NONLINEAR FORCE FEEDBACK IMPEDANCE CONTROL

by

John J. Wlassich

BSME, University of Rhode Island
(1983)

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 1986

Signature of Author _____
Department of Mechanical Engineering
February 10, 1986

Certified by _____
Neville Hogan
Thesis Supervisor

Accepted by _____
Ain A. Sonin
Chairman, Departmental Committee on Graduate Students

NONLINEAR FORCE FEEDBACK IMPEDANCE CONTROL

by

John J. Wlassich

Submitted to the Department of Mechanical Engineering
on January 17, 1986, in partial fulfillment of
the requirements for the degree of
MASTER OF SCIENCE in Mechanical Engineering

ABSTRACT

A critical issue for manipulators that interact with their environment is the stable control of interface forces. Impedance control is a strategy to specify a manipulator's reaction to external disturbances. The goal of this thesis was to determine the efficacy of force feedback applied to impedance control.

A torque control law was derived to impose a controllable apparent impedance on a serial link mechanism. In this study the imposed impedance was a linear second order behavior in Cartesian coordinates of the linkage endpoint. No inverse kinematic calculations were necessary to reach specified endpoint positions. The control law was implemented on a PDP-11/23 computer and tested using a two link mechanism. Unconstrained endpoint speed and acceleration of the mechanism exceeded the fastest commercially available robots. Trajectory control was satisfactory. The desired impedance was achieved within hardware limitations.

Force feedback improved dynamic interactive behavior. Force feedback was shown to modulate the apparent mass of the endpoint. Stable interaction with a rigid environment was demonstrated for conditions that exceed stability constraints reported in the current literature. Simple models of the system were developed to explain the stability results of this thesis and relate them to current force feedback literature.

Thesis Supervisor: Professor Neville Hogan

Title: Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

It gives me great pleasure to acknowledge Neville Hogan for the guidance he gave me during the course of this investigation. I admire his gifted insight. The patience and respect he showed me was deeply appreciated. He will continue to have a profound influence on my development as a researcher.

I thank my parents for supporting and caring for me always. The many cards and letters from my sister Marilyn kept my desk full of joyous thoughts. I love you all.

Special thanks go to Cary and Bill. They answered my questions kindly and took the time to teach me on several occasions. Ed, I learned from your careful approach to problem solving. Carolina, my heartfelt thanks for acquiring the references I needed so much to complete this thesis. This research was supported by Daewoo Heavy Industries and was performed in the Eric P. and Evelyn E. Newman Laboratory of Biomechanics and Human Rehabilitation and in the Laboratory for Manufacturing and Productivity.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

B	damping coefficient
B_d	desired damping coefficient
C	compliance
$C(\Theta,\dot{\Theta})$	coriolis and centripetal terms
D_k	first order difference
E	modulus of elasticity
F_a	actuator force
F_e	external force
F_c	command force
F_{TD}	force transducer force
G_a	combined amplifier gain
G_{ch1}	amplifier channel #1 gain
G_{ch2}	amplifier channel #2 gain
G_f	force feedback gain
G_p	plant or position feedback gain
G_{ta}	transconductance amplifier gain
$G(\Theta)$	gravity terms
G_v	velocity feedback gain
h	generic increment
h_1, h_2	distance to center of mass of link #1, link #2
h_3, h_4	distance for center of mass of follower and input
$H(\Theta,\dot{\Theta})$	acceleration terms
I_1, I_2	link #1, link #2 inertia
$I(\Theta)$	inertia tensor
$J(\Theta)$	manipulator Jacobian
k	as a subscript, means "current"
K	stiffness coefficient
K_d	desired stiffness coefficient
K_{TD}	force transducer stiffness
L_1, L_2	length of link #1, link #2

L_3, L_4	lengths of four bar follower and input links
$L(\Theta)$	link geometry
m_1, m_2	mass of link #1, link #2
m_3, m_4	mass of four bar follower and input links
M_a	actual mass
M_d	desired mass
Φ	a function of ...
$R(\Theta)$	rotation matrix
SF	sampling frequency
$S(\Theta)$	static friction terms
Θ	absolute joint angle
Θ_{1a}	joint #1 absolute angle
Θ_{2a}	joint #2 absolute angle
Θ_{1r}	joint #1 relative angle
Θ_{2r}	joint #2 relative angle
W	mobility
V	actual velocity
V_e	velocity dictated by the environment
V_k	first order smoothing
$\dot{V}$	actual acceleration
$V(\dot{\Theta})$	viscous damping terms
X	actual position
X_c	command position
x_e	position dictated by the environment
Y_E	admittance of environment load
Y_M	admittance of manipulator

Introduction

1.1 Issues addressed in this investigation.

This investigation addresses a fundamental problem encountered by manipulators which interact with their environments: controlling and regulating interface force. The general problem of stable mechanical interaction is not restricted to robotics. Human limb protheses must work on objects to be useful.

Functional manipulations may have three phases. In phase one the mechanism is unconstrained by its manipulatory task; the manipulator is free of any load. Control schemes for movements in this phase are well understood and have been applied successfully. Phase two is the moment of transition between free and loaded movements. In phase three work is done by the manipulator on its environment. A smooth, stable behavior is desired but rarely accomplished in phases two and three by control schemes in use today. Usually they require powerful actuators and rigid structures. High interface forces can can result due to inaccuracies in the position and orientation of tool relative to workpiece. The desire for accuracy implies high-gain position feedback control, and this often exacerbates the problem.

Damage to robot and or load during movements or during a function such as assembly has been one of the primary motivations for investigating force feedback [36], [51]. Examples of tasks that require all phases from free to coupled motions are: putting pegs in holes, threading bolts in holes, and opening a door. The peg in hole problem occurs often in assembly [36]. An elegant solution to this problem is the passive Remote Center Compliance

device [11]. The RCC is based on a good kinematic knowledge of the problem. Threading bolts in holes is the next most common assembly operation. A passive device for bolting was designed by Dean [10]. This investigation is part of that project [22].

1.2 One proposed solution to this problem - impedance control.

One of the fundamental contraints on interacting systems is that it is not possible for one system to prescribe both the effort and the flow at the point of interaction [41], [45]. For example, an ideal position controller may impose a precise displacement on an object, but it can not at the same time control the interaction force. Likewise an ideal force controller may exert a desired force but cannot control position in the same direction. An alternative is to control one of the two variables at the interaction port while also specifying the relation between them. This strategy is at the heart of impedance control; the specified relation is impedance.

Impedance control assumes the environment is an admittance (e.g. a mass) to which forces may be applied. This is not a restrictive assumption when one considers , for example, the objects that are to be manipulated by a typical assembly robot. Tools and workpieces are often metal and appear as nearly rigid masses to the robot. The applied forces account for spring and damper like aspects of the load that appear behind the rigid mass that the robot first encounters. Given that the environment is modelled as an admittance, causality dictates that the mechanism should appear as an impedance [21].

How then does impedance control command desired endpoint positions? To an impedance control algorithm position commands set up potential energy

wells [2]. The mechanism seeks these wells to achieve a lower energy state and thus attains the desired position. A completely analagous situation is that of a spring whose rest length has been changed from a previous length and whose endpoint now seeks to come to equilibrium at the new rest length. Thus at steady state spring force and position are zero.

Anything attached to the endpoint experiences a force proportional to the spring's stiffness (an impedance) and displacement. Position commands in impedance control take the form of changes in spring rest length and are thus transformed to desired forces. Ultimately positional accuracy depends on, for example, the chosen stiffness and dissipative elements associated with the load or impedance controlled device.

By varying a device's impedance, as seen by its environment, the manipulator can be "tuned" to the task [21]. An example of a task where variable impedance would be beneficial is finding an edge. For many reasons, such as mechanical deflections, controller accuracy, and sensor accuracy, the tool to edge distance may be in error. A low stiffness, moderately damped mechanism might approach the edge and find it (using a contact sensor) at high speed since the resulting interface force will be low. On contact the mechanism could tune up its stiffness parallel to the edge to track it closely but keep its perpendicular stiffness low and damping high to react to irregularities projecting from the edge. This example resembles deburring, an operation where mechanical interaction between manipulator and environment is fundamental.

Thus the response of the manipulator to external disturbance can also be controlled. The mechanism's actuators provide forces that are a function of the desired impedance (i.e. stiffness and damping) and endpoint diplacements,

whether internally generated as a position command or externally provided by loading.

Impedance control need not only be implemented by electronic feedback means [27]. By varying the physical stiffness and damping of hardware actuators, they made be made to react to impacts (recall "phase two" of the free to loaded control problem) in a continuous manner that may not be achievable by any digital controller, no matter how small the sampling period. Each approach has its benefits and drawbacks. It appears that pure discrete control can work on errors, in position for example, with better accuracy than impedance controllable hardware can [27], [7]. A scheme to combine intrinsic and electronic forms of impedance control, thereby reaping benefits from both schemes, is under consideration.

The study reported here was a purely electronic implementation of impedance control. A digital algorithm was developed to control a two serial link mechanism in a plane perpendicular to the gravity vector. The nonlinear nature of the hardware lead to a nonlinear impedance controller. Force feedback was incorporated to uncover issues in interface force control. An implicit goal of this investigation was to compare the benefits of programming flexibility with the problems inherent to digital control implementation [5] as they relate to impedance control.

1.3 Nonlinear force feedback impedance control.

Impedance control, in this inquiry, was entirely discrete. The control law was derived for continuous conditions and implemented digitally with the assumption that the sampling period would be small enough to approximate continuous control. The aim of nonlinear force feedback impedance control,

henceforth referred to as NFFIC, was to impose a desired linear, second order, time invariant behavior on a nonlinear mechanism. The mechanism most resembled a SCARA [32] robot. Desired impedance was linear in endpoint (Cartesian) coordinates and nonlinear in joint coordinates. The hardware provided a real life test bed for determining controller performance.

1.4 Brief review of some of the following chapters.

Chapter 2 - Force Feedback, Force Control, and Impedance Control.

Current methods to control nonlinear mechanisms and interface forces are reviewed. A quantitative explanation of force feedback action is developed. The use of force feedback in impedance control is motivated. A brief discussion of causality is presented to familiarize the reader with concepts used in the next chapter.

Chapter 3 - Derivation of Nonlinear Force Feedback Impedance Control.

The NFFIC algorithm is derived for n degrees of freedom using linear algebra. A robot model is proposed and a model of the desired behavior is presented. These models in conjunction with formulations based on link geometry are used to derive a closed form solution for actuator torques. A stability criterion for the one degree of freedom case is developed.

Chapter 4 - Hardware Setup.

A description of the mechanism built to test NFFIC, the computer, and support devices and electronics is given. Characterization of the hardware, design constraints, and safety considerations are reviewed.

Chapter 5 - Controller Implementation.

A detailed look at the three control schemes developed. Two of these schemes, analog position and force feedback, and digital linear position, velocity, and force feedback are used to test NFFIC stability. The digital algoritm for the NFFIC law specific to the mechanism decribed in chapter four is developed. Some discrete differentiation formulae for rate feedback are evaluated and their relative efficacy demonstrated. Software features are explained.

Chapter 6 - Dynamic Interaction Test Results.

Stability and relative performance, in regimes delineated by desired apparent mass relative to actual apparent mass, is presented. Stability is demontsrated for a more extensive set of conditions than reported in current force feedback literature. Attempts at stabilizing force feedback in regimes where instability occurred are described.

Chapter 7 - Dynamics Investigation.

An investigation into the two link mechanism's dynamics. Models are developed to understand better the attempts at force feedback stabilization described in the previous chapter. Force feedback stability results from this investigation are related to stability results in current force feedback literature.

Chapter 8 - Unconstrained Motion Test Results.

NFFIC test results when the mechanism is free to move without external disturbance are presented. Path tracking, limits of achievable stiffness and

damping, and mechanism speed and acceleration comparisions are tabulated. An alternative on the discrete differentiation scheme used to generate a velocity feedback signal is presented and its effect on performance is detailed.

Chapter 9 - Discussion and Conclusions.

The principle findings of this investigation are summarized and further directions for research are suggested.

Force Feedback, Force Control and Impedance Control

2.1 Scope of this chapter.

This chapter begins by reviewing control schemes that are intended to control manipulators and/or interface forces. Stability is shown to be the key issue in force control today. The action of a force feedback loop is then explained. The concept of causality is briefly reviewed and tied to impedance control.

2.2 Controllers in current literature that are designed to deal with manipulation.

The advent of relatively fast and powerful digital computers has lead to research aimed at designing controllers to tackle the fundamentally nonlinear [21] task of manipulation.

Nonlinear control stragegies are proposed to reduce position errors encountered when linear controllers work on fast moving nonlinear mechanisms. This section reviews attempts at manipulator control that incorporate or could incorporate force feedback. The computational complexity of nonlinear controllers is fast becoming a nonissue. Efficient formulations along with faster microprocessor clock cycles are allowing discrete controller bandwidths of 10 Hz and more [7]. The general problem of nonlinear control strategies can be studied in [15].

There appears to be two categories of nonlinear control strategies for manipulators. One approach might make use of external force information, however it is not stated explicitly. The other approach uses force feedback

explicitly to work on interface forces.

2.2.1 Non-explicit use of force information.

Recursive Lagrangian or Newton Euler [31], [19].

This strategy seeks to solve the equations of motion (derived by Lagrangian or Newtonian means) for the required joint torques to reach a commanded position, in real time. The recursive aspect is that each link's coupled dynamics are solved using previous link information. Endpoint reference positions are specified. Joint positions, angular velocities, and accelerations are measured. The proper torque to get a desired endpoint motion relies on an accurate model of the mechanism. A load inertia must be known or calculated for coupled motions to be precise. The author has inferred that since equations of motion include generalized forces, interface force could be part of this control scheme as inputs to calculated actuator torques. However the problem of trying to specify both position and force at the same time will arise, which is physically impossible. A full six degree of freedom simulation, without force information, has been shown to cycle under 5 ms [30].

Resolved Motion Position Control [40].

Desired tool orientations are specified. Joint positions are sensed to calculate the actual tool position by the forward kinematics approach [40]. An error is then derived from the difference between actual and desired tool positions. This error is then resolved into joint coordinates using a Jacobian transformation. Position control loops are closed about each joint to force the tool position error to zero. An alternate approach to the independent position

loops is to estimate or measure joint velocities and accelerations for use with joint position in a Lagrangian or Newton Euler formulation.

Resolved Motion Rate Control [53].

The desired endpoint velocity is transformed into joint coordinates by an inverse manipulator Jacobian. A velocity feedback loop is then closed about each joint. Tool position errors are reduced by the velocity control loops. One can speculate however that in the middle of a movement, when velocity is at a maximum, configuration dependent forces (coriolis, centripetal) might be large compared to control. External forces may be included into an extension of this method which is covered after this subsection.

Resolved Motion Acceleration Control [31].

