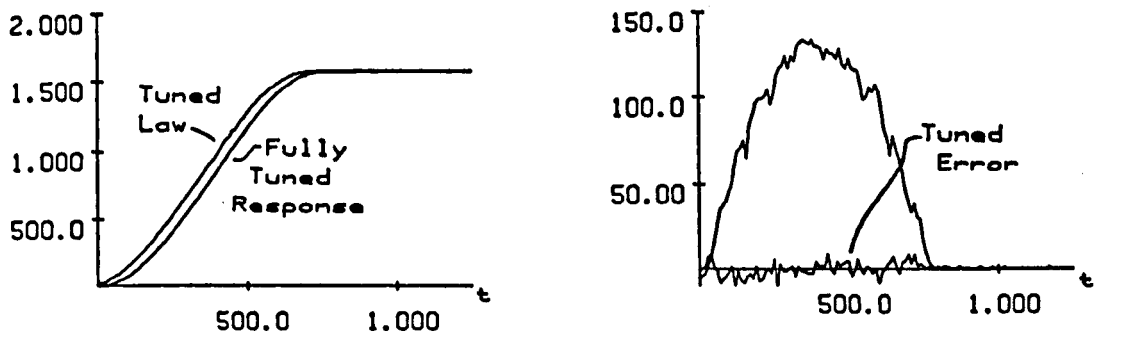
(a) Untuned Command and Response



(b) One Quarter of Motion Tuned



(c) One Half of Motion Tuned



(d) Motion Completely Tuned

Figure 6.10 Self Learning Based Tuning - 'Feedback' System

gearbox served to aid the development of the filtering process used in the final, robot implementation. The other three figures include the filter.

During the tuning process, the tuned command developed a sharp kink in it, e.g. figure 6.10(b), past the current datum sample. This is one of the reasons that some form of 'blending' is necessary, from the tuned law to the reference command. The fully tuned error curve has a similar peak error magnitude as that of the gearbox backlash. The error bounds were selected based on the backlash.

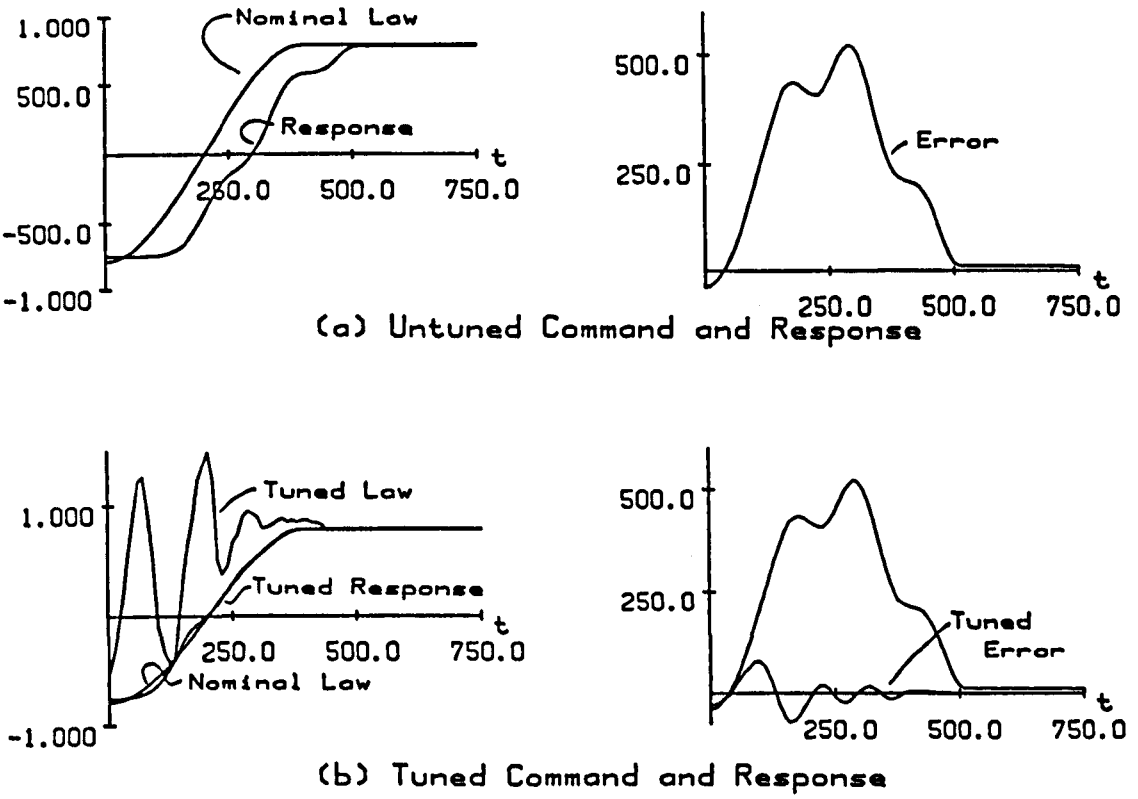


Figure 6.11 Self Learning Based Tuning - 'Aerotech' Motor

6.3.3 Single Axis Implementation - High Performance Drive

The next step was to try the scheme out on a high performance drive, similar to those used on industrial robots. The 'Aerotech' brushed dc motor was tested in a direct drive arrangement, figures 5.3 and 5.4. Development this time was aided by reverting to analogue control. Gain changes, debugging and signal conditioning were facilitated. To keep the IEEE488 communications overhead to a minimum, IEEE driver routines had been used with minimal error checking or handshaking. (Certain events caused the system to hang up, and it was impossible to establish whether the fault was related to the IEEE routines or the feedback control routines for example, exceeding their 16 bit integer arithmetic range. Performing control separately on the analogue computer alleviated this problem).

To test the learning system's capabilities, feedback gains were set to give both a low accuracy and oscillatory response. Backlash and static friction were found to be minimal. The system natural frequency (closed loop) was approximately 6.7 Hz, so a sample rate of 100 Hz was adequate. The gross error component is virtually eliminated (figure 6.11), and the oscillatory component attenuated toward the end of the motion. Note the steady state error with tuning is better than that without. This will only occur if the system repeatability is high. The tuned command is discontinuous and highly oscillatory, but the error response has been improved. The tuning process with the later refinements may have improved the situation.

6.3.4 Multi-Axis Implementation - Industrial Robot

The target implementation was the Hall Little Giant robot, figure 6.12. The three major axes were used in the tests, all being hydraulically driven using Moog 76 series flow proportional servo valves. Analogue position control comprised a proportional gain and a stabilising pressure feedback on the swing and vertical axes only. The experimental scheme was controlled by the Apricot PC, an i8086 based commercial microcomputer, in Pascal. An 8087 maths co-processor and 384 kbytes of memory enhanced system performance. In principle, the structure of the software was similar to the two earlier implementations. Many modifications and additions were made to give

greater versatility, easier adaptation and allow the same program to be used when running both the manual and the automatic motion generation schemes. These include the capability to handle any number of motion segments of various types and any number of machine axes subject only to memory limitations. The program modules are listed and described in appendix B.

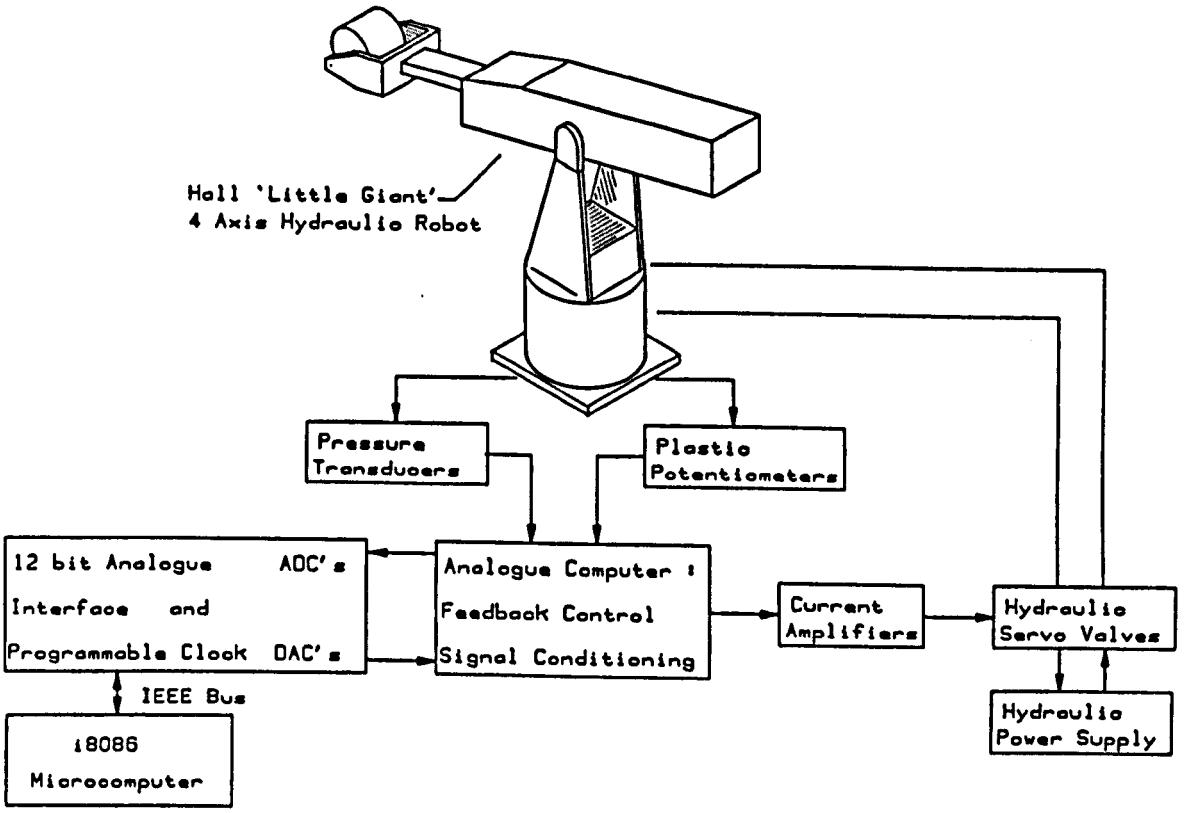


Figure 6.12 Control System Hardware for Self Learning System

The analogue interface contained three, 12 bit DAC's and one 12 bit ADC fitted with a 32 channel multiplexer. There was also a programmable real time clock used to control the sampling interval and the multiplexer synchronisation. Using the IEEE 'unchecked' routines, the multiplexer spasmodically went out of step, reading channels 1 for 2, 2 for 3 and 3 for 1. This would wreak havoc with the system. Operations such as accessing memory outside of the normal addressing range or writing characters to the VDU during real time control appeared to increase the risk of it occurring. The sample interval was

set to 100 Hz. Control gains were empirically maximised to yield high static accuracy consistent with stability requirements. Small increases in the gains chosen would yield instability in certain configurations. The position feedback was provided by 0.1% linear potentiometers. The differential feedback provided by Hall's custom manufactured strain gauge based transducers was found to be extremely noisy and induce steady state errors which varied as a function of position. Analogue filters were developed to improve the signal and extract the dc component. The signal conditioning, buffering and control circuits used are shown in figure 6.13.

The variety of possible motions is of course infinite, so one specific motion comprising about 25% of each axis full stroke was selected for most of the tests. The conditions defining the motion are shown in table 6.14. The durations of the rise and fall segments were varied from 1.26 seconds to 2 seconds on the main test motion. The nominal motion command is defined by a fifth degree polynomial and has zero velocity and acceleration constraints at each end. The three dwell segments were set at 1,1 and 0.5 second durations. The results for the main test motion are presented in figures 6.15 to 6.22 inclusive. The vertical axis annotation is in terms of the 12 bit integer units (0 to 4095). Leaving the data in terms of these units aids the interpretation of the error curves by effectively normalising the results. Horizontal axis numerics are in terms of the number of sample intervals, each one of which being 10 ms, for example 250 is equivalent to 2.5 seconds.

Referring back to the learning algorithm, a conservative $\underset{\sim}{L}$ matrix was initially chosen with diagonal elements of 0.8. A subsequent change to 0.9 increased the rate of convergence without detriment. The axis error limits were set as 2,3 and 4 units for the swing, vertical and horizontal axes, respectively. These figures correspond approximately to the static errors and noise on each of the axes. The horizontal axis was prone to higher positioning errors, for at least two reasons. Firstly without pressure feedback, the loop gain had to be lower. Secondly the hydraulic cylinder on this axis has a long stroke and is very difficult to bleed. Air in the cylinder caused an effect similar to backlash, leading to an intermittent and disturbing hammering action.

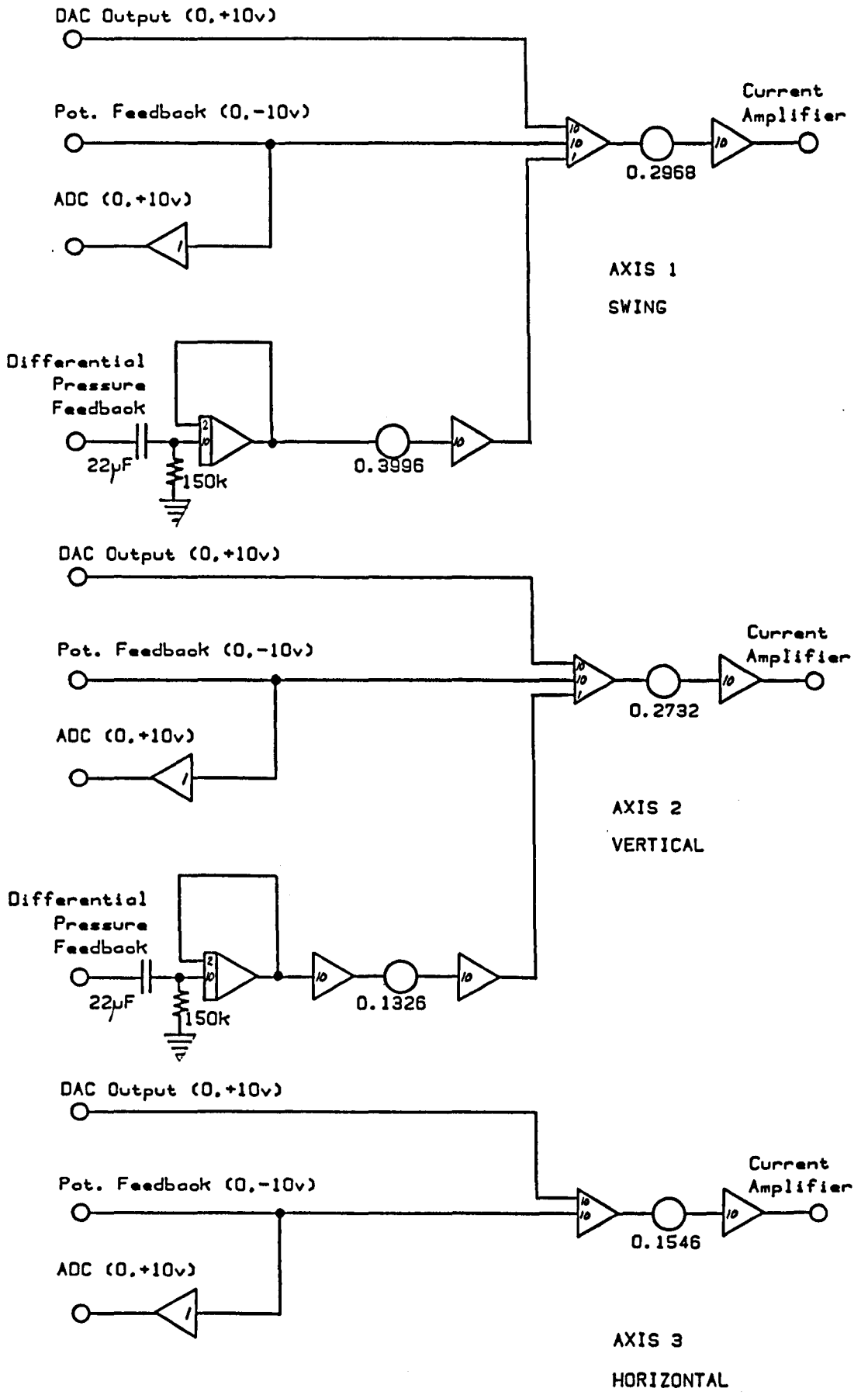


Figure 6.13 Signal Conditioning Buffering and Control Circuits

The number of cycles required to fully tune the command motion law varied in the range 7 to 24, although 12 to 15 was typical. Certain of the error curves exceed the error bounds set. Note that for comparison purposes only the lift segment was tuned. This means the total change made to each of the command laws, only has a non zero value during the lift segment. This makes it easier to distinguish it from the error curve, plotted on the same graph. The robots load capacity, net of the additional wrist actuator (unused 'fan' axis) was approximately 20 kg. The additional load mass weighed 16.2 kg and corresponded to the 'loaded' condition.

	Axis 1 Swing	Axis 2 Vertical	Axis 3 Horizontal
Low Condition	-0.3927 rads -22.5 deg 1163 units	+1.4399 rads +82.5 deg 1535 units	+0.8 m +800 mm 1676 units
High Condition	+0.3927 rads +22.5 deg 2431 units	+1.7017 rads +97.5 deg 2559 units	+1.0 m 1000 mm 2751 units
Motion Range	0.7854 rads 45 deg 768 units (±120 deg max)	0.26181 rads 15 deg 1024 units (±30 deg max)	0.2 m 200 mm 1075 units (762 mm max)

Figure Number	Motion Duration	Load Condition	Tuning Status
6.15	2.0	unloaded	fully tuned
6.16	2.0	loaded	fully tuned
6.17	1.5	unloaded	fully tuned
6.18	1.5	loaded	fully tuned
6.19	1.5	unloaded	tuned 6 cycles only
6.20	1.5	loaded	tuned 6 cycles only
6.21	1.26	unloaded	fully tuned
6.22	1.26	loaded	fully tuned

Table 6.14 The Tuning Process - Results Table

*(Each cycle comprised 1 second dwell followed by a tuned 'rise' of the duration specified from the low to the high condition, another 1 second dwell, then an un-tuned 'fall' from the high condition back to the low condition, with a 0.5 second dwell at the end of the cycle).

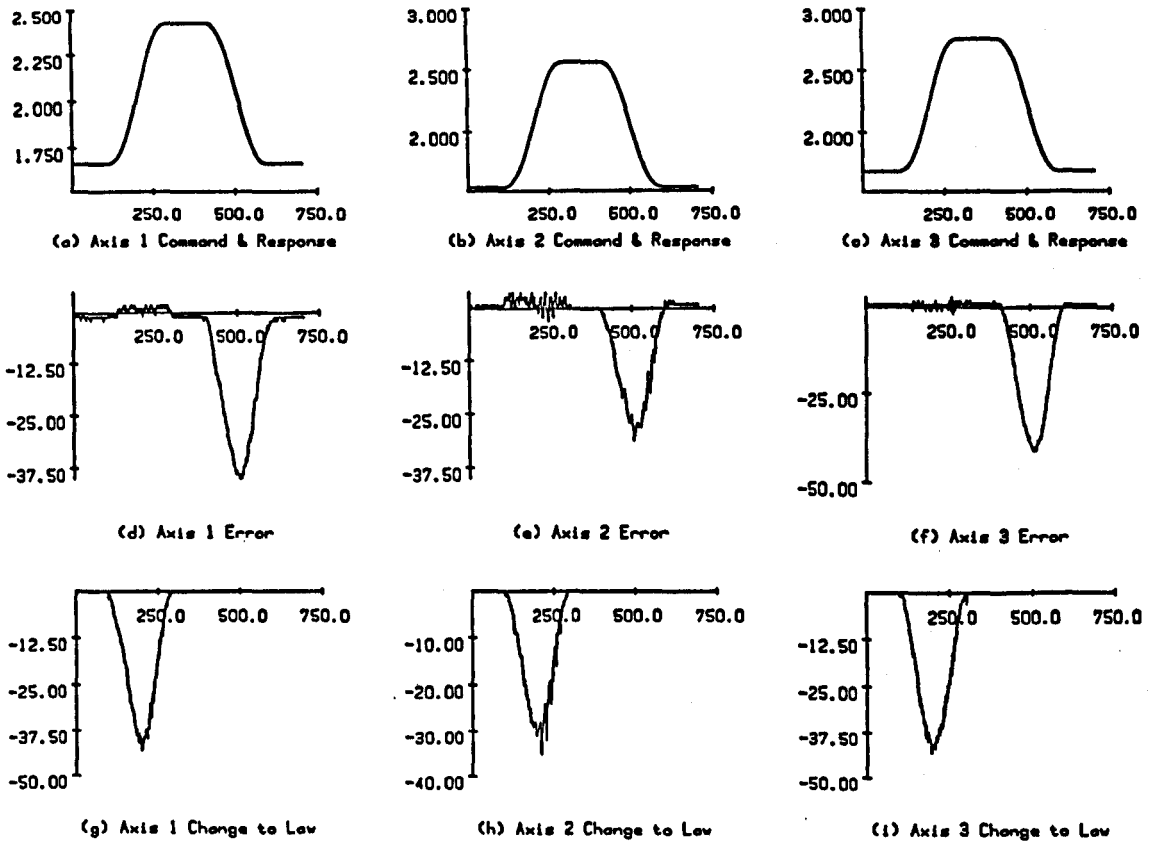


Figure 6.15 Fully Tuned Motion, 2 Second Rise, Unloaded

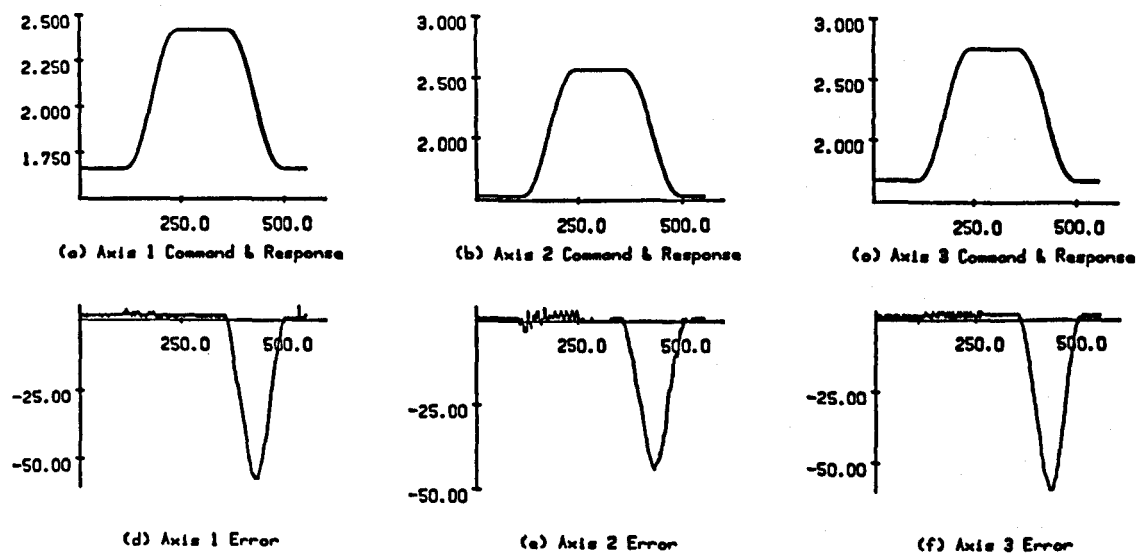


Figure 6.16 Fully Tuned Motion, 2 Second Rise, Loaded

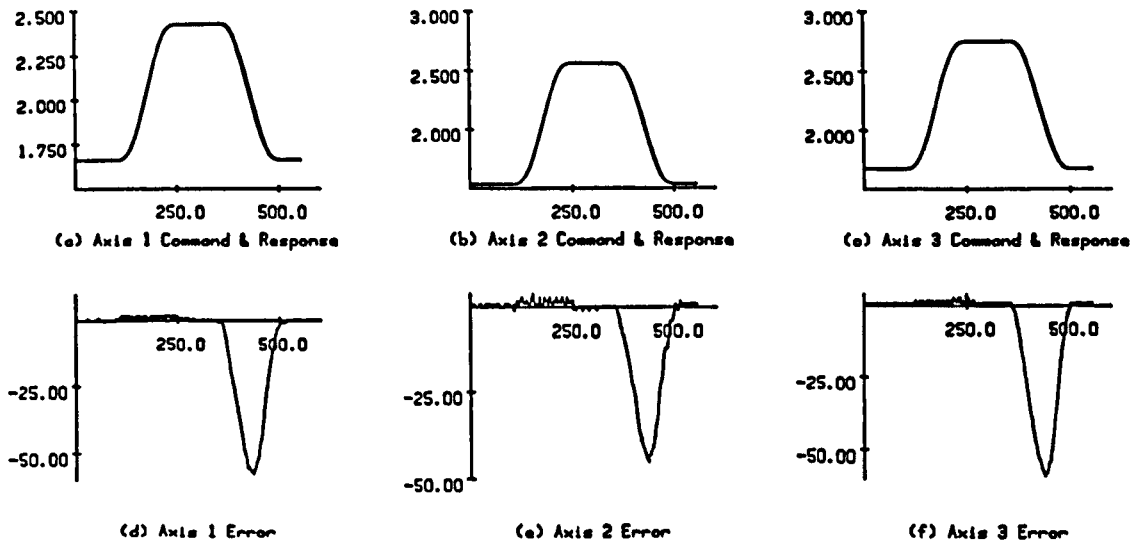


Figure 6.17 Fully Tuned Motion, 1.5 Second Rise, Unloaded

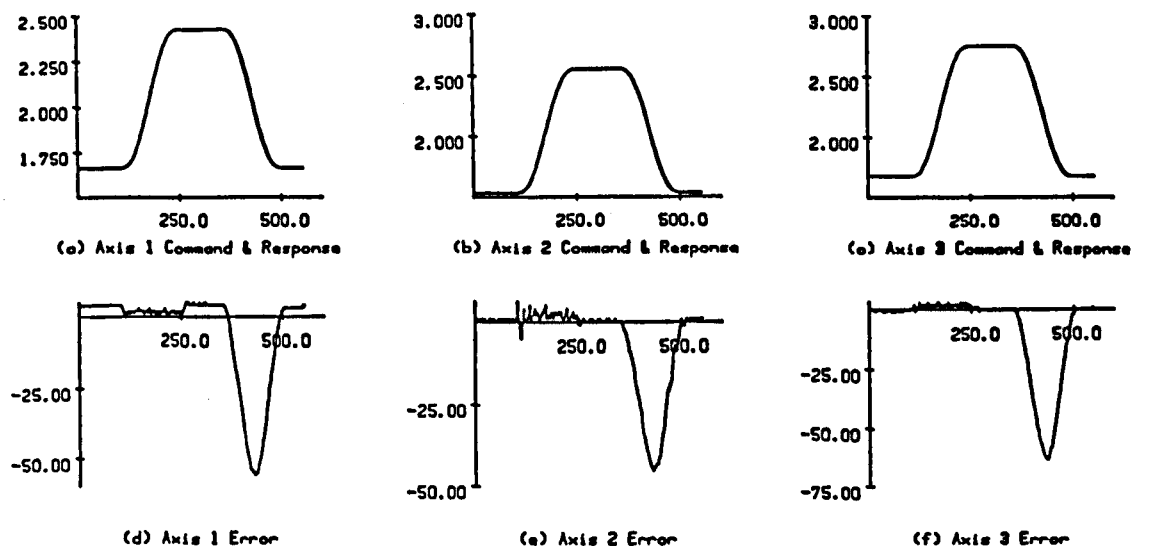


Figure 6.18 Fully Tuned Motion, 1.5 Second Rise, Loaded

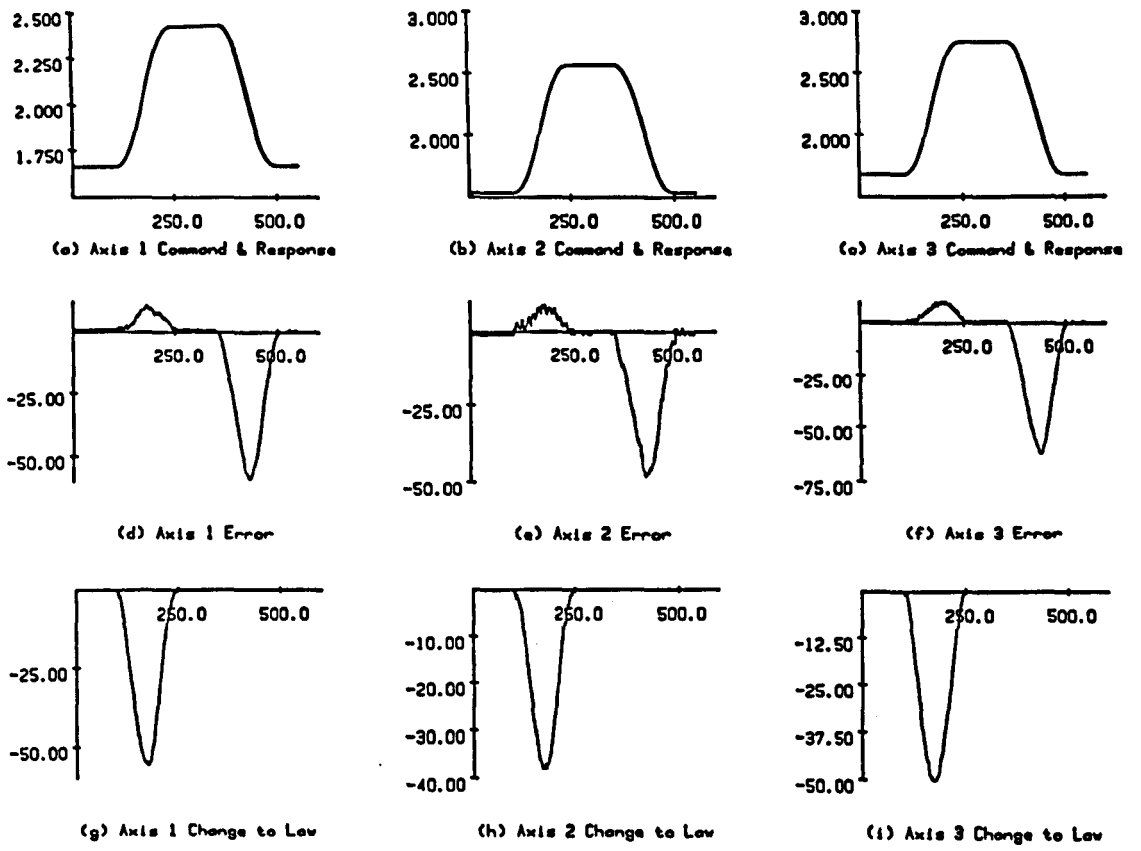


Figure 6.19 Tuned 6 Cycles, 1.5 Second Rise, Unloaded

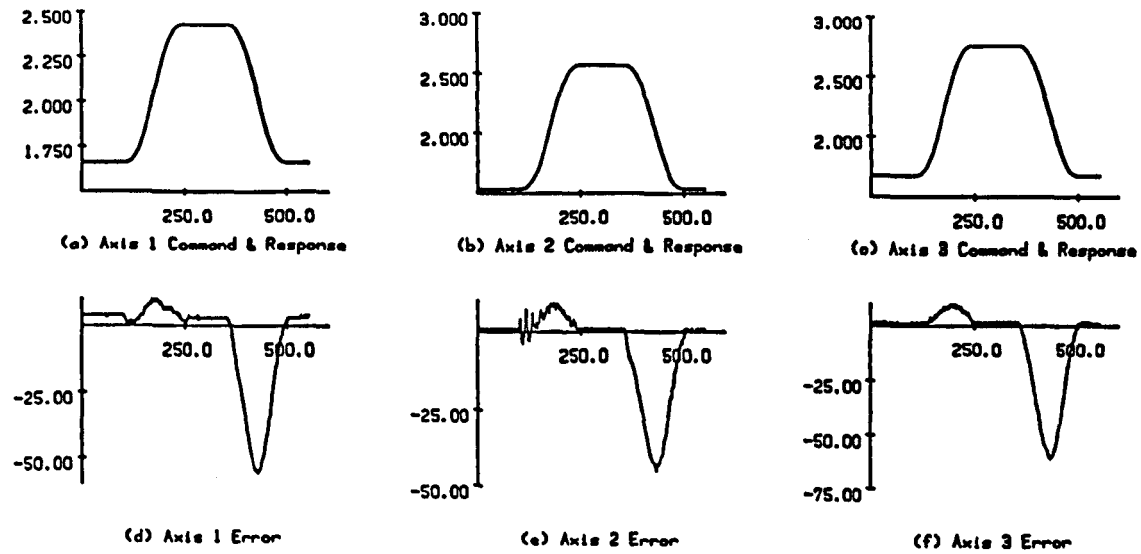


Figure 6.20 Tuned 6 Cycles, 1.5 Second Rise, Loaded

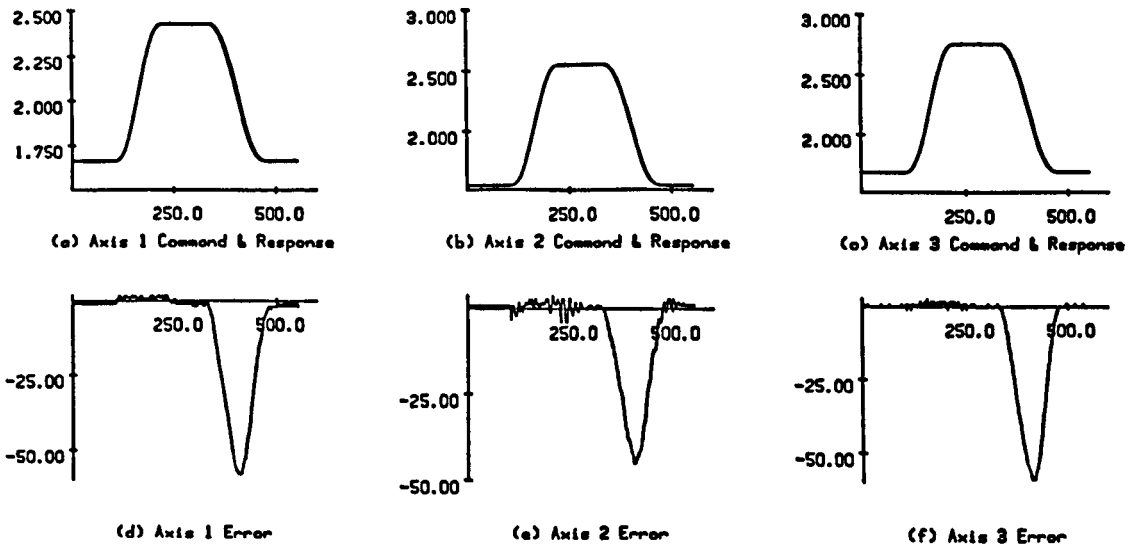


Figure 6.21 Fully Tuned, 1.26 Second Rise, Unloaded

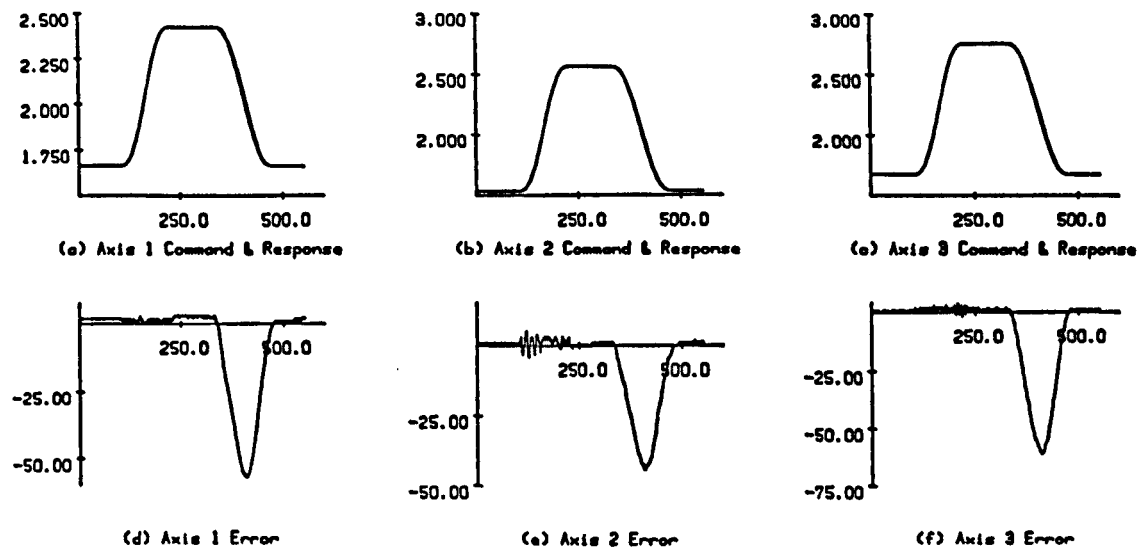


Figure 6.22 Fully Tuned, 1.26 Second Rise, Loaded

6.4 Industrial Robot Results

6.4.1 Observations on Results

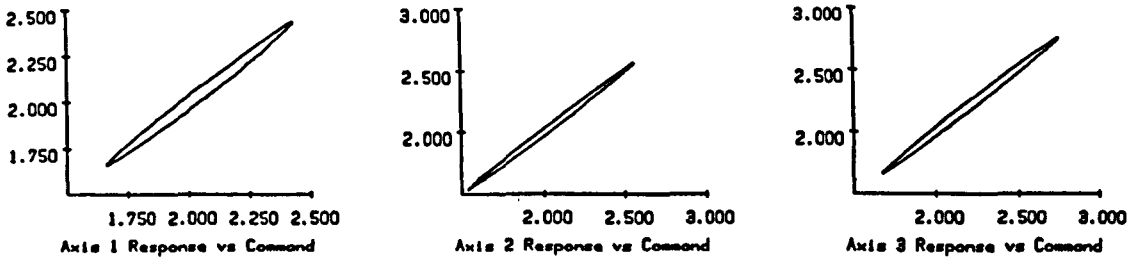
(i) Presence of the load requires more change to the command law to attain the same tolerance, but damps out the erratic oscillatory component in the three axis responses. Axis 2 is sensitive to the load, its oscillatory amplitude increasing with it.

