Application of an Acceleration Feedback Algorithm to Manipulator Position Control

Moshe Cohen

B.A. Physics, Cornell University, 1983

Department of Electrical Engineering

McGill University

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

M. Eng., McGill University, 1987

June 3, 1987

RESEARCH THESES/REPORTS

THESIS NO. 87-10TM

TITLE: Application of an Acceleration Feedback Algorithm of Manipulator Position Control

AUTHOR(S) Moshe Cohen

Department of Electrical Engineering

McGill University

Montréal, CANADA

DATE: June, 1987

GRANT OR CONTRACT:

NO. OF PAGES: 149

SUPERVISOR: Dr. L. Daneshmend

Abstract

The dynamical equations of robot manipulators are nonlinear and coupled, resulting in a system for which position control is difficult and often inaccurate. The joint position control algorithms of manipulators are generally implemented on microprocessors located at each joint, and are therefore inaccessible for examination of performance, modification, or improvement. This research presents an environment for easy implementation of joint position control algorithms on a manipulator and performs comparative experiments of two control algorithms using this environment. An acceleration feedback algorithm without feedforward compensation has been recently developed as a robust controller for manipulator joint position control. This thesis implements this algorithm along with a proportional-derivative algorithm for comparison, and evaluates their stability and performance on a PUMA 260, using the environment developed in this research.

Résumé

Les équations dynamiques d'un robot sont généralement couplées et non-linéaires, et par suite l'asservissement de position du robot est complexe et imprécis. De plus, les algorithmes d'asservissement des joints sont généralement implantés sur des microprocesseurs situés à chaque joint ainsi ils sont inaccessibles pour l'étude de performance, modification ou amélioration. Ce projet introduit un environnement adéquat pour l'étude d'algorithmes d'asservissement de position des joints d'un robot, et fait une étude comparative de deux algorithmes particuliers. Un algorithme en boucle fermée avec accélération sans anticipation, a été développé de façon à fournir un asservissement robuste de position. Cette thèse décrit l'implantation de cet algorithme, et celle d'un algorithme du type P-D, afin d'évaluer et comparer leurs stabilité et performance.

Acknowledgements

Thanks to Mom, Dad and Sis, whose unambiguous support and unfailing confidence in me throughout this research brought hope into my heart and gave me the strength and courage to proceed toward my goals. My thanks go out to my advisor, Laeeque Daneshmend, who provided a great deal of guidance, help, creative energy, and enthusiasm for my work, and without whom none of this would have ever happened. Finally, I'd like to thank McGill University, the Department of Electrical Engineering, the McGill Research Center for Intelligent Machines, the Computer Vision and Robotics Laboratory, and in particular my fellow graduate students for providing the type of atmosphere and support which made this work possible. Particular thanks go to John Lloyd and John Studenny for their extensive technical assistance, as well as to Vincent Hayward, Bruno Blais, Faycal Kahloun, Allan Dobbins, Iskender Paylan, Mike Huculak, Martin Boyer, Abdol-Reza Mansouri, Cem Eskenazi, Mike Parker, Daniel Kornitzer, Chantal David, Paul Freedman, Gregory Carayanis, Ajit Nilakantan, Mike Sabourin, and numerous others.

Contents

List of Tables ... viii

Chapter 1 Introduction ... 1

1.1 Sensory Robotics ... 2

1.1.1 Manipulators ... 2

1.1.2 Sensory Information ... 4

1.1.3 Artificial Intelligence ... 6

1.2 Position Control ... 8

1.2.1 Means of Implementation ... 8

1.2.2 Measurement of Performance ... 10

1.3 Statement of the Problem ... 11

1.3.1 Testing Control Schemes ... 11

1.3.2 Accurate Manipulator Position Control ... 12

1.3.3 Goals of the Research ... 12

Chapter 2 Manipulator Control ... 14

2.1 Description of the Problem ... 14

2.1.1 Manipulator Kinematics ... 14

2.1.2 Manipulator Dynamics ... 16

2.1.3 System Model ... 17

2.2 Manipulator Control ... 27

2.2.1 Classical Solutions ... 27

2.2.2 Adaptive Solutions ... 28

2.2.3 Robust Solutions ... 29

Chapter 3 Acceleration Feedback 32

3.1 Presentation of the Theory 32

3.1.1 Single Joint Application 33

3.1.2 Extension to Multiple Joints 34

3.1.3 Discrete Version 35

3.2 Discussion of Stability 37

3.2.1 Lyapunov Stability Theory 37

3.2.2 Stability of Acceleration Feedback 41

3.3 Theoretical Predictions 45

3.3.1 Stability 45

3.3.2 Performance 46

3.4 Simulation 47

Chapter 4 Experimental Environment 54

4.1 Hardware Environment 54

4.1.1 The Puma 260 Manipulator 54

4.1.2 Other Equipment 55

4.2 Software Environment 55

4.2.1 UNIX, NFS 57

4.2.2 RCCL, RCI 57

4.3 The Testbed 58

4.3.1 Features 59

4.3.2 Analysis Routines 65

Chapter 5 Controller Experiments 66

5.1 System Identification 66

5.1.1 Notation 67

5.1.2 Preliminary Tests 67
5.1.3 Frequency Response 70
5.1.4 Recursive Least Squares 71
5.2 Coding the Control Schemes 74
5.2.1 PD 74
5.2.2 Acceleration Feedback 76
5.2.3 Integral control 77
5.2.4 Other Controller Issues 77
5.3 Tuning the Control Schemes 79
5.3.1 PD 79
5.3.2 Acceleration Feedback 82
5.4 Controller Experiments 82
5.5 Analysis 120
Chapter 6 Conclusion 127
6.1 The Environment 127
6.1.1 Suitability to the Task 127
6.1.2 Suggestions for Improvement 129
6.2 The Testbed 131
6.2.1 Suitability to the Task 131
6.2.2 Suggestions for Improvement 131
6.3 Test Results 132
6.3.1 Validity of Test Results 132
6.3.2 Performance of Acceleration Feedback VS PD 133
6.4 Final Comments 135
Appendix A. Simulation Parameters 137

Appendix B. Experimental Conditions 140

References *144*

List of Tables

4.1 Kinematic Parameters for the PUMA 260 [Lloyd 84] 55

5.1 Conversion of K_u to SI Units 68

5.2 Error Table: Set 1 122

5.3 Error Table: Set 2 125

5.4 Error Table: Set 3 126

B.1 Experimental Configurations 141

B.2 Experimental Movements 142

Chapter 1 Introduction

The Robotics Institute of America has defined the term *robot* in the following manner: "A robot is a programmable, multifunction manipulator designed to move material, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks" [Holland 83]. It is clear from this definition that accurate trajectory tracking is required in order for a manipulator to function as specified. The goals of this work are two: The first goal is to create an environment by which position control algorithms can be tested and evaluated on a manipulator. The second goal is to present acceleration feedback control as an example of a position control algorithm and to test it in comparison with a well established proportional-derivative control scheme using the testbed mentioned earlier.

Background leading to this research is presented in the first chapter. Sensory robotics and position control theory are both described briefly, and then the problems to be tackled are stated explicitly

The second chapter further develops the problem of manipulator position control, describing the challenges presented by this problem and discussing some of the solutions that have been proposed to date

Chapter three presents acceleration feedback control as a better solution to the problem of manipulator position control. The theory is developed in some detail and stability is discussed. In addition, some theoretical predictions of performance are presented as are simulation results. The material in this chapter relies heavily on previous work performed by Studenny [Studenny 87].

The fourth and fifth chapters are devoted to a presentation of the work performed in the course of this research. Chapter four starts off with a description of the working environment present at the start of the research, and then moves on to describe the control algorithm testbed, which was the contribution of this research to the working environment. Chapter five describes experiments on system identification, explains the methods used to code and tune the controllers, and concludes with extensive comparative tests of the two control algorithms.

A deeper discussion of the research results may be found in chapter six. The environment and the testbed are both examined for suitability to the evaluation of position control algorithms. The performance results of acceleration feedback control and proportional-derivative (PD) control are discussed in depth, and conclusions are reached concerning their viability as manipulator position control algorithms. Finally, the entire approach to this research is reexamined in light of developments in the field and practical considerations.

1.1 Sensory Robotics

1.1.1 Manipulators

A great variety of manipulators are available for a large variety of intended applications. Classifications of robotic manipulators include lifting capacity, power source, application, degrees of freedom, and type of joints.

The lifting capacity of a manipulator is defined as the maximum weight the robot can safely and routinely lift while fully extended. Generally, manipulators are classified as light duty - up to fifteen pounds, medium duty - up to fifty pounds, and heavy duty.

Robots may derive motive power from a variety of sources. Some light duty manipulators use stepping motors, while heavy duty manipulators are often hydraulically powered. Most general purpose manipulators use servomotors, and pneumatically powered manipulators exist as well.

The applications for which manipulators are intended are nearly as varied as the actual manipulators. Examples of applications include welding, spray painting, assembly, and general purpose manipulators.

The degrees of freedom of a manipulator define its ability to move within a space. To achieve any position and orientation within a workspace, a manipulator must be capable of six degrees of freedom. Often, however, the types of tasks for which a manipulator has been intended do not require this capability, and manipulators with fewer degrees of freedom may be used. Such manipulators are cheaper, simpler, and may operate more quickly due to the reduced computational complexity in trajectory generation and torque calculations.

In general, manipulator joints are defined as either prismatic or revolute. Prismatic joints function as a piston, changing the length of link they move, and revolute joints sweep through an angle, changing the orientation of the link they control. Most commercial robots are wrist partitioned, meaning that they have three degrees of freedom in the arm and three concentrated at the wrist. A position is usually reached through the three degrees of freedom in the arm, and the manipulator is classified by the types of joints in the arm. For example, manipulator whose arm consists of three revolute joints is called an RRR manipulator, and one whose arm consists of two revolute joints and a prismatic joint is classified as RRP. The wrist typically consists of three revolute joints, sometimes bundled together as a spherical joint, and is used primarily to control the orientation of the end effector.

1.1.2 Sensory Information

In order for a robot to operate effectively within a changing environment, it must be provided with sensory information and a means of interpreting this information. Computer vision equipment and algorithms provide a great deal of this data, as well as range detection devices and tactile sensors. To be of any great use in manipulator control, this information must be processed in real time and must not use excessive resources. In general, industrial machine vision is characterized by the following three criteria. (1) The necessity to control the environment, (2) well defined performance and success criteria, and (3) extreme sensitivity to cost. Typically, a machine vision system is constructed as a series of pipelined modules, each with a specific function for which it is optimized. The steps involved include visual sensing, segmentation, description, recognition, and interpretation. The first two steps listed above are termed low level vision, and use only local information with no assumed knowledge of the scene. The latter steps are called high level vision, and involve an understanding of what the image means.

Input to computer vision systems is generally provided by either a vidicon camera or a *charge-coupled-device* (CCD). A vidicon camera is a television camera in which a beam of light is scanned over an array of closely spaced capacitors with a common baseplate; these capacitors lose charge as they are struck by incident light. The current signal from the common plate of the capacitors is collected by a contact lead as the video output. These devices have several disadvantages for robotic applications. These include distortion, power consumption, high voltages, and their large size. CCD's can generally be thought of as a large array of photosensitive detectors, each of which represents one pixel of the image; the binary value of each pixel is available as the output of these devices. The resolution of the camera is directly related to the number of pixels per image, which means that as the resolution improves, so does processing time. This is a major disadvantage of CCD cameras, as well as the difficulty of interfacing them to computers. Data from pictures or slides may be input to a computer using frame, row, column, window, and pixel grabbers which digitize images into memory.

The large volume of data arriving from the input devices listed above imposes harsh requirements on the memory and speed of most processors. To facilitate image analysis, it is common practice to reduce the amount of data by thresholding the image to reduce gray level data to binary data. Other data reduction techniques include moments, projections, segmentation, labeling, windowing and histograms. Other operations then performed as part of low level vision include noise filtering, convolution, edge detection, closing edge contours, and template matching. Once these operations are completed, high level vision may be performed using template or model matching, statistical approaches, or topological techniques.

Two dimentional image analysis is subject to errors due to varying illumination, shadows, and texture. One solution to these problems is the use of three dimentional range image analysis. This technique is not only more complex, however, but the equipment to acquire the range data is costly. The data may be obtained by stereo vision (triangulation), laser or ultrasonic range sensors, or structured light - patterns of light projected on an object. Of these, stereo vision is sensitive to occlusion, ultrasonic sensors have low spatial resolution, laser scanners are slow and expensive, and structured light techniques require a strictly controlled environment.

In robotics applications, a combination of ranging devices at the gripper to give rapid range and orientation data, and fixed cameras to give scene information offer an attractive combination. Mounting cameras on or near the end effector has the advantage of allowing a smaller image to be used, but required special calibration software. The uses of vision techniques in robot control include trajectory planning, obstacle avoidance, and adaptive position correction. This imposes a requirement on the system to contain routines which interpret moving images as well as the stationary image analysis techniques described above.

Since manipulators must interact with their environment by direct physical contact, sensory information regarding force and torque is of great value in robotic applications. Force and torque sensors measure three components of the force and three components of the torque acting between the gripper and the manipulated object. The difficulties posed by such sensors include a high price, as they require calibrations to compensate for drifts, and a relatively long calculation time. Hybrid control uses both position and force data, and is rapidly gaining popularity. In motions which involve a manipulator interacting with its environment, such as fitting workpieces together or screwing in a bolt, hybrid control is essential for providing control once the manipulator and the object are in contact.

1.1.3 Artificial Intelligence

A programming technique which is finding ever growing use in robotic applications involves the use of artificial intelligence, or expert systems. Artificial intelligence differs from other types of programming in an emphasis on symbol manipulation. Symbols are manipulated by inference, and particularly by deduction, which is characterized by a system called the predicate calculus. The predicate calculus consists of a language for expressing propositions, and rules for inferring new propositions from those already available. The set of previously known facts is referred to as axioms, and rules of inference are known as theorems. Another form of inference is called abduction, in which explanations are generated from a set of facts This form of inference may lead to false conclusions, but is necessary to allow conclusions to be reached from a possibly incomplete set of facts.

An artificial intelligence package generally contains three components: a user interface program to acquire data, an inference engine, and a data base. The user interface program accepts data which it then formulates in predicate calculus form. The inference engine uses theorems to operate on new data and available propositions, and to insert new propositions into the data base. The data base stores all of the propositions available thus far both from input data and output from the inference engine.

Many problems can not be solved in closed form, and a solution must be found by selection from a set of possible actions. Scheduling the activities of a mobile robot is an example of the type of problem which requires searching for an optimal solution. Search problems involve moving a system from an initial state to a goal state, often by seeking an optimal path solution. From a given state, an operator is applied to reach its immediate successor or successors, and the set of all possible states reached by subsequent operations is called the search space of the problem. Search algorithms are characterized by operator-ordering functions which select from a set of operators, and by state-evaluation functions which estimate the distance from a given state to the nearest goal state. Choices of these two types of functions give rise to the various types of search algorithms.

Artificial intelligence techniques are useful in representing and recognizing scenes during late, or high level, vision. In this application, the first stage involves deciding on a set of shape primitives to adequately describe an object. Objects in a scene can then be compared with models of various objects in a data base to achieve recognition. The set of possibilities is called a discrimination-net, and must be traversed using a search technique to allow efficient processing of the image.

In manipulator control, artificial intelligence techniques find use primarily in two fields. The first application involves path planning and collision avoidance, and the second application involves task level programming. Task level programming simplifies the process of robot programming by requiring the user to specify only the relationships between objects, as opposed to the detailed manipulator motions required to perform the task. Thus the user provides details concerning the position of objects in the present and the future, and the system uses an inference engine to provide the interpretation of these task level commands into detailed motion commands.

The process of learning is likely to be essential for manipulators to operate in a completely unstructured environment. In this context, learning is defined as an advanced artificial intelligence technique in which the conclusions of inference on propositions and theorems produce new theorems as well as new propositions. The study of learning is truly at its infancy at this time, and concrete results are as yet meager.

1.2 Position Control

As the title of this work indicates, this research involves the implementation of position control algorithms on a physical system. It is therefore necessary to briefly present the physical components that are used in position control, as well as their effect on implementation In addition, the standards of measurement by which the performance of the implemented control algorithms will be judged must also be described in some detail.

1.2.1 Means of Implementation

In this research, the only types of position control systems to be considered will be those involving feedback All feedback systems must contain the following elements to control a plant: sensors to obtain information about the plant, a comparator to compare the actual values obtained from the plant to desired values, a controller to calculate the commands which will drive the plant toward the desired values, and output devices to effect these commands

In particular, the sensory information required for joint position control involves joint position, velocity, and in some cases, acceleration. Joint position data may be obtained in a variety of ways One of the simplest types of position sensors is the potentiometer, an electromechanical transducer that converts mechanical energy into electrical energy For a given input voltage, the output voltage of the potentiometer is proportional to a mechanical displacement, which can be either rotational or translational Though relatively cheap and reliable, potentiometers are prone to wear, and being analog devices, are somewhat cumbersome to interface to digital computers

A type of device found increasingly in modern position control systems is the encoder, which comes in two major subtypes, incremental and absolute. Incremental encoders are available in rotary and linear forms, but both types involve the same four basic components: a light source, a rotary disk, a stationary mask, and a sensor The disk has alternating transparent and opaque sectors whose size determines the incremental period. As the disk rotates, light from the source is alternatively passed to and blocked from the sensor, and a count of the number of pulses at the sensor provides a measurement of position from a known starting point Disadvantages of incremental encoders include an inability to deal with temporary power failures, and the requirement for an additional mask to indicate the starting position Furthermore, a single encoder can not provide information on the direction of travel, and another encoder 90 degrees out of phase with the first must be provided for this purpose. These disadvantages often counterbalance the relative simplicity and low cost of incremental encoders Absolute encoders use the same basic components as incremental encoders, however they use a multiple track disk which defines each shaft position in terms of a binary number or Gray code. Unlike incremental encoders, absolute encoders are not sensitive to power glitches, noise or transients, but are somewhat more complex in construction.

Tachometers are electromechanical devices used to measure velocity. The device works essentially as a generator, with the output voltage proportional to the magnitude of the angular velocity While velocity feedback is often used in robotic applications to improve position control, many manipulators are not equipped with tachometers due to weight, size and cost restrictions In such cases velocity may be inferred from position measurement, particularly when the implementation is carried out on a digital computer, in which a position measurement may be differentiated numerically to obtain velocity. Acceleration is likewise generally inferred from position or velocity data, as currently available accelerometers are inaccurate and prohibitively expensive.

Comparison of the desired signal with its actual value may be performed using an up down counter, or more commonly in a digital computer. Typically, in modern control systems, the control algorithm itself is also implemented on the same computer.

Output to the manipulator is effected through a digital to analog converter (DAC), which converts the computed command from a binary code in the computer to an analog voltage available as output. This signal is then amplified and transmitted to the joint motors using current or voltage amplifiers. A current amplifier is a device which supplies a current proportional to its input voltage and has a high output resistance. It is used in torque control applications such as manipulator joint control since a motor's armature current is proportional to the loading torque. Disadvantages of control loops utilizing this type of amplifier are a time constant which depends on the manipulator position and payload, and the absence of internal damping due to back emf. Thus, control algorithms using a current amplifier must add a derivative-gain to provide damping for stability, and the presence of steady state error requires an integral term as well. An alternative approach involves using a voltage amplifier, which provides output voltage proportional to an input voltage. This type of amplifier controls velocity rather than torque, and therefore requires tachometer feedback. It does possess natural damping, reducing the need for a derivative controller, but otherwise operates in a similar manner to the current amplifier In reality, systems utilizing the two types of amplifiers operate similarly, with some differences in time constants and gains

1.2.2 Measurement of Performance

Evaluation of control algorithms requires the development of performance measures by which algorithms may be tested and compared There are several different ways in which such algorithms may be evaluated, and the use of a variety of measures gives a clear indication of the capabilities of the system.

Feedback control is primarily concerned with an error signal, which is generated as the difference between a desired position and an actual position obtained from the plant. In particular, trajectory tracking for manipulators involves position control in the presence of varying dynamics and disturbances. An important measure of performance is the steady state error, which is defined as the error when time goes to infinity. In reality, the steady state error may be measured shortly after transient effects have settled down, and is unlikely to change later on Systems give steady state errors that vary with the type of input signal. Typical test inputs include step functions, ramps, parabolas, and sinusoids. Measurements of average steady state errors of a system in response to a variety of input signals give a good evaluation of its performance.

The transient response of a system to various input signals is also of value in testing control algorithms Two measures of performance are of particular interest in evaluating the transient response of a system. One involves the length of time it takes the system to relax to its steady state, and the other is the percent overshoot, if any, from the desired value Though in this research we are more interested in steady state error for trajectory following, transient response is important to ensure adequate transition from a stationary state to steady state motion.

The frequency response of a system is likewise of interest in measuring the capabilities of a control algorithm Of primary interest is the stability bandwidth, which gives an indication of the range of frequencies for which this system may operate. Gain and phase lag of the system are likewise important measures of performance.

Examination of control algorithms with respect to the above performance criteria is likely to produce a comprehensive evaluation of the capabilities of the system Various input test signals should be used to assure the consistency of the system performance over its full expected range of operation.

1.3 Statement of the Problem

It is now necessary to briefly present the problems this research is intended to address and the contribution this work is to provide toward the solution of these problems.

1.3.1 Testing Control Schemes

The position control algorithms of commercial manipulators are often embedded within the joint control microproccessors at each joint, and are therefore inaccessible to external observers. The performance of these control schemes is thus very difficult to measure, and the only available data on performance is the externally observable motion of the manipulator. Key data concerning the inputs and outputs of the controllers in terms of torques and currents is thereby missing, and joint control is often regarded as a "black box" type of problem. Furthermore, modification or replacement of the control schemes at the joint controllers is extremely difficult, as is measurement of the performance of new algorithms.

1.3.2 Accurate Manipulator Position Control

The dynamical equations describing a robot manipulator under rigid body assumptions are generally nonlinear, coupled second order differential equations. Robot manipulator control laws are designed to maintain the desired joint trajectories with a small bounded error, despite the nonlinear system dynamics. Unfortunately, the nonlinear effects under which manipulator joint control must operate are of such great magnitude that many control algorithms do not function well in practice. Furthermore, many control algorithms which do produce an adequately small bounded error are sensitive to parameter variation, or are very restricted in terms of operational bandwidth. While many algorithms have been developed in theory and quite a few have been tested under simulated conditions, relatively few have been actually tested on a manipulator operating under realistic conditions of nonlinear effects. Thus, the development of a position control algorithm which produces an adequately small bounded error over a large bandwidth of frequencies, and which is not overly sensitive to parameter variation, remains a topic of research. Finally, the implementation of such an algorithm on an actual manipulator and its evaluation based on the performance criteria listed above is likewise an open question.

1.3.3 Goals of the Research

The research work described in this text addresses the two problems described above, and thus the goals of the research may be regarded as two distinct efforts. The first thrust of this work involves tackling the inaccessibility of control schemes operating on manipulator joints. Using the RCCL/RCI system [Lloyd 85], a robot control interface running on a VAX 750 computer, and a PUMA 260 robot, an attempt is made to create a general purpose testbed by which various control algorithms may be implemented and tested under realistic conditions. The goal of the testbed is to provide the user with a means of implementing a control algorithm to replace the ones which normally run at the joint controllers. The controller then runs on the VAX, allowing convenient user interface to test the system under various conditions and with a variety of input signals. Commands are sent to the manipulator, and data is read back from the joint encoders for analysis. The system allows motion of multiple joints, thereby creating a test environment which includes inertial and Coriolis effects as disturbances. Analysis tools to decipher the data are provided as well, allowing the user to obtain quantitative and qualitative results by which to evaluate the algorithm.

The second line of research in this project involves the implementation and comparison of two control algorithms on the PUMA manipulator, using the testbed described above. Studenny [Studenny 87] proposed acceleration feedback control without feedforward compensation to be a stable control law in application to robot manipulators. An algorithm which implements this control law has been coded and tested on the system as described above. For comparison, a proportional-derivative algorithm has also been implemented and tested under the same conditions. Since neither scheme eliminates steady state error, an optional integral controller was also coded to allow accurate end point positioning. The goal of this research is to evaluate the performance of the two control algorithms and to analyze the suitability of the hardware, the software environment, the testbed, and the control algorithms to the task of manipulator position control.

Chapter 2 Manipulator Control

2.1 Description of the Problem

It is necessary at this point to include a description of manipulator kinematics and dynamics, as well as the mathematical background on which they are based. The general problem of control will depend on the system model developed from the kinematics and dynamics, and a framework must be introduced to allow the study of control solutions which have been developed to date

2.1.1 Manipulator Kinematics

The kinematics of robot manipulators may be described in terms of homogeneous transformations, Euler angles, and other formalisms We will concentrate here on a kinematic description using homogeneous transformations, since this is the formalism adopted by many of the control strategies we will discuss in later sections; it is also the one on which the RCCL/RCI system, which forms the basis of most of the programming of this research, is based as well The problem of kinematics has been extensively discussed in the literature, for example in [Lee 82], [Paul et al 81a], [Paul et al 81b], and many other sources, and is generally well understood. A brief discussion follows.

A (3 x 3) rotation matrix may be defined as a transformation matrix which operates on a vector in 3-D Euclidian space and maps its coordinates expressed in a rotated coordinate frame from a known reference frame. The transformation from frame O to frame i can be expressed as

$$p_i = A_O{}^i p_O \tag{2.1}$$

where p is a point in the space.

Since a (3 x 3) matrix does not allow the expression of translation, a fourth column is added to the matrix, resulting in a (4 x 4) matrix of the following form:

$$A = \begin{pmatrix} rotation & translation \\ 0 & 1 \end{pmatrix} \tag{2.2}$$

This type of matrix is called a homogeneous transformation matrix, and maps a position vector expressed in homogeneous coordinates from one coordinate frame to another. Several homogeneous transform matrices representing various rotations and translations may be multiplied together to yield the final position and orientation of the desired frame. This is of great use in manipulator kinematics, in which we are concerned with the position and orientation of an open spatial kinematic chain. Each link in the chain may be viewed as a rotated or translated frame with respect to the preceding link, and the product of the transform matrices of all the links gives the position and orientation of the end effector with respect to the base coordinate frame. To facilitate the development of a formalism describing the frames at each link, a representation of joint and link parameters was developed by Denavit and Hartenberg [Denavit 55] In the Denavit-Hartenberg representation, each link is assigned parameters and variables with respect to the preceding link. Each link is assigned one variable, joint angle in the case of revolute joints, and joint travel in the case of prismatic joints. Other parameters depend on the fixed position and orientation of the link with respect to its predecessor. The forward kinematics problem is simply that of multiplying the A matrices of each link in a particular configuration to arrive at the T matrix describing the position and orientation of the end effector:

$$T = A_1 A_2 A_3 \dots A_n \tag{2.3}$$

The inverse kinematics problem is more involved. Here the goal is to find the A matrices that can lead to a known T matrix. This problem is of great importance in manipulator kinematics, as the space in which a manipulator must operate is in a world coordinate frame, while the manipulator is controlled in joint space. The inverse kinematics solutions give the joint positions or velocities required to produce a given end effector position and orientation, or Cartesian velocity. The solution is generally not unique, and typically requires the solution of the inverse Jacobian matrix at each position. Other methods have been developed to simplify the calculations, and for some manipulators, such as the PUMA used in this research, there exist closed form solutions to the problem.

2.1.2 Manipulator Dynamics

The dynamic equation for a six degree of freedom manipulator has been shown to be highly nonlinear and consists of inertial terms, coupling inertial forces, friction, and gravity loading. There are several mathematical constructs used to describe manipulator dynamics, the most commonly used of which are the Newton-Euler method and the Euler-Lagrange formulation [Lee 81], [Luh 83], [Driels 84]. Other formulations have been proposed to alleviate some of the inefficiencies associated with Newton-Euler and Euler-Lagrange [Kane 83].

The Euler-Lagrange method produces a set of coupled, nonlinear differential equations. These equations are not so efficient as those produced by Newton-Euler, but their performance may be improved by using a recursive Euler-Lagrange formulation [Hollerbach 80] or other techniques [Mahil 82]. In the Euler-Lagrange formulation, kinetic and potential energies are derived from the homogeneous transforms describing the kinematics. Using the Lagrange function

$$L = KE - PE \tag{2.4}$$

and applying the Euler-Lagrange method, we obtain the following dynamic equation:

$$D(q)\ddot{q} + H(q, \ddot{q}) + G(q) = \tau \tag{2.5}$$

where τ is the torque necessary to drive the system. D represents the inertial terms. H centripetal and Coriolis forces, and G gravity loading.

The Newton-Euler method is fast and accurate, and consists of forward and backward recursive equations which may be applied to each link sequentially. The forward recursion propagates kinematics information from the base coordinate frame to the end effector frame. The backward recursion carries the forces and moments exerted on each link from the end effector to the base frame. The derivation of the Newton-Euler formulation is more difficult than Euler-Lagrange, and involves cross product terms. It is based on the fact that the torque applied at a joint can be determined from the moment exerted on a link by its predecessor. Using the angular velocities and accelerations of the preceding and following links, the Newton-Euler equations may be derived from cross product equations of these parameters.

Another solution to the dynamics equation was presented in [Kane 83], and involves the use of Kane's Dynamical Equations. This formulation introduces generalized inertia forces K_r^* and generalized active forces K_r, which are determined from joint positions, through intermediate calculated values of generalized velocities, angular velocities, accelerations, and angular accelerations. Using Kane's equation

$$K_r^* + K_r = 0 \tag{2.6}$$

one can obtain values for joint torques. The authors demonstrated that this formulation is more efficient than even the most optimized Euler-Lagrange and Newton-Euler schemes

2.1.3 System Model

In the discussion of the system model, the following notation will be followed throughout. The variable θ will be used to denote joint position, and θ_d the desired joint position. For actuators controlled by a voltage amplifier, the input voltage will be denoted as V_{in}, and for those controlled by a current amplifier, the input current will be specified as DAC, to signify the value on the digital to analog converter

The simplest and most commonly used control algorithms fall into the classical category, drawing both from frequency response and state space methods. In principle, these techniques rely on the assumption that the coupling and nonlinearities inherent in manipulator dynamics can be adequately compensated for through feedback. A manipulator is therefore regarded as a conglomerate of independent rigid bodies and actuators, and the algorithms in use draw from linear time invariant control theory.

The most basic of these schemes completely disregard manipulator dynamics and attempt to control the system using the error signal alone. Each link of the manipulator is described as a second order linear time invariant system, and proportional-derivative (PD) control is typically used to achieve the desired position. Frequently an additional integral term is added to the control law in order remove any steady state position error. The resulting proportional-derivative-integral (PID) control algorithm is one found very frequently in commercial robots.

A cornerstone of nearly all classical attempts to control robotic manipulators is to regard each joint of the manipulator as a second order system. This implies that the dynamic equations of the manipulator can be described mathematically by second order differential equations, and effects that do not fit into this description are treated as small disturbances. Two questions must now be answered in further detail: The first question involves demonstrating that a manipulator joint can be adequately modeled as second order system, in particular applying this analysis to the puma 260. The second question involves showing that this analysis is still valid in light of the joint's being part of a larger system ~~which~~ exerts dynamical forces greater than those which appear in each joint individually. If both these conditions are satisfied we can then go on to discuss methods of controlling the system

Approaching this problem from the point of view of manipulator dynamics, we first look at a single link manipulator [Paul 81]. Given an effective link inertia J and an actuator modeled by a gain k_m and viscous damping F, if we ignore coulomb friction, we can model the actuator as shown in figure 2.1.

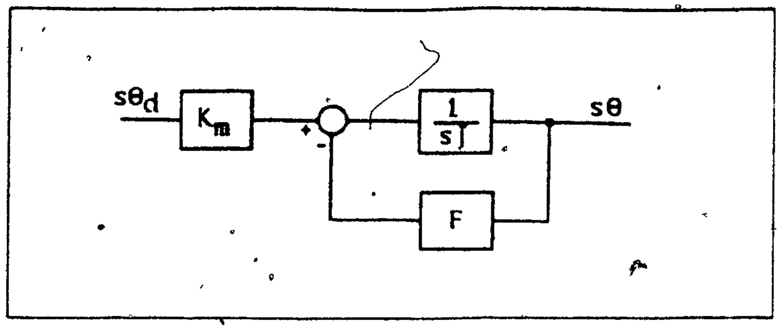

Figure 2.1 Simple Model of Actuator Driven by a Voltage Amplifier [Paul 81]

The transfer function of this system is

$$\frac{s\theta(s)}{s\theta_d(s)} = \frac{k_m}{sJ + F} \tag{2,7}$$

By providing both velocity and position feedback, Paul then arrives at a system which looks like the one in figure 2.2.

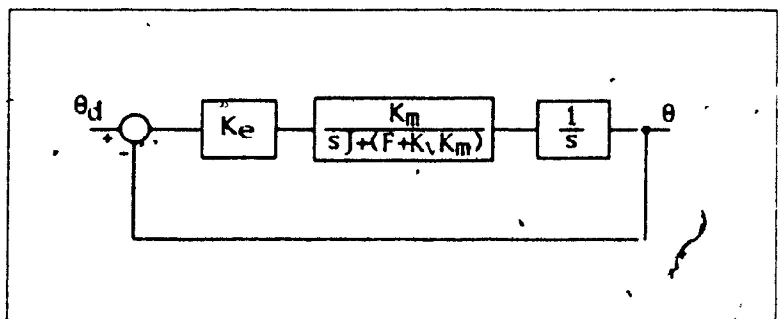

Figure 2.2 Model of Actuator with Position and Velocity Feedback [Paul 81]

The transfer function of this system is

$$\frac{s\theta(s)}{s\theta_d(s)} = \frac{k_e k_m}{s^2 J + (F + k_v k_m)s + k_e k_m} \tag{2.8}$$

This equation can then be manipulated into the form of a second order system

$$\frac{1}{s^2 + 2\varsigma\omega_n s + {\omega_n}^2} \tag{2.9}$$

by setting

$$\omega_n = \sqrt{\frac{k_e k_m}{J}} \tag{2.10}$$

and

$$\varsigma = \frac{F + k_v k_m}{2\sqrt{J k_e k_m}} \tag{2.11}$$

The discussion above may strike the reader as overly simplistic and therefore unconvincing. A considerably more detailed discussion may be found in [Luh 83], in which a single joint manipulator is analyzed taking into account actuator inertia (J_a), manipulator inertia (J_m) at the actuator side, link inertia (J_l), damping at the actuator side (B_m) and load side (B_l), average friction torque (f_m), gravitational torque (τ_g), generated actuator shaft torque (τ_m), internal load torque (τ_l), angular displacement at actuator shaft (θ_m) and load side (θ_l), and the gear ratio (n) Even with the detail in this example, Luh applies unity position feedback and arrives at the overall transfer function

$$\frac{s\theta(s)}{s\theta_d(s)} = \frac{1}{s^2 + \frac{RB_{eff}+K_I K_b}{RJ_{eff}} s + \frac{K_\theta K_I}{RJ_{eff}}} \tag{2.12}$$

where

$$\begin{gathered} R = motor\ armature\ winding\ resistance \\ B_{eff} = B_m + n^2 B_l \\ K_I = torque\ constant \\ K_b = back\ EMF\ constant \\ K_\theta = conversion\ constant\ from\ optical\ encoder\ to\ voltage \end{gathered} \tag{2.13}$$

$$J_{eff} = J_a + J_m + n^2 J_l \tag{2.14}$$

All other constants were eliminated as insignificant, and the single link manipulator was therefore shown to be a second order system. Another presentation of this subject may be found in [Craig 85], in which the discussion starts off with the force equation $F = ma$ which is already second order, and is extended to manipulators.

Thus far the analysis has been carried out in terms of continuous time formulation. Since all of our experimental work concerning control algorithms is carried out using a digital computer, it is necessary to demonstrate that the model we have chosen remains valid in discrete form. The entire system is shown in figure 2.3:

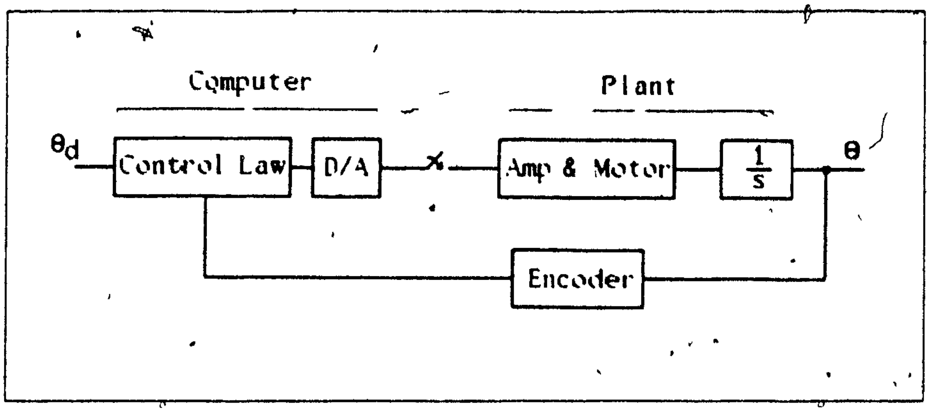

Figure 2.3 Digital System for Actuator Control Using Position Feedback

Let

$$K_A = amplifier\ gain$$

$$\frac{K_m}{\tau_m s + 1} = transfer\ function\ of\ motor\ and\ load$$

we get

$$\frac{\theta(s)}{V_{in}(s)} = \frac{K_A K_m}{s(\tau_m s + 1)} \tag{2.15}$$

or

$$\frac{\theta(s)}{V_{in}(s)} = \frac{K}{s(\tau_m s + 1)} \tag{2.16}$$

Applying unity position feedback, we get

$$\frac{\theta(s)}{\theta_d(s)} = \frac{K}{\tau_m s^2 + s + K} \tag{2.17}$$

As before, a second order system.

To discretize this, we must include a zero order hold in the plant equation. The system is now shown in figure 2.4.

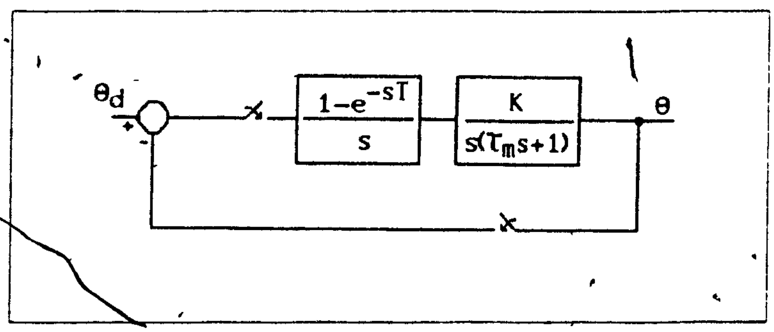

Figure 2.4 Discrete Model of Actuator with Position Feedback

$$z = e^{-st} \tag{2.18}$$

$$\left(\frac{1 - e^{-st}}{s}\right)\left(\frac{K}{s(\tau_m s + 1)}\right) = (1 - z^{-1})\left(\frac{1}{s^2}\frac{K}{s(\tau_m s + 1)}\right) = D(z)G(z) \tag{2.19}$$

$$G(z) = Z\left(\frac{K}{s^2(\tau_m s + 1)}\right) = \frac{T_s z}{(z - 1)^2} - \frac{(1 - e^{\frac{-T_s}{\tau_m}})z}{\frac{1}{\tau_m}(z - 1)(z - e^{\frac{-T_s}{\tau_m}})} \tag{2.20}$$

$$\frac{\theta(z)}{\theta_d(z)} = \frac{D(z)G(z)}{1 + D(z)G(z)} = \frac{b_0 z + b_1}{z^2 + a_0 z + a_1} \tag{2.21}$$

where

$$a_0 = b_0 - (1 + e^{\frac{-T_s}{\tau m}}) \tag{2.22}$$

$$a_1 = e^{\frac{-T_s}{\tau m}} + b_1 \tag{2.23}$$

$$b_0 = [T_s - \tau_m(1 - e^{\frac{-T_s}{\tau m}})]K \tag{2.24}$$

$$b_1 = [\tau_m(1 - e^{\frac{-T_s}{\tau m}}) - T_s e^{\frac{-T_s}{\tau m}}]K \tag{2.25}$$

Thus, in discrete form, a single link manipulator may still be regarded as a second order system.

Unlike the manipulators described above, the joints of the Puma 260 use current amplifiers, and not the more common voltage amplifiers. This should mean that a constant input voltage produces a constant acceleration rather than a constant velocity at the output shaft of the motor. This relation follows from the torque equation.

$$torque = K_T I_a \tag{2.26}$$

where K_T is a constant. Since the amplifier output is a current, a constant input produces a constant torque, which by

$$torque = J\dot{\omega} + f\omega \tag{2.27}$$

produces a constant acceleration damped by rolling friction. Initially modeling the motor as contributing a gain K'_m and the amplifier with gain K_A, our model looks like the one in figure 2.5.

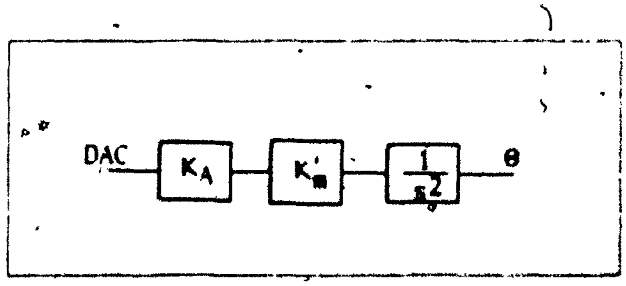

Figure 2.5 Model of Motor Driven by a Current Amplifier

In the joints of the Puma 260, friction contributes a significant enough disturbance that it was decided to include a velocity-dependent frictional term into the model, as shown in figure 2.6:

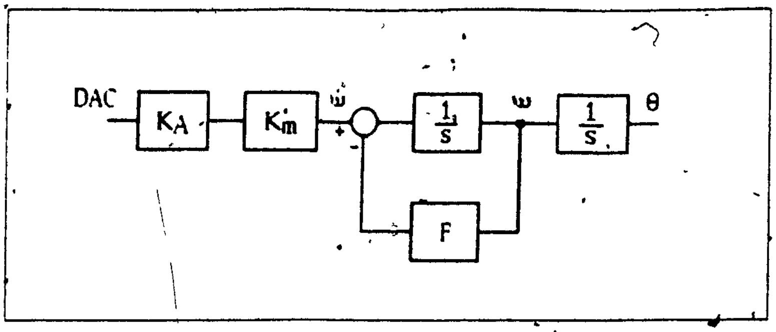

Figure 2.6 Revised Model, including Viscous Friction

This system has an overall transfer function of

$$\frac{\theta}{DAC} = \frac{K_m}{s^2 + K_f s} \tag{2.28}$$

where

$$K_m = K_A K'_m \tag{2.29}$$

With unity position feedback, we further get

$$G(s) = \frac{\theta(s)}{\theta_d(s)} = \frac{K_m}{s^2 + K_f s + K_m} \tag{2.30}$$

a second order system.

It now remains to be demonstrated that a discretized version of this model behaves in the same manner. To obtain this model, we must use a zero order hold and the equation [Franklin 81]

$$G_{ZOH}(z) = (1 - z^{-1}) Z\left(\frac{G(s)}{s}\right) \tag{2.31}$$

arriving at

$$G_{ZOH}(z) = \frac{K_m(K_fT_s - 1 + e^{-K_fT_s}) + (1 - e^{-K_fT_s} - K_fT_se^{-K_fT_s})}{K_f^2(z-1)(z-e^{-K_fT_s})} \tag{2.32}$$

where T_s is the sampling interval. This is already a second order system, and applying unity position feedback we get yet another second order system.