The desired endpoint position, velocity, and acceleration to complete a move are transformed into joint coordinate torques. Actual joint angles are measured and compared to desired joint angles. The error is used in a quadratic function that aims to asymptotically drive the error to zero. The two coefficients in the quadratic function are chosen based on a model of the manipulator. The model assumes small tool orientation errors and a short sampling interval. Nonlinear quantities such as coriolis, centripetal, and gravity terms are not calculated. One might speculate that ignoring these terms may lead to tool orientation errors beyond those anticipated.

Configuration Space [43].

This approach requires an off-line computation of all possible geometry dependent terms of a nonrecursive Lagrangian formulation, for the entire

mechanism workspace. Once computed, the real time control algorithm essentially looks up the correct values given the measured manipulator configuration. The real time algorithm is thus fast, but requires storage of large arrays of data. Memory size may not be an issue since the cost for such storage is decreasing. However, different loads require different data arrays. This approach might then be constrained by finite memory size to tasks that have been accurately modelled.

Model Referenced Adaptive Control [12].

Position and velocity are fed back. The adaptive part of the controller modulates the gains on these feedback states in order to achieve performance specified by a desired model. The error between desired and modelled states is worked on by a minimizing scheme that uses a quadratic error function. Like NFFIC, the response to a load is predicted by the model that is being emulated. Feedback gains however do not have physical meaning as in NFFIC where feedback gains are impedances.

Resolved Motion Force Control [55].

Actual tool position is calculated from measured joint positions and velocities. Desired tool acceleration is estimated. Desired tool position and velocity is compared to calculated position and velocity to form error signals. Position and velocity errors are resolved into accelerations which are combined with the estimated desired acceleration. This final tool coordinate acceleration is multiplied by a model of the load inertia to form the desired tool force. Desired tool force is then transformed to actuator torques by the manipulator's Jacobian. Sensed interface force and desired force are then

subjected to a minimizing scheme to reduce torque errors. The force converging scheme is intended ultimately to reduce position errors.

2.2.2 Explicit use of force information.

Explicit Interface Force and Position Control [50], [18].

Efforts to apply classical linear control techniques have had successes. Most require slow movements and do not address large moves. Despite these restrictions, simple mechanisms with well defined tasks may be better suited to such techniques than a more complex controller. The conflict of simultaneous position and force commands will always result in less than desireable behavior.

Hybrid Position/Force Feedback Control [42].

The geometry of tool and object to be worked on determine which axis (in tool coordinates) is to be force controlled and which axis position controlled. Nominally force is controlled along the axis perpendicular to the work surface and position controlled parallel to the work surface. A general method to determine the correct axis/feedback state combination is explored in [33]. Each joint is position and force controlled by resolving the tool force axis and position axis into individual joint coordinates. The "guarded move" referred to at the transition from an uncoupled manipulator to one that is contacting a surface is an effort to deal with instabilities noted at this transition. This moment of transition is a point where both position and force are trying to be specified simultaneously.

Stiffness Control [46].

Sensed external forces are multiplied by a compliance matrix in the feedback path. The result is subtracted from the reference position. The feedback path can be characterized by the compliance relation:

$$X_C = C F_e \qquad \text{eq. 2.0}$$

This equation says that for an applied force a position is returned. This last sentence describes an admittance. In the general case if the environment were an impedance making the robot appear as an admittance would be the correct causal choice. However, given the arguments in section 1.2, the environment is an admittance and specifies the motion; in casual conflict with a robot trying to specify motion. Argueably, with high enough compliance loop gain the result is masked plant dynamics replaced with inverse loop dynamics, namely a stiffness. The manipulator now has causality consistent with an admittance model of the environment provided. The requirement then is a low intrinsic stiffness manipulator if a range of desired stiffnesses is to be achieved.

Damping Control [53].

Force is fed through a damping matrix before being added to a specified velocity. This approach has no causal conflict since damping has no causal preference [45]. Damping control could be used in conjunction with RMRC, since rate is specified. An interesting stability result is developed in [53]. In essence it says that force feedback stability margin can be increased by using a relatively compliant force sensor, given that one needs a stiff manipulator for accurate positioning and that the types of loads to be worked on are also

stiff. A stiffer force sensor can be used for the same manipulator if the environment is more compliant than in the last case. An investigation into sensor stiffness and controller bandwidth in [44] yields essentially the same conclusion.

2.2.3 Force sensing as used in industry today.

Most force information schemes applied to commercially available robots use "logic branching" [52]. Examples include torque overload sensing and edge finding. A control scheme may call for turning a bolt until a certain torque is reached [13]. Continuous force feedback in industry involves human operators to close the loop; for instance telemanipulators to work on hazardous substances. The author knows of no electronic feedback control scheme, in wide use today, that uses force information to accomplish or directly assist in manipulation.

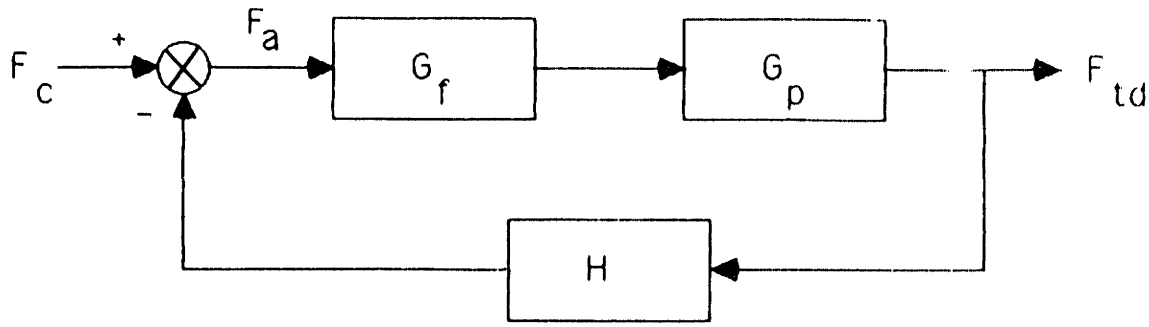
2.3 What does explicit force feedback accomplish?

Figure 2.0 represents a one degree of freedom negative force feedback loop. The transfer function for this system is:

$$\frac{F_{td}}{F_c} = \frac{G_f G_p}{1 + G_f G_p H} \qquad \text{eq. 2.1}$$

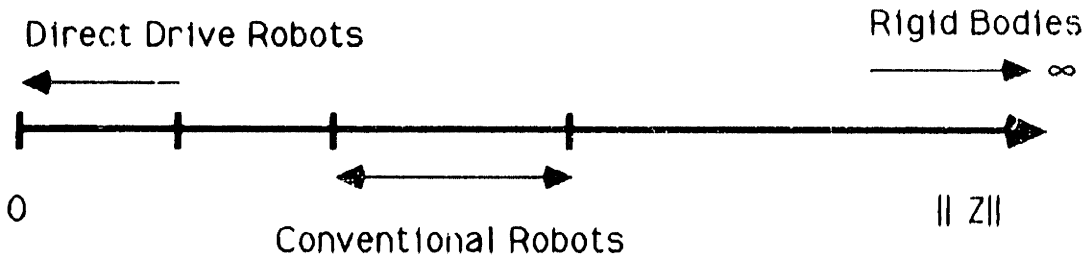
As Gc is increased (from the loop inversion theorem [38]):,

$$\lim_{G_f \to \infty} \frac{F_{td}}{F_c} = \frac{1}{H} = 1 \quad ; \quad H \equiv 1 . \qquad \text{eq.2.2}$$



Explicit negative force feedback control.

Figure 2.0



Span of intrinsic impedance.

Figure 2.1

This result means that the force output from the plant can be set precisely by the commanded force. As viewed by an observer external to the plant, forces perturbing the endpoint are assisted by command forces. Implicit in this statement is the result that plant dynamics are masked or wiped out. An example is power steering in an automobile; one does not feel the linkage inertias and road friction. For mechanical systems masked dynamics means that intrinsic mass, stiffness, and damping are altered.

Suppose one were to change the feedback sign so that figure 2.0 now represented positive force feedback. Positive feedback does not necessarily imply an unstable controller The resulting transfer function would be:

$$\frac{F_{td}}{F_c} = \frac{G_f G_p}{1 - G_f G_p H} \qquad \text{eq. 2.3}$$

and as before,

$$\lim_{G_f \to \infty} \frac{F_{td}}{F_c} = -1 \qquad \text{eq. 2.4}$$

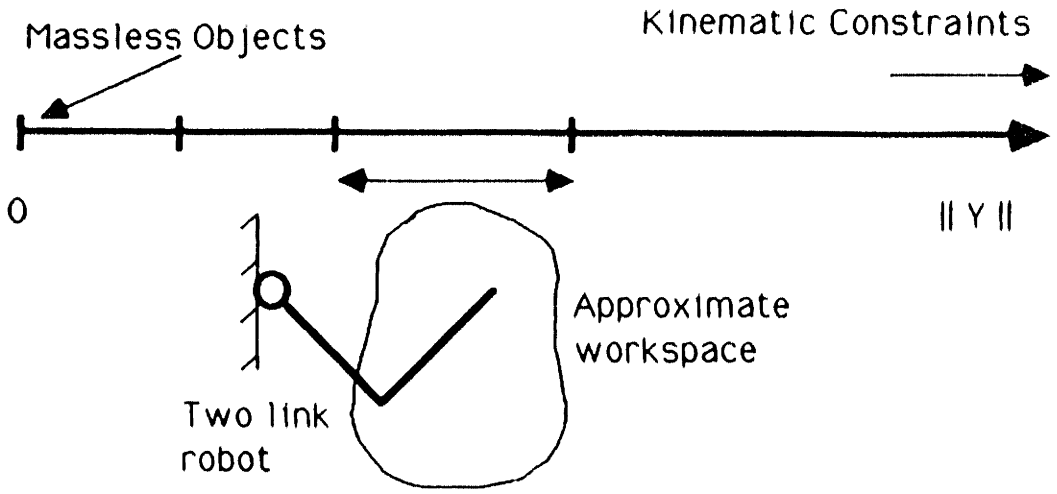
Equation 2.4 says that the system being controlled would generate a force equal but opposite to the desired force. From the point of view of an observer external to the plant, what does positive force feedback appear to be? The plant will resist applied external forces in a positive force feedback loop. In summary, negative force feedback attenuates mechanism dynamics, positive force feedback amplifies dynamics.

In classical control terms it appears logical to work on the state you wish to control. Single input single output control works well in many applications. A modern control approach is to view system behavior before and

after control scheme implementation. The change in behavior can be described in dynamical terms. A high gain negative feedback position control loop results in an effective increase in stiffness. Velocity feedback modulates apparent damping. The results can be depicted on a pole zero plots [38]. Whether or not poles move as a result of intrinsic hardware changes or by the action of an electronic controller is not apparent to an observer external to the system.

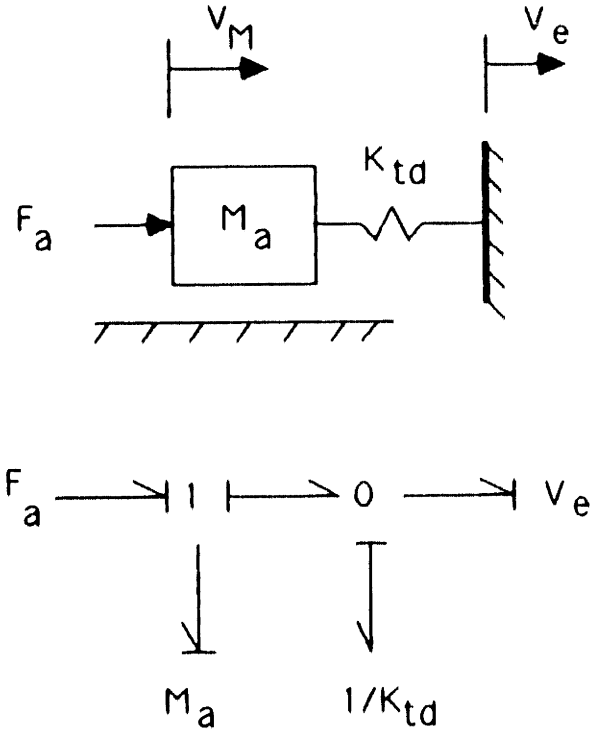
What is meant exactly by attenuating or amplifying plant dynamics? The dynamics, for the simple control loop in figure 2.0, reside in the plant. What kind of plants exist? Figure 2.1 represents a span of mechanical system impedances. On the low end are, for example, frictionless joints and drive trains of a hypothetical robot. On the high end is a rigid body. A direct drive robot [4] appears near the origin, its intrinsic friction and damping are very low. A typical industrial robot, an IBM 7565 for example, has a clearly viscious behaviour and measurable stiffness. Commercially available robots appear somewhere to the right of direct drive robots. These aspects of open loop dynamics are commonly referred to as backdrivability.

A similar span for admittance can be drawn, see Figure 2.2. The origin respresents kinematic linkages; massless rigid bodies. Large admittances are real rigid bodies, the ideal being a mechanical ground, which appears far to the right. A two link mechanism where both links form a line has a large admittance as seen by the environment looking straight down the links. The same two link device, moving within a space loosly defined by the dashed line in figure 2.2, has a varying apparent endpoint inertia [21] as seen from the same vector as the previous case. A mechanism's apparent inertia then is a function of the mass of its links, drive train, and geometry. Intrinsic



Span of intrinsic admittance.

Figure 2.2



Model to show force feedback action on a low impedance plant.

Figure 2.3

impedance can be minimized by designing low friction joints and drive trains but some amount of admittance will always remain as a physical consequence of the manipulator's mass.

A force feedback loop around a direct drive robot thus modulates inertia since impedance is already low. The same loop closed on more typical position controlled robots works on a combination of inertia, stiffness, and friction; the complete dynamics. The serial link mechanism in this thesis most closely resembles a direct drive robot. No attempt was made to mechanically decouple the device's inertia tensor. The proposed NFFIC law presented in the next chapter intends to decouple the links electronically.

One could suggest that to vary the admittance, as seen by the environment, force feedback may not be necessary. A strong, lightwieght link decoupled from the bulk of the robot, a micromanipulator for instance, could present the tool to the workpiece. Any interactions now accelerate a much smaller mass than if the robot were doing the manipulation directly.