(ii) This oscillatory component reduces as the motion duration decreases. The servo valve is known to display improved behaviour when operating away from the non-linearities which occur at or near the no flow condition. Non-linearities are more pronounced at low flows. There is also significant static friction in certain of the other mechanical elements. These effects are more pronounced in the lower speed motions, e.g. the 3 second motion shown in figure 8.4.

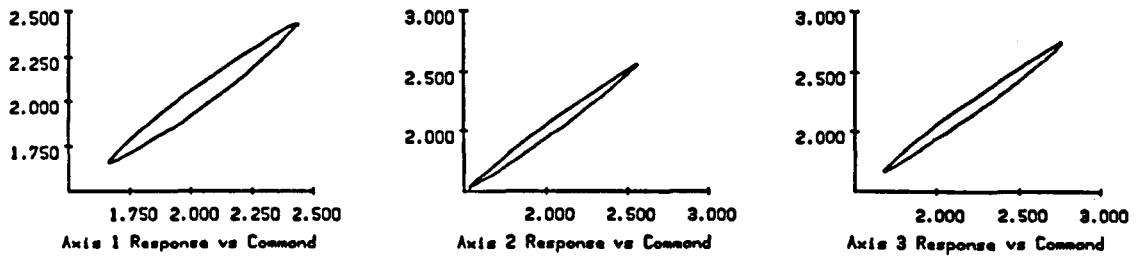
(iii) Axes 1 and 3 are particularly well behaved. Tuned dynamic error magnitudes in several instances are lower than the static errors. Again, this will only occur if system repeatability is significantly greater than system accuracy.

To confirm the validity of the approximation of the $\underset{\sim}{L}$ matrix to an identity matrix, figure 6.23 shows the axis responses plotted against the axis command motions. These are derived from the responses of figures 6.15 and 6.22.

Two further sets of results are of interest. An intermittent fault in the potentiometer gear drive appeared on axis 3. At first sight it was thought to emanate from a sticking servo-valve spool. The resultant spike in the response was still progressively reduced by the tuning process. A violent impulse was generated in the tuned command, in order to oppose it, figure 6.24. Reasonable tracking was achieved, but the jolt imparted to the axis was unhealthy. Later, a small brass gear driving the axis feedback potentiometer was identified as intermittently slipping on its hub. After re-rivetting, the fault disappeared.



(a) Unloaded, 2 Second Rise



(b) Loaded, 1.26 Second Rise

Figure 6.23 Axis Responses (vertical) Plotted Against Axis Commands (horizontal)

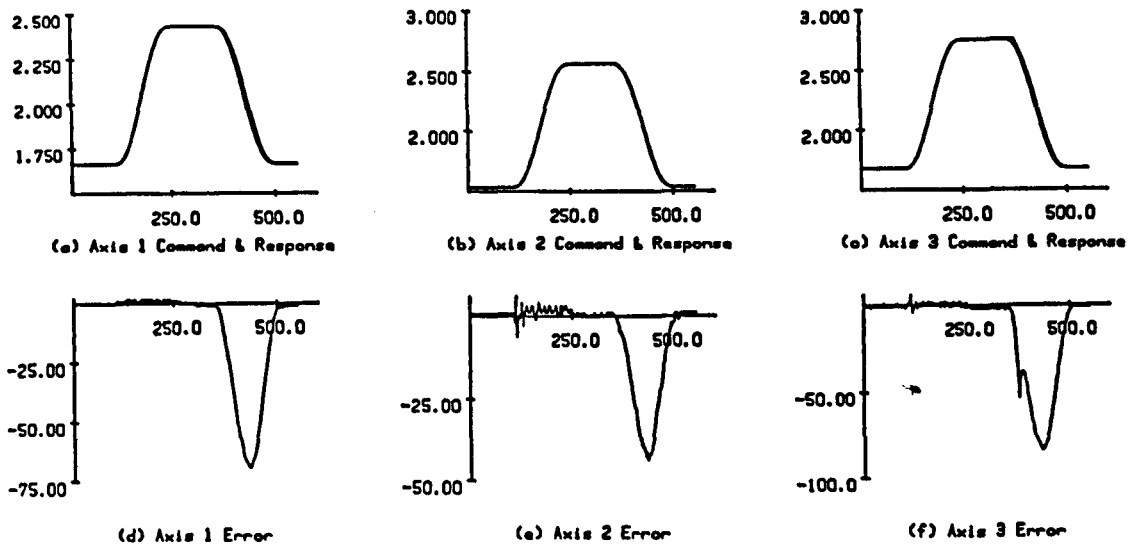


Figure 6.24 Results with 'Spike' in Axis 3 Response

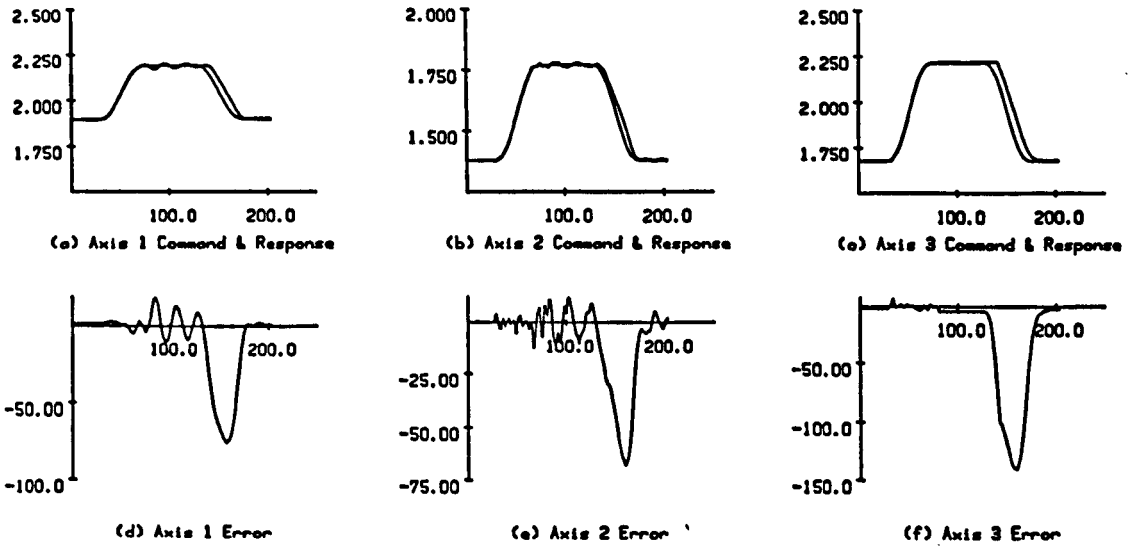


Figure 6.25 Results with Low Period Ratio

Results presented in figure 6.25 were obtained from a low period ratio motion (~ 2), with a necessarily smaller displacement range than the main test motion. (See table 6.26 for details). The dominant natural frequency in the configuration chosen appears at around 4.5 Hz. Selecting the motion duration at 0.5 seconds should then excite some vibratory response. The tuning process was still able to cope to some extent, levelling out the error during the tuned segment. The segment following was untuned, resulting in the residual response actually being accentuated. The changes required to the reference law are so severe that, when tuning ceases, there is a large step change left in the command law. After a low period ratio segment, tuning (if applied) should be retained for some period, probably until either a high period ratio motion or a dwell is encountered. This feature could be accomplished automatically. Axis 2 has its own frequency (around 13.2 Hz in this configuration), superimposed onto the fundamental of 4.5 Hz. The 4.5 Hz coupling between axes 1 and 2 is probably by virtue of the common hydraulic supply, as much as the inertial coupling present.

	Axis 1 Swing		Axis 2 Vertical		Axis 3 Horizontal	
Low Condition	-0.15	rads	+1.4	rads	+0.8	m
	-8.6	deg	+80.2	deg	+800	mm
	1900	units	1379	units	1676	units
High Condition	+0.15	rads	+1.5	rads	+0.9	m
	+8.6	deg	+85.9	deg	900	mm
	2194	units	1770	units	2214	units
Motion Range	0.3	rads	0.1	rads	0.1	m
	17.2	deg	5.7	deg	100	mm
	294	units	391	units	538	units
	(±120 deg max)		(±30 deg max)		(762 mm max)	

Table 6.26 Low Period Ratio Motion - Test Conditions

*(Each cycle comprised a ½ second dwell followed by a tuned 'rise' of ½ duration from the low to the high condition, a ½ second dwell, then an un-tuned 'fall' from the high condition back to the low condition of ½ second duration, with a ½ second dwell at the end of the cycle).

6.4.2 Practical Problems Encountered

(i) Relative drift between the potentiometer power supply and the ADC reference voltage. This caused the robot to appear to shift its datum, increasing the number of cycles needed to complete the tuning process. Constant re-calibration was therefore required. (In a commercially constructed version, this is overcome by using the fed back potentiometer supply voltage as the ADC reference).

(ii) Variations in the robot oscillatory response probably because of the inconsistent hydraulic oil temperature. Ideally the hydraulic oil would be maintained at constant temperature to ease comparison of results. This is impractical to achieve because the robot duty cycle varied as did the ambient temperature. The design of the hydraulic power supply was such that it attempted to maintain a constant pressure, any excess flow being throttled across the relief valve. The motor therefore continually ran at around its full power of 5.5 kw. When the energy was not being used by the robot, it was dissipated into the oil. As oil temperature increases, its propensity to absorb air increases. The bulk modulus decreases, changing the apparent mechanical stiffness of the hydraulic actuator. Axis 2 particularly, became intermittently oscillatory. The amount of

free air in this axis could be estimated by moving the cylinder rod with power on. Free movement measured could be as high as 30 mm. Attempts at bleeding the cylinder were thwarted by its design.

(iii) It was impossible to retain the datum settings of the actuators and feedback transducers. Numerous components periodically worked loose. Potentiometers moved on their mountings, bolts fell off and jammed in other parts of the mechanism, the linear slide bearing clearances needed continual attention, one of the cylinder mounting brackets snapped etc... In summary, the 'Little Giant' had been sold probably before it was out of the normal prototype stage.

The work of this chapter is reported in Vernon, Rees Jones and Rooney, 1986 [6.2].

7 Model Based Tuning Using an MRACS Control Scheme

Tuning motion laws based on the robot dynamics has been shown to be extremely difficult. Another approach is to force the robot to behave as if its dynamics could be characterised adequately by very simple dynamic models. The motion laws would then be tuned, based on this much simpler dynamics. One controller which is claimed to be capable of achieving such a simplification is the Model Referenced Adaptive Controller (section 3.5).

7.1 A Model Referenced Adaptive Control Scheme

As discussed in section 3.5, because of the array of problems encountered in manipulator control, most of which are position variant, one is led by its very definition to an adaptive controller.

A self-adaptive control system can be defined as one which has the capability of changing its parameters through an internal process of measurement, evaluation and adjustment to adapt to a changing environment, either external or internal to the plant under control.

A disadvantage of these systems is that they tend to use complex algorithms. Computation delays can then degrade the performance by reducing the bandwidth. Landau 1974 [3.18] suggests two options; one in which parameters or gains are changed, and the other in which the adaptor signal provides an extra input to the plant. Both of these options can be shown to be equivalent. The chosen scheme is the Model Referenced Adaptive Controller with the latter, 'signal synthesis adaptation'. The four objectives considered in Landau's method are :

(i) Simple control laws.
(ii) Strong stability characteristics.
(iii) High speed of adaptation.
(iv) Systematic design method.

The controllers robustness enables it to cope with large parameter variations and disturbances. Manipulators are not considered in his work.

The manipulator system can be broken down into a linear and a non-linear part. The non-linear part comprises by far the larger section of the equations (section 3.4). In the method applied by Stoten 1980 [3.20] the non-linear functions of the state are assumed to be bounded and are neglected, being treated as disturbances in the remainder of the design process. Stoten also neglects Landau's model gain matrix. For the control schematic, see figure 3.4.

The adaptor control algorithms have been developed by Landau and for this scheme a proportional plus integral adaptation is used. The model following feedback and feedforward gains are derived using the Erzberger conditions for perfect model following, Erzberger 1968 [7.1]. Stability of the adaptor scheme is ensured using the Popov hyperstability criterion. This criterion is easy to use and is behind the structure of Landau's adaptor. Linear compensator stability is achieved using Lyapunov's second method.

These references are used as a basis for the design of a controller. The property of the controller which is particularly of interest, is its ability to characterise the dynamics of an otherwise complex plant in a simple fashion.

The model may (within limits) be chosen to exhibit the desired closed loop characteristics. Its form may be developed to satisfy the Erzberger conditions. These conditions are essentially related to the structure of the $\underset{\sim}{A}_m$, $\underset{\sim}{B}_m$, $\underset{\sim}{A}_p$, $\underset{\sim}{B}_p$ matrices and not the actual parameter values.

The scheme is now developed for the specific problem, and as seen in the schematic, breaks down into four sections :

(i) The plant - i.e. the manipulator and associated transducers.

(ii) The reference model.

(iii) The model follower comprising conventional feedback and feedforward gains.

(iv) The adaptor.

Significant advantages in the design are attained if a decentralised approach is adopted, because of the resultant simplicity of various matrices.

7.2 Development of the Controller

7.2.1 The Plant

From the rigid model equations, a state space formulation may be obtained giving :

$$\dot{\underset{\sim}{x}}_p = \underset{\sim}{A}_p \underset{\sim}{x}_p + \underset{\sim}{B}_p \underset{\sim}{u}_p + \underset{\sim}{F}(t) \qquad (7.1)$$

where $\underset{\sim}{x}_p(t)$ - the state vector

$\underset{\sim}{A}_p$ - system coefficient matrix

$\underset{\sim}{B}_p$ - system input coefficient matrix

$\underset{\sim}{u}_p$ - control input vector

$\underset{\sim}{F}(t)$ - vector of non-linear functions

The non-linear dynamic equations derived in section 3.4 are fully included in this representation. Only the constant inertial, viscous friction and scaling terms appear in the state space representation, all the rest are contained within the $\underset{\sim}{F}(t)$ matrix. Expanded out this takes the form :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix}_p = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & A_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & A_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & A_{66} \end{bmatrix}_p \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}_p + \begin{bmatrix} 0 & 0 & 0 \\ B_{21} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & B_{42} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & B_{63} \end{bmatrix}_p \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}_p + \begin{bmatrix} 0 \\ F_1 \\ 0 \\ F_2 \\ 0 \\ F_3 \end{bmatrix} \qquad (7.2)$$

Note

$$\underset{\sim}{x}_p = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ d_3 \\ \dot{d}_3 \end{bmatrix} \qquad (7.3)$$

If viscous friction coefficients are present, i.e. b_{11}, b_{12}, b_{13} and there is a scaling constant for each axis b_{01}, b_{02}, b_{03}, then the form of the $\underset{\sim}{A}$ and $\underset{\sim}{B}$ elements are :

$$A_{22} = -b_{11}/\bar{H}_{11} \qquad B_{21} = b_{01}/\bar{H}_{11}$$

$$A_{44} = -b_{12}/\bar{H}_{22} \qquad B_{42} = b_{02}/\bar{H}_{22}$$

$$A_{66} = -b_{13}/\bar{H}_{33} \qquad B_{63} = b_{03}/\bar{H}_{33} \qquad (7.4)$$

where these $\bar{H}_{ii}$ are the constant terms taken from the H_{ii} elements of equation 3.17. The non-linear functions in $\underset{\sim}{F}(t)$ are bounded, and of course highly significant. For the case of time varying plants, Landau shows that it is possible to guarantee that the plant to model state error remains bounded, when $\underset{\sim}{A}_p$ is time varying. It is therefore hypothesised that an adaptor can be designed which will completely negate this component. If this is possible, the remaining system components can be analysed for their model following properties. The $\underset{\sim}{F}(t)$ matrix is chosen to be considered as a disturbance, and from hereon neglected.

7.2.2 The Reference Model

Choosing the model to be comprised of linear second order differential equations and so incorporate properties of inertia, damping and stiffness :

$$\dot{\underset{\sim}{x}}_m = \underset{\sim}{A}_m \underset{\sim}{x}_m + \underset{\sim}{B}_m \underset{\sim}{u}_m \qquad (7.5)$$

or

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix}_m = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\sigma_1 & -\sigma_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\sigma_1 & -\sigma_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\sigma_1 & -\sigma_2 \end{bmatrix}_m \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}_m + \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}_m \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}_m \qquad (7.6)$$

This is a decoupled model, one for each of the three manipulator axis sub-systems. The same dynamics are required in each, for coordinated synchronous motion. Each of the sub-systems has two second order eigenvalues of $\underset{\sim}{A}_m$ at :

$$\frac{-\sigma_2 \pm \sqrt{\{\sigma_2^2 - 4\sigma_1\}}}{2} \qquad (7.7)$$

These are chosen to be non-oscillatory and fast. Critically damped values are therefore taken. The model form is chosen with hindsight, as this simplifies evaluation and satisfaction of the Erzberger perfect model following criteria, also solution of the Lyapunov equation. To some extent, it limits manipulator performance, unless the models change as a function of manipulator configuration.

7.2.3 The Model Follower Gains

Erzberger 1968 [7.1] derived conditions for perfect model following, which if satisfied are necessary but not sufficient to enable zero plant/model state error. From the MRACS design selected (figure 3.6) it is seen that :

$$\underset{\sim}{u}_{p1} = -\underset{\sim}{K}_p \underset{\sim}{x}_p + \underset{\sim}{K}_u \underset{\sim}{u}_m \qquad (7.8)$$

where $\underset{\sim}{K}_p$ - plant state feedback gain matrix

$\underset{\sim}{K}_u$ - input vector feedforward gain matrix

hence on imposing perfect model following :

$$\underset{\sim}{x}_m - \underset{\sim}{x}_p = \underset{\sim}{O} \qquad (7.9)$$

$$\dot{\underset{\sim}{x}}_m - \dot{\underset{\sim}{x}}_p = \underset{\sim}{O} \qquad (7.10)$$

Substitution of equations 7.8, 7.9 and 7.10 into the state equations for the model and plant gives :

$$\underset{\sim}{A}_m - \underset{\sim}{A}_p + \underset{\sim}{B}_p \underset{\sim}{K}_p = 0 \qquad (7.11)$$

and $$\underset{\sim}{B}_m - \underset{\sim}{B}_p \underset{\sim}{K}_u = \underset{\sim}{O} \qquad (7.12)$$

The Erzberger conditions are sufficient to enable equations 7.11 and 7.12 to be solved in terms of $\underset{\sim}{K}_u$ and $\underset{\sim}{K}_p$ and are :

$$(\underset{\sim}{I}_6 - \underset{\sim}{B}_p \underset{\sim}{B}_p^+)\underset{\sim}{B}_m = \underset{\sim}{O} \qquad (7.13)$$

$$(\underset{\sim}{I}_6 - \underset{\sim}{B}_p \underset{\sim}{B}_p^+)(\underset{\sim}{A}_m - \underset{\sim}{A}_p) = \underset{\sim}{O} \qquad (7.14)$$

$\underset{\sim}{B}_p$ is non square and so $\underset{\sim}{B}_p^+$ is the left-Penrose pseudo inverse of $\underset{\sim}{B}_p$. Winsor and Roy 1970 [7.2] show that the solutions are :

$$\underset{\sim}{K}_p = \underset{\sim}{B}_p^+ (\underset{\sim}{A}_p - \underset{\sim}{A}_m) \qquad (7.15)$$

and $$\underset{\sim}{K}_u = \underset{\sim}{B}_p^+ \underset{\sim}{B}_m \qquad (7.16)$$

In the chosen MRACS configuration, $K_m = 0$. Stoten 1980 [3.2] used this scheme for a 2 d.o.f. planar linkage, achieving good results in simulation. Obtaining the inverse :

$$\underset{\sim}{B}_p^+ = \begin{bmatrix} 0 & 1/B_{21} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/B_{42} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/B_{63} \end{bmatrix} \qquad (7.17)$$

Equations 7.9 and 7.10 are satisfied and so the scheme satisfies the necessary conditions for perfect model following.

Using the Winsor and Roy equations :

$$\underset{\sim}{K}_p = \begin{bmatrix} \sigma_1/B_{21} & (A_{22}+\sigma_2)/B_{21} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_1/B_{42} & (\sigma_2+A_{44})/B_{42} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_1/B_{63} & (\sigma_2+A_{66})/B_{63} \end{bmatrix} \qquad (7.18)$$

and

$$\underset{\sim}{K_u} = \begin{bmatrix} 1/B_{21} & 0 & 0 \\ 0 & 1/B_{42} & 0 \\ 0 & 0 & 1/B_{63} \end{bmatrix} \qquad (7.19)$$

The model follower gains are now specified.

7.2.4 The Adaptor

The adaptor should be designed to ensure strong stability characteristics and that the model following state error goes to zero :

i.e. $\lim_{t \to \infty} e(t) = 0 \qquad (7.20)$

Landau suggests the adaptor is decomposed into two parts :

(i) A linear time invariant part characterised by :

$$\underset{\sim}{v} = \underset{\sim}{D}\ \underset{\sim}{e} \qquad (7.21)$$

where $\underset{\sim}{v}$ - equivalent feedback system linear component (linear compensator signal)

$\underset{\sim}{D}$ - equivalent linear part gain matrix

$\underset{\sim}{e}$ - plant to model following error

(ii) A section generating $\delta\underset{\sim}{K_p}(t,e)$ and $\delta\underset{\sim}{K_u}(t,e)$, the gain changes required to nullify the errors due to disturbances and parameter variations. $\underset{\sim}{K_p}$ and $\underset{\sim}{K_u}$ are calculated using specific manipulator parameters.

Using proportional plus integral adaptation, the general form of the second part of the adaptor is given by :

$$\delta\underset{\sim}{K_p}(t,e) = \int_0^t \underset{\sim}{\Phi_1}(\underset{\sim}{v},\tau,t)d\tau + \underset{\sim}{\Phi_2}(\underset{\sim}{v},t) + \delta\underset{\sim}{K_p}(0) \qquad (7.22)$$

$$\delta\underset{\sim}{K_u}(t,e) = \int_0^t \underset{\sim}{\phi_1}(\underset{\sim}{v},\tau,t)d\tau + \underset{\sim}{\phi_2}(\underset{\sim}{v},t) + \delta\underset{\sim}{K_u}(0) \qquad (7.23)$$

where $\underset{\sim}{\Phi_1}$(mxn) and $\underset{\sim}{\phi_1}$(mxm) are non-linear time varying relations between $\delta\underset{\sim}{K_p}$ and $\delta\underset{\sim}{K_u}$ and the values of $\underset{\sim}{v}$ for $0 \leq \tau \leq t$.
$\underset{\sim}{\Phi_2}$(mxn) and $\underset{\sim}{\phi_2}$(mxm) are non-linear time varying relations between $\delta\underset{\sim}{K_p}$, $\delta\underset{\sim}{K_u}$ and the values of $\underset{\sim}{v}(t)$. These are transient terms which vanish for $\underset{\sim}{v} = 0$ or $\underset{\sim}{e} = 0$. The adaption signal becomes :

$$\underset{\sim}{u_{p2}}(t) = \delta\underset{\sim}{K_p}\ \underset{\sim}{x_p} + \delta\underset{\sim}{K_u}\ \underset{\sim}{u_m} \qquad (7.24)$$

7.2.5 Adaptor Non Linear Section

The condition for stability, given by Popov's hyperstability theorem is :

$$\int_0^{t1} \underset{\sim}{v}^T(t)\ \underset{\sim}{w}(t)\ dt \geq -\mu^2 \qquad \forall t1 \geq 0 \tag{7.25}$$

μ is a finite, positive constant. From the schematic

$$\underset{\sim}{v} = \underset{\sim}{D}\ \underset{\sim}{e} \tag{7.26}$$

and
$$\underset{\sim}{w} = [\delta\underset{\sim}{K}p(t,e) - \underset{\sim}{B}p^+(\underset{\sim}{A}m - \underset{\sim}{A}p) - \underset{\sim}{K}p]\ \underset{\sim}{x}p + [\delta\underset{\sim}{K}u(t,e) - \underset{\sim}{B}p^+\underset{\sim}{B}m + \underset{\sim}{K}u]\ \underset{\sim}{u}m \tag{7.27}$$

The adaptor form, based on Landau, is shown in figure 7.1. Landau constructs the non-linear section of the adaptor using the form of the Popov criterion as an aid :

$$\delta\underset{\sim}{K}p(t,\underset{\sim}{v}) = \int_0^t \bar{\underset{\sim}{\Gamma}}1\ \underset{\sim}{v}(\underset{\sim}{\Omega}1\ \underset{\sim}{x}p)^T d\tau + \underset{\sim}{\Gamma}1\ \underset{\sim}{v}(\underset{\sim}{\Omega}1\ \underset{\sim}{x}p)^T + \delta\underset{\sim}{K}p(0) \tag{7.28}$$

and

$$\delta\underset{\sim}{K}u(t,\underset{\sim}{v}) = \int_0^t \bar{\underset{\sim}{\Gamma}}2\ \underset{\sim}{v}(\underset{\sim}{\Omega}2\ \underset{\sim}{u}m)^T d\tau + \underset{\sim}{\Gamma}2\ \underset{\sim}{v}(\underset{\sim}{\Omega}2\ \underset{\sim}{u}m)^T + \delta\underset{\sim}{K}u(0) \tag{7.29}$$

$\bar{\underset{\sim}{\Gamma}}1, \underset{\sim}{\Omega}1, \bar{\underset{\sim}{\Gamma}}2, \underset{\sim}{\Omega}2$ are positive definite matrices. $\underset{\sim}{\Gamma}1, \underset{\sim}{\Gamma}2$ are positive or non-negative definite matrices. No derivation is presented for the values of the matrix elements, selection is apparently arbitrary within the above limits. The appropriate dimensions in this case are :

$\bar{\underset{\sim}{\Gamma}}1, \underset{\sim}{\Gamma}\bar{1}, \underset{\sim}{\Gamma}2, \underset{\sim}{\Gamma}2, \underset{\sim}{\Omega}2$ (3x3) and $\underset{\sim}{\Omega}1$ (6x6)

All six matrices can be chosen diagonal (for simplicity) according to the above conditions, Stoten suggests :

$$\bar{\underset{\sim}{\Gamma}}1 = \underset{\sim}{\Gamma}1 = \bar{\underset{\sim}{\Gamma}}2 = \underset{\sim}{\Gamma}2 = \underset{\sim}{\Omega}2 = \underset{\sim}{I}3 \text{; the identity matrix, and :}$$

$$\underset{\sim}{\Omega}1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \qquad \text{where } 0 < \mu < 1 \text{ hence } \mu \text{ reduces the velocity term weighting} \tag{7.30}$$

These gain matrices are found not to be arbitrary in practice, in this work, particularly if stability, robustness and zero error in the limit are to be attempted. Bundell 1985 [7.3] describes problems due to high frequency excitation and plant/model mismatch.

Another problem which becomes apparent with Landau's adaptor in a real implementation and has not been described elsewhere, is the dependence

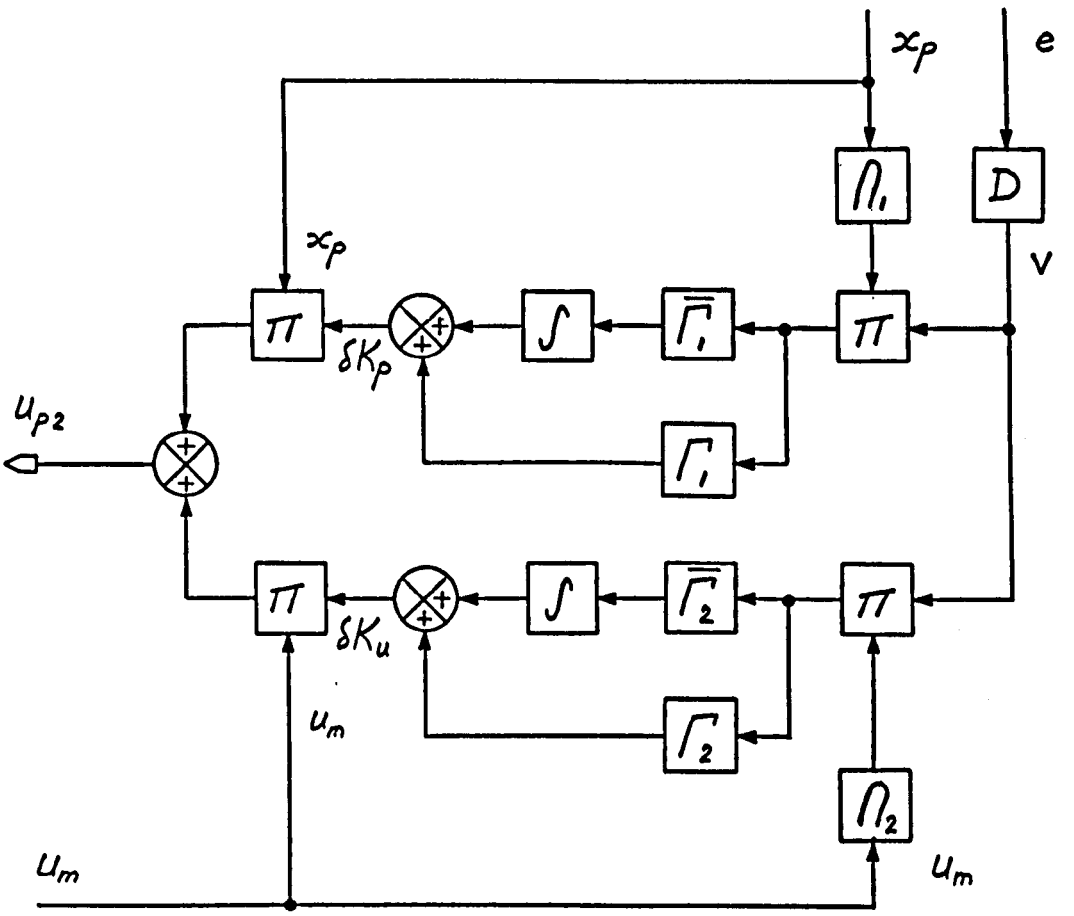


Figure 7.1 Proportional Plus Integral Adaptation

of these functions on the magnitudes of $\underset{\sim}{x}_p$ and $\underset{\sim}{u}_m$. It should be noted that gain changes are functions of these vectors. When operating closely to zero values for example, minimal gain changes result, giving very low stiffness. For this reason the response characteristics vary artificially depending on the region of the state that the axis is working. With Landau's structure it is doubtful whether this problem can be overcome.