Though a single link manipulator has been shown to be adequately described as a second order system, the inertial forces due to the motion of other links of the manipulator must be considered before the analysis is complete. One example of such analysis may be found in [Paul 81]. The three forces generated by the motion of other links include inertial coupling, centripetal forces, and Coriolis force. The torque equation,

$$F_i = J_{ii}\ddot{q}_i \tag{2.33}$$

may be rewritten as

$$F_i = J_{ii}\ddot{q}_i = (D_{ii} + I(n)_i)q_i \tag{2.34}$$

where D_{ii} is the effective link inertia, $I(n)_i$ is the actuator inertia, and n is the gear ratio. The coupling shows up in D_{ii} and is a function of the manipulator geometry. Paul suggests the use of feedforward compensation to overcome coupling, since manipulator geometry is generally well known in advance, but contends that frequently the effect is so minor as to not require any form of compensation. Centripetal and Coriolis forces occur only at high speeds and do not cause instability. They may, however, generate position errors. Giving the example of a two link manipulator, a motion of the first joint θ_1 by $\Delta\theta_1$ results in inertial torque

$$T_1 = mr^2\ddot{\theta}_1 = \frac{4mr^2\Delta\theta}{T^2} \tag{2.35}$$

and a centripetal torque

$$T_2 = mr\dot{\theta}_1 r = \frac{4mr^2\Delta\theta}{T^2}\Delta\theta \tag{2.36}$$

Clearly, for small $\Delta\theta$ the inertial torques dominate and we have the relation

$$T_{centripetal} = T_{inertial}\Delta\theta \tag{2.37}$$

Errors arising from Coriolis forces can similarly be shown to be small enough that the overall nature of the system is unaltered, and disturbances may be compensated for by feedback. Thus, even in the presence of nonlinear forces due to the motion of other links of the manipulator, each link may still be described as a second order system of the form shown earlier. These results are confirmed by [Luh 80], resulting in the assertion that manipulator dynamics may be described as disturbed second order differential equations, where the disturbances are small except under high velocity and acceleration conditions.

The importance of obtaining second order systems is twofold: first, such systems are well known mathematically, and many analysis tools have been developed to deal with them. Second, within classical control theory there are many design procedures suitable for specifying control schemes for second order systems. The above analysis showing single and multiple link manipulators to be of second order allows control, design and analysis using standard classical methods and largely ignoring the inherent nonlinearities in the system.

2.2 Manipulator Control

Several methods to solve the robot arm control problem have been developed. These solutions can roughly be classified into three groups: classical, adaptive, and robust.

2.2.1 Classical Solutions

Classical solutions are deemed classical in the sense that the design procedure is derived largely from classical control theory. The techniques described in this section include proportional-integral-derivative (PID) control, various feedforward and computed torque techniques, and a smattering of other methods. The common denominator of these techniques is that the emphasis is on minimizing an error term which is derived from a second order system model of the plant. In the case of PID, only the error is considered. Feedforward and computed torque techniques generally require an accurate model of the the plant and associated parameters. All of the solutions presented in this section allow only fixed control parameters, and largely ignore the question of sensitivity to parameter variation and modeling errors.

Of particular interest is a paper by Luh, Walter, and Paul, [Luh 81], in which they propose resolved acceleration control as a control algorithm for position control of manipulators. While different from the acceleration feedback algorithm which will be described in the next chapter, the idea of using acceleration in the feedback loop to compensate for the inherent nonlinearities of the system produced an equation whose form is remarkably similar to the one used in this research. Unlike the algorithm of this thesis, however, the algorithm described in [Luh 81] premultiplied the control equation by an inertia matrix, and required a perfect cancellation of the nonlinear terms by a calculated model. It was therefore not shown to be robust to parameter uncertainty, and was computationally expensive. The ideas of this paper grew out of previous work [Whitney 69], in which the main emphasis was to control a Cartesian trajectory from joint space. An extension of this scheme, in which the position of the end effector was to be controlled in Cartesian space directly using convergent force control was presented in [Wu 82].

An optimal control solution was proposed by Kahn [Kahn 71], in which time was the criterion to be minimized. The theory was developed both directly for the nonlinear system, and indirectly for a linearized version. Though accuracy is lost, and the system results in only suboptimal control for the second case, the computational complexity involved with applying the optimality condition to the full nonlinear case was sufficient to warrant development of the linearized system. This system was not shown to be robust, nor was stability proven.

An unusual algorithm may be found in a paper by Albus [Albus 75], in which he attempted to model the controller on a functional description of the human brain. Rather than computing a control function, this system consisted of a look-up table guided by heuristics and a complex memory management system. Though fast, the problems presented by such a system involved interpolation between various points in the table, and an inability to deal with variations in load.

The common theme among the above articles, with the exception of [Albus 75] is that they are based on classical or state space control theory. They typically do not draw from nonlinear control theory, multivariable control, or robust stabilization. No attempt is made to estimate the parameters of the system on-line, and the method used to allow system operation in the face of nonlinearities is typically to feed forward compensation for a precomputed estimate of the disturbance terms. Such schemes work well at low frequencies, and provide reasonable tracking if the system parameters are well known. Often, however, the system parameters are difficult to identify, and errors in the estimates generate large tracking errors.

2.2.2 Adaptive Solutions

Given a model of the plant whose structure is known but whose individual parameters are not, one answer to the problem of parameter uncertainty involves computation of the parameters based on an on-line estimation method. Thus, adaptive solutions use estimation techniques to dynamically adjust either the plant or the controller parameters, and by this avoid the errors in control due to precomputed parameters.

A survey of adaptive control algorithms may be found in [Daneshmend 87]. Adaptive control algorithms may be broken up into two major subgroups: joint space, and task space techniques. Joint space techniques are based on estimating the parameters of the joint dynamics directly Most are model referenced (MRAC), and use either least squares algorithms or other estimation scheme Both single input single output (SISO) and multiple input multiple output (MIMO) schemes are presented, though the simulation results do not point conclusively towards the superiority of one or the other. Task space and hybrid control schemes are also described in this paper, and attempt to manipulate the joint based on a higher level Cartesian description of the problem or on a combination of position and force data. Much work remains to be done in this field, both in joint space and task space. Stability and robustness issues are not adequately addressed for many of these schemes, and improved analysis of system performance and limitations need to be generated for these schemes as well. A major limitation of many adaptive algorithms is the computation time required for each iteration. Relatively few of them have been implemented in practice on a manipulator, and even in simulation, the relative complexity of these algorithms makes them difficult to evaluate.

2.2.3 Robust Solutions

Another approach to the problem of parameter variation involves the development of control strategies which are insensitive to parameter and error variation. These algorithms are based on the premise that if a controller is designed to be robust with respect to modeling error, relatively little must be known about the plant in order to achieve accurate control. Errors in modeling parameters are lumped along with other disturbances, which the system is designed to reject. Robust control algorithms may even be designed to reject disturbances with no knowledge of the plant, though they are typically augmented with at least a simplified feedforward-based model to reduce the magnitude of the error they must cope with.

Many of the robust schemes proposed are based on multivariable design. For example, Desa and Roth [Desa 85] suggest using a control system derived from multivariable, robust servomechanism theory. Though they claim that the design is meant to be insensitive to changes in the plant and controller parameters, the paper starts with the following three assumptions, the links are rigid, all of the states are available, and there are no torque limits. They also assume that they have good models both of the plant and the nature of the disturbances. As such, the algorithm is not very robust, and is computationally expensive.

There is growing interest in using algorithms based on variable structure systems for manipulator control. A review of variable structure and sliding mode system theory may be found in [Utkin 77]. In general, these schemes are state feedback algorithms, in which the control can switch at any instant from one to another of a member of a set of continuous functions of the state. The problem is that of defining the set of possible control functions and selecting the switching logic to choose among them. An advantage of variable structure systems is that they allow the combination of useful properties of each of the individual functions Furthermore, the combination of schemes may create a system that possesses properties not found in any of the components, and in the extreme can combine a set of unstable functions into an overall stable system The state of the system during a phase in which its trajectory describes a motion not inherent in any of the component functions is called the sliding mode, and is the principal advantage of such schemes. Once in this mode, the system is relatively insensitive to variations in plant parameters and other disturbances.

Morgan and Ozguner [Morgan 85] proposed a control algorithm based on variable structure systems. They point out the deficiencies of two popular approaches, namely precise modeling and design of special purpose manipulators, and suggest a variable structure controller for manipulator position control. Since sliding mode controllers are said to be robust to parameter variation once in the sliding mode, no system model is necessary for control, and the inherent nonlinearities in the system are treated as disturbances. The problems with sliding mode schemes such as this one are in reaching the sliding mode from the initial state, and chattering at the switching surface once in the sliding mode. The authors suggest feeding forward a disturbance term based on a simplified model, and low pass filtering the disturbance signal. Thus, though the controller is designed to work without a model, at least a simplified model is used to augment its performance.

A further development of a sliding mode controller is found in [Slotine 86]. The principal difference between this controller, which the author calls a suction controller, and other sliding mode schemes is in the use of a saturation function around the switching surface instead of a sign function. Thus, instead of jumping from -1 to 1 around the switching surface, the controller moves in a line between those two points. This, in theory, should reduce the effects of chattering around the switching surface.

Robustness issues are of extreme importance in the control of robot manipulators. The dynamic parameters of the manipulator are often difficult to obtain and are typically subject to considerable error. The structure of manipulator dynamics is inherently both nonlinear and time varying, and any control algorithm designed for manipulator position control must provide a means for overcoming these variations. Nonlinear, multivariable, or sliding mode control algorithms therefore try to incorporate robustness directly into the design of the controller, rather than eliminating the disturbance terms explicitly, as in the case of adaptive and classical schemes.

The form of the equations describing manipulator dynamics has been shown to be highly nonlinear and coupled. Most manipulators use local servocontrollers, which disregard the dynamics, and treat dynamical effects as a disturbance which must be rejected. Vukobratović [Vukobratović 83] suggested that controllers of this form allow adequate performance only at low speeds and seriously degrade at higher velocities and accelerations. He further made the claim that the solution to this problem must involve compensation for dynamic effects which must be calculated globally for the entire manipulator While a variety of schemes have been put forth which indeed treat the problem in this manner, the solutions they offer are typically computationally expensive and depend on the accuracy of the model. In contrast, the acceleration feedback theory proposed by Studenny [Studenny 83], [Studenny 84], [Studenny 86a], [Studenny 86b], [Studenny 87], involves only local controllers, but due to the nature of the dynamics, provides some compensation for dynamics effects directly within the feedback loop The following sections will present the theory and discuss important issues such as stability, performance, and limitations

3.1 Presentation of the Theory

Acceleration feedback is a simple control law which is applied locally at the joint level, and which does not exact a high computational burden. It was first applied to manipulator control by Luo and Saridis [Luo 82], who showed that the control law is optimal in an LQ sense, however required that all the nonlinearities be removed by feedforward compensation. Studenny's work involved removing the feedforward terms, and demonstrating that the control law is stable, robust, and approaches optimality.

3.1.1 Single Joint Application

Recall the dynamical equation for a manipulator joint:

$$J(q)\ddot{q} + C(q)\dot{q}^2 + G(q) = u \tag{3.1}$$

$$\text{where } J(q) = inertia$$

$$C(q)\dot{q}^2 = Coriolis\ force$$

$$G(q) = gravity\ force$$

$$u = controlled\ torque\ input$$

$$\text{and } q = joint\ position\ in\ joint\ space$$

The acceleration feedback law is

$$u = -K_u[(\ddot{q} - q_d) + a_1(\dot{q} - \dot{q}_d) + a_0(q - q_d)] \tag{3.2}$$

$$\text{where } K_u = high\ gain\ feedback$$

$$a_0, a_1 = desired\ gains$$

$$\text{and } q_d = desired\ position$$

Defining quantities:

$$\Delta q = (q - q_d) \tag{3.3}$$

$$\Delta \dot{q} = (\dot{q} - q_d) \tag{3.4}$$

$$\Delta \ddot{q} = (\ddot{q} - \ddot{q}_d) \tag{3.5}$$

we can rewrite the above equation in the form:

$$u = -K_u[\Delta\ddot{q} + a_1\Delta\dot{q} + a_0\Delta q] \tag{3.6}$$

Setting the dynamics equal to the control law, we arrive at

$$J(q)\ddot{q} + C(q)\dot{q}^2 + G(q) = -K_u[\Delta\ddot{q} + a_1\Delta\dot{q} + a_0\Delta q] \tag{3.7}$$

Thus, solving for $\Delta\ddot{q}$, we get

$$\Delta\ddot{q} = -a_1\Delta\dot{q} - a_0\Delta q + \frac{1}{J(q) + K_u}[J(q)a_0\Delta q + J(q)a_1\Delta\dot{q} + G(q) - J(q)\ddot{q}_d + C(q)\dot{q}^2] \tag{3.8}$$

If K_u is made arbitrarily large, this equation reduces to

$$\Delta\ddot{q} = -a_1\Delta\dot{q} - a_0\Delta q \tag{3.9}$$

and the nonlinearities are overcome by the high gain feedback. Thus, the parameters a_0 and a_1 are used to determine system performance, and the overall gain K_u is used to decouple and linearize the system. The major difference between this controller and previous similar formulations is this high gain K_u, without which the system performance remains inertia dependent.

3.1.2 Extension to Multiple Joints

The controller discussed in the previous section has been extended to the multiple joint case. The proof of stability, which will be presented in a later section, was first developed for the single joint case, and was extended to the multiple joint case by Holder's inequality and norm arguments. The main difference between the single joint formulation and the multiple joint formulations appear in the dynamical equations. Now the inertial terms are a function of all of the joints, and the terms in the dynamical equations are n x n matrices. The dynamics now appear as follows:

$$J(\underline{q})\ddot{\underline{q}} + QC(\underline{q})\dot{\underline{q}}^2 + \underline{G}(\underline{q}) = \underline{u} \tag{3.10}$$

$$\text{where } J(\underline{q}) = \text{inertia}$$

$$\dot{Q} = \begin{pmatrix} \dot{\underline{q}}^T & \underline{0}^T & \dots & \underline{0}^T \\ \underline{0}^T & \dot{\underline{q}}^T & \dots & \underline{0}^T \\ \vdots & \vdots & \ddots & \vdots \\ \underline{0}^T & \underline{0}^T & \dots & \underline{q}^T \end{pmatrix} \tag{3.11}$$

$$\dot{Q}C(\underline{q})\dot{\underline{q}}^2 = Coriolis\ force$$

$$\underline{G}(\underline{q}) = gravity\ force$$

$$\underline{u} = controlled\ torque\ input$$

$$and\ \dot{\underline{q}} = joint\ position\ in\ joint\ space$$

The control law remains the same as in the single joint case.

$$\underline{u} = -K_u[\Delta\ddot{\underline{q}} + A_1\Delta\dot{\underline{q}} + A_0\Delta\underline{q}]. \tag{3.12}$$

with the exception that

$$A_1, A_2 = LQ\ designed\ gain\ matrices$$

$$K_u = positive\ definite\ diagonal\ gain\ matrix$$

Thus, by this design, each joint is controlled by its own controller, which is independent of configuration and of the other joints. Note that the nonlinear terms still disappear due to the high frequency gain K_u, and that the system performance is determined by A_0 and A_1 alone.

3.1.3 Discrete Version

Since control of the manipulator is performed on a digital computer, it is necessary to review the theory in light of the constraints imposed by the discrete time system. Two of the major drawbacks of a digital implementation of control laws involve quantization error and time lag. Conversely, the flexibility of digital implementation makes it very attractive.

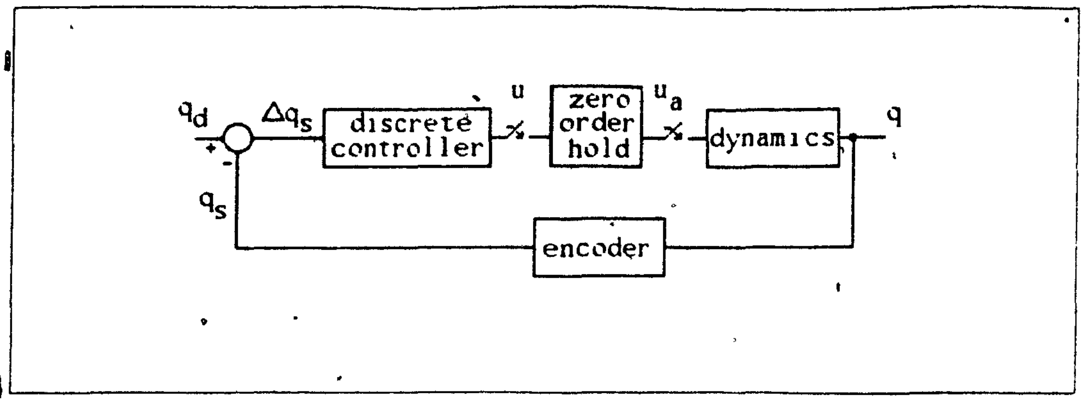

Figure 3.1 Sampled Data Manipulator Position Control System

In the digital version of the theory, it is assumed that only position data is available, and therefore velocity and acceleration values must be calculated using second order numerical differentiators. The entire system appears as follows:

The hold device we are using is a zero order hold, which must therefore be added to the control equation. Given the continuous time system

$$u = -K_u[s^2 + A_1 s + A_0] \tag{3.13}$$

convert it to digital form with the following substitutions:

$$s => D^1(z) = \frac{(3 - 4z^{-1} + z^{-2})}{2T_s} \tag{3.14}$$

$$s^2 => D^2(z) = \frac{(1 - 2z^{-1} + z^{-2})}{T_s{}^2} \tag{3.15}$$

where

$$z^{-1} = e^{-sT_s} \tag{3.16}$$

and T_s is the time between sampling intervals.

Thus, in discrete form, the acceleration feedback law appears as follows:

$$u = -K_u(z)[D^2(z) + A_1 D^1(z) + A_0]\Delta q_s \tag{3.17}$$

Note that in this case, Δq_s is a quantized position error, which is dependent on the quantization of the encoders of the manipulator joints, and must by necessity exhibit a time delay of one sampling interval.

3.2 Discussion of Stability

The mathematical approach used by Studenny [Studenny 87] to prove stability of the acceleration feedback control law was based on Lyapunov stability theory. In this section it is intended to explain Lyapunov stability analysis, and examine the way it was applied to acceleration feedback control.

3.2.1 Lyapunov Stability Theory

Lyapunov stability analysis is used to determine the stability of disturbed non-linear systems which can not be analyzed using traditional methods. The Lyapunov direct method is based on the idea that the rate of change of the energy of a system is an indicator of its stability [Casti 85]. To understand the meaning of this statement it is first necessary to establish the concept of stability and to define what is meant by energy

Stability is defined with respect to disturbed control systems A definition of stability therefore requires first a mathematical description of a control system, a description of disturbances, and a means to describe the effect of such disturbances on the system

To describe the stability of systems, Letov [Letov 61] introduces the following definitions

A state space description of an undisturbed system is

$$\frac{dx^*_k}{dt} = X_k(x^*_1 \ldots x^*_n) \tag{3.18}$$

A change of variables is made to account for disturbances:

$$x_k = x_k^* + y_k \tag{3.19}$$

The new disturbed system may therefore be modeled as:

$$\frac{dy_k}{dt} = Y_k(y_1 \ldots y_n) \tag{3.20}$$

in which

$$Y_k(y_1 \ldots y_n) = X_k(x_1{}^* + y_1 \ldots x_n{}^* + y_n) \tag{3.21}$$

The disturbances are caused by the y_k terms, and when these equal zero we return to the undisturbed system. The first definition of stability given by Letov is that of Bounded-Input-Bounded-Output (BIBO) stability applied to disturbances. Given a set of disturbances $y_{k0} = (y_{10} \ldots y_{n0})$ which produce a disturbed motion $y_k = y_k(y_{10} \ldots y_{n0}, t)$, the system is called stable if for all y_k such that $\|y_{k0}\| < \eta$, the disturbed motion will satisfy $\|y_k(t)\| < \epsilon$ for all $t > 0$, where $\eta = \eta(\epsilon)$ and ϵ are some positive constants smaller than infinity. Restated, this means that for a bounded disturbance the resulting disturbed motion is also bounded.

A geometric interpretation of this result is also provided. Undisturbed motion is called stable with respect to y_k if for any positive number A it is possible to choose another number $\lambda(A)$ such that for all disturbances y_{k0} satisfying $\sum {y_{k0}}^2 <= \lambda$, the disturbed motion y_k satisfies $\sum y_k(t)^2 < A$ for all $t > 0$. Furthermore, it is now possible to define regions of stability described by A and λ. If in addition to the above conditions, as t approaches infinity, the system satisfies

$$lim\ y_k(t) = 0 \tag{3.22}$$

the system is said to be asymptotically stable. Graphically, it is possible to define a region and a subregion such that every disturbance bounded within the region results in an output bounded by the subregion. Asymptotic stability further stipulates that this subregion shrinks to a point, usually taken as the origin, as time approaches infinity.

Having described stability and asymptotic stability, an explanation of Lyapunov's direct method may be presented. The goal of this analysis is to determine the existence of λ satisfying the above conditions and to calculate the size of the region defined by $\sum y_{k0}{}^2 <= \lambda$, in which undisturbed motion is assured. The Lyapunov direct method involves the calculation of functions $V = V(y_1 \ldots y_n)$, called Lyapunov functions, whose total derivatives with respect to time have certain properties which assure stability.

These Lyapunov functions may be thought of as generalized energy functions. In fact, for physical systems, very often the total energy of the system is the function chosen for the role of Lyapunov function. In such cases, the physical meaning of a stable equilibrium point is one in which the energy is at a local minimum. Thus the total time derivative of the energy function of a physical system is always negative near a stable equilibrium point. For many systems it is impossible to define a meaningful energy function. In such cases the Lyapunov function is not the energy of the system but may be considered a generalized energy function in that its derivative with respect to time is an indicator of the stability of the system.

V is called sign invariant if it does not change sign over the entire region on which it is defined. If in addition, it assumes zero values only at the origin, it is termed definite. Finally, if the sign of V is positive, then V is said to be positive definite. Lyapunov stability theory assumes that V is always chosen to be positive definite. If V is positive definite, then the equation V = C = constant represents a family of closed curves. If this constant C is decreased to zero, then the region contracts to a point at the origin. Thus the curves defined by V = C intersect all paths leading from the origin to infinity.

Lyapunov's first theorem states that the disturbed system,

$$\frac{dy_k}{dt} = Y_k(y_1 \ldots y_n) \tag{3.23}$$

is stable if it is possible to find a positive definite function $V = V(y_1 \ldots y_n)$ whose total time derivative $\frac{dV}{dt} <= 0$. Note that by virtue of the definition of $\frac{dV}{dt}$, i.e.

$$\frac{dV}{dt} = \sum \frac{dV}{dy_k}\frac{dy_k}{dt} \tag{3.24}$$

the sign of $\frac{dV}{dt}$ is determined by the Lyapunov function and the original system.

Lyapunov's second theorem states that given the system and Lyapunov function, described above, $\frac{dV}{dt} < 0$ implies asymptotic stability, since now $\frac{dV}{dt}$ can vanish only at the origin and the disturbed system tends towards the undisturbed system.

Another statement of Lyapunov's direct method is made possible using the state space description of the system directly [Willems 76] Given a system described by $\dot{x} = Ax$, with x an $n \times 1$ column vector and A an $n \times n$ matrix, and given any matrix C such that (A,C) is observable, there exists a positive definite symmetric solution Q to the Lyapunov equation.

$$A^T Q + QA = -C^T C \tag{3.25}$$

We recall here that observability means that the rank of the matrix

$$(C^T, C^T A^T, \ldots, C^T A^{T^{n-1}}) = n \tag{3.26}$$

Furthermore, if the system Σ is asymptotically stable, this description yields a method for constructing quadratic Lyapunov functions $V(x) = \frac{1}{2}x^T Qx$ for Σ. Thus, describing a system in state space, and finding a matrix C for which (A,C) is observable not only provides a way of determining the stability of the system, but allows the construction of the Lyapunov function as well, often the most difficult step in Lyapunov stability analysis. Note that Lyapunov's direct method gives only sufficient conditions for stability. This means that the time derivative of V determines whether the system is stable, but does not indicate instability. Since $\frac{dV}{dt}$ is a function of both the system and an arbitrarily chosen Lyapunov function, failure to determine stability may indicate either that the system is unstable or that the Lyapunov function chosen is unsuitable for determining the stability of this particular system. This is a major drawback of Lyapunov stability theory, however the analysis may still be applied successfully to a variety of problems, including, for example, acceleration feedback control of robot manipulators.

3.2.2 Stability of Acceleration Feedback

The problem at hand relies on combination of the acceleration feedback control law and the equation governing robot manipulator dynamics. To be compatible with Lyapunov stability analysis, the system is described using a state space formulation under rigid body assumptions

Before stability can be determined for this system, it is convenient to formulate it as an LQ problem [Anderson 71]. This is a linear quadratic form of the optimal control problem involving a system and a performance criterion. The goal is to control the system while minimizing the performance criterion. Given a system:

$$\dot{z} = Az + Bu \tag{3.27}$$

where

$$z = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \tag{3.28}$$

$$A = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix} \tag{3.29}$$

$$B = \begin{pmatrix} 0 \\ I \end{pmatrix} \tag{3.30}$$

and a performance index

$$J = \int (z^T Qz + u^T u)dt \tag{3.31}$$

Define Q to diagonalize A_0 and A_1, and choose u to minimize J. If we define $y = \dot{u}$, we get the solution to the LQ problem as

$$u = -Kz \tag{3.32}$$

where

$$K = B^T P = [A_0, A_1] \tag{3.33}$$

B and P are found by solving the Riccati equation,

$$A^T P + PA - PBB^T P + Q = 0 \tag{3.34}$$

Note the similarity to the Lyapunov equation described earlier. This makes the LQ formulation extremely convenient for Lyapunov stability analysis.

The closed loop equation is therefore

$$\dot{z} = (A - BK)z \tag{3.35}$$

and in terms of y,

$$\ddot{y} = -A_0 y - A_1 \dot{y} \tag{3.36}$$

Thus, in formulating the acceleration feedback control law as an LQ problem we provide a natural basis from which to carry out Lyapunov stability analysis. Furthermore, using the matrix formulation of the Lyapunov equation, the selection of a Lyapunov function for the system is also simplified.

Turning now to the application of Lyapunov stability analysis to the closed loop robot manipulator system, we note that the system may be described by the following equation:

$$\Delta\dot{z} = (A - BK)\Delta z + n \tag{3.37}$$

where

$$\Delta z = \begin{pmatrix} \Delta q \\ \Delta\dot{q} \end{pmatrix} \tag{3.38}$$

$$A - BK = \begin{pmatrix} 0 & I \\ -A_0 & -A_1 \end{pmatrix} \tag{3.39}$$

$$n = \begin{pmatrix} 0 \\ f \end{pmatrix}. \tag{3.40}$$

The Lyapunov function for this system is defined as

$$V\Delta z = \Delta z^T P \Delta z \tag{3.41}$$

Note that for this function

$$P_{min}\|\Delta z\|^2 <= V(\Delta z) <= P_{max}\|\Delta z\|^2 \tag{3.42}$$

$$P_{min} = \lambda_{min}(P) \tag{3.43}$$

$$P_{max} = \lambda_{max}(P) \tag{3.44}$$

and $\lambda_i(P)$ is the i^{th} eigenvalue of P.

We now recall that in performing a Lyapunov stability analysis, we study the behavior of $\dot{V}(\Delta z)$; in particular we wish to find conditions for and the region in which $\dot{V}(\Delta z) < 0$. From the definition of the LQ problem and the equation of motion, it is possible to arrive at the formula

$$\dot{V}(\Delta z) = -\Delta z^T(Q + K^T K)\Delta z + 2\Delta z^T P n \tag{3.45}$$

Imposing the constraint that $\dot{V} < 0$, we get

$$\Delta z^T[Q + K^T K]\Delta z > \|2\Delta z^T P n\| \tag{3.46}$$

This further implies that

$$\Delta z^T[Q + K^T K]\Delta z > 2\|P_{12}\Delta q + P_{22}\Delta \dot{q}\|\|P\|\|f\|q \tag{3.47}$$

This leads to an equation of $\dot{V} < 0$, where $\dot{V}$ is a function of the acceleration feedback law parameters, the robot manipulator dynamical equation, and K_{min}, where

$$K_{min} = \lambda_{min}(Q + K^T K) \tag{3.48}$$

A full description of this function may be found in [Studenny 87]. For our purposes here it is sufficient to establish that this function exists and that stability can be established if the matrices are chosen in such a way that K_{min} renders the function negative. The choice of matrices also establishes the region in which the above stability condition holds.

Thus it has been shown that it is possible to obtain a closed loop system for robot manipulator control using acceleration feedback. and the the stability of the system may be proven using a Lyapunov function. This system is therefore stable for all Δz within certain limits defined above.

3.3 Theoretical Predictions

It is predicted that PD and Acceleration feedback will perform similarly under conditions of small disturbances and low bandwidth. At higher frequencies and under greater disturbances, it is expected that acceleration feedback will continue to perform well beyond the range in which PD begins to fail. Due to the limitations imposed by the sampling interval of 14 msec, neither scheme is expected to provide extremely rigid control under heavily disturbed conditions The limitations brought about by the sampling interval include a ceiling on gains, a restriction on filter frequency, and a smaller operating bandwidth. Further restrictions on these parameters are caused by the torque limits on the motors, as well as velocity and acceleration limits set by the RCI system

3.3.1 Stability

The system has been simulated extensively by Studenny [Študenny 87]. Though the manipulator used in that simulation study was a Unimation PUMA 600, the results should apply to the PUMA 260 as well, which has the same architecture.

From the discussion in [Studenny 87] and from the simulation results presented therein, the following predictions can be made concerning the stability of the system. The closed loop system is expected to be stable for any K_u under ideal conditions, however the unstructured high frequency disturbances inherent in the system are expected to cause instability for a large K_u Since a high K_u is necessary for good decoupling of the system, it is recommended to use a low pass filter to reduce the effects of the high frequency uncertainties. Thus, there is a gain bandwidth product limitation which must be taken into account in designing the compensator.

The digital implementation of the acceleration feedback algorithm places several other restrictions on stability The sampling interval at which the system operates determines the highest gain at which the system will be stable This is apparent from the discrete Nyquist analysis in [Studenny 87], in which it is also stated that the minimum operating sampling frequency is on the order of 80 Hz Due to the Nyquist criterion, the bandwidth of the system is limited as well For example, a system whose sampling interval is 72 Hz is expected to show aliasing at 36 Hz, though degradation of the signal may become apparent at much lower frequencies Typically, conservative use of the system would allow frequencies one order of magnitude below the Nyquist rate, and in this case would allow signals of up to about 4 Hz Another discrete effect is that of quantization, which is a measure of the precision of the encoders of the joints in terms of encoder counts per radian. Poor quantization has a very severe effect on performance, and may lead to instability The nominal quantization for the PUMA is 5.5×10^{-4}, which is shown in simulation to be stable.

Friction and disturbances may cause performance degradation, but are shown in simulation not to cause instability Thus, the system is expected to perform in a stable manner, provided the gain bandwidth limitation is adhered to, the sampling frequency is above the minimum required, and a low pass filter is used to compensate for the high frequency unstructured disturbances

3.3.2 Performance

The performance analysis for the ideal case examined two factors: the magnitude of the high gain feedback K_u, and friction. Under ideal conditions performance is expected to improve with rising K_u, and the required torque to sustain this performance is not expected to rise with K_u. Friction is not expected to be a problem under nominal conditions, though extremely high values of static friction are expected to cause sticking and spikes. For the nonideal case, the unstructured uncertainties are not expected to cause serious degradation in performance, provided that a low pass filter is included in the compensator.

The discrete time implementation impacts performance in several ways. The sampling interval does not cause variation in performance, provided that the rate is fast enough to ensure stability for the given compensator. Increasing the sampling rate significantly beyond the minimum required may even cause a degradation of performance, as the effects of rounding errors become more severe. A faster sampling rate does impact performance only in that it allows higher gains to be used, thereby allowing a stiffer system. The sampling interval used in this research is 72 Hz, limiting the magnitude of the gain K_u to about the minimum gain at which the acceleration feedback law improves performance over PD and indicates that the system is likely not to be very stiff. Quantization is directly related to performance, and the quantization of the PUMA is expected to provide reasonable tracking, without serious degradation in performance.

Friction affects performance in exactly the same manner as in the continuous time formulation, and only excessive values of stiction are expected to cause degradation in the form of spikes. The nominal values of friction for the PUMA should not cause this effect. For a system with appropriate gains and low pass filtering, unstructured disturbances are not expected to cause serious degradation in performance. Thus, it is expected that the system will track a trajectory with small error, under realistic conditions and in a digital implementation.

3.4 Simulation

The simulation results in [Studenny 87] do not compare the performance of acceleration feedback with that of PD control under similar conditions. To that end, simulation experiments were carried out as part of this research, for a two link manipulator representing joint 2 and joint 3 of the PUMA 260 manipulator. The simulation experiments were conducted using the ACSL (Advanced Continuous Simulation Language) [ACSL 81] package and were based on simplified dynamics. The purpose of this simulation was to provide information about predicted performance of the two controllers for situations resembling those to be tested in practice on the manipulator.

Four sets of simulation experiments were carried out. In the first, acceleration feedback and PD controllers were compared for the case of a single joint, in the presence of simplified dynamics The controllers were tuned to provide similar position and velocity dependent responses, and the test was in the effect of acceleration on the trajectory following. In the second set, a second joint was added to provide a coupling disturbance, with a PD controller. The magnitude of the disturbance was increased to the point in which the effect on trajectory following became noticeable. The third set of experiments once again compared the performance of PD and acceleration feedback, however this time under heavily coupled disturbed conditions. The final set of experiments showed the effect of sampling interval on the digital implementation of the acceleration feedback algorithm. All of these experiments were conducted using a very simplified model of the dynamics, and are not intended as a comprehensive simulation of the manipulator The simulation is of a two link manipulator designed to roughly resemble joint 2 and joint 3 of the PUMA 260 in size and performance, and the experiments were carried out at a sampling interval of 14 msec. The purpose of this simulation is to provide a framework with which to predict the results of the comparative experiments found in chapter six. For a comprehensive simulation, the reader may refer to [Studenny 87]. The units on the graphs are all in radians, for position, and seconds for time.

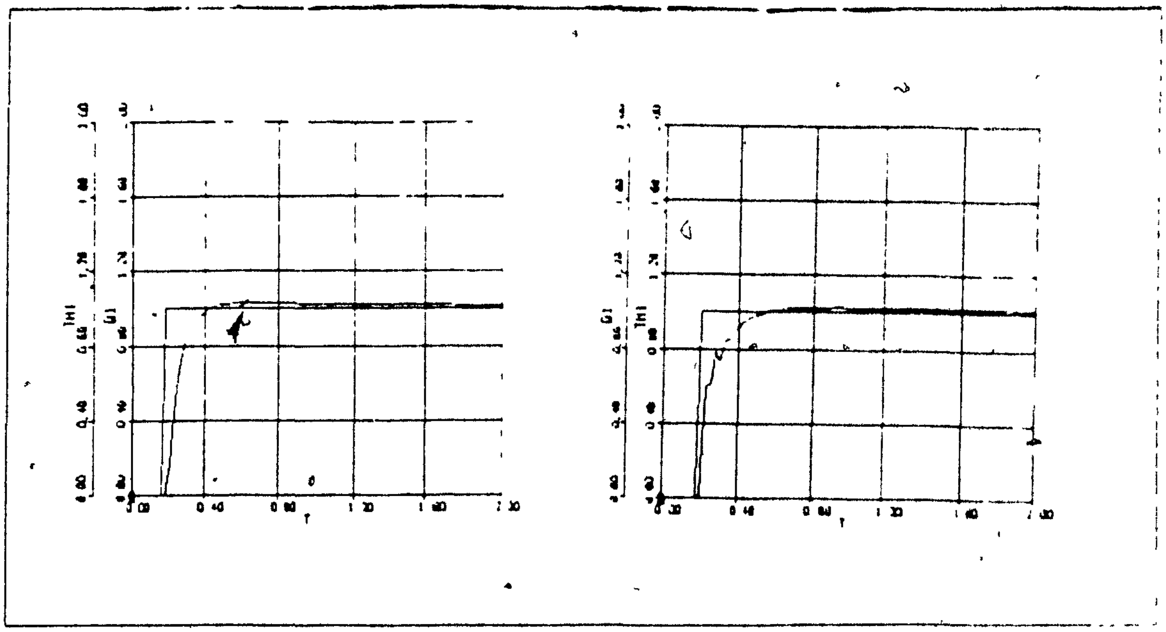

Figure 3.2a Step Response PD (left). Acceleration Feedback (right)

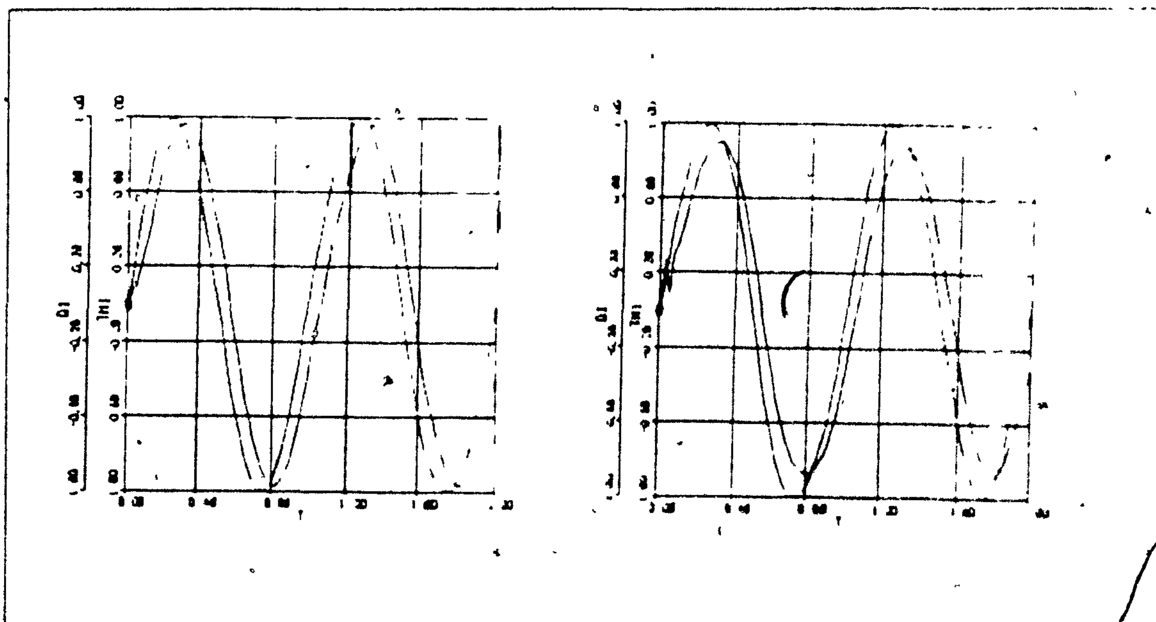

Figure 3.2b Sinusoidal Tracking PD (left). Acceleration Feedback (right)

Figure 3.2 PD vs Acceleration Feedback. Single Joint

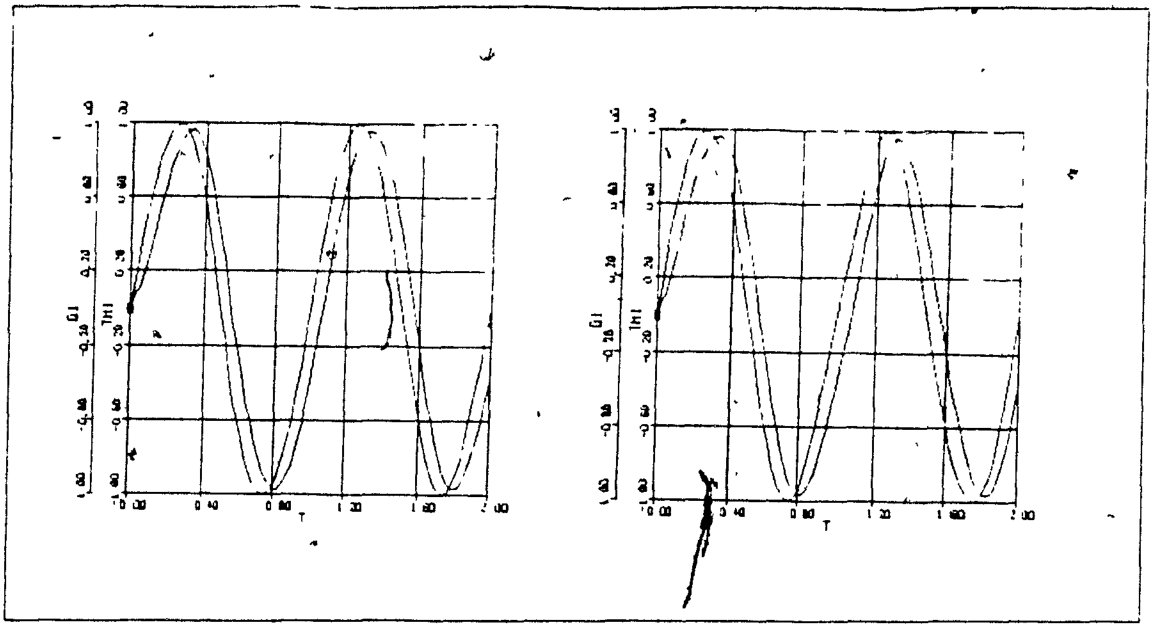

Figure 3.3a No Cross Coupling (left). Small Cross Coupling (right)

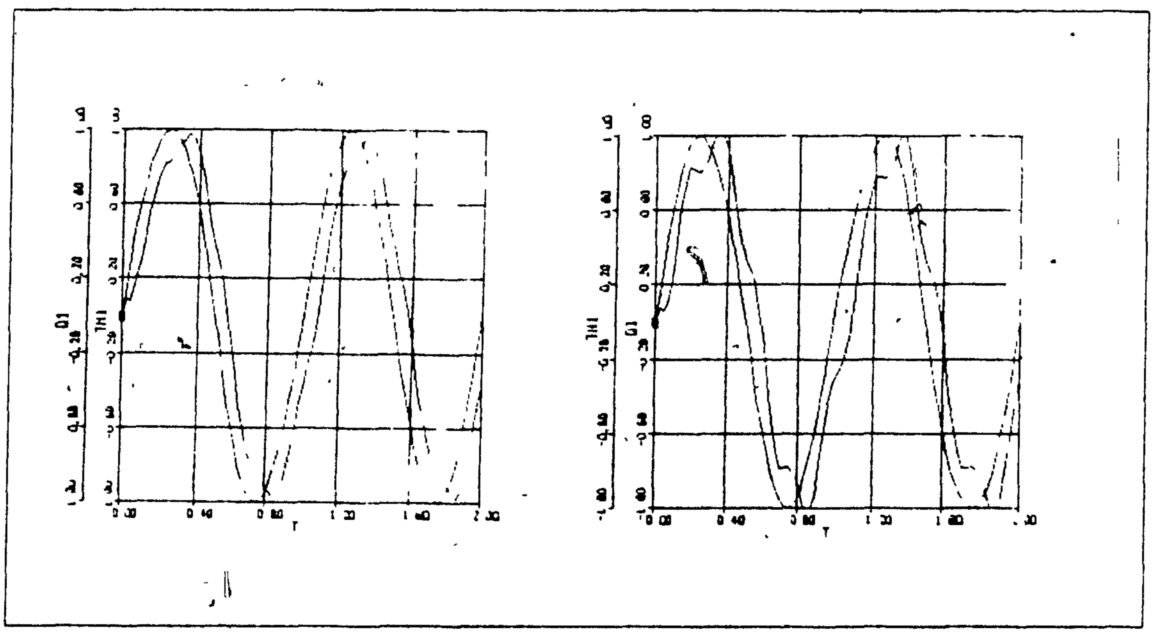

Figure 3.3b Significant Cross Coupling (left). Severe Cross Coupling (right)