Does this discussion of force feedback action on dynamics hold up analytically? Consider a mass M_a with forces acting on it, see Figure 2.3. This is a model of a one degree of freedom robot, without friction, interacting with an unyielding environment. Motions imposed by the environment are represented by V_e. A commanded force F_c, moves the robot. A force transducer with some stiffness K_{td}, senses external forces. This model most closely resembles a direct drive robot. Force feedback gain must be shown to modulate the apparent mass of the model if the discussion in the previous paragraphs is valid. The bond graph for this system appears in figure 2.3 from which the equations of state can be derived:

$$\begin{vmatrix} \dot{X} \\ \dot{V} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ \frac{-K_{td}}{M_a} & 0 \end{vmatrix} \begin{vmatrix} X \\ V \end{vmatrix} + \begin{vmatrix} 0 & 0 \\ \frac{1}{M_a} & \frac{K_{td}}{M_a} \end{vmatrix} \begin{vmatrix} F_a \\ X_e \end{vmatrix} \qquad \text{eq. 2.4}$$

$$\begin{vmatrix} F_{td} \end{vmatrix} = \begin{vmatrix} K_{td} & 0 \end{vmatrix} \begin{vmatrix} X \\ V \end{vmatrix} + \begin{vmatrix} 0 & -K_{td} \end{vmatrix} \begin{vmatrix} F_a \\ X_e \end{vmatrix} . \qquad \text{eq. 2.5}$$

impose an explicit force feedback law on the system, like figure 2.0

$$F_a = G_f (F_c - F_{td}) . \qquad \text{eq. 2.6}$$

The controlled system is now characterized by:

$$\begin{vmatrix} \dot{X} \\ \dot{V} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ \frac{-K_{td}}{M_d} & 0 \end{vmatrix} \begin{vmatrix} X \\ V \end{vmatrix} + \begin{vmatrix} 0 & 0 \\ \frac{1}{M_a} & \frac{K_{td}}{M_d} \end{vmatrix} \begin{vmatrix} F_a \\ X_e \end{vmatrix} . \qquad \text{eq. 2.7}$$

Where:

$$M_d = \frac{M_a}{1 + G_f} \qquad \text{eq. 2.8}$$

Note the open loop mass, M_a, can be related to the closed loop apparent mass, M_d, by the force feedback gain, G_f.

To reconcile the difference between the new closed loop mass definition

in both the physical characteristics matrix (the left 2 by 2 in equation 2.7) and the input matrix (the right 2 by 2 in equation 2.7), try a slightly modified explicit force feedback control law:

$$F_a = (1 + G_f) F_{td} - G_f F_c . \qquad \text{eq. 2.9}$$

As before, form the closed loop behavior,

$$\begin{vmatrix} \dot{X} \\ \dot{V} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ \frac{-K_{td}}{M_d} & 0 \end{vmatrix} \begin{vmatrix} X \\ V \end{vmatrix} + \begin{vmatrix} 0 & 0 \\ \frac{1}{M_d} & \frac{K_{td}}{M_d} \end{vmatrix} \begin{vmatrix} F_a \\ X_e \end{vmatrix} . \qquad \text{eq. 2.10}$$

Now the two square matrices of equation 2.10 contain closed loop mass only. Figure 2.4 is a block diagram representation of the closed loop system of equation 2.10.

In this example, force feedback gain, G_f, was driving the analysis. Suppose one were to desire a particular closed loop apparent mass, what force feedback gain will accomplish this task? Closed loop mass refers to the system's apparent mass as seen by the environment external to the closed loop system. Reformulate equation 2.8.

$$G_f = \frac{M_a}{M_d} - 1 \qquad \text{eq. 2.11}$$

Now some useful results appear. G_f, as defined in figure 2.0, is for negative force feedback. If one asks for a desired mass less than actual mass,

G_f remains positive, the loop sign is negative and the plant's actuators will assist any external forces. If one asks for a desired mass greater than actual mass, G_f goes negative, the loop sign is positive and the plant's actuators resist any external forces. If desired and actual mass are equal, force feedback is turned off and the plant's actuators do not respond to external forces sensed by a force transducer.

Figure 2.5 depicts the entire range of force feedback gain versus normalized desired mass. The vertical axis is in nondimensional units of force feedback gain and the horizontal axis is in units of multiples of actual mass defined by:

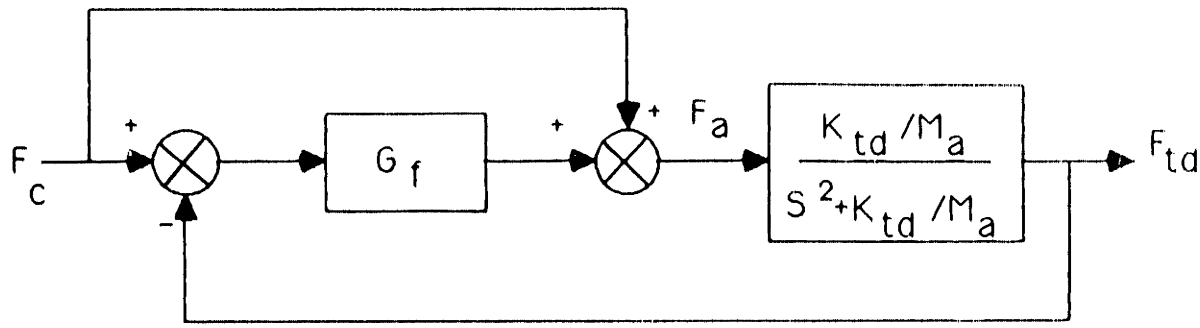
$$K \equiv \frac{M_d}{M_a} \qquad ; M_d = K M_a \ . \qquad \text{eq. 2.12}$$

Combining with equation 2.11 ,

$$G_f = \frac{1}{K} - 1 \qquad \text{eq. 2.13}$$

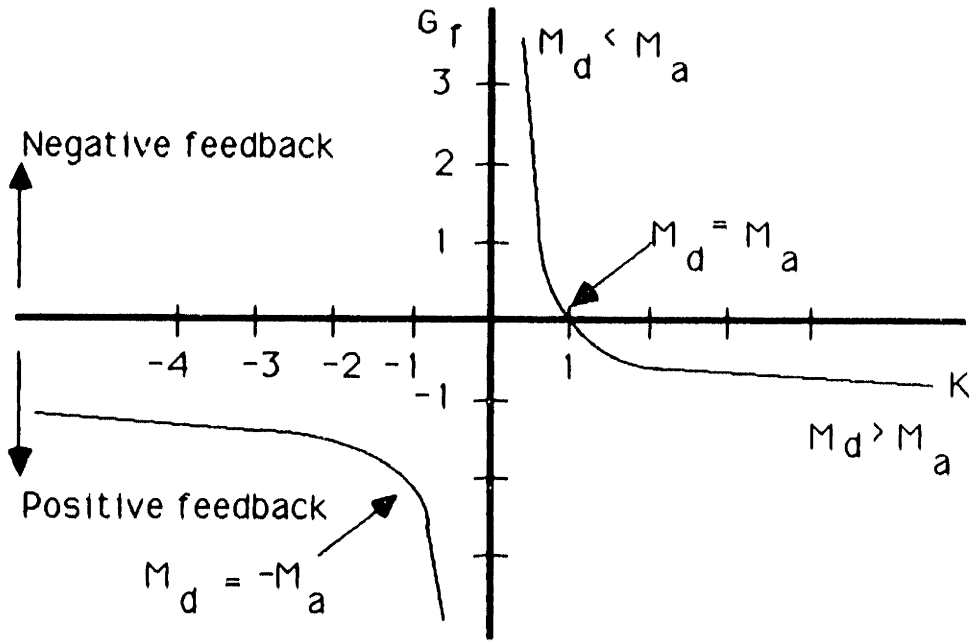
This theoretical development yields the same results as the qualitative dissussion at the start of this section. The action of force feedback is a tuning of manipulator dynamics, i.e. impedance control. Some important results can be drawn from figure 2.5. Note that for desired mass less than actual mass, force feedback gain gets large very quickly. For desired mass greater than actual mass, force feedback gain approaches negative one rather slowly.

The incremental change in positive force feedback gain decreases as



Adjusted force feedbacck control.

Figure 2.4



Force feedback gain versus normalized mass.

Figure 2.5

apparent mass increases. The opposite is true for negative force feedback; the incremental change in negative force feedback grows as apparent mass decreases. The effect is a larger absolute force feedback gain to decrease apparent mass than to increase apparent mass by the same percentage. Actuators in the system will require more power to acheive reduced mass than increased mass. The issue of force feedback gain will be linked to stability in chapter six.

2.4 Causal motivation for impedance control.

In previous section force feedback was shown to modify the intrinsic dynamics of a mechanism. If one could replace the reduced intrinsic dynamics with a desired behavior, the response to disturbances impinging on the mechanism could be controlled. Further, if the replaced dynamics were in the form of an impedance, deviations from targeted positiions would give rise to forces in a predictable way. Consider as an example a spring and damper in parallel. If the spring and damper are deflected the resulting force output by them is:

$$F = K(X) + B(\dot{X}) \qquad \text{eq. 2.14}$$

If mechanical impedance, denoted by K and B in equation 2.14 could be controlled, then the force resulting from perturbations in length is also controlled. In fact, K and B need not be linear.

Specifying a target impedance is not an arbitrary choice. Since NFFIC is designed for the class of mechanical devices that manipulate, the kinds of objects to be manipulated are predominantly inertial. Whether it is a human using a limb prothesis to lift a cup or a robot performing Z stack assembly,

objects first present themselves as masses. There can be springs and dampers behind the mass, which impart forces to the mass, hence the assumption taken in this work that admittances can be accompanied by forces. One might argue that a vat of grease is almost entirely a viscous element. This is true but not part of the class of environments under consideration. The class being considered is not very restrictive when one considers all of the tasks that can be modelled with masses first, followed by springs, dampers or other force sources. Investigating NFFIC response to environments other than those just described is beyond the scope of this work.

If the environment is an admittance, the correct causal choice for a manipulator's behavior is that of an impedance [21]. An implicit assumption here is that it is the robot which we wish to control rather than environment. It is therefore the duty of the impedance controlled robot to tailor its behavior to suit various environments.

Admittances dictate the flow resulting from efforts applied to them. As an example consider a mass:

$$V_{result} = \int F_{input}/M \qquad M : 1 \left| \frac{V}{F} \right. \qquad \text{eq. 2.15}$$

Constitutive Equation Bond Graph Element

This equation is written in its integral causal form [45]. Acausal equations appear with derivatives of inputs. Equation 2.15 says that the mass sums forces and gives back a velocity. If we wish to move this mass we do so by applying forces. To dictate its movements would conflict with its inherent behavior, yet this is fundamentally what position controlled robots seek to accomplish. The result is inertial accelaration loading. The vector velocity

of the load, and the trajectory tracing of the robot conflict.

A physical example of the force summing action of a mass is a boat in water. If several people were to push on the boat along different vectors the result would be an acceleration along only one vector. Only viscous friction would cause the motion to come to steady state.

A more natural, causal way to manipulate objects, other than position control, is to impress a force on them. This can be accomplished by having the manipulator appear as an impedance. The way impedances and admittances complement each other is known as a causal duality.

Impedances dictate efforts resulting from flows applied to them. An example of an impedance is a spring:

$$F_{result} = \int K \dot{X}_{input} \qquad C : K \left| \frac{F}{\dot{x}} \right. \qquad \text{eq. 2.16}$$

Constitutive Equation Bond Graph Element

Equation 2.16 says that positions are summed and forces returned, the dual of the admittance described by equation 2.15. A spring is a strain element. Deflect it and a force pushes back. Can one dictate the force and deflection of a spring simultaneously? No, these are related by a single parameter, the spring constant K.

One could however, choose a deflection and pick from various springs to find the right K that results in the force desired. This is the crux of impedance control, tuning a mechanism's impedance to achieve the desired response due to positional deviations arising from constraints imposed by the environment.

Figure 2.6 is a bond graph reprsentation of an unloaded impedance

controlled mechanism [21]. The actuators provide torques, a function of desired impedance and equilibrium position, which then moves the links, admittances. Figure 2.6 is a bond graph of a loaded impedance controlled mechanism as well. The torques drive an equivalent admittance comprised of the manipulator and environment admittances. The relation between controller and robot has not changed for the uncoupled to coupled phases.

In summary, sections 2.3 and 2.4 intended to prove that one does not need force feedback for interface force control. If one uses force feedback, the result is in general a modulation of intrinsic manipulator dynamics. Impedance control is not an arbitrary strategy; the motivation behind impedance control is based on underlying energetic relationships of elements in a system [21].

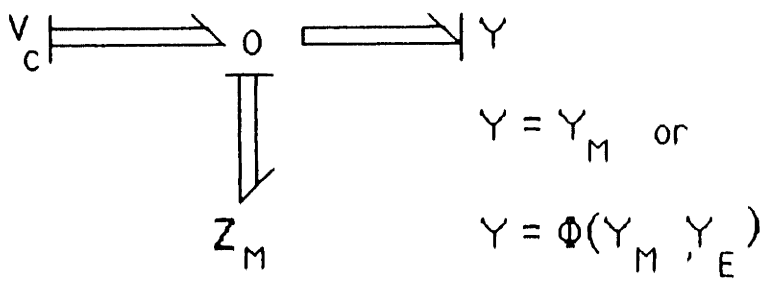
2.4.1 Some other advantages of impedance control.

Two benefits of impedance control not mentioned so far are extensive mechanism modelling and precise measurement of load inertia are not required. A force feedback impedance controlled mechanism does not depend on a highly accurate model of its open loop dynamics since the action of force feedback is to diminish the importance of intrinsic dynamics. The derivation of the torque control law in the next chapter will explain this point analytically. The second benefit is the way in which an impedance controlled device reacts to flows dictated by its load. Deviations from planned trajectories due to an imprecisely known load inertia or grip location inaccuracies are controlled by specifying the desired behavior of the manipulator.

The desired behavior considered in this work is that of a linear decoupled time invariant second order system. A second order system rather

than first order system was chosen because inertia, as well as stiffness and damping, make up the desired system impedance. A first order system, while simpler and thus more attractive, can not incorporate all three aspects of a system's dynamics.

A good understanding of second order systems exists [38], [9], [16]. One can predict responses with relative ease. A linear decoupled system should have well behaved motions. For example, the endpoint should move in a straight line as point to point commands are given. A higher order model was not chosen since it would be computationally more complex and yields no significant added flexibility to the target dynamics.



Unloaded or loaded impedance controlled mechanism.

Figure 2.6

Nonlinear Force Feedback Impedance Control Derivation

3.1 Scope of this chapter.

This chapter derives the multiple degree of freedom NFFIC control law using linear algebra. The computational complexity of the NFFIC control law is compared to other nonlinear controllers. Stability is then analyzed for one degree of freedom. Finally, force set point methods in impedance control are discussed.

3.2 Torque control law for n degrees of freedom

NFFIC can be broken down into two components. First, an explicit force feedback loop, equation 2.17, is closed around the entire manipulator from the point of interaction. The intent is to mask actual mechanism dynamics. Second, a desired behavior is then imposed in place of the actual dynamics. In this investigation the desired behavior is a decoupled, linear time invariant second order system.

The torque control law can be outlined as follows. A model of the manipulator is formulated. The model of how one wishes it to behave is then formed. An equation that relates link accelerations to endpoint, cartesian, accelerations is then derived. These three formulations are solved simultaneously to yield a single torque control law. The solution is algebraic and in closed form. No recursion is needed.