7.2.6 Adaptor Linear Section

If the non-linear part satisfies the Popov integral inequality then :

$$\underset{\sim}{z}(s) = \underset{\sim}{D}(s\underset{\sim}{I} - \underset{\sim}{A}_m)^{-1}\underset{\sim}{B}_p \qquad (7.31)$$

where $\underset{\sim}{z}$ is the transfer matrix of the equivalent linear part and must

be strictly positive real. For stability to be assured :

$$\underset{\sim}{D} = \underset{\sim}{B_p}^T \underset{\sim}{P} \tag{7.32}$$

where $\underset{\sim}{P}$ is the positive definite symmetric matrix solution of the Lyapunov equation (second method) :

$$\underset{\sim}{A_m}\underset{\sim}{P} + \underset{\sim}{P}\underset{\sim}{A_m} = -\underset{\sim}{Q} \tag{7.33}$$

To simplify solution, $\underset{\sim}{Q}$ is chosen diagonal. It is positive definite, again of nominally arbitrary magnitude. The Q_{jj} terms weight the jth components of the model state error. Multiplying $\underset{\sim}{Q}$ by a scalar component has the effect of multiplying all the gains of the P+I amplifiers. The even numbered rows weight the velocities and the odd, the displacements.

The direct method for the solution of the Lyapunov equation may break down due to ill conditioning. Series methods are therefore put forward by Stoten, Jameson's method being used.

These solution methods are extremely tedious for state vectors of dimension six. Computing the values numerically is practical, but given the sparcity of $\underset{\sim}{A_m}$ and the fact that $\underset{\sim}{Q}$ is chosen diagonal, an algebraic solution can be obtained (see appendix A). This enables more insight into the precise selection of various components and aids the 'mechanisation' of the MRACS design process.

It is clear that the patterns present in the solution may be extrapolated to cover the 'n' axis case. Solutions for the $\underset{\sim}{P}$ matrix elements are :

$$\underset{\sim}{P}(6x6) = \begin{bmatrix} \bar{P}_1 & & 0 \\ & \bar{P}_3 & \\ 0 & & \bar{P}_5 \end{bmatrix} \tag{7.34}$$

where

$$\underset{\sim}{\bar{P}_i} = 1/2 \begin{bmatrix} (\sigma_2/\sigma_1 + 1/\sigma_2)Q_{ii} - Q_{i+1,i+1}/\sigma_2 & Q_{ii}/\sigma_1 \\ -Q_{ii} & (Q_{i+1,i+1} - Q_{ii})/\sigma_2 \end{bmatrix} \tag{7.35}$$

for i = 1,3 and 5

To be positive definite, all principal minors must be positive, hence :

$$\frac{Q_{i+1,i+1} - Q_{ii}}{\sigma_2} > 0 \tag{7.36}$$

and

$$(\sigma_2^2 + 2\sigma_1)Q_{ii}Q_{i+1,i+1} - \sigma_2^2 Q_{ii}^2 - \sigma_1 Q_{i+1,i+1}^2 > 0 \tag{7.37}$$

for $\underset{\sim}{Q}$ to be positive definite, the only condition is that :

$$Q_{ii} > 0 \quad \text{for } i = 1,2...6 \tag{7.38}$$

The linear compensator can now be computed as :

$$\underset{\sim}{D} = \begin{bmatrix} -B_{21}Q_{11}/2 & B_{21}(Q_{22}-Q_{11})/(2\sigma_2) & 0 & 0 & 0 & 0 \\ 0 & 0 & -B_{42}Q_{33})/2 & B_{42}(Q_{44}-Q_{33})/(2\sigma_2) & 0 & 0 \\ 0 & 0 & 0 & 0 & -B_{63}Q_{55}/2 & B_{63}(Q_{66}-Q_{55})/(2\sigma_2) \end{bmatrix} \tag{7.39}$$

Controller elements can now be stated in a more complete form. The conventional linear section component :

$$\underset{\sim}{u}_{p1} = \begin{bmatrix} (-\sigma_1 x_{1p} - (A_{22}+\sigma_2)x_{2p} + u_{1m})/B_{21} \\ (-\sigma_1 x_{3p} - (A_{44}+\sigma_2)x_{4p} + u_{2m})/B_{42} \\ (-\sigma_1 x_{5p} - (A_{66}+\sigma_2)x_{6p} + u_{3m})/B_{63} \end{bmatrix} \tag{7.40}$$

Because of practical system limitations, the conventional feedback section was implemented on an analog computer. A velocity signal was not available, so the differential pressure feedback was used in its place. This had to be heavily slugged by the filter to prevent noise problems. Considering the non-linear adaptor, the result for de-centralised control is :

$$\underset{\sim}{u}_{p2} =$$

$$\begin{bmatrix} x_{1p}\int_0^t v_1 x_{1p}\,d\tau + \beta x_{2p}\int_0^t v_1 x_{2p}\,d\tau + u_{1m}\int_0^t v_1 u_{1m}\,d\tau + v_1 x_{1p}^2 + v_1\beta x_{2p}^2 + v_1 u_{1m}^2 \\ x_{3p}\int_0^t v_2 x_{3p}\,d\tau + \beta x_{4p}\int_0^t v_2 x_{4p}\,d\tau + u_{2m}\int_0^t v_2 u_{1m}\,d\tau + v_2 x_{3p}^2 + v_2\beta x_{4p}^2 + v_2 u_{2m}^2 \\ x_{5p}\int_0^t v_3 x_{5p}\,d\tau + \beta x_{6p}\int_0^t v_3 x_{6p}\,d\tau + u_{3m}\int_0^t v_3 u_{3m}\,d\tau + v_3 x_{5p}^2 + v_3\beta x_{6p}^2 + v_3 u_{3m}^2 \end{bmatrix} \tag{7.41}$$

where the linear compensator gives :

$$\underset{\sim}{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -B_{21}Q_{11}(x_{1m}-x_{1p})/2 + B_{21}(Q_{22}-Q_{11})(x_{2m}-x_{2p}) \\ -B_{42}Q_{33}(x_{3m}-x_{3p})/2 + B_{42}(Q_{44}-Q_{33})(x_{4m}-x_{4p}) \\ -B_{63}Q_{55}(x_{5m}-x_{5p})/2 + B_{63}(Q_{66}-Q_{55})(x_{6m}-x_{6p}) \end{bmatrix} \tag{7.42}$$

Scaling terms will be required. In the steady state, it is a condition that :

$$\underset{\sim}{u}_{1p} = 0 \tag{7.43}$$

hence

$$\underset{\sim}{K}_p \underset{\sim}{x}_p = \underset{\sim}{K}_u \underset{\sim}{u}_m \tag{7.44}$$

this implies $\underset{\sim}{x}_{2p} = \underset{\sim}{x}_{4p} = \underset{\sim}{x}_{6p} = 0$

hence for scaling purposes :

$$\begin{aligned} -\sigma_1 x_{1p} &= u_{1m} \\ -\sigma_1 x_{3p} &= u_{2m} \\ -\sigma_1 x_{5p} &= u_{3m} \end{aligned} \tag{7.45}$$

7.3 MRACS Model Based Tuning of the Trajectory

The MRACS model to be used in the scheme comprises 'n' decoupled second order systems, effectively force driven (as opposed to motion driven). It is therefore a relatively straightforward case of that shown in section 5.4. The case can be treated with each degree of freedom separated. Each decoupled model is :

$$\ddot{q}_j + 2\zeta w_n \dot{q}_j + w_n^2 q_j = w_n^2 F_j \qquad (7.46)$$

and each has the same set of coefficients; ζ and w_n for coordinated motion. In this joint based trajectory case, 5th degree polynomials are selected. Boundary conditions comprise zero velocity and acceleration. The desired jth axis motions are defined by :

$$q_j(t) = \sum_{i=0}^{5} c_{ij} t^i \quad 0 \le t \le T \qquad (7.47)$$

where for each of the axes, (dropping the joint 'j' subscript for brevity) :

$$\begin{aligned} c_0 &= d_1 \\ c_1 &= c_2 = 0 \\ c_3 &= 10(d_2-d_1)/T^3 \\ c_4 &= -15(d_2-d_1)/T^4 \\ c_5 &= 6(d_2-d_1)/T^5 \end{aligned} \qquad (7.48)$$

given the axis model :

$$\ddot{q}(t) + 2\zeta w_n \dot{q}(t) + w_n^2 q(t) = w_n^2 r(t) \qquad (7.49)$$

If we let :

$$\begin{aligned} \sigma_1 &= 2\zeta w_n \\ \sigma_2 &= w_n^2 \end{aligned} \qquad (7.50)$$

and assuming the tuned motion is defined by :

$$r_j(t) = \sum_{i=0}^{5} b_i t^i \qquad (7.51)$$

then solutions can be found for the b_i as follows :

$$\begin{aligned} b_0 &= c_0 \\ b_1 &= 6c_3/\sigma_2 \\ b_2 &= (12c_4+3\sigma_1 c_3)/\sigma_2 \\ b_3 &= (20c_5 + 4\sigma_1 c_4+\sigma_2 c_3)/\sigma_2 \\ b_4 &= (5\sigma_1 c_5+\sigma_2 c_4)/\sigma_2 \\ b_5 &= c_5 \end{aligned} \qquad (7.52)$$

In continuous state space form, each axis sub-system is :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\sigma_2 & -\sigma_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\sigma_2 \end{bmatrix} \qquad (7.53)$$

where $x_1 = q$ and $x_2 = \dot{q}$, giving the discrete solution :

$$\underset{\sim}{x}_{i+1} = \underset{\sim\sim}{P}\underset{\sim}{x}_i + \underset{\sim\sim}{Q}\underset{\sim}{r}_i \qquad (7.54)$$

which for the critical or overdamped case with sample interval δt (section 3.3) :

$$\underset{\sim}{P} = \frac{1}{\sigma_2 - \sigma_1} \begin{bmatrix} \sigma_2 \epsilon^{-\sigma_1 \delta t} - \sigma_1 \epsilon^{-\sigma_2 \delta t} & \epsilon^{-\sigma_1 \delta t} - \epsilon^{-\sigma_1 \delta t} \\ -\sigma_1 \sigma_2 (\epsilon^{-\sigma_1 \delta t} - \epsilon^{-\sigma_2 \delta t}) & \sigma_2 \epsilon^{-\sigma_2 \delta t} - \sigma_1 \epsilon^{-\sigma_1 \delta t} \end{bmatrix} \qquad (7.55)$$

and

$$\underset{\sim}{Q} = \frac{1}{\sigma_2 - \sigma_1} \begin{bmatrix} \sigma_2 (1 - \epsilon^{-\sigma_1 \delta t}) - \sigma_1 (1 - \epsilon^{-\sigma_2 \delta t}) \\ \sigma_1 \sigma_2 (\epsilon^{-\sigma_1 \delta t} - \epsilon^{-\sigma_2 \delta t}) \end{bmatrix} \qquad (7.56)$$

For any specific case, all the matrix elements are constant. The model response computation for each axis reduces to six multiplications and four additions, (plus the scaling to 12 bit integers). Note that the algebraic solution for the bj coefficients will only function for the specific case of fifth degree joint based polynomials. With the above discrete solution (equation 7.54), a tuned law can be obtained for any motion. There is an extra 'cost' in that the motion velocity must be available to compute the tuned law. There may also be a small drift due to an accumulation of computational errors in the tuned law.

7.4 MRACS Implementation

It can be shown that a perfect response can be attained, given a motion law tuned to the exact dynamics of the system being driven. The proposed MRACS should though be capable of making the robot dynamics appear as if they approximated to three de-coupled second order systems, with pre-specified and hence known characteristics. If this can be achieved in practice then near perfect responses should be possible, using much simpler dynamic models.

7.4.1 Single Axis Tests

As a starting point, an application to a single degree of freedom electric motor and load was tested. The motor used was that described in figure 6.9. The controller was implemented in full on an EAL580 analog computer (circuit shown in figure 7.2). The objective here was simply to demonstrate the ability of the controller to force a system to behave in a specified manner. Model parameters were set to give :

(i) An oscillatory system with 0.5 damping coefficient and the circular natural frequency $w_n = 1/\sqrt{2}$, figure 7.3

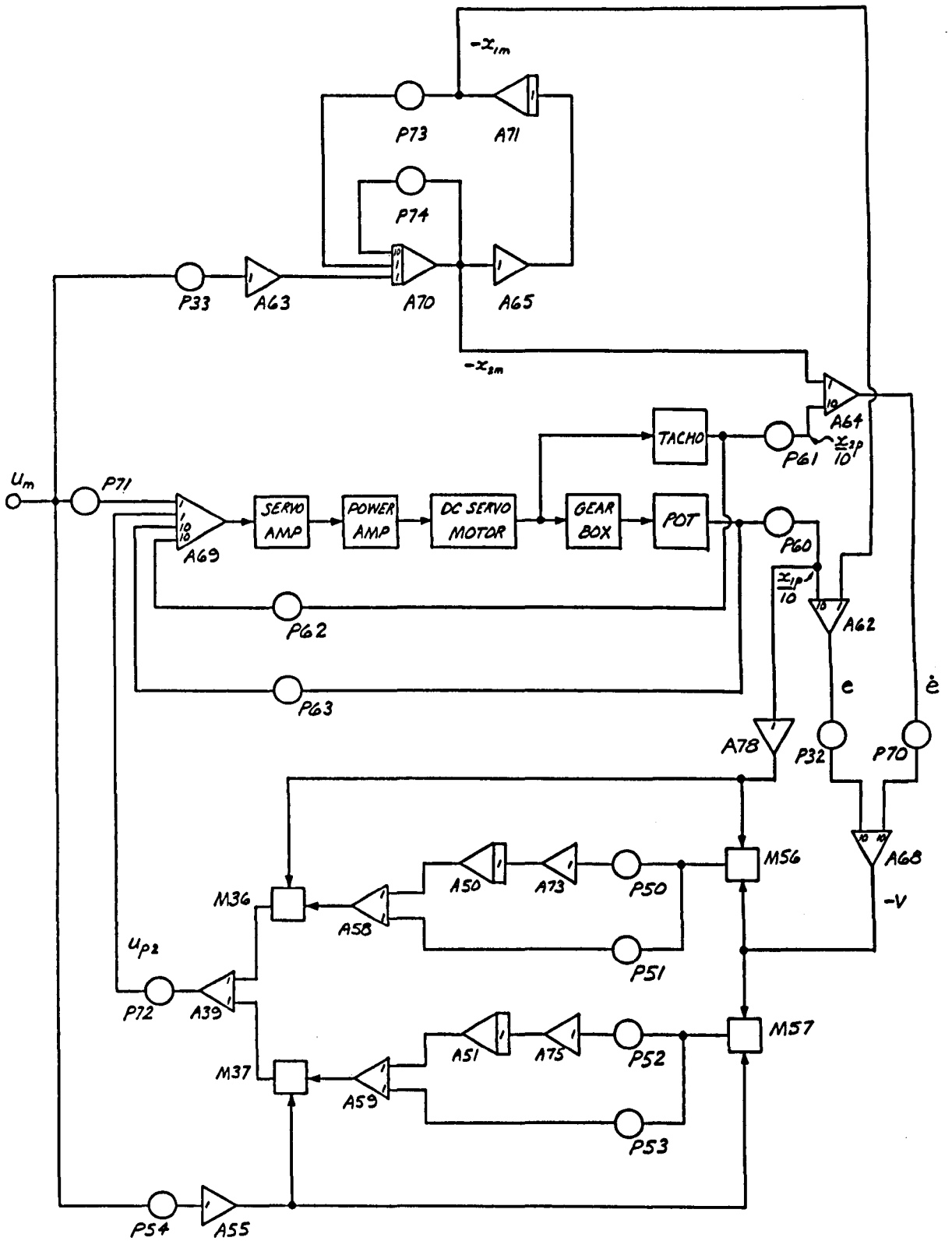


Figure 7.2 Single Axis Analog Based MRACS Circuit

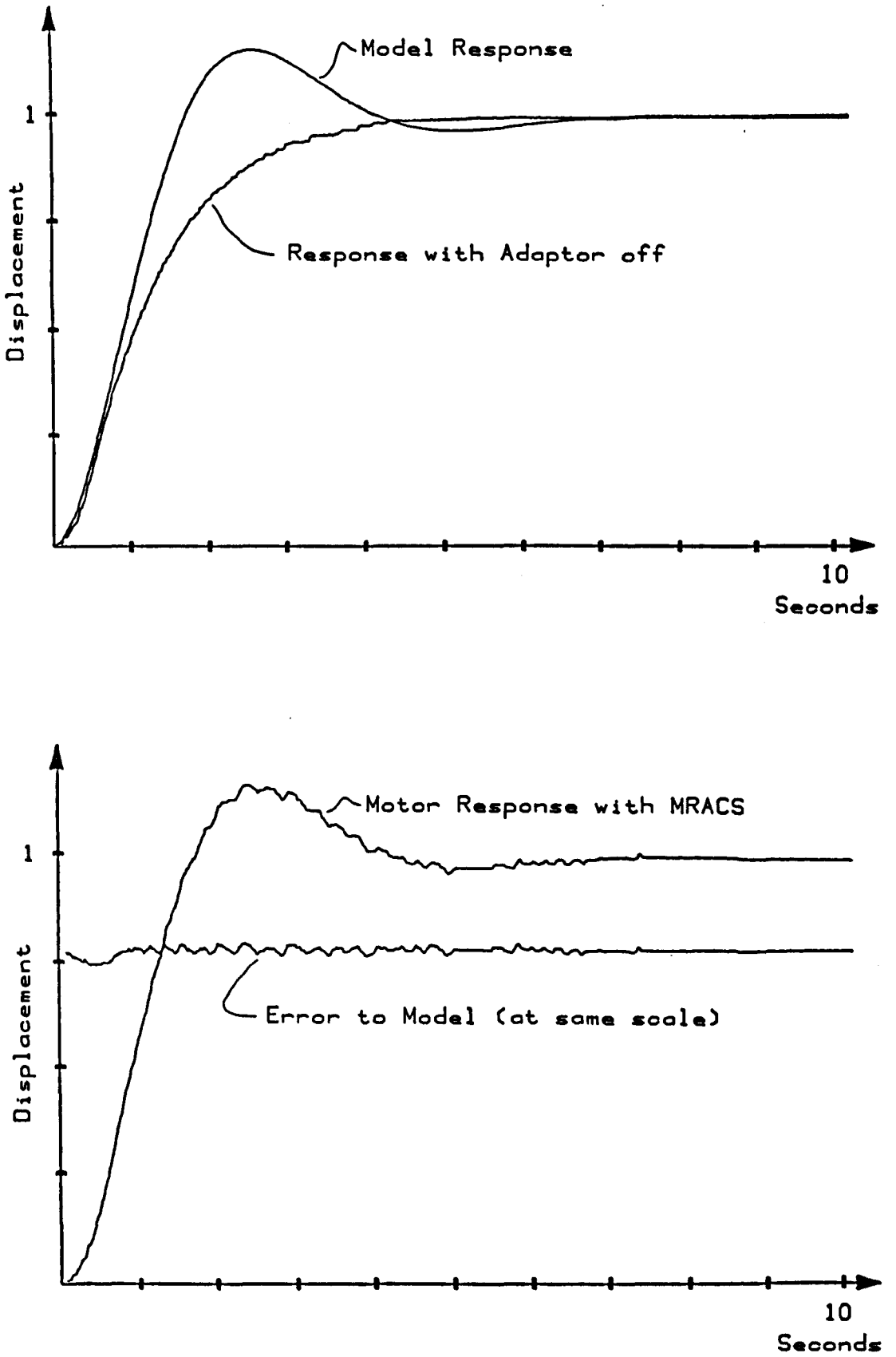
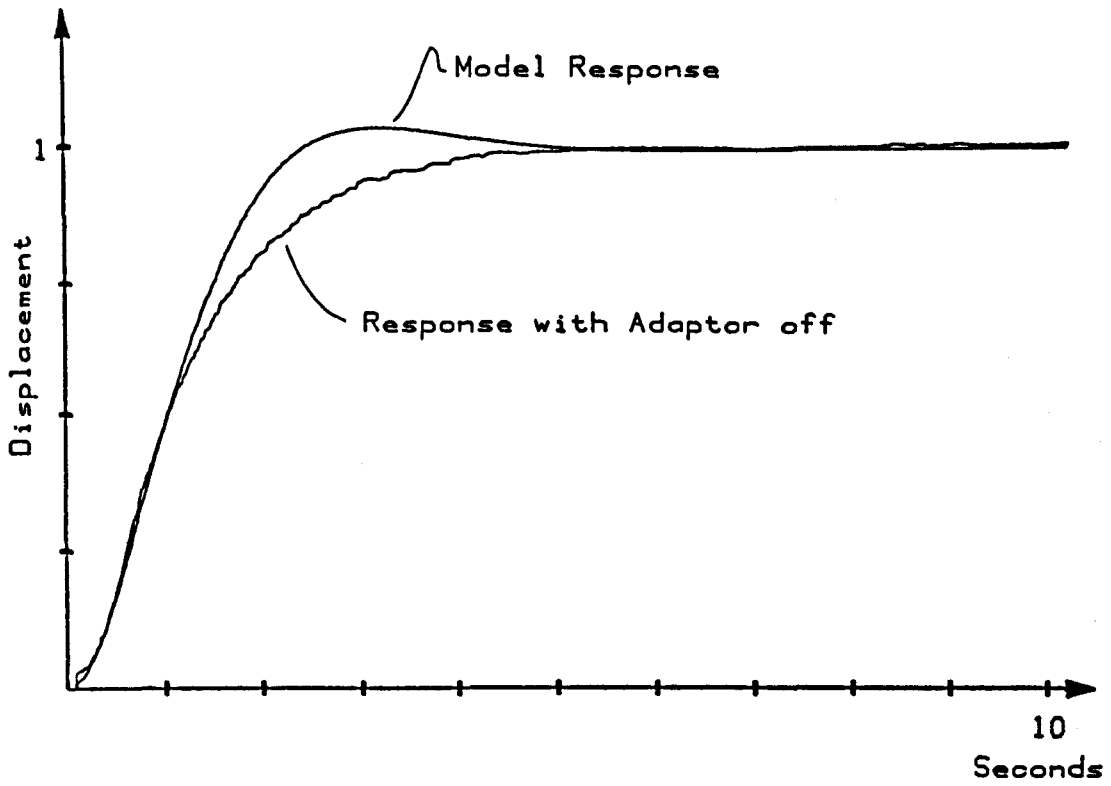


Figure 7.3 Model Following : Damping Coefficient 0.5



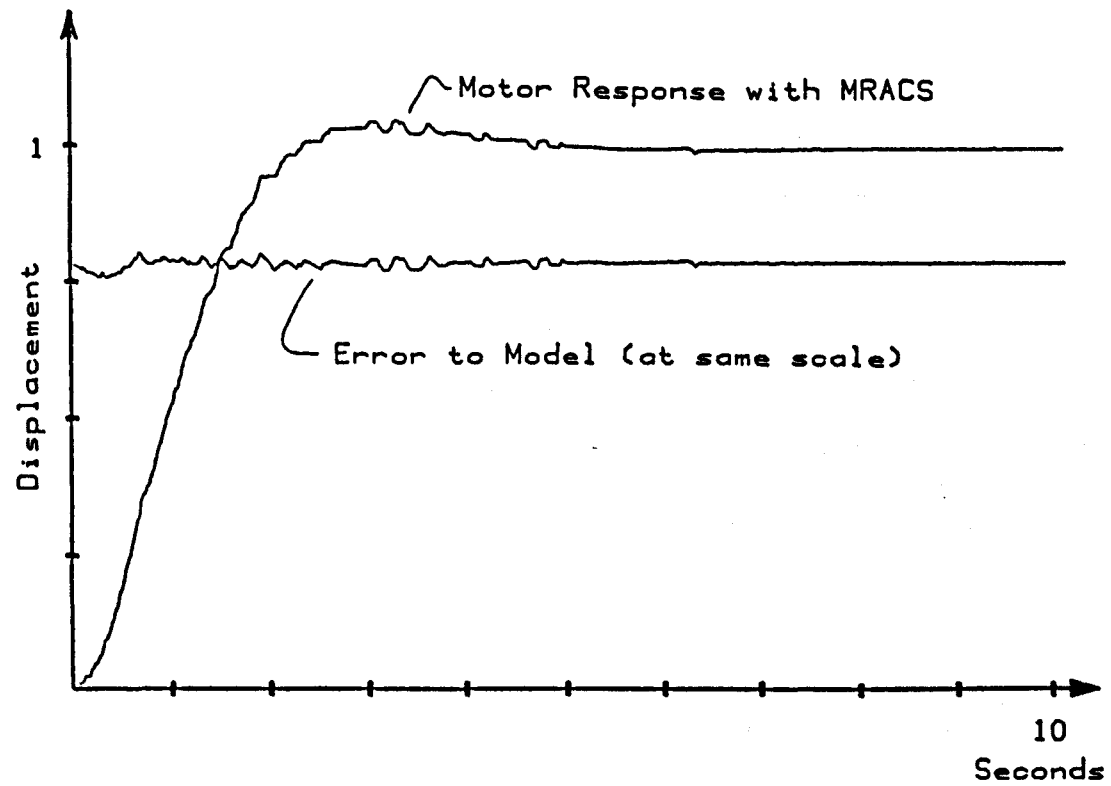


Figure 7.4 Model Following : Damping Coefficient 0.7071

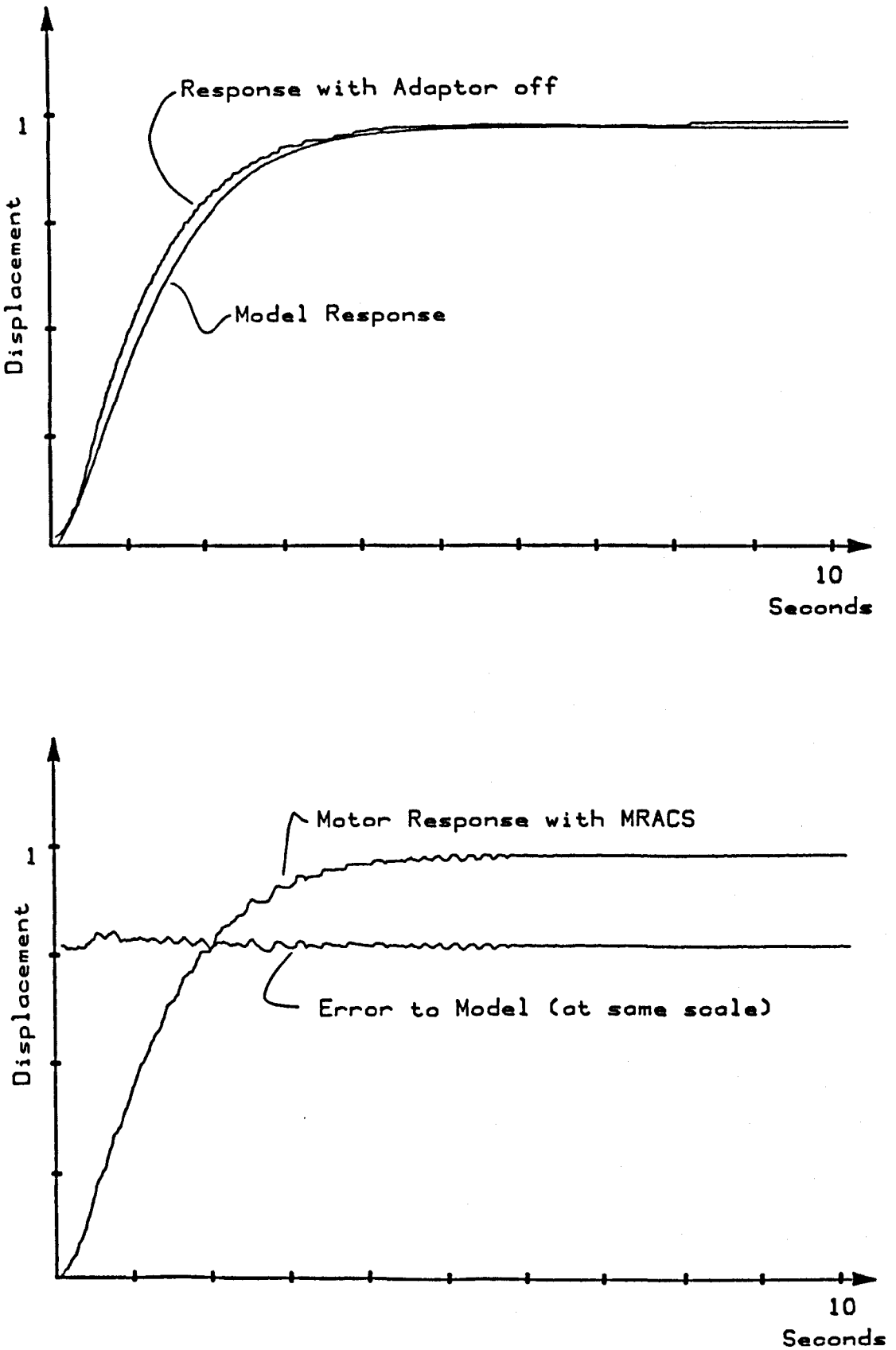


Figure 7.5 Model Following : Critical Damping

(ii) An oscillatory system with $1/\sqrt{2}$ damping and $w_n = 1/\sqrt{2}$, figure 7.4.

(iii) Critical damping with a settling time to 4τ of 1 second (close to the servo-motor systems natural 'dynamics', figure 7.5

Step response results for the three cases are presented.
As already explained in section 6.3, the motor unit possessed a large amount of backlash and Coulomb friction. The potentiometer used was also wirewound and coarse in resolution. It would appear that even with these difficulties, the controller was able to give the required result.

7.4.2 Industrial Robot Implementation

Moving on to the robot system, based on the 'Little Giant' as used earlier, some preliminary work was needed before implementation. The complete MRACS controller derived in section 7.2 is outlined diagrammatically in figure 7.6. The controller as derived has been scaled to the actual system and discretised using simple transforms. Velocity data is also needed. Whereas in chapter 8, velocity estimates can be computed off-line (permitting use of the more accurate central differences formula), here velocity must be estimated in real time. The method used therefore is a backward differences formula.

For any feedback controller to work well, its dynamics must be significantly faster than those of the system being controlled. If the single axis analog results are indicative, high frequency components (up to 750 Hz) are used within the control signal. These may be derived from the backlash, coupled to the adaptor's ability to change gains at very high rates. In the robot system, the oscillatory dynamics can probably be characterised well, as being within the 1 to 30 Hz region, hence at ten times this 300 Hz sampling would probably be a little faster than necessary. The computational hardware has a basic communications overhead constraint of 147 Hz which, less the computation, eventually ended up at around 67 Hz. It is clear that this will reduce the performance and/or stability of the controller.

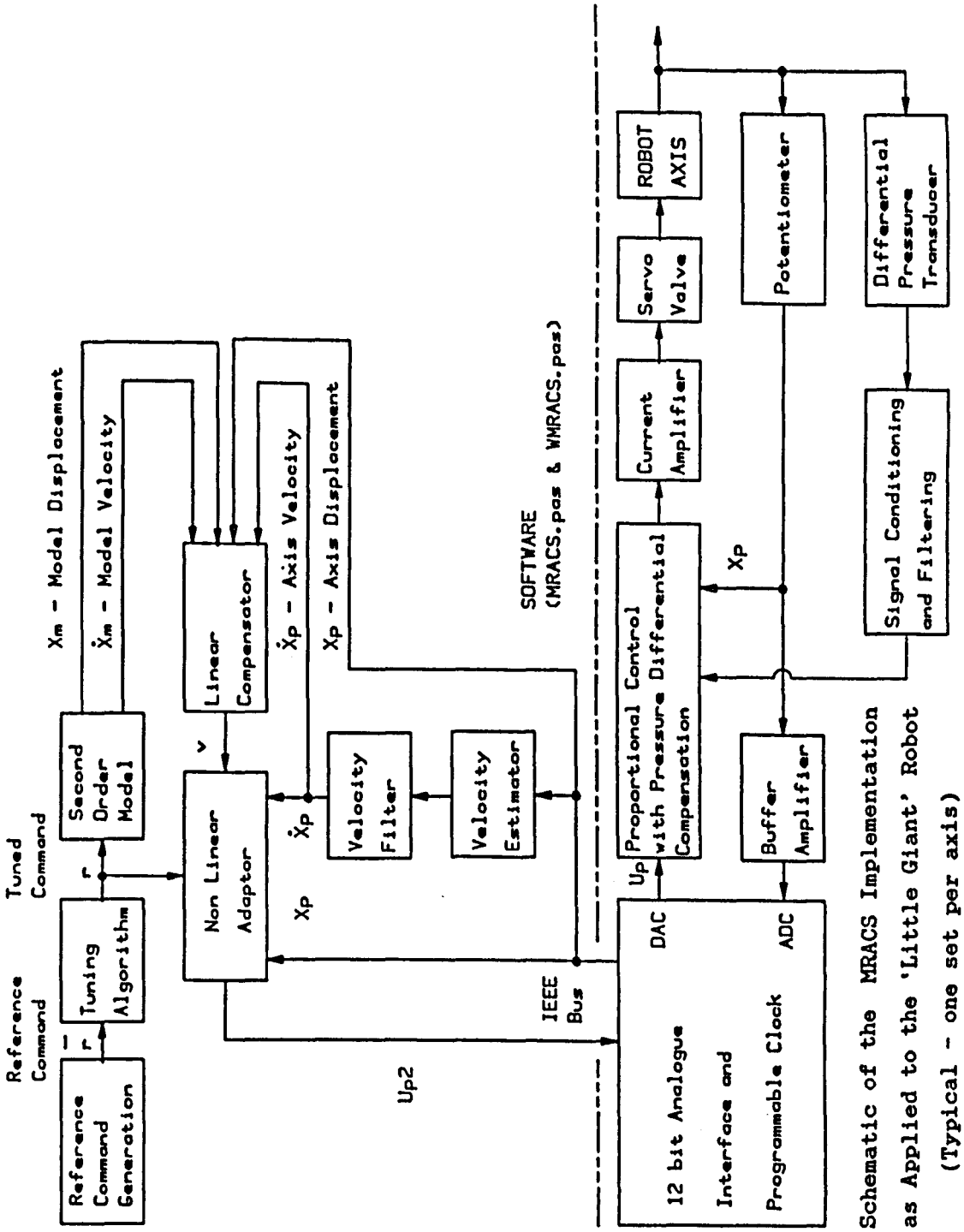


Figure 7.6 Schematic of the MRACS Implementation as Applied to the 'Little Giant' Robot (Typical - one set per axis)

7.4.3 Software for the Tuned MRACS System

Referring to the last section of the program MRACS.PAS (appendix A), the velocity is computed using fourth order backward differences :

$$\dot{y}_i \approx \frac{1}{6\,\delta t}\{\, 5y_i - 6y_{i-1} + 3y_{i-2} - 2y_{i-3} \,\} \qquad (7.57)$$

where i is the sample number.

which is derived from the differentiated Newton-Gregory backward differences formula, with all the appropriate difference terms substituted and cancelled. It is computationally more efficient to retain the velocity term as an integer and scaling is required anyway, so the 6 δt term can be omitted. This is further taken advantage of by precomputing a suitably scaled velocity command. Discretisation and signal noise demand that a filter be added to the velocity estimate. It comprises more or less a single discrete integrator with lag. The only remaining active or dynamic terms are the integrators used in the non-linear adaptor. The zero order hold equivalence principle tells us that the conversion of the continuous time integrator to the discretised version is given by :

$$G(z) = \left[\frac{z-1}{z} \right] \mathcal{Z} \left[\frac{1}{s^2} \right] = \left[\frac{z-1}{z} \right] \frac{\delta t . z}{(z-1)^2} = \frac{\delta t}{z-1} \qquad (7.58)$$

which over the interval [O,t] is equivalent to :

$$\int_0^t f(t)\, dt = \int_0^t f(i.\delta t)\, dt \sim \sum_{i=0}^{i=t/\delta t} f(i.\delta t)\, \delta t \qquad (7.59)$$

for δt small. Expressed as a computational algorithm :

$$intf[i] = intf[i-1] + f[i-1]*\delta t \qquad (7.60)$$

which is computed cyclically within the control loop, given :

$$intf[O] = O; \quad t = i*\delta t; \; f = f(t); \; intf = \int_0^t f(t)\, dt \qquad (7.61)$$

which is only accurate if the change in f(t) within one sample interval is small. From equation 7.42, the linear compensator for each axis can be reduced to :

$$v_j = K_{2j}(x_{jm}-x_{jp}) + K_{1j}(\dot{x}_{jm}-\dot{x}_{jp}) \qquad (7.62)$$

where K_{1j} and K_{2j} must be positive constants. In order to weight the displacement tracking accuracy (over the velocity tracking), the K_{2j} term should be larger than K_{1j}.