Figure 3.3 The Effect of Coupling Disturbance

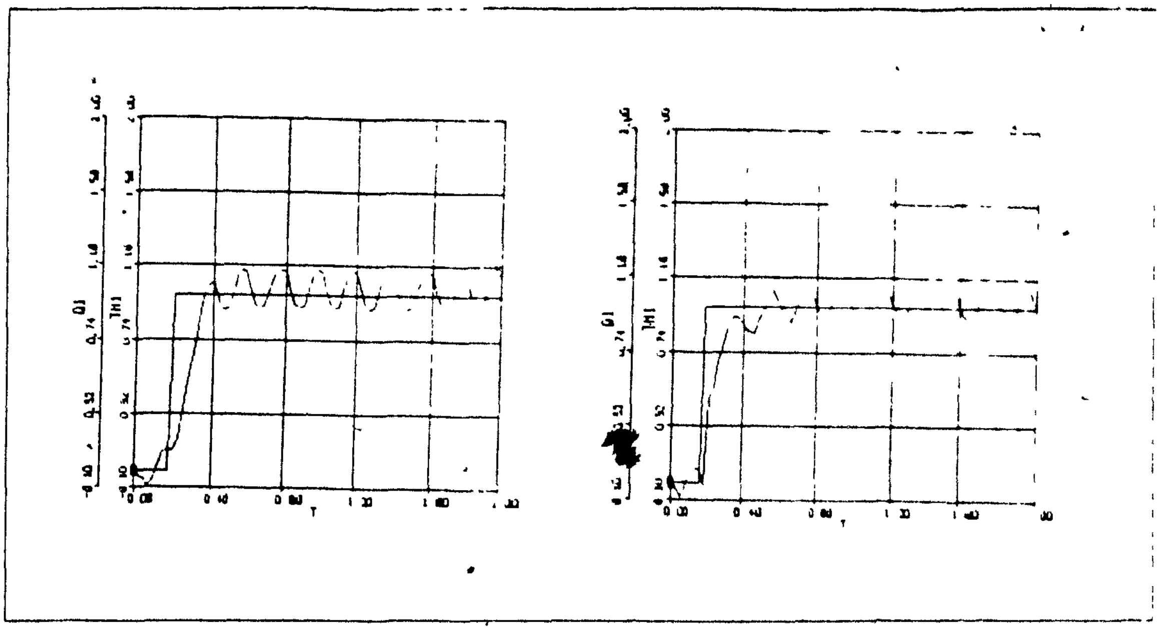

Figure 3.4a Step Response PD (left), Acceleration Feedback (right)

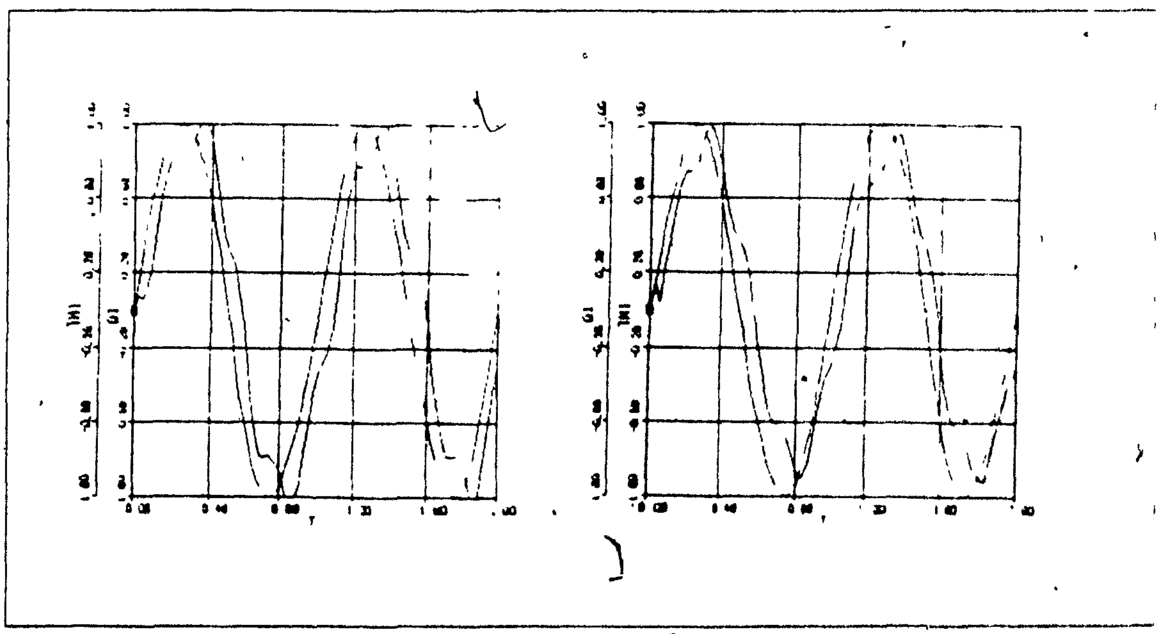

Figure 3.4b Sinusoidal Tracking PD (left), Acceleration Feedback (right)

Figure 3.4 PD vs Acceleration Feedback in the Presence of Cross Coupling Inertial Effects

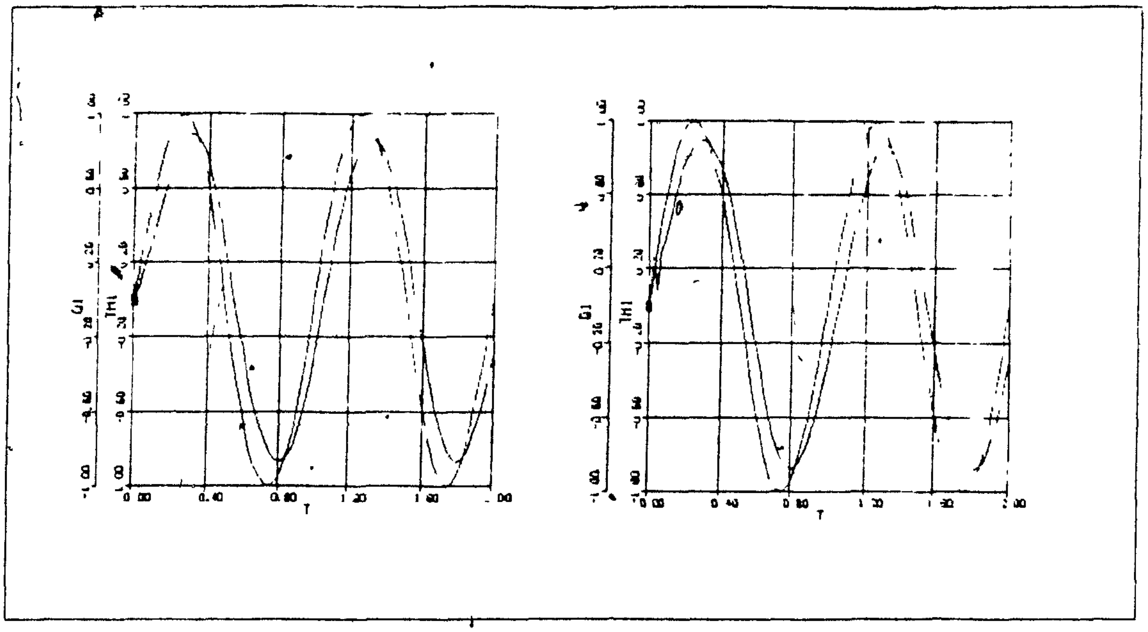

Figure 3.5a $T_s = 7msec$ (left), $T_s = 14msec$ (right)

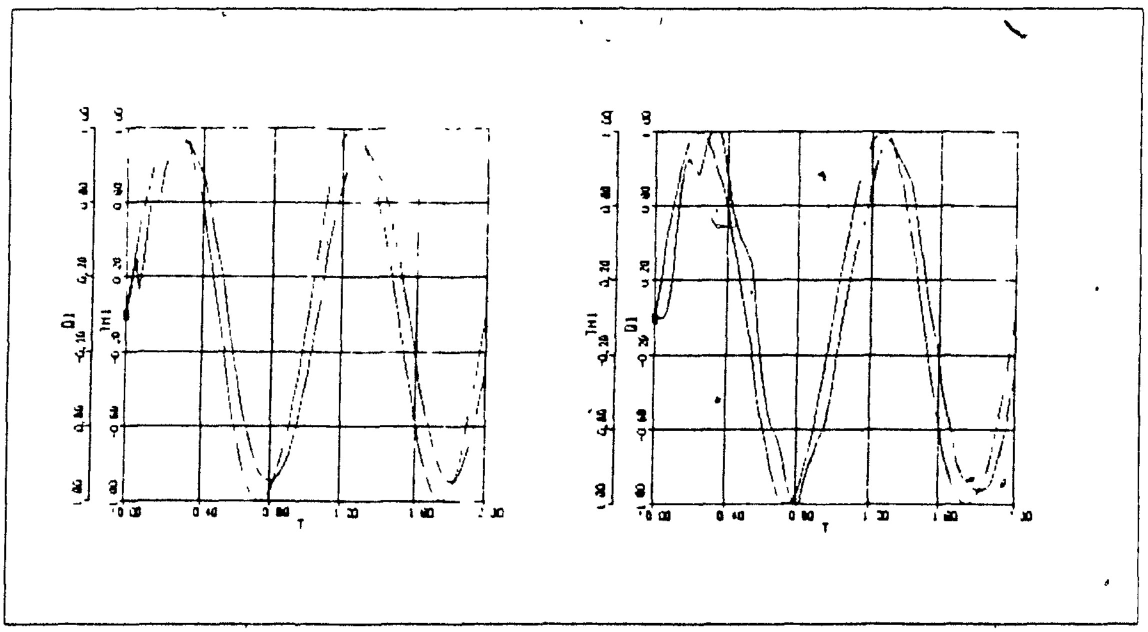

Figure 3.5a $T_s = 28msec$ (left), $T_s = 56msec$ (right)

Figure 3.5 The Effect of Sampling Interval

The simulation results presented in the preceding figures lead to the following predictions of performance. For the case of a single joint, the performance of acceleration feedback is expected to be similar to that of PD. PD may even be somewhat smoother, as it requires only one differentiation. A heavy cross coupling disturbance is demonstrated to have a severe degrading effect on trajectory following. Whether these effects are severe in practice on the PUMA 260 remains to be seen. In the presence of cross coupling effects, acceleration feedback is shown to give a small performance improvement over PD. Finally, the sampling rate is shown to have a degrading effect on system performance, and has been shown to lead to instability or force a lowering of the maximum allowable controller gains.

This chapter begins with a description of the hardware and software environments present at the start of this research The final section of this chapter describes the testbed, which is an addition to the environment This addition allows the testing and evaluation of position control algorithms on a manipulator.

4.1 Hardware Environment

The hardware environment for this research consists of a Unimation PUMA 260 manipulator controlled by a VAX-11/750 computer through an LSI-11/03 computer. Many of the components of the system have been modified to allow real time control and data acquisition from the manipulator by use of the RCCL/RCI software environment. This research did not require system modifications beyond those already performed to run RCCL and RCI.

4.1.1 The Puma 260 Manipulator

The heart of the research involves the manipulator. The PUMA 260 is a six degree of freedom, 3-R, wrist partitioned, anthropomorphic, general purpose manipulator. The joint actuators are permanent magnet DC servo motors. Each joint is controlled by a microcontroller, which is a microprocessor using optical encoder position feedback. The kinematic equations and solutions of the PUMA 260 used in this research may be found in [Lloyd 84]. These solutions are based on the conventions established by Hartenberg and Denavit, which were briefly described in an earlier section. A diagram of the PUMA 260 is shown in figure 4.1, and it's kinematic parameters are given in table 4.1.

Joint i	d_i (mm)	θ_i (deg)	α_i (deg)	a_i (mm)
1	0	θ_1	90	0
2	0	θ_2	0	203.20
3	126.24	θ_3	-90	0
4	203.20	θ_4	90	0
5	0	θ_5	-90	0
6	0	θ_6	0	0

Table 4.1 Kinematic Parameters for the PUMA 260 [Lloyd 84]

4.1.2 Other Equipment

The LSI-11 is the standard Unimation controller for the PUMA 260. Normally, it runs the VAL language, by which the robot is controlled. To allow the robot to be controlled via a VAX minicomputer [Carayanis 83], this software has been replaced with a monitor which passes information from the robot to the VAX and back. The communication time involved is significant, measuring about 7 msec, and places a restriction on the complexity of implemented control algorithms. The VAX-11/750 is a multiuser minicomputer located on a large ethernet network. The system loads both on the VAX running the control software and on the network are important considerations in planning the feasible tasks at a given session.

4.2 Software Environment

The robot programming environment used in this research is based on the C

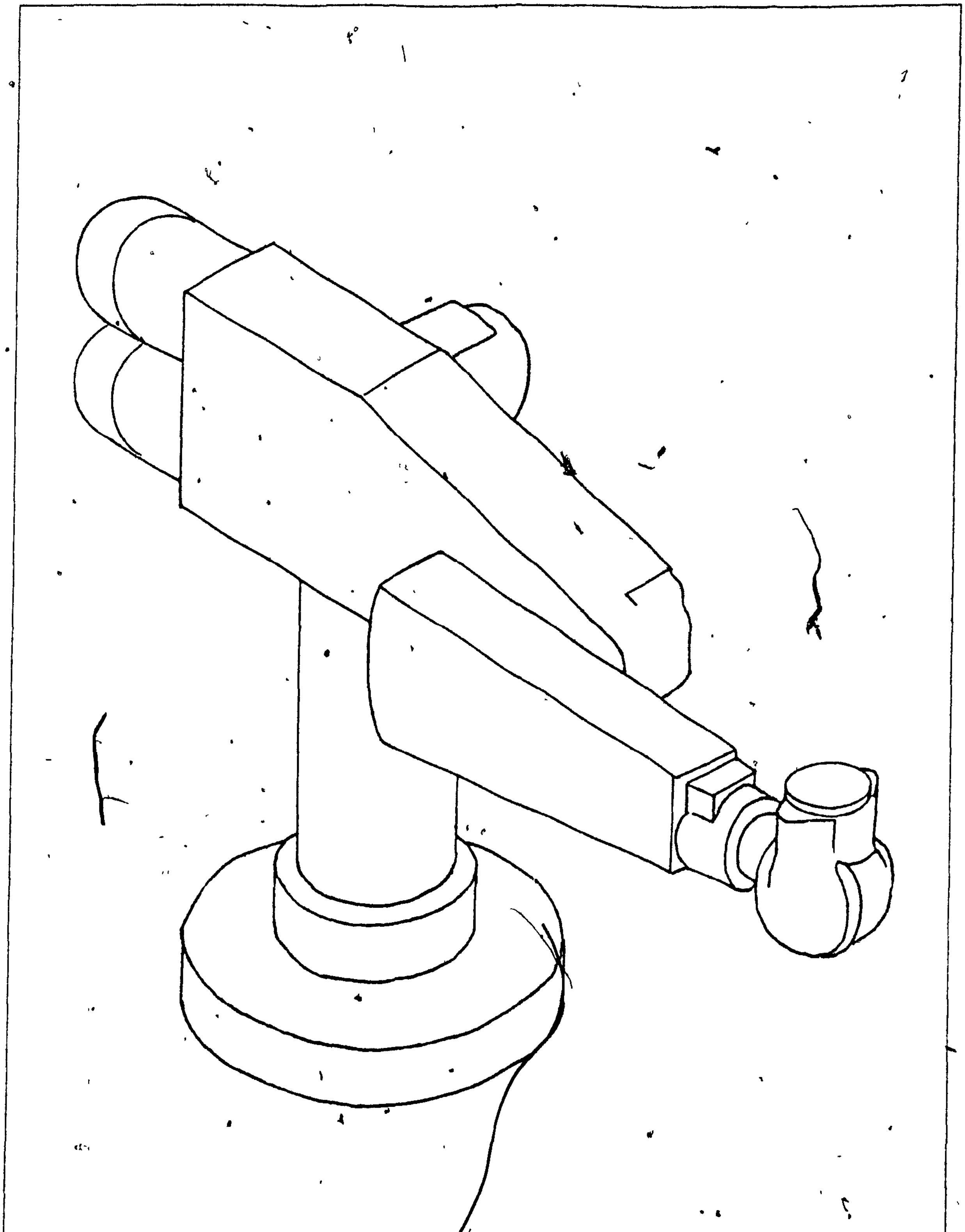

Figure. 4.1 The PUMA 260 [Lloyd 85]

programming language, and runs under the UNIX operating system. The environment consists of two layers: RCI, and RCCL. RCI (Robot Control Interface) is the low level of the environment, and the one most relevant for this research. RCI is a software facility for creating real time robot control procedures in the C language. RCCL is a layer built on top of RCI, which provides trajectory generation facilities and allows control of the robot in Cartesian coordinates. All of the code written in the course of this research uses RCI only, however RCCL is briefly described below for completeness.

4.2.1 UNIX, NFS

The VAX-11/750 used in this research to run the robot control software was running UNIX 4.3 and NFS, which is a network file server. It was connected by an Ethernet link to several VAX and SUN computers and was therefore sensitive to network traffic. Communication between the VAX and the LSI-11 controlling the robot took about 7 msec, leaving about 7 msec for control computations on the VAX end. In periods of low network activity, all six joints could be controlled simultaneously, however the VAX became easily overloaded in times of high network traffic, resulting in communication timeouts and restricting the number of simultaneously moving joints.

4.2.2 RCCL, RCI

RCCL is a package of C subroutines which are used to control a manipulator in Cartesian coordinates. The RCCL routines provide a user with a high level interface to the robot, in relative isolation from the commands into which the high level routines ultimately translate. Inverse and forward kinematics are automatically computed, allowing a high level of task description and trajectory generation. RCCL also allows force control in addition to position control, and RCCL primitives may be used as input blocks to a higher level of trajectory planning. The isolation of the user from the control parameters is an advantage at the task level, however renders RCCL unusable for position control algorithm research.

For this purpose. RCI. the lower level of commands through which RCCL communicates to the robot. must be used.

RCI is a development environment for real time robot control software [Lloyd 85] Application programs are written as conventional C programs. The main. or planning part of the program is used for user interaction and calls two control routines. also written in C. The planning level and control level communicate by two shared data structures One. called *chg*. is used to transmit commands to the manipulator. and the other. called *how*. is used to collect information about the manipulator The control task produces commands which are transmitted directly to the manipulator. by writing appropriate information into the *chg* structure. A joint may be controlled in one of two modes. setting a joint position. or setting a joint current. The former is widely used in applications which involve trajectory generation. The control routines calculate the trajectory. and send position commands through the *chg* structure The position control and servoing take place at the joint controllers using the Unimation controllers Current mode is useful in applications involving control research A control algorithm is implemented directly in the control routines of RCI. and current setpoints are transmitted to the joint motors. The result is that the joint controllers merely pass the current information to the manipulator. and the entire control routine is implemented on the host computer In this research. the manipulator is controlled by current mode. and all of the control takes place on the VAX. The Unimation robot controller is still used to perform some error checking on the commands sent out to the PUMA and on the data returned through the *how* structure. Details concerning the information contained in the *how* and *chg* structures. as well as on the organization and function of the RCI system may be found in [Lloyd 85].

4.3 The Testbed

The general approach of the testbed is similar to that found in [Valvanis 85]. The design of the testbed must provide the following functionality:

The host computer must be powerful enough to execute the control functions in real time.

The control loop is closed right at the host computer, allowing great flexibility in choosing the control scheme.

The system parameters and gains must be adjustable on line.

Analysis routines must be provided to facilitate meaningful interpretation of the data.

The features of the system are detailed below. In general, the testbed developed as part of this research differs from the one mentioned above in that there is no attempt made to solve system dynamics. While this rules out computed torque based control algorithms, it allows the control routines to run at a much faster rate than could be achieved using the other system. Since sampling time is related to bandwidth and gain limitations, a useful testbed must run quickly enough to allow testing under meaningful conditions. Thus, if the testbed allows testing under very limited conditions, the conclusions of the tests may not be meaningful in application to faster motion. Almost any controller will provide adequate position control for low bandwidth tests. Compensation for dynamics can easily be added to the testbed, however it is expected that the addition of such calculations will have an effect on the minimum sampling interval. It is further worth noting that this controller testbed is not intended to perform Cartesian control. All control is performed in joint space, as this is the level at which position control ultimately takes place, and adding code to resolve Cartesian coordinates into joint coordinates would only consume computational time without any real relevant contribution to the study of the control algorithms being tested.

4.3.1 Features

The testbed consists of two programs, one of which contains the controllers and robot interface, and the other of which contains the available test routines and parameter settings. Manipulator control and data acquisition is performed by the control level routine, which is invisible to the user, and the user interface is conducted at the planning level. The programs are linked together via the RCI system, and the whole ensemble is started up using a command at the VAX terminal. Upon startup, the system presents the user with a prompt which allows access to a key tree matcher for entering commands and parameters. On line help is available by typing "?" at the prompt.

All six joints may be run simultaneously, however as the number of joints increases, so does the likelihood of communication timeouts. The system moves the joints only in joint space, as Cartesian trajectory generation would take too long to compute and is outside the scope and purpose of this work. The initial state of the system lets the user control joint 6 only, and all of the other joints are locked. Joints may be unlocked using the *set unlock (jointnumber)* command, and locked using the *set lock (jointnumber)* command A locked joint is frozen in place, and locking joints which are not in use is good practice both for safety reasons and to save computation time.

Since dealing with six joints at once may be overwhelming to the user, it was decided that parameters may be set and data displayed for only one joint at a time. The choice of joint is determined by the *set joint (jointnumber)* command. Once a joint is selected, control parameters and other system data may be determined for that joint alone It is also used to select which joint is to be displayed when using the *data dump*, *data file*, *show parameters*, and *show equation* commands The system defaults, as before, to joint 6.

As tests are run, the system keeps track of data such as demand, response, error, filtered error, velocity, acceleration, and current, in a set of data arrays. The size of these arrays may be changed using the *data default array size (size)* command. Upon startup, this value is set to 1024, which is the maximum allowed. This maximum was chosen for two reasons: first, the likelihood of a timeout occurring during a test increases with the length of time the test is run. Second, running many of the test for longer periods of time would cause the joint to travel beyond its limit stops. Running at a sampling interval of 14 msec, this allows a maximum testing time of 14.3 seconds, which has shown to be adequate.

Generally it is best to work at the smallest sampling interval possible, and the system defaults to the minimum of 14 msec. Sometimes, due to system loading, it may be necessary to sample at a slower rate, and the *set sampling interval (time in msec)* may be used. Allowable sampling intervals are 7, 14, 28, and 56 msec, however an interval of 7 msec generates a timeout almost immediately. It is also worth noting that as the sampling time increases, control gains must be reduced to maintain stability.

The purpose of this testbed is to test controllers, and at present two controllers - PD and acceleration feedback - are available. The system defaults to PD, and the controller may be changed using the *set controller (controller)* command. It should be noted that only the controller for the currently selected joint is affected by this command. The command *set controller idle* may be used to shut off control completely. This is of use in viewing desired trajectories before using them as input to the manipulator. Note that the PD controller by itself does not activate integration. The integral part of the controller is configured as a separate piece of code which may be activated in conjunction with either the PD controller or with acceleration feedback.

Additionally, since integral control is not of interest during trajectory following, the integrators can be set to turn on only within a limited band of error about the desired position. This band may be adjusted using the *set approach zone (min/max) (value)* command.

Parameters for the controllers may be set using the *set tuning parameter (parameter) (value)* command. This is performed for the currently selected joint only. Upon initialization, the controllers come up with parameters which are stable, however do not necessarily give optimal performance. They can then be modified on a per-parameter per-joint basis.

To facilitate running the system under various configurations, *save parameters* and *load parameters* functions have been implemented. Thus, a given configuration may be stored on disk and recovered at will. A filename is requested from the user, and these parameter files are automatically named filename.sav. Note that this procedure saves the configuration of the entire system, including the currently selected joint. Thus, loading a configuration suitable for testing a certain set of joints leads to a convenient point at which the joint is ready for testing As a safety procedure the demanded position on all of the joints is set to the current position, thereby zeroing the error when a new configuration is loaded. This is done to prevent large system reactions due to new parameters acting on a leftover error.

Finally, a low pass filter may be set to filter the error. The way it is set up, the filter accepts cutoff frequencies of 0-100 Hertz, and if a higher number is input, the filter shuts off The filter has been seen to cause instability, and most of the time it is recommended to operate without it. The problem lies in the fact that to satisfy the Nyquist rate imposed by the sampling interval, the pole of the low pass filter must be set so low that it starts to interact with the poles and zeros of the controller. The code has been left in, however, in anticipation of future hardware improvements which will allow operation of the system at a higher sampling frequency.

There are a variety of tests which may be performed to evaluate the performance of the joints These include *servo, step, ramp, sine, snake, square, and accelerate*. All have the same form. *test (testname) (jointnumber)*. They then prompt the user for various parameters such as amplitude and frequency. A *snake* test is a combination of a ramp and a sine wave, which was given a special name for ease of input *Accelerate* places a value directly on the the DAC output to the joint, which remains as a current setpoint for the joint for the duration of the test. Note that in the absence of friction, this is expected to produce a constant acceleration. Due to the effects of friction, however, this instead produces an acceleration to a constant terminal velocity. No tests actually begin operation during configuration. This was done to allow the user to configure different tests simultaneously on each joint. If a controller of a joint is set to *idle* or a joint is *locked*, configuring a test for it will have no effect, and it will not move. Having configured all of the desired tests for the various joints, the user the types *test go* to actually runs the tests. This command causes all the joints which have been configured to run, to start moving simultaneously. The tests continue to run for the duration specified by the *default data array size*, as previously described.

While tests are run, data is collected into a number of arrays. To view the data, one can use the *data dump (samples)* command, which dumps a specified number of samples onto the screen. To save the data in a file, type *data file*, and the system will prompt for a filename. The file is automatically named filename.log. Together with the data, the relevant parameters defining the system are stored alongside it in a file called filename.prm. It is worth noting that all data display and storage operations are conducted with respect to the currently set joint. A *data clean* function has also been provided to clear the arrays, but this has not been shown to be of great use during operation. The decision to make data storage and display operate on a one joint basis was made to avoid storing data for joints which are not relevant for the current tests, and also to avoid overwhelming the user with an untenable quantity of data. In the current system, the user can choose which joints are relevant, and to store data for those joints alone.

Error trapping has been left to the RCI system, however an error handler has been implemented to allow the testbed to recover from an error. There are several types of error that may be trapped by RCI. The include the following:

Time out

Maximum Velocity Exceeded

Maximum Current Exceeded

Maximum Requested Current Exceeded

Joint Position Out of Range

Several other types of errors exist and are trapped by RCI, however the above are the most frequently occurring during operation of the testbed. The error handler of the testbed receives the trapped error from RCI, displays an appropriate message on the screen, and returns control to the testbed. For safety, the error handler shuts off the controllers by setting them to *idle*, and sets the desired position to the current position, thereby setting the error to zero. Power is not shut off automatically, though this feature could be easily added if seen to be necessary

Several errors can not be trapped by RCI. Examples of such errors include a joint being out of control and about to hit something, or a joint not currently exceeding limits, but traveling at a velocity high enough to break its limit stops. The present solution to these problems is to exercise care and judicious use of the *arm power off* button A solution to the first problem is to calculate the forward kinematics for the manipulator at all times and to set limits on its work space, however this is not practical both due to the large amounts of calculation involved and due to the fact that the environment may change without notice A solution to the second problem would involve shutting the power off automatically within a certain distance from the limit stops. This has the disadvantage in that to be effective, such a strategy would seriously reduce the available workspace, and has therefore not been done. Unlike the solution to the previous problem, however, this solution is both tenable and practical, and may easily be added to the system. Regardless of any safety features, use of a system which may knowingly send manipulator joints out of control must be undertaken with care, and one should be ready to shut power off manually in case of the unexpected

To avoid having to reconfigure or exit the program each time an error or panic occurs, a *poweron* command has been implemented. This command may also be used as a reset function, in that it first sets all of the errors to zero, and only then turns power on It has been found to be of great use in eliminating residual errors from previous testing. Finally, the *exit* command is used to terminate the program.

4.3.2 Analysis Routines

The testbed provides a means of obtaining and storing data from the manipulator. To allow the user of the system to analyze and understand this data, several analysis routines have been provided, and will be listed below:

For system identification, a parameter estimation routine using least squares approximation has been configured. This routine takes as input parameters a file name from which to obtain the data, a number representing the length of the array, an initial estimate vector, and a Kalman gain matrix. The output of this program is a set of parameters from which the system gains may be inferred.

For frequency response, two routines are provided: the first one is a fast Fourier transform (FFT) routine, which takes as input a file name and a number for the length of the array, and returns the gain at the frequency in which the test was conducted. The second routine provides a means for easily detecting zero crossings of the input and output sinusoids visually Using this data, it is possible to derive phase information for the system.

Finally, for analysis of response, a routine is provided to calculate maximum, minimum, average, and standard deviation of the joint tracking error for a given trajectory.

For graphical analysis, a program was written to use the graphics plots of the ACSL simulation package. The plotting routines provided in ACSL allow considerable flexibility and were deemed suitable for the graphing need of this research. A small front-end routine is provided to allow ACSL to accept PUMA data as generated by the testbed *data file* command, and no modifications were made to the ACSL package itself.

Chapter 5 Controller Experiments

Several issues are discussed in this chapter. First, the experiments on system identification are presented. This is followed by sections describing the coding and tuning of the control algorithms. Finally, the actual controller experiments are presented in great detail, and an analysis follows.

5.1 System Identification

The identification of the system parameters is helpful in that it allows easier tuning and more meaningful analysis of the experimental results. Due to the nonlinear and time varying nature of this system, the problem of system identification is a difficult one. The gain of the system varies with time in a nonlinear fashion, as do all of the other parameters associated with its motion. In particular, friction poses the greatest challenge to system identification. Both static and dynamic friction have been seen to vary enormously with time, position, velocity, and other factors which are hard to identify. For example, values of static friction have been shown to more than double within ten minutes. This greatly complicates the problem, as friction not only varies wildly, but is for many types of motion the dominant factor in the dynamics of the manipulator system. This is particularly true in this model of PUMA manipulator, which is small and therefore has relatively low gravitational and inertial contributions.

5.1.1 Notation

All of the work done in this research was performed in terms of the simplest units obtainable from the system, for ease of programming and interaction with the manipulator. A presentation of the units employed in the experiments is therefore included at this point, and a conversion from these units to the SI system is provided as well. All subsequent discussion will be conducted in terms of the *natural* units of the manipulator system, and further conversion of specific parameters is left up to the reader

The most natural units in which to work are in terms of the hardware components available. The joint encoders deliver a number in the thousands which is referred to as an encoder count, or occasionally a basic length unit (BLU) [Koren 85]. Current is delivered to the joint motor by placing a value on the digital to analog converter (DAC) and is read in using an analog to digital converter (ADC). These devices have units associated with their values which will be referred to as DAC units and ADC units, respectively. Thus, the forward loop gain of the system is expressed in terms of DAC units per encoder count squared, and the control gains are in similar units as well

Given the gear ratio, a conversion from ADC units to motor torque (in Nm), a conversion from ADC units to DAC units, and a conversion from encoder counts to radians, it is possible to express the system gains in SI units. The values of these constants were obtained experimentally by Lloyd [Lloyd 85]. For instance, the conversion of the high gain feedback K_u from DAC unit per second squared to Newton meter per second squared proceeds as follows.

$$K_u(\frac{Nm}{sec^2}) = Gear\ ratio \times \frac{ADCtoMTOR}{ADCtoDAC} \times \frac{EncoderCounts}{Radians} \times K_u(\frac{DACunits}{sec^2}) \tag{5.1}$$

and the results are tabulated below.

5.1.2 Preliminary Tests

The preliminary tests which were run to determine the system gain were based

Joint	$K_u(\frac{DAC}{sec^2})$	Gear Ratio	$\frac{ADCtoMTOR}{ADCtoDAC}$	$\frac{EncoderCounts}{Radian}$	$K_u(\frac{Nm}{sec^2})$
1	1	46.7	2.0316×10^{-4}	7435	70
2	1	69.9	2.0861×10^{-4}	11136	162
3	1	42.9	2.1689×10^{-4}	6841	64

Table 5.1 Conversion of K_u to SI Units

on a very simplified model of an actuator powered by a current amplifier, as shown in figure 5.1:

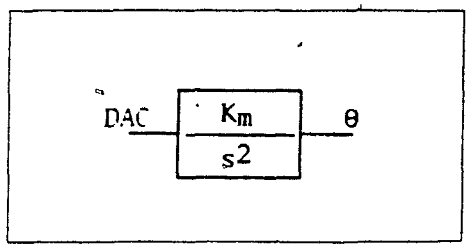

Figure 5.1 Actuator Driven by a Current Amplifier

The test was based on the assumption that given a fixed input current i, we should get a fixed steady state acceleration a proportional to i by the system gain K_m. A number of experiments were carried out to verify this assumption, yielding values for acceleration that were neither steady state nor constant. The experiments were conducted by placing a constant current as input, in the form of a value on the digital to analog converter (DAC) of the joint. A range of current values was tested, going from a value barely enough to move the joint to a value roughly twice in magnitude. A typical result is presented in figure 5.2.

These experiments were carried out on joint 6 of the PUMA, and show that the simplified system model can not adequately describe the system. Specifically, viscous friction must be introduced into the model, accounting for the fact that a constant current produced a constant velocity, as opposed to a constant acceleration. The system model shown in figure 5.1 above was therefore replaced by the one in figure 5.3.

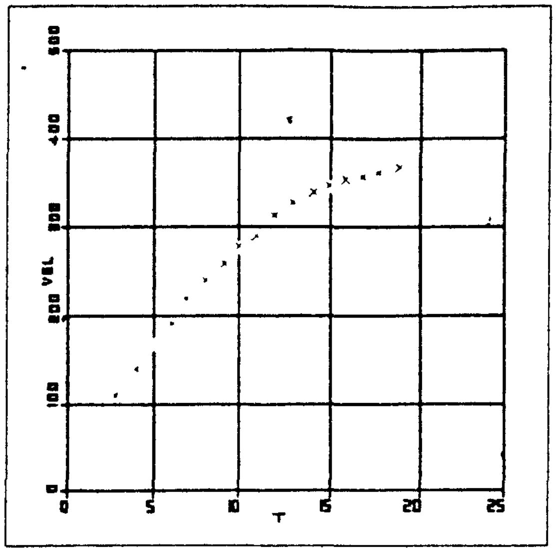

Figure 5.2 Velocity vs Time for Constant Current Input

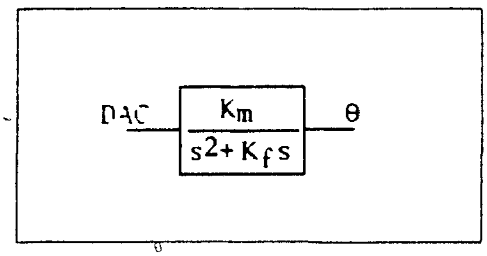

Figure 5.3 Actuator Driven by a Current Amplifier with Viscous Friction

The new model contains two unknowns. K_m and K_f. By applying proportional control to this system in a closed feedback loop and stimulating the system with a step input, these constants can be determined. The closed loop system has the transfer function

$$\frac{\theta}{\theta_d} = \frac{K_p K_m}{s^2 + K_f s + K_p K_m} \tag{5.2}$$

This is a second order system, whose damping and characteristic frequency can be determined from the response of the system to a step input. Given that the system is underdamped with proportional control alone, a step response should yield values for maximum overshoot Y_{max} and time to maximum overshoot T_{max}. The damping ς and characteristic frequency ω_n are then derived from this data:

$$\varsigma = \sqrt{\frac{(ln|Y_{max}|)^2}{(ln|Y_{max}|)^2 + \pi^2}} \tag{5.3}$$

$$\omega_n = \frac{\pi}{T_{max}\sqrt{1-\varsigma^2}} \tag{5.4}$$

The constants K_m and K_f may now be determined as

$$K_m = \frac{{\omega_n}^2}{K_p} \tag{5.5}$$

$$K_f = \sqrt{\varsigma^2 4 K_p K_m} \tag{5.6}$$

Experiments were carried out to verify this model, and the following results were observed: The time to reach maximum overshoot was very stable, and varied inversely with K_p. The value of ω_n, which depends heavily on T_{max} was also relatively stable. The damping and by necessity the frictional gain varied erratically, however, as the value of friction changed with the position of the joint, temperature, and time. K_m was determined to have a value of about 1000, in DAC units, but was subject to considerable variation. No usable value could be determined for K_f, as it varied from 1 to 419.

5.1.3 Frequency Response

Since acceleration feedback is essentially a frequency based technique, it was deemed important to obtain at least some data concerning the frequency response of this system. Gain and phase plots for joint 6 and joint 3 were obtained using closed loop proportional control with unity feedback. The input to the manipulator was a series of sinusoid ranging in frequency from 0.05 Hertz to 6 Hertz. The amplitudes of the sinusoids were varied from 200 to 1000 encoder counts, and multiple measurements were taken. In figure 5.4 the gain results are displayed as dark squares. Double squares indicate identical measurements, and hollow squares indicate that the amplitude of the input sine wave was too small for the joint to overcome friction.

A fast Fourier transform (FFT) algorithm was then applied to the data to extract gain information. The results verify the validity of the second order model proposed earlier to describe the system. The expected response for this underdamped system is a constant gain, rising to a resonant peak at the break frequency, and then dropping at a rate of 40

DB per decade. In the figure that follows, the data from the experiments on joint 3 and joint 6 is presented. Superimposed on the data is a line of -40 DB per decade, and the data is seen to cluster about that line, verifying the model

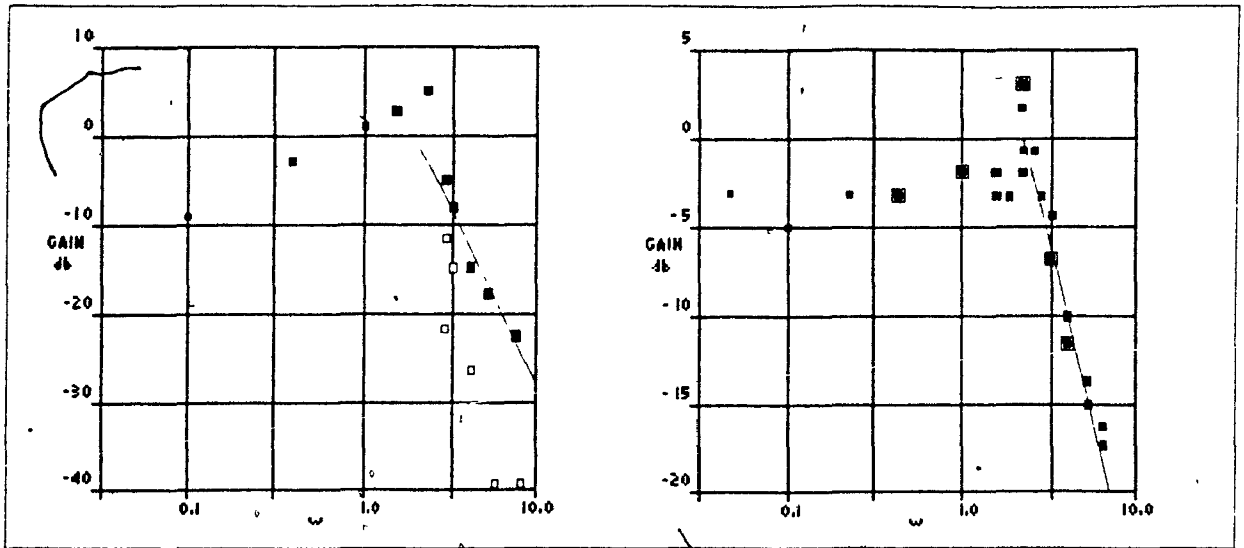

Figure 5.4 Gain Plot of Joint 3 (left) and Joint 6 (right)

An algorithm using the zero crossings of the system input and response to various frequency sine waves was then used to obtain phase information. The dramatic increase in phase lag with frequency is helpful in explaining the tendency of the system to go unstable at higher frequencies It is also worth noting that joint 3 is more susceptible to this effect than joint 6 This is attributed to the higher friction on joint 6 whose higher damping reduced the phase lag.

5.1.4 Recursive Least Squares

The final effort in system identification was undertaken using parameter estimation by recursive least squares. The procedure used was based on an algorithm found in [Clark 81]. The discrete model of a closed loop system consisting of proportional controller, a zero order hold, and a plant including an actuator powered by a current amplifier and

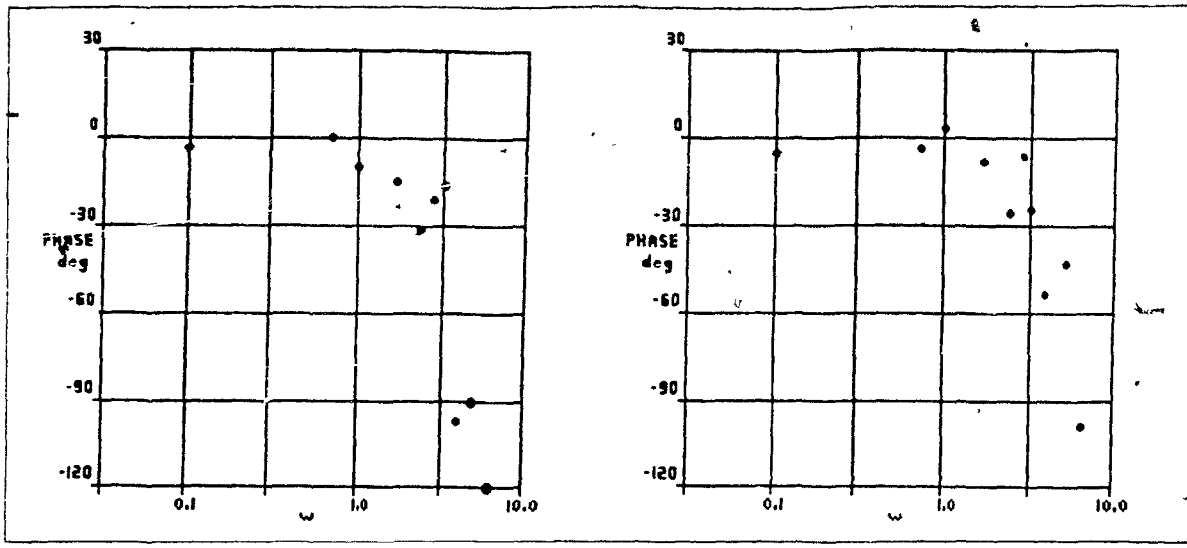

Figure 5.5 Phase Plot of Joint 3 (left) and Joint 6 (right)

accounting for viscous friction, has the following transfer function:

$$\frac{y(k)}{u(k)} = \frac{N(z)}{K_f^2(z-1)(z-e^{-K_fT}) + N(z)} \tag{5.7}$$

where

$$N(z) = K_p[K_m(K_fT - 1 + e^{-K_fT})z + (1 - e^{-K_fT} - K_fTe^{-K_fT})] \tag{5.8}$$

This may be rewritten as

$$\frac{y(k)}{u(k)} = \frac{a_0z + a_1}{b_0z^2 + b_1z + b_2} \tag{5.9}$$

where

$$a_0 = K_pK_m(K_fT - 1 + e^{-K_fT}) \tag{5.10}$$

$$a_1 = K_pK_m(1 - e^{-K_fT} - K_fTe^{-K_fT}) \tag{5.11}$$

$$b_0 = K_f^2 \tag{5.12}$$

$$b_1 = a_0 - b_0(e^{-K_fT} + 1) \tag{5.13}$$

and

$$b_2 = a_1 + b_0e^{-K_fT} \tag{5.14}$$

If we further manipulate the equation by dividing through by b_0 and rearranging terms, we arrive at the equation

$$y_n = -ay_{n-1} - by_{n-2} + cu_{n-1} + du_{n-2} \tag{5.15}$$

where

$$a = \frac{b_1}{b_0} \qquad b = \frac{b_2}{b_0}$$
$$c = \frac{a_0}{b_0} \qquad d = \frac{a_1}{b_0}$$

This is a polynomial suitable for estimation by the technique proposed in [Clark 81]. The procedure was coded up and tested on simulated data Convergence was very fast for the simulated plant, arriving at values within one percent of the desired parameters within twenty iterations The technique was then applied to two joints of the PUMA 260 with the following results Convergence of this procedure for actual PUMA data was precarious and slow. During several runs, even after settling on certain values, the parameters would jump in magnitude or sign, before converging again Typically, after such a jump, the parameters would converge to the same values as before the jump. During a few of the runs, however, the values diverged or cycled instead of settling. Tests were performed on joint 3 and joint 6, using a variety of initial estimate vectors and Kalman gain matrices. The values obtained were as follow For joint 3:

$$a = 0.095 \qquad b = 0.024$$
$$c = 0.950 \qquad d = -1.83$$

For joint 6:

$$a = 0.075 \qquad b = 0.050$$
$$c = 0.030 \qquad d = -1.10$$

These figures led to the following values of K_m and K_f for the two joints. For joint 3:

$$K_m = 25.2$$
$$K_f = -339.2$$

And for joint 6:

$$K_m = 325.0$$
$$K_f = -87.6$$

Generally, the parameters for joint 3 converged more easily and reliably than those of joint 6. This is attributed to the fact that friction is much greater on joint 6 than on joint 3. Since friction has been shown to vary unpredictably, it is expected to give the most trouble in estimation The algorithm assumed a disturbance an unknown error term with a zero mean distribution, which is not necessarily the case for this system. Thus, the estimation of the system parameters of the PUMA 260 manipulator in a reliable manner probably requires an algorithm of greater complexity than the one presented here.