This investigation assumes the mechanism is a multilink manipulator. Reference [21] derives the equations of motion in relative joint coordinates for

such manipulators as do [43], [31], [19]. The result is repeated in equation 3.1 and is not an uncommon nonlinear formulation for robot motions.

$$I(\Theta)\,\dot{\Omega} + C(\Theta,\Omega) + V(\Omega) + S(\Theta) + G(\Theta) = \Gamma_a + \Gamma_e \qquad \text{eq. 3.1}$$

Some terms of equation 3.1 can be removed or neglected for the specific type of manipulator used in this experiment. In this investigation a planar mechanism perpendicular to the gravity vector was used; the gravity term in equation 3.1 can be removed. Joints and actuators were assumed nondissipative so no viscous or friction terms need appear. These terms can be included and do not alter the approach to the control torque solution. Simplifications do not compromise the intent of this section; to show how NFFIC can be derived. Equation 3.1 is then reduced to:

$$I(\Theta)\,\dot{\Omega} + C(\Theta,\Omega) = \Gamma_a + \Gamma_e \qquad \text{eq. 3.2}$$

The behavior we wish to impose is that of a linear time invariant second order system in cartesian coordinates (that of the tool):

$$M_d\,\dot{V} + B_d X + K_d\,(X - \dot{X}_c) = F_e\,. \qquad \text{eq. 3.3}$$

Rewriting equation 3.3 :

$$M_d\,\dot{V} = -K_d(X - X_c) - B_d\,\dot{X} + F_e$$

$$M_d\,\dot{V} = F_c + F_e \;\; ; \;\; F_c \equiv -K_d\,(X - X_c) - B_d\,\dot{X} \qquad \text{eq. 3.4}$$

Equation 3.4 says that the command force is a function of the target system impedance (B_d and K_d) and actual system deviations from equilibrium positions (X). X_c is included to provide a commanded equilibrium position.

This command force now can be an input to an explicit force feedback control loop so that the actual system will respond with the targeted behavior. The actual system dynamics will be masked by the action of the force feedback loop allowing the target dynamics to dominate. The external force acts as it did before; to provide interface force information for disturbance responce. Equation 3.4 can also be interpreted as meaning that the mechanism should look to an external load as an inertia field of our choice. Equation 3.4 embodies the target dynamics; M_d, B_d, and K_d.

External forces are ideally sensed at the point of manipulation. Stiffness (K_d) and damping (B_d) should be controllable in any direction. For a six degree of freedom mechanism K_d and B_d are then full 6 by 6 matrices. The desired inertia as seen by the environment, M_d, would have no cross product terms and have small elements to reduce inertial loading. A constant, decoupled inertia tensor is computationally simple and decouples the target dynamics. Links can be controlled independently since no dynamic coupling exists. For a six degree of freedom mechanism the target inertia tensor would take the form:

$$M_d = \begin{vmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & i & 0 & 0 \\ 0 & 0 & 0 & 0 & i & 0 \\ 0 & 0 & 0 & 0 & 0 & i \end{vmatrix} \qquad \text{eq. 3.5}$$

Two reference frames have been mentioned so far. To relate joint coordinates, relative joint angle frame, to tool coordinates, cartesian reference frame, one starts with link geometry:

$$X = L(\Theta) . \qquad \text{eq. 3.6}$$

Differentiate equation 3.6 to relate joint angular velocities to endpoint linear velocities:

$$V = \dot{X} = J(\Theta)\,\Omega . \qquad \text{eq. 3.7}$$

Differentiate again, using the chain rule, to relate link accelerations to tool accelerations:

$$\dot{V} = J(\Theta)\,\dot{\Omega} + H(\Theta,\Omega) \qquad \text{eq. 3.8}$$

where:

$$H(\Theta,\Omega) = [J(\Theta)]\,\Omega . \qquad \text{eq. 3.9}$$

Continuing with the NFFIC derivation, rewrite equation 3.4 :

$$\dot{V} = M_d^{-1}(F_c + F_e) . \qquad \text{eq. 3.10}$$

Combine equations 3.8 and 3.10,

$$M_d^{-1}(F_c + F_e) = J(\Theta)\,\dot{\Omega} + H(\Theta,\Omega) \qquad \text{eq. 3.11}$$

Rewrite equation 3.2,

$$\dot{\Omega} = I(\Theta)^{-1}[\, \Gamma_a + \Gamma_e - C(\Theta,\Omega)] . \qquad \text{eq. 3.12}$$

Insert equation 3.12 into equation 3.11 by replacing $\dot{\Omega}$,

$$M_d^{-1}(F_c + F_e) = J\{ I(\Theta)^{-1}[\, \Gamma_a + \Gamma_e - C(\Theta,\Omega)]\} + H(\Theta,\Omega) \qquad \text{eq. 3.13}$$

Equation 3.13 has mixed reference frames. One can relate forces at the endpoint to torques at the manipulator actuators using the Jacobian transformation (from the principle of virtual work [21]). Tool coordinate forces can arise from actuator torques, torques at the manipulator's actuators can originate from external forces:

$$\Gamma_a = J^t F_a \qquad \text{eq. 3.14}$$

and

$$\Gamma_e = J^t F_e \ . \qquad \text{eq. 3.15}$$

Subsitute equations 3.13 and 3.14 into equation 3.12 and expand:

$$M_d^{-1}(F_c + F_e) = J I^{-1} J^t F_a + J I^{-1} J^t F_e - J I^{-1} C + H \ . \qquad \text{eq. 3.16}$$

Solve for the actuator forces, the efforts we wish to control, and collect terms:

$$F_a = [J I^{-1} J^t]^{-1} \{M_d^{-1}(F_c + F_e) + J I^{-1} C - H\} - F_e \ . \qquad \text{eq. 3.17}$$

Now convert back to actuator torques by substituting equation 3.16 into 3.13 :

$$\Gamma_a = J^t [J I^{-1} J^t]^{-1} \{M_d^{-1}(F_c + F_e) + J I^{-1} C - H\} - F_e \ . \qquad \text{eq. 3.18}$$

Include the expression for F_c, equation 3.4, into equation 3.18 so that feedback parameters and gains can be seen:

$$\Gamma_a = J^t [J I^{-1} J^t]^{-1} \{M_d^{-1}(K_d(X_c - X) - B_d \dot{X} + F_e) + J I^{-1} C - H\} - J^t F_e \ . \qquad \text{eq. 3.19}$$

Equation 3.19 is the torque control law to implement nonlinear force feedback impedance control.

The torque control law implements linear impedance in cartesian endpoint space. At the joint level, the control torques produce nonlinear

impedance control. The nonlinearity arises from the dependance of the inertia tensor (I) on geometry, the Jacobian (J) to relate joint coordinates to tool coordinates, and the square of the joint angular velocities used in the acceleration terms.

Note that no "inverse kinematics" [40] computations are needed to specify an endpoint position. An inverse kinematics solution is not guaranteed to exist for a particular mechanism [40]; equation 3.19 circumvents this constraint. When one wishes the system to have an apparent mass, damping, and stiffness the necessary feedback gains to achieve this target behavior are comprised of physically meaningful quantities. This feature of NFFIC can make potentially complex analyses simpler. The cluster of matrices, $[J I^{-1} J^{t}]^{-1}$, is a weighted pseudoinverse of the Jacobian with the weighting matrix being the inertia tensor, $I(\Theta)$. The appearance of a pseudoinverse is merely a by product of the torque control law and was not an intended result.

3.3 NFFIC relative computational complexity.

The computational complexity of the torque control law was evaluated using a method presented in [19]. The complexity of equation 3.19 is on the order of fast, nonlinear controllers proposed in current literature. Given the speed with which these representative controllers operate, for six degrees of freedom, one can conclude that NFFIC would run at nearly the same speed. Table 3.0 compiles typical complexity data [7], in which equation 3.19 has been added.

3.4 Stability analysis.

Can any stability criteria be derived from equation 3.19? The following

Controller	Multiplies & Divides	Adds & subtracts
Full Lagrangian	66,271.	51,548.
Recursive Newton- Euler	852.	738.
Configuration Space	468.	264.
Cotter Impedance Control	≈800.	≈600.
NFFIC	1164.	1008.

Relative computational complexity.
Table 3.0

analysis is for the one degree of freedom case. In [20] and [25] impedance control stability is analyzed for the general case.

One could rederive the torque control law for one degree of freedom or, more simply, linearize equation 3.19 by standard methods [8]. By either method the result is:

$$F_a = \frac{M_a}{M_d} K_d (X_c - X) - \frac{M_a}{M_d} B_d \dot{X} - (\frac{M_a}{M_d} - 1) F_e \qquad \text{eq. 3.20}$$

where:

$$\frac{M_a}{M_d} K_d \equiv \text{position feedback gain ,}$$

$$\frac{M_a}{M_d} B_d \equiv \text{velocity feedback gain , and}$$

$$\left[\frac{M_a}{M_d} - 1 \right] \equiv \text{force feedback gain .}$$

Equation 3.20 shows the feedback gains of a linearized version of NFFIC. Suppose, however, that an element comes between the manipulator endpoint and the environment that is not an admittance. An example of such an element that is predominantly a spring with low inertia is the force transducer. It physically comes between the manipulator endpoint and the environment. For completeness, consider the transducer as having intrinsic damping as well.

The NFFIC law is closed about the tip of the mechanism and does not include the force transducer dynamics. How is the target behavior is altered, and can instability occur?

The total system behavior must then include the sensor stiffness in parallel with the desired stiffness, and sensor damping in series with the desired damping. The desired system behavior, equation 3.3, should be rewritten to include the new total values of stiffeness and damping:

$$M_d \dot{V} + B_t \dot{X} + K_t (X - X_c) = F_e \qquad \text{eq. 3.21}$$

where:

$$K_t = \frac{K_d K_{td}}{K_{td} + K_d} > 0 \qquad \text{eq. 3.22}$$

and,

$$B_t = B_{td} + B_d > 0 . \qquad \text{eq. 3.23}$$

Equations 3.22 and 3.23 are nominally always true. Thus the <u>desired</u> behavior is stable (the response may change) no matter what combination of desired system stiffness and force transducer stiffness is chosen. Real stability depends on other variables such as parameter errors. This analysis has the reassuring result that at least <u>in theory</u> an impedance controlled system will remain stable when coupled to a rigid environment through a damped, compliant transducer.

3.5 Force set point in impedance control.

In the current literature the central motivation for the use of force feedback is to regulate interface force. The usual approach is to specify a desired force set point in tool coordinates (Cartesian coordinates is this work) and back out the required joint torque by a simple matrix transformation. The advantages of this approach are that it is computationally simple and a detailed knowledge of the task geometry is not needed. The mechanism need only position itself near the object. Contact may be detected using the force sensor. Unfortunately, this approach has a fatal flaw: stability is not guaranteed and in fact is regarded as one of the central problems of robot force control [53].

In contrast, impedance control provides considerable stability robustness [20], [25]. But how can a specified interface force be maintained with impedance control? Since a deflection of the manipulator endpoint results in a predictable force, one effective way is to specify an endpoint position a distance inside the surface. Given the stiffness of the robot (and the stiffness of the contacting force sensor, if there is one) that distance is chosen to be such as to produce the desired force. On the positive side this approach preserves the stability robustness of impedance control since the desired impedance has not changed. On the negative side accurate force control requires precise knowledge of sensor and gripper stiffness. Furthermore, errors in the knowledge of the location of the environment translate directly into errors in interface force. The pro's and con's of these two approaches are summarized in table 3.1.

These last two sections brought up some aspects of NFFIC stability. NFFIC is designed to suit both unloaded and loaded mechanisms. The transition

through all three phases, as described in section 1.1, is smooth, much as a physical spring and damper might behave. An impedance controlled device can always be stable for endpoint position step inputs. Further, endpoint accuracy depends on closed loop stiffness.

These previous two sentences described a type zero system. Type zero systems are characterized by stable response to step inputs with finite error. This type of response is desirable for mechanisms that are expected to interact with their environments. An impedance controlled device could trace a path by setting up a supervisory program that specifies successive equilibrium positions, in small step increments, along the path. Chapter five details this method as it was applied to this work.

Force set point method	Pro's	Con's
A. $F_{set} \equiv$ constant $(\Gamma_{set} \equiv J^t F_{set})$	Insensitive to environ - ment geometry.	Stability not guarnteed.
B. $X_{des} \equiv F_{set}/K + X_e$ $K = \varphi(K_p, K_{td})$	Stability Guarnteed.	Needs X_e, not always known

Force set point methods.
Table 3.1

Hardware Setup

4.1 Scope of this chapter.

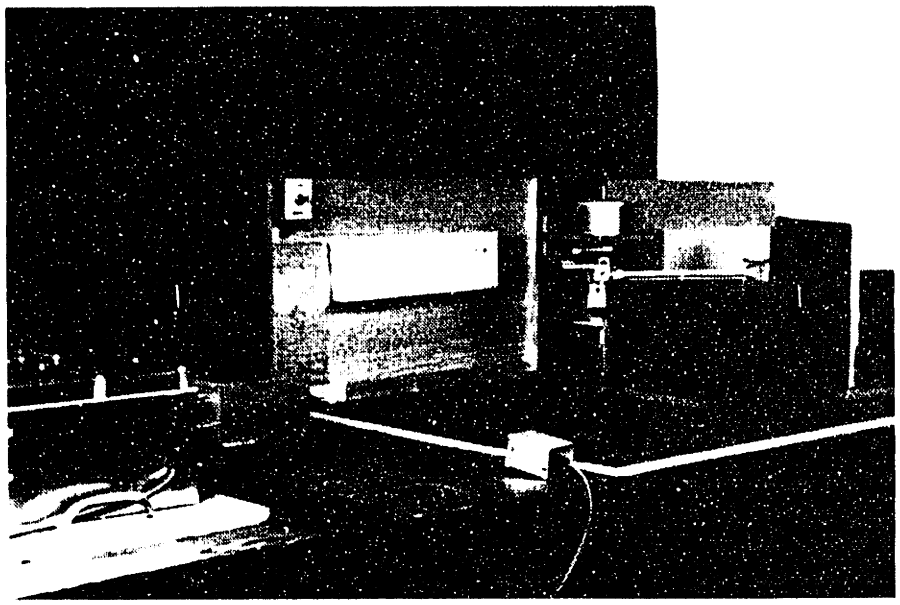
This chapter describes the mechanism used in this investigation to test impedance control. The "proof mass" is defined and constraints on its design are detailed. Electromagnetic and radio frequency interference (EMI/RFI) in the electronics of the hardware is quantified. Experiments to calculate scaling factors are explained in this chapter and in appendix I. Finally, safety issues are explored.

4.2 Why not simulate?

One goal of this investigation is to test the performance of NFFIC. When one simulates, the results are based on models. These models are are a subset of real world behavior. During the early stages of this work it was agreed that a hardware model of a mechanism would yield more insight than a simulation, for the same development time. The setup described in this chapter is a testbed for NFFIC evaluation and is not meant to perform assembly or practical manipulation. The purpose of the device is to encounter the basic forms of interaction; the three phases mentioned in the introduction. Parts of the apparatus are from a previous analysis, [7]. Lessons from [7] and [27] helped shape the design.