The non-linear adaptor in the diagram (figure 7.1) is also reduced in form. The combination of gains into the overall factors f_1, f_2, f_3 and f_4 helps to reduce the amount of real time control computation (figure 7.6). This 'lumping together' is permissible because several of the gain terms are of nominal magnitude. The motion tuning is carried out as already described, giving a solution for the model input as :

$$u_{1m}{}_{tuned} = (u_{1m}(i+1) - P_{11}u_{1m}(i) - P_{12}\dot{u}_{1m}(i))/Q_{11} \qquad (7.63)$$

Note that to evaluate this expression, the command velocity must be available.

7.4.4 Joint Based Motions with Dynamic Tuning

Various system parameters were established including suitable model time constants, a sampling interval, gains and filter constants. Typical values are shown in the MRACSDAT data file, table 7.7. The initial nominal motion used was that of section 6.3.4, enabling ease of comparison. The flowchart for the program is shown in figure 7.8. With zero velocity error gain, the system was found to be hopelessly unstable. Figure 7.9 shows reasonable tracking for a short period, then a rapidly increasing error term towards the end of the first motion segment. The sudden cessation in response is due to the power being cut off for safety reasons. Note the units of time along the horizontal axis are in sample intervals of 15ms, hence the nominal motion contains 524 samples.

0	debug - controls quantity of debugging information
0.015	delta - sample interval
1E-5	$K_1[1]$ - velocity error gain, axis 1
5E-4	$K_2[1]$ - displacement error gain, axis 1
1E-5	$K_1[2]$ - velocity error gain, axis 2
5E-4	$K_2[2]$ - displacement error gain, axis 2
1E-5	$K_1[3]$ - velocity error gain, axis 3
5E-4	$K_2[3]$ - displacement error gain, axis 3
1E-5	f_1 - grouped terms adaptor gain 1
1E-5	f_2 - grouped terms adaptor gain 2
5E-7	f_3 - grouped terms adaptor gain 3
1E-4	f_4 - grouped terms adaptor gain 4
2	solution type for tuning 1 - discrete 2 - continuous
48	σ_1 model parameter
576	σ_2 model parameter
1.68	duration of rise and fall, 2nd and 4th segments
1.5	dwell time, 1st and 3rd segments
1.5	dwell time, 5th and final segment
50	number of dummy samples executed before run
0.25	velocity filter decay constant

Table 7.7 MRACS Program - Variable Parameters for Operation

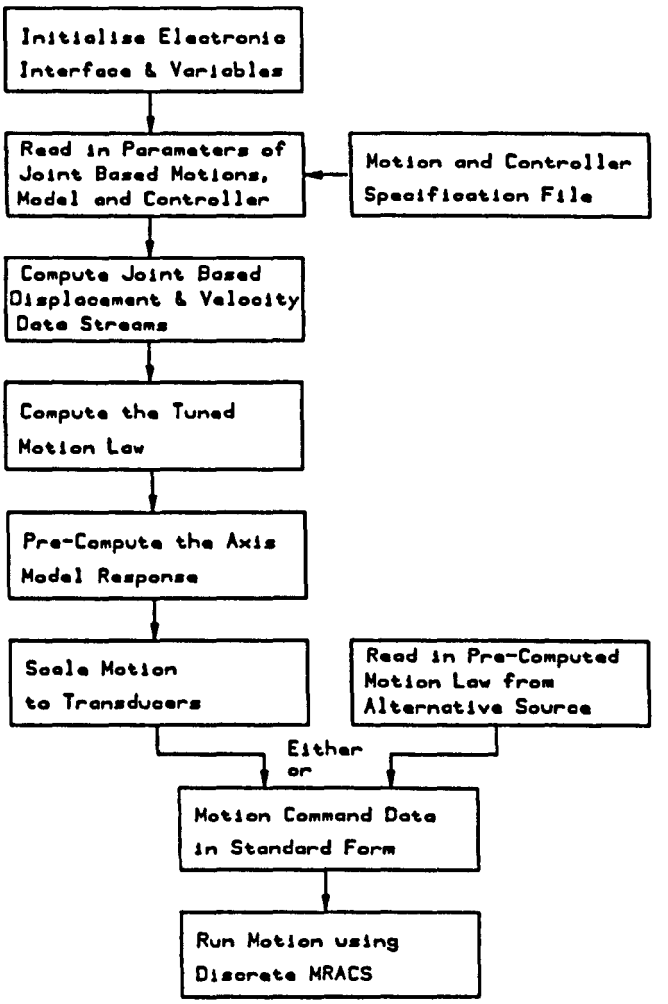


Figure 7.8 MRACS Program Flowchart

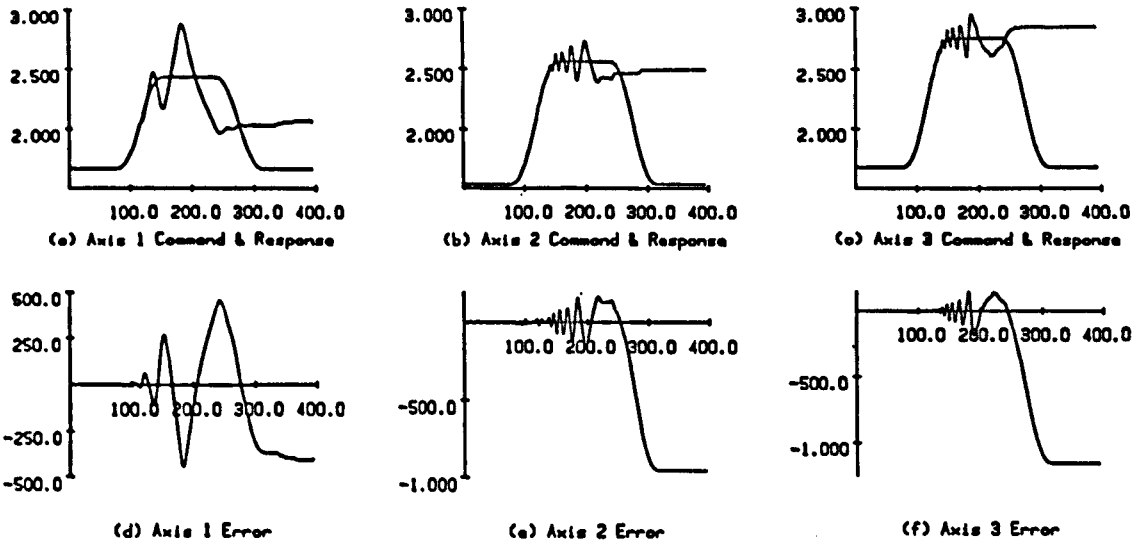


Figure 7.9 Gain Adjustment - Unstable Response

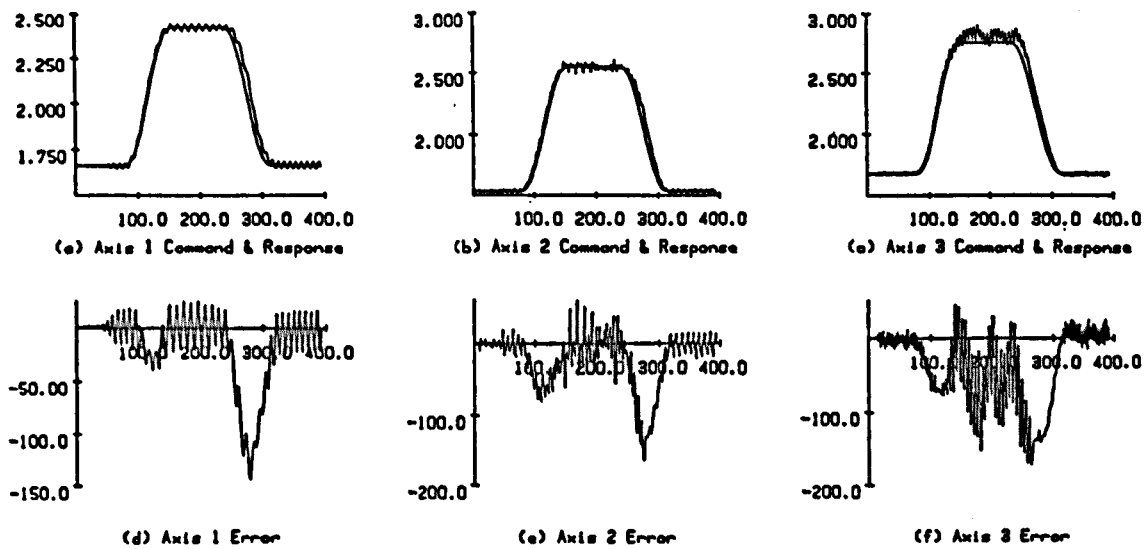


Figure 7.10 Gain Adjustment - Marginal Instability

Introducing the velocity error feedback term, (at 100 times its final value), resulted in a bounded hence marginal instability (figure 7.10). Progressive adjustment of the gains led to an acceptable set of values (in stability terms). Figure 7.11 shows a response to the nominal motion using the reference values of table 7.7.

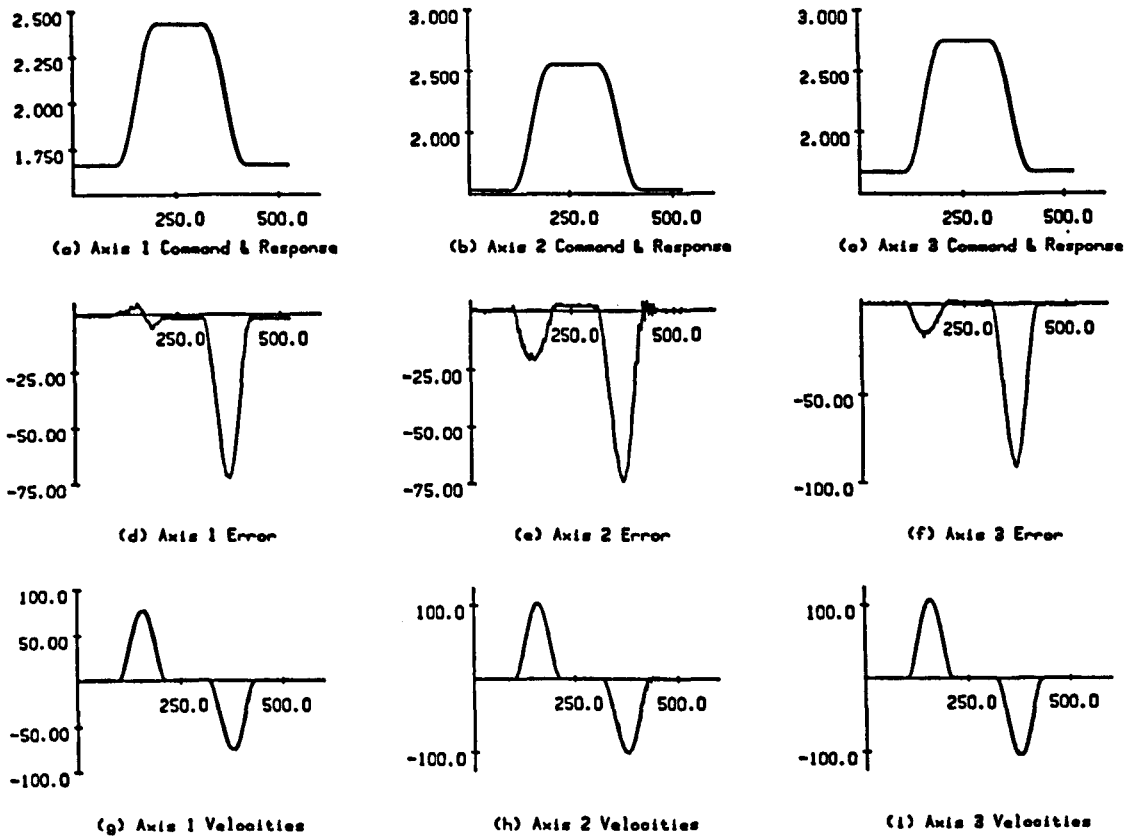


Figure 7.11 MRACS Gains - Reference Value Results

The first motion segment is the MRACS model based, tuned one, the second motion segment being un-tuned. The format is similar to that of section 6.4. Note that the response leads the command for axes 2 & 3. Axis error is part lag then lead for roughly equal durations. In all cases a significant reduction in error is achieved. Command and response velocities are also shown in figure 7.11. The shortest sample interval possible for the full controller was established as 14.8 milliseconds. (The communications overhead left a computation time available of 8.8 ms. In this time, some 60 multiplications, 50

additions, 40 assignments and various other operations are performed in both integer and floating point arithmetic in MS-Pascal).

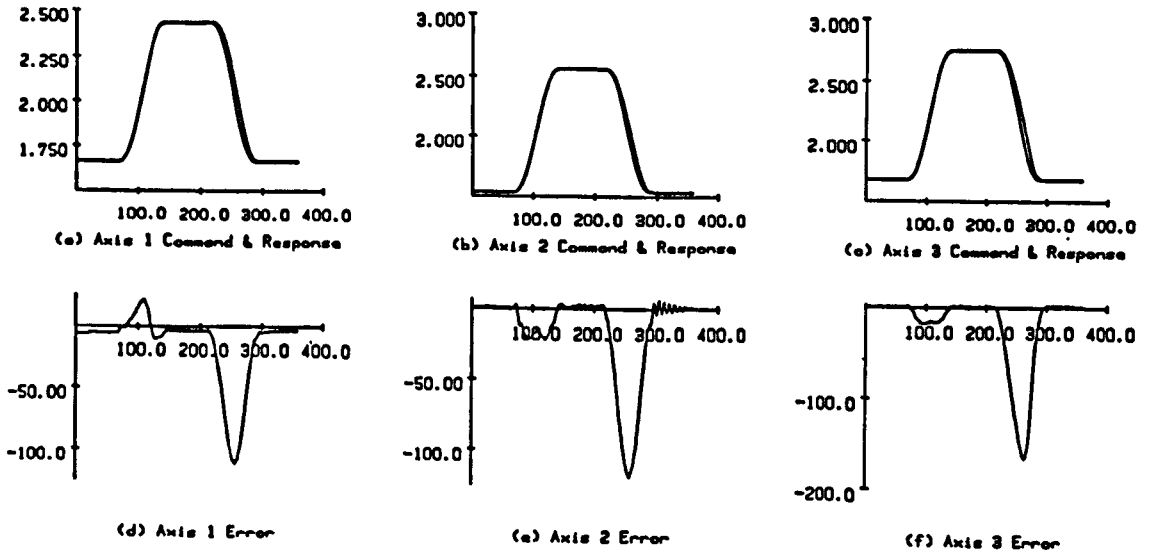


Figure 7.12 Reduced Motion Duration

Reducing the motion duration from 1.68 seconds to 1.2 seconds, causes a large increase in peak error of the un-tuned motion (figure 7.12), from 92 to 164 units. The tuned motion segment errors remain about the same at 21 units.

Returning to the original reference settings, but adding the 16.2kg load, resulted in responses as in figure 7.13. Compared to the reference results of figure 7.11, the changes in overall error magnitudes are small.

Disturbances consisting of a 1 Hz sine wave, 1v peak to peak, added into the displacement servo-amplifiers summing junction, in conjunction with the load were provided to each axis separately. In displacement, this corresponds to an enforced error of approximately 41 units peak to peak on each axis. Results indicate disturbance error values of 41,40 and 46 units for axes 1,2 and 3; figures 7.14 to 7.16 respectively. In hindsight, the disturbances should have been added elsewhere. The locations at which they were input simply confused the feedback displacement data.

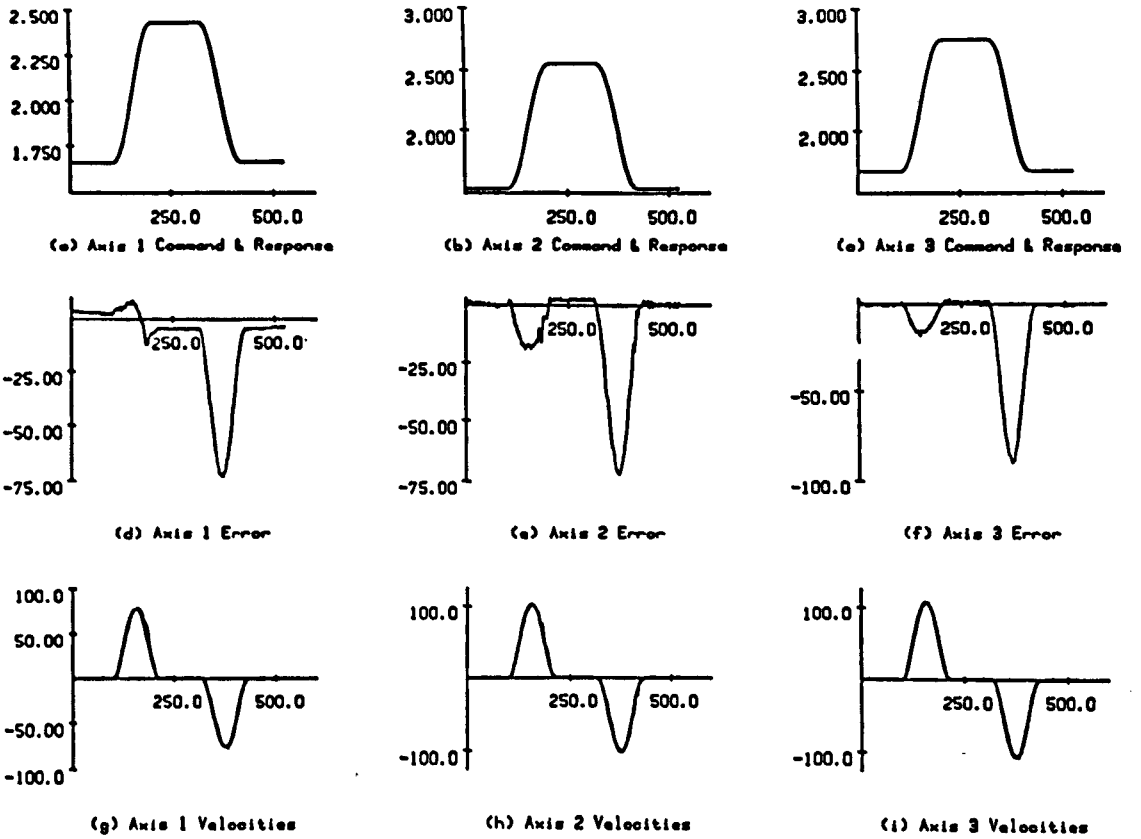


Figure 7.13 Increased Load

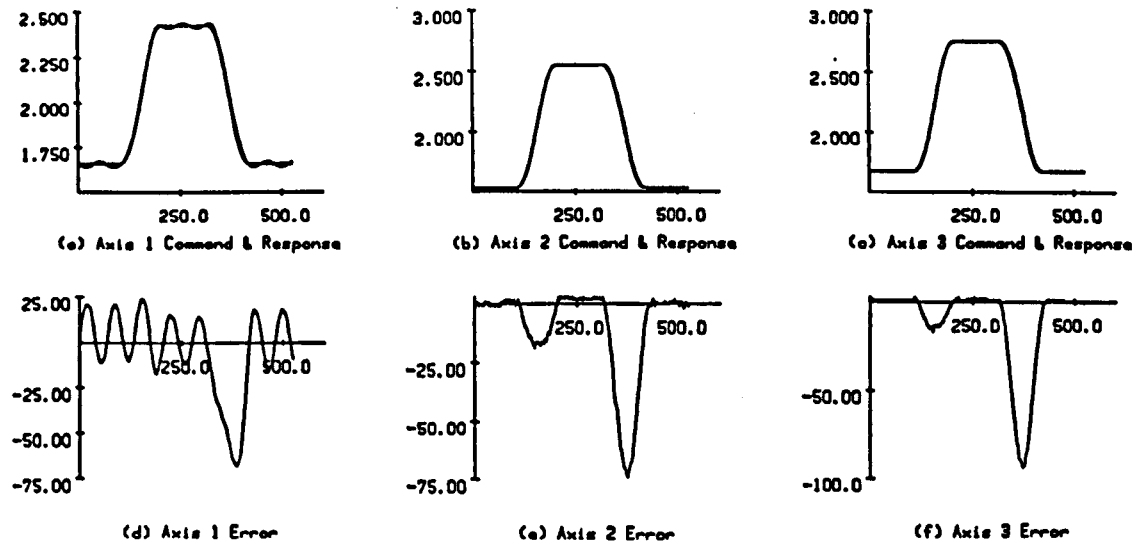


Figure 7.14 1 Hz Disturbance on Axis 1

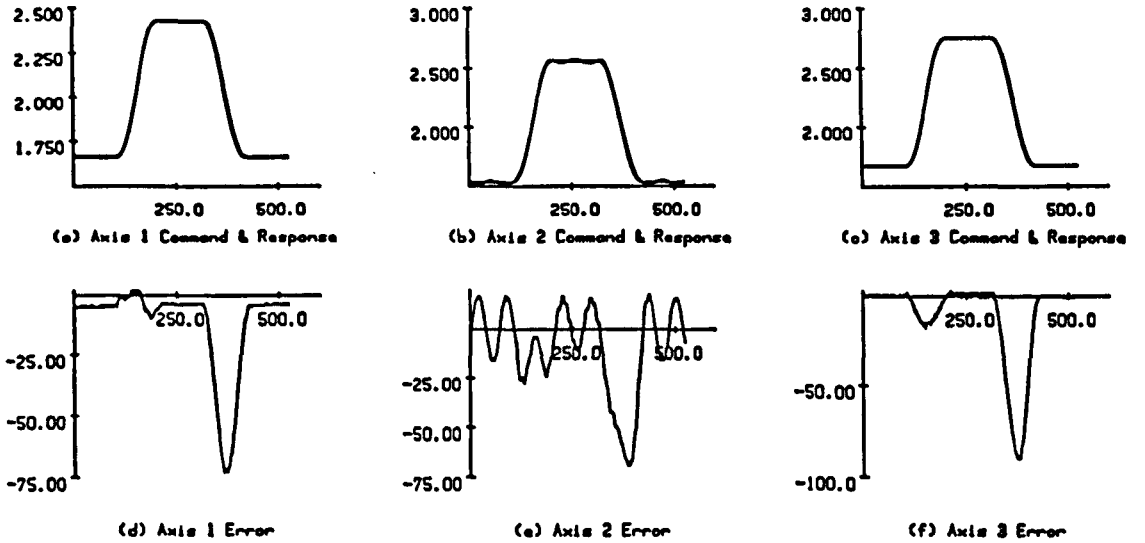


Figure 7.15 1 Hz Disturbance on Axis 2

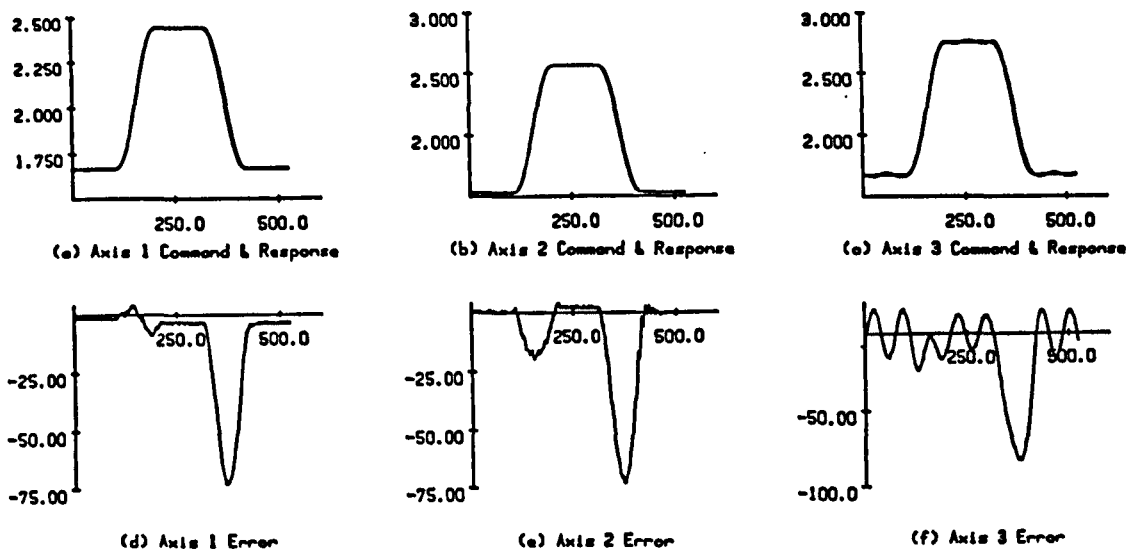
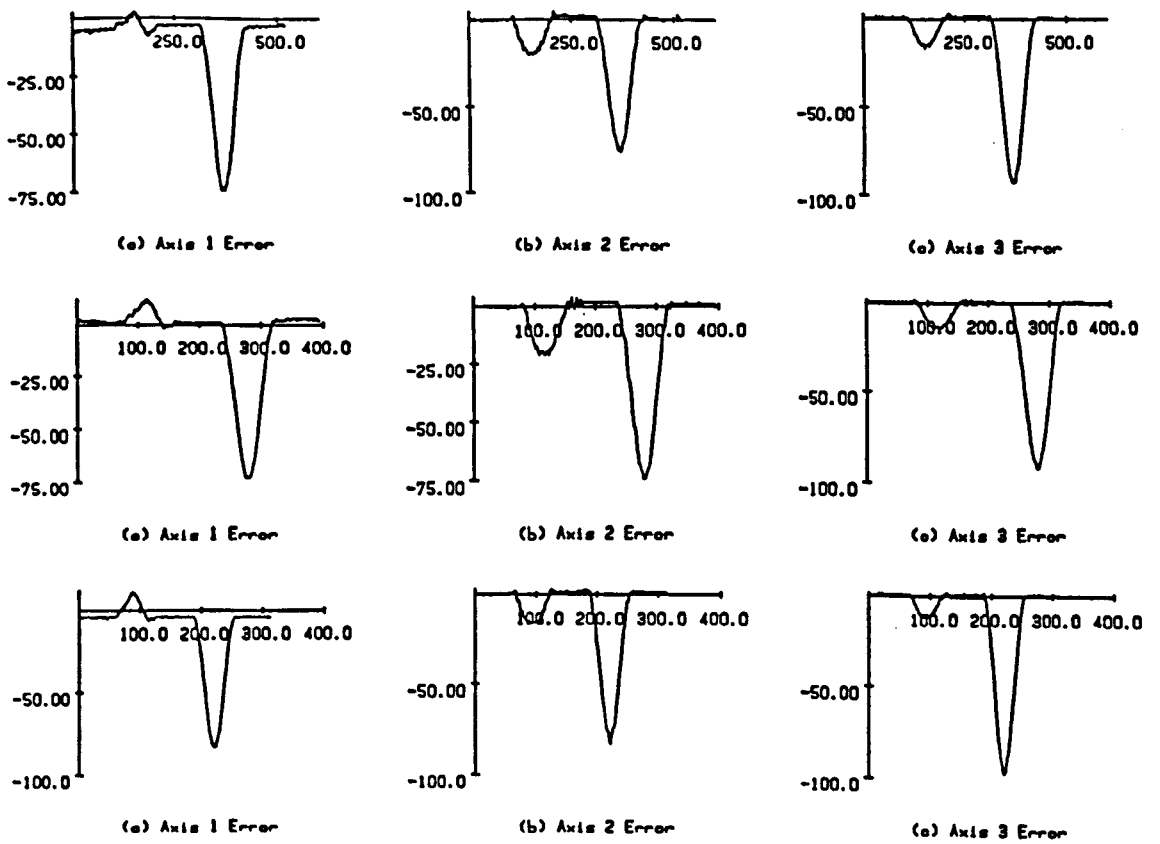


Figure 7.16 1 Hz Disturbance on Axis 3



Upper Figure 7.17 MRACS at 15ms Sample Interval
Centre Figure 7.18 MRACS at 20ms Sample Interval
Lower Figure 7.19 MRACS at 25ms Sample Interval

Earlier experiences suggested gains needed to be increased. The excessively long sample interval was considered to be the obstacle. It could not be reduced with the given hardware, but some insight might be obtained by increasing it. Figures 7.17 to 7.19 show the errors from the motion, operated at 15,20 and 25 ms. Note that the units of time change accordingly, although the scale width (overall duration) remains the same at 7.86 seconds. (524, 393 and 314 samples respectively). The differences in error level are negligible. The implications are either :

(i) 15ms is adequate as a sampling interval in this case.

or (ii) The non-linear adaptor is not contributing significantly to the system performance. Most of the performance improvement stems from the motion tuning, not the controller.

7.4.5 Cartesian Based Motions with Dynamic Tuning

Motions used in the tests carried out so far comprised joint based trajectories of the same basic form. The infinite array of potential variables to some extent necessitates this strategy. The analytic work carried out in earlier sections is by no means restricted to such simple motions. The tuning method particularly, is not specific to joint based motions. In order to demonstrate the wider capabilities, Cartesian motions were tested.

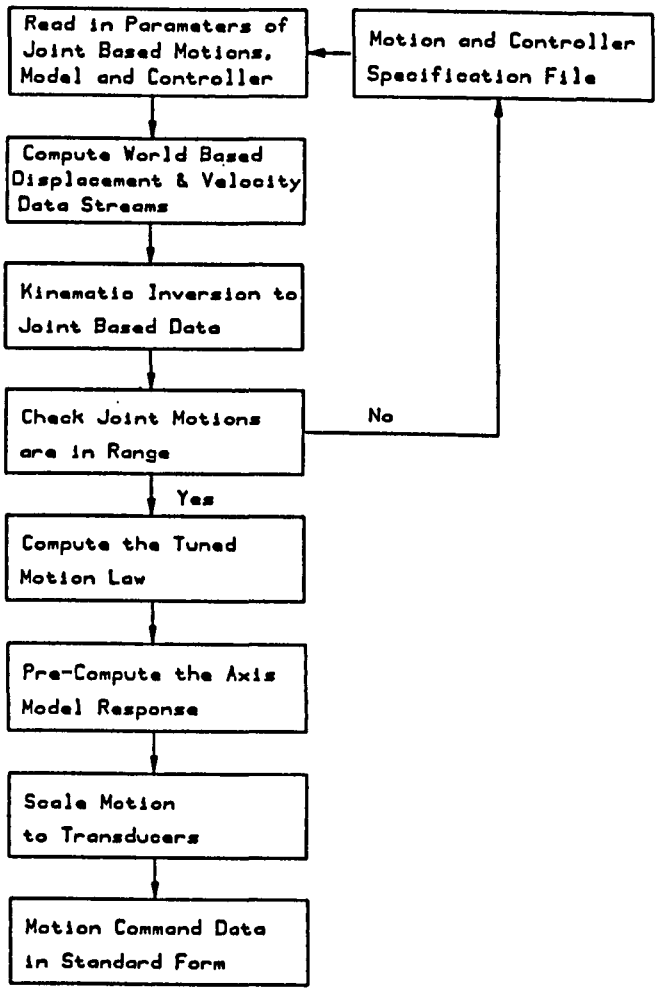


Figure 7.20 WMRACS Program Flowchart
(Coupled to figure 7.8)

The program used for the earlier MRACS implementation takes in motion data of the format described in chapter 4. All that is needed to generate Cartesian motions is a kinematic inversion, some means of

specifying the motion, checking its feasibility and dynamically tuning it. The program, called WMRACS.PAS (appendix B) is outlined in the flow chart, figure 7.20.