5.2 Coding the Control Schemes

In the discussion that follows, the following notation will be used throughout. θ represents the joint's position, and θ_d represents the desired joint position. Calculated quantities are as follows:

$$error = \theta_d - \theta \tag{5.16}$$

$$velocity\ error = \frac{d}{dt} error \tag{5.17}$$

$$acceleration\ error = \frac{d}{dt} velocity\ error \tag{5.18}$$

5.2.1 PD

The PD controller is modeled as shown in figure 5.6

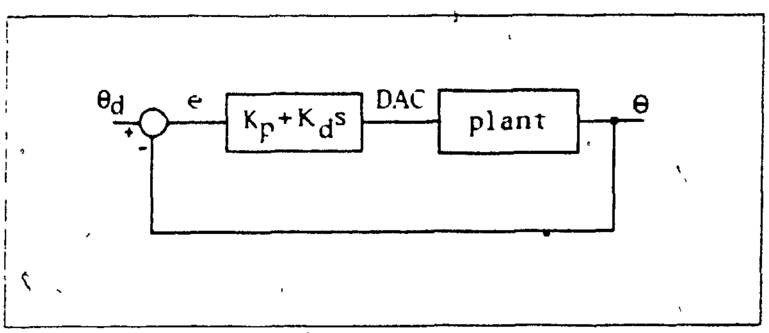

Figure 5.6 Closed loop system of plant and PD controller

and is described by the equation

$$DACvalue = error * K_p + (velocity\ error) * K_d \tag{5.19}$$

The complication in this type of controller is in way that velocity error is derived. Ideally, this error should be measured by a tachometer, and should therefore be as easily available as position error. One of the goals of this research, however, is to demonstrate that velocity and acceleration errors derived from position data provide a clean enough signal to allow stable and accurate control without use of tachometers and accelerometers. Two methods of obtaining velocity errors were investigated, one was by using the backward triangular rule [Franklin 81],

$$s = \frac{z-1}{T_s z} \tag{5.20}$$

which is essentially a first order differentiator, and gives a velocity error of

$$velocity\ error = \frac{error[n] - error[n-1]}{T_s} \tag{5.21}$$

The second method which was investigated and ultimately incorporated into the testbed involved using a second order differentiator [Conte 80]

$$D_1(z) = \frac{3 - 4z^{-1} + z^{-2}}{2T_s} \tag{5.22}$$

This yielded a velocity error of

$$velocity\ error = \frac{3 * error[n] - 4 * error[n-1] + error[n-2]}{2T_s} \tag{5.23}$$

The second order differentiator is more accurate and less subject to noise than the first order differentiator using the backward triangular rule. It is therefore the one recommended by Studenny [Studenny 87]. The one drawback this differentiator has is that upon startup it takes one sampling interval longer than the first order differentiator to properly initialize. This leads to a jerk in motion at the beginning of a trajectory, however this has not been seen to cause great problems since the velocity error smooths out two sampling intervals later.

5.2.2 Acceleration Feedback

The acceleration feedback controller is more involved, as it includes three errors, namely position, velocity, and acceleration, and three gain parameters. The system appears as shown in figure 5.7.

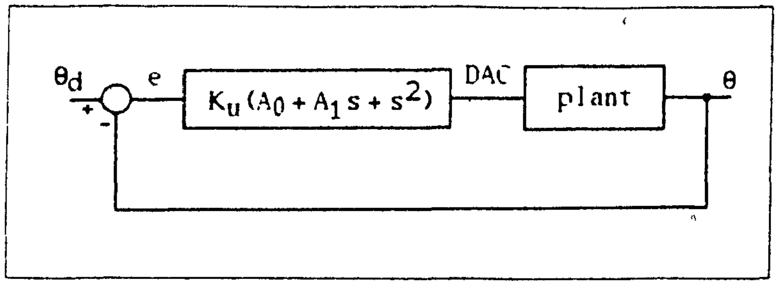

Figure 5.7 Closed loop system of plant and acceleration feedback controller

The controller is described by the equation

$$DACvalue = K_u * (error * A_0 + (velocity\ error) * A_1 + (acceleration\ error)) \quad (5.24)$$

The velocity error was obtained using the second order differentiator, as in the case of the PD controller. Acceleration error was also obtained using a second order differentiator, which looked like

$$D_2(z) = \frac{1 - 2z^{-1} + z^{-2}}{{T_s}^2} \quad (5.25)$$

This yielded an acceleration error of

$$acceleration\ error = \frac{error[n] - 2 * error[n-1] + error[n-2]}{{T_s}^2} \quad (5.26)$$

The use of identical differentiators for both controllers is necessary to ensure a valid comparison The differentiator used for acceleration causes a powerful jerk upon startup, but like the velocity calculation settles down after two sampling intervals. It is important to note that while this jerk does not have any bearing on the evaluation of the controllers once in a smooth trajectory, the maximum current demand caused by the uninitialized differentiators sets limits on the magnitude of step the system can react to without exceeding current or velocity limits.

5.2.3 Integral control

Integral control is implemented separately from the other controllers, and may be used in conjunction with either acceleration or PD. The integrator is considered as a system by itself, and was evaluated as such. The open loop block diagram of the integral controller appears in figure 5.8.

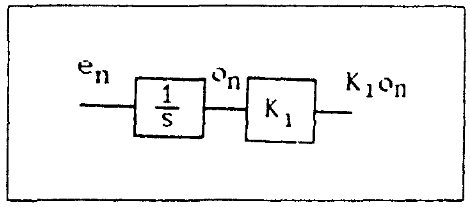

Figure 5.8 Integral Controller

Using Tustin's rule on the integrator alone, not including the gain, we get

$$\frac{1}{s} = \frac{T_s}{2}\frac{1 + z^{-1}}{1 - z^{-1}} \tag{5.27}$$

Thus

$$integrator\ output[n] = integrator\ output[n-1] + \frac{T_s}{2}(error[n] + error[n-1]) \tag{5.28}$$

This integrator output is then multiplied by the integrator gain K_i and the result is a DACvalue which is added to the previously computed DACvalue resulting either from the PD or acceleration feedback controllers. Generally, an integrator in intended to function in point positioning by correcting steady state position error. Thus, in this implementation, the integrator was designed so as only to operate if the position error is within a specified band on either side of the desired position. This band need not be symmetric, and may be adjusted on line as described in an earlier section. Outside this band, all of the code pertaining to the integral controller is simply bypassed.

5.2.4 Other Controller Issues

In implementation, it was noticed that initial current demands due to the action of the uninitialized differentiators frequently exceed the current limits imposed by the RCI system. To avoid system shutdown and the termination of a test due to such spurious spikes in demand current, code was included to clamp the current demand at the maximum allowed by RCI. Though this procedure invalidates the data for samples on which clamping takes place, it should be noted that this typically occurs only during the first two sampling intervals, in which the differentiators are not initialized and the results are of questionable value anyway. This feature also proved to be of great value from a safety standpoint, as it prevents the manipulator from shutting off and skidding out of control at high velocities

The acceleration test mentioned in an earlier section functions by replacing the DACvalue calculated by the controllers with the value specified by the user when the test is configured All of the calculations continue as before, and the replacement of the DACvalue is performed immediately before output to the joint.

Likewise, setting a controller to idle does not shut down the calculations performed by the controllers The calculations continue as before, however the calculated value is never sent to the joint This allows the user to place a joint's controller in idle mode, examine the DACvalues computed by the controllers, and decide upon further action based on that data

As recommended in [Studenny 87], a low pass filter was implemented to eliminate high frequency uncertainties from being multiplied by the high gain feedback K_u of the acceleration feedback controller In this implementation, the low pass filter was coded to act on the position error Velocity and acceleration errors were then calculated from the filtered error Since numerical differentiators are inherently unstable and susceptible to noise, this was deemed useful in reducing the noise content of their input signal as well. A first order low pass filter

$$C(s) = \frac{1}{\frac{s}{\omega_C} + 1} \tag{5.29}$$

was used for this purpose. Using Tustin's rule, this low pass filter was transformed into its digital equivalent

$$C(z) = \frac{T_s\omega_c + T_s\omega_c z^{-1}}{(2 + T_s\omega_c) + (T_s\omega_c - 2)z^{-1}} \tag{5.30}$$

yielding code of the following form:

$$error[n] = \frac{T_s\omega_c}{2 + T_s\omega_c}\left(\frac{2 - T_s\omega_c}{T_s\omega_c} error[n-1] + raw\ error[n] + raw\ error[n-1]\right) \quad (5.31)$$

Two problems may be noted with this filter. First of all, there is an unavoidable phase lag of one sampling interval using this filter. This is in addition to the unavoidable phase lag of at least one sampling interval caused by the nature of digital control Second, due to the limitations of Nyquist rate, the pole of the low pass filter could not be placed far enough out as to not interfere with the poles and zeros of the controllers. Both of these factors led to instability for certain types of motion. Thus, a mechanism to bypass the filter had to be implemented, as described in an earlier section.

5.3 Tuning the Control Schemes

A system for interactively synthesizing control schemes for manipulators was proposed by [Vukobratović 82] Though their system is comprehensive, it was not deemed useful for our purposes for a number of reasons First, the system relied on at least nominal knowledge of the dynamic parameters of the manipulator For the most part, the controller specified by this type of system grows out of the assumption that the dynamics can be known accurately before synthesizing the control. For our purposes, neither the PD nor the acceleration feedback schemes were designed assuming any knowledge of the plant. The only assumption made was that of a second order model Second, the scheme proposed in this paper was very elaborate, and required excessive computation. We must therefore tune the control schemes using a different method.

5.3.1 PD

Preliminary tuning of the PD controllers was performed by experiment using a step input and measuring the rise time and overshoot at various settings of proportional and derivative gains Contrary to expectation, for a fixed proportional gain, damping did not increase to infinity with derivative gain, but instead reached an optimum value and then retreated. Looking at the equation for the simplest model of the closed loop system in continuous time, we have

$$T = \frac{K_d K_m s + K_p K_m}{s^2 + K_d K_m s + K_p K_m} \tag{5.32}$$

The zero in the transfer function starts contributing at lower frequencies as derivative gain is increased, a fact which sets an upper limit on the derivative gain. An optimal setting was found to have a proportional gain of $K_p = 1$ and a derivative gain of $K_d = 2$. These values still gave an underdamped system, however with minimum overshoot and nearly critical risetime. The only way to obtain an overdamped system would have been to decrease the proportional gain, however this gain must be high enough to create demand currents high enough to overcome friction. The settings mentioned above have the combined qualities of being small enough not to demand velocities, currents and accelerations beyond the capacity of the PUMA, while being high enough to significantly overcome friction. In practice, the range between these two extremes on the manipulator used in this work was fairly limited.

Several tuning methods for PD algorithms may be found in the process control literature [Douglas 72], [Smith 85]. Most popular among them is the Ziegler-Nichols method. This method, like many methods of system tuning, consists of two steps: determination of the dynamic characteristics of the plant, and estimation of the tuning parameters. The beauty of this method is in the simplicity of both these parts. The only system parameters which must be determined are the *ultimate gain* and the *ultimate period*. The ultimate gain is defined as the gain K_{ul} at which the system starts behaving as an oscillator, with proportional control only. The ultimate period is the period T_{ul} of the oscillation at this gain. Once these parameters have been determined, the tuning constants are defined as follows:

$$K_p = \frac{K_{ul}}{1.7} \tag{5.33}$$

$$K_d = \frac{K_p T_{ul}}{2} \tag{5.34}$$

Another method for control parameter tuning is called the *Reaction Curve* method, and is based on the system's response to a step input. From this response, the system gain K, dead time t_d, and time constant τ may be determined. The gains for a PD controller are then calculated as:

$$K_p = \frac{1}{K}\frac{\tau}{t_d}(1.25 + \frac{t_d}{6\tau}) \tag{5.35}$$

$$K_d = K_p t_d \frac{(6 - \frac{2t_d}{\tau}}{22 + \frac{3t_d}{\tau})} \tag{5.36}$$

The Ziegler-Nichols method was used to determine the PD tuning parameters of the PUMA 260, with the following results. For joint 6:

$$K_{ul} = 1.4$$

$$T_{ul} = 0.154 sec$$

Therefore,

$$K_p = \frac{1.4}{1.7} = 0.82$$

$$K_d = \frac{0.82 * 0.154}{2} = 0.063$$

For joint 3:

$$K_{ul} = 1.6$$

$$T_{ul} = 0.196 sec$$

Therefore,

$$K_p = \frac{1.6}{1.7} = 0.94$$

$$K_d = \frac{0.94 * 0.196}{2} = 0.092$$

Clearly, the damping prescribed by the Ziegler-Nichols method is far smaller than the ideal damping obtained experimentally, as the method is designed to result in a system with a 50 percent overshoot. In the experiments conducted later in this chapter, the experimentally obtained damping constant is used, as the one obtained by Ziegler-Nichols gives too great an overshoot and places the system too close to its stability boundary. The proportional gain obtained by Ziegler-Nichols is, however, closely related to that obtained by experiment.

5.3.2 Acceleration Feedback

Preliminary tuning of the acceleration feedback was based on the structure of the acceleration feedback control law and on the results of the tuning experiments of the PD controller. The three control gains of the acceleration feedback controller K_u, A_0, and A_1, may be divided into two groups. The high gain feedback K_u is used primarily for linearization and decoupling. Used to render the system more robust to the effects of inertial disturbances, it is not involved in determining system tuning parameters. The other two gains, A_0 and A_1, determine system performance in much the same fashion as K_p and K_d in the PD case. A fair comparison of the performance of the two control laws could therefore be obtained by tuning A_0 and A_1 to the settings of K_p and K_d respectively as determined for the PD controller. K_u is then set to a value large enough to be significant with respect to the inertial disturbances, a value which has been determined to be about one order of magnitude higher. Setting the gain $K_u = 1$ in our case can be shown to achieve this type of ratio.

5.4 Controller Experiments

The goals of the controller experiments are to demonstrate the effects of the nonlinear terms in the dynamics equations on the trajectory following characteristics of the manipulator and to compare the performance of the PD and acceleration feedback controllers in the absence and presence of these effects. In the literature, one can find varied assertions concerning the effects of the nonlinear terms in the dynamics. If these effects are not significant, the control of manipulator joint position reduces to a linear problem, one for which many solutions perform adequately. If the effects are significant, the problem is indeed highly nonlinear, and it is worthwhile spending great effort in finding improved controllers. Some of the experiments in this section are aimed at assessing the magnitude of these effects. Once this has been done, experiments comparing the performance of the PD and acceleration feedback controllers may proceed. In theory, the performance of these controllers should be similar under conditions of low nonlinear effects but should differ in the presence of high nonlinear disturbances. Since acceleration feedback is claimed in theory [Studenny 87] to linearize the problem and make it robust to nonlinear effects, it is expected that under conditions of high disturbance, acceleration feedback will outperform PD in terms of tracking error and overshoot

The first set of experiments consists of five parts The first part is meant to look at the various terms in the dynamic equation, and demonstrate the effects of the disturbances on an individual basis In each case, the tests are run on a single joint in two configurations, one in which nonlinear effects are expected to be small, and the other in which they are expected to be significant Joint 1 is used to demonstrate the effects of inertia, joint 3 is used to demonstrate the effects of gravity, joint 2 the effects of both gravity and inertia, and joint 6 the effect of friction The test on joint 6 was run in only one configuration since there is no configuration in which friction is expected to reduce significantly. PD control was used throughout this part, as it was not the controller that was being examined Step and sine inputs were used throughout all five sections of this set of experiments, as they were expected to give and indication of error, overshoot, and response time

The second and third parts of this set of experiment are used to compare the performance of PD and acceleration feedback for single joint operation. The second section compares their performance for configurations in which inertial, gravitational, and frictional forces are expected to be small, and the third section compares their performance in configurations under which these effects are expected to be significant.

The fourth section extends the analysis of the first section to the multiple joint case. In this section the effects on one joint's motion on the position control of another joint are examined. As before, a variety of configurations are tested, and only PD control is used

The fifth and final section of this set of experiments compares the performance of the PD and acceleration feedback controllers in the presence of nonlinear effects due both to the dynamics of the joint being tested and the motion of other joints of the manipulator.

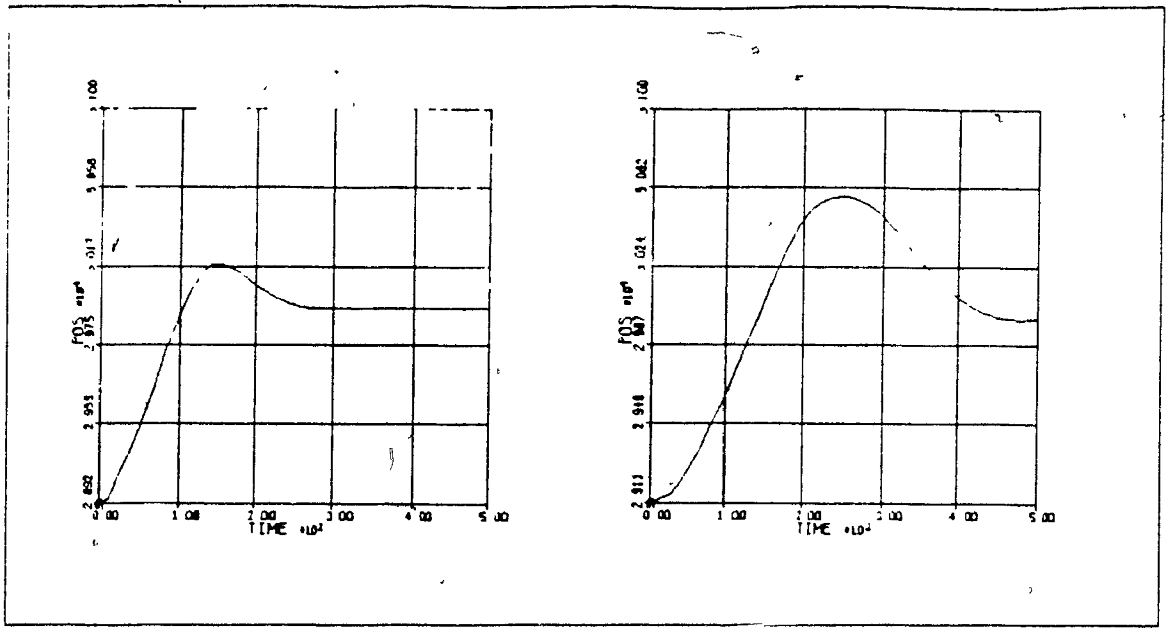

Figure 5.9a Step Response Unloaded (left), Loaded (right)

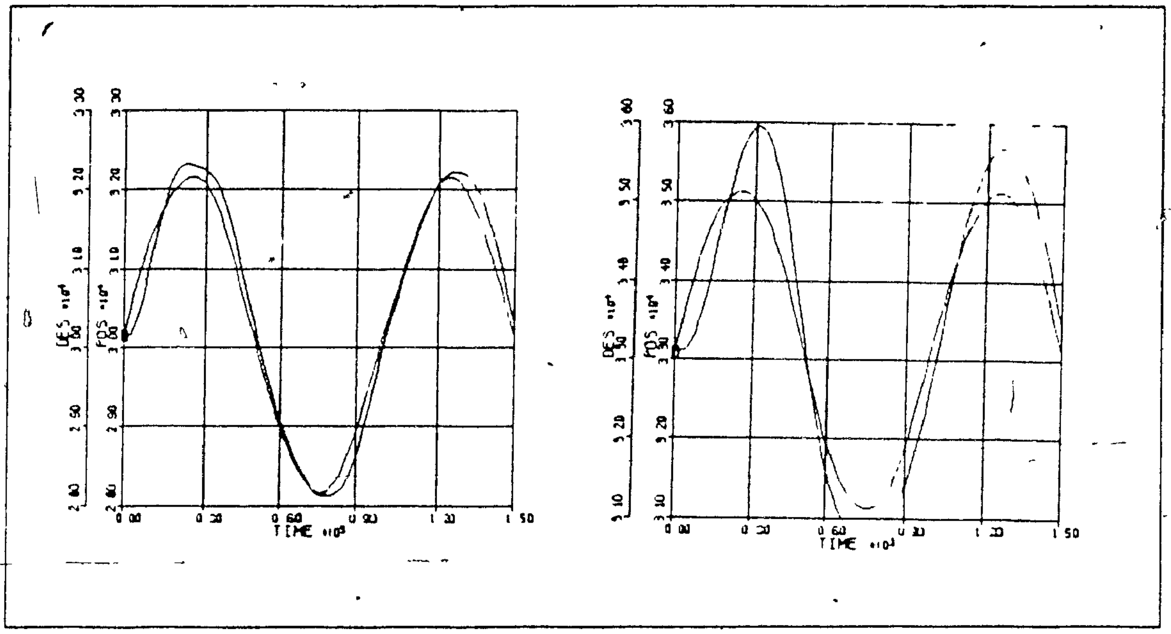

Figure 5.9b Sinusoidal Tracking Unloaded (left), Loaded (right)

Figure 5.9 Effects of Inertia Joint 1

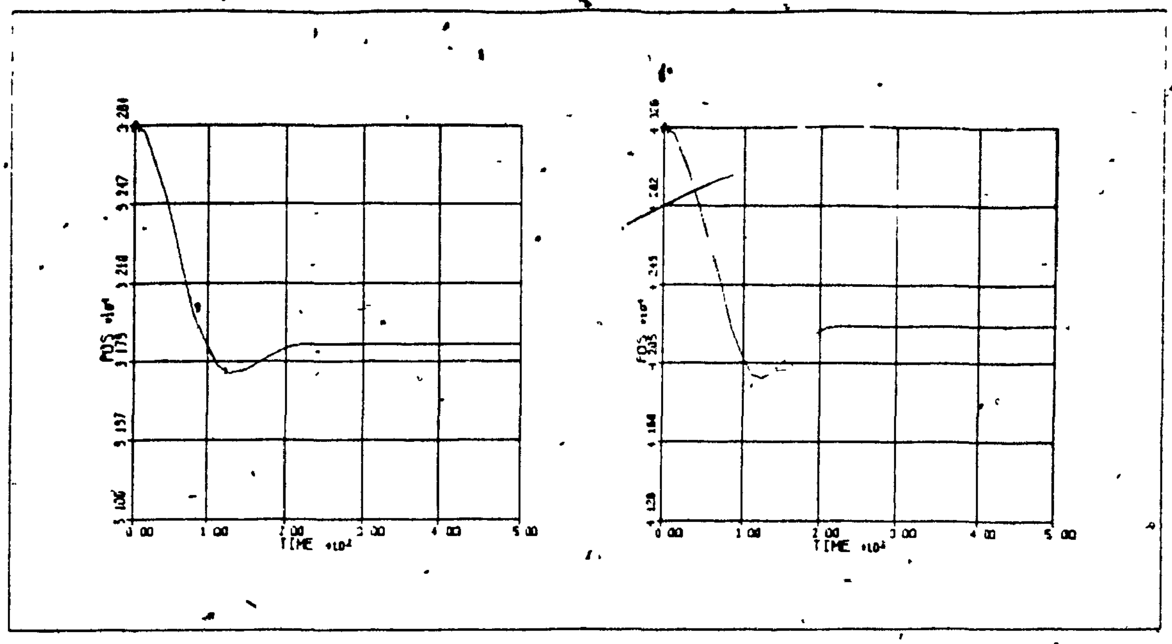

Figure 5.10a Step Response. Unloaded (left). Loaded (right)

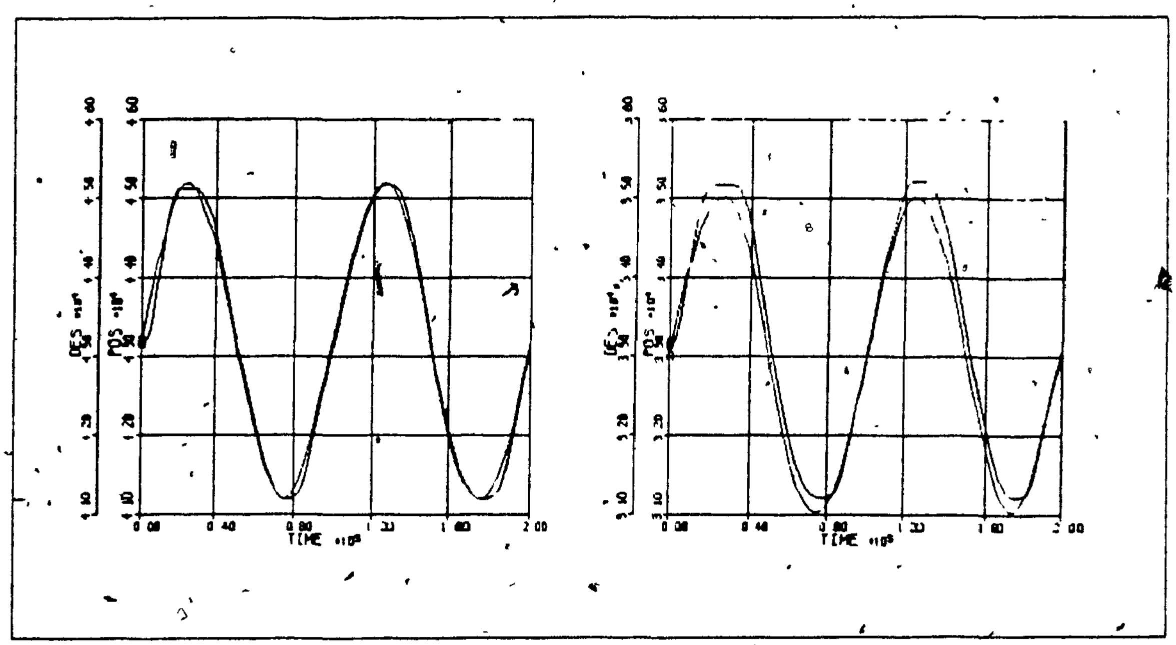

Figure 5.10b Sinusoidal Tracking Unloaded (left). Loaded (right)

Figure 5.10 Effects of Gravity Joint 3

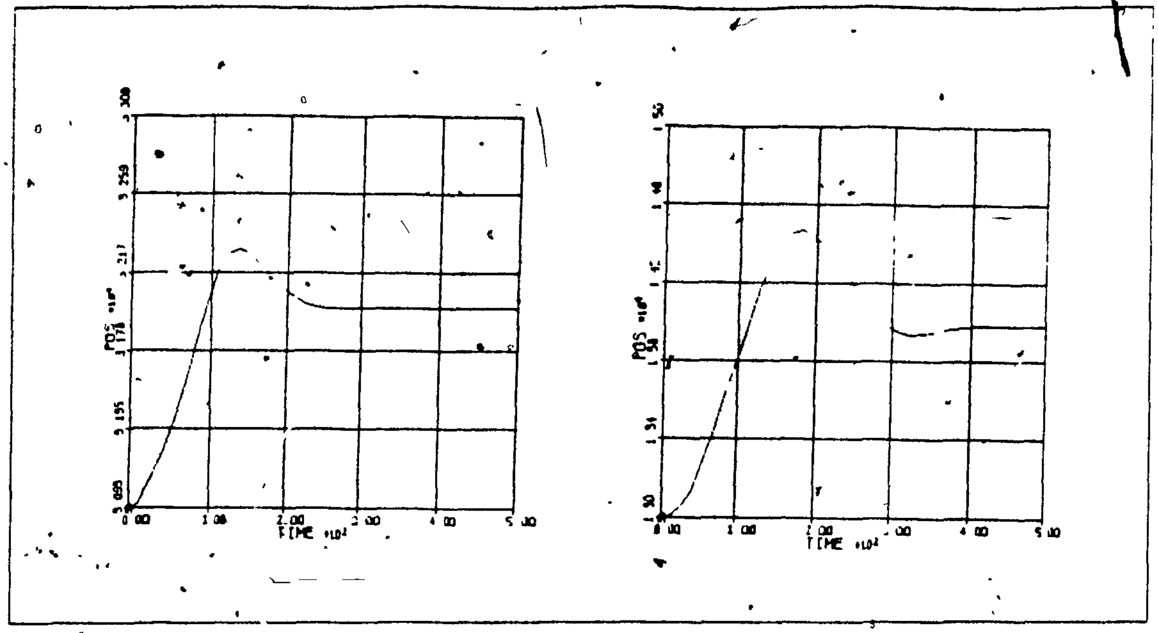

Figure 5.11a Step Response Unloaded (left), Loaded (right)

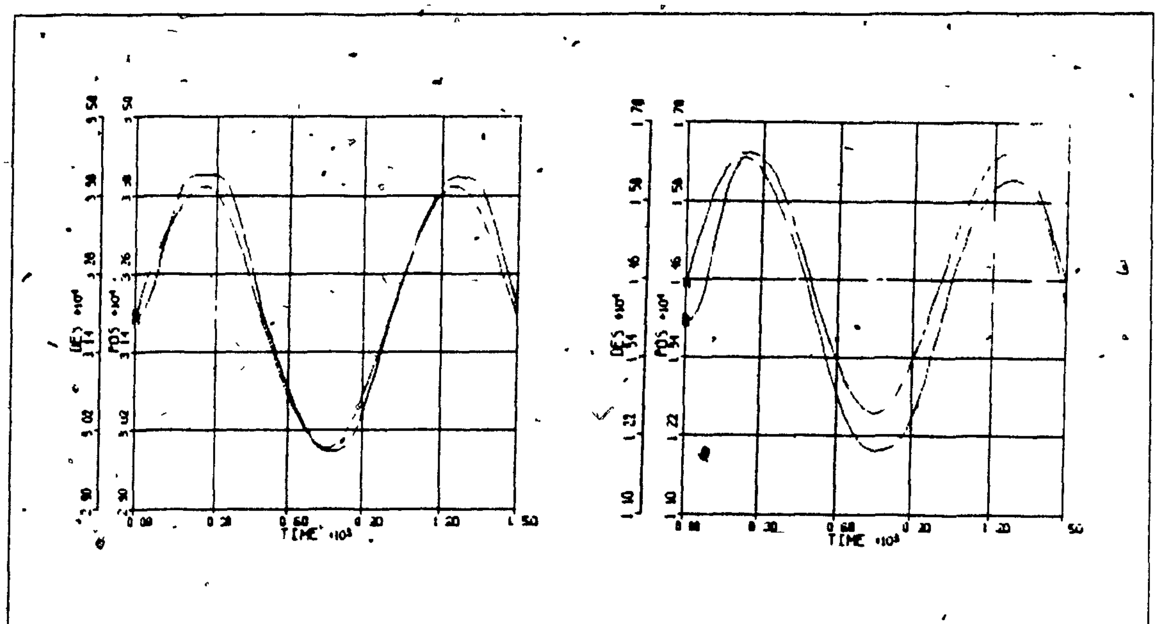

Figure 5.11b Sinusoidal Tracking. Unloaded (left), Loaded (right)

Figure 5.11 Effects of Inertia and Gravity: Joint 2

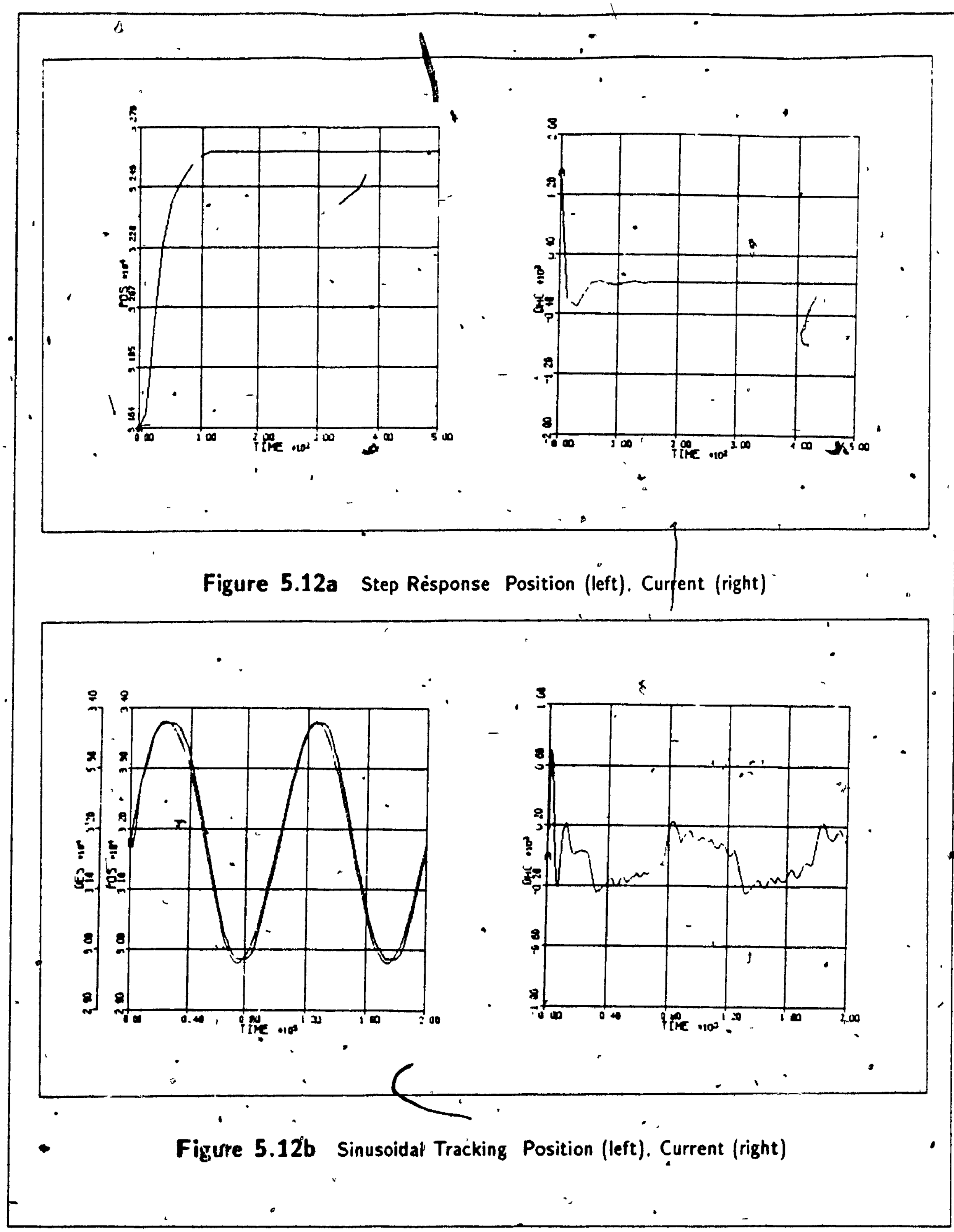

Figure 5.12a Step Response Position (left), Current (right)

Figure 5.12b Sinusoidal Tracking Position (left), Current (right)

Figure 5.12 Effects of Friction Joint 6

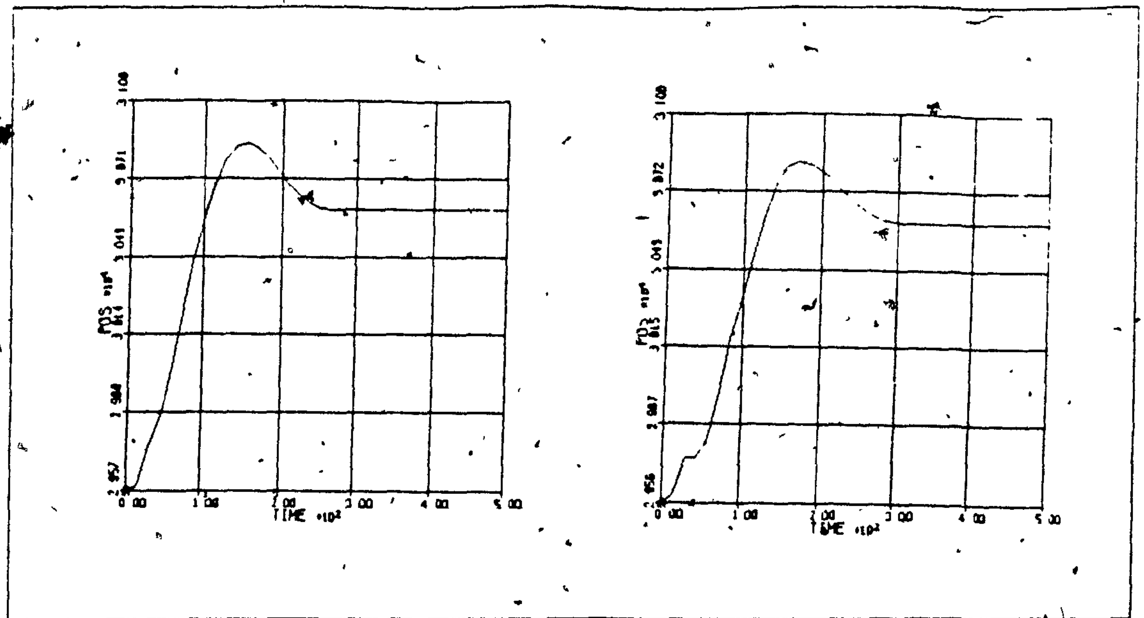

Figure 5.13a Step Response: PD (left), Acceleration Feedback (right)

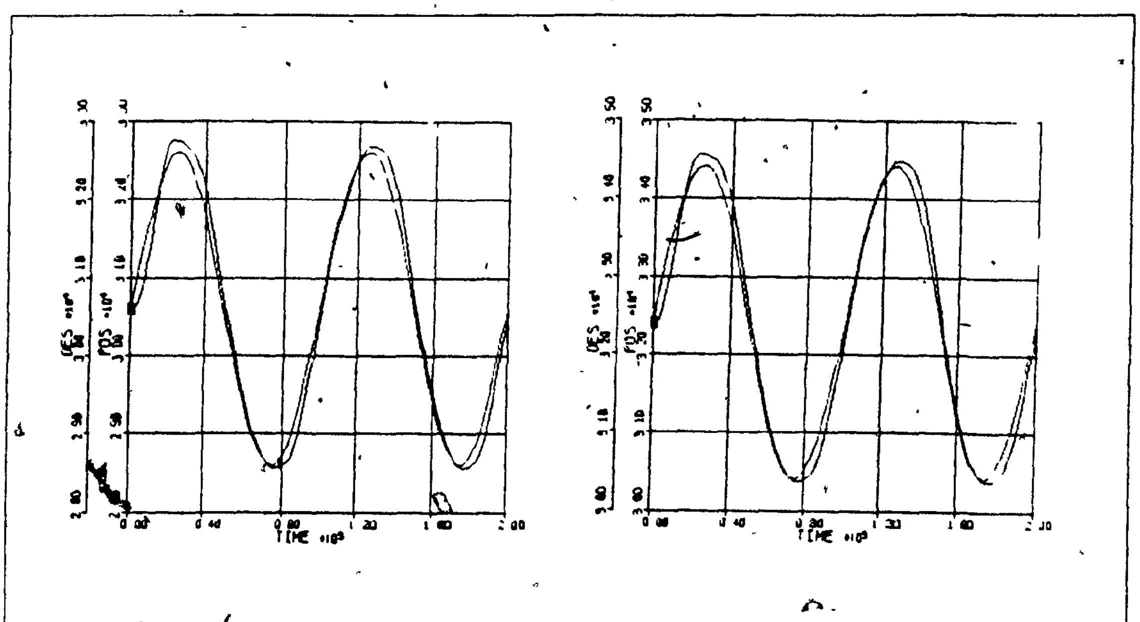

Figure 5.13b . Sinusoidal Tracking PD (left), Acceleration Feedback (right)

Figure 5.13 PD vs Acceleration Feedback Joint 1, Small Inertia Disturbance

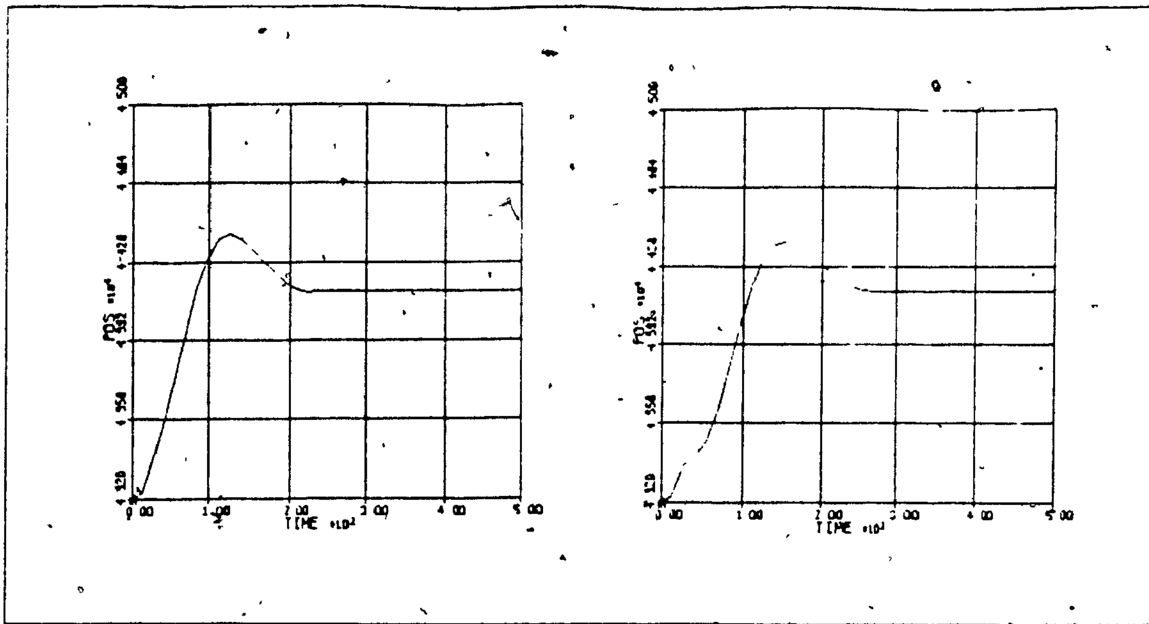

Figure 5.14a Step Response PD (left), Acceleration Feedback (right)