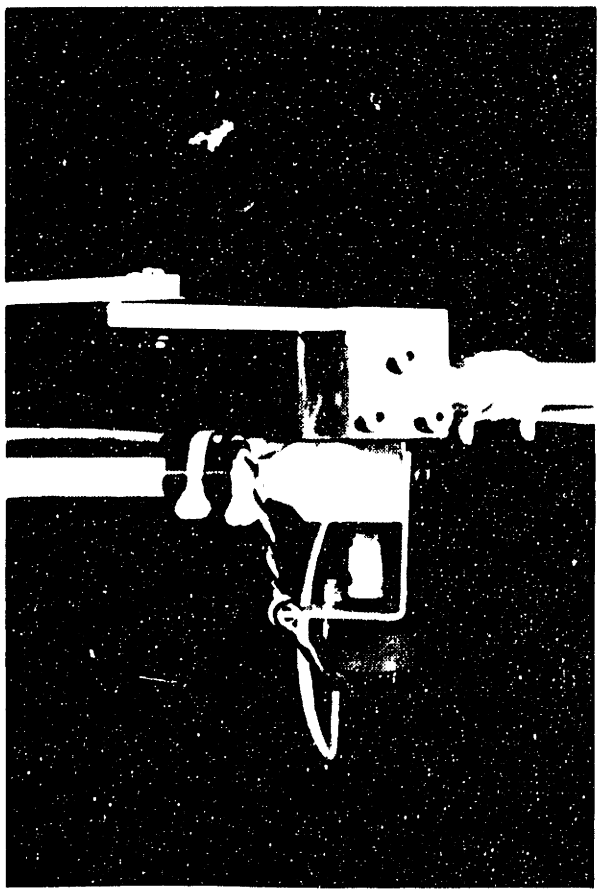
4.3 Major components of the hardware setup.

Figure 4.0 is a side view of the test setup. The two link mechanism can be seen to the right of center of the photograph. Each link is one foot long



Control algorithm testbed.

Figure 4.0



Joint #2.

Figure 4.1

from pivot to pivot and made from 2024 Aluminum. The slender links of the four bar linkage, which drive the outer arm of the mechanism, are made of 7076-T6 Aluminum. The motors to drive each link appear vertically stacked in aluminum housings against the wood backplane of the mechanism. The large cube to the right in figure 4.0 is a concrete block used as a rigid body for collisions. The mahogany face was used to mount various objects for interaction with the two link mechanism. The block is one foot square at the base and 10 inches high. Two switches are provided that are tied to the motor motor disable lines on the servo amplifiers. One can be seen as a small silver colored box on the backplane of the apparatus. The other switch, in the foreground of figure 4.0, was on a six foot line for the experimenter to hold or wear.

The low rectangular box to the left of center in figure 4.0 is a sensor signal conditioning unit built for this research. At the far left of the photograph are the two servo amplifiers with transformers.

The two links of the planar mechanism were each directly driven by DC torque motors. The motors are current supplied by pulse width modulated servoamplifiers. Relative joint angles are sensed by a rotary potentiometer located below each of the two joints. The joint between the two links consists of a vertical pin on which 4 bearings rotate. Two bearings are in each link. The outer link is driven through a rigid parallelogram linkage. Servo amplifier and motor data are listed in Tables 4.0 and 4.1.

Figure 4.1 is a close up of joint #2. Note the potentiometer under the joint. A collar joins the potentiometer shaft and the joint pin. One can mechanically null the potentiometer for any link configuration by loosening the collar at either end and rotating the potentiometer shaft or joint pin to the

Description	Link #1	Link #2
Manufacturer and type	PMI SSA-54-10-30	PMI SSA-40-07-20
Continuous current (amps)	±10.0	±7.0
Peak current, .5 sec. (amps)	±30 @ ±54 VDC	±20 @ ±40 VDC
Switching frequency (Hz)	5.0 K	5.0 K
Transconductance gain (amps/volt)		
Input channel #1	150 thru 1500	100 thru 1000
Input channel #2	75 thru 750	50 thru 500
Input channel #3	38 thru 380	25 thru 250

Servo Amplifier Data.
Table 4.0

Description	Link #1	Link #2
Manufacturer and type	PMI U12M4H	PMI U9M4H
Torque (oz·in)	186.0	80.6
Peak torque (oz·in)	2110.0	727.0
Current (amps)	8.02	8.61
Peak current (amps)	84.5	72.0
Torque constant (oz·in/amp)	25.0	10.2
Rotor inertia (oz·in·s^2)	0.021	0.0058

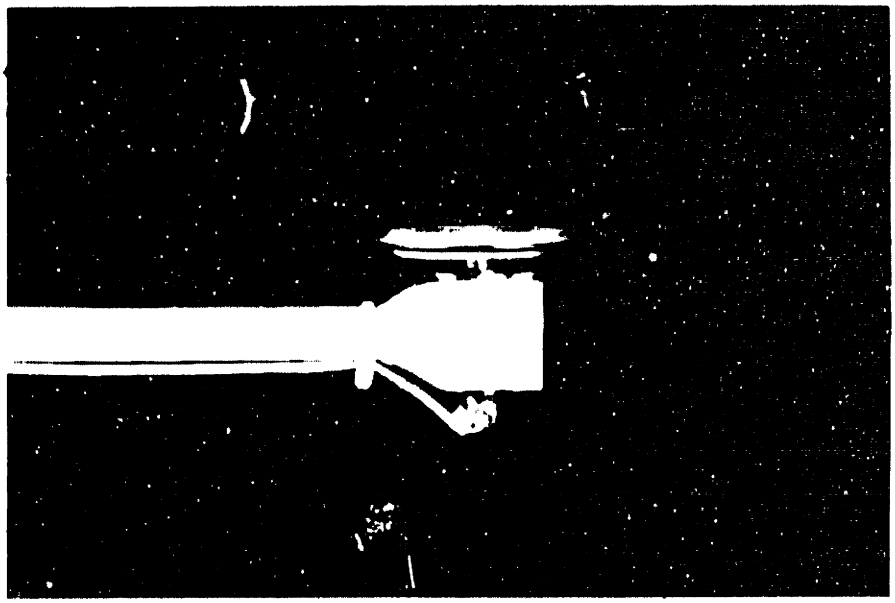
Motor Data.
Table 4.1

desired setting. A similar setup for joint #1 mates a potentiometer with the shaft of the motor driving link #1.

Figure 4.2 is a side view of the end of link #2. To the the right of the photograph, mounted vertically through the end of the outer link, is the force transducer. The thin disk on the end of the stalk of the force sensor is decribed in more detail in the next section. Figure 4.3 gives manufacturers data on the force transducer. The force transducer is actually a commercially available two axis joystick. Deflections at the tip of the stalk produce bipolar voltages.

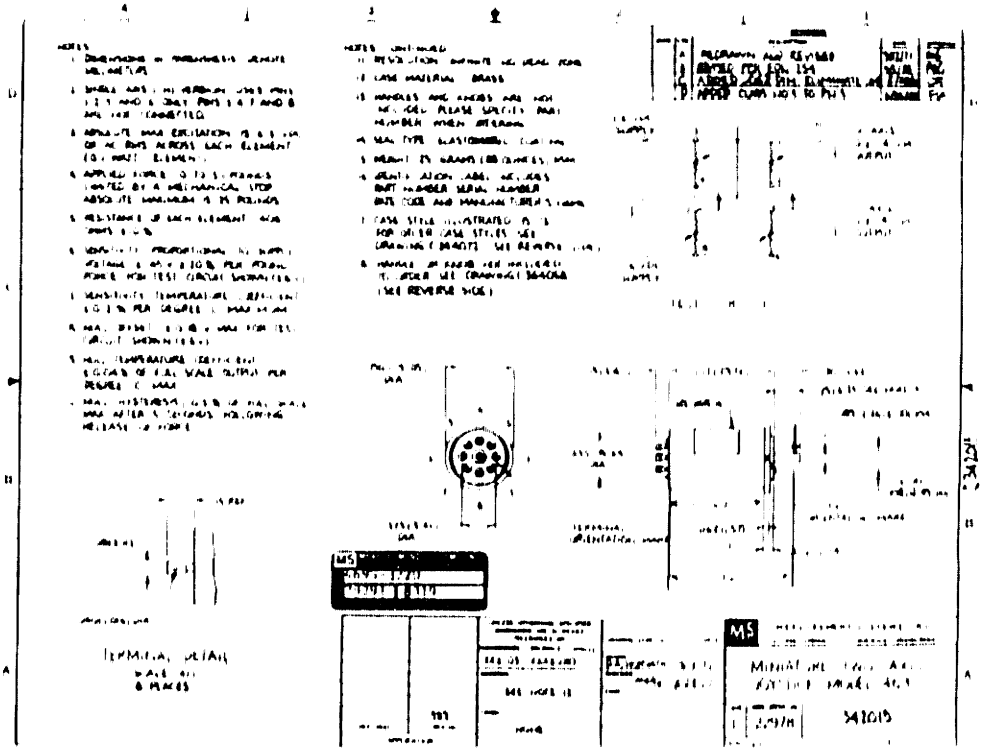
The feedback signals were voltage scaled and filtered in the signal conditioning unit shown by figure 4.4. Each channel consisted of a summing circuit, to null any sensor offset, a voltage amplifier to scale the sensor signal, and an active single pole filter to limit aliasing [5] of digitally reconstructed signals. Each single pole filter was set at a break frequency of 80 Hz. The break point was set high enough so as not to cause significant phase lags for signals around 10 Hz, the largest estimated bandwidth of the digital control schemes. The signal conditioning circuit was repeated four times to accomodate the two joint potentiometer signals and the two force transducer signals (one for the x axis and one for the y axis). The cicuit was constructed from 1/4 watt 5% resistors, tantalum capacitors, and operational amplifiers (μA747).

A Digital Equipment Corporation LSI PDP 11/23 digital computer was used to implement the software control algorithms. Control algorithms used in this investigation are detailed in the next chapter. An I/O board was constructed by two graduate students, Cary Abul-Haj and William Murray. It makes available for the user and computer a real time clock, Schmitt triggers,



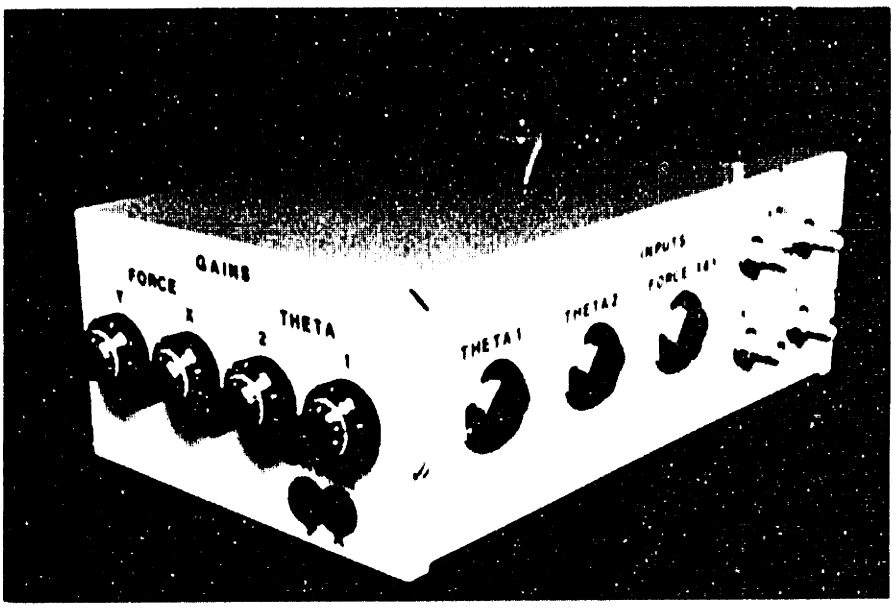
Force sensor with proof mass.

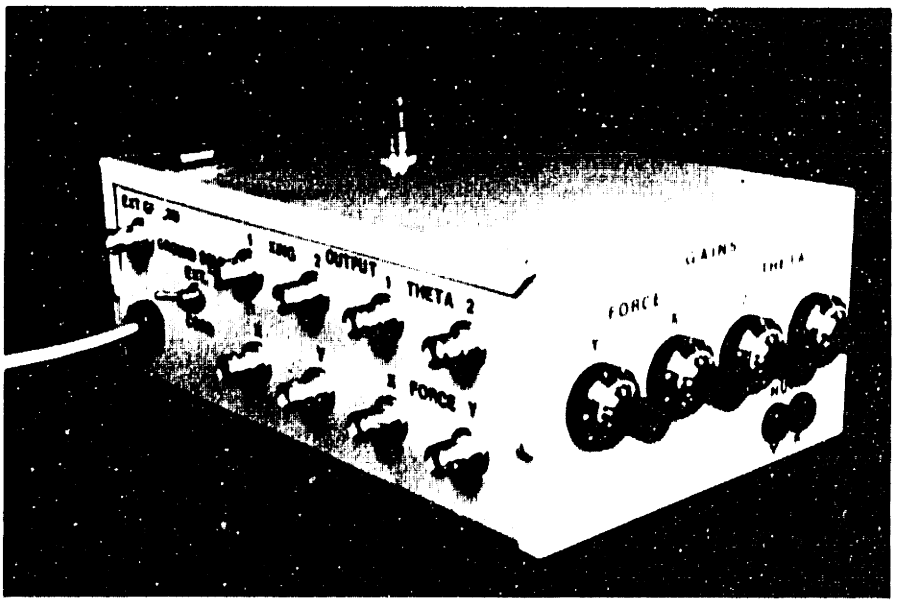
Figure 4.2



Joystick data sheet.

Figure 4.3





Signal conditioning unit.

Figure 4.4

a 16 channel analog to digital converter (A/D), 4 digital to analog converters (D/A's), and 16 ports each of TTL input and output. The A/D and D/A's have 12 bits of resolution for ±10 volts range. The 11/23 has 16 bit integer and 32 bit hardware multiply and divide capability. One can program in FORTRAN IV and MACRO-11, the 11/23 machine code. MACRO-11 features multiple addressing modes, five usable registers, and a mnemonic form of code including add, subtract, multiply, and divide [DEC manual]. I/O panel devices appear as addresses which were easily accessed under program control.

A software means of disabling the motors was constructed to back up the motor disable switches provided for the user. An open collector hex inverter was configured to pull the amplifier motor disable line low on command from a TTL output from the computer.

How an external load would couple to the force transducer was given consideration. The force sensor was located at the tip of link #2 in order to get the most accurate reading of interface force. If one measured motor current to back out forces at the tip, the dynamics of the apparatus would certainly influence the force signal. Interactions between environment and force sensor stalk was by a disk called a "proof mass" in this work.

There were three constraints on the proof mass size. First, it should act as a link between the transducer and the outside that can rotate with low friction as the mechanism endpoint tracks an edge. The O ring bumper on the edge models a gripper's finger pads. The O ring can be removed for metal to metal contact to simulate a tool.

A second constraint on the proof mass size is to keep it small and symmetric. That way a sensitive force measurement is possible by not "loading" the object to be manipulated. Recall all manipulator dynamics behind

the force transducer are wiped away by force feedback action. Any dynamics inherent to the force sensor remain. In effect, this constraint dictates no proof mass at all.