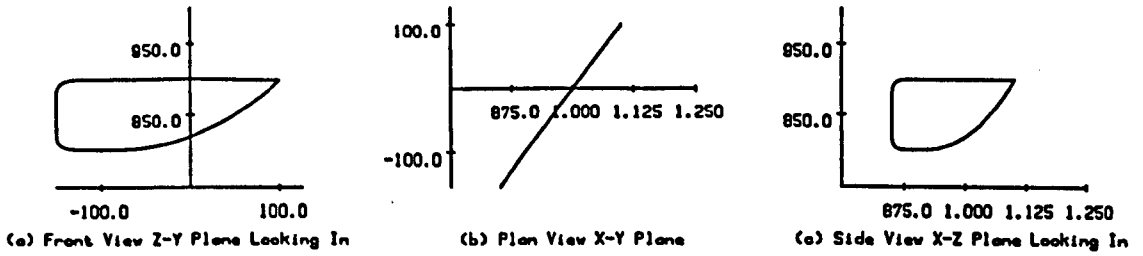
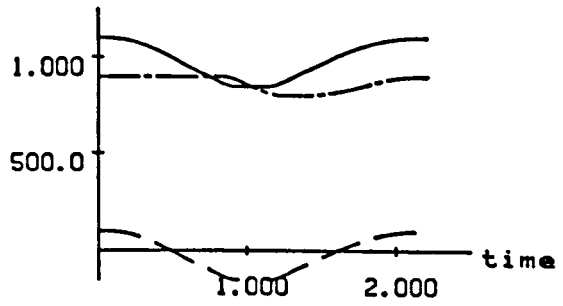


Figure 7.21 Cartesian Based Test Motions - Paths in Space

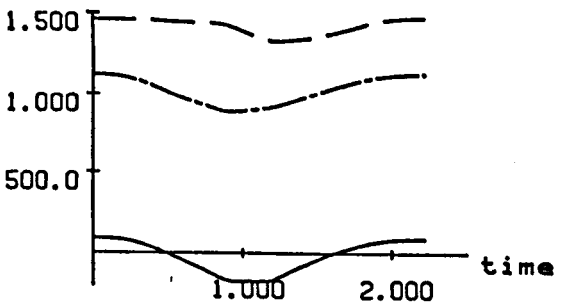
The first Cartesian motion generated was a horizontal straight line, in which the return is made via a vertical descent, followed by an arcuit retraction. Three views of the motion in Cartesian space are shown in figure 7.21. This is a five segment motion, the boundary conditions being defined in table 7.22. The nominal X,Y and Z displacements are shown in figure 7.23(a), with the corresponding nominal joint motions in figure 7.23(b). Likewise, X,Y and Z velocities along with their nominal joint counterparts are shown in figures 7.23(c) and 7.23(d).

	X(m)	Y(m)	Z(m)	t(sec)
d_0	1.1	0.1	0.9	
v_0	0	0	0	
a_0	0	0	0	0
d_1	0.9	-0.1	0.9	
v_1	-0.35	-0.35	0	
a_1	0	0	0	0.75
d_2	0.85	-0.15	0.875	
v_2	0	0	-0.3333	
a_2	0	0	0	0.975
d_3	0.85	-0.15	0.825	
v_3	0	0	-0.3333	
a_3	0	0	0	1.125
d_4	0.9	-0.1	0.8	
v_4	0.35	0.35	0	
a_4	0	0	0	1.35
d_5	1.1	0.1	0.9	
v_5	0	0	0	
a_5	0	0	0	2.1

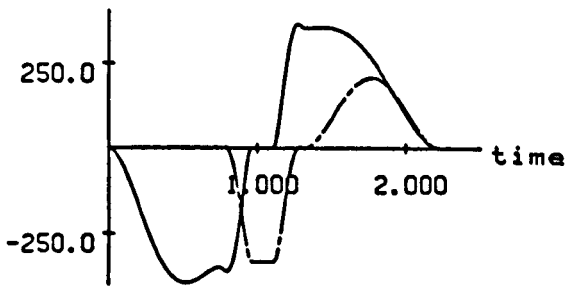
Table 7.22 WMRACS Program Motion Boundary Conditions



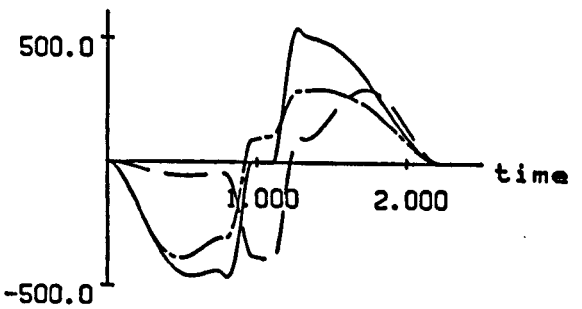
(a) Nominal Cartesian Displacements



(b) Nominal Joint Displacements



(c) Nominal Cartesian Velocities



(d) Nominal Joint Velocities

Figure 7.23 Nominal Joint Axis & Cartesian Coordinate Motions

Key

X or Axis 1 ————

Y or Axis 2 — — —

Z or Axis 3 —·——

The distortions due to the kinematic transformations can be seen, but it is noted that these distortions are not large for the region of motion considered and the particular robot. The responses (scaled as 12 bit integer units, as opposed to the earlier figures which are 'real' joint variable values in terms of radians/metres etc.) are shown for axes 1,2 and 3 in figure 7.24. Repeating the same motion with each of the segments at half the duration gave a great increase in error magnitudes, as is seen in figure 7.25.

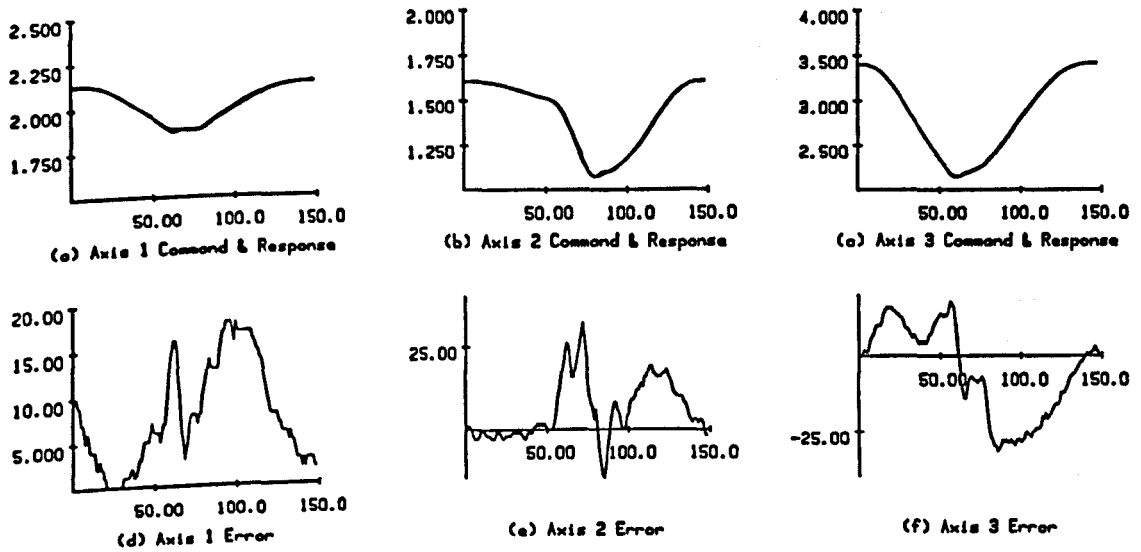


Figure 7.24 Cartesian Based Test Motion - 2.1 Second Duration

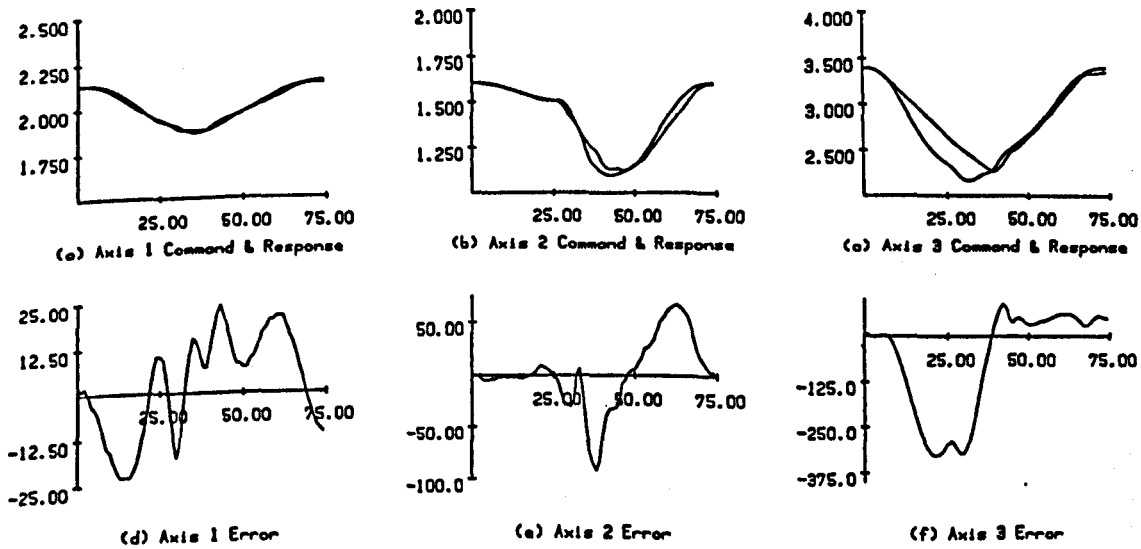


Figure 7.25 Cartesian Based Test Motion - 1.05 Second Duration

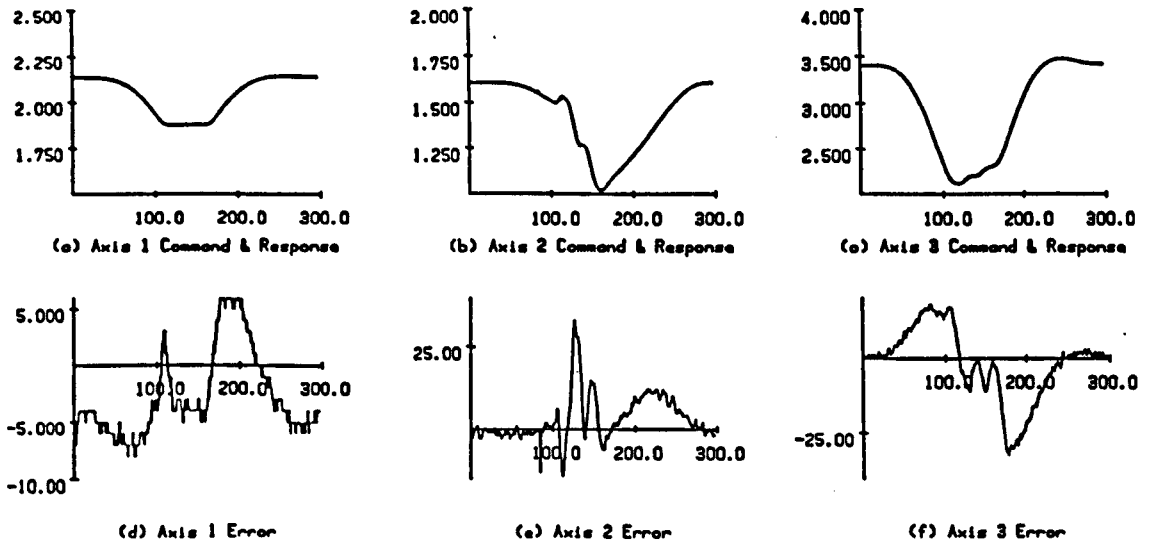


Figure 7.26 Cartesian Based Test Motion - 4.2 Second Duration

The large increase appears predominatly due to the saturation of axis 3, and the subsequent interactions between the axes. Doubling the motion durations relative to the first set, gave the responses shown in figure 7.26. Note the added complication in interpreting the results, due to the polynomial wandering introduced, in changing the durations. Boundary conditions could have been altered to take account of this. The error is aggravated by the increased complexity in the command being tracked.

The results from the MRACS scheme are not rewarding. The promise of earlier single axis work was not demonstrated in the expanded scheme. The reasons for the degradation are thought to include :

(i) Sample interval. The single axis analogue version had a far wider bandwidth than the final digital implementation.

(ii) System gains were severely limited, probably by virtue of the sample interval.

(iii) Filtering on the differential pressure feedback introduced additional lags in the system.

(iv) Velocity estimation noise is a function of displacement resolution and sample interval, both of which were not high enough.

8 Self Learning for Motion Speed Increase

The conventional optimisation of motion duration is dependent upon the development of accurate dynamic models and the knowledge of all the parameter values [3.2, 3.3, 4.6, 8.1, 8.2, 8.3]. It is then a complex task to compute the optimum motion, which is specific to one particular robot and a given motion.

Instead of working from the standpoint of artificially imposed limits, why not speed up the robot until it exhibits signs of saturation? If it were possible to develop a simple, probably experimentally based technique, a number of the problems might be resolved.

It is necessary to stop speeding up at saturation to :

(i) prevent a loss of axis coordination (giving rise to collisions).
(ii) reduce vibration causing low component lives

8.1 Optimum Motions Along a Fixed Path in Space.

Problem Type - In practice the problem could be specified as to minimise time taken to move along a fixed path in space, given boundary conditions which probably include :

$$\begin{aligned} &\text{displacements} && q(0),\ q(T) \\ &\text{velocities} && \dot{q}(0) = \dot{q}(T) = 0 \\ &\text{accelerations} && \ddot{q}(0) = \ddot{q}(T) = 0 \end{aligned} \qquad (8.1)$$

for the dynamic system :

$$H(\underset{\sim}{q})\ddot{\underset{\sim}{q}} + \underset{\sim}{V}(\underset{\sim}{q},\dot{\underset{\sim}{q}})\dot{\underset{\sim}{q}} + \underset{\sim}{G}(\underset{\sim}{q}) = \underset{\sim}{\tau} \qquad (8.2)$$

subject to actuator constraints such as :

$$\underset{\sim}{\tau}_{min}(t) \leq \underset{\sim}{\tau} \leq \underset{\sim}{\tau}_{max}(t) \qquad (8.3)$$

8.1.1 Optimal Control

Pontryagin's Maximum principle states that a necessary condition for a maximum of the performance index is that the system Hamiltonian be a minimum with respect to the control input (and vice versa for the minimum principle). In practice the solutions for the required control input are intractable for all but the simplest of systems. The robot

trajectory duration minimisation can be described as a two point boundary value problem applied to a set of coupled non-linear differential equations. Solutions have been obtained by linearising the dynamics of particularly simple classes of robot.

8.1.2 Searching Techniques

A phase plane representation of the motion along a fixed path or alternatively a tesselated grid or discretised state or torque space (Look up table) is searched using varying degrees of knowledge about the system. With constraints such as those imposed by actuators, regions within the search 'maps' become inadmissable. Whether or not the solutions obtained are minimum time will depend on the solution strategy.

8.1.3 Model Based Techniques

Classic analyses of the minimum time problem consider the system open loop, pure torques being generated exactly as required. They are simulations, in which the parameters are arbitrary and have configurations which inherently simplify the dynamics. Assumptions made include perfect parameter and motion variable measurement, the knowledge of higher derivatives of motion variables, the absence of noise, infinite sampling frequency and the classic rigid body dynamic model. Constrained to the computer environment, the performance of such models is not surprisingly high.

More recent analyses are likely to include some actuator limitations (albeit simply modelled) and closed loop control.

For each given set of robot joint displacements and velocities, Shiller and Dubowsky 1985 [3.2] compute ranges of maximum and minimum joint acceleration capabilities. By writing the equations of motion in the form :

$$\underset{\sim}{H}(\underset{\sim}{q})\ddot{\mu} + \underset{\sim}{V}(\underset{\sim}{q})\dot{\mu}^2 + \underset{\sim}{G}(\underset{\sim}{q}) = \underset{\sim}{\tau} \qquad (8.3)$$

where μ is the displacement along the pre-defined path, the contributions to μ from each of the joints are considered in turn, and the overall maximum and minimum measured, these being the combined

limits imposed by each of the actuators. These are functions of μ, $\dot{\mu}$. The minimum time motion is achieved when $\ddot{\mu}$ is equal to either the combined maximum or minimum. The remaining problem is to find the switching points. This is performed on the path velocity/displacement curve, (phase plane) using the limiting tangential path velocity (the velocity at which there is no torque available to accelerate along the path) to locate the optimum switching points). It appears an elegant technique, and requires :

(i) A full dynamic model and coefficient values.
(ii) Actuator saturation limits as functions of joint displacement and velocity.
(iii) Predefined paths.
(iv) Multiple switching points.
(v) Substantial iterative computation.

The models are the conventional basic dynamics with the addition of actuator saturation. Instantaneous changes in acceleration, velocity and torque are required.

Shin and McKay 1985 [8.1] adopt a similar technique. They express the dynamic equations in terms of the base parameter defining motion along the fixed 'geometric path constraint'. Input torque constraints (functions of the path parameters and its derivative) are incorporated. Manipulation of the second derivative of the path parameter (a single scalar) yields explicit bounds on it. They assume the application of Pontryagin's maximum priciple to be impractical in this case. Alternatively, advantage is taken of the form of their cost function.

Effectively, the maximum acceleration vectors on the plane are computed forward from the path start point, maximum deceleration vectors back from the end point. If they intersect then the single switching point path gives the minimum time trajectory. They may not because of various inadmissable regions. Strategies are described to cope with these other situations.

Goor 1984 [3.11] states that robot dynamics is dominated by that of the motor. This assumption leads to bang-bang motion control

requirements. Because of the simpler dynamic form, adaptive velocity feed forward terms are computed, nominally eliminating the dependancy of accuracy on speed. The reduced complexity is justified by magnitude comparisons between the constituents of the dynamic coefficients.

Seeger and Paul 1985 [8.2] uses Paul's earlier piecewise linear trajectories (figure 4.1, constant joint velocity with 'rounded corners') they then model the robot as :

$$u_j = b_{1,j}\ddot{\theta}_j + b_{2,j} \qquad (8.4)$$

where u_j - axis j input current

$\ddot{\theta}_j$ - axis j angular acceleration

$b_{1,j}$ total inertia seen at the motor

$b_{2,j}$ gravity loading and friction

The constants $b_{1,j}$ and $b_{2,j}$ are different for each transition and each joint. The simplifications made are admitted to be radical. Maximum axis motor currents are measured. On the next run, the segment duration is based on a fraction of the measured maximum current divided by the axis current limit. Because the motion laws are more or less constant velocity, it is easy to assess whether or not velocity limits are being exceeded.

Sahar and Hollerbach 1984 [8.3] point out that the application of standard optimal control to linearised dynamic models is cast into doubt because of the invalidity of neglecting velocity products. Their method appears to be a tree search of all possible solutions although the limited number of possible velocities at the end of each segment reduced the search size. The resolution of search was not very fine. Observations made are :

(i) The fastest path is close to a straight line in joint space
(ii) The path is symmetric about the mid-point between the starting point and the destination
(iii) The torque switches twice at most

It is computationally intensive, requiring from minutes to hours for a low resolution grid search. The three observations support the use of symmetric joint polynomials as motion laws.

8.1.4 Hollerbach's Time Scaling of Trajectories

Hollerbach 1984 [5.1] describes a means of recomputing the torque profile required for a motion, when the motion duration is reduced. Knowledge of the dynamic model is assumed, and is presented in a slightly different form than usual. To recompute the torque profile requires the separation of the gravitational term from the initial torque profile. The remainder is then multiplied by the speed increase factor squared, and then added to the gravitational term. Hollerbach does not consider the problem that the reduction in duration will cause the sample points to differ from the earlier computation. Some kind of interpolation is therefore required, if the dynamics are not to be completely recomputed. It does appear to be a very powerful technique for time optimisation of co-ordinated motions along specific paths. Hollerbach applies it to a simple DC servo-motor driven robot in a computer simulation.

By computing the available torque over the motion, net of the gravitational term, it is possible to compute a single scalar defining the speed increase which can be applied to the motion to just achieve actuator saturation. There are still simplifications, but they are minor compared to other methods. An interesting modification would be possible if the joint torques (as measured) were available on a robot. The gravitational torque terms are generally easy to compute. Computing the gravitational torques and subtracting from the total measured would then give the sub-total of the Inertial, Coreolis and Centripetal torques. If the actuator torque saturation limits were known, then the speed increase factor (for a specific co-ordinated motion) could be directly computed, without resorting to a large computational resource. In this half analytical/half experimental manner, the duration could be optimised, for a specific motion.

8.1.5 Summary of Optimised Motion Properties

Analytic/model based schemes for minimum time trajectories infer :

(i) Minimum time displacement paths are continuous in value to velocity at least, implying path functions which are at least quadratic.

(ii) Optimal torque 'commands' comprise switches from one extreme limiting state, to another. The number of these switches, although it can theoretically be large, is usually only a few, two being typical. The resultant velocity profile is, broadly speaking, a skewed parabolic curve. Hence if a polynomial displacement law approximation was used it would have to be at least cubic.

(iii) At any one time, only one actuator will, in general, be at its limit. Therefore saturation occurs at one actuator only and will be in a limited region of the motion.

(iv) The analytic minimum time problem becomes scalar and hence tractable if the path in space is predefined. The remaining 'variable' is the motion along the path, as a function of time.

(v) The fastest path can be approximated to a 'straight line' in joint space i.e. the joint motions are proportionately synchronous/co-ordinated to the same parameter i.e. time.

(vi) The path is near symmetric about the mid-point between the starting point and the destination.

(vii) Analytic methods assume knowledge of payload mass properties and complete manipulator dynamics.

(viii) Minimum time analysis is simplified if carried out in joint co-ordinate space, as the kinematic transformations are eliminated. A minimum time requirement often implies that the path is relatively arbitrary, hence there is reduced need for Cartesian co-ordinate based motions, other than for collision avoidance.

8.2 Actuator Characteristics

If motion durations are to be reduced without changing the manipulator design, then the limitations imposed by the actuators must be considered. Drives used in robots are mostly electric or hydraulic. Few pneumatic servo systems exist and so will not be considered.

8.2.1 Electrical Motors

Electrical motors which can be used as servo devices are generally:

DC Armature voltage control, fixed field current
DC Armature current control, permanent magnet, brushed
DC Armature current fixed, field current control
DC Field current control, permanent magnet, brushless
AC Two phase induction motor

The steady state characteristics (torque,speed) of these devices are well known, and in broad terms, shown in figures 8.1(a) to (c). Industrial robots do not operate under steady state conditions, so we are interested in the dynamically available excess torques. These will be functions of motor inertia and viscous friction, coil inductance and resistance, load characteristics, angular velocity, angular acceleration and to a lesser extent, jerk.

Limitations imposed within the motor will include maximum flux density, peak and continuous current limits due to winding temperature limits, which limit torque output, commutator (if any) current/speed characteristics, servo amplifier current limits, maximum bearing or commutator running design speed.

Grouping all the electrical motors together, a steady state torque limit is imposed by the lesser of :

(i) Flux saturation current
(ii) Windings temperature
(iii) Commutator current limit
(iv) Servo amplifier current limit
(This is normally set to suit motor parameters i,ii and iii)

Similarly, a steady state speed limit will arise from the lesser of :

(i) Viscous friction balancing output torque
(ii) Bearing rotational speed limitations
(iii) Back emf generated, balancing the supply voltage
(iv) Commutation speed limit

In between the two limits, the torque/speed limiting curve will be defined by one or more of the aforementioned, or the overall motor power curve.

In very broad terms, all these steady state torque/speed limitations may be encompassed by the curves shown in figure 8.3. Assuming the robot to possess more or less constant viscous friction at the joints, the dynamically available excess torque can be taken from the difference between the current constant speed/torque value and the upper torque limit on the curve.

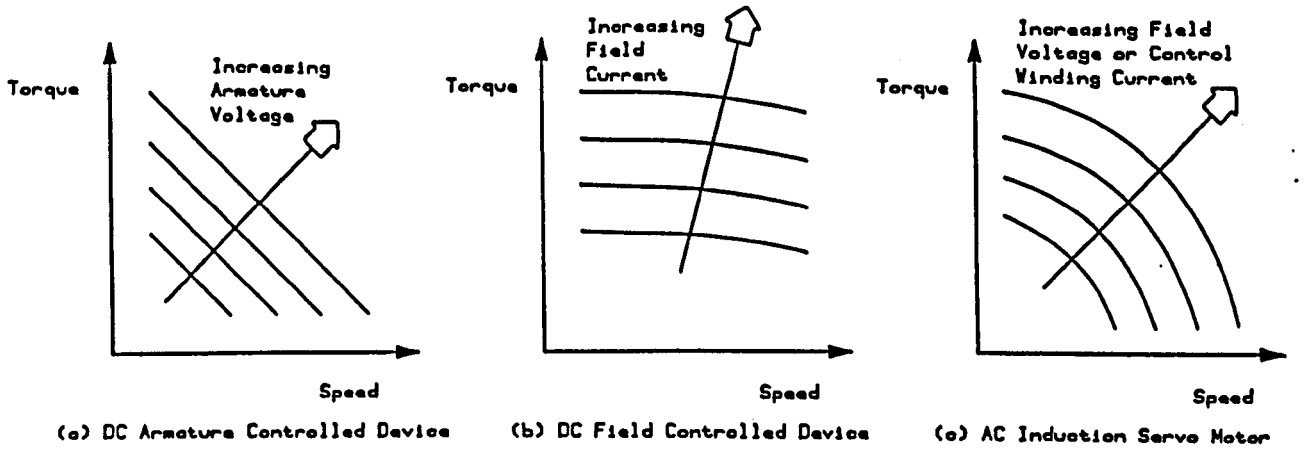


Figure 8.1 Electrical Servo Motor Characteristics

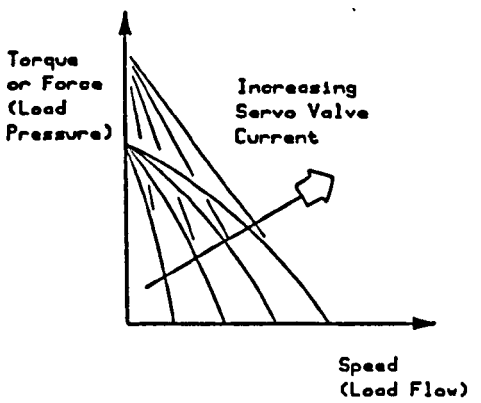


Figure 8.2 Hydraulic Servo Characteristics

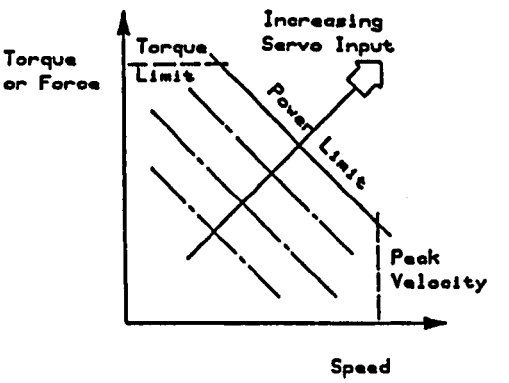


Figure 8.3 Servo Actuator Characteristics

8.2.2 Electro-Hydraulic Motors

The electro-hydraulic servo system presents similar limitations in performance. One, which is not so evident in the electrical system, can be termed power coupling. When multiple actuators demand power simultaneously, in the electrical case the supply rails can be maintained near design level because of the very large reserve in the basic power supply. In the hydraulic case, the total flow pressure combination available is limited. All the hydraulic actuators are taken from a common supply source, hence they are more or less at the same pressure. Should one draw a large flow at low pressure, for example in a gravitationally aided movement, the line pressure drops, and the other actuators are starved of flow. Considering a single hydraulic linear actuator or ram, the approximate force/speed curve, as determined by a flow control servo-valve is shown in figure 8.2.

The dynamically available excess forces will be dependent on the ram and load inertia and viscous friction, leakage, compressibility effects, the velocity and acceleration. The hydraulic servo is not as 'linear' as the DC motor. It is often modelled around a perturbation point, especially when the load reaction is small.

Steady state force limitations are very simple to arrive at, being the maximum zero or low flow system pressure, multiplied by the effective actuator area. Likewise, steady state speed limitations are imposed at zero or low pressure by the maximum pump flow rate. The steady state performance of the electro-hydraulic actuator can therefore be approximated to be similar to the electrical motor, in broad terms, as shown in figure 8.2.

8.3 Trajectory Time Scaling in Motion Generation

The trajectory which satisfies the required set of boundary conditions, is initially described by a function of time. Once computed, it becomes a sequence of displacement data points. Whilst a function, the most obvious way to produce the same displacement law profile for a new duration, is to recompute the law. For joint polynomial trajectories, solutions for the coefficients are defined in section 4.3 as :

$$\underset{\sim}{c} = \underset{\sim}{T}^{-1}\underset{\sim}{b} \qquad (8.5)$$

When algebraically inverted for the fifth degree symmetric case it requires around 33 multiplications, 15 additions, and the re-computation of the law i.e. 5 multiplications and 5 additions per sample. This computation is required for each of the axes, and is added to an already heavy computational load.

If the trajectory is also specified in terms of world co-ordinates then there is further overhead due to the additional kinematic solutions. This will vary depending on the robot configuration. For the three spatial axes of the Little Giant, it corresponds to around : 2 inverse tangents; 2 square roots; 15 multiplications; 14 additions; 2 cosines and 1 sine which may be up to 30,000 cpu execution cycles depending on the processor used.

Noting that the path shape in space does not change with time for co-ordinated motions, one option is to utilise the already computed path data and time scale using some interpolation method. (It would be simpler to change the sample interval, but this is not normally flexible.)

The most basic method, that of linear interpolation, is defined by :

$$r(to + \mu\delta t) \approx r(to) + \mu[r(to + \delta t) - r(to)] \qquad (8.6)$$

where $0 < \mu < 1$

In practice, the computation is reduced to around 5 additions and 3 multiplications per axis (appendix A). The error accumulation in its repeated usage is also shown to be small, for representative motions. There is also no need to re-compute the kinematics.

8.4 Scheme Strategy and Saturation Detection

8.4.1 Proposed Starting Point

A scheme for increasing speed can be developed, with the following characteristics :

(i) trajectory data defining the path in space as a function of time. A fifth degree polynomial defines the trajectory, which has been shown to exhibit an effective compromise between the

potential for increasing vibration and maintaining computational efficiency.

(ii) the trajectory duration can be progressively reduced using a simple relationship like :

$$T_{k+1} = R_k . T_k \qquad (8.7)$$

where T - motion duration
k - run number 1,2,3....
R - duration reduction factor

(iii) the trajectory data points can be re-computed sufficiently accurately using a simple linear interpolation formula.

Hollerbach's 1984 [5.1] time scaling property implies that there is a speed increase squared factor plus a constant offset, defining the change in the total torque demand as a function of motion duration. Hence, when progressively reducing the motion duration, in order to obtain approximately linear increase in demand torque, the decrease in duration can be derived by the recursive expression :

$$h_{k+1} = \sqrt{(2 - \frac{1}{h_k^2})} \qquad h_k = \frac{1}{R_k} \qquad (8.8)$$

where h is a scalar defining the change in time scale. This equation approximates to equation 8.7 if $R_k \approx 1$.

8.4.2 The Need for Saturation Avoidance

Once an actuator is operating in a saturated region, the servo system can no longer maintain proper control over the motion. This can lead to the loss of tracking accuracy, joint axis co-ordination and give rise to collisions. As an actuator enters and leaves saturation, there is a discontinuity at some derivative of the force or torque provided by the actuator. The lower the order of discontinuity, the greater the significance of the resultant vibration. This vibration will not only make the task more difficult to perform, but may reduce the components' working lives. Operating continually in a saturated region may also cause overheating and resultant rapid wear or damage.

8.4.3 Specification of an Idealised Saturation Parameter

Some parameter is required as an indicator of saturation, whilst the motion is cyclically reduced in duration. An idealised parameter would

have the following properties :

(i) It must cope with arbitrary motion boundary conditions. (Velocity and acceleration conditions should also be considered in general, but are neglected here.)
(ii) It should cope with varying and probably unkown loads at the gripper.
(iii) The computation required to evaluate it must not be excessive.
(iv) It should be repeatable and immune from noise sources such as signal noise, joint repeatability and discretisation.
(v) It should not be robot specific.
(vi) It should be a clear indicator of saturation, i.e. sensitivity should be high, giving a marked 'switching' level.
(vii) It should be able to detect saturation arising in a small region of the response on any axis.

Some of these requirements are conflicting and so a compromise must be reached. It is hypothesised that scalar quantities can be found, which are normalised to be independent of displacements and time, and satisfy most of the above requirements. Retaining a constant motion law function is later found to simplify the otherwise general nature of the problem.

The progressive onset of saturation is difficult to detect by eye in any one displacement response. For reduction factors close to unity, the change due to saturation between any pair of responses is small. Qualitatively, the robot itself behaves less smoothly and may begin to judder visibly, or overshoot its endpoint.

8.4.4 Onset of Saturation in Robot Responses

Looking in more detail at the displacement response and errors to a typical command, figures 8.4 to 8.10 cover the same motion, with durations 3, 1.5, 1.21, 1.03, 0.93, 0.83, 0.54 seconds. The motion details are shown in table 8.11. The left hand motion segment is progressively reduced, the right hand segment is held constant for reference.