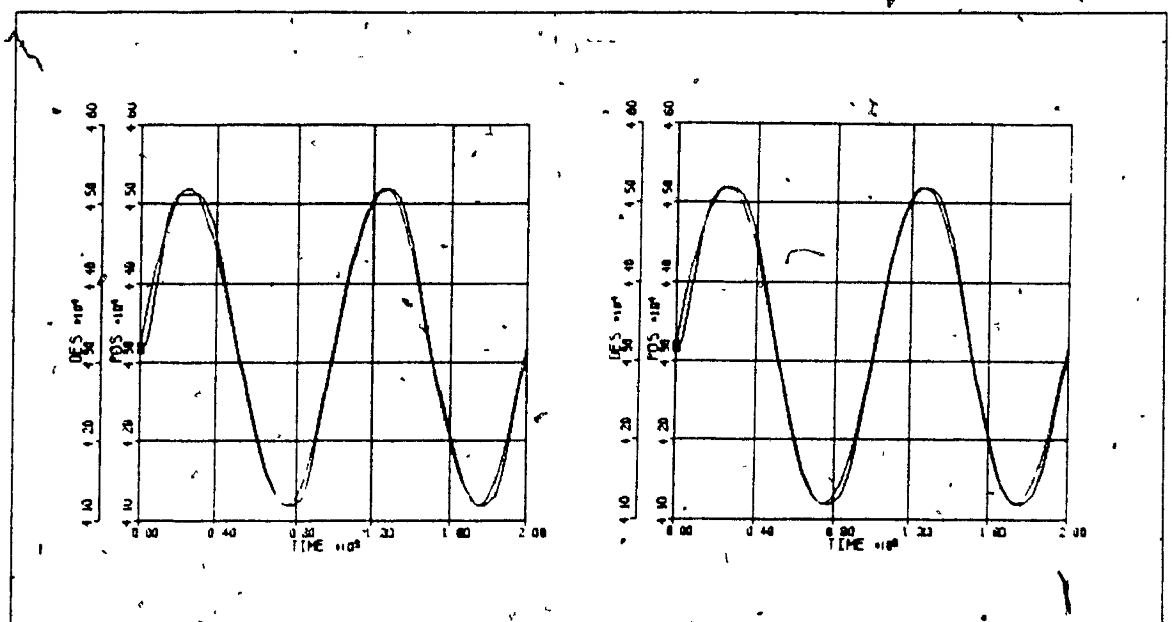

Figure 5.14b Sinusoidal Tracking PD (left), Acceleration Feedback (right)

Figure 5.14 PD vs Acceleration Feedback, Joint 3, Small Gravity Disturbance

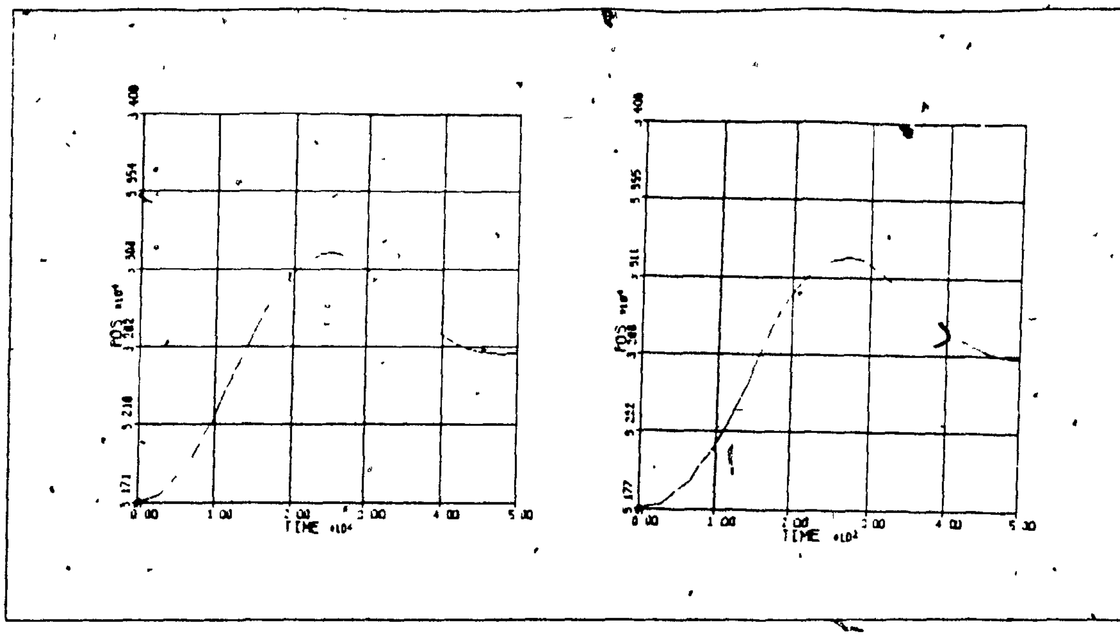

Figure 5.15a Step Response PD (left). Acceleration Feedback (right)

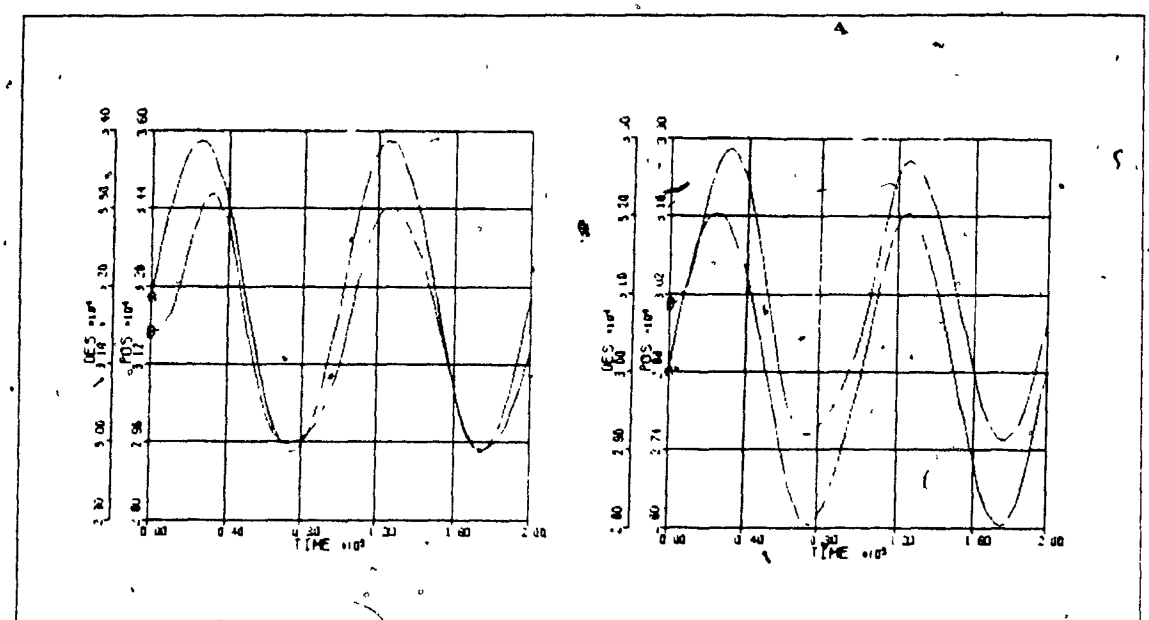

Figure 5.15b Sinusoidal Tracking PD (left). Acceleration Feedback (right)

Figure 5.15 PD vs Acceleration Feedback Joint 1. Large Inertia Disturbance

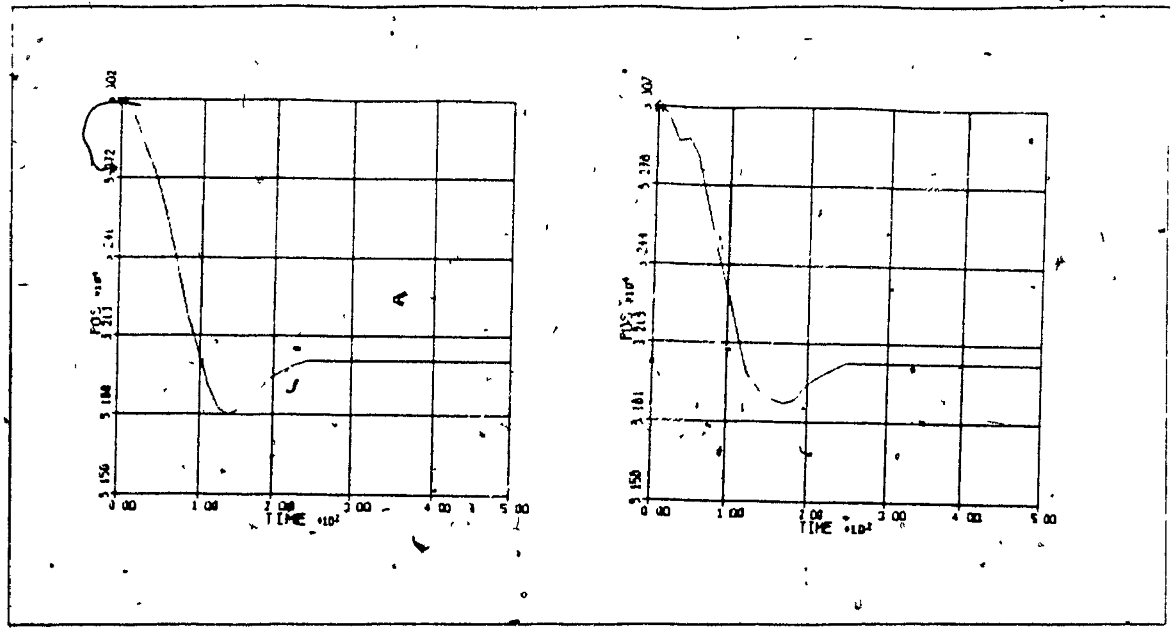

Figure 5.16a Step Response. PD (left), Acceleration Feedback (right)

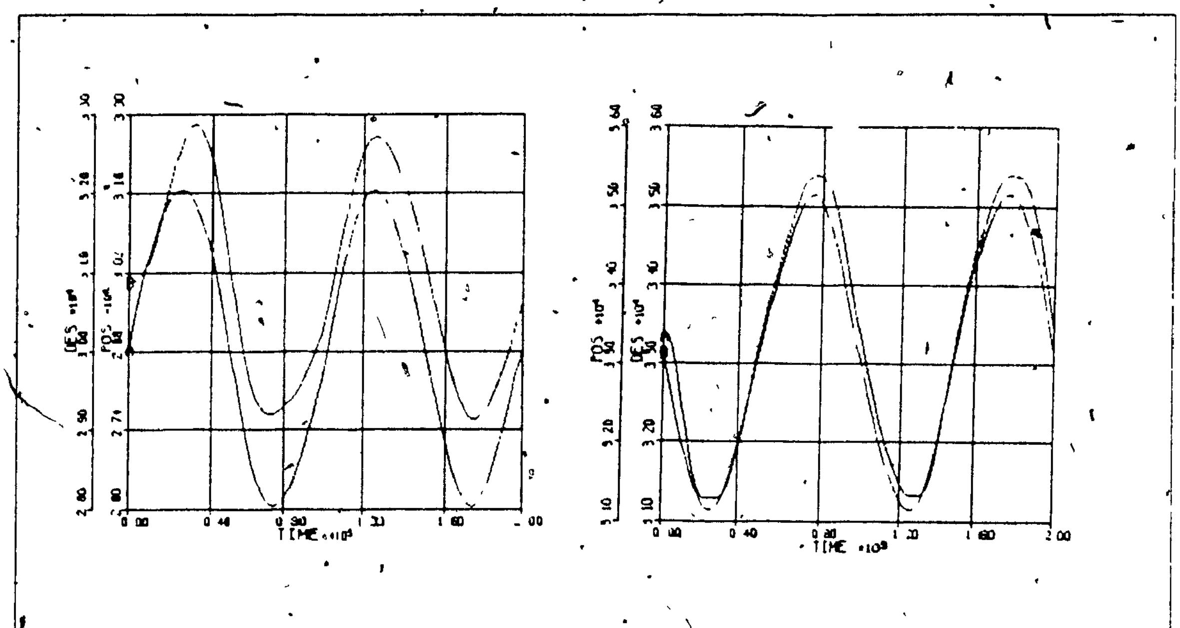

Figure 5.16b Sinusoidal Tracking. PD (left). Acceleration Feedback (right)

Figure 5.16 PD vs Acceleration Feedback: Joint 3, Large Gravity Disturbance

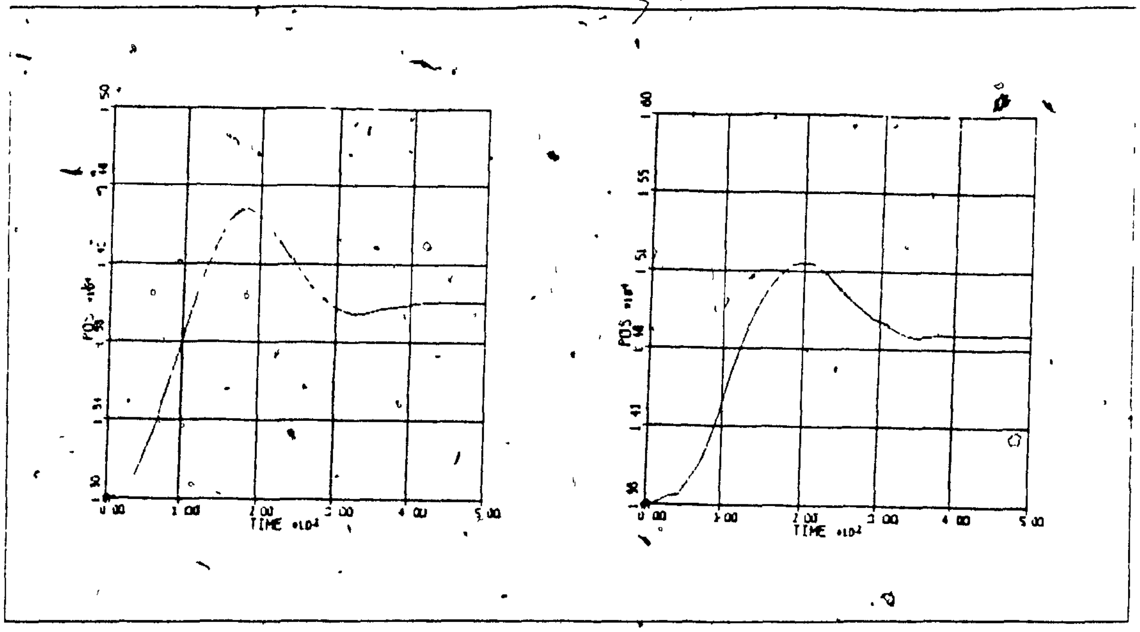

Figure 5.17a Step Response. PD (left) Acceleration Feedback (right)

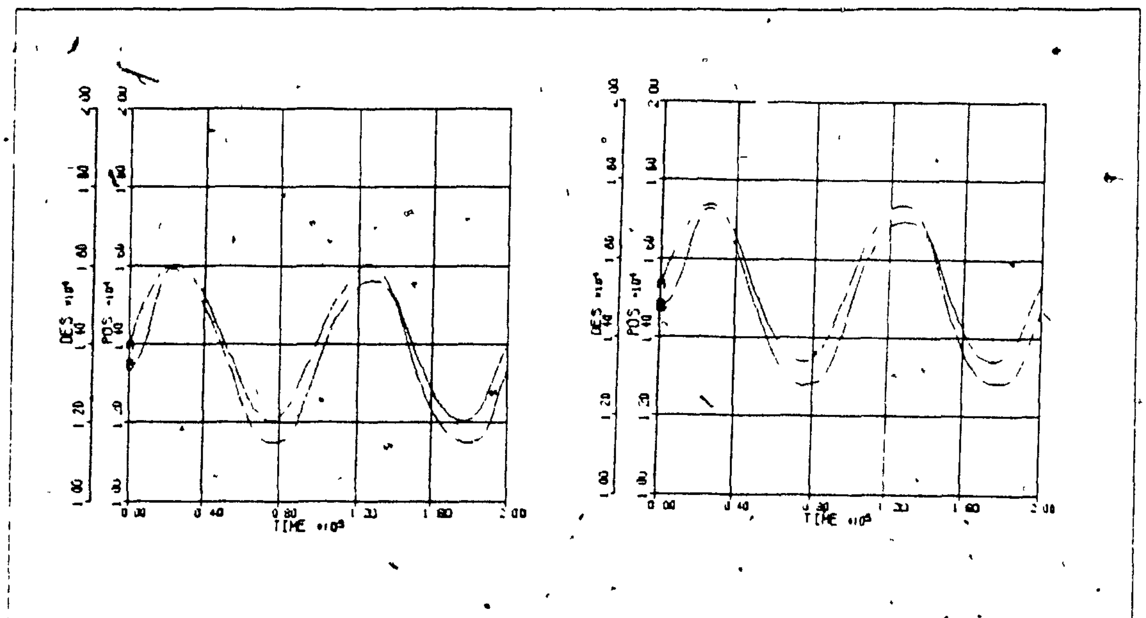

Figure 5.17b Sinusoidal Tracking PD (left). Acceleration Feedback (right)

Figure 5.17 PD vs Acceleration Feedback Joint 2. Large Gravity and Inertia Disturbance

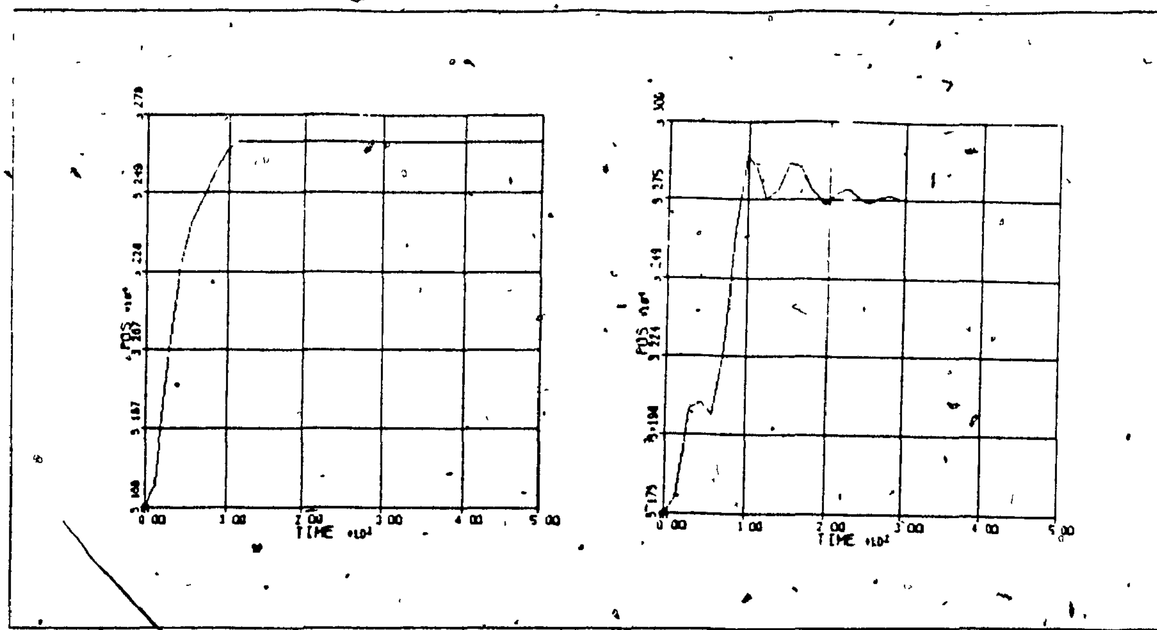

Figure 5.18a Step Response PD (left). Acceleration Feedback (right)

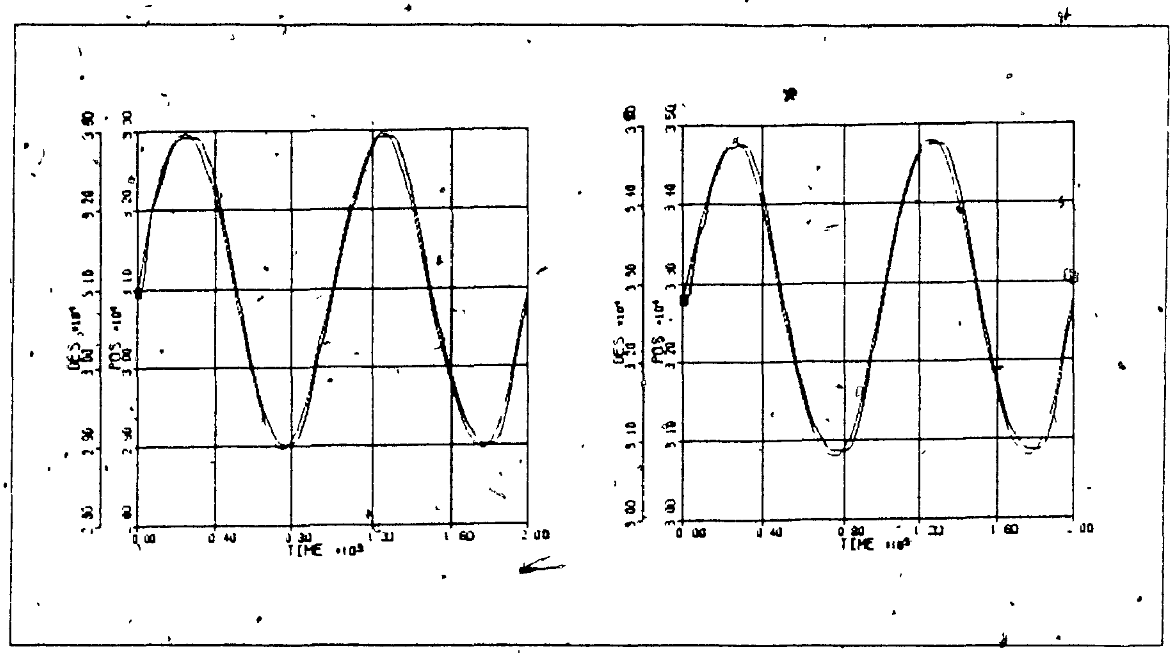

Figure 5.18b Sinusoidal Tracking PD (left). Acceleration Feedback (right)

Figure 5.18 PD vs Acceleration Feedback Joint 6. Large Friction Disturbance

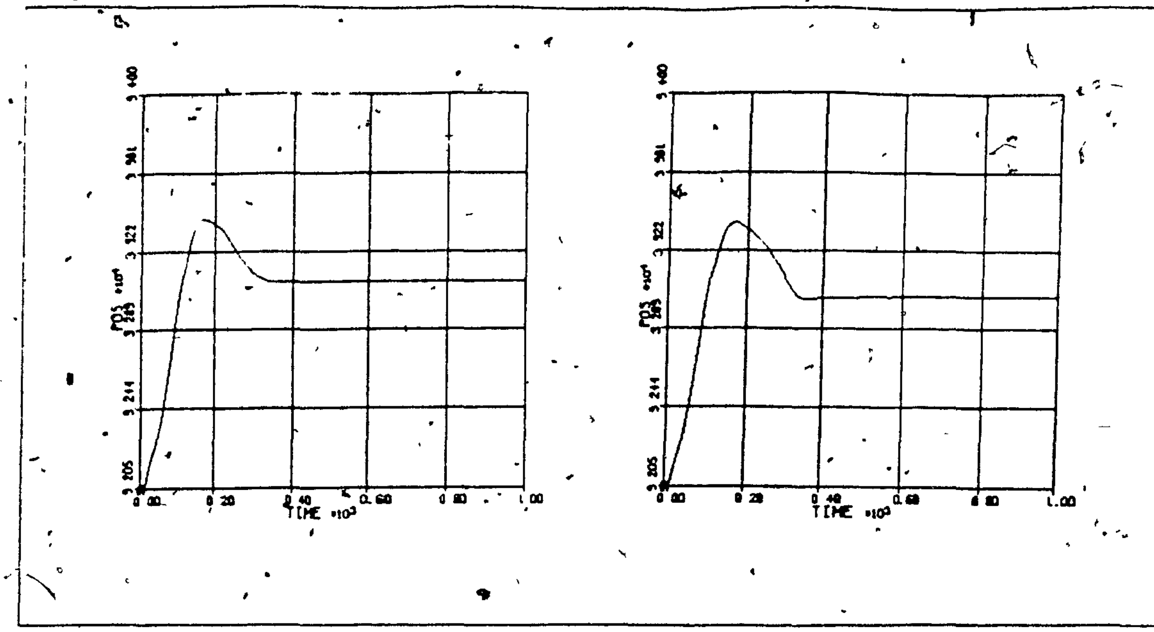

Figure 5.19a Arm Up: Joint 3 Still (left). Low Freq Sine (right)

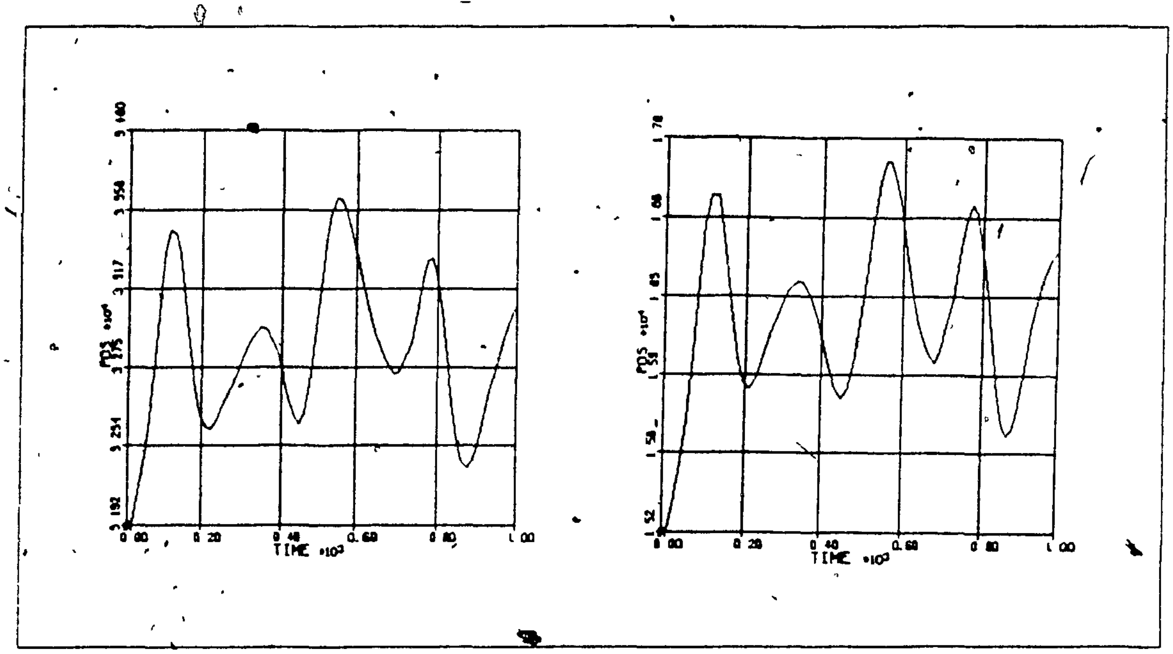

Figure 5.19b Joint 3 High Freq Square. Arm Up (left). Out (right)

Figure 5.19 Cross Coupling Effects Step Response of Joint 2

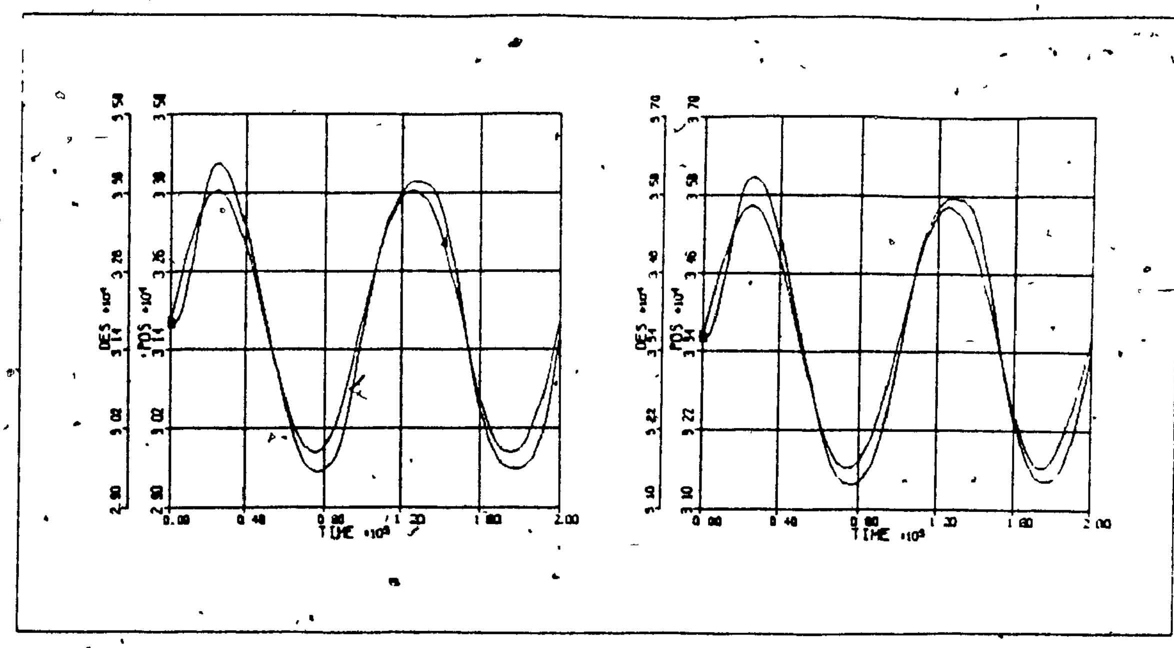

Figure 5.20a Arm Up, Joint 3: Still (left), Low Freq Sine (right)

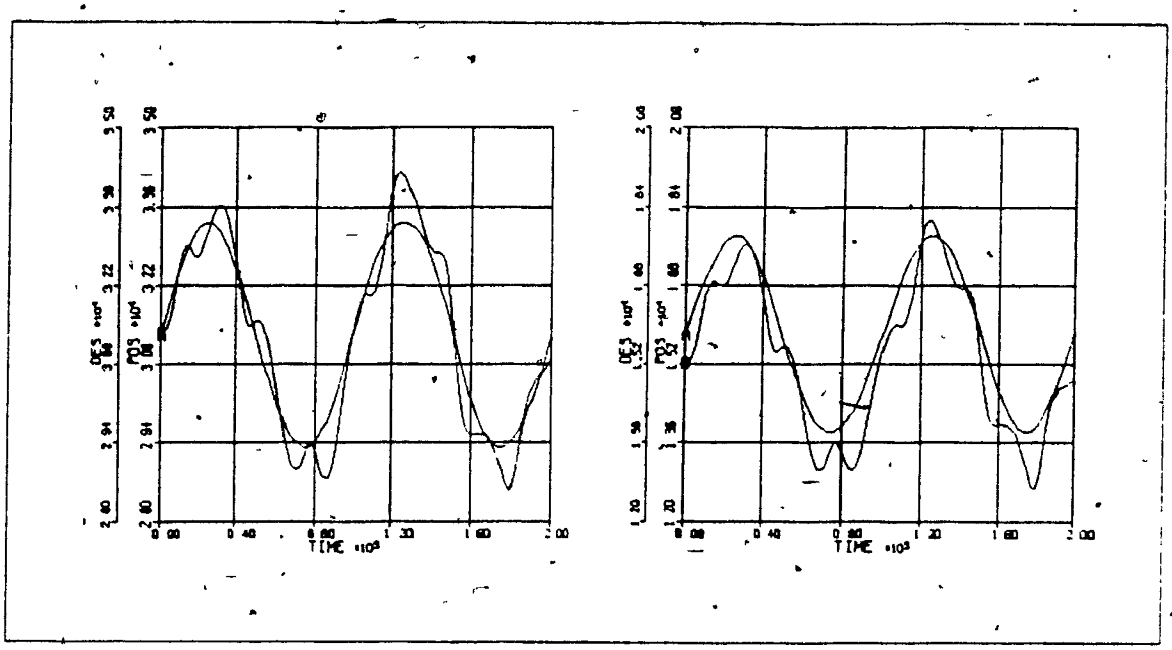

Figure 5.20b Joint 3 High Freq Square, Arm Up (left), Out (right)

Figure 5.20 Cross Coupling Effects Sinusoidal Tracking of Joint 2

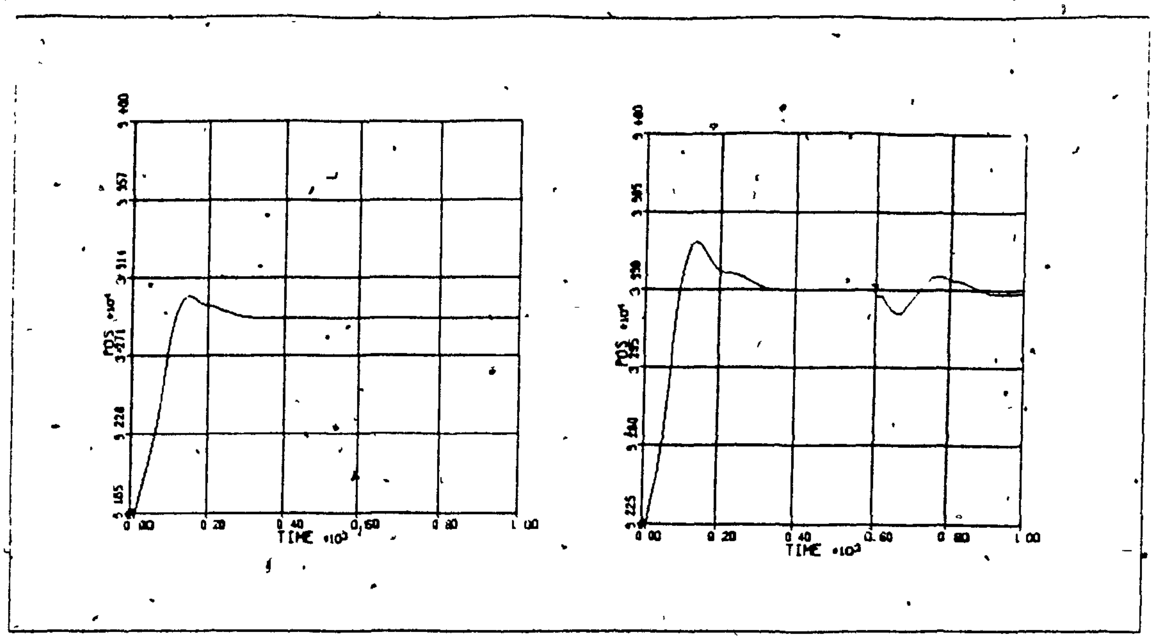

Figure 5.21a Arm Up, Joints 2,3 Still (left), High Freq Square (right)

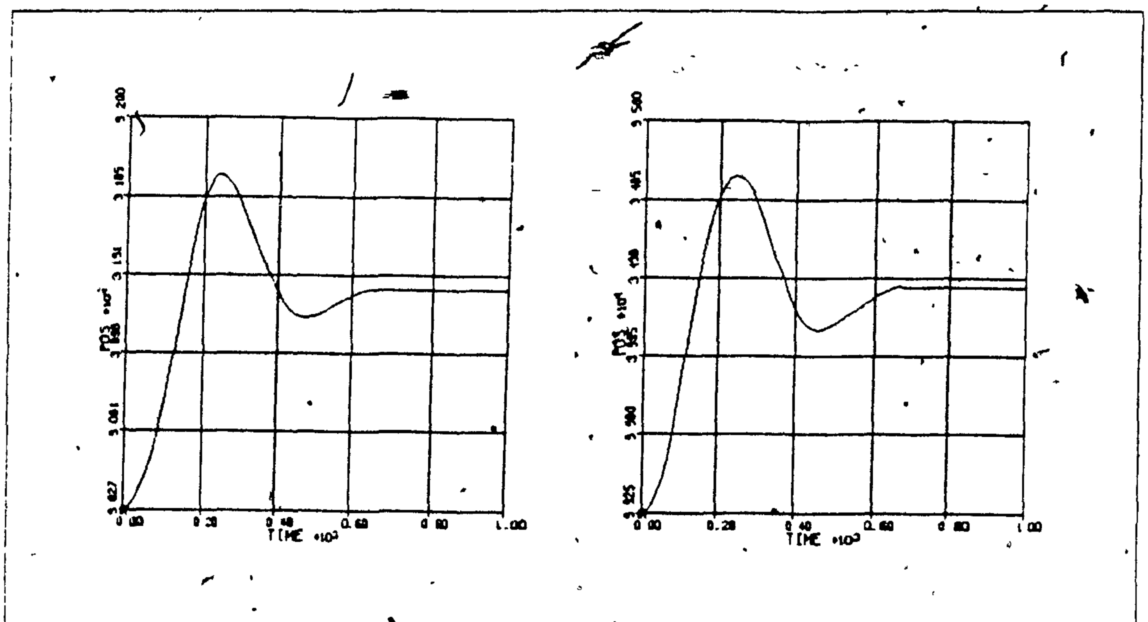

Figure 5.21b Arm Out, Joints 2,3 Still (left), High Freq Square (right)

Figure 5.21 Cross Coupling Effects Step Response of Joint 1

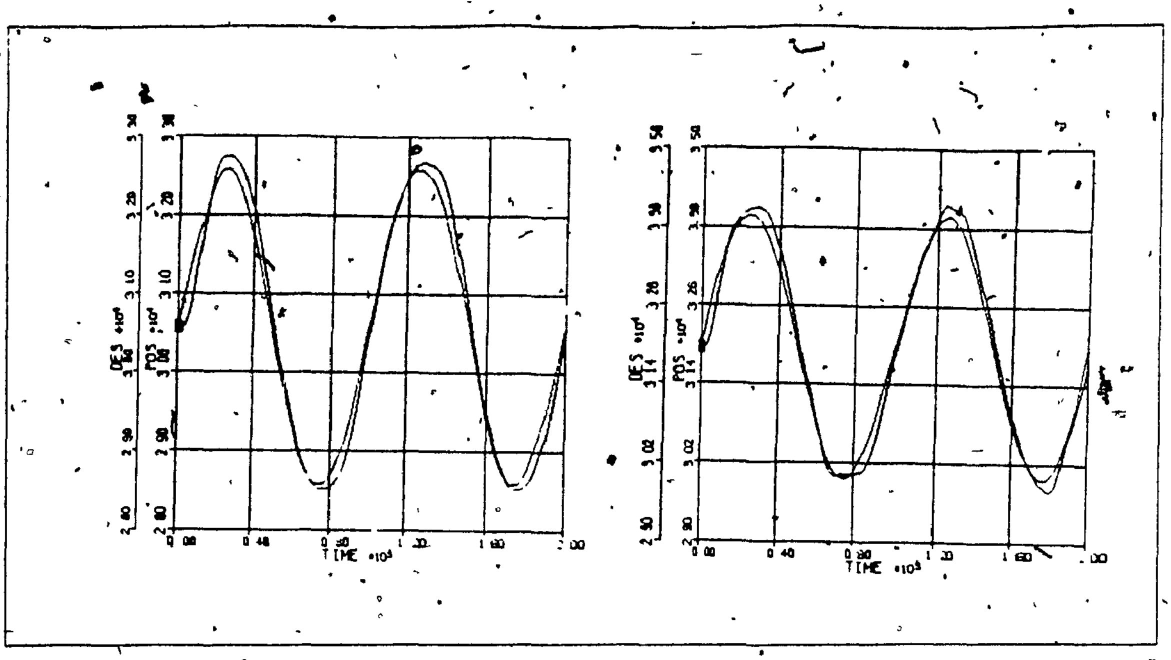

Figure 5.22a Arm Up. Joints 2,3 Still (left). High Freq Square (right)

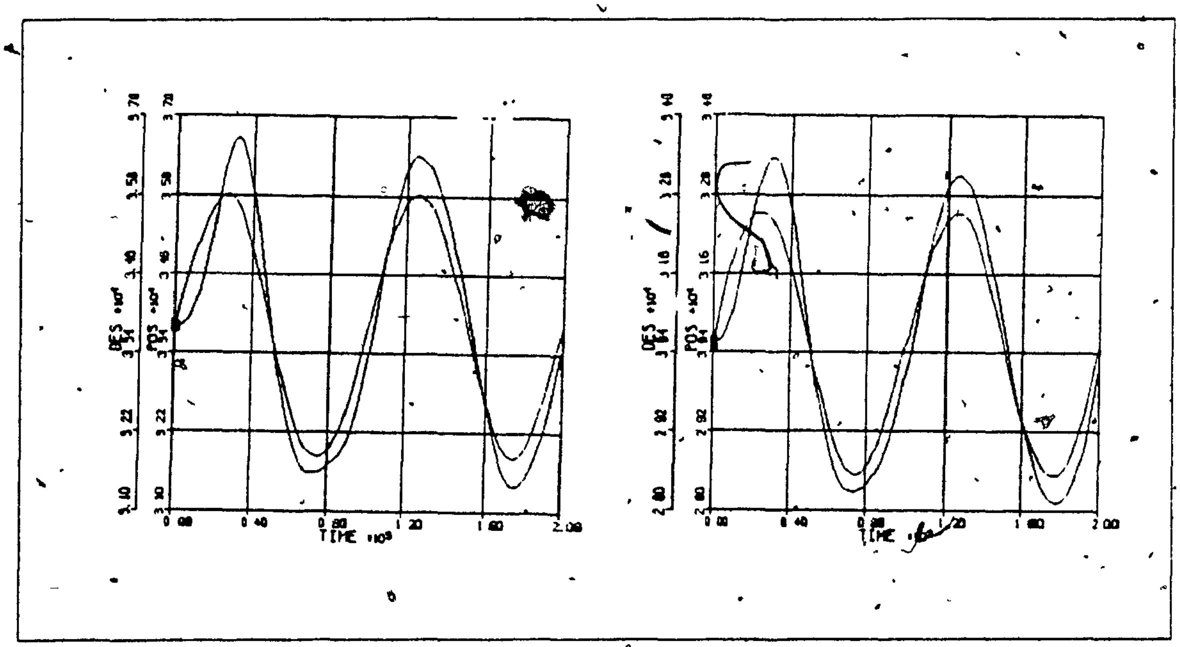

Figure 5.22b Arm Out. Joints 2,3 Still (left). High Freq Square (right)

Figure 5.22 Cross Coupling Effects: Sinusoidal Tracking of Joint 1

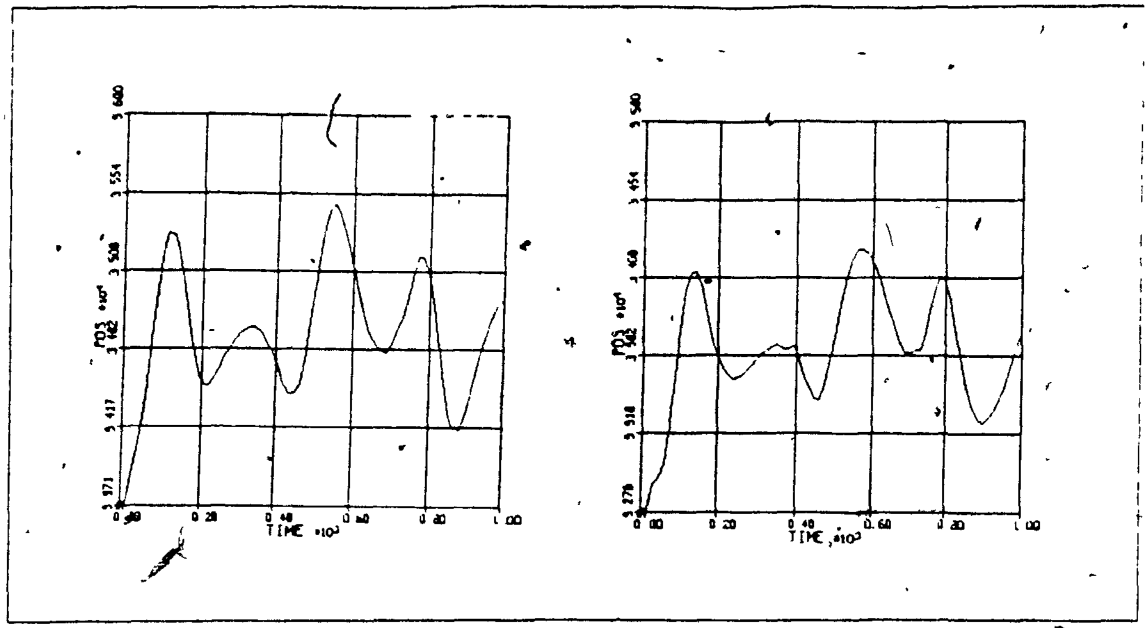

Figure 5.23a Step Response, PD (left), Acceleration Feedback (right)