The third constraint on the proof mass was to keep the force transducer active during uncoupled moves. The proof mass thus turns the force sensor into a accelerometer. Given force sensor stiffness and maximum expected joint angular accelerations one can back out a proof mass size that provides a maximum force signal for a maximum endpoint acceleration.

These three constraints were in opposition. A sensitive force transducer needs a small proof mass whereas a sensitive accelerometer (in this specific case) requires a large proof mass. Additionally, too large a proof mass might oscillate on the joystick at a frequency that will excite the controller. These constraints are explored in the next section.

4.4 Proof mass sizing.

The basic geometry of the proof mass was chosen by its constraints. A one inch disk will contact any surface before the physical end of link #2 given the location of the force transducer. A round shape ensures a symmetric mass distribution in the plane of the two link mechanism and allows for rotation to track an edge. Only the thickness need be determined based on the the mass chosen.

The mass of the disk was chosen by calculating an estimate of the maximum joint angular accelerations to be expected and back calculating the inertia needed to deflect the force transducer to full scale but not beyond. A check of the natural frequency of the proof mass on the joystick was calculated to ensure the oscillations would be well above approximately 5 Hz.

Two estimates are involved in this frequency check: maximum joint acceleration and controller bandwidth.

The NFFIC needed hardware to run, but the hardware incorporated the proof mass, so to estimate endpoint acceleration manufacturers' motor and amplifier data was used. From [39] the manufacturer recommends calculating motor acceleration by:

$$\alpha_{mp} = \frac{\Gamma_{mp}}{J_l + J_m} \quad . \qquad \text{eq. 4.0}$$

The links are the load. Link inertia calculations will be covered in the next section. Equation 4.1, and all subsequent equations in this analysis, were solved for motor #1 and servo amplifier #1 since they are the more powerful of the motor/amplifier pairs. The inertia needed to deflect the force sensor to its maximum amount was calculated by:

$$J_{pm} = \frac{\Gamma_{tdm}}{\alpha_{mp}} \qquad \text{eq. 4.1}$$

where:

$$\Gamma_{tdm} = F_{tdm} L_1 \quad . \qquad \text{eq. 4.2}$$

To get the proof mass rotational inertia transformed to an axis perpendicular and through its center, use the parallel axis transfer theorem:

$$J_{pmo} = J_{pm} - M_{11} L_1^2 \qquad \text{eq. 4.3}$$

where the proof mass inertia for a disk is:

$$J_{pmo} = \frac{1}{2} M_p R_p^2 \quad . \qquad \text{eq. 4.4}$$

Now one could solve for proof mass mass and proceed to solve for the necessary proof mass thickness. Knowing the density of 2024 aluminum:

$$M_p = \rho_{al} V = \rho \pi R_p T_p \qquad \text{eq. 4.5}$$

Two proof masses were made based on equations 4.1 through 4.6. One is designed to deflect the force transducer full scale, called the "heavy" proof mass, and the other only half scale, called the "light" proof mass. Given the estimated values used in this design, it was considered prudent to have at least two proof masses for experimental use.

A second estimate of proof mass size was based on an approximation of controller bandwidth. A 12 millisecond sampling period was achieved by a form of nonlinear impedance control in [7] that ran in a combination of FORTH and FORTRAN. From prior experiences the author believed a NFFIC sampling period of 5 to 10 ms was plausible using MACRO-11. A conservative controller bandwidth range was estimated by [40]:

$$BW = \frac{SF}{20} \quad . \qquad \text{eq. 4.6}$$

The result being 5 to 10 Hz.

The light and heavy proof masses were then checked for their natural frequency when fixed to the joystick. A reasonable model for this setup is a mass and spring ;

$$\omega_{pm}^2 = \frac{K_{td}}{M_p} \qquad \text{eq. 4.7}$$

For the parameters involved, the light proof mass has a theoretical undamped natural frequency of 118 Hz, 10 to 20 times the controller bandwidth, and the heavy proof mass has a natural frequency of 81 Hz, 8 to 16 times the theoretical controller bandwidth. These margins were judged sufficient.

4.5 Hardware characterization.

Manufacturer's specifications for constants, such as amplifier transconductance gain, were measured under ideal conditions. Individual products can vary from the ideal product tested. The electromechanical components in this study were tested for such constants so that the inevitable scaling factors needed for control could be calculated as accurately as possible. Appendix I details three important empirical results. The potentiometer linearity was confirmed, force transducer stiffness verified, and the scaling factor of the servo amplifier command voltage to the endpoint output force was determined. These empirical tests taught a great deal about the hardware. Some constants deviated from their specified values by as much as 25%. Secondary results included scaling factors not provided by the manufacturers. By acquiring an intimate knowledge of the hardware used in this investigation more concrete conclusions on controller performance could be made.

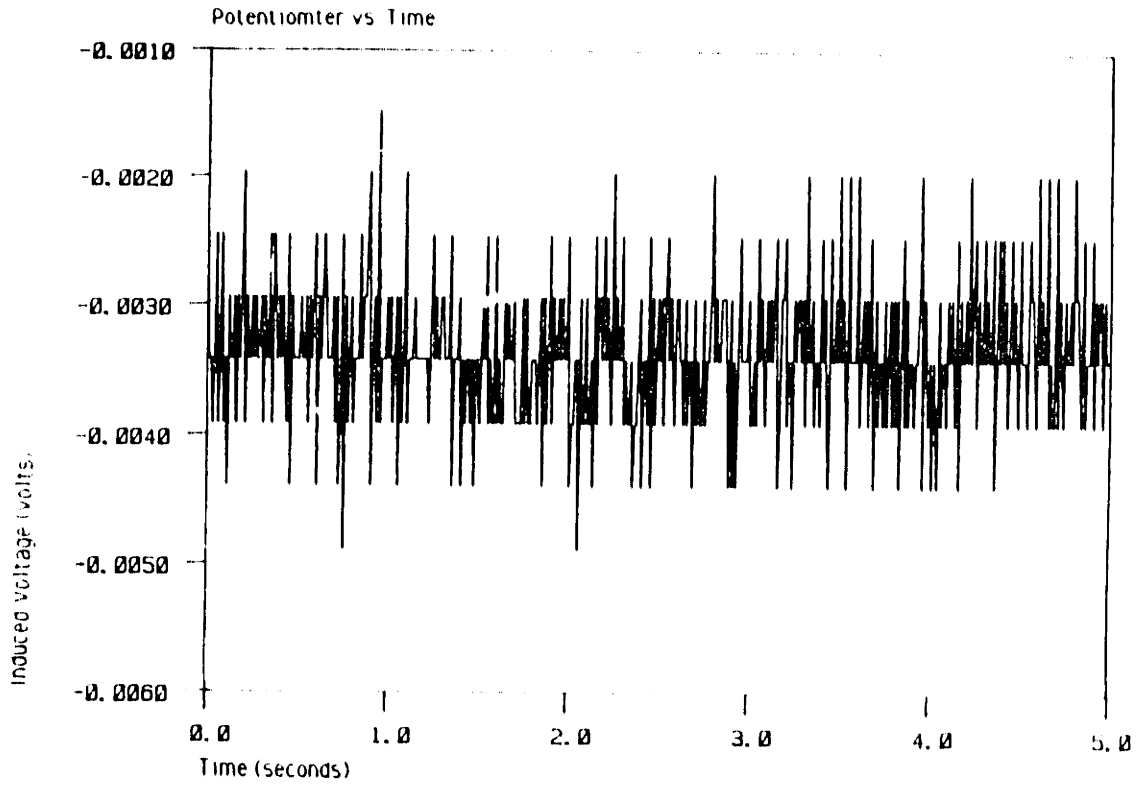
Eliminating voltages induced by EMI/RFI at the experiment site was no trivial matter. A preliminary test before grounding and shielding showed ±12 mV at 60 Hz from a active potentiometer. The 60 Hz noise most likely

originated from power cables in the lab and from fluorescent lighting. Given the A/D resolution, in Appendix II, this kind of noise could result in a false position variation of ±0.1 inches for some link geometries. Trying to filter the 60 Hz noise would have been difficult. To get the attenuation needed at 60 Hz, a single pole filter set to break at 30 Hz or a multiple pole filter (Bessel, Chebychev) set at 50 Hz yield phase lags on the order of 20 degrees for signals as low as 10 Hz; the range of expected perinent data.

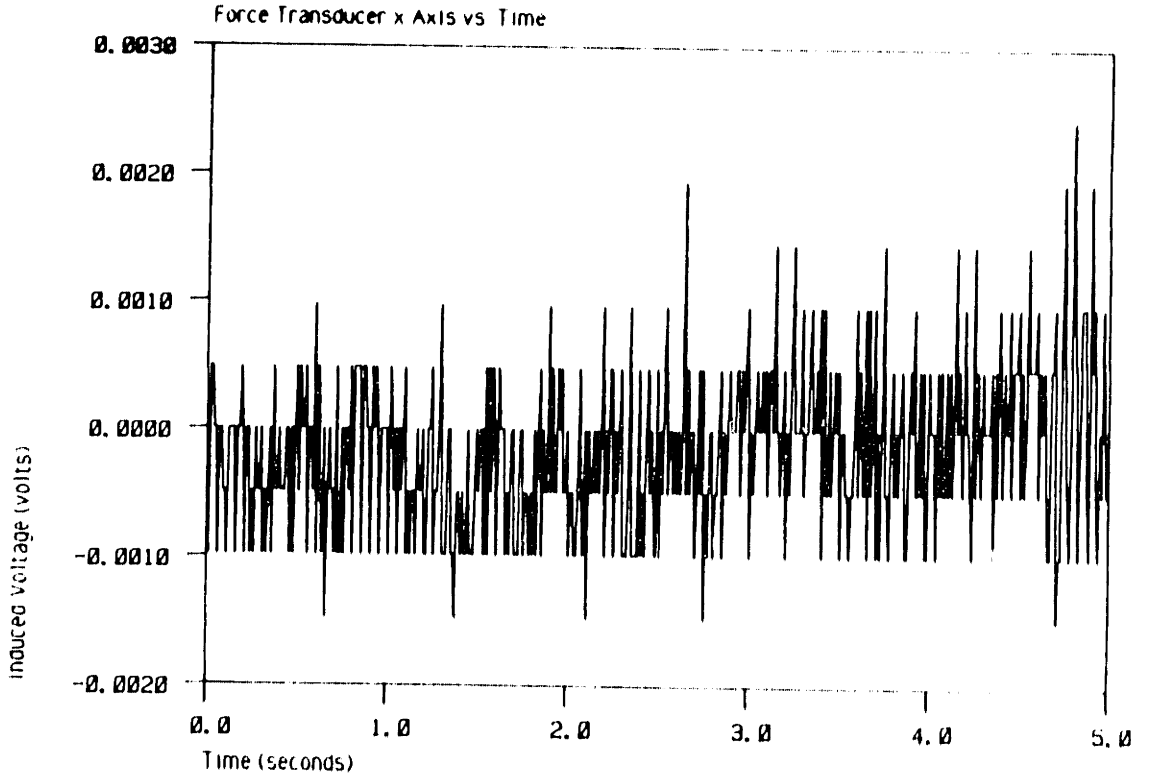
Another approach was to twist pairs of wires and shield anything electrical in sight. The result was a noise reduction to ±1mV at 60 Hz, which was considered acceptable. Digitized plots of induced voltage at the force transducer and potentiometer, figure 4.5 show the maximum amplitude of EMI/RFI after shielding.

Voltages induced in the computer were another problem At the A/D channels spurious voltages on the order of ±2 mV were common, figure 4.x. This noise can't be analog filtered. No attempt was made at grounding in a manner different than the existing one.

Figure 6.7 shows what should be a straight line from A to B. It was made by commanding the mechanism to trace a circle with a flat face of the rigid body obstructing part of the trajectory The software to complete this move is covered in the subsequent chapter. The actual joint angles from the signal conditioning unit were recorded on magnetic tape. These joint angle signals were then played back through a two pole Chebyshev filter set to break at 30 Hz before being digitized. The filter was intended to attenuate spurious signals and limit aliasing of the meaningful data expected to occur below 10 Hz. The digitizing program could sample at several hundred hertz, depending on the length of sampled data. Data played back on magnetic tape was typically



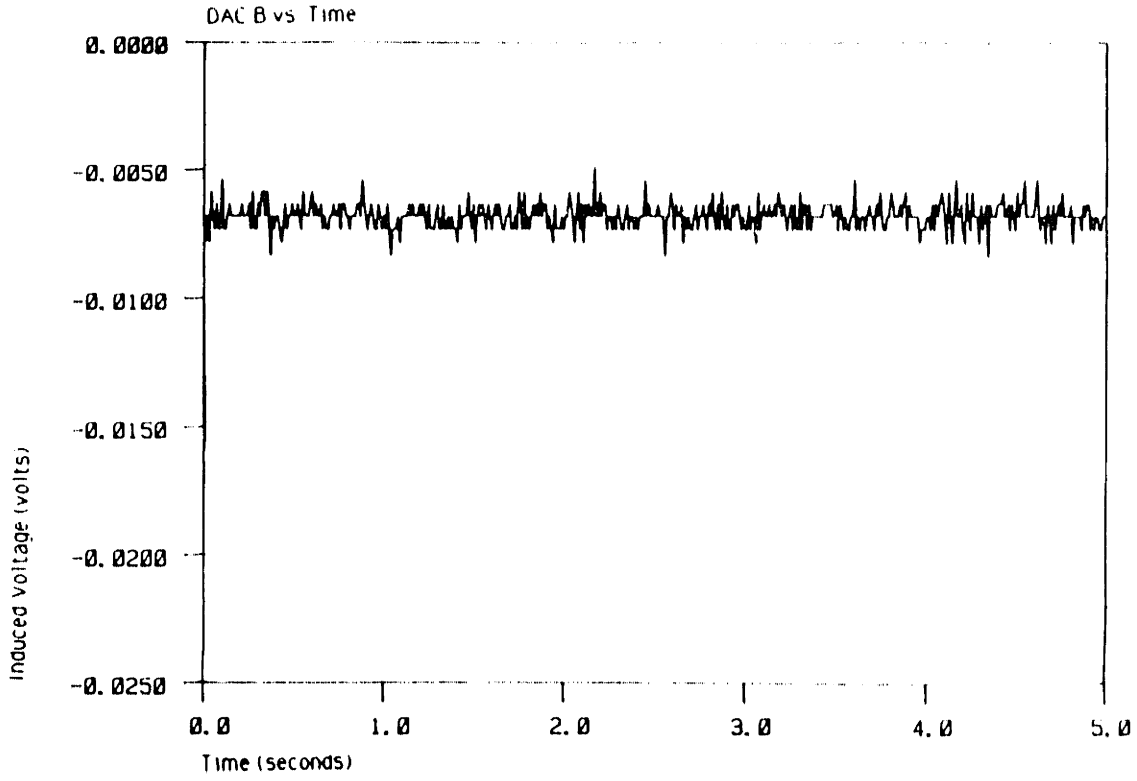
1.