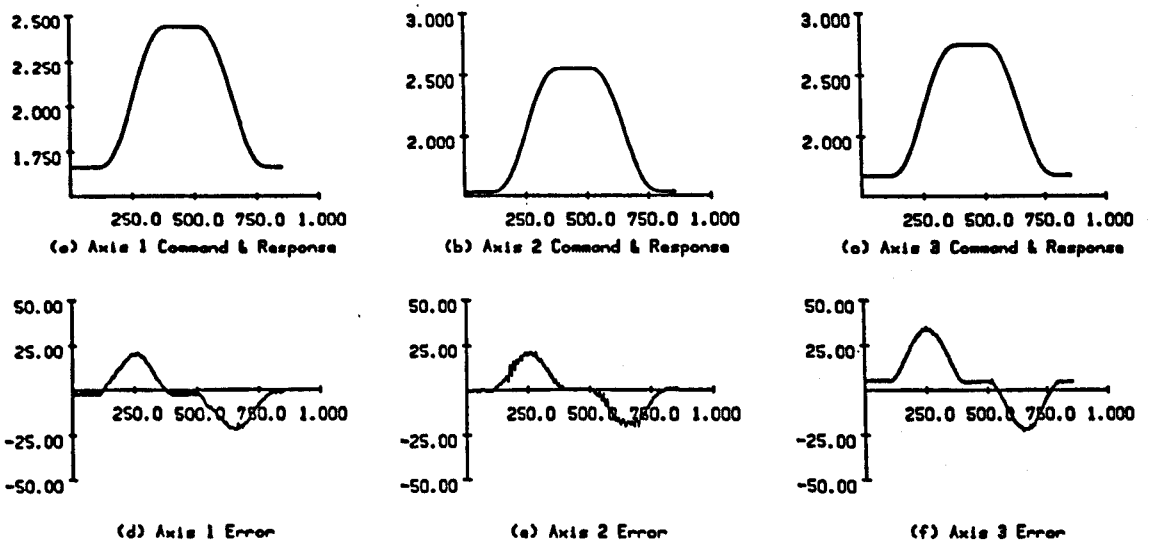


Figure 8.4 Test Motion - 3 Second Rise, 3 Second Fall

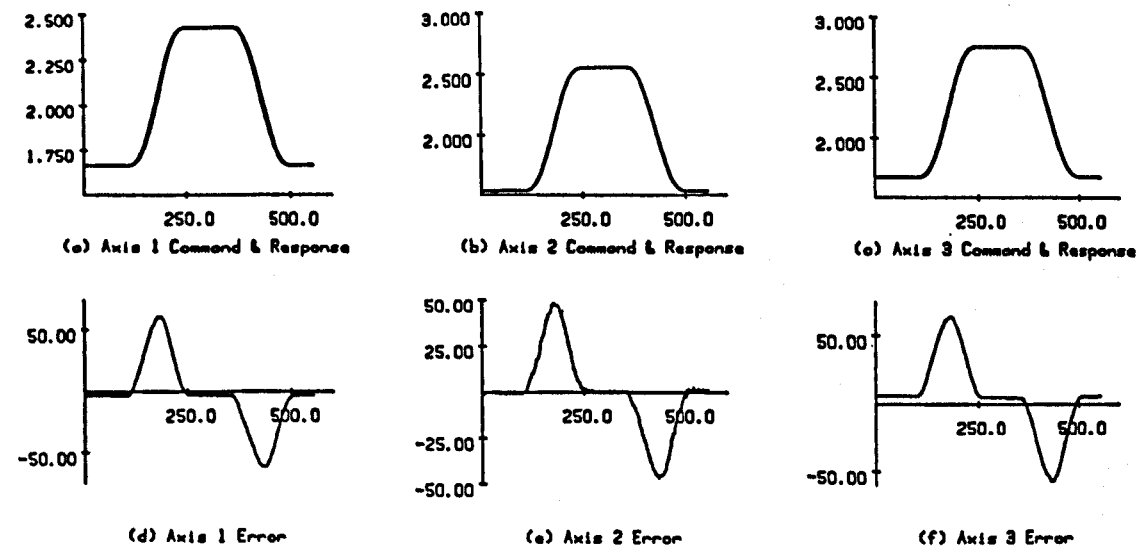


Figure 8.5 Test Motion - 1.5 Second Rise, 1.5 Second Fall

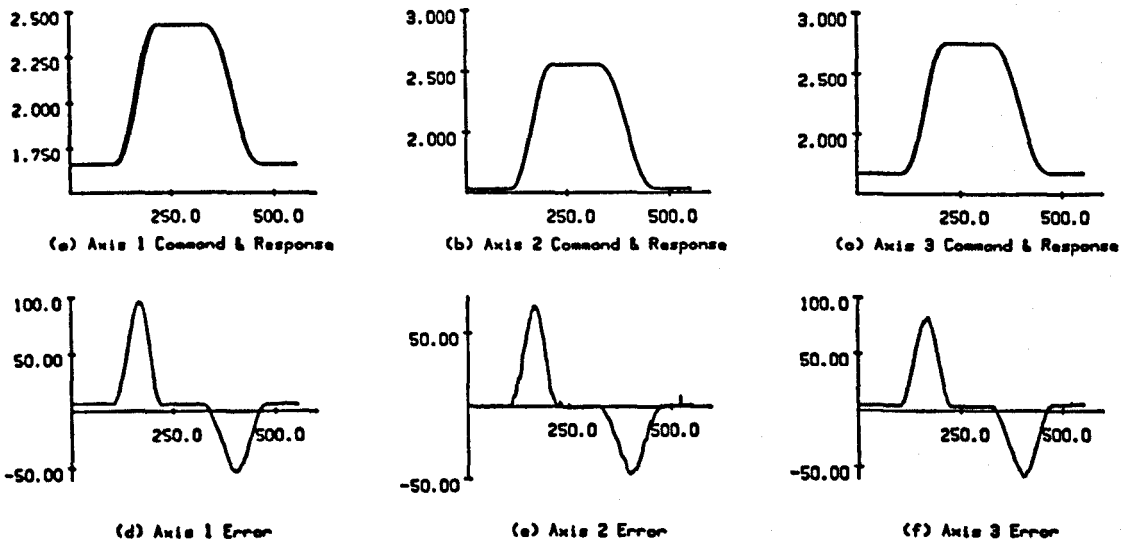


Figure 8.6 Test Motion - 1.21 Second Rise, 1.5 Second Fall

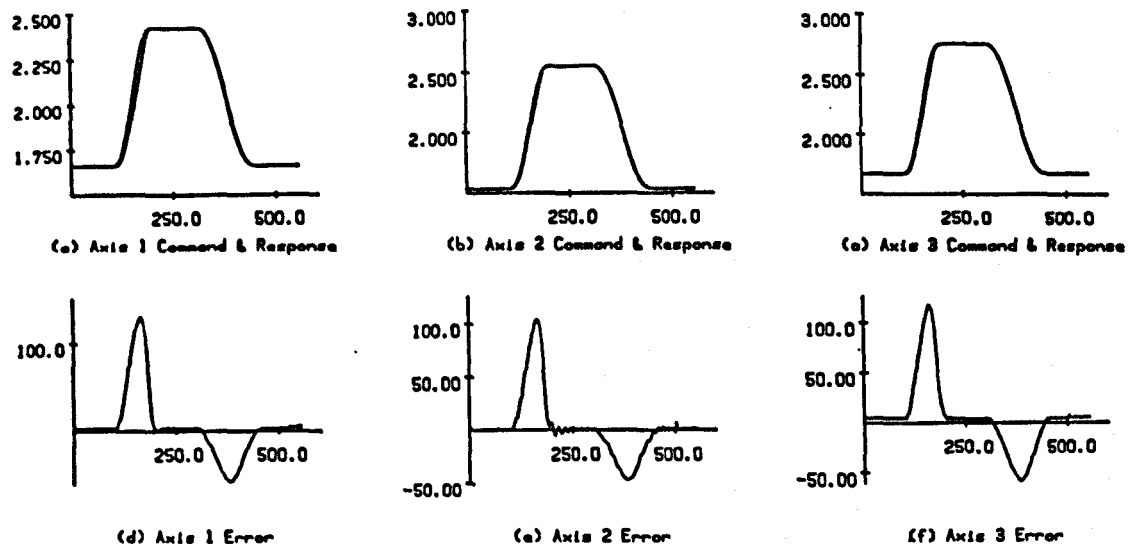


Figure 8.7 Test Motion - 1.03 Second Rise, 1.5 Second Fall

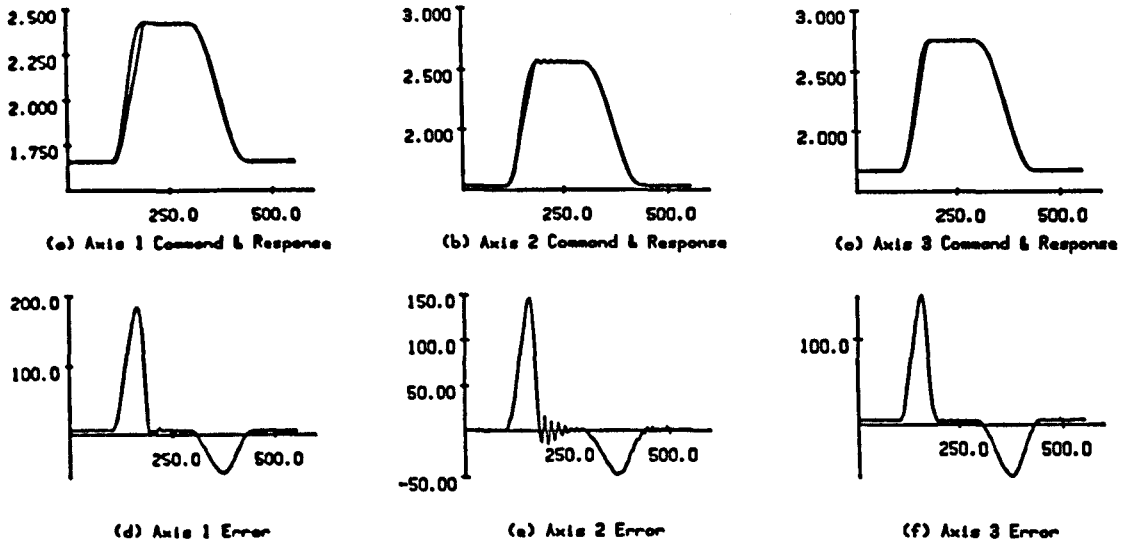


Figure 8.8 Test Motion - 0.93 Second Rise, 1.5 Second Fall

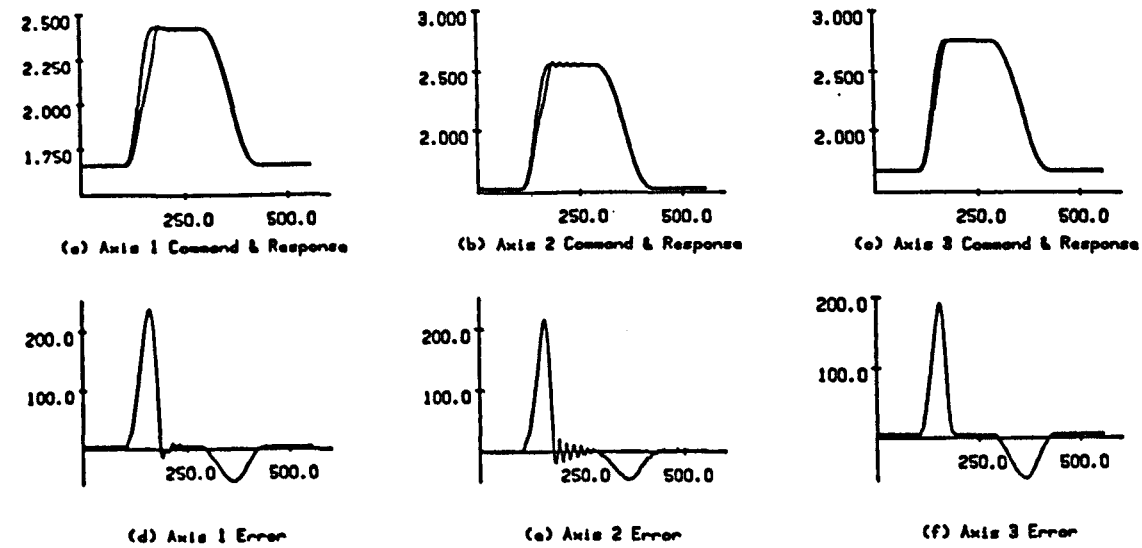


Figure 8.9 Test Motion - 0.83 Second Rise, 1.5 Second Fall

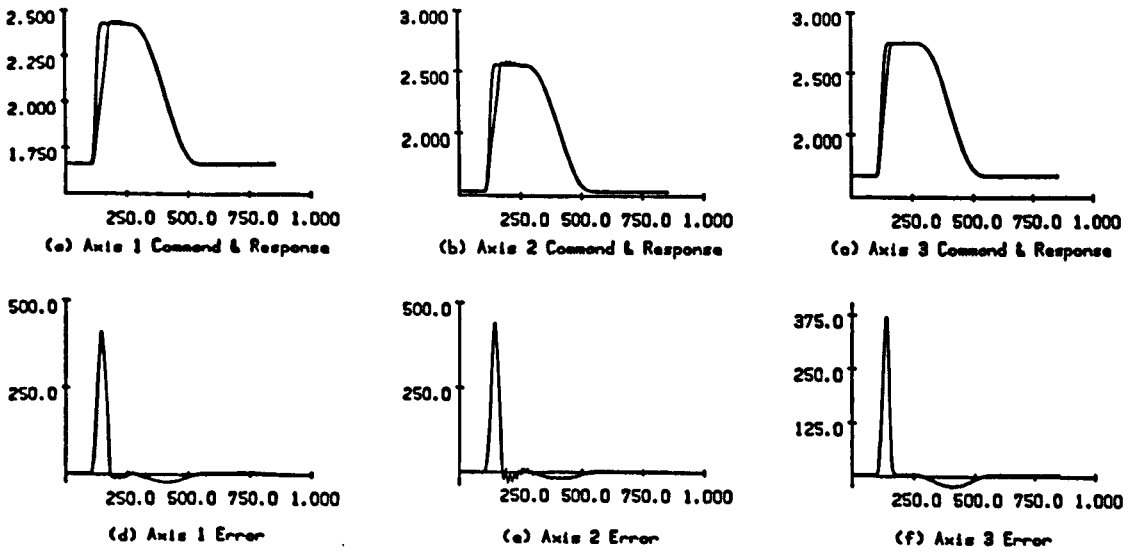


Figure 8.10 Test Motion - 0.54 Second Rise, 1.5 Second Fall

	Axis 1 Swing		Axis 2 Vertical		Axis 3 Horizontal	
Low Condition	-0.3927	rads	+1.4399	rads	+0.8	m
	-22.5	deg	+82.5	deg	+800	mm
	1163	units	1535	units	1676	units
High Condition	+0.3927	rads	+1.7017	rads	+1.0	m
	+22.5	deg	+97.5	deg	1000	mm
	2431	units	2559	units	2751	units
Motion Range	0.7854	rads	0.26181	rads	0.2	m
	45	deg	15	deg	200	mm
	768	units	1024	units	1075	units
	(±120 deg max)		(±30 deg max)		(762 mm max)	

Table 8.11 Test Motion Boundary Conditions

The 3 second duration, figure 8.4, actually appears too slow. The robot is driven via servo valves and hydraulic cylinders. The Moog servo valves used possess non-linearities in control flow (between 50% to 200% of normal flow gain) at low flows because of the spool null position design and manufacturing tolerancing. Hydraulic cylinders also possess high levels of Coulomb friction, due to the high sealing forces. Although there is integral action in the servo loop, the oil chamber is also known to contain entrained air. It is these three i.e. Coulomb friction, fluid compressibility and integral action which causes this situation when servoed at low cylinder ram velocities.

Reducing the duration to 1.5 seconds, figure 8.5, the error curves appear smoother and more predictable. The steady state errors can

usually be nullified, being due to DAC and ADC offsets, various servo and power amplifier offsets. The integral term, at first sight should cope with at least some of these, but the integrator (spool valve) possesses probably the most serious offset fluctuation within itself. The valve offset varies with temperature, acceleration, supply pressure, quiescent current and back pressure, each of which can account for around 2% variations.

Manifested in static error, the final value is typically less than 10 of the 2^{12} (12 bit) units. Where necessary (e.g. chapter 7) more effort has been applied to obtain the minimal null. (The design of robot causes the valve null setting procedure to be extremely dangerous).

Reducing the duration to 1.21 seconds, figure 8.6 again yields acceptable results. Note the presence of noise 'spikes' in some of the responses. The worst of these corresponds to about 5 of the 12 bit units or 10 mV. These are significant in the selection of a saturation detection parameter, in that they rule out some of the options.

Figure 8.7 shows the same motion, now at 1.03 seconds. There are noticeable (and audible) oscillations in the responses of axes 1 and 2. The previous duration (1.21 secs.) has probably the shortest duration, consistent with an acceptable response, for this particular motion. Note there is no exact solution to the shortest interval. Apart from probably varying with wear, oil temperature and time, it is a compromise between peak acceptable forces, vibrations and tracking errors versus the motion duration.

The response at 0.93 seconds duration (figure 8.8) shows marked saturation. Axis 1 and 2 responses can be approximated well to straight lines, suggesting a velocity or flow saturation. Once the response catches up with the command, the axis attempts to stop dead. The servo-valve is capable of closing completely, the robot's momentum is then opposed by a column of oil. The result is a pronounced oscillation in each of the axes. Axis 3 response is virtually unaffected. For this particular motion, it draws flow at a lower pressure than the other axes. There is both a low inertial and gravitational component. The errors are still large, but do not appear

saturated.

Much the same can be said for figure 8.9, with a duration of 0.83 seconds, except that axis 3 is also beginning to velocity saturate. Dropping the duration much further, to 0.54 seconds (figure 8.10) shows little change in the form of the response, the peak errors have risen still further. Sustained cycling at such durations took their toll with some bolts shearing, others working loose.

8.5 Saturation Detection Philosophies

8.5.1 Robot Model Based Philosophies

A complete robot and actuator model could be used to compute the point at which saturation occurs. Motions could then be designed to avoid such conditions. Motion feasibility checking was carried out in this way on the three planar axes of the Little Giant robot and is reported in Rees Jones et al, 1984 [4.8]. This requires the model derivation, parameter measurement and complete dynamics computation for the entire motion. The result is a yes or no answer; ideally some speed up or down factor should be given as output.

Measurement of actuator parameters such as flow or current, force or torque also enable saturation detection. Coupled with a simple actuator model the saturation point may be predicted shortly before it is reached and evasive action taken. Seeger and Paul 1985 [8.2] and others adopt similar approaches.

8.5.2 Input and Response Based Philosophies

Without reference to dynamic models, there are clearly patterns exhibited within the saturated results. It should therefore be possible to isolate these patterns and so identify their presence. The results as presented suggest it is easy to detect saturation from the displacement response. In a practical implementation it would be completely unacceptable to advance as deeply into saturation, as say in figure 8.10. The onset would have to be detected in its early stages. The sensitivity of the method therefore needs to be high.

Large tracking errors in themselves are unusable as a saturation parameter. They are a function of motion duration, and the complete sequence of configurations throughout the motion. Without some means of normalising the error magnitude, for the widely varying motion displacements, durations, robot configurations etc. their magnitude has no meaning.

The presence of flat segments in the response can be detected by differencing successive response data points. It would appear that saturation in this particular robot is dominated by flow constraints.

Oscillations are clear to the eye. It may be possible to automate detection of these in the early stages of saturation. Methods for carrying this out include :

(i) Extraction of frequency components.
(ii) Detecting sequences of sign changes in slope or velocity reversals.
(iii) Peak response acceleration measurement.

There may be a solution here, but it is more complex than at first sight. (i) requires a lot of computation, and possesses problems in discerning between normal response frequency components and those due to oscillation. (ii) can be carried out easily, but as seen in the unsaturated responses, (3, 1.5 and 1.21 seconds) there are oscillations present, which are not necessarily saturation related. The scheme would require some form of oscillation amplitude dependence or filtering, which in turn would require normalisation for the variety of motions possible. (iii) could entail additional transducers.

It may be that certain sections of a response are repeated when a system is saturated. Correlating response segments would require analysis to firstly locate the two relevant parts, from two successive responses, and then a cross correlation technique could be applied to compare them.

Distortion in the displacement error curve can be quantified but relative to what? Without bringing in dynamic models, one possibility,

is to relate current error levels to previous ones. The shapes change progressively. The change in error curve shape between figures 8.6 and 8.7 (1.21 and 1.03 seconds) is very small. Detection would therefore be difficult.

The detection of changes in displacement error does possess some attractions. Computing the peak errors (the variable peakerrj or SP_1) for progressively reduced durations, figure 8.12(a), it is clear there is a change in the characteristic. These peak curves were susceptible to repeatibility errors. Superseding them with absolute average errors (absumerrj or SP_2, the lower curves in figure 8.12(a)) yielded some improvement.

Attempts were made at differencing the data, 2nd and 3rd order differences did show changes in the regions of interest. These changes are not consistent and vary widely in magnitude. Figure 8.12(b) shows the change in average error after each run.

If a simple function could be fitted to the unsaturated region of the average error curve, then departure from the curve fit may serve as an indication of saturation. Curve fits to the average error were found to include functions such as 1/T, quadratics and cubics. The change in the average error is found to be progressive, making the departure due to saturation very difficult to distinguish.

The response of the manipulator for any one motion appears as if it could be modelled as a low order system. Deriving an error function for a fifth degree polynomial when driving a second order system yields a gross component :

$$\frac{2880\,\zeta\,(d_2 - d_1)}{(w_n T)^5} \qquad (8.9)$$

where ζ - second order damping ratio

d_1, d_2 - initial and final displacement boundary conditions

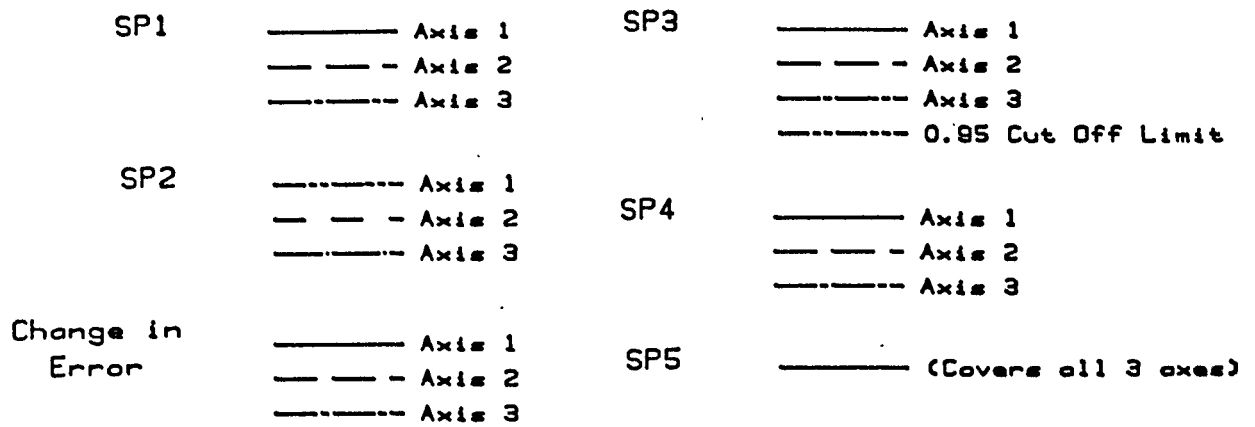
w_n - system natural frequency

T - motion duration

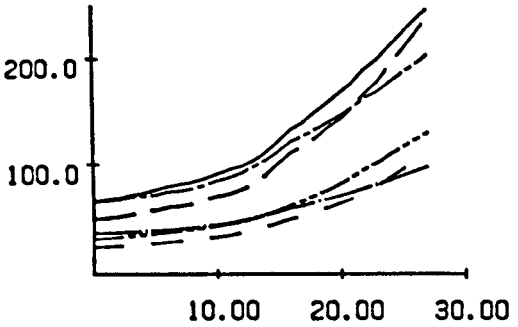
The transient error component is far more complex. Curve fits to the above function were found severely lacking. A more realistic quantity in detecting the change in slope of the average curve, is found simply to be the ratio of last average error divided by the current average

error, (satparam$_j$ or SP$_3$, figure 8.12(c)). This is automatically normalised, but it still suffers from inconsistencies and does not have a sufficiently pronounced change in it.

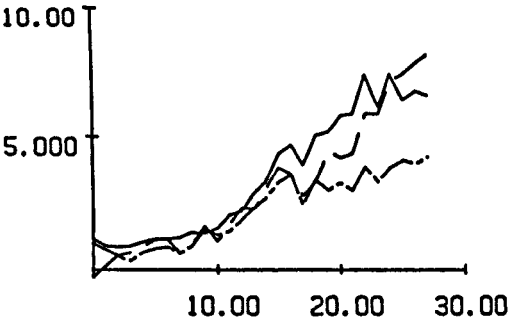
In an attempt to increase consistency and sensitivity, the previous ratio was computed as a rolling average (vecsat$_j$ or SP$_4$, figure 8.12(d)). This takes at least three cycles before it has a value and diplays no increase in consistency. If averaged over all the axes, (scasat or SP$_5$, figure 8.12(d)) it is more consistent, but will no longer detect the saturation of individual axes. Figures 8.13 to 8.17 show the same parameters for a wider range of test conditions.



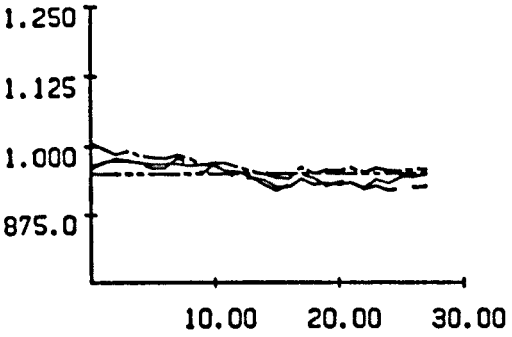
Key to Figures 8.12 to 8.17



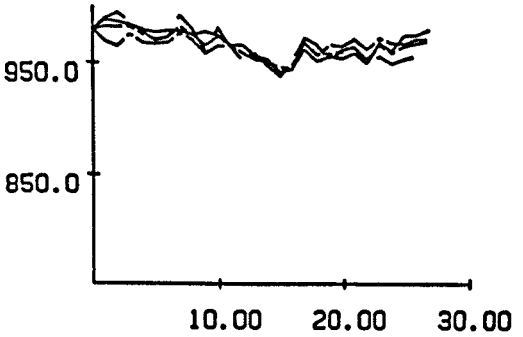
(a) SP1 & SP2 vs Run Number



(b) Change in Error vs Run Number

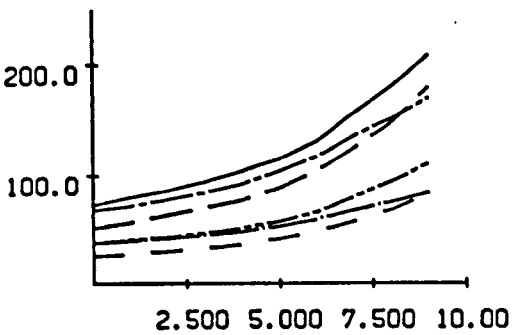


(c) SP3 vs Run No. (& 0.95 cut off)

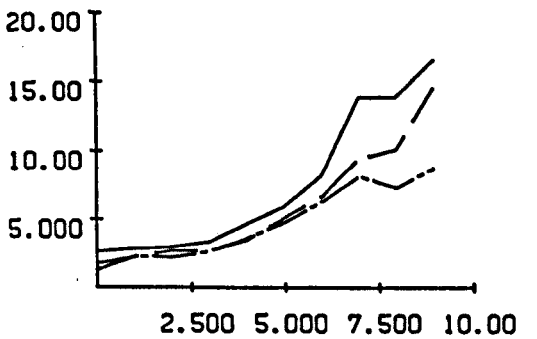


(d) SP4 & SP5 vs Run Number

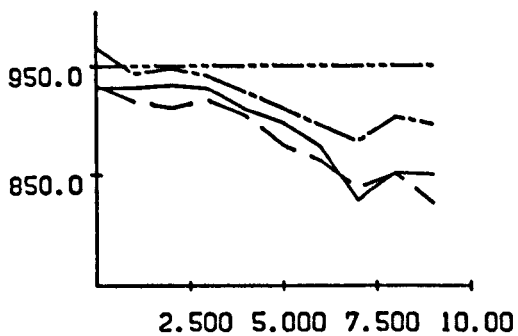
Figure 8.12 Load Off, Automatic Cut Out Off, Initial Duration 1.5s, 30 Cycles, Reduction 0.98



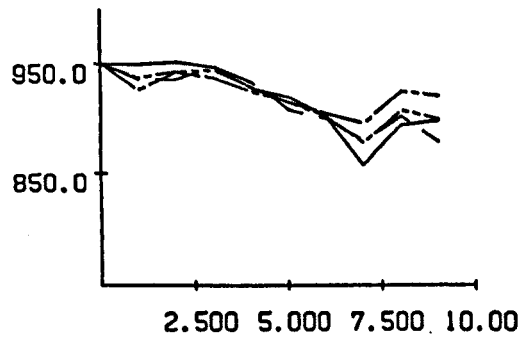
(a) SP1 & SP2 vs Run Number



(b) Change in Error vs Run Number

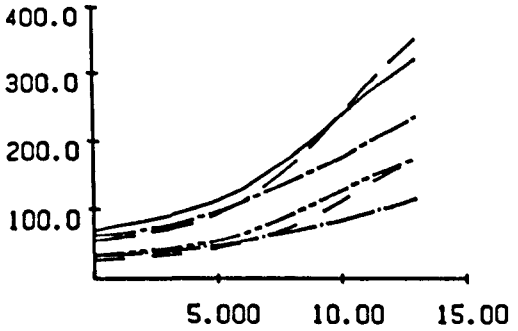


(c) SP3 vs Run No. (& 0.95 cut off)

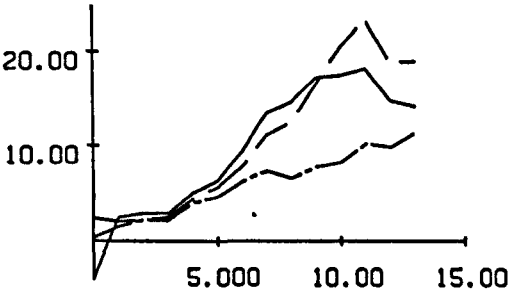


(d) SP4 & SP5 vs Run Number

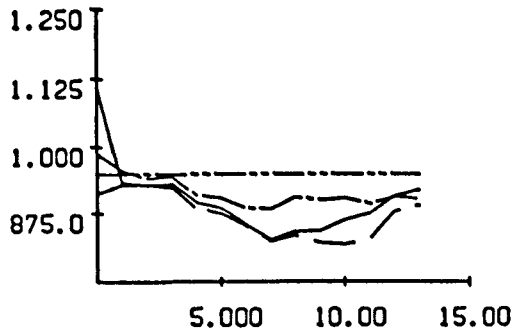
Figure 8.13 Load Off, Automatic Cut Out Off, Initial Duration 1.5s, 12 Cycles, Reduction 0.95



(a) SP1 & SP2 vs Run Number



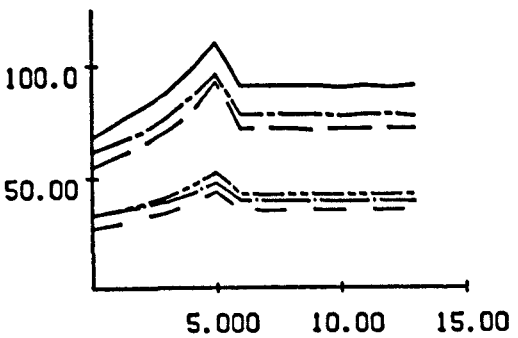
(b) Change in Error vs Run Number



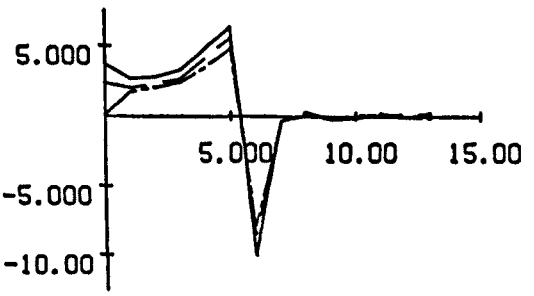
(c) SP3 vs Run No. (& 0.95 cut off)

(d) SP4 & SP5 vs Run Number

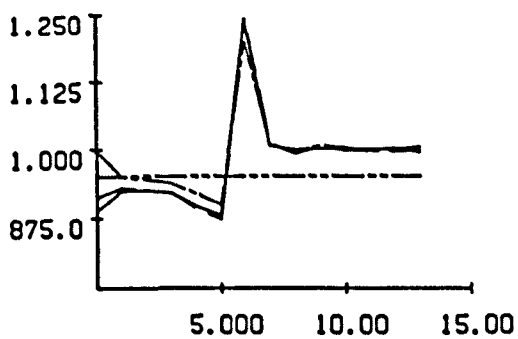
Figure 8.14 Load On, Automatic Cut Out Off, Initial Duration 1.5s, 16 Cycles, Reduction 0.95



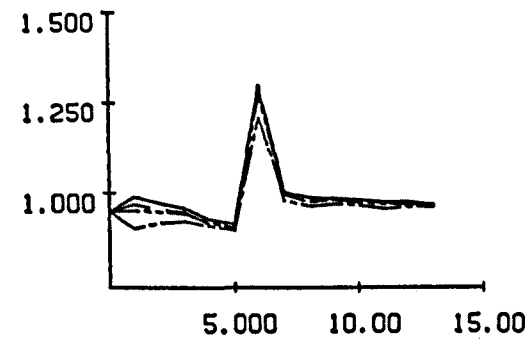
(a) SP1 & SP2 vs Run Number



(b) Change in Error vs Run Number

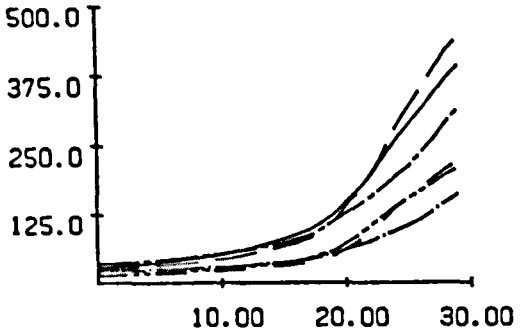


(c) SP3 vs Run No. (& 0.95 cut off)

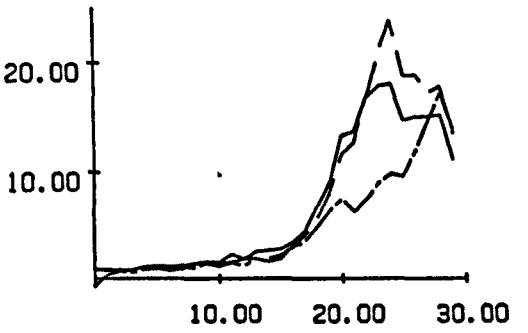


(d) SP4 & SP5 vs Run Number

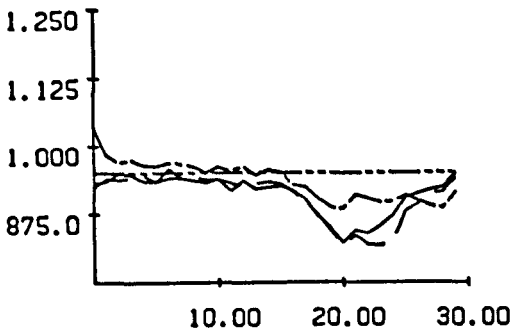
Figure 8.15 Load On, Automatic Cut Out On, Initial Duration 1.5s, 16 Cycles, Reduction 0.95