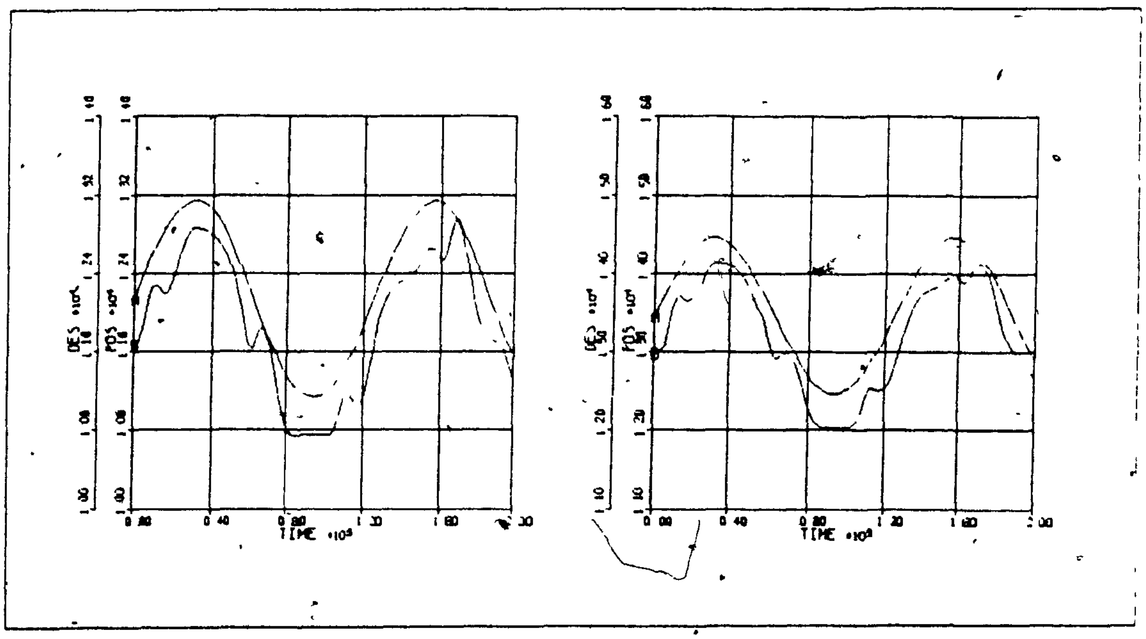

Figure 5.23b Sinusoidal Tracking, PD (left), Acceleration Feedback (right)

Figure 5.23 PD vs Acceleration Feedback Joint 2, Large Cross Coupling Disturbance, Arm Up

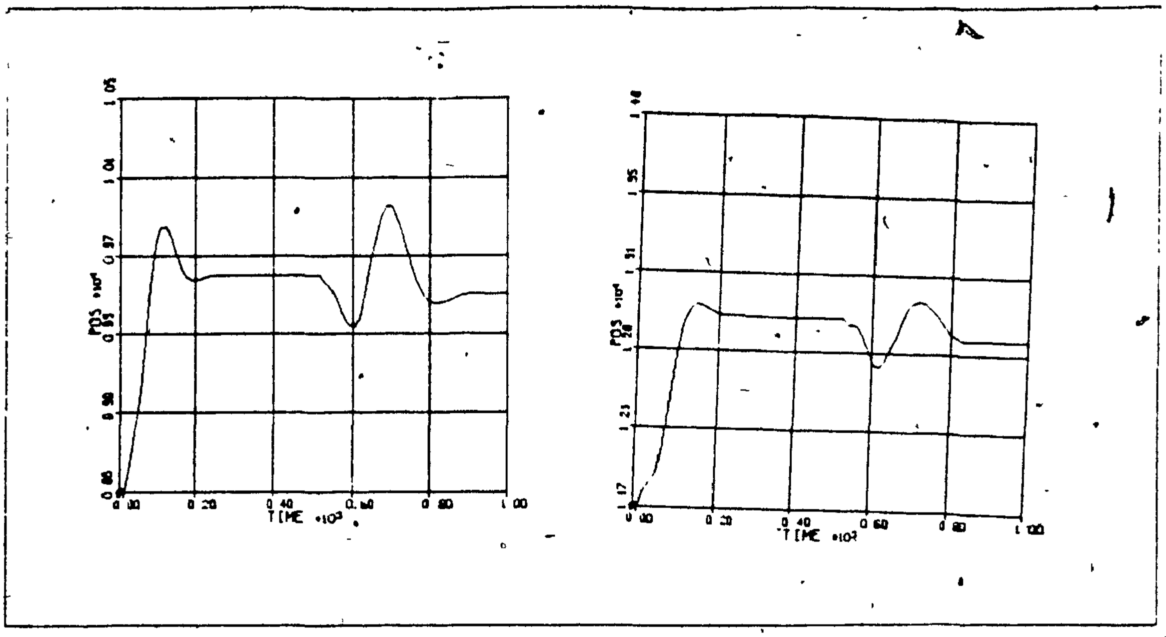

Figure 5.24a Step Response, PD (left), Acceleration Feedback (right)

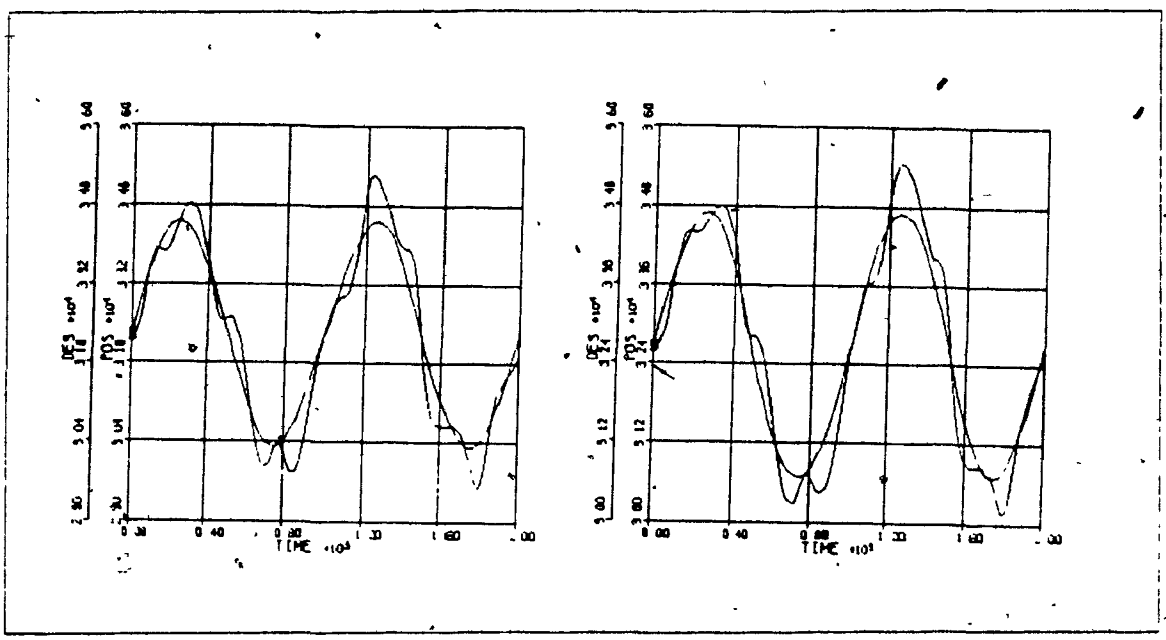

Figure 5.24b Sinusoidal Tracking, PD (left), Acceleration Feedback (right)

Figure 5.24 PD vs Acceleration Feedback Joint 2, Large Cross Coupling Disturbance, Arm Out

The next set of experiments examined the performance of acceleration feedback and PD using different gains and under different types of disturbances. The purpose of this section was to provide a complete investigation of the relative performance of the two schemes in a large range of their operation, and to ensure that both schemes were being tested at the same level of optimality of tuning parameters

The first experiment in this set dealt with varying the tuning gains. It is expected that the damping characteristics of the system under acceleration feedback control will not vary with the high gain feedback K_u but only with the tuning gains A_0 and A_1. It is therefore also expected that acceleration feedback with a given set of tuning parameters will give the ~~same~~ damping characteristics and performance as PD with the same gains, however the high gain feedback K_u should allow us to expect a smaller following error This experiment compared the performance of the two schemes in the following manner First, PD was tested with two sets of gains, a factor of four greater than each other. Then acceleration feedback was tested without varying the tuning gains, however letting the high gain feedback K_u vary by a factor of four. It was expected to see different performance for these conditions, due to the hypothesis mentioned above

The next experiment also dealt with variation in gain, but now under conditions of large disturbances Also in this section, the response of a joint moving at high frequency with different tuning gains was evaluated as well. Finally, a joint moving at high frequency while disturbed by a large cross coupling motion of another joint was also examined

The third experiment in this set involved high frequency inertial disturbances Now the disturbance, and not the joint being tested, were delivered at a high frequency, and the effect on the response of the joint was measured Two sets of conditions were tested in this experiment. In the first the inertial disturbance was provided by the arm alone. In the second set, a one kilogram weight was attached to the end effector, increasing the magnitude of the inertial disturbance

The final experiment in the second set involved changing the damping term of both control laws to examine its effect on tracking error. In theory, increasing the damping term (K_d or A_1) should have the effect of reducing the tracking error. Furthermore, it is expected that the inclusion of an acceleration term should allow the use of a higher damping constant under acceleration feedback control than under PD.

These experiments were carried out to demonstrate the performance of the two control algorithms under conditions and in regimes in which their performance is expected to be different. Motivation for these experiments came from the first set of experiments, in which the performance of the two schemes was demonstrated to be similar. The second set of experiments attempts to explore the conditions under which their performance will differ, and to explain the reasons for the similar performance observed in the first set.

The results are presented graphically in the next few pages, and an analysis follows. Note that in the case of graphs with multiple variables plotted, the plots are always of desired vs actual position, and always sinusoidal. In every such plot, the desired position appears as a perfect sine wave starting at time = 0, and the actual position typically starts after a delay, and exhibits distortion and overshoot. The units of the graphs list time in milliseconds and positions in encoder counts. The analysis that follows refers to the figures, as well as to quantitative error analysis results which appear in tables in next section.

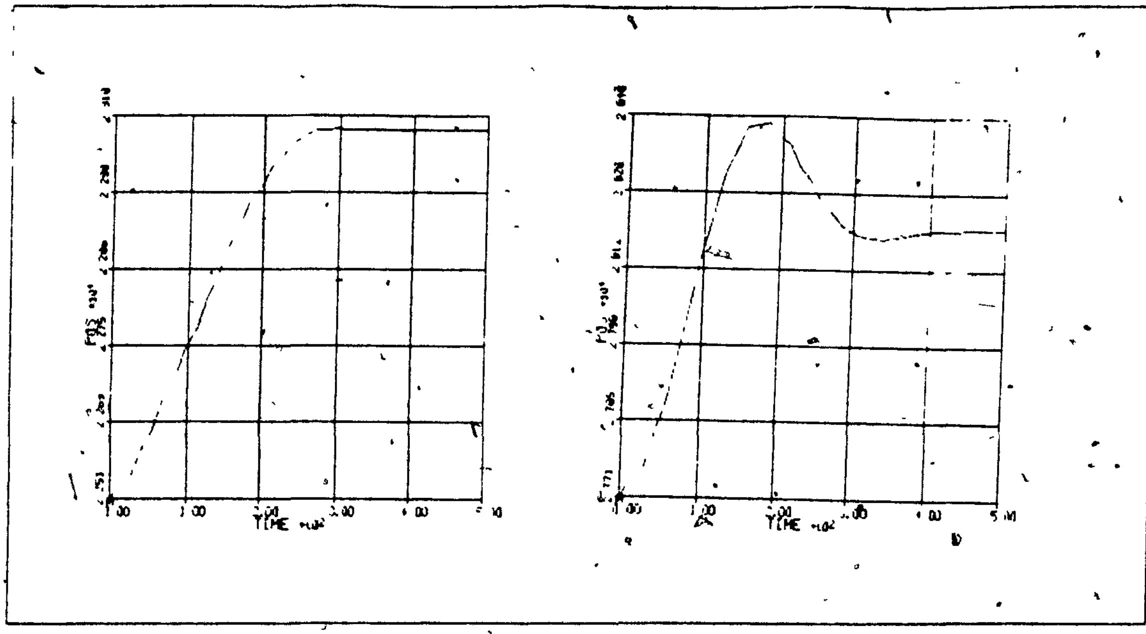

Figure 5.25a PD. K_p=0.5, K_d=1.0 (left), K_p=2.0, K_d=4.0 (right)

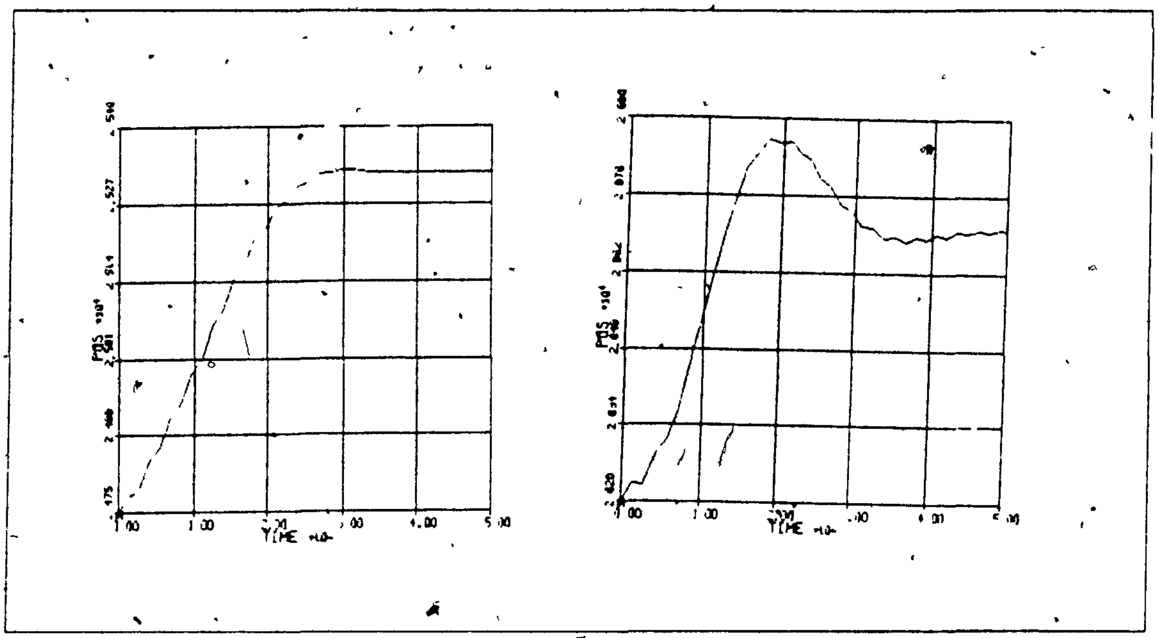

Figure 5.25b Acceleration Feedback. K_u=0.5 (left), K_u=2.0 (right)

Figure 5.25 Step Response Joint 1. Effects of Variation in Gain

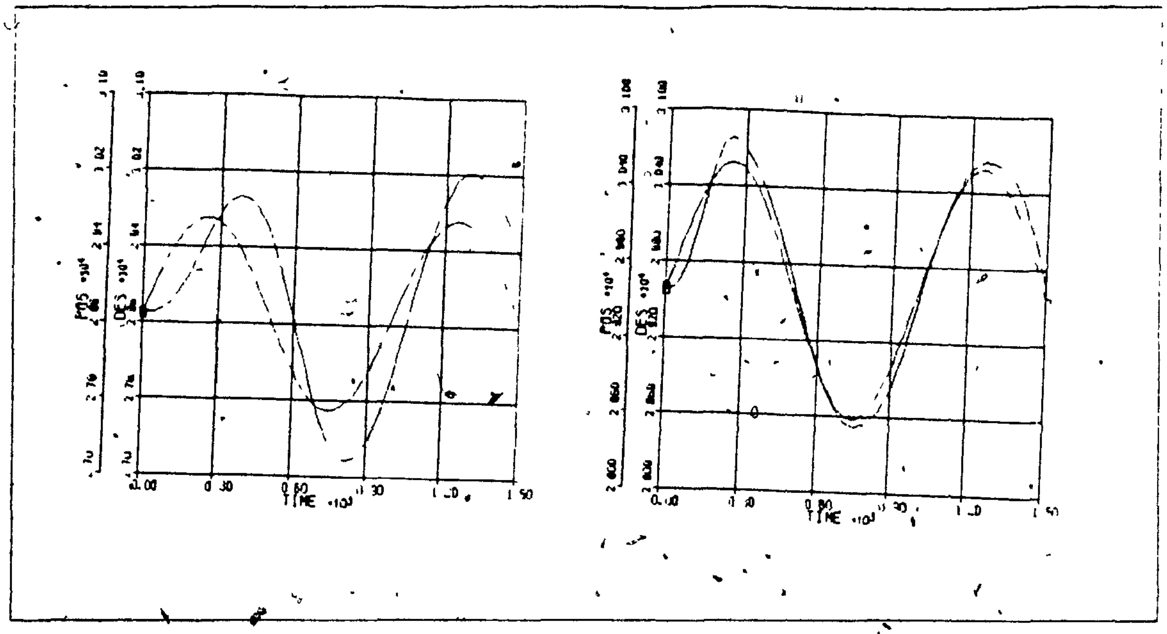

Figure 5.26a PD. K_p=0.5, K_d=1.0 (left). K_p=2.0, K_d=4.0 (right)

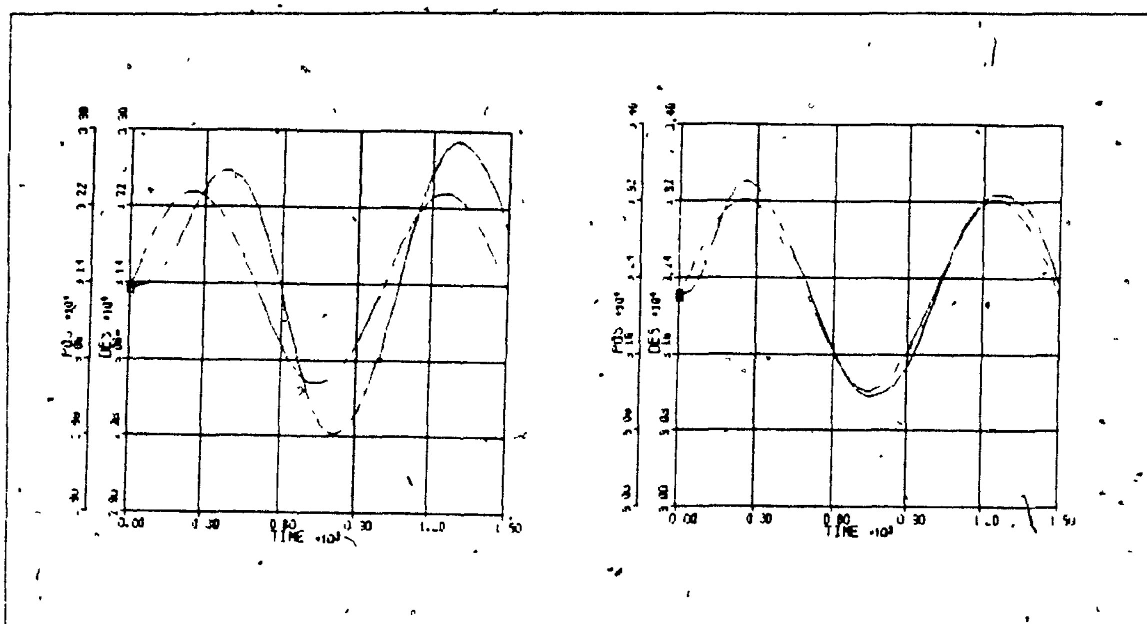

Figure 5.26b Acceleration Feedback K_u=0.5 (left). K_u=2.0 (right)

Figure 5.26 Sinusoidal Tracking Joint 1. Effects of Variation in Gain

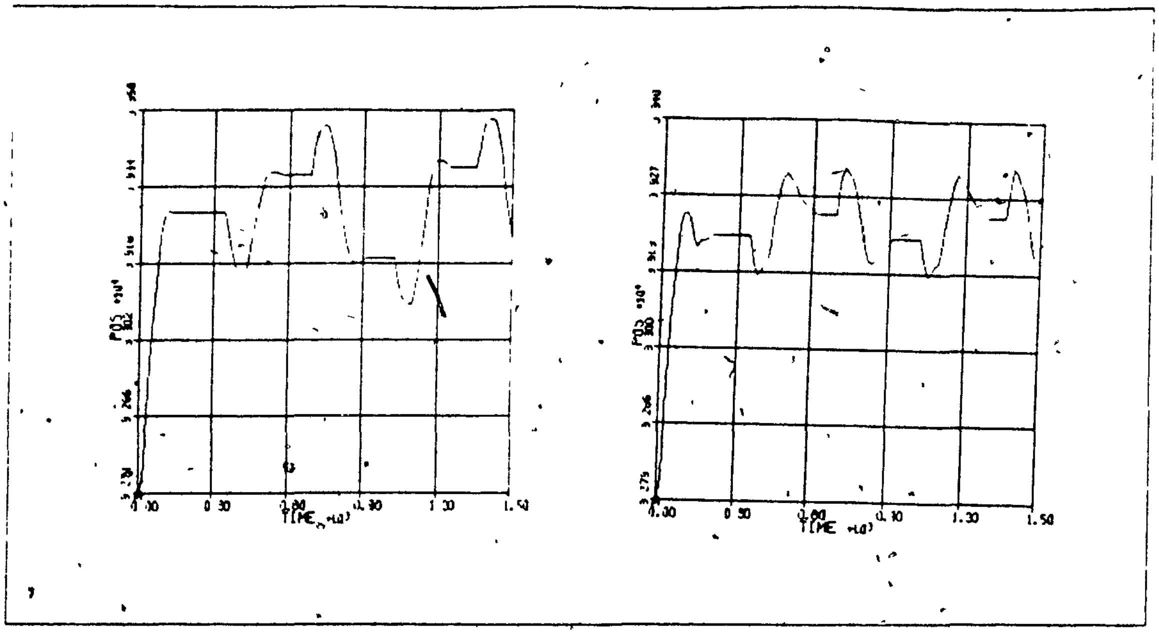

Figure 5.27a PD K_p=0.5. K_d=1 0 (left). K_p=2 0. K_d=4 0 (right)

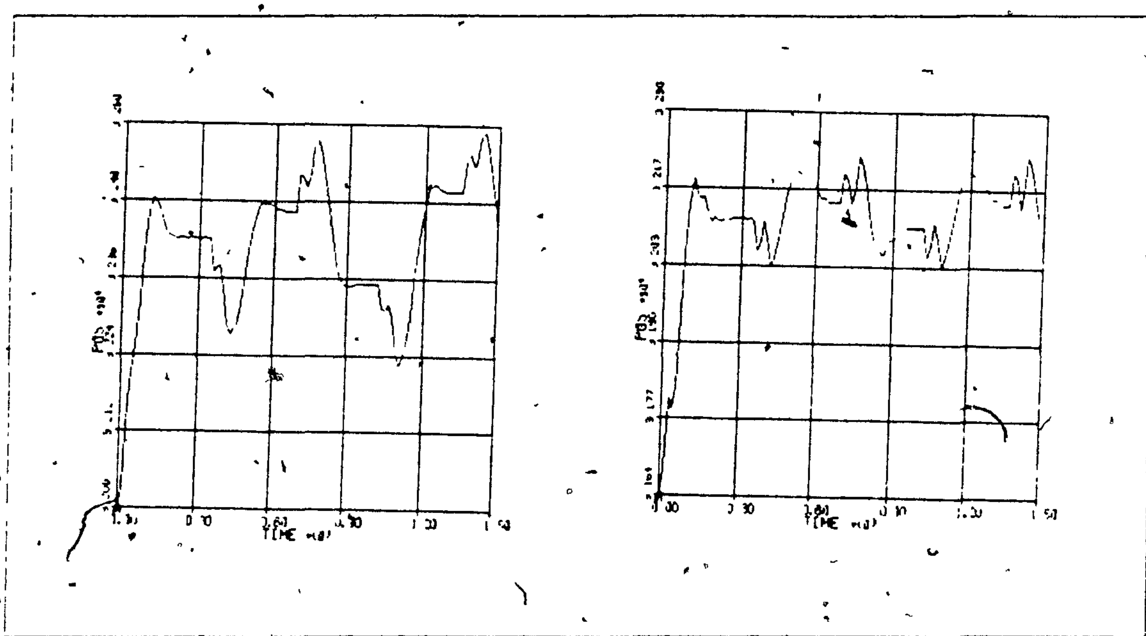

Figure 5.27b Acceleration Feedback K_u=0 5 (left). K_u=2.0 (right)

Figure 5.27 Step Response Joint 2. Effects of Variation in Gain. Large Disturbance

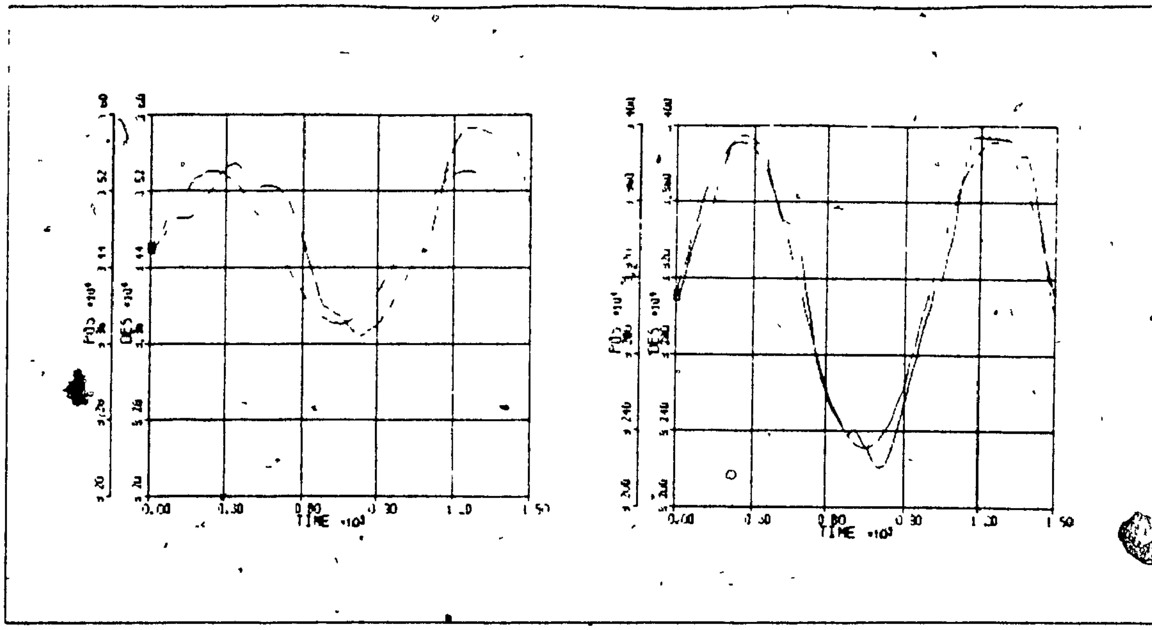

Figure 5.28a PD: $K_p=0.5$, $K_d=1.0$ (left), $K_p=2.0$, $K_d=4.0$ (right).

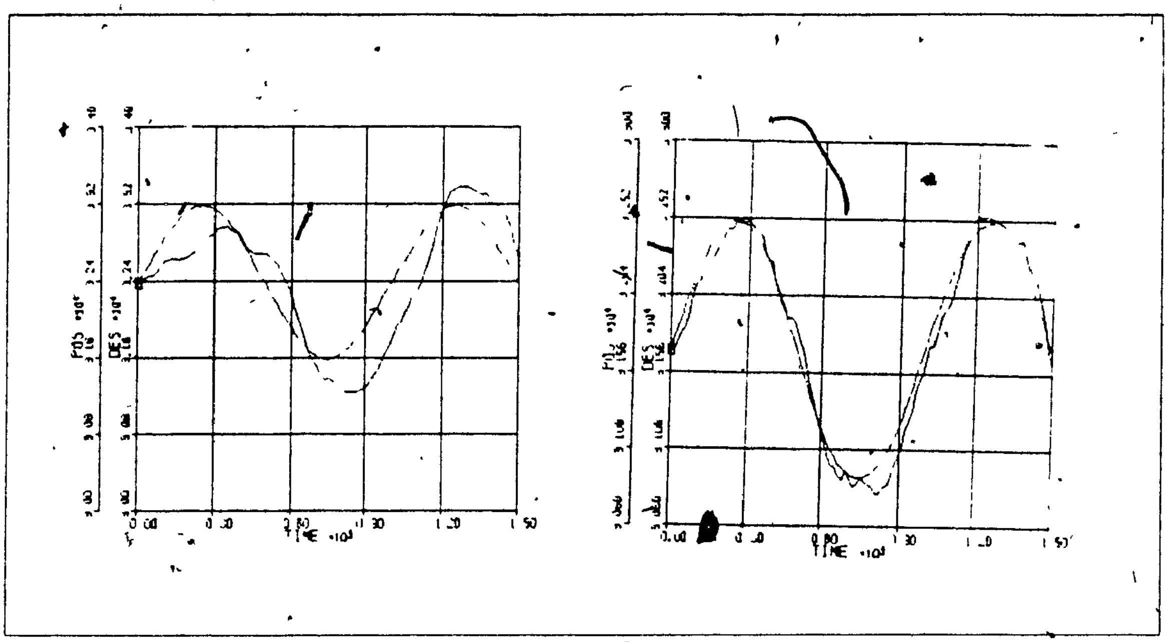

Figure 5.28b Acceleration Feedback $K_u=0.5$ (left), $K_u=2.0$ (right)

Figure 5.28 Sinusoidal Tracking Joint 2, Effects of Variation in Gain, Large Disturbance

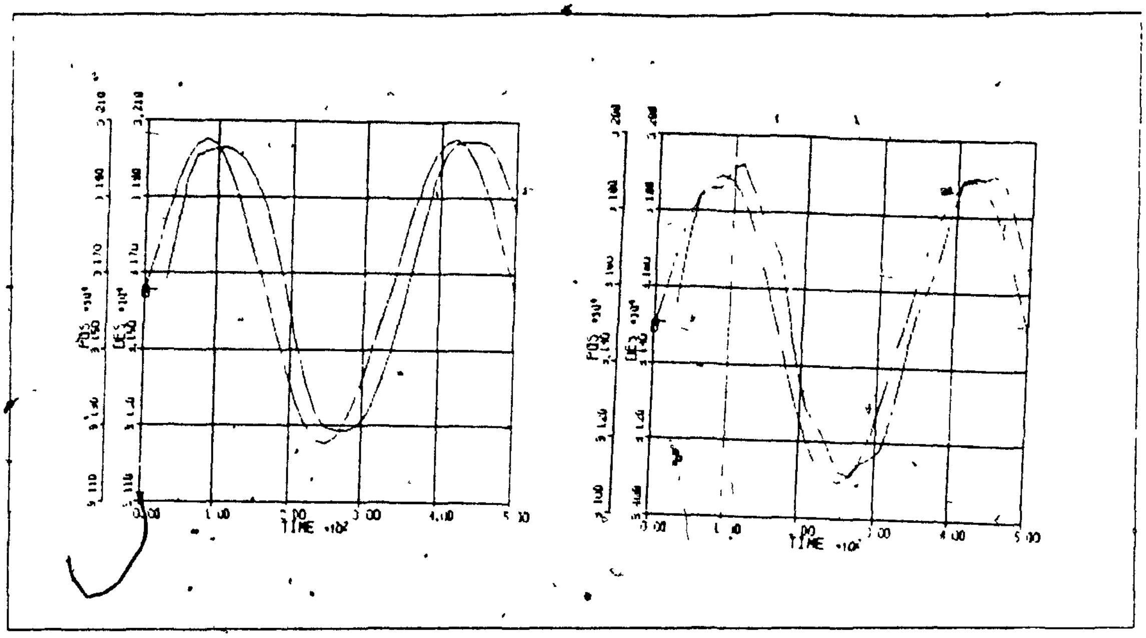

Figure 5.29a PD K_p=1 0, K_d=2 0 (left), K_p=1 3, K_d=1 3 (right)

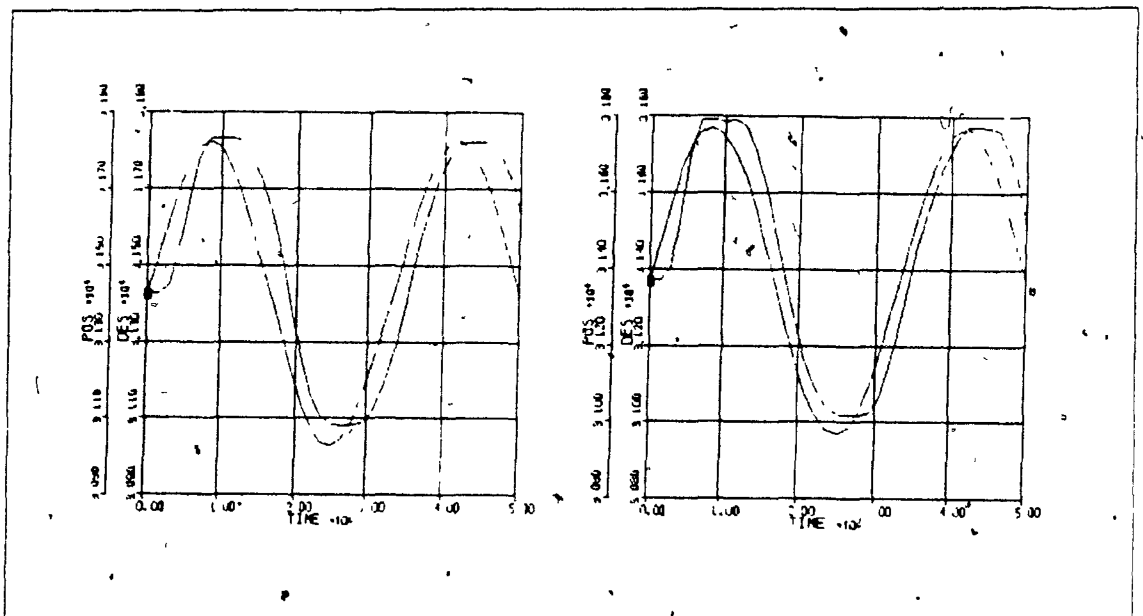

Figure 5.29b Acceleration Feedback K_u=1 0 (left), K_u=1 3 (right)

Figure 5.29 Sinusoidal Tracking Joint 6, Effects of Variation in Gain, High Frequency

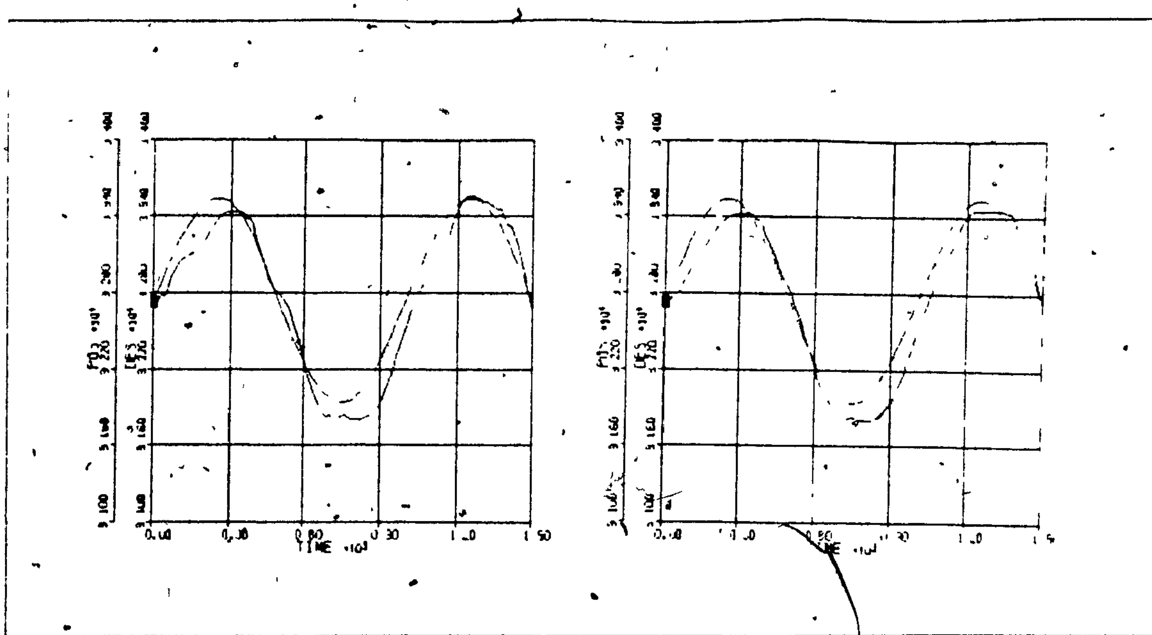

Figure 5.30a $Ku=1\,0, A_0=1\,0, A_1=2\,0$ (left), $Ku=2\,0, A_0=0\,5\ A_1=1\,0$ (right)

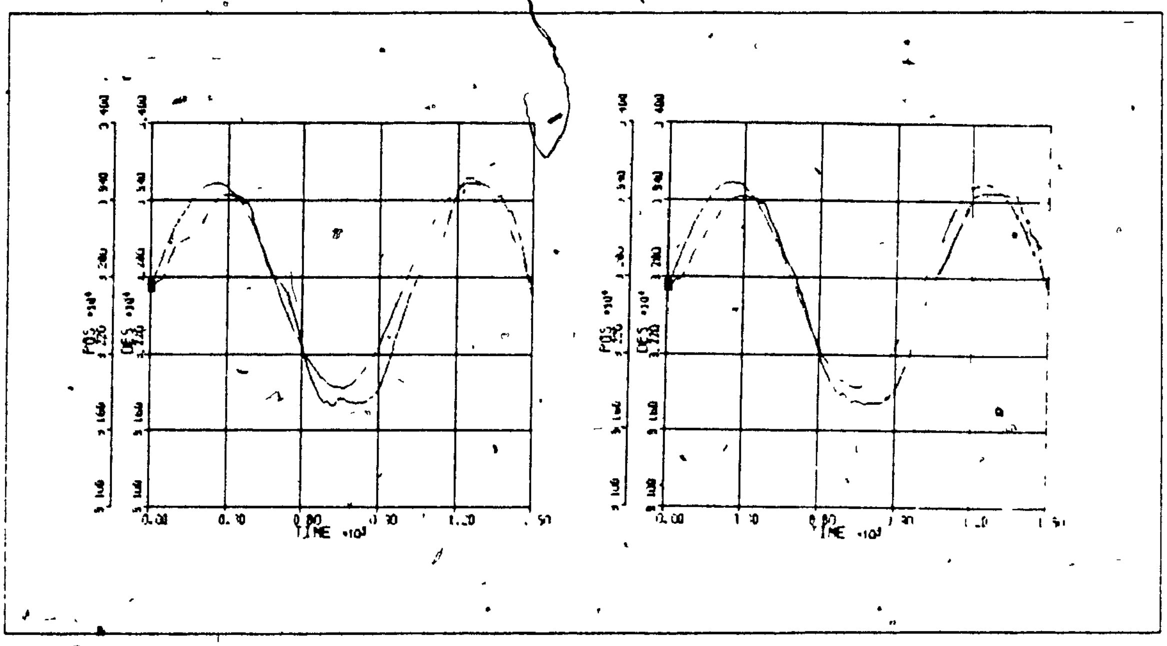

Figure 5.30b $Ku=1\,0, A_0=1\,0, A_1=2\,0$ (left), $Ku=2.0, A_0=0\,5, A_1=1\,0$ (right)

Figure 5.30 Sinusoidal Tracking Joint 2. Effects of Variation in Gain, Large Disturbance, High Frequency

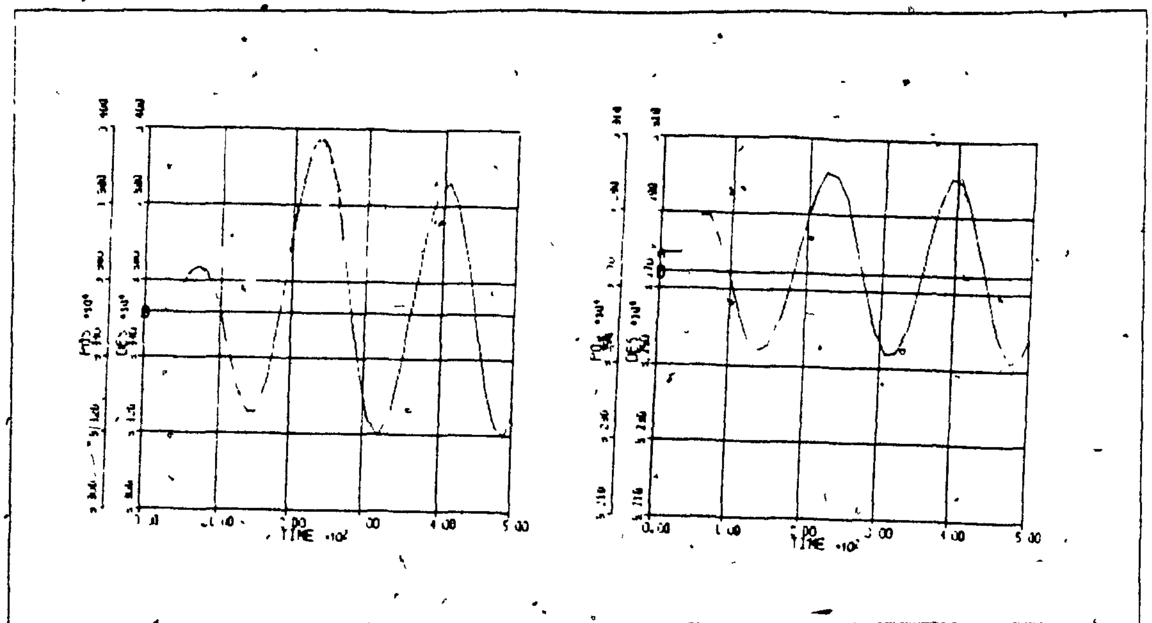

Figure 5.31a Servo PD (left). Acceleration Feedback (right)

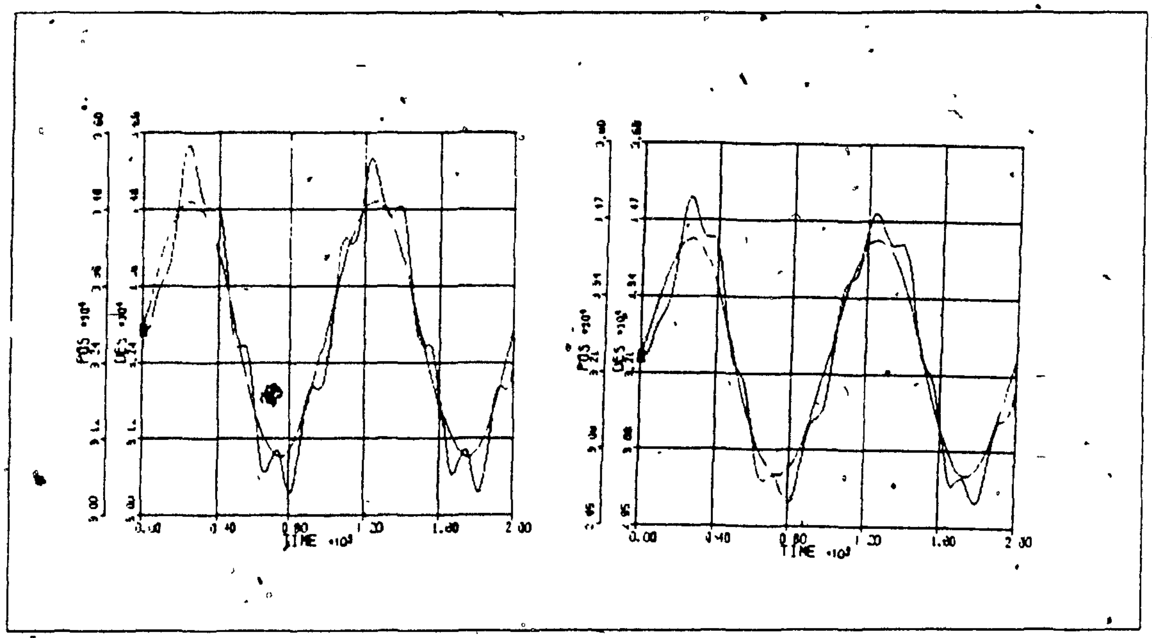

Figure 5.31b Sinusoidal Tracking PD (left). Acceleration Feedback (right)