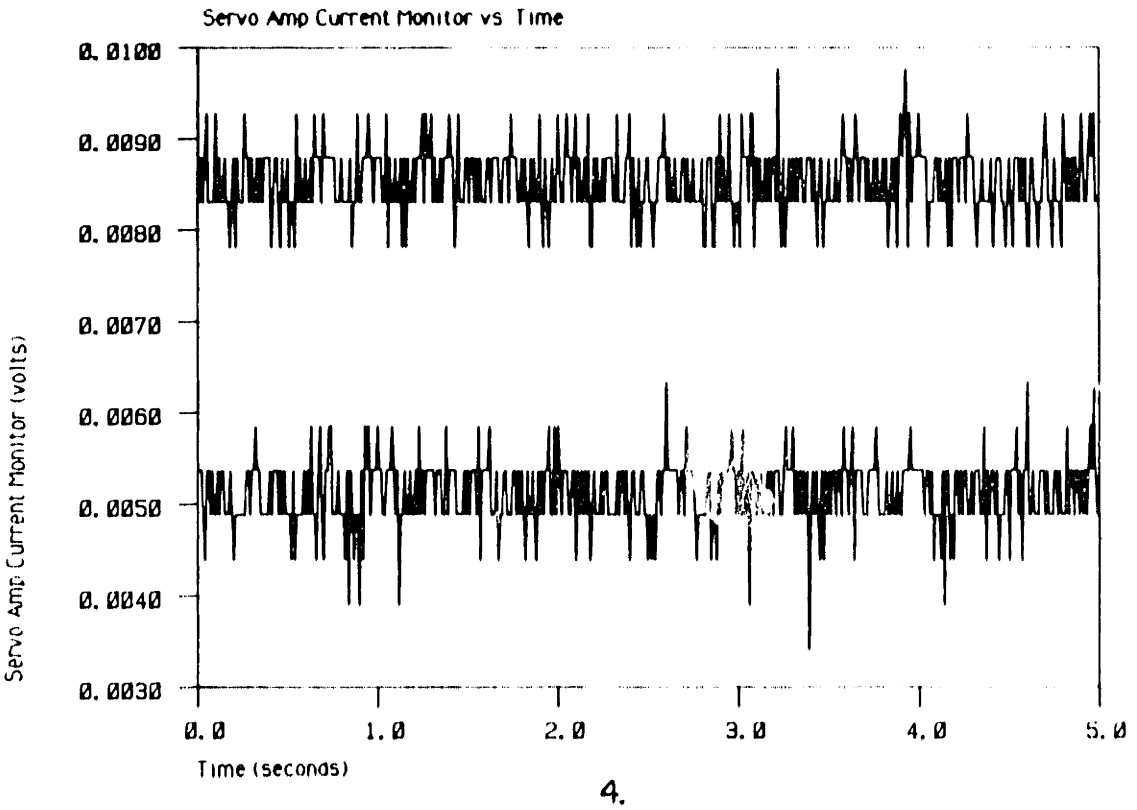
2.

Induced voltages in some components of the apparatus.

Figure 4.5A



3.



4.

Induced voltages in some components of the apparatus.

Figure 4.5B

sampled from 100 to 200 Hz. The digitized data was then processed by a routine created for displaying results in this investigation. The processed data, in the form of joint angles, endpoint trajectories, and command signals was sent to a plotting program available for general use. The high frequency spikes on what should be a straight line are most likely a digitization errors presumably from A/D noise. In data presented in later chapters these spikes appear again and are not part of actual endpoint trajectory traces. Link inertias were calculated by the sectional method [17]. Inertias of the force transducer, proof mass, and potentiometers were added to the appropriate links. As a order of magnitude check to the final results the link inertia in [7], found by empirical means, was compared to the sectional results since the links in both cases had comparable geometry. An addition check was made by assuming the link is a slender rod and calculating a rotational inertia based on actual mass. The results of this comparision for link #1 and link #2 are shown in table 4.2. From these results the sectional method to determine link inertia was deemed sound. It is interesting to note that despite the added effort to determine the link inertias by sectional means, the slender rod approximation remains a very good estimate.

The inertias of the four bar links were also added to the inertia tensor of the mechanism model, equation 3.19. Exactly how the four bar added to the inertia tensor is covered in the next chapter. The masses and centers of gravity of the four bar links and of link #1 and #2 were measured directly to yield the best possible estimate of the actual system inertia tensor.

4.6 Safety issues.

It became clear that while the links are lighweight, their ends could

Method	Rotational Inertia (lbm in^2)	
	Link #1	Link #2
Slender Rod	25.3	30.0
Empirical [Cotter]	26.0	34.0
Sectional	25.5	33.7

Link inertia estimates.
Table 4.2

move very fast. Severe injoury could result if the mechanism were to strike a person while performing such fast moves. Measures were taken to prevent accidents. None occurred.

A yellow border on the table on which the apparatus was set extended beyond the movement envelope of the mechanism, see figure 4.0. Frequently, during all computer programs, a message came on the screeen to remind the operator to stay clear of the yellow line. All digital controllers checked for commanded endpoint positions outside of the allowed workspace and advised the user to correct the command. No allowable endpoint position resulted in the mechanism interfering with the backboard of the apparatus.

Two motor disable switches, one on the apparatus backboard and one worn or held by the experimenter give the experimenter control over movements. The linear digital controller, explained in the next chapter, checked for command voltage saturation and joint angle limits. Two inch thick compliant foam pads were put in place on the backboard to protect the mechanism should it move in an uncontrolled manner. Another pad was located on the tip of the mechanism and could be removed when force transducer interactions were desired.

Controller Implementations

5.1 Scope of this chapter.

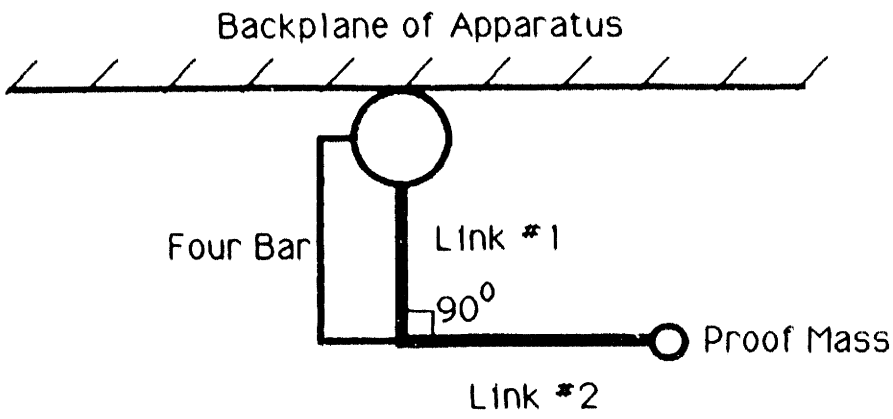
This chapter describes the three control strategies implemented in this investigation. The motivation for three control strategies is explained. Issues specific to digital implementation are reviewed. The elements of the matrices in the NFFIC law that are unique to the two link mechanism used in this thesis are described in this chapter and in appendix IV.

5.2 Why 3 control strategies?

The two control schemes besides NFFIC, analog position/force (PF) feedback and digital linear position, velocity, and force (PVF) feedback, were conceived as a means of testing NFFIC should stability problems arise. Software errors that result in unexpected behavior in NFFIC might be detected by running NFFIC in a nearly linear region (see figure 5.0 showing the mechanism's "start" position) and then running the digital linear PVF controller in the same position with commensurate gains.

Digital sampling phenomena, conjectured as causing NFFIC performance errors, could then be checked by running the analog position and force (PF) feedback controller. In this case, NFFIC would be configured so velocity information was nulled (zero damping gain), providing for a basis of comparison with analog PF control.

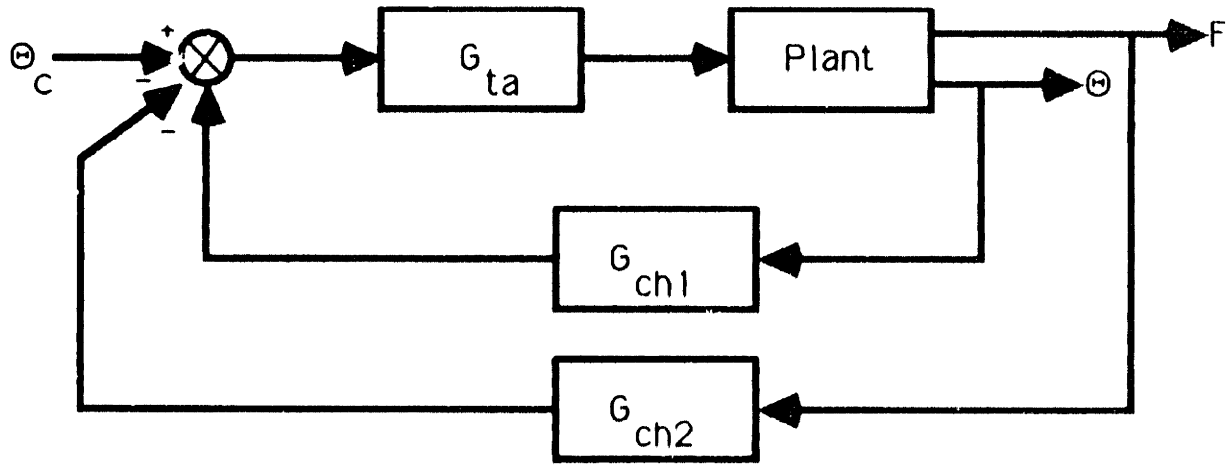
The main point intended by implementing the two other control schemes was to isolate NFFIC performance should undesired behavior show up. The hardware setup was characterized so that any aberrant behavior could be



Top View

Two link mechanism in the start position.

Figure 5.0



Analog position and force feedback control.

Figure 5.1

isolated as NFFIC or hardware related.

5.3 Analog Position and Force Feedback.

Figure 5.1 is a block diagram of this scheme for one link. Each link had its own independent PF controller. Feedback gains were adjusted by the analog input channels on the servoamplifiers.

An empirical procedure was used to ensure commensurate feedback gains when testing the NFFIC algorithm against either analog PF control or digital linear PVF control. For comparison purposes the the effective damping (and hence the velocity feedback gain) was zero for all controllers. First, a desired stiffness and mass was chosen using the NFFIC algorithm. Then one of the other two controllers was set up. Position feedback gain was increased to the point that the force transducer gave the same reading for the same deflection from equilibrium as it did under NFFIC control. Force feedback gain was then increased to the point that the experimenter judged (by sense of touch) was comparable to NFFIC control. This position feedback gain was then adjusted by again deflecting the mechanism mechanically and recording the force output. If position and force feedback gains had been chosen correctly force traces from NFFIC and the other controller had the same amplitude. This procedure was reiterated until a satisfactory match was obtained.

5.4 Digital control related issues.

One such digital issue was that of coordinate frames. For this apparatus, three frames existed. Figure 5.2 is a top view of the mechanism slightly moved from the start position. The first frame was the absolute cartesian reference frame whose origin was at the motor's shafts. Its axes

are labelled X1, X2. The second coordinate frame was the absolute joint angle frame Θ_1, Θ_2. Absolute joint angles must in turn be derived from relative joint angles. Absolute joint angles were measured from the X1 axis with counterclockwise being the positive sense. Relative joint angles were measured from the start position with the same sense. The relation between absolute and relative joint angles was:

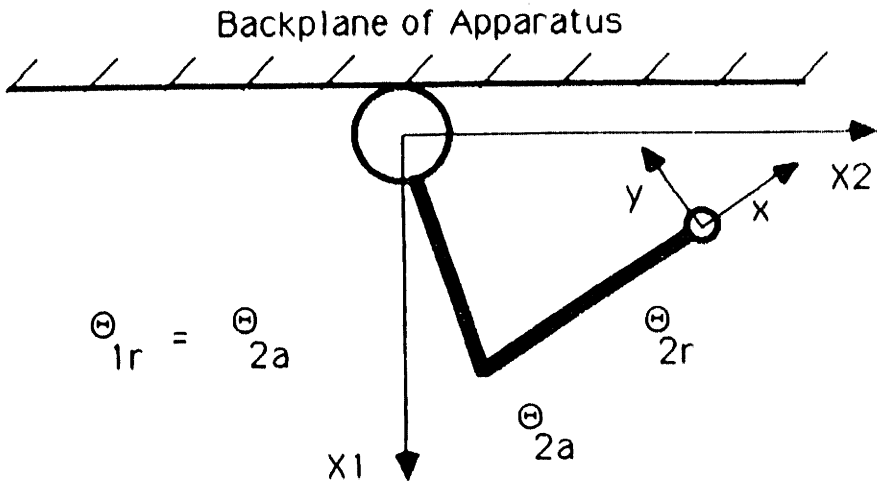
$$\Theta_{1a} = \Theta_{1a} \qquad \text{eq. 5.0}$$

$$\Theta_{2a} = \Theta_{1a} + (\Theta_{2r} + 90^{\circ}) \qquad \text{eq. 5.1}$$

The third reference frame was the force transducer axes x, y. To get transducer coordinates into absolute cartesian coordinates one needs a rotation matrix R [40]:

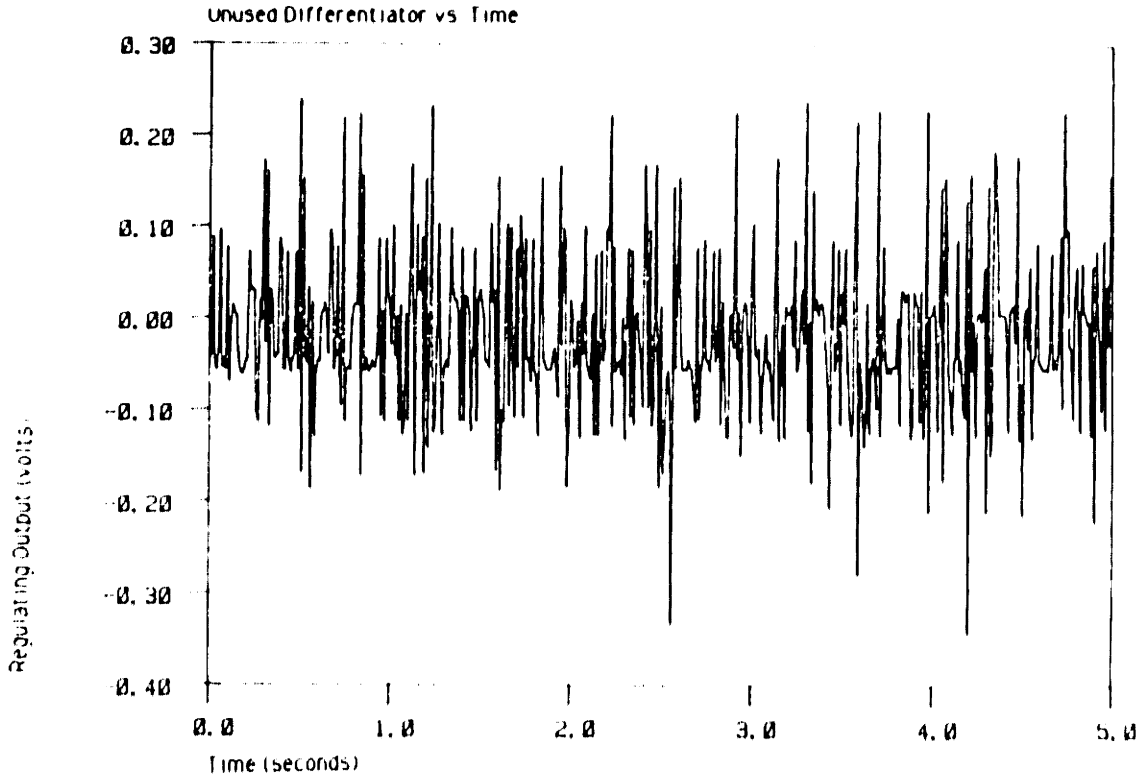
$$\begin{vmatrix} X1 \\ X2 \end{vmatrix} = \begin{vmatrix} \cos\Theta_{2a} & -\sin\Theta_{2a} \\ \sin\Theta_{2a} & \cos\Theta_{2a} \end{vmatrix} \begin{vmatrix} x \\ y \end{vmatrix} . \qquad \text{eq. 5.2}$$

Another issue was that of relating joint angles to endpoint absolute cartesian coordinates. Using figure 5.2 again, note that (X1,X2) can be related to $(\Theta_{1a},\Theta_{2a})$ by:



Two link mechanism reference frames.