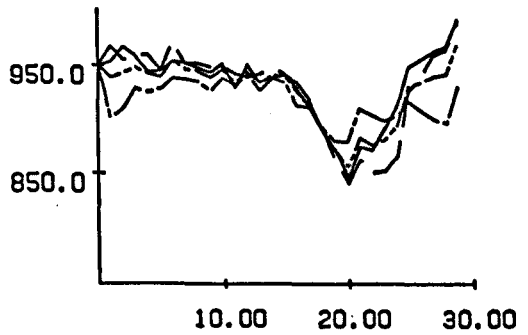
(a) SP1 & SP2 vs Run Number



(b) Change in Error vs Run Number

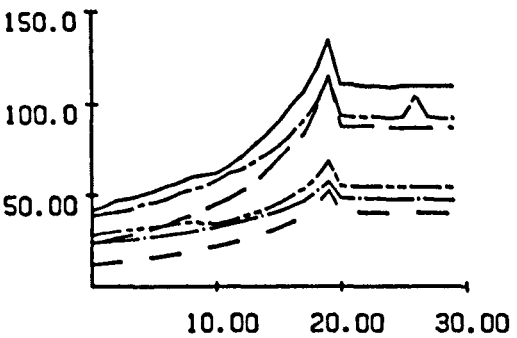


(c) SP3 vs Run No. (& 0.95 out off)

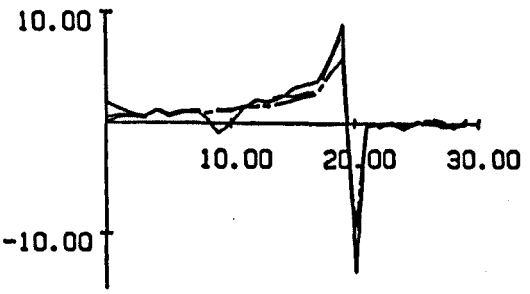


(d) SP4 & SP5 vs Run Number

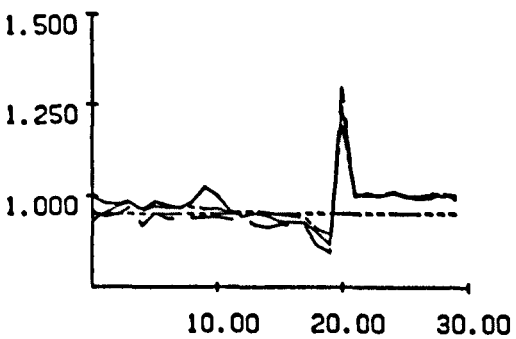
Figure 8.16 Load On, Automatic Cut Out Off, Initial Duration 3s, 32 Cycles, Reduction 0.95



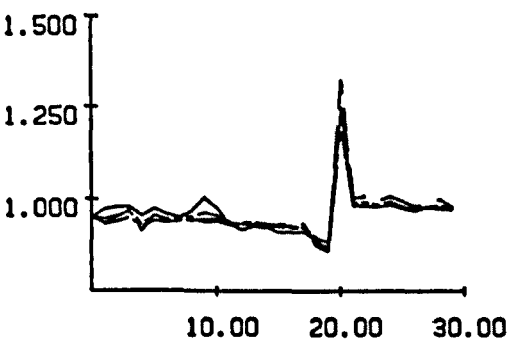
(a) SP1 & SP2 vs Run Number



(b) Change in Error vs Run Number



(c) SP3 vs Run No. (& 0.95 out off)

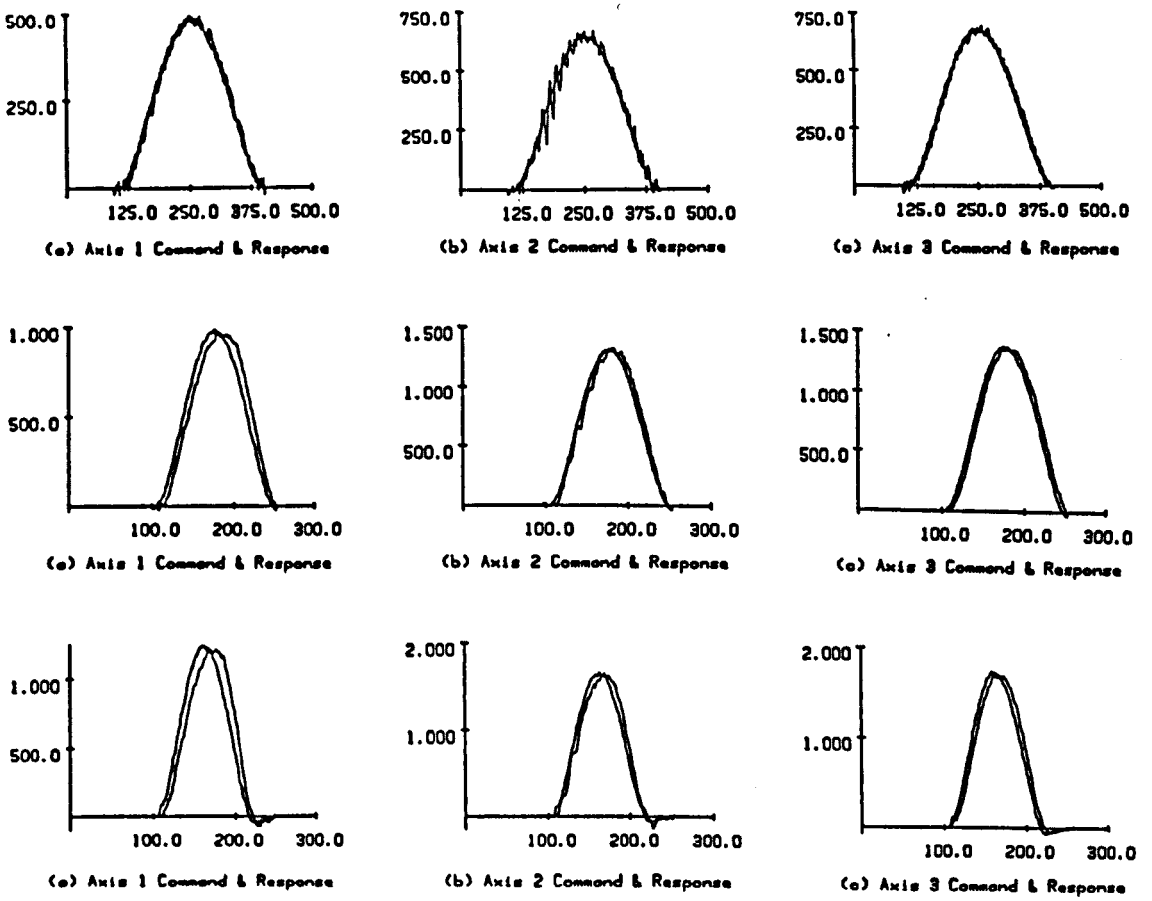


(d) SP4 & SP5 vs Run Number

Figure 8.17 Load On, Automatic Cut Out On, Initial Duration 3s, 32 Cycles, Reduction 0.95

8.2.3 Axis Velocity Based Parameters

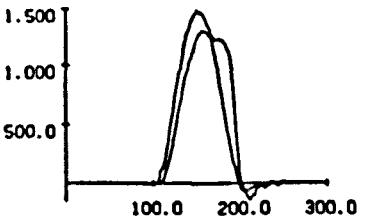
The parameters proposed so far do not possess the attributes required. There are promising signs, but the indicators are too specific, not sufficiently sensitive or inappropriate. In general, measuring quantities at higher order derivatives increases sensitivity to change. Considering harmonics and their time derivatives, for example, the amplitudes increase with derivative order.



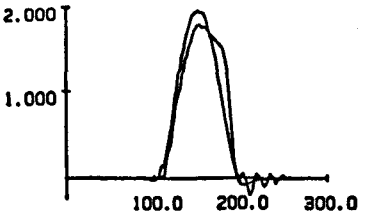
Upper Figure 8.18 Axis Velocity Curves 3 Second Rise (Filtered)
Centre Figure 8.19 Axis Velocity Curves 1.5 Second Rise (Filtered)
Lower Figure 8.20 Axis Velocity Curves 1.21 Second Rise (Filtered)

This particular robot has no velocity transducers. Velocity was therefore estimated using a central differences formula. To take account of noise, a second order digital filter was added. The lag

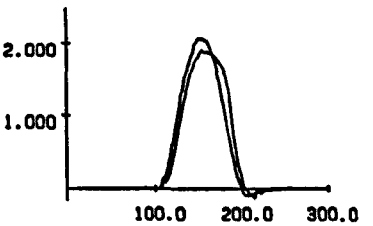
introduced by the filter was reduced with a compensating lead term. Figures 8.18 to 8.23 show the velocity command and response for the same set of motion durations (3 seconds down to 0.54 seconds). Some unfiltered curves are included (figure 8.24) to show the effects of discretisation and mechanical noise. Note the filter dynamics caused a small zero shift error.



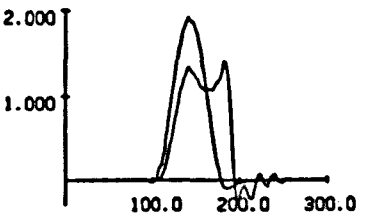
(a) Axis 1 Command & Response



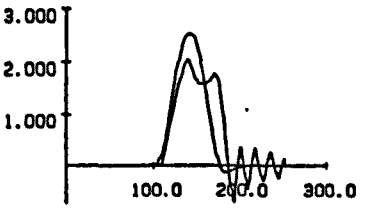
(b) Axis 2 Command & Response



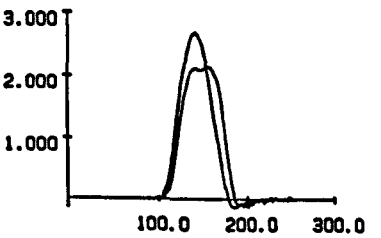
(c) Axis 3 Command & Response



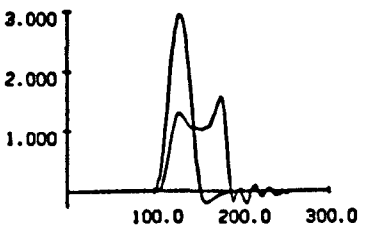
(a) Axis 1 Command & Response



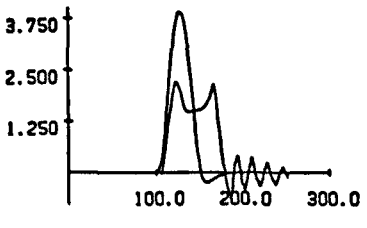
(b) Axis 2 Command & Response



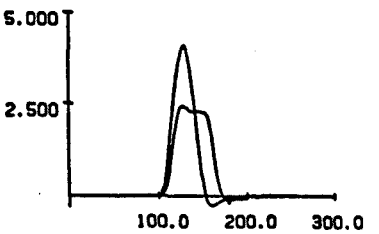
(c) Axis 3 Command & Response



(a) Axis 1 Command & Response



(b) Axis 2 Command & Response



(c) Axis 3 Command & Response

Upper Figure 8.21 Axis Velocity Curves 1.03 Second Rise (Filtered)
Centre Figure 8.22 Axis Velocity Curves 0.83 Second Rise (Filtered)
Lower Figure 8.23 Axis Velocity Curves 0.54 Second Rise (Filtered)

Displacement error resolution is small by virtue of the differencing of 12 bit resolution displacement transducer data. The velocity estimate suffers from the same discretisation and resolution problems to a greater extent. If velocity error is then computed from such data it possesses a very low signal to noise ratio, making it impractical

to use. Another option is to use the velocity in its own right. One feature which characterises a velocity curve for a stop-go-stop motion is that it starts and ends with zero velocity. This means that for the whole motion duration, the shape of the area under the curve is entirely defined by the function. It is also plain from the velocity response curves (figures 8.20 and 8.21) that saturation has occurred, when the duration has reduced from 1.21 to 1.03 seconds.

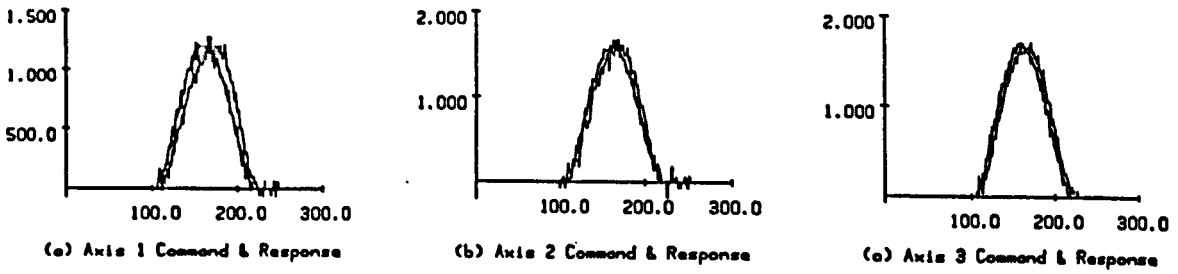


Figure 8.24 Axis Velocity Curves 1.21 Second Rise (Unfiltered)
(c.f. Figure 8.20)

The velocity curve shape can be quantified by the ratio peak to average. For a specific polynomial command this ratio is constant. In our case a 5th degree stop-go-stop, the ratio is 15/8. It is also easy to compute this ratio for the response velocity curve. Dividing the peak/average response velocity by the peak/average for the command results in a normalised standard value of unity for perfect tracking. This is found to be the most sensitive parameter of those tested, (velshapfacj or SP8 figure 8.25), and is probably the least affected by repeatability noise.

Figure 8.26 shows the use of SP8 when actually controlling the reduction of motion duration. Note that once saturation is detected, the motion duration is reset to a longer, previously acceptable value. Advantages with this ratio include :

(i) the peak velocity region is commonly the region which saturates first, so the parameter's sensitivity to localised saturation can be high.

(ii) the parameter is less sensitive to displacement repeatibility variations from cycle to cycle. This is because it

is only based on velocity values, from the current cycle. It can be evaluated on the first run, as compared with two or three runs required for some of the other parameters suggested.

Unless the system under control has a reasonable 'real time' feedback controller, there may be more velocity response curve distortion due to coupling etc. than was experienced in these tests. The work of this chapter is reported in Vernon, Rees Jones and Rooney 1986 [8.4].

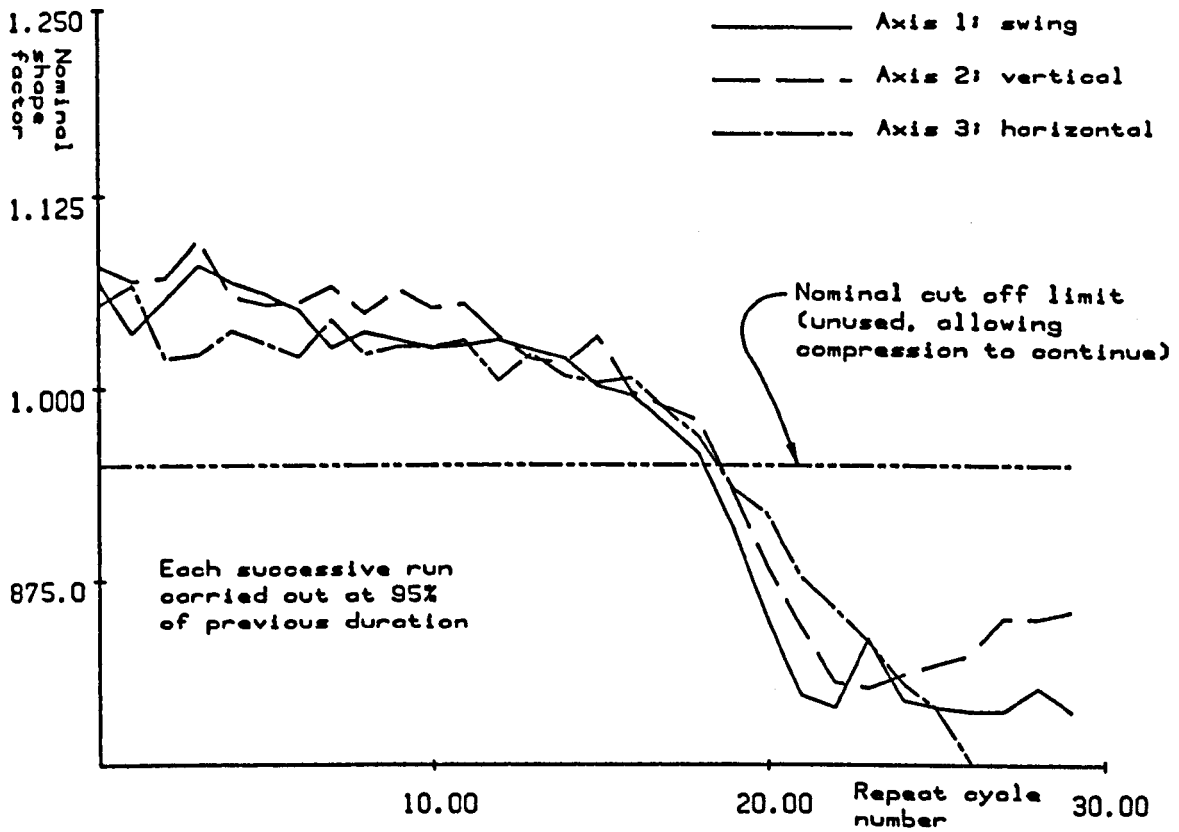
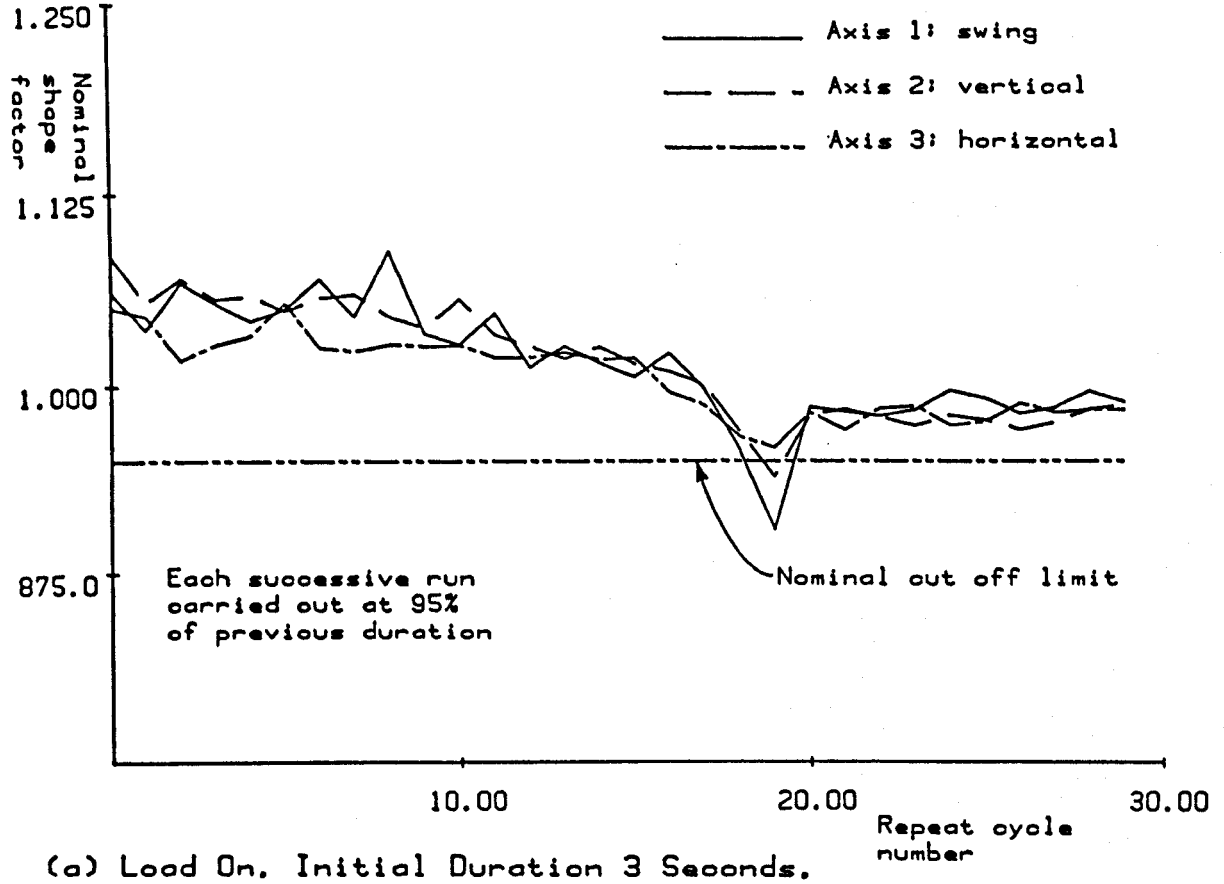
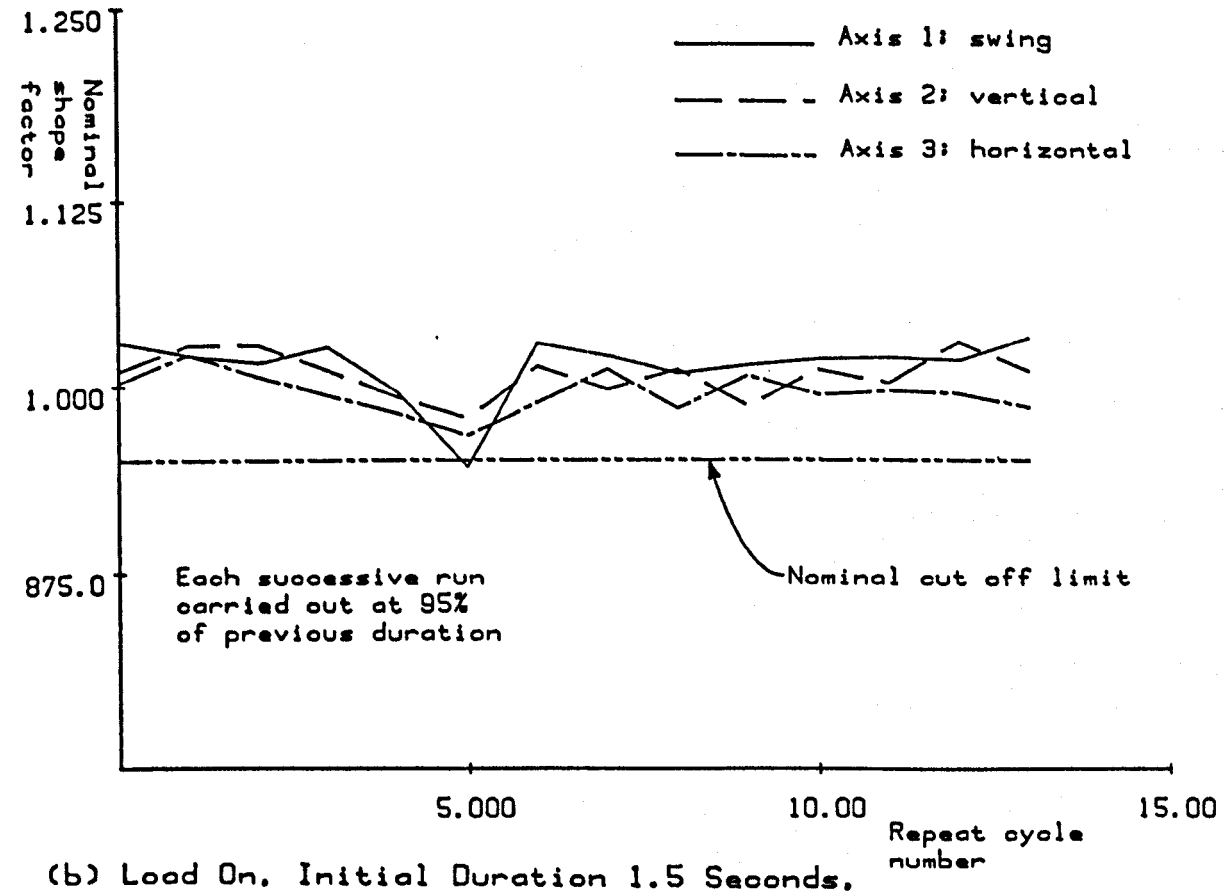


Figure 8.25 SPs as a Saturation Parameter



(a) Load On, Initial Duration 3 Seconds, 32 Cycles, Reduction 0.95



(b) Load On, Initial Duration 1.5 Seconds, 16 Cycles, Reduction 0.95

Figure 8.26 SPs Used for Automatic Cut Out

8.5.4 Definitions of the Saturation Parameters Tested

Based on the various options discussed, potential saturation quantifying parameters were computed as follows. More details are included in the module MODMVFAS.PAS, part of the program RUN.PAS in appendix B. The formal saturation parameter definitions are :

SP_1 The peak error for each axis during the motion :

$$peakerr_j = \left| r_{i,j,k} - y_{i,j,k} \right|_{max} \qquad (8.10)$$

SP_2 The average absolute error for each axis during the motion :

$$absumerr_j = \frac{\sum_{i=is,k}^{if,k} \left| r_{i,j,k} - y_{i,j,k} \right|}{if,k - is,k} \qquad (8.11)$$

SP_3 The ratio of previous average error to the current average error for each axis, during the motion :

$$satparam_j = SP_{2,k} \; / \; SP_{2,k+1} \qquad (8.12)$$

SP_4 The rolling average of SP_3

$$vecsat_j = (k-2)\, SP_3 \; / \sum_{k=3}^{kc} SP_3 \qquad (8.13)$$

SP_5 The average of SP_4 for all n axes :

$$scasat = \sum_{j=1}^{n} SP_4 \; / \; n \qquad (8.14)$$

SP_6 The normalised ratio of peak absolute response velocity to the average absolute velocity :

$$velshapfac_j = \frac{8 \left| \dot{y}_{j,k} \right|_{max} (if,k - is,k)}{15 \left| \sum_{i=is}^{if} y_{i,j,k} \right|} \qquad (8.15)$$

8.5.5 Maintenance of Tracking Accuracy.

As stated earlier, loss of tracking accuracy can have serious consequences. Not least of these may be that the motion segment directly after the one which has been tuned for speed, may require high tracking accuracy at its commencement. The scheme presented in chapter 6 was therefore applied to a motion segment subsequent to its pseudo optimisation for duration. Figure 8.27 shows the combined effects of using both schemes on a typical motion. Note that the peak errors, at the maximum speeds are of the same order as the static errors.

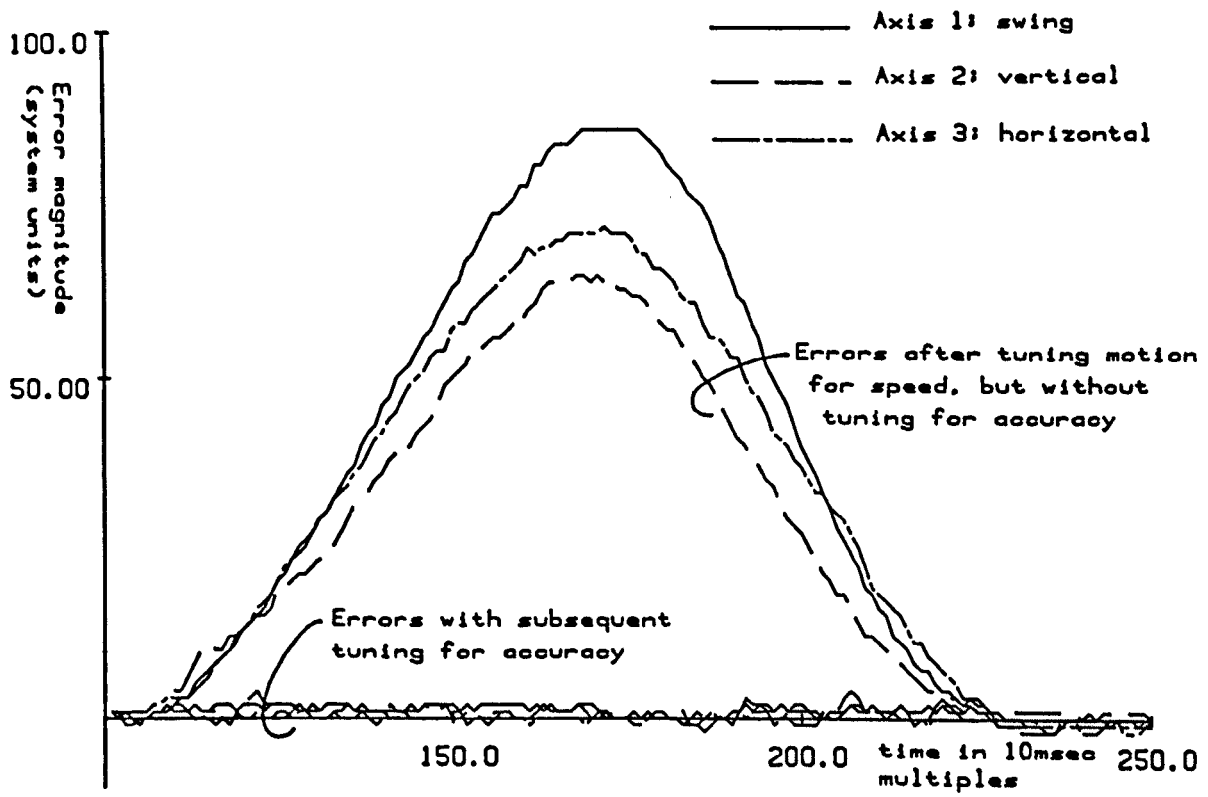


Figure 8.27 Tracking Errors, Rise Segment Only

The two systems proposed offer an improvement in tracking accuracy and speed. The philosophy is one which accepts there are faults in both the feedback controller and the motion generator. It could be argued that a dynamic model plus a high performance feedback control can eliminate these problems. Whether this can be achieved without resorting to high performance (expensive) computational hardware is another question. The computation in both the proposed schemes can be carried out off-line, when the robot is not moving. In low cost robots it may therefore be possible to achieve performance similar to that of high cost robots, if certain constraints on the robot cycle can be permitted.

9 Conclusions and Recommendations

For a robot to be viable in a manufacturing environment, it must provide a satisfactory degree of dynamic performance at an acceptable cost, amongst other criteria. If its performance can be improved without a substantial increase in cost, then the viability or saleability of the robot will increase.

The dynamic performance of robots, specifically the tracking accuracy and motion duration, is influenced by both the nominal motion profile and the feedback control method employed. An idealised scheme to optimise performance might include a complete dynamic model. It would then be used in a computation scheme to optimise the design of a feedback controller and to modify the motion profile for reduced tracking errors.

Model Based Motion Tuning

Conventional trajectory tuning methods require the substitution of the nominal command in place of the response derivatives in a dynamic model. The feasibility of this approach was considered. Dynamic parameter values are needed for the models. The more difficult dynamic parameter measurements are probably the link inertias and the actuator transfer functions. A set of axes about which inertia measurements can be taken is proposed. The advantages are that the transformation from the measurement inertial frame to the required frame is simple and well conditioned. Complete actuator system transfer functions were facilitated by the development of a scheme which readily and accurately identified the transfer function coefficients. These are too complex for their introduction into motion tuning systems and even though the accuracy achieved is to within ±½dB it is still not high enough.

Linearising the dynamics was found to be a lengthy process and doesn't appear to simplify the dynamic tuning or increase the potential accuracy.

Such model based tuning schemes seem to be fraught with difficulties; the measurement of the dynamic model parameters is a major task in

itself; dynamic models cannot be made sufficiently accurate and a large amount of computational power is required. Further simplifications to the dynamics make the tuning of motion easier, but tuning validity reduces as the accuracy of the model decreases. Single degree of freedom linear system tuning algorithms are derived, catering for any order differential equation and degree of polynomial.

The tuning of continuous motion laws based on linear dynamic models requires either complex analytic solutions, which are dedicated to a particular motion law, or an iterative and computationally intensive numerical solution. On the other hand, dynamic model based tuning for discrete motion laws and discrete dynamic models results in an explicit solution, which is simple to compute and not a function of the whole motion law, just its instantaneous value. This opens up a large range of possibilities for tuning motions in digital systems.

Three schemes have been developed and experimentally tested to tackle the improvement of dynamic performance, in the absence of accurate dynamic models.

MRACS Model Based Tuning of Motion

Model Referenced Adaptive Controller Schemes (MRACS) can be designed to facilitate the characterisation of otherwise complex system dynamics. In one of the three schemes an MRACS attempted to force the robot to behave as if it were linear and decoupled, enabling simple model based dynamic tuning methods to be applied to the motion laws. Its promise as a technique was demonstrated, but the controller performance was degraded by practical limitations.

Initially in a single axis implementation, it was shown that an MRACS can be set up to force a system to behave as desired, without significant modification to the controller, merely the parameters of the model it tracks. Landau's design method was selected as a building block. The main benefit of the MRACS model is the designer can arbitrarily specify the dynamics (within actuator saturation limitations), and hence use them in motion tuning.

Compared to the self learning systems, the MRACS motion tuning scheme potentially offers better performance as it should be less sensitive to parameter variations and disturbances, because it is active in real time. This was not verified in the actual system tests.

The MRACS motion tuning sub-system functioned perfectly, with zero errors between the nominal motion and the model response. The scheme has been demonstrated with both joint and world (Cartesian) based motion laws.

The MRACS itself was unable to hold the robot response to the desired characteristics, even though model time constants were longer than necessary. The reasons for this are probably the sample interval limitation due to the computer hardware; system gain limitations, related to the sample interval; the differential pressure feedback signal introducing further lags because of the necessarily heavily slugged signal filtering; and finally the velocity estimate (based on displacement differencing) was insufficiently accurate.

Self Learning Tuning of Motion

A computer controlled robot contains all the elements necessary for an autonomous self experimentation system. This feature was exploited in the derivation and implementation of two further schemes which are termed self learning. In these, the robot's trajectory was stored as a set of discrete data. Algorithms were developed for tuning this data subsequent to each run. Additional costs are minimal, comprising a small amount of extra memory required for motion data tables. Use of the algorithms requires minimal knowledge of the dynamics, is largely independent of the particular robot, requires no additional transducers and the small amount of extra computation needed can be carried out off-line if required.

The first of the self learning schemes utilised the fact that if a robot motion is dynamically repeatable, previous response data can be used to null out errors before they arise. The algorithm was used to cyclically improve the tracking errors. Once reduced, the updating process can be curtailed. Should the process be subject to parameter variations, the process can be retained.