Figure 5.31 Joint 2. High Frequency Disturbance Rejection

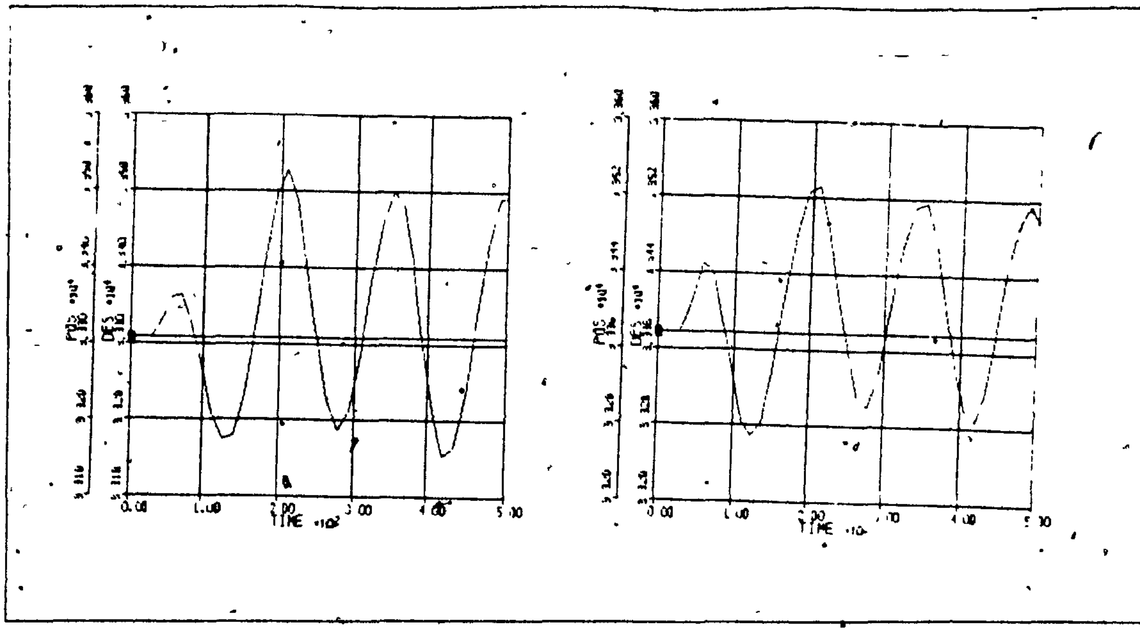

Figure 5.32a Servo PD (left), Acceleration Feedback (right)

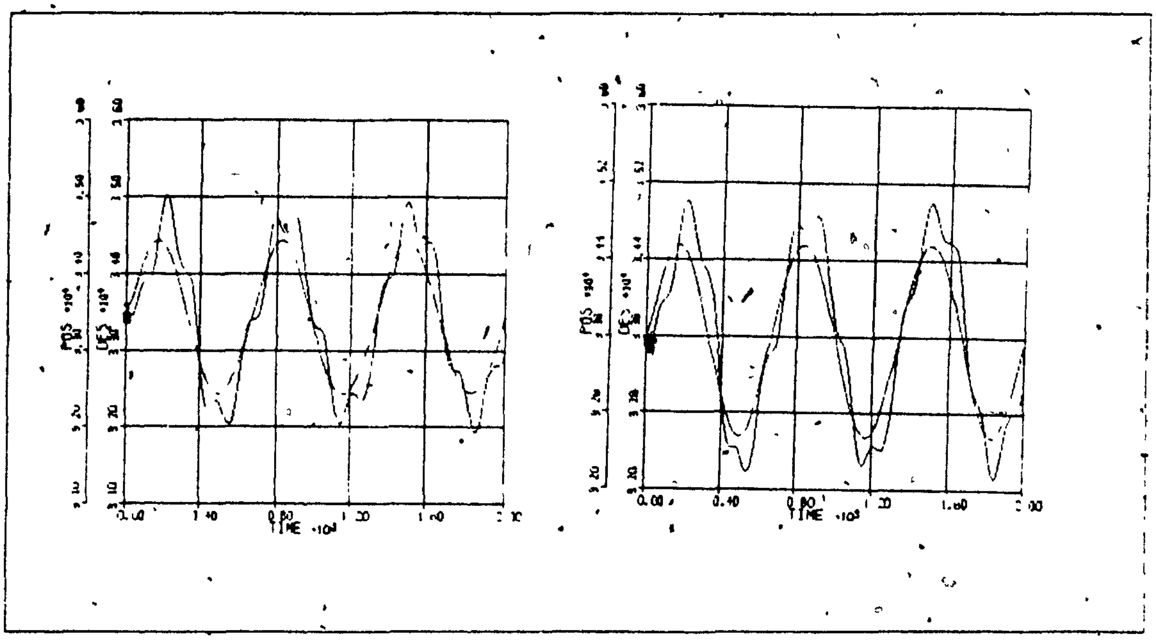

Figure 5.32b Sinusoidal Tracking PD (left), Acceleration Feedback (right)

Figure 5.32 Joint 2, High Frequency Disturbance Rejection

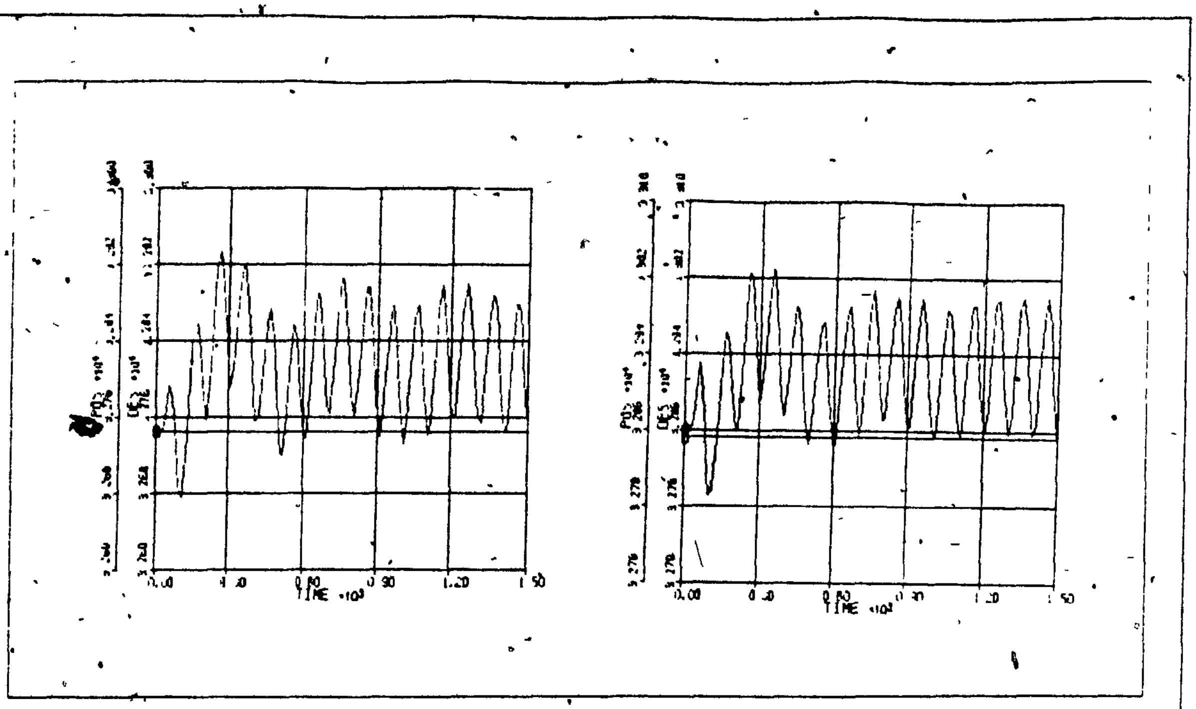

Figure 5.33a Servo PD (left), Acceleration Feedback (right)

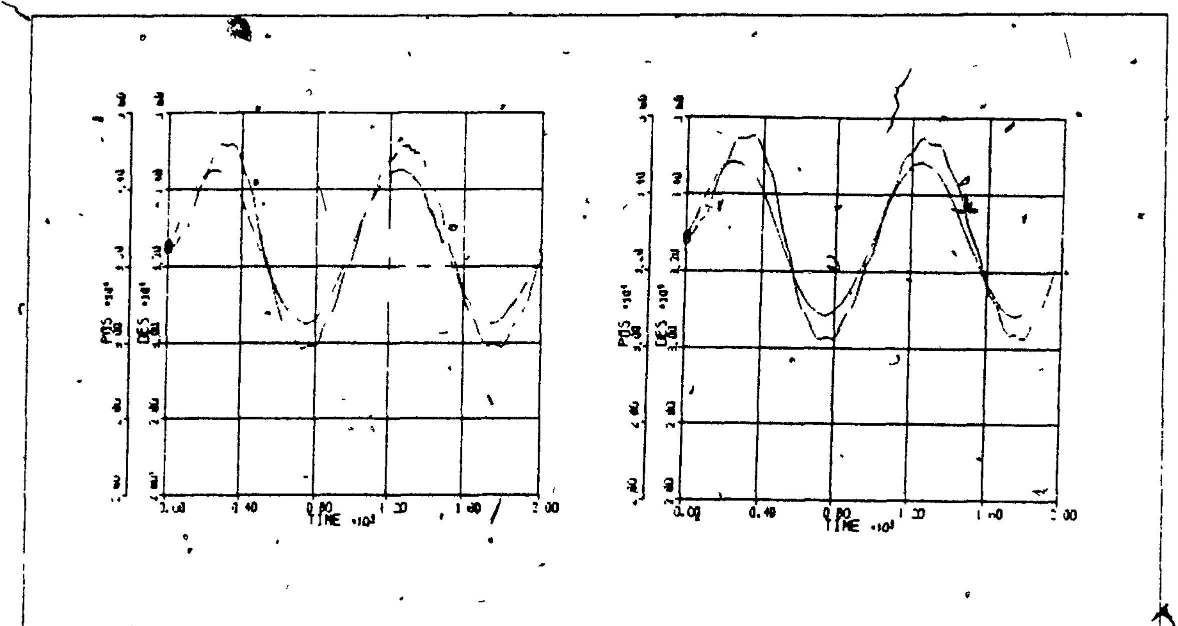

Figure 5.33b Sinusoidal Tracking PD (left), Acceleration Feedback (right)

Figure 5.33 Joint 2, High Frequency Large Inertial Disturbance Rejection

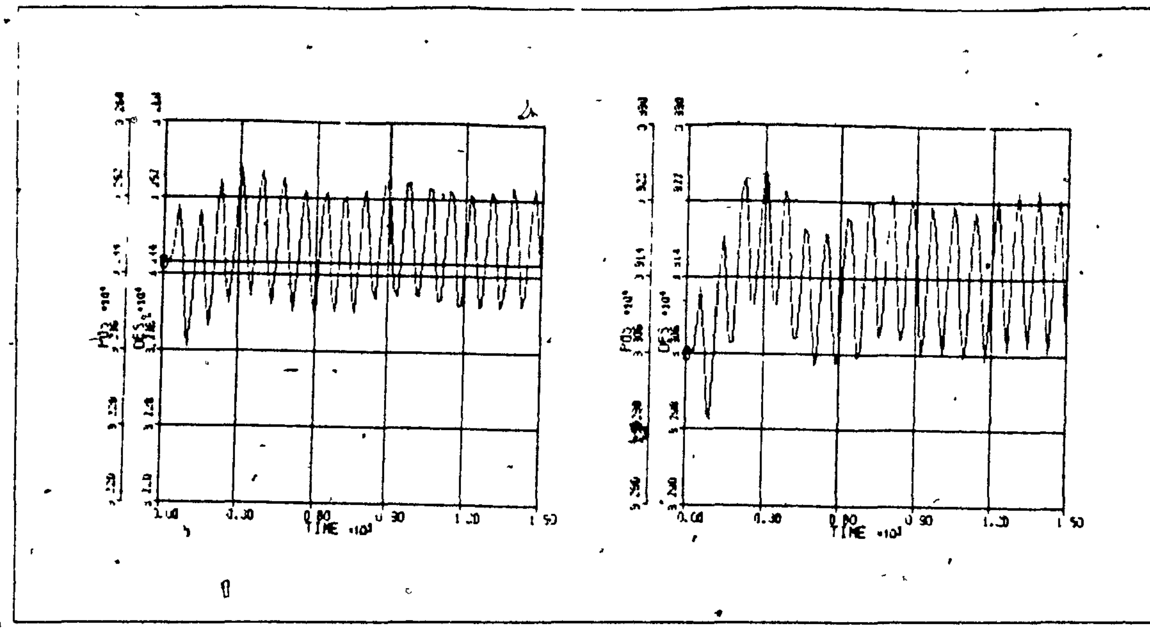

Figure 5.34a Servo PD (left). Acceleration Feedback (right)

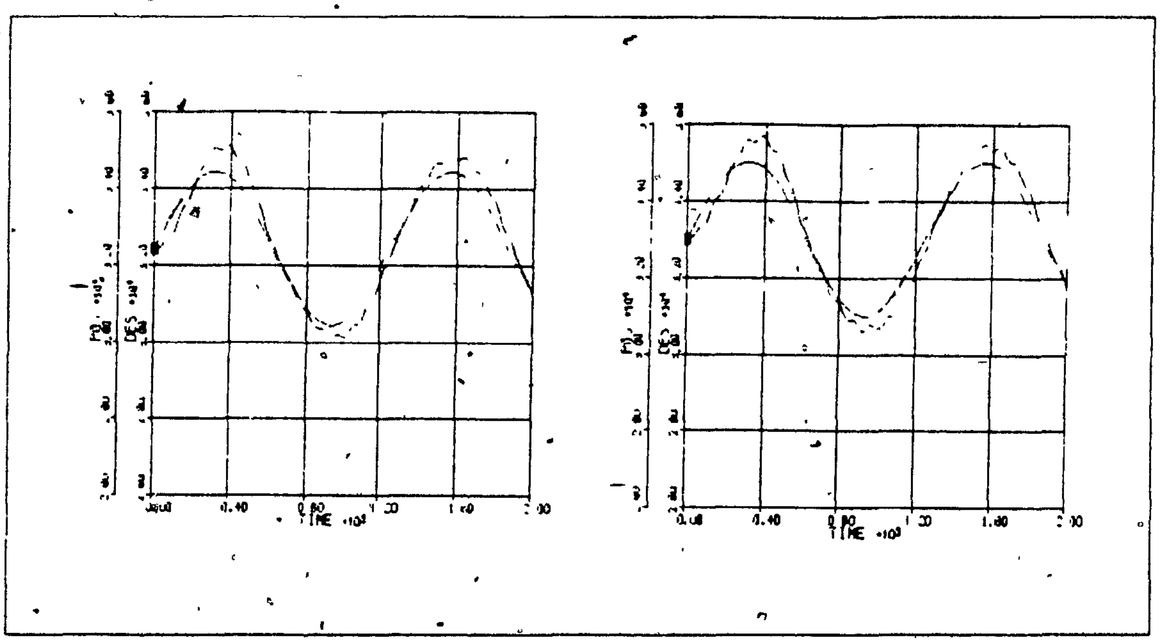

Figure 5.34b Sinusoidal Tracking. PD (left). Acceleration Feedback (right)

Figure 5.34 Joint 2. High Frequency Large Inertial Disturbance Rejection

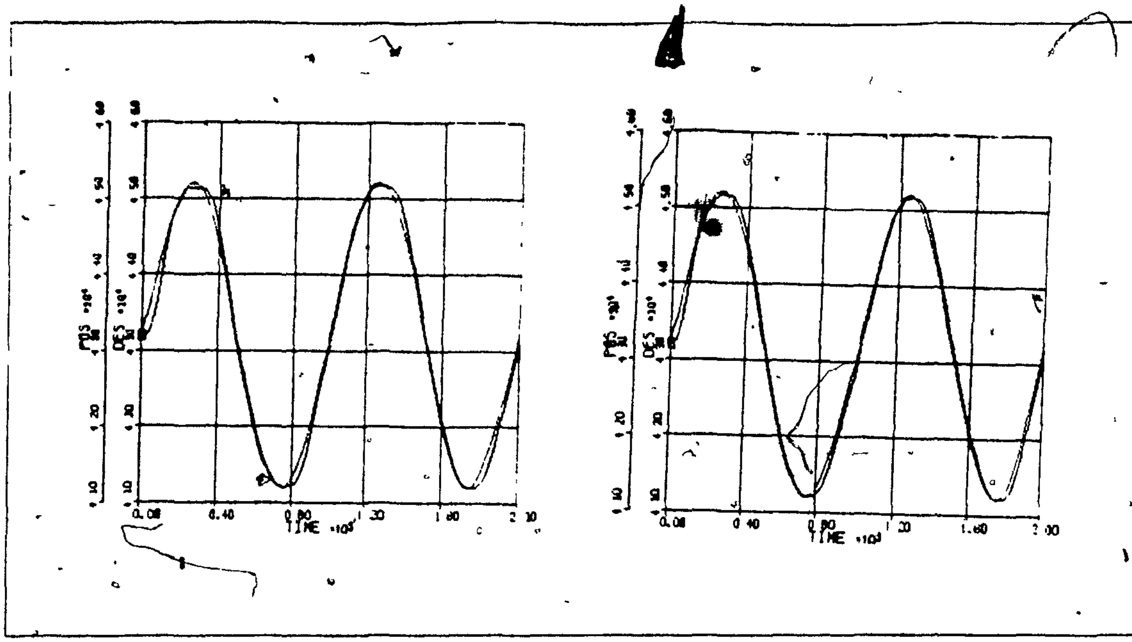

Figure 5.35a $K_p = 1$. $K_d = 2$ (left). $K_d = 3$ (right)

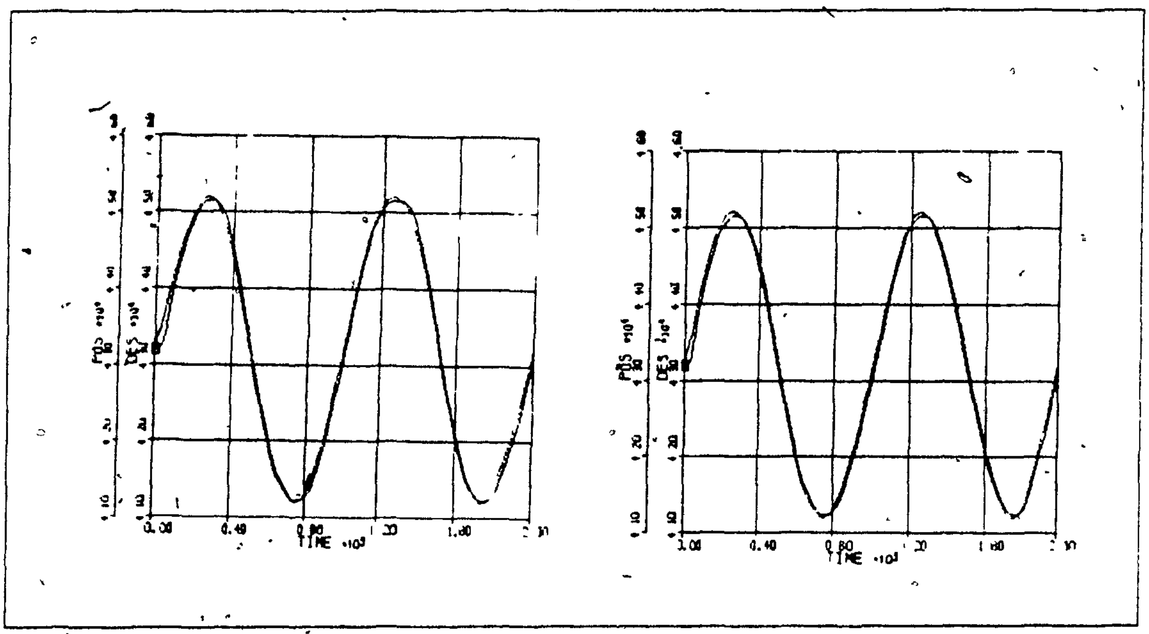

Figure 5.35b $K_p = 1$. $K_d = 4$ (left). $K_d = 6$ (right)

Figure 5.35 Joint 3. Variation in Damping using PD Control

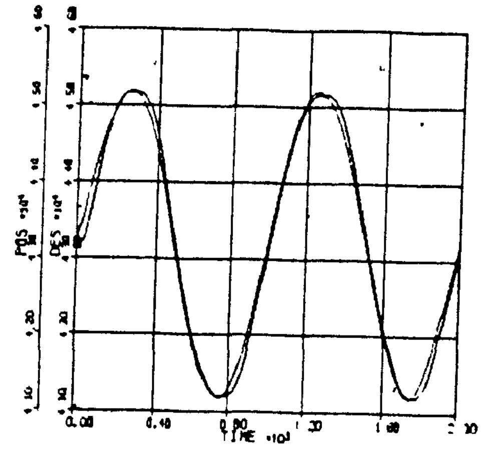

Figure 5.36a $K_u = 1,\ A_0 = 1,\ A_1 = 2$ (left), $A_1 = 3$ (right)

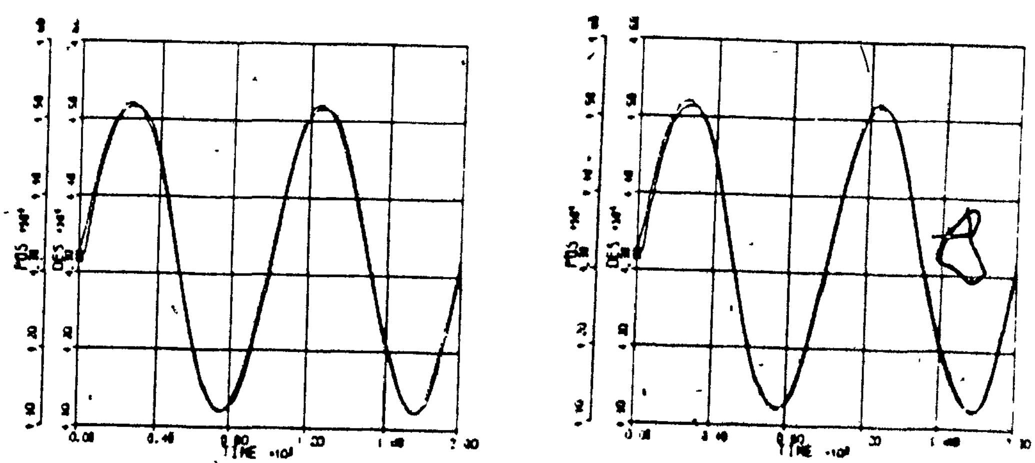

Figure 5.36b $K_u = 1,\ A_0 = 1,\ A_1 = 4$ (left), $A_1 = 6$ (right)

Figure 5.36 Joint 3, Variation in Damping using Acceleration Feedback Control

The third set of experiments examined the effect of varying the sampling interval of the system. The entire control cycle was run at sampling intervals of 28, 14, and 7 milliseconds, and the control gains were increased to the point of instability. It is expected that faster sampling intervals would allow higher control gains to be used. This experiment was carried out using acceleration feedback control only, and varying the high gain feedback K_u.

Following this experiment, a demonstration of a typical motion of three joints simultaneously was run using PD control and acceleration feedback control. Joint 1 and joint 2 were made to follow a ramp, while joint 3 tracked a sinusoid. This was not so much an experiment but a demonstration of the typical operation of the three joints of the manipulator under either control algorithm.

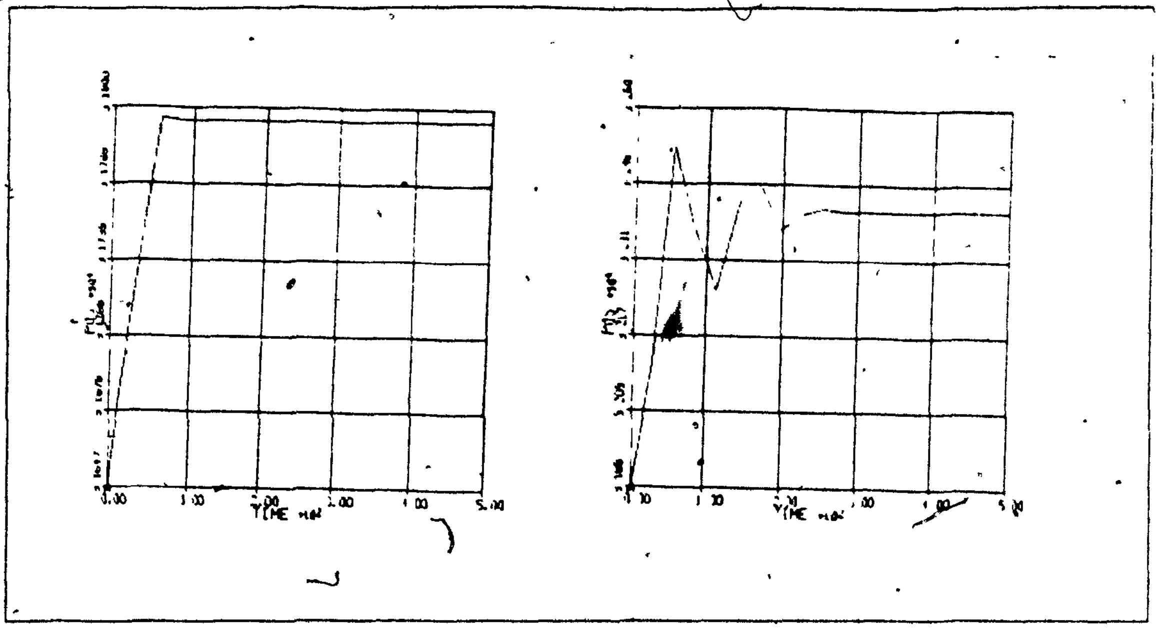

Figure 5.37a $K_u = 0.1$ (left), $K_u = 0.3$ (right)

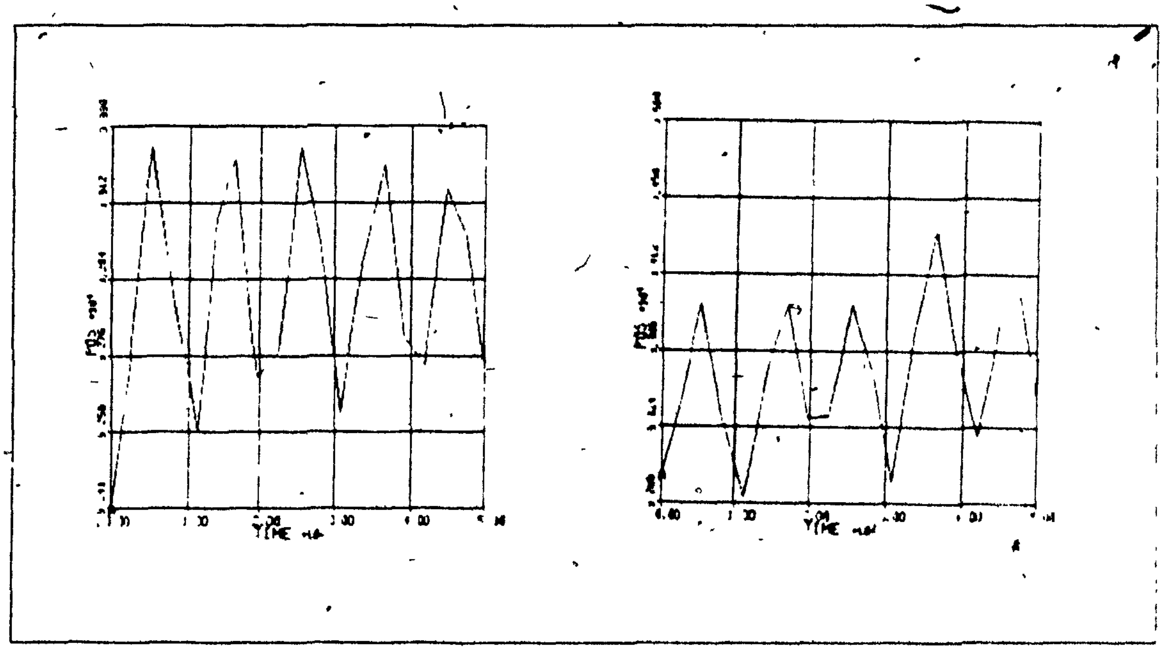

Figure 5.37b $K_u = 0.4$ (left), $K_u = 0.5$ (right)

Figure 5.37 Joint 6 Effects of Sampling Interval on Maximum Gain $T_s = 28$ msec

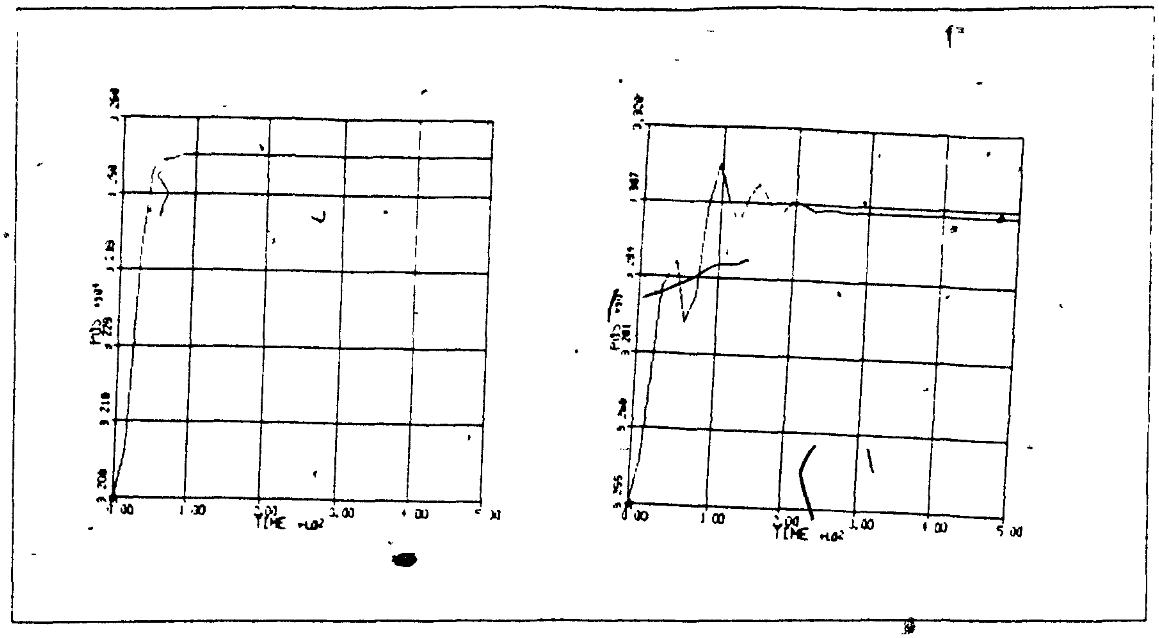

Figure 5.38a $K_u = 0.5$ (left). $K_u = 1.0$ (right)

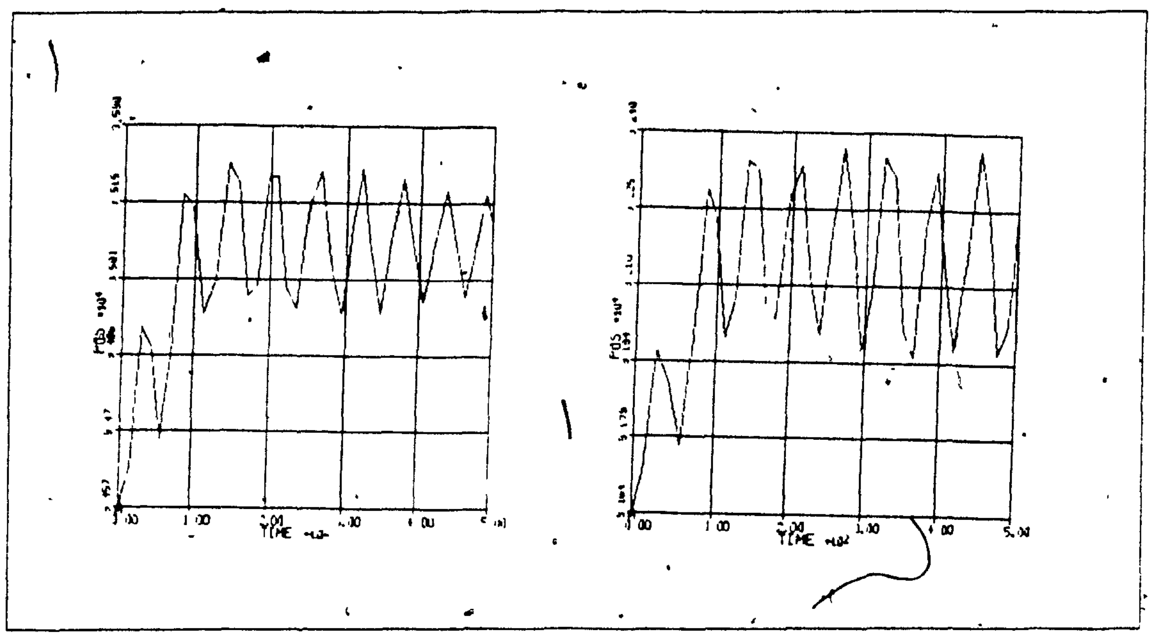

Figure 5.38b $K_u = 1\ 35$ (left). $K_u = 1\ 4$ (right)

Figure 5.38 Joint 6, Effects of Sampling Interval on Maximum Gain: $T_s = 14$ msec

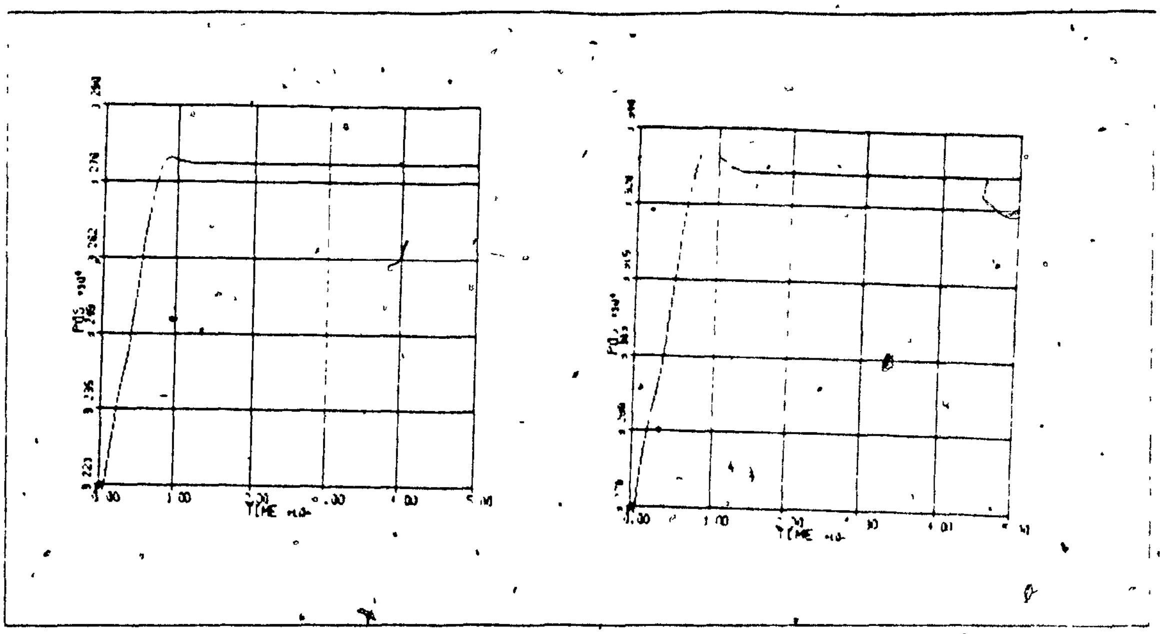

Figure 5.39a $K_u = 1.0$ (left) $K_u = 1.4$ (right)

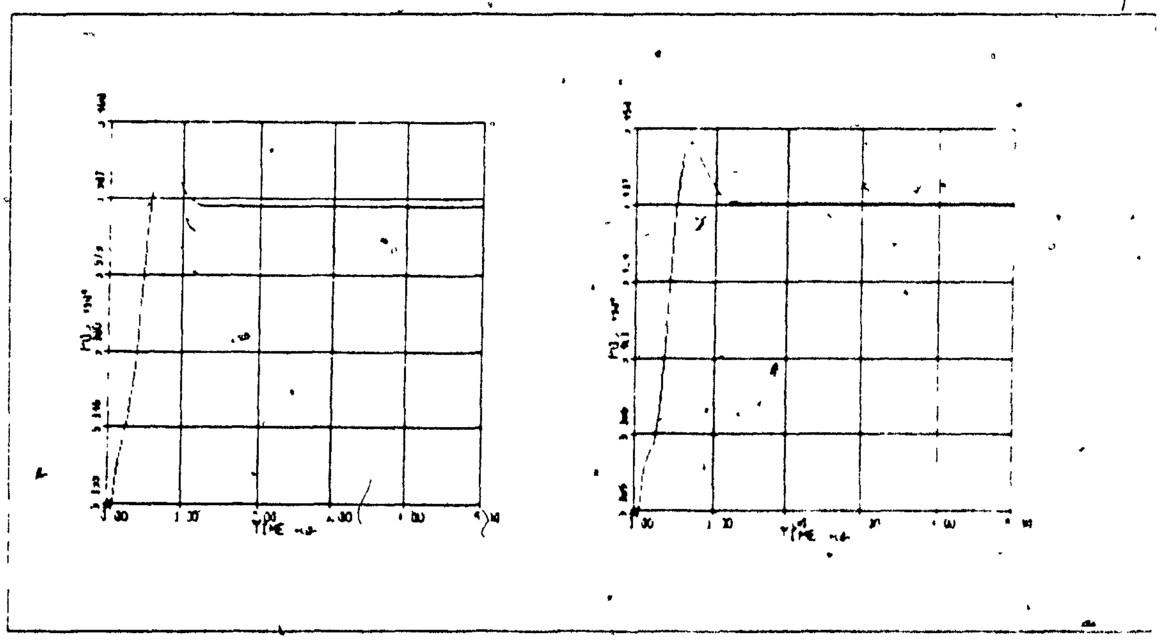

Figure 5.39b $K_u = 2.0$ (left) $K_u = 2.5$ (right)

Figure 5.39 Joint 6. Effects of Sampling Interval on Maximum Gain $T_s = 7$ msec

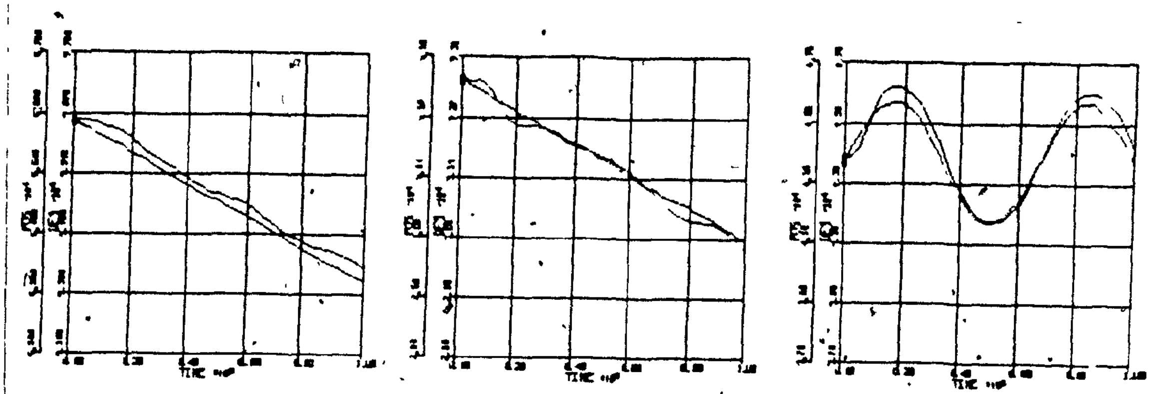

Figure 5.40a PD $K_p = 1.\ K_d = 2$

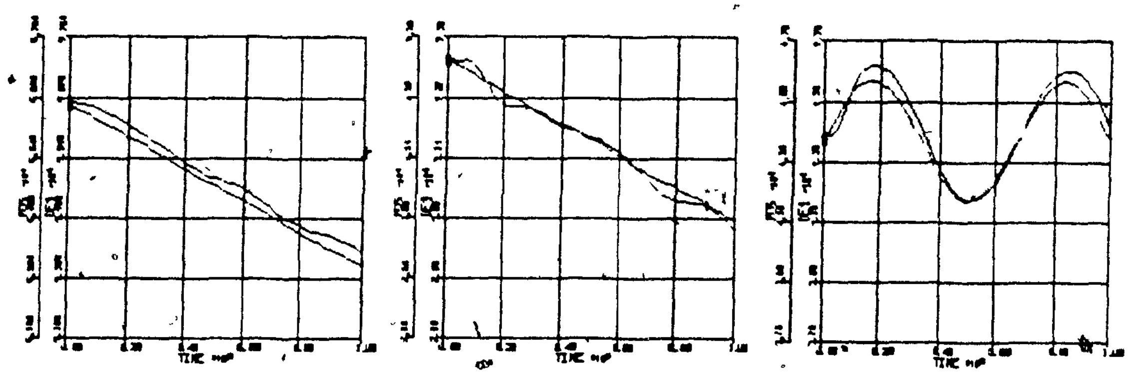

Figure 5.40b Acceleration Feedback $K_u = 1.\ A_0 = 1.\ A_1 = 2$

Figure 5.40 Typical Motion of Three Joints of the PUMA 260

5.5 Analysis

From the first section of the first set of experiments (figures 5.9-5.12) concerning the severity of the nonlinear effects, the following observations may be made. The inertia of a link controlled by a joint's actuator may cause significant degradation of performance, and varies greatly with the position and orientation of all of the other joints. This term in the dynamical equation is definitely not to be ignored, and the consequence of ignoring it will be a marked degeneration in performance. The effect of gravity is less severe for most joints, but also depends greatly on position and orientation. At least in the case of joint 3, the effect was not very significant, except in causing a steady state offset from the desired trajectory. The combination of gravity and inertia is, once again, significant and appears most prominently in joint 2 of the manipulator. Friction is very significant for some of the joints, and is actually beneficial in providing damping to an otherwise underdamped system. The drawbacks of this frictional effect include sticking of the joint controller due to static friction, and the variation of friction in an unpredictable manner. The effects of sticking due to friction appear vividly in figure 5.12b.

In theory, acceleration feedback is supposed to perform similarly to PD under conditions of low dynamical effects. The experiments in the second section of this set (figures 5.13-5.14) were designed to test this hypothesis. Indeed, both for step and sinusoidal responses, the performance of the two control algorithms was similar and quite satisfactory. The jerk at the beginning of motion in response to a step input, for the case of acceleration feedback, can be attributed to the time it takes for the second derivative operator to initialize correctly. A step input causes a discontinuity in position, velocity and acceleration and the second order differentiators' response reflects this situation. Since the primary use of acceleration feedback is in trajectory following, the behavior at the beginning of motion in response to as large step is not of great significance. It is worth noting, however, as it may impact disturbance rejection as well.

Under large dynamic disturbances, it is expected that acceleration feedback will outperform PD for trajectory following. The experiments in the third section of this set (figures 5.15-5.18) test this hypothesis. It becomes clear from the results of these experiments, that under the conditions of disturbance tested in this section, there was no appreciable difference in performance between acceleration feedback and PD. The results showed nearly identical performance for the two schemes, except in the case of frictional disturbance, which was high enough to make PD behave overdamped, while acceleration feedback still behaved in an underdamped manner. A discussion of the reasons for the similarity in performance between the two schemes under the conditions described above appears later in this section

Another aspect of the dynamics whose effect on tracking error must be assessed is dynamic coupling between simultaneously moving joints. The experiments described in figures 5.19-5.22 measure the effect of one joints motion on the precision of another. The first two figures describe a situation in which the motion of joint 3 acts as a disturbance to the motion of joint 2. The next two figures describe a situation in which the motion of joint 2 and joint 3 disturbs that of joint 1 It is possible to tell from these figures that joint 3 greatly disturbs joint 2, however joint 1 is relatively unaffected by the motion of other joints. The tabulated errors also demonstrate this fact numerically.