The self learning scheme for improving tracking accuracy was applied successfully, first in a simulation, then on a DC servomotor and load with significant backlash and Coulomb friction, followed by a DC servomotor set up to have a highly oscillatory response, and finally the three axis hydraulic robot with varying load and motion duration. The number of cycles required to achieve completion of the motion tuning varied typically between 12 and 15. The error bounds achieved dynamically were in many cases as low as the static errors. A low period ratio motion (≈2) was also accommodated, yielding substantially reduced tracking errors.

The second of these schemes involved an incremental reduction in the duration of a given motion. Various parameters for detecting saturation were proposed and tested. A normalised ratio of peak to average velocity was shown to be promising.

Optimisation of motion duration is conventionally carried out using dynamic models. Simulations yield good results because the parameters are exact and pre-defined, and the simple actuator models used such as scalar torque and speed limits simplify the optimisation. In a real implementation many problems would arise, not least of which would be parameter measurement and model verification. The scheme developed in this project operated within the actual limitations of the robot, which are very difficult to define precisely within a model. Once the limitations are reached, the response becomes saturated. The problem is to quantify this saturation, so its onset can be reliably detected before damage is done to the robot or its environment. A number of potential saturation quantifying parameters were tested. A normalised ratio of peak to average response velocity was found to satisfy most of the requirements for such a quantity. At low speed it remained at a constant value of around unity. Its value dropped sharply as the velocity response distorted. Other advantages of the parameter are that it is relatively insensitive to displacement repeatability variations; and the peak velocity tends to occur in the region which saturates first.

Combining these two schemes, tuning for speed to near saturation then tuning for accuracy, provides a method for obtaining a near minimum time trajectory, with maximum possible tracking accuracy, at low cost.

Motion Design and Generation

As part of the development of the scheme, a complete manual motion generation scheme has been developed and implemented as software. It gives the facility to generate multi-axis, multi-segment polynomial motions, of degree 1 to 9. All the practical boundary condition combinations are available and it caters for any number of axes and segments.

A semi-automatic version has also been constructed, capable of replacing the functions of the manual scheme, allowing high level control of the motion. This enabled the development and testing of the motion modification algorithms in a more representative environment and reduced the effort involved, in specifying motions.

Vibration responses of the robot, when driven by several polynomial laws indicated that the fifth degree offered the minimum vibration levels. Comparison of the properties of the laws indicated it is the simplest of the laws studied not to contain a step in the acceleration curve. It also has a low peak velocity and acceleration. Further, the fifth degree law is versatile in that control over boundary conditions is available at up to acceleration. Generic conclusions about the degree of law and its interaction with sample interval, period ratio and motion duration cannot be made as the error tracking performance varies widely depending on the precise values of each of the aforementioned parameters. Broadly speaking the fifth degree, if it is bettered by other laws for certain of these parameter values, it is not by much. It is concluded therefore that the fifth degree is the best, all round polynomial motion law, for use in this scheme.

Further Work

Polynomial motion laws, and for that matter most mathematical functions used for motion laws are found wanting. Although boundary conditions can be used to control the motion shape, it is only to a small extent. If laws were implemented entirely as discrete data tables, then the motion shape could be guaranteed. For example monotonicity and a lack of inflexions could be assured. Properties of the law could be defined in advance, then the law would be constructed

based on them, rather than in the case of the mathematical function used as a law, where the properties are a consequence of the law.

A discrete data table system capable of matching arbitrary boundary conditions would be a substantial task. An alternative might be to produce a filter which would process mathematically based functions used as motion laws, and eliminate the undesirable features.

The MRACS controller requires a faster computer in order to achieve its full potential. The coefficients of the gain matrices used within the adaptor do not appear to be arbitrary. Some work should be devoted to establishing an optimum analytic solution to them.

The self learning algorithm for increasing accuracy requires a stability analysis, to be used safely in more general systems. The setting of target error bounds for the scheme required some experimentation. This could be built into the scheme, by automatically measuring the robot's repeatability before commencing tuning.

The self learning scheme for reducing motion duration requires a more suitable saturation parameter. This is needed for use with more general cases (i.e. other than single segment, stop-go-stop motions).

Both of the self learning schemes should be tested on other robots and types of machinery. Many manufacturing processes use machines with much shorter cycle times than robots, with a highly repetetive cycle content, making self learning schemes even more attractive.

[2.1] DUFFY, J. Analysis of mechanisms and robot manipulators. Wiley, New York, 1980

[2.2] PIEPER, D.L. The kinematics of manipulators under computer control. PhD Thesis, Computer Sciences Department, Stanford University, 1968.

[2.3] DENAVIT, J. and HARTENBERG, R.S. A kinematic notation for lower-pair mechanisms based on matrices. ASME Journal of Applied Mechanics, vol. 22, pp215-221, June 1955.

[3.1] STODDARD, D.A. Polydyne cam design. Parts I, II and III, A symposium on internal combustion engine valves, Cleveland Electric Printing Company, pp177-216, 1953.

[3.2] SHILLER, Z. and DUBOWSKY, S. On the optimal control of robotic manipulators with actuator and end effector constraints. Proceedings of the IEEE International Conference on Robotics and Automation, pp614-620, St. Lois, MO., March 1985.

[3.3] KAHN, M.E. and ROTH, B. The near minimum-time control of open loop articulated kinematic chains, ASME Journal of Dynamic Systems, Measurement and Control, pp164-172, September 1971.

[3.4] PAUL, R.P.C. Modelling, trajectory calculation and servoing of a computer controlled arm. Stanford Artificial Intelligence Laboratory, Report AIM 177, 1972.

[3.5] LUH, J.Y.S., WALKER, M.W. and PAUL, R.P.C. On-line computational scheme for mechanical manipulators. ASME Journal of Dynamic Systems, Measurement and Control, vol.102, pp69-76, June 1980.

[3.6] HOLLERBACH, J.M. A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. IEEE Trans. on Systems, Man and Cybernetics, vol.SMC-10, No. 11, pp730-736, November 1980.

[3.7] HORAK, D.T. A simplified modeling and computational scheme for manipulator dynamics. ASME Journal of Dynamic Systems, Measurement and Control, vol.106, pp350-353, December 1984.

[3.8] WALKER, M.W. and ORIN, D.E. Efficient dynamic computer simulation of robotic mechanisms. ASME Journal of Dynamic Systems, Measurement and Control, vol.104, pp205-211, 1982.

[3.9] WHITNEY, D.E. Resolved motion rate control of manipulators and human prostheses. IEEE Trans. On Man-Machine Systems, vol.MMS-10,

No.2, 1969.

[3.10] GOLLA, D.F., GARG, S.C. and HUGHES, P.C. Linear state-feedback control of manipulators. Mechanism and Machine Theory, vol.16, pp93-103, 1981.

[3.11] GOOR, R.M. A new approach to minimum time robot control. Research Publication GMR-4869, General Motors Research Laboratories, Warren, Michigan, November 1984.

[3.12] HEWIT, J.R. and BURDESS, J.S. Fast dynamic decoupled control for robotics, using active force control. Mechanism and Machine Theory, vol.16, No.5, pp535-542, 1981.

[3.13] GALATIS, G., HADZISTYLIS, A. and HEWIT, J.R. Experimental investigation of active force control of robot and manipulator arms. Proceedings of the 6th CISM-IFToMM Symposium, Ro.man.sy - 86, Cracow, Poland, September 1986.

[3.14] BEJCZY, A.K. Robot arm dynamics and control. Technical memorandum 33-669, Jet Propulsion Laboratory, February 1974.

[3.15] LIEGEOIS, A., FOURNIER, A. and ALDON, M.J. Model reference control of high velocity industrial robots. Proceedings of the 1980 Joint Automatic Control Conference, San Francisco, 1980.

[3.16] DUBOWSKY, S. and DESFORGES, D.T. The application of model referenced adaptive control to robotic manipulators. ASME Transactions, Journal of Dynamic Systems, Measurement and Control, vol.101, pp193-200, 1979.

[3.17] DONALSON, D.D and LEONDES, C.T. A model referenced parameter tracking technique for adaptive control systems. Transactions of the IEEE on Applications and Industry, vol.82, pp241-262, 1963.

[3.18] LANDAU, I.D. Adaptive Control. Dekker, 1979.

[3.19] HOROWITZ, R. and TOMIZUKA, M. An adaptive control scheme for mechanical manipulators - compensation for non-linearity and decoupling control. Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control, ASME paper No. 80-Wa/DSC-6, 1980.

[3.20] STOTEN, D.P. The adaptive control of manipulator arms. Department of Mechanical Engineering, University of Liverpool, Technical Report A/050/80, 1980.

[4.1] BRADY, M. Trajectory planning. Robot motion - planning and control, The MIT press, 1982.

[4.2] TAYLOR, R.H. Planning and execution of straight line manipulator trajectories. IBM Journal of Research and Development, vol.23, No.4, 1979.

[4.3] PAUL, R. Robot manipulators - mathematics, programming and control. The MIT press, 1981.

[4.4] KWAKERNAAK, H. and SMIT, J. Minimum vibration cam profiles. Journal of Mechanical Engineering Science, vol.10, No.3, 1968.

[4.5] MUTJABA, M.S. Discussion of trajectory calculation methods. Within: Exploratory study of computer integrated assembly systems. BINFORD, T.O. et al, Artificial Intelligence Laboratory report AIM 285.4, Stanford University, 1977.

[4.6] VUKOBRATOVIC, M. and KIRCANSKI, M. One method for simplified manipulator model construction and its application in quasi-optimal trajectory synthesis. Mechanism and Machine Theory, vol.17, No.6, pp369-378, 1982.

[4.7] REES JONES, J. A comparison of the dynamic performance of tuned and non-tuned multiharmonic cams. Proceedings of the 2nd IFToMM International Symposium on linkages and computer aided design methods, SYROM'77, vol.III-2, pp553-564,Bucharest, Romania, 1977.

[4.8] REES JONES, J. et al. Optimum trajectory generation and control of manipulators. Final report: SERC Grant GR/B/51840, Mechanisms and Machines Group, Liverpool Polytechnic, 1984.

[5.1] HOLLERBACH, J.M. Dynamic scaling of manipulator trajectories. Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, vol.106, pp102-106, March 1984.

[5.2] ASADA, H. The kinematic design and mass redistribution of manipulator arms for decoupled and invariant inertia. Proceedings of the 6th CISM-IFToMM Symposium, Ro.man.sy - 86, Cracow, Poland, September 1986.

[5.3] AN, C.H., ATKESON, J.D., GRIFFITHS, J.D. and HOLLERBACH, J.M. Experimental evaluation of feedforward and computed torque control. Proceedings of the 6th CISM-IFToMM Symposium, Ro.man.sy - 86, Cracow, Poland, September 1986.

[5.4] FEATHERSTONE, R. The simulator verification experiment. Department of Artificial Intelligence working paper, No. 178, University of Edinburgh, April 1985.

[5.5] BAYSEC, S. Simulation of the dynamics of hydraulically actuated planar manipulators. PhD Thesis, Department of Mechanical, Marine and Production Engineering, Liverpool Polytechnic, 1983.

[5.6] ATKESON, C.G., AN, C.H. and HOLLERBACH, J.M. Estimation of inertial parameters of manipulator loads and links. Proceedings of the

3rd International Symposium on Robotics Research, Gouvieux (Chantilly), France, October 1985.

[5.7] CRAIG, J.J. Adaptive control of manipulators through repeated trials. Proceedings of the Joint Automatic Control Conference, pp1566-1573, paper FP2, 1984.

[5.8] ARIMOTO, S., KAWAMURA, S. and MIYAZAKI, F. Bettering operation of dynamic systems by learning : A new control theory for servomechanism or mechatronics systems. Proceedings of the 23rd IEEE Conference on Decision and Control, paper TP4, pp1064-1069, Las Vegas, NV, December 1984.

[5.9] ARIMOTO, S., KAWAMURA, S. and MIYAZAKI, F. Can mechanical robots learn by themselves ? Proceedings of the 2nd International Symposium of Robotics Research, Kyoto, Japan, August 1985.

[6.1] SAHIRAD, M., RISTIC, M. and BESANT, C.B. Implementation of adaptive positional control and active compliance for robots. Proceedings of the Institution of Mechanical Engineers - UK Research in Advanced Manufacture, 10-11 December 1986.

[6.2] VERNON, G.W., REES JONES, J. and ROONEY, G.T. Dynamic command motion tuning for robots:- a self learning algorithm. 6th CISM - IFToMM Symposium, Ro.man.sy - 86, Cracow, Poland, September 1986.

[7.1] ERZBERGER, H. Analysis and design of model following control systems by state space techniques. Proceedings of the Joint Automatic Control Conference, pp572-581, Ann Arbor, Michigan, 1968.

[7.2] WINSOR, C.A. and ROY, R.J. The application of specific optimal control to design of desensitised model following control systems. IEEE Transactions on Automatic Control, paper AC-15, pp326-333, 1970.

[7.3] BUNDELL, G.A. Robust decentralized model-reference adaptive control in the manipulator control problem. Internal report, Division of Information Engineering, Department of Engineering, University of Cambridge, 1985.

[8.1] SHIN, K.G. and McKAY, N.D. Minimum time control of robotic manipulators with geometric path constraints. IEEE Transactions On Automatic Control, vol. AC-30, No.6, June 1985.

[8.2] SEEGER, G.H. and PAUL, R.P. Optimizing robot motion along a predefined path. Proceedings of the IEEE International Conference on Robotics and Automation, pp765-770, St. Lois, MO., March 1985.

[8.3] SAHAR, G. and HOLLERBACH, J.M. Planning of minimum time trajectories for robot arms. MIT Artificial Intelligence Laboratory Memo No. 804, November 1984.

[8.4] VERNON, G.W., REES JONES, J. and ROONEY, G.T. Automatically increasing the speed of industrial robots and multi-axis machines. Proceedings of the Institution of Mechanical Engineers - UK Research in Advanced Manufacture, 10-11 December 1986.

A1 Matrix equation solutions to polynomial coefficients

The 15 matrix equations required for the sets of boundary conditions in table 4.3, result in the following coefficient solutions :

1 $c_0 = d_1$

$c_1 = (d_2 - d_1)/T$

2 $c_0 = d_1$

$c_1 = v_1$

$c_2 = (d_2 - d_1 - v_1 T)/T^2$

3(i) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = (2(d_2 - d_1) - 2v_1 T - a_1 T^2)/(2T^3)$

3(ii) $c_0 = d_1$

$c_1 = v_1$

$c_2 = (3(d_2 - d_1) - (2v_1 + v_2)T)/T^2$

$c_3 = (2(d_1 - d_2) + (v_1 + v_2)T)/T^3$

4(i) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = j_1/6$

$c_4 = (6(d_2 - d_1) - 6v_1 T - 3a_1 T^2 - j_1 T^3)/(6T^4)$

4(ii) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = (4(d_2 - d_1) - (3_1 + v_2)T - a_1 T^2)/T^3$

$c_4 = (6(d_1 - d_2) + (4v_1 + 2v_2)T + a_1 T^2)/(2T^4)$

5(i) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = j_1/6$

$c_4 = i_1/24$

$c_5 = (24\,(d_2 - d_1) - 24v_1T - 12a_1T^2 - 4j_1T^3 - i_1T^4)/(24T^5)$

5(ii) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = j_1/6$

$c_4 = (5\,(d_2 - d_1) - (4v_1 + v_2)T - 3a_1T^2/2 - j_1T^3/3)/T^4$

$c_5 = (4\,(d_1 - d_2) + (3v_1 + v_2)T + a_1T^2 + j_1T^3/6)/T^5$

5(iii) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = (20\,(d_2 - d_1) - (12v_1 + 8v_2)T + (a_2 - 3a_1)T^2)/(2T^3)$

$c_4 = (30\,(d_1 - d_2) + (16v_1 + 14v_2)T + (3a_1 - 2a_2)T^2)/(2T^4)$

$c_5 = (12\,(d_2 - d_1) - 6(v_1 + v_2)T + (a_2 - a_1)T^2)/(2T^5)$

6(i) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = j_1/6$

$c_4 = i_1/24$

$c_5 = (72\,(d_2 - d_1) - (60v_1 + 12v_2)T - 24a_1T^2 - 6j_1T^3 - i_1T^4)/(12T^5)$

$c_6 = (120\,(d_1 - d_2) + (96v_1 + 24v_2)T + 36a_1T^2 + 8j_1T^3 + i_1T^4)/(24T^6)$

6(ii) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = j_1/6$

$c_4 = (30\,(d_2 - d_1) - (20v_1 + 10v_2)T + (a_2 - 6a_1)T^2 - j_1T^3)/(2T^4)$

$c_5 = (48\,(d_1 - d_2) + (30v_1 + 18v_2)T + (8a_1 - 2a_2)T^2 + j_1T^3)/(2T^5)$

$c_6 = (60\,(d_2 - d_1) - (36v_1 + 24v_2)T + (3a_2 - 9a_1)T^2 - j_1T^3)/(6T^6)$

7(i) $c_0 = d_1$

$c_1 = v_1$

$c_2 = a_1/2$

$c_3 = j_1/6$

$c_4 = i_1/24$

$$c_5 = (168\,(d_2 - d_1) - (120v_1 + 48v_2)T + (4a_2 - 40a_1)T^2 - 8j_1T^3 - i_1T^4)/(8T^5)$$

$$c_6 = (840\,(d_1 - d_2) + (576v_1 + 264v_2)T + (180a_1 - 24a_2)T^2 + 32j_1T^3 + 3i_1T^4)/(24T^6)$$

$$c_7 = (360\,(d_2 - d_1) - (240v_1 + 120v_2)T + (12a_2 - 72a_1)T^2 - 12j_1T^3 - i_1T^4)/(24T^7)$$

7(ii)

$$c_0 = d_1$$

$$c_1 = v_1$$

$$c_2 = a_1/2$$

$$c_3 = j_1/6$$

$$c_4 = (210\,(d_2 - d_1) - (120v_1 + 90v_2)T + (15a_2 - 30a_1)T^2 - (4j_1 + j_2)T^3)/(6T^4)$$

$$c_5 = (168\,(d_1 - d_2) + (90v_1 + 78v_2)T + (20a_1 - 14a_2)T^2 + (2j_1 + j_2)T^3)/(2T^5)$$

$$c_6 = (420\,(d_2 - d_1) - (216v_1 + 204v_2)T + (39a_2 - 45a_1)T^2 - (4j_1 + 3j_2)T^3)/(6T^6)$$

$$c_7 = (120\,(d_1 - d_2) + (60v_1 + 60v_2)T + (12a_1 - 12a_2)T^2 + (j_1 + j_2)T^3)/(6T^7)$$

8

$$c_0 = d_1$$

$$c_1 = v_1$$

$$c_2 = a_1/2$$

$$c_3 = j_1/6$$

$$c_4 = i_1/24$$

$$c_5 = (336\,(d_2 - d_1) - (210v_1 + 126v_2)T + (18a_2 - 60a_1)T^2 - (10j_1 + j_2)T^3 - i_1T^4)/(6T^5)$$

$$c_6 = (1680\,(d_1 - d_2) + (1008v_1 + 672v_2)T + (270a_1 - 102a_2)T^2 + (40j_1 + 6j_2)T^3 + 3i_1T^4)/(12T^6)$$

$$c_7 = (720\,(d_2 - d_1) - (420v_1 + 300v_2)T + (48a_2 - 108a_1)T^2 - (15j_1 + 3j_2)T^3 - i_1T^4)/(6T^7)$$

$$c_8 = (840\,(d_1 - d_2) + (480v_1 + 360v_2)T + (120a_1 - 60a_2)T^2 + (16j_1 + 4j_2)T^3 + i_1T^4)/(24T^8)$$

9

$$c_0 = d_1$$

$$c_1 = v_1$$

$$c_2 = a_1/2$$

$$c_3 = j_1/6$$

$$c_4 = i_1/24$$

$$c_5 = (3024\,(d_2 - d_1) - (1680v_1 + 1344v_2)T + (252a_2 - 420a_1)T^2 - (60j_1 + 24j_2)T^3 + (i_2 - 5i_1)T^4)/(24T^5)$$

$$c_6 = (5040\,(d_1 - d_2) + (2688v_1 + 2352v_2)T + (630a_1 - 462a_2)T^2 + (80j_1 + 46j_2)T^3 + (5i_1 - 2i_2)T^4)/(12T^6)$$

$$c_7 = (6480\,(d_2 - d_1) - (3360v_1 + 3120v_2)T + (636a_2 - 756a_1)T^2 - (90j_1 + 66j_2)T^3 + (3i_2 - 5i_1)T^4)/(12T^7)$$

$$c_8 = (7560\,(d_1 - d_2) + (3840v_1 + 3720v_2)T + (840a_1 - 780a_2)T^2 + (96j_1 + 84j_2)T^3 + (5i_1 - 4i_2)T^4)/(24T^8)$$

$$c_9 = (1680\,(d_2 - d_1) - (840v_1 + 840v_2)T + (180a_2 - 180a_1)T^2 - (20j_1 + 20j_2)T^3 + (i_2 - i_1)T^4)/(24T^9)$$

These solutions are incorporated into the manual motion generation system, in the program POLY.PAS. This program is listed in appendix B.

A2 Solution to one case of the Lyapunov equation

If equation 7.33 is algebraically computed for arbitrary non-zero P_{ij} and Q_{ii} elements ($\underset{\sim}{Q}$ is diagonal) then 36 equations in 36 unknowns are obtained. These are very simple in form, it is easy to show that the equations are :

$$\begin{bmatrix} S(12\times12) & & 0 \\ & S(12\times12) & \\ 0 & & S(12\times12) \end{bmatrix} \begin{bmatrix} P_1(6\times1) \\ P_2(6\times1) \\ P_3(6\times1) \\ P_4(6\times1) \\ P_5(6\times1) \\ P_6(6\times1) \end{bmatrix} = \begin{bmatrix} Q_1(6\times1) \\ Q_2(6\times1) \\ Q_3(6\times1) \\ Q_4(6\times1) \\ Q_5(6\times1) \\ Q_6(6\times1) \end{bmatrix}$$

clearly the three sets are decoupled. The individual matrices are :

$$\underset{\sim}{P_i} = \begin{bmatrix} P_{i1} \\ P_{i2} \\ P_{i3} \\ P_{i4} \\ P_{i5} \\ P_{i6} \end{bmatrix} \qquad \underset{\sim}{Q_i} = \begin{bmatrix} -Q_{i1} \\ -Q_{i2} \\ -Q_{i3} \\ -Q_{i4} \\ -Q_{i5} \\ -Q_{i6} \end{bmatrix}$$

and

$$\underset{\sim}{S} = \left[\begin{array}{ccc|ccc} \underset{\sim}{M_1} & & 0 & & & \\ & \underset{\sim}{M_1} & & & \underset{\sim}{I_6} & \\ 0 & & \underset{\sim}{M_1} & & & \\ \hline & & & \underset{\sim}{M_2} & & 0 \\ & -\sigma_1 \underset{\sim}{I_6} & & & \underset{\sim}{M_2} & \\ & & & 0 & & \underset{\sim}{M_2} \end{array}\right]$$

where I_6 is the (6x6) identity matrix, and :

$$\underset{\sim}{M}_1 = \begin{bmatrix} 0 & -\sigma_1 \\ 1 & -\sigma_2 \end{bmatrix} \qquad \underset{\sim}{M}_2 = \begin{bmatrix} -\sigma_1 & -\sigma_1 \\ 1 & -2\sigma_2 \end{bmatrix}$$

The sets of equations can be solved individually or alternatively matrix methods can be used for the inverse $\underset{\sim}{S}$ matrix. The latter has some advantages as the three equation sets are otherwise not identical. The solution is given by :

$$\underset{\sim}{S}^{-1} = \left[\begin{array}{ccc|ccc} \underset{\sim}{M}_3 & & 0 & \underset{\sim}{M}_6 & & 0 \\ & \underset{\sim}{M}_3 & & & \underset{\sim}{M}_6 & \\ 0 & & \underset{\sim}{M}_3 & 0 & & \underset{\sim}{M}_6 \\ \hline \underset{\sim}{M}_5 & & 0 & \underset{\sim}{M}_4 & & 0 \\ & \underset{\sim}{M}_5 & & & \underset{\sim}{M}_4 & \\ 0 & & \underset{\sim}{M}_5 & 0 & & \underset{\sim}{M}_4 \end{array}\right]$$

and

$$\underset{\sim}{M}_3 = \begin{bmatrix} -(\sigma_2/\sigma_1 + 1/\sigma_2) & 1/2 \\ -1/(2\sigma_1) & -1/(2\sigma_2) \end{bmatrix} \qquad \underset{\sim}{M}_4 = \begin{bmatrix} -1/(2\sigma_2) & 0 \\ 0 & -1/(2\sigma_2) \end{bmatrix}$$

$$\underset{\sim}{M}_5 = \begin{bmatrix} 1/2 & -\sigma_1/(2\sigma_2) \\ 1/(2\sigma_2) & 0 \end{bmatrix} \qquad \underset{\sim}{M}_6 = \begin{bmatrix} -1/(2\sigma_1) & 1/(2\sigma_2) \\ -1/(2\sigma_1\sigma_2) & 0 \end{bmatrix}$$

It is clear that the patterns present in the solution may be extrapolated to cover the 'n' axis case. Solutions for the $\underset{\sim}{P}$ matrix elements are :

$$\underset{\sim}{P}(6x6) = \begin{bmatrix} \bar{P}_1 & & 0 \\ & \bar{P}_3 & \\ 0 & & \bar{P}_5 \end{bmatrix} \qquad (7.34)$$

where

$$\underset{\sim}{\bar{P}}_i = 1/2 \begin{bmatrix} (\sigma_2/\sigma_1 + 1/\sigma_2)Q_{i\,i} - Q_{i+1,\,i+1}/\sigma_2 & Q_{i\,i}/\sigma_1 \\ -Q_{i\,i} & (Q_{i+1,\,i+1} - Q_{i\,i})/\sigma_2 \end{bmatrix} \qquad (7.35)$$

for i = 1,3,5,.....,n

A3 Linear interpolation for motion duration reduction

Options for interpolation include the use of finite differences, e.g. Newton-Gregory, Bessel or Everett for Forward, Backward or Central Differences. Central are probably the most relevant, using fewer terms and giving more accuracy. Computation rises quickly with the number of terms used and hence accuracy. Bessel's Central Differences formula requires around 11 multiplications and 14 additions for the first four terms.

Polynomial data fits are also used for interpolation, but again computation is significant. Polynomial fit coefficients must be computed for every interpolation. Lagrange's three point interpolation for example, requires 9 multiplications and 14 additions. Linear (first order) interpolation is therefore worthy of further study. It requires minimal computation, but is it sufficiently accurate ?

Considering the fifth degree symmetric and normalised law, the peak error between an exact solution and a linear interpolation will occur in the region of peak curvature. This occurs for maximum second derivative values, i.e. acceleration in our case. The peak accelerations for the above law arise at :

$$t_{peak} = \frac{1}{2} \pm \frac{1}{\sqrt{(12)}}$$

The law itself is defined by :

$$r(t) = 10t^3 - 15t^4 + 6t^5$$

Assuming there are to be say 50 samples in the interval [0,1] then taking as the region to be interpolated :

$$t_o,\ t_o+\delta t = t_{peak} \pm 0.01$$

will cause the error due to interpolation to be a maximum. For the lower t_{peak} value, the error at the centre of the region will be given by :

$$e = r(t_o) + \tfrac{1}{2}[r(t_o + \delta t) - r(t_o)] - r(t_o + \delta t/2)$$
$$= 2.89 \times 10^{-4}$$

If for example, the motion described represents 1000 encoder units, then the error will be less than one unit. Further, the interpolation would not normally be carried out at the centre of a sample, but close to one end of it. It still remains to check the compounding effect of errors, when motions are repeatedly compressed in time, by interpolation. A practical scheme for motion interpolation is therefore presented. Defining the motion and its 'compression' parameters as follows :

T	motion duration
R	reduction factor - change of duration
n_{old}	old number of data points defining the motion
n_{new}	new number of data points defining the motion
i_{old}	sample number in old data set
i_r	sample number in old data set (floating point value)
i_{new}	sample number in new data set
i_s	sample number at start of motion segment
i_f	sample number at finish of motion segment

The new number of samples per axis, required to represent the motion is defined by :

$$n_{new} = \text{trunc}\ (R \times n_{old})$$

Considering an arbitrary proportion of the motion, then to arrive at the same displacement value requires that :

$$\frac{inew - is}{nnew} = \frac{iold - is}{nold}$$

The sample number inew in the new set will vary from is to is + nnew. In order to compute r(inew) it is necessary to locate the appropriate point in the old trajectory. The above equation yields a real (not integer) solution for iold. In practice iold may be derived from :

$$ir = \frac{(nold - 1)(inew - is)}{nnew - 1} + is$$

This is not an integer, and it lies in the interval :

$$iold \leq ir \leq iold + 1$$

so the correct sample interval has been located on which the interpolation is to be based. The -1 offsets are required to give correct solutions at the ends of the motion. The proportion along the sample is defined by :

$$ir - iold$$

Hence the full, linearly interpolated value of the trajectory is given by :

$$rnew(inew) = trunc\{[rold(iold + 1) - rold(iold)](ir - iold) + rold(iold) + 0.5\}$$

The truncation is required to represent the trajectory in integer form. As the trajectory is repeatedly interpolated, banker's rounding is incorporated such that accumulative errors will tend to average out.

Testing this interpolation algorithm for a typical trajectory, interpolated six times and comparing with the exact function evaluation gave maximum errors of 1 or 2 units over full 12 bit resolution. The errors occur in the dynamic regions of motion, as expected. In these regions, position tracking accuracy is at its lowest anyway. These errors are therefore considered negligible.

B Software listings (on diskette, inside back cover)

The programs listed are :

MO.BAT - a batch file which runs the manual motion generation scheme and passes file names between the different programs.

POLY.PAS - permits the specification of motions including the entry of a wide range of boundary conditions, the selection of polynomial degree, the number of axes to be driven and the number of motion segments per axis. The coefficient solutions are in-built, as are some facilities such as cascading of boundary conditions through all the axis segments to reduce the amount of data entry.

GENMO.PAS - generates the motion as specified by POLY.PAS. The command structure used by GAL.SA and RUN.PAS can be entered into the trajectory data file, making the manual and semi-automatic trajectory data files identical.

RUN.PAS - receives the trajectory data file, interprets the commands within it and executes the real time motion generation. It is comprised of a further four modules :

MODREAD.PAS - reads the setup data, initialises variables, transforms the motion data into the required form.

MODRTIME.PAS - carries out the real time motion generation and command interpretation.

MODMVACC.PAS - is the scheme for cyclically improving the trajectory tracking accuracy.

MODMVFAS.PAS - is the scheme for cyclically reducing the trajectory duration subject to saturation constraints.

TEACH.SA - enables the rapid entry of robot coordinate data, for use within the GAL.SA motion generating language. The coordinate values are assigned to ASCII string identifiers which are easier to handle. The coordinate data can be entered as transducer, joint or world values. Within TEACH.SA are two further modules :

KINSUB.SA - this module comprises the kinematic transformations which are required for use in TEACH.SA.

FORCE.SA (provided by Motorola) - the VME10 sees the keyboard as

a terminal, which creates some difficulties with the input from the keyboard. FORCE.SA is a patch to overcome this problem.

GAL.SA - is the semi-automatic motion generation system compiler.

TEST1 to TEST12 - examples of the text files used as input to GAL.SA.

MRACS.PAS - the stand alone motion generation, tuning and control program used for the MRACS scheme.

WMRACS.PAS - the Cartesian motion generating pre-processor for use with MRACS.PAS.

C Publications

Pocket, on inside back cover :

VERNON, G.W., REES JONES, J. and ROONEY, G.T. Dynamic command motion tuning for robots:- a self learning algorithm. 6th CISM - IFToMM Symposium, Ro.man.sy - 86, Cracow, Poland, September 1986.

VERNON, G.W., REES JONES, J. and ROONEY, G.T. Automatically increasing the speed of industrial robots and multi-axis machines. Proceedings of the Institution of Mechanical Engineers - UK Research in Advanced Manufacture, 10-11 December 1986.

ERRATA G W Vernon 26th June 1988

Appendix - For practical reasons, software listings are not included in the text. For further information, contact G W Vernon, c/o Prof. J Rees Jones, Mechanisms and Machines Group, Liverpool Polytechnic, Byrom St., Liverpool, L3 3AF.

p26 Line between equations 2.50 and 2.51 should read :

"and equation 2.11 gives :"

p43 The last term in equation 3.4 should read :

"+ $\underset{\sim}{p}_i^* \times \underset{\sim}{f}_{i+1}$"

p69 Item (ix) should read :

(ix) The robot initialisation and calibration must be in-built."

p80 Line after equations 4.28 should read :

"These conditions are conflicting...."

p102 Section 5.3.3, first para, last sentence, should read :

"This is found to be reasonable, given that its elimination can be achieved by ensuring the initial conditions are correctly computed."

p106 Equation 5.49 - the derivative dots above the r and y should be omitted.

p111 Table 5.15. The peak jerk for a cubic polynomial should read as ∞ not 12.

p116 Equation 6.8. The relative magnitudes of the changes in x, y and r need comparison before the change in x (between runs) can be neglected.

p117 Equation 6.15. A necessary condition for the system to operate is that the input and response vectors, r and y are scaled to each other, such that in the steady state, their values are numerically equal.

p122 Section 6.2.5, second para., fourth sentence should read :

"... it would need to be stored ..."

p151 Section 7.2.3, first para., last sentence, figure number should read 3.4 not 3.6.

p151 The text between equations 7.16 and 7.17, first sentence should read : "In the chosen configuration, Landau's Model Feedforward Gain; $K_m = 0$ ..."

p155 Equation 7.33, first term should read :

"$\underset{\sim}{A}_m{}^T + \ldots$"

p156 Equation 7.41. The β term should be replaced by μ.

p156 Equation 7.42. The second term in all 3 rows of the vector should be divided by $(2\sigma_x)$.

p157 Equation 7.53 - the input vector has been omitted; r should be inserted, in this case of dimension 1.

p158 Top of page, should read "... (section 5.3.2) :"

p179 Section 8.1.1. The statement "(and vice versa for the minimum principle)" should be omitted.

p180 Section 8.1.3, first para., last sentence, replace with :

"Not surprisingly, the performance of such computer based simulations is high."

p181 Paragraph on reference [8.1], second to last sentence should read : "... maximum principle to be impractical ..."