The next series of experiments (figures 5.23-5.24) assesses the relative performance of acceleration feedback and PD under conditions of severe cross coupling disturbance. In these experiments, acceleration feedback gave a slight but noticeable performance edge over PD, as is seen both in the figures and the error table

In summary, the following may be learned from this first set of experiments: Inertial, gravity, frictional, and cross coupling disturbances may all be significant, and cause severe degradation in error tracking on the joints of the PUMA 260. Acceleration feedback as implemented above does not cause serious improvement in performance over PD in most regimes of operation, except for a slight improvement in the face of cross coupling disturbances. The reasons for this lack of significant improvement may be explained in the following manner: Unless the high gain feedback K_u is very high in relation to the tuning

Fig	Exp	Mean Error	Deviation	Min	Max	Initial Offset
5 9	b1	183	98	-355	320	0
5 9	b2	411	127	-556	526	27
5 10	b1	105	46	-180	211	0
5 10	b2	230	98	-432	108	149
5 11	b1	195	106	-436	281	54
5 11	b2	540	157	140	893	568
5.12	b	132	45	-221	178	0
5 13	b1	182	96	-350	318	1
5.13	b2	185	103	-364	324	0
5.14	b1	106	45	-176	204	0
5.14	b2	106	46	-172	196	0
5 15	b1	407	126	-553	521	1
5.15	b2	411	130	-566	529	0
5 16	b1	261	106	-477	128	185
5 16	b2	255	104	-470	119	211
5.17	b1	539	157	115	879	549
5 17	b2	550	166	109	909	572
5 18	b1	135	45	-216	206	0
5 18	b2	135	43	-213	187	0
5.20	a1	287	139	-433	561	2
5.20	a2	288	136	-602	483	1
5 20	b1	471	239	-921	1104	47
5 20	b2	636	322	-376	1284	579
5 22	a1	185	100	-360	319	2
5 22	a2	224	118	-552	553	2
5 22	b1	397	122	-543	499	4
5 22	b2	393	122	-547	512	44
5.23	b1	434	219	-862	976	0
5 23	b2	394	203	-766	841	19
5.24	b1	426	144	-19	756	469
5 24	b2	422	125	64	728	483

Table 5.2 Error Table Set 1

parameters A_0 and A_1, the acceleration feedback control law reduces to essentially a PD structure, and the contribution of the acceleration term is not noticeable According to the results in [Studenny 87], the effect of K_u should be primarily that of decoupling and error reduction, and the stability and damping of the system should still be determined by the tuning parameters. This hypothesis prompted the next set of experiments, dealing with PD and acceleration feedback using different gains, and under conditions of high disturbances.

The first experiment in this set was intended to demonstrate that the performance of the acceleration feedback control algorithm is determined by the gains A_0 and A_1. In this experiment, the performance of PD with gains of $K_p = 0.5$ and $K_d = 1$ were compared with acceleration feedback whose control gains were $A_0 = 1$ and $A_1 = 2$, but whose high gain feedback was $K_u = 0.5$ Theoretically, if the performance of acceleration feedback is determined by the tuning gains only, it would be expected that the results for the two tests would be different. The experiment was repeated for PD with gains of $K_p = 2$ and $K_d = 4$, and acceleration feedback whose control gains were $A_0 = 1$ and $A_1 = 2$, with the high gain feedback $K_u = 2$. From the graphed results of these experiments (figures 5.25-5.26) and from the error tables, it seems that the results were once again, nearly identical At least in the configuration of our system, with the hardware environment in which we are working, and using the control algorithms coded and tuned in the manner detailed above, the performance of acceleration feedback is affected both by the tuning gains and by the high gain feedback.

The next group of experiments (figures 5.27-5.30) extended this analysis to joints impacted by large disturbances and joints moving at high frequency. Once again, the results demonstrated that the high gain feedback K_u has a much greater effect on system performance than was expected from the theory. In part, this is believed to be due to the relatively small magnitude of K_u and the small ratio between K_u and the tuning gains A_0 and A_1.

In the third experiment of this set, the effect of a high frequency disturbance on the positional accuracy of a stationary and moving joint were examined with PD and acceleration feedback control. Two types of conditions were tested. In the first (figures 5.31-5.32), joint 3 was made to move at a frequency of 7 Hz to disturb joint 2, which was either stationary or following a sinusoid. In this experiment, acceleration feedback provided an improvement of 30 percent in tracking error over PD for joint 2 when stationary, and

10-15 percent when tracking a sinusoid. The second set of conditions were similar (figures 5.33-5.34). however this time the disturbance was at a higher frequency of 12 Hz. and a 1 kilogram weight was attached to the end effector to increase the magnitude of the disturbance. In this case, the performance of the two control algorithms was comparable

The last experiment in this set was meant to demonstrate the effect of increasing the damping terms A_1 and K_d In theory, the inclusion of an acceleration term in the acceleration feedback control law should allow us to increase the magnitude of the damping term, therefore increasing the accuracy of the positional tracking of acceleration feedback beyond that allowed by PD. From figures 5.35-5 36, and from the error table, it is clear that this is not the case. The highest allowable damping was identical for both control algorithms under similar conditions, and the performance was also similar

Thus, in summary, the performance of acceleration feedback was again similar to that of PD, as was found in the the first set of experiments. It is also important to note that the one type of condition under which acceleration feedback did show a significant improvement was in the case of a joint disturbed by a large inertial coupling disturbance, also as found in the first set of experiments. It seems the the problem may lie in the magnitude of K_u and the relatively small ratio between K_u and the tuning gains A_0 and A_1.

One reason that the gains of the system had to be so limited in magnitude is related to the sampling interval of the RCI system It is expected that the allowable gains would increase in magnitude if the sampling frequency were increased. Using a borrowed LSI-11/23, it was possible to test this theory by running the system at sampling intervals of 28, 14, and 7 milliseconds, and raising the gain to the point of instability From figures 5.37-5 39, it is clear that the stability of the system is far more robust at a faster sampling frequency, and that the allowable gains are much higher. The highest allowable gain K_u was 0.4 at a sampling interval of 28 milliseconds, 1.35 at 14 milliseconds, and 4.0 at 7 milliseconds. Higher gains would allow the system to track trajectories with far greater accuracy, and to test the acceleration feedback control algorithm with a much larger ratio

Fig	Exp	Mean Error	Deviation	Min	Max	Initial Offset
5.26	a1	491	168	-709	639	0
5.26	a2	101	57	-197	173	0
5.26	b1	494	170	-712	648	0
5.26	b2	102	56	-195	175	0
5.28	a1	342	186	-760	443	0
5.28	a2	94	47	-208	196	0
5.28	b1	360	146	-461	681	0
5.28	b2	94	48	140	224	0
5.29	a1	130	58	-221	164	0
5.29	a2	94	40	-163	119	11
5.29	b1	122	58	-230	170	0
5.29	b2	95	41	-185	148	0
5.30	a1	176	86	-153	351	3
5.30	a2	176	85	-151	348	3
5.30	b1	177	85	-186	343	3
5.30	b2	178	86	-189	344	3
5.31	a1	274	104	-434	370	0
5.31	a2	170	64	-272	235	51
5.31	b1	384	201	-726	757	0
5.31	b2	332	168	-651	631	0
5.32	a1	124	49	-213	149	0
5.32	a2	83	33	-144	94	0
5.32	b1	326	165	-660	618	0
5.32	b2	306	140	-615	544	0
5.33	a1	93	49	-157	11	0
5.33	a2	95	50	-159	4	12
5.33	b1	500	181	-844	750	0
5.33	b2	502	181	-844	700	85
5.34	a1	53	23	-99	57	2
5.34	a2	102	53	-172	6	4
5.34	b1	349	123	-597	519	12
5.34	b2	354	124	-582	561	33

Table 5.3 Error Table Set 2

of gains.

The final figure (5.40) demonstrates a typical motion of the first three joints of the PUMA 260. The performance of the two control algorithms is nearly identical under

Fig	Exp	Mean Error	Deviation	Min	Max	Initial Offset
5.35	a1	123	50	-203	227	0
5.35	a2	115	43	-166	183	0
5.35	b1	110	38	-153	164	0
5.35	b2	102	31	-132	160	0
5.36	a1	119	49	-201	229	0
5.36	a2	112	41	-162	177	0
5.36	b1	107	36	-145	171	0
5.36	b2	100	30	-130	161	0

Table 5.4 Error Table: Set 3

these conditions. Thus, it can be shown that under typical conditions, acceleration feedback as implemented in this research is stable and robust, but does not give a great improvement over PD, in terms of tracking error and response time.

Chapter 6 Conclusion

6.1 The Environment

The hardware environment used in this research consisted of a PUMA 260 manipulator controlled by a VAX computer through an LSI-11 controller. The software environment consisted of the RCI system running under UNIX 4.3 with NFS.

6.1.1 Suitability to the Task

The PUMA 260 manipulator presents both advantages and disadvantages to the study of control algorithms. Its small size is an advantage in that control routines may be taken to the limit of their stability without fear of damage to people or equipment Care must be taken to prevent the PUMA from causing damage to itself, but it is so small and weak that it may be physically prevented from doing so by holding it manually The disadvantages of this manipulator also stem from its weak motors, as well as from the high and variable friction on several of its joints. The weakness of the motors severely restricts the magnitude of velocities, accelerations and torques which the manipulator can accept. Friction, and especially the extremely high static friction on the wrist joints require a fairly high current to even move a joint from a stationary position. At times, the current required to get a joint to move was around twenty percent of the maximum allowable current. This combination of friction and weak motors greatly restricted the operating range of the controllers. Another side effect of the small size of the PUMA was that it became difficult to introduce high cross coupling inertial terms in a known manner. For instance, the motion of joint 3 did not overly disturb joint 2. This deficiency was partly overcome by moving joint 3 at a high frequency, and also by attaching a 1 kilogram weight to the end effector, thus increasing the inertia terms. The variability of the friction terms also made the manipulator dynamics difficult to model and unreliable, thus making tuning more difficult and performance hard to predict. The advantage brought about by high friction was that the effects of high frequency uncertainties and disturbances were completely damped out, and allowed operation of the control schemes without use of a low pass filter on the error.

The VAX-11/750 under UNIX 4.3 and NFS provided at best adequate performance for the control task. Being a time sharing system, it was very sensitive to system load and network activity. The frequent occurrence of timeouts during the control cycle proved to be a major obstacle during the course of the research. The LSI-11/03 also provided slow response, and much of the control cycle was taken up by communication time between the LSI-11 and the VAX, or the LSI-11 and the PUMA. It is estimated that this communication alone took about seven milliseconds, which added to the four milliseconds of control computations brought the cycle time to eleven milliseconds. Since the control cycles of RCI operate in factors of seven milliseconds, the minimum sampling interval that could be used with this equipment was fourteen milliseconds, or 72 Hertz. At this sampling rate, the operating bandwidth is limited to about four Hertz, and control gains are severely limited in magnitude. This turned out to be a problem of great significance, as it was difficult to produce strong inertial effects with small allowed bandwidth and small gains, while at the same time the small gains caused system performance to be sluggish, leaving relatively high position errors. In summary, it was impossible to produce a very stiff controller at this sampling rate, and difficult to create very disturbed conditions with which to test it. Late in the research, the LSI-11/03 was replaced with a faster LSI-11/23 whose communication time was about two milliseconds. This brought the entire control cycle to six milliseconds, and allowed operation at a sampling interval of seven milliseconds, or about 142 Hertz. At this sampling interval, the testbed could operate only under conditions of low system load and no network activity, but the tests that were carried out showed a three-fold increase in allowable gain and much better error tracking.

The RCI system proved to be a very flexible and convenient platform with which to perform control algorithm research. Splitting the program into a planning and a control section allowed the creation of a very user-friendly research environment without impact on the timing constraints of the control sections. Interaction of the RCI system with the manipulator through the *how* and *chg* structures directly allowed very precise knowledge of system parameters and clear notions of the consequences of commands. The fact that all of the programming is done using C functions allows a new user to learn the RCI system ~~relatively~~ quickly, and make modification of the system or the use of UNIX system commands very easy The only drawback of the RCI system encountered in this research is in the communications overhead required to transfer information between the VAX and the LSI-11.

As it stands, the system on which the testbed was configured provided the minimum performance allowable to produce viable results for this research. The allowed sampling frequency of 72 Hertz is in the range specified in [Studenny 87] as the minimum frequency at which the system is expected to stay stable. The gains that could be used at this sample frequency provided a stiff enough system that error measurements could be conducted and compared with similar tests. The biggest restriction was in operating range and bandwidth. Though the testing that was conducted included motions which produced significant disturbances, producing these motions required operation of the manipulator at the edge of its stability and operating capabilities

6.1.2 Suggestions for Improvement

One suggestion for improvement of the operating environment involves doing the research on a manipulator with a greater operating range between the minimum to overcome friction and the maximum allowable current, velocity, and acceleration. In particular, raising the ceiling on current, velocity and acceleration would allow testing of regimes at which control algorithms really start to break down. One possible solution would be the use of a specially designed manipulator which lends itself to control experimentation. Specifically, this manipulator need only have two or three degrees of freedom to allow a full investigation of inertial and cross-coupling effects, and should be constructed in such a manner as to allow variation in coupling, friction, and inertial effects. To be effective as a testbed for manipulator control research, this manipulator must be able to emulate the charactersitics of widely available commercial manipulators, however it would not be intended for industrial applications itself.

As for the computing environment, it would be advisable to perform the control computations on a faster computer, such as a MicroVAX, which would operate in a stand-alone mode and would not support network activity. It was found that network activity was the most major restriction in terms of computing time. The LSI-11/03 must be replaced by a faster processor, such as the LSI-11/73, which would allow the communications processing at less than one millisecond. It would be advisable to configure a system whose sampling frequency is as close to one KiloHertz as possible. It has been demonstrated that controllers operating at that frequency can provide very stiff control, such as the original controllers of the PUMA 260 manipulator.

As for the RCI system, the only suggestions for improvement would be to allow sampling intervals which are not multiples of seven, thus allowing the user to optimize the sampling interval for the amount of computations required. The seven millisecond interval is a limitation of the Unimation controller, however, and can not be easily changed. It may be useful to give the user some control over the information that is to be passed between the VAX and the LSI-11, also allowing optimization according to the application.

As mentioned above, the current system did provide performance that met the minimum requirements to be useful, however expanding its range of performance would allow testing at more challenging regimes and more definitive verification of the performance of controllers under very disturbed, high velocity and acceleration motions.

6.2 The Testbed

The testbed is a general-purpose utility which allows the testing of control algorithms on the PUMA 260 manipulator. It allows individual controllers with separate parameters for each joint, and enables the user to configure a number of test motions on the various joints and collect the data for later analysis. A number of analysis and graphics routines join the testbed to allow visual and numerical interpretation of the test data.

6.2.1 Suitability to the Task

The most notable attribute of this testbed is its flexibility and ease of configuration. A new control algorithm with the corresponding user interface takes about two hours to program For example, a sliding mode controller was recently implemented on this testbed, and the time of implementation from equations on paper to the first controller experiment was under two hours. The variety of test patterns which can be performed by the joints and the flexibility of choosing any controller with any parameters for each joint separately render this testbed a useful utility for a large number of control research projects. The alternative approach of downloading control routines directly into the joint microprocessors does not provide the same easy access to data and parameters, and removes the user from the direct control of the manipulator's configuration. Furthermore, a microprocessor implementation does not offer the computing power of the minicomputer used in this research. Using the full range of the PUMA 260's velocity, acceleration, and torque, and with the addition of a weight at the end effector, this testbed allowed comprehensive testing of the control algorithms, albeit with some restrictions. Those restrictions were due primarily to the environment, however, and not to the testbed.

6.2.2 Suggestions for Improvement

The major improvement in the design of the testbed involves increasing the sampling frequency at which it could run. Though the timing constraints are primarily due to the environment, the testbed could be somewhat optimized by eliminating unused functions. Otherwise, further improvements in the testbed may include increasing its flexibility and ease of operation. Examples of this include storing test patterns in data files, much as system parameters are currently stored This would allow the user to call up preconfigured test routines without retyping the desired tests Additional test patterns may also be added. In the realm of analysis routines, the major improvement could come in the area of a convenient graphing routine which would be attached to the testbed The current configuration is rather cumbersome and causes a delay between the time at which a test is performed and the time the data is plotted Otherwise, the testbed works well as it stands, and no major improvements are seen to be necessary

6.3 Test Results

This section relies heavily on the material presented at the end of chapter four A detailed analysis of the individual experiments has already been undertaken in that chapter, and this section is meant to summarize the results and draw conclusions from them

6.3.1 Validity of Test Results

Before any conclusions may be drawn regarding the performance of the control algorithms, the results themselves must be examined to ensure that they accurately represent the performance of the control schemes Two types of trajectories were used in these experiments: step inputs, and sinusoids The use of step inputs gives an indication of the time response of the system and of its damping It is a reasonable approximation of a large disturbance impinging on the joint of the manipulator. The sinusoids give nice statistical results on tracking errors and frequency domain performance Using these two types of input, it was possible to get a complete picture of the system's performance under most operating conditions.

Some of the experiments in the first set of results were meant to examine the magnitude of the dynamic effects. This was done to define the regimes in which to carry out comparative tests of the two control algorithms, and to determine if the effects are severe enough to warrant special attention. The results showed that some of the effects are severe, and outlined the regimes of operation in which they occur.

The comparative experiments between the two control algorithms were numerous and explored many conditions and configurations. Undisturbed and disturbed conditions were tested for single and multiple joints. Coupling disturbances were examined with disturbances of varying magnitudes and frequencies; joint motion was studied for various magnitudes and frequencies of the tested joint itself. Finally, the gains of both algorithms were varied to ensure that the conclusions about the algorithms' performance are not limited to one configuration. Thus, the tests conducted are believed to be sufficient to generate valid conclusions concerning the relative performance of the two algorithms.

If there is any weakness in the results, it is in the limitations on operating gains and frequencies imposed by the hardware environment. It could be argued that the characteristics of the acceleration feedback theory can not be fully explored at the sampling frequency used in this research, however since both algorithms are tested under exactly the same conditions, and since the environment does allow a stable implementation of acceleration feedback as characterized in [Studenny 87], it is believed that the results presented in this section are a valid description of the performance of the algorithm, and may be extended to other implementations as well.

6.3.2 Performance of Acceleration Feedback VS PD

Acceleration feedback control, as implemented above, provided stable control for the PUMA 260 joints. The error tracking and response time were not, however, very significantly better than PD coded in the same manner and tested under the same conditions. At first, it was thought that the similarity in performance of the two algorithms was due to the low gains and bandwidths allowable under the operating environment. however further tests failed to confirm this hypothesis. Initially, it was observed that the two algorithms perform similarly for one joint, for both inertially loaded and inertially unloaded conditions The only conditions under which acceleration feedback consistently performed better than PD was in rejecting coupling disturbances from other joints Recalling the manipulator dynamics equation,

$$J(q)\ddot{q} + QC(q)\dot{q}^2 + G(q) = u \tag{6.1}$$

one can notice that it is the coupling terms which are most directly affected by acceleration, and that is the area in which acceleration feedback should give the greatest improvement The similarity of performance of the two algorithms in the absence of coupling disturbances and the superiority of acceleration feedback in the presence of these effects was demonstrated repeatedly and under a variety of types of motion and disturbance Joint frequency disturbance frequency, disturbance amplitude, controller gains, and damping were all varied in the course of the experiments, and the relative performance of the two algorithms remained as described above The experimental results presented in chapter five correspond nicely to the simulation experiments performed in chapter three In the simulations by Studenny [Studenny 87], better tracking performance was achieved, however these simulations were carried out using the larger PUMA 600 as a model and at a faster sampling interval than available on our system Thus, both the gains and the torque limits were higher in these simulations, accounting for the improved performance

There are several disadvantages of this implementation of the acceleration feedback algorithm which should be pointed out at this time First, the use of numerical differentiators to derive acceleration is inherently much noisier than the differentiator used to derive velocity The erratic value of acceleration produce by this method can lead to instability and large torque values, especially in the face of impulse disturbances Better performance may be obtained for manipulators equipped with tachometers, since the first difference operator is much less noisy The requirement that the algorithm be implemented with a low pass filter to smooth out the high frequency uncertainties is likewise a disadvantage of the acceleration feedback theory, increasing the phase lag of the closed loop system

Unless the sampling frequency is increased to a point in which the cutoff frequency of the low pass filter can be set very high, the filter may interfere with the controllers, leading to instability. It was found that the filter was not necessary for stable operation of the controller, presumably due to the high friction of the joints which damped out such effects. Only when the high gain feedback K_u was made so high so that the system was driven almost entirely by acceleration was the low pass filter effective in preventing instability, however the performance of the system was not very different in this configuration than in those driven by a combination of position, velocity and acceleration Finally, a recurring theme in this work is the limitations on the system imposed by the operating environment. In assessing the implementation of the algorithm it would be unwise to ignore the effect of the sampling frequency on gains and bandwidth. It may still be that under different circumstances allowing higher controller gains and higher frequency test conditions, the advantages of acceleration would become more apparent. Still, the conditions under which the algorithm was tested are within the nominal conditions under which acceleration feedback theory should operate, and are believed to be representative of the overall performance ot the algorithm

6.4 Final Comments

The idea of using acceleration feedback to create a system which is robust with respect to the nonlinear disturbances present in manipulator dynamics is a very useful approach for manipulator position control Decoupling the system without resorting to a complex model of the dynamics is of extreme value for a system whose dynamics are as involved and varying as a manipulator joint. The acceleration feedback algorithm implemented here provides stable control of the manipulator joints without an excessive computational burden. It does not provide extreme improvements over PD control under the conditions tested in this research, except for a slightly improved robustness with respect to coupling disturbances.

Since a manipulator must interact with other objects in its environment to be of any use, it seems that position-only control algorithms are inadequate for providing a framework to control a manipulator at times in which it is making contact with another object. Thus, the idea of using controllers which do not include force dependent terms is in itself questionable for robot manipulators performing typical industrial tasks such as assembly. Acceleration feedback control provides stable position control with reasonable error tracking, however, since it does not allow for force control as well, it would have to be coupled with a force controller to give a hybrid control system capable of handling the entire range of tasks effectively.

Appendix A. Simulation Parameters

The simulation consisted of a two link manipulator, where the links were modeled as point masses, as in [Craig 85]. The simulation model was as follow, in ACSL format:

```
e1 = th1 - q1                              ( "error = desired - actual position"

e2 = th2 - q2

tom1 = ki*(tau1 - kb*q1d/n)                "motor torque calculation"

tom2 = ki*(tau2 - kb*q2d/n)

B = ja/(n*n)                               "inertial terms calculation"

k22 = l2*l2*m2

XX = k22 + B

k11 = l2*l2*m2 + 2*l1*l2*m2*cos(q2) + l1*(m1+m2)

YY = l2*l2*m2 + l1*l2*m2*cos(q2)

Z = k11 + B

v1 = (-1)*m2*l1*l2*sin(q2)*q2d*q2d - 2*m2*l1*l2*sin(q2)*q1d*q2d

g1 = m2*l2*g*cos(q1+q2) + (m1+m2)*l1*g*cos(q1)

v2 = m2*l2*l1*sin(q2)*q1d*q1d

g2 = m2*l2*g*cos(q1+q2)

detm = l1*l1*l2*l2*m2*(m1+m2-m2*cos(q2)*cos(q2))+B*(k11+k22)+B*B

u1 = tom1/n - ba*q1d/(n*n) - v1 - g1

u2 = tom2/n - ba*q2d/(n*n) - v2 - g2

q1dd = ( u1*XX - u2*YY )/detm              "acceleration calculation"

q2dd = ( u2*Z - u1*YY )/detm

time = integ( 1.0, 0.0 )
```

```
q1d = integ( q1dd, q1ddz )

q2d = integ( q2dd, q2ddz )

q1 = integ( q1d, q1dz )

q2 = integ( q2d, q2dz )
```

The controllers were coded in the following form. Note the numerical differentiators.

```
interval tsamp = 0 014

q1e = e1                                               "error"

q2e = e2

v1e = (3 * e1 - 4 * e1o + e1oo) / (2 * tsamp)          "velocity error"

v2e = (3 * e2 - 4 * e2o + e2oo) / (2 * tsamp)

a1e = (e1 - 2 * e1o + e1oo) / (tsamp * tsamp)          "acceleration error"

a2e = (e2 - 2 * e2o + e2oo) / (tsamp * tsamp)

t1p = kp1 * q1e + kd1 * v1e                            "pd controller"

t2p = kp2 * q2e + kd2 * v2e

t1a = ku1 * (ap1 * q1e + ad1 * v1e + a1e)              "af controller"

t2a = ku2 * (ap2 * q2e + ad2 * v2e + a2e)

tau1 = rsw (afc, t1a, t1p)                             "controller switch"

tau2 = rsw (afc, t2a, t2p)

e1oo = e1o

e2oo = e2o

e1o = e1

e2o = e2
```

Finally, the parameters were as follows:

```
set pi = 3.1415
```

set ki = 0.75 "torque sensitivity"

set n = 0.1 "gear ration"

set l1 = 0.2, l2 = 0.2 "link lengths"

set m1 = 4.0, m2 = 400.0 "link masses"

set g = 0.0 "link masses"

set ja = 10.0, ba = 0 0 "actuator inertia and friction"

set kd1 = 2000.0 "joint 1 derivative gain"

set kd2 = 2000.0 "joint 2 derivative gain"

set kp1 = 1000.0 "joint 1 proportional gain"

set kp2 = 1000.0 "joint 2 proportional gain"

set afc = .true. "Acceleration Feedback on"

set stp = .false. "Step input off"

set ku1 = 100.0 "joint 1 afc gain"

set ku2 = 100.0 "joint 2 afc gain"

set ad1 = 20.0 "joint 1 derivative gain"

set ad2 = 20.0 "joint 2 derivative gain"

set ap1 = 10.0 "joint 1 proportional gain"

set ap2 = 10.0 "joint 2 proportional gain"

The setpoints listed above represent initial conditions upon startup of the simulation. Some of these factors varied during actual testing. For the tests listed in figure 3.2, the mass of joint 2 was set to zero, and it was held still. This gave a single joint test for the algorithms For the tests in figure 3 3., the mass and amplitude of oscillation of joint 2 were gradually increased to provide a growing cross coupling disturbance. They were given as ($m_2 = 0$ $s_2 = 0$) ($m_2 = 1$ $s_2 = 1$) ($m_2 = 400$ $s_2 = 5$) and ($m_2 = 400$ $s_2 = 15$), respectively. The last set of parameters is the one which was used in figure 3.4 to compare the control schemes The final experments (figure 3.5) used an acceleration feedback controller. The control gains used throughout were the default gains, as listed above

Appendix B. Experimental Conditions

This appendix presents details of the experimental conditions under which the various experiments in chapter five were carried out. Relevant information includes the choice of control algorithm, control gains, arm position, and a description of the type of motion used during the experiment.

In the section on system identification, the preliminary test (figure 5 2) involved placing a value on the DAC and measuring the position as the joint reacted to this current setpoint. Twelve tests were performed on joint 6 of the PUMA, with DAC values ranging from 180 to 400. In the particular experiment presented in figure 5 2, a DAC value of 300 was used. The experiments involving the step response of joint 6, a proportional controller with $K_p = 2$ was used. Input to the joint was a step of 100 encoder counts, and twelve tests were performed.

For the frequency response tests, the input to the system was a series of sinusoids ranging in frequency from 0.05 Hz to 6 Hz, and in amplitude from 200 encoder counts to 1000 encoder counts A proportional controller was used with $K_p = 0.3$ for joint 6 and $K_p = 0.2$ for joint 3. Least squares parameter estimation was carried out for joints 3 and 6 as well. The input to this algorithm was a high frequency (4 Hz) square wave of amplitude 300 encoder counts, and a proportional controller with $K_p = 1$ was used in the experiment.

The Ziegler-Nichols tuning method requires the use of a proportional controller, and involves the proportional gain to be raised until steady oscillation occurs. At each new gain setting, the joint was "kicked" using a 200 encoder count step input. For the manual tuning experiments, the proportional gain K_p was set to unity, and the derivative gain K_d ranged from 0 to 10.

The experimental conditions for the controller experiments are listed in the tables below Explanation of the entries in the tables, as well as additional information

Fig	Joint	Ctrl	K_p	K_d	K_u	A_0	A_1	Config
5.9	1	PD	1	2				up/out
5.10	3	PD	1	2				down/out
5.11	2	PD	1	2				up/out
5.12	6	PD	1	2				ready
5.13	1	PD/AF	1	2	1	1	2	up
5.14	3	PD/AF	1	2	1	1	2	down
5.15	1	PD/AF	1	2	1	1	2	out
5.16	3	PD/AF	1	2	1	1	2	out
5.17	2	PD/AF	1	2	1	1	2	out
5.18	6	PD/AF	1	2	1	1	2	ready
5.19	2;3	PD	1	2				up/out
5.20	2;3	PD	1	2				up/out
5.21	1,2,3	PD	1	2				up/out
5.22	1,2,3	PD	1	2				up/out
5.23	2;3	PD/AF	1	2	1	1	2	up
5.24	2;3	PD/AF	1	2	1	1	2	out
5.25	1	PD/AF	vary	vary	vary	vary	vary	out
5.26	1	PD/AF	vary	vary	vary	vary	vary	out
5.27	2;3	PD/AF	vary	vary	vary	vary	vary	up
5.28	2,3	PD/AF	vary	vary	vary	vary	vary	up
5.29	6	PD/AF	vary	vary	vary	vary	vary	ready
5.30	2;3	AF	vary	vary	vary	vary	vary	up
5.31	2,3	PD/AF	1	2	1	1	2	up
5.32	2,3	PD/AF	1	2	1	1	2	up
5.33	2;3,W	PD/AF	1	2	1	1	2	up
5.34	2,3,W	PD/AF	1	2	1	1	2	up
5.35	3	PD	1	vary				down
5.36	3	AF			1	1	vary	down
5.37	6	AF			vary	1	2	ready
5.38	6	AF			vary	1	2	ready
5.39	6	AF			vary	1	2	ready
5.40	1,2,3	~~PD/AF~~	1	2	1	1	2	

Table B.1 Experimental Configurations

concerning the experiments follows.

In the case of experiments involving multiple joints, the following notation holds. Joints being tested are listed first. If multiple joints are tested, they are separated by commas. Joints acting as disturbances follow a semicolon. If multiple joints are used to create a disturbance, they too are separated by commas. A 'W' indicates that a weight of 1 kg has been attached to the end effector to increase the magnitude of the disturbance.

Fig	Joint	Test	Amplitude	Frequency	Disturbance	Frequency	Amplitude
5.9a	1	step	1000enc				
5.9b	1	sine	2000enc	1Hz			
5.10a	3	step	1000enc				
5.10b	3	sine	2000enc	1Hz			
5.11a	2	step	1000enc				
5.11b	2	sine	2000enc	1Hz			
5.12a	6	step	1000enc				
5.12b	6	sine	2000enc	1Hz			
5.13a	1	step	1000enc				
5.13b	1	sine	2000enc	1Hz			
5.14a	3	step	1000enc				
5.14b	3	sine	2000enc	1Hz			
5.15a	1	step	1000enc				
5.15b	1	sine	2000enc	1Hz			
5.16a	3	step	1000enc				
5.16b	3	sine	2000enc	1Hz			
5.17a	2	step	1000enc				
5.17b	2	sine	2000enc	1Hz			
5.18a	6	step	1000enc				
5.18b	6	sine	2000enc	1Hz			
5.19a	2,3	step	1000enc				
5.19b	2,3	step	1000enc		square	1.5Hz	3000enc
5.20a	2,3	sine	2000enc	1Hz			
5.20b	2;3	sine	2000enc	1Hz	square	1.5Hz	3000enc
5.21a	1;2,3	step	1000enc				
5.21b	1;2,3	step	1000enc		square	0.9Hz	1500enc
5.22a	1;2,3	sine	2000enc	1Hz			
5.22b	1;2,3	sine	2000enc	1Hz	square	0.9Hz	1500enc
5.23a	2;3	step	1000enc		square	1.5Hz	3000enc
5.23b	2,3	sine	2000enc	1Hz	square	1.5Hz	3000enc
5.24a	2,3	step	1000enc		square	1.5Hz	3000enc
5.24b	2,3	sine	2000enc	1Hz	square	1.5Hz	3000enc
5.25	1	step	500enc				
5.26	1	sine	1000enc	1Hz			
5.27	2,3	step	500enc		square	1.5Hz	1000enc
5.28	2;3	sine	1000enc	1Hz	square	1.5Hz	1000enc
5.29	6	sine	400enc	3Hz			
5.30	2,3	sine	8000enc	1Hz	square	1.5Hz	1000enc
5.31a	2;3	servo			sine	6Hz	500enc
5.31b	2;3	sine	2000enc	1Hz	sine	6Hz	500enc
5.32a	2;3	servo			sine	7Hz	300enc
5.32b	2,3	sine	1000enc	1.5Hz	sine	7Hz	400enc
5.33a	2;3	servo			sine	10Hz	200enc
5.33b	2,3	sine	2000enc	1Hz	sine	10Hz	200enc
5.34a	2,3	servo			sine	12Hz	200enc
5.34b	2;3	sine	2000enc	0.8Hz	sine	12Hz	200enc
5.35	3	sine	2000enc	1Hz			
5.36	3	sine	2000enc	1Hz			
5.37	6	step	500enc				
5.38	6	step	500enc				
5.39	6	step	500enc				

Table B.2 Experimental Movements

The configurations of the arm have been called *up*, *out*, *down*, and *ready*. In the *up* position, the arm is fully extended upward. The *out* position extends the arm horizontally from the shoulder. In the *down* position, link 2 is extended horizontally from the shoulder, but link 3 points down toward the table. The *ready* position is equivalent to the RCCL *park* position, in which link 2 extends vertically upward from the shoulder, and link 3 is horizontal.

References

ACSL 81 *Advanced Continuous Simulation Language (ACSL) User Reference Guide*, Mitchell and Gauthier Associates, 1981

Albus 75 Albus, J.S., "A New Approach to Manipulator Control The Cerebellar Model Articulation Controller," Transactions of the ASME, Journal of Dynamic Systems, Measurement and control, Vol 97, September 1975.

Anderson 71 Anderson, B. and Moore, *Linear Optimal Control*, Prentice Hall, 1971.

Asada 85 Asada, H., and Slotine, J-J E, *Robot Analysis and Control*, John Wiley and Sons, 1985.

Ballard 82 Ballard, D H, and Brown, C.M, *Computer Vision*, Prentice-Hall Inc, 1982.

Betts 70 Betts, J.A., *Signal Processing, Modulation and Noise*, Hodder and Stoughton, London, 1970.

Brigham 74 Brigham, E. Oran, *The Fast Fourier Transform*, Prentice-Hall Inc, 1974.

Carayannis 83 Carayannis, Gregory, "Controlling a PUMA 260 Robot from a VAX computer," *Computer Vision and Robotics Laboratory Technical Report No. 83-3*, McGill University, 1983.

Casti 85 Casti, John L., *Nonlinear System Theory*, Mathematics in Science and Engineering, Richard Bellman, ed, Vol.175, Academic Press, 1985.

Charniak 85 Charniak, E., and McDermott, D, *Introduction to Artificial Intelligence*, Addison Wesley Publishing Company, 1985.

Clark 81 Clark, D.W., "Introduction to Self-Tuning and Adaptive Control, Theory and Applications", C J. Harris and S.A. Billings, eds., *IEE*

Control Series 15, H. Nicholson and B.H. Swanick, series eds., Peter Peregrinus Ltd., London, 1981.

Craig 86 Craig, John J., *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, 1986.

Conte 80 Conte, S.D., and de Boor, Carl, *Elementary Numerical Analysis*, McGraw-Hill, 1980.

Daneshmend 87 Daneshmend, Laeeque, K., "Parameter Adaptive Control of Manipulators A Survey,", *in preparation*, 1987.

Denavit 55 Denavit, J., and Hartenberg, R.S., "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *ASME Journal of Applied Mechanics*, June 1955

Desa 85 Desa S., and Roth, B., "Synthesis of Control Systems for Manipulators Using Multivariable Robust Servomechanism Theory," *The International Journal of Robotics Research*, Vol. 4, No 3, Fall 1985.

Douglas 72 Douglas, J.M., *Process Dynamics and Control*, Vol 1-2, Prentice-Hall Inc., 1972

Driels 84 Driels, M R., and Yang, Z H, "Alternative Methods for the Analysis of Robot Arm Dynamics," Journal of Robotic Systems, Vol. 1, No. 4, 1984

Douglas 72 Douglas, J.M., *Process Dynamics and Control*, Vol. 1-2, Prentice-Hall Inc., 1972.

Franklin 81 Franklin, Gene F., and Powell, J. David, *Digital Control of Dynamic Systems*, Addison-Wesley, 1981.

Gonzalez 82 Gonzales, R.C., and Safabakhsh, R., "Computer Vision Techniques for Industrial Applications and Robot Control," *Computer*, Vol. 15, No. 12, December 1982.

Hayward 84 Hayward, V., and Lloyd, J., *RCCL Users Guide*, Department of Electrical Engineering, McGill University, April 1984

Hayward and Paul 84 Hayward, V., and Paul, R.P., "Introduction to RCCL: A Robot Control C Library", *First IEEE Conference on Robotics*, Atlanta, Georgia, March 1984.

Holland 83 Holland, J M., *Basic Robotics Concepts*, Howard W. Sams and Co., Inc., 1983.

Hollerbach 80 Hollerbach, John M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-10, No. 11, November 1980

Hudson 82 Hudson, Ronald L., and Kelly, Frederick A., "The development of Equations of Motion of Single-Arm Robots," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-12, No 3, May/June 1982

Jarvis 82 Jarvis, J F., "Research Directions in Industrial Machine Vision A Workshop Summary," *Computer*, Vol. 15, No 12, December 1982

Kahn 71 Kahn, M E., Roth, B., "The Near Minimum Time Control of Open Loop articulated Kinematic Chains," Transactions of the ASME, Journal of Dynamic Systems, Measurement and control, Vol ??, September 1971

Kane 83 Kane, Thomas R., and Levinson, David A., " The Use of Kane's Dynamical Equations in Robotics," *The International Journal of Robotics Research*, Vol. 2, No. 3, Fall 1983

Kent 86 Kent, E W., and Schneier, M O., "Eyes for Automatons," *IEEE Spectrum*, Vol. 23, No. 3, March 1986

Kernighan 78 Kernighan, W K., and Ritchie, D M., *The C Programming Language*, Prentice Hall, 1978

Koren 85 Koren, Y., *Robotics for Engineers*, McGraw-Hill, 1985

Lee 82 Lee, C G.S., "Robot Arm Kinematics, Dynamics, and Control", *IEEE Computer*, December 1982.

Letov 61 Letov, A M., *Stability in Nonlinear Control Systems*, Princeton University Press, Princeton, New Jersey, 1961

Lloyd 84 Lloyd, John, and Studenny, John, "Kinematic Equations and Solutions for the McGill Vision/Robotics PUMA 260," *Computer Vision and Robotics Laboratory Technical Report No. 84-3*, McGill University, 1984

Lloyd 85 Lloyd, J., *Implementation of a Robot Control Development Environment*, M. Eng. Thesis, Department of Electrical Engineering, McGill University, December 1985.

Lloyd 86 Lloyd, J., *RCCL Reference Manual*, Department of Electrical Engineering, McGill University, January 1986.

Lozano-Pérez 85 Lozano-Pérez, T., and Brooks, R.A., "An Approach to Automatic Robot Programming," Artificial Intelligence Laboratory Memo No 842, Massachusetts Institute of Technology, April 1985.

Luh 80 Luh, J. Y. S., Walker, M.W., and Paul, R.P.C., "On Line Computational Scheme for Industrial Manipulators," Transactions of the ASME, Journal of Dynamic Systems, Measurement and control, Vol. 102, June 1980

Luh 81 Luh, J. Y. S., Walker, M.W., and Paul, R.P.C., "Resolved Acceleration Control of Mechanical Manipulators," *IEEE Transactions on Automatic Control*, Vol AC-25, No. 3, June 1981

Luh 83 Luh, J. Y. S., "An Anatomy of Industrial Manipulators and Their Controls," *IEEE Transactions on Automatic Control*, Vol AC-28, No 2, February 1983

Mahil 82 Mahil, Surjit S., "On the Applicatio of Lagrange's Method to the Descriptiton of Dynamic Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-12, No. 6, November/December 1982

Morgan 85 Morgan, Russel G., and Osguner, Umit, "A Decentralized Variable Structure Control Algorithm for Robotic Manipulators," *IEEE Journal of Robotics and Automation*, Vol. RA-1, No 1, March 1985

Paul 81 Paul, R.P., *Robot Manipulators. Mathematics, Programming, and Control*, MIT Press, 1981

Paul et al 81a Paul, R.P, Shimano, B., and Mayer, G.E., "Kinematic Control for Simple Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 6, June 1981.

Paul et al 81b Paul, R.P., Shimano, B., and Mayer, G.E., "Differential Kinematic Control Equations for Simple Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-11, No. 6, June 1981.

Slotine 85 Slotine Jean-Jacques E., "Robustness Issues in Control," Proceedings International Conference on Robotics and Automation, St Louis, Missouri, March 1985

Smith 85 Smith, Carlos A., and Corripio, Armando B., *Principles and Practice of Automatic Process Control*, John Wiley and Sons, 1985

Studenny 83 Studenny, J., and Bélanger, P.R., "Robot Manipulator Control By Acceleration Feedback: A single Joint Paradigm," Computer Vision and Robotics Laboratory Technical Report Number 83-16R, Department of Electrical Engineering, McGill University, 1983

Studenny 84 Studenny, J., and Bélanger, P.R., "Robot Manipulator Control By Acceleration Feedback," *Proceedings of 23rd Conference on Decision and Control*, Las Vegas, Nevada, December, 1984

Studenny 86a Studenny, J., and Bélanger, P.R., "Robot Manipulator Control By Acceleration Feedback: Stability, Design and Performance Issues," Computer Vision and Robotics Laboratory, Technical Report TR-86-9R, McGill University, March 1986

Studenny 86b Studenny, J., and Bélanger, P.R., "Digital Implementation of the Acceleration Feedback Control Law to Robotic Manipulators," Computer Vision and Robotics Laboratory, Technical Report TR-86-3R, McGill University, June 1986

Studenny 87 Studenny, J., "Control of Robotic Manipulators Using Acceleration Feedback," Ph.D. Thesis, The Department of Electrical Engineering, McGill University, January 1987

Utkin 77 Utkin, Vadim I., "Variable Structure Systems with Sliding Modes," *IEEE Transactions on Automatic Control*, Vol. AC-22, No. 2, April, 1977

Valvanis 85 Valvanis, Kimon P., Leahy, Michael B., and Saridis, George N., "Real Time Evaluation of Control Methods," Proceedings International Conference on Robotics and Automation, St. Louis, Missouri, March 1985

Vukobratović 82 Vukobratović, M., and Stokić, D., "A Procedure for the Interactive Dynamic Control Synthesis of Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-12, No. 4, July/August 1982

Whitney 69 Whitney, D.E., "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man Machine Systems*, Vol. MMS-10, June 1969

Willems 76 Willems, J.C., "Lyapunov Functions for Diagonally Dominant Systems", *Automatica*, Vol. 12, Pp. 519-523, Pergammon Press, 1976

Winston 84 Winston, P.H., and Horn, B.K.P., *LISP*, Addison-Wesley Publishing Company, 1984.

Wu 82 Wu, Chi-Haur, and Paul, R.P., "Resolved Motion Force Control of Robot Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-12, No. 3, May/June 1982.