Figure 5.2



Unused differentiator.

Figure 5.3

$$X = L(\Theta)$$

$$\begin{vmatrix} X1 \\ X2 \end{vmatrix} = \begin{vmatrix} L_1 \cos\Theta_{1a} + L_2 \cos\Theta_{2a} \\ L_1 \sin\Theta_{1a} + L_2 \sin\Theta_{2a} \end{vmatrix} \qquad \text{eq. 5.4}$$

Differentiating equation 5.3 produces the mechanism Jacobian:

$$V = J(\Theta)\,\Omega$$

$$\begin{vmatrix} V1 \\ V2 \end{vmatrix} = \begin{vmatrix} -L_1 \sin\Theta_{1a} & -L_2 \sin\Theta_{2a} \\ L_1 \cos\Theta_{1a} & L_2 \cos\Theta_{2a} \end{vmatrix} \begin{vmatrix} \Omega_{1a} \\ \Omega_{2a} \end{vmatrix} . \qquad \text{eq. 5.5}$$

Since link lengths were one foot apiece, and the system of measure used was $ft \cdot lb_f \cdot sec$, the link lengths can be factored out and set to unity in the digital control algorithms. For the NFFIC complexity comparison, section 3.2, link lengths were left in. With the Jacobian one could now relate endpoint forces to torques at the actuators shown by equation 5.6.

$$\Gamma_{td} = J^t F_{td} . \qquad \text{eq. 5.6}$$

Equations 5.4, 5.5, and 5.6 pointed out yet another issue. Sine and

cosine needed to be calculated in real time. Three methods were explored to do this. First, a Taylor series expansion of sine and cosine was considered. The idea was to compute the expansion using MACRO-11 in real time. A calculator program showed that to get three significant figures accuracy, the least judged necessary, one needed to expand to five terms at least.

The second method to get sine and cosine was to precompute these values in increments of .5 degrees for $\pm$90 degrees. A look up scheme was conceived that entailed only one array. Since sine and cosine are 90 degrees out of phase, the cosine of an angle could be located by subtracting the angle from 90 degrees and finding the sine. Addressing modes available in MACRO-11 could make the look up procedure dependent on registers only; a fast approach.

The third proposition was to call a FORTRAN subroutine from the MACRO-11 control program and compute sine and cosine in FORTRAN. Scaling from degrees to radians became necessary for FORTRAN compatability. Values could be shared by common blocks. This was potentially the slowest means yet the implementation in code was the least complex.

The third method was chosen for the following reasons. Calling a FORTRAN subroutine by a MACRO-11 program, converting from degrees to radians, and computing sine and cosine resulted in a 1.3 ms cycling time. Since preliminary results showed that the NFFIC program would have a sampling period under 8 ms, the 1.3 ms cycling time was considered satisfactory. The ease of encoding this approach and the accuracy afforded by floating point sine and cosine computation made the choice even better. See appendix III for a listing of a FORTRAN sine and cosine subroutine, TRGPNT.FOR for example.

Perhaps the reader has noticed the lack of tachometers in the hardware description. It was agreed that since NFFIC was to be implemented digitally an investigation into digital differentiatiopn might prove revealing. Since the differentiation schemes were in code, testing went quickly. Preliminary tests showed that for the mechanism regulating at a set point, the addition of velocity feedback resulted in seemingly random endpoint motions. It was thus inferred that the very nature of discrete differentiation caused an amplification of signal noise present in the system. The performance criterion chosen was a minimization of endpoint perturbations. Typically these motions were on the order of ±.1 inch at 5 Hz. The electronic noise in the system, see section 4.4, had been reduced enough that for zero damping gain the mechanism's endpoint did not move perceptibly while regulating at an equilibrium position with a desired stiffness and apparent mass. An implicit criterion for performance was that all differentiation schemes must be fast (less than .5 ms).

Fundamentally, discrete differentiation increases system noise and is subject to error [28], [26]. Consider a varying position signal with noise. If the sampling period is large the secant joining two positions is well determined but the slope of the secant is a poor approximation of the slope of the tangent (instantaneous velocity). If the sampling period is small, the secant is poorly determined due to signal noise, truncation errors, and sampling phenomena but its slope is a good approximation to a bad estimate of the true tangent at that point in time [26].

Command signals generated by a control algorithm, using a particular differentiation strategy, were recorded and compared to other differentiation strategies. This way one could get a quantitative measure of the relative noise

amplification of various differentiation strategies. Two numerical differentiation approaches were tried. Both were followed by averaging. The present sample estimate of the derivative was added to the previous sample's estimate with the result divided by two. In all cases this technique helped to "smooth" out differentiator noise.

One numerical differentiator was a multipoint Lagrange formula [34]]. Typical of these central difference equations was the five point formula weighted to the mid point:

$$\dot{\Phi}_2 \approx \frac{1}{2h}(\Phi_0 - 8\Phi_1 + 8\Phi_3 - \Phi_4) . \qquad \text{eq. 5.7}$$

Many variations of this theme were tried including greater and fewer pivotal points, and newest to oldest point weighting. A variation's merit turned out to be clearly distinguishable by watching endpoint motions.

The other means of numerical differentiation tried was higher order backward differences [28]. This approach is of the form:

$$\dot{\Phi}_0 \approx \frac{1}{h}(\Delta\Phi_0 - \frac{1}{2}\Delta^2\Phi_0 + \frac{1}{3}\Delta^2\Phi_0 - \frac{1}{4}\Delta^2\Phi_0 + - \ldots) \qquad \text{eq. 5.8}$$

where:

$$\Delta\Phi_0 = \Phi_1 - \Phi_0 . \qquad \text{eq. 5.9}$$

Variations on this theme included from one to five terms accuracy, in equation 5.8, and differences ranging from 1 to 4 sampling periods, in equation 5.9. The best results were from a first order, three sampling period, difference :

$$\dot{\Phi}_0 \approx \frac{\Delta\Phi_0}{h} \qquad \text{eq. 5.10}$$

where:

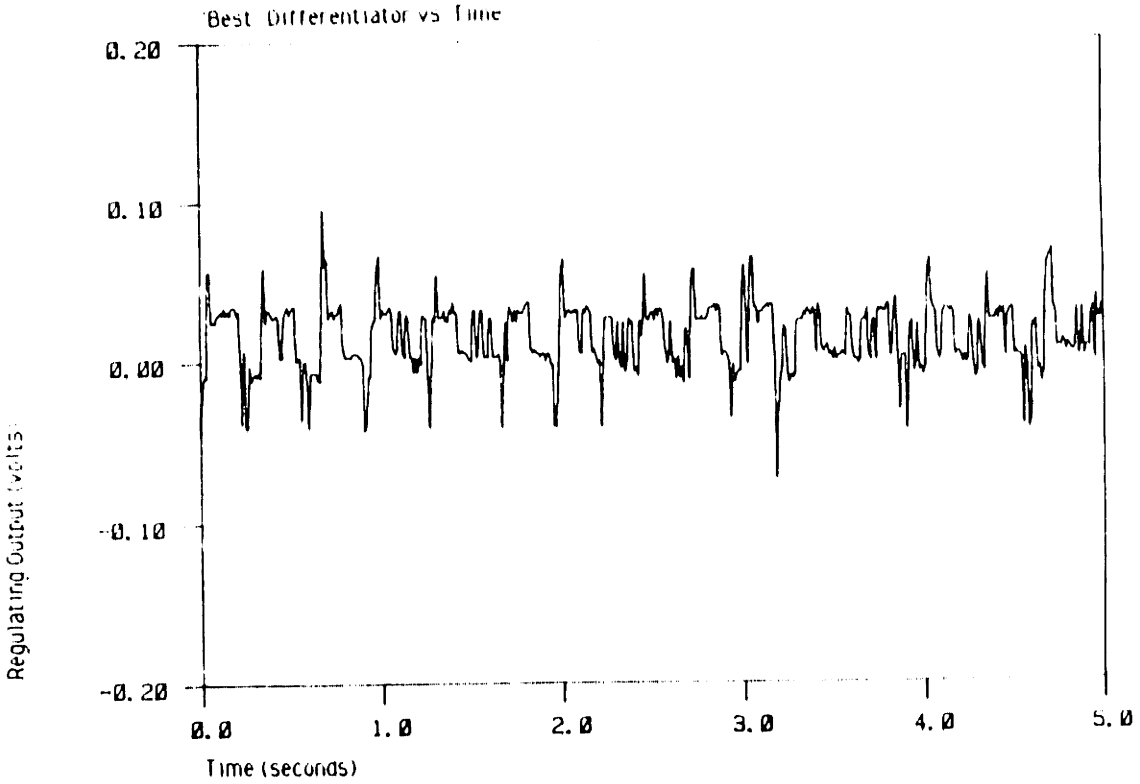
$$\Delta\Phi_0 = \Phi_3 - \Phi_0 \ . \qquad \text{eq. 5.11}$$

The command voltage comparison was measured while the NFFIC software, explained in the next section, was active. The differentiator was used on the absolute cartesian frame position of the endpoint to compute the endpoint's linear velocity. The mechanism was set up in the start position. Position and force feedback signals were grounded and the loop gains configured to be nominally zero. A velocity feedback gain was chosen that would be within a reasonable range for future experiments.

Figure 5.3 shows a representative command voltage result for equation 5.7. Figure 5.4 shows a typical command voltage result for equations 5.10 and 5.11. The first order, three sampling period, difference equation, 5.10 and 5.11, was the best choice. Qualitatively, equation 5.7 resulted in higher frequency and amplitude motions at the endpoint than equations 5.10 and 5.11.

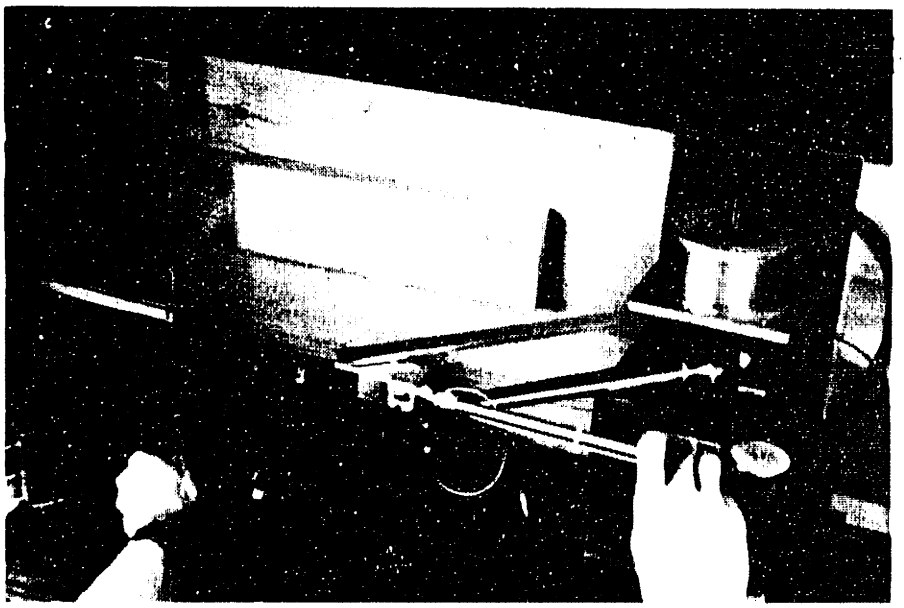
An important note here is that equations 5.10 and 5.11 represent the best of a bad situation. The endpoint was not steady and grew worse for increasing feedback gain. Typically for 1.5 lbf/ft/sec damping endpoint motions of ±.1 inch at approximately 3 Hz were noticed. Endpoint deflections were measured empirically by holding a scale to the outer link and noting relative distances travelled.

Two other issues uncovered while implementing the digital linear PVF control strategy were resolution and scaling factors. The A/D and D/A's



Best differentiator.

Figure 5.4



Joint angle calibration technique.

Figure 5.5

converted over ±10 V with 4096_{10} units of resolution. Input signals should span ±10 V to make use of maximum resolution. The signal conditioning box, section 4.3, was used to scale sensor signals so that full scale defections corresponded to ±10 V. For the potentiometers this meant ±90 degrees of swing. For the force transducer, full scale deflection along ±x or ±y should lead to ±10 V output. After conversion, these signals had to be given physical meaning again in the digital domain. All calculations done in integer arithmetic had to be scaled up by 1000 to keep three significant figures of accuracy. Appendix II contains scaling factors used in the digital controllers. The program listings, appendix IV, shows the use of the scaling factors. Scanning the NFFIC software will show that a great deal of computation involved scaling up and scaling down by 1000. One was constrained by the limit of the 16 bit words from allowing scaling factors to build upon one another. Feedback gains were also constrained by arithmetic saturation. Ultimately, the limits imposed on tests detailed in the next chapters were attributed to 16 bit integer arithmetic related. Encoding to give 32 bit accuracy, without the benefit of a 32 bit bus, was considered unnecessarily tedious.

The final issue brought up by the digital implementation of the linear PVF control scheme was calibration. Analog nulling was already discussed in section 4.3, but how was the controller to recognize its start position? Zero volts from both potentiometers was made to coincide with the mechanism's start position, figure 5.0. Counterclockwise rotations produced negative voltage, clockwise rotations resulted in positive voltage (for negative feedback since the angles were defined with opposite sign). First, since 45 degrees is one half of the 90 degrees that corresponds to 10 V from the potentiometers, 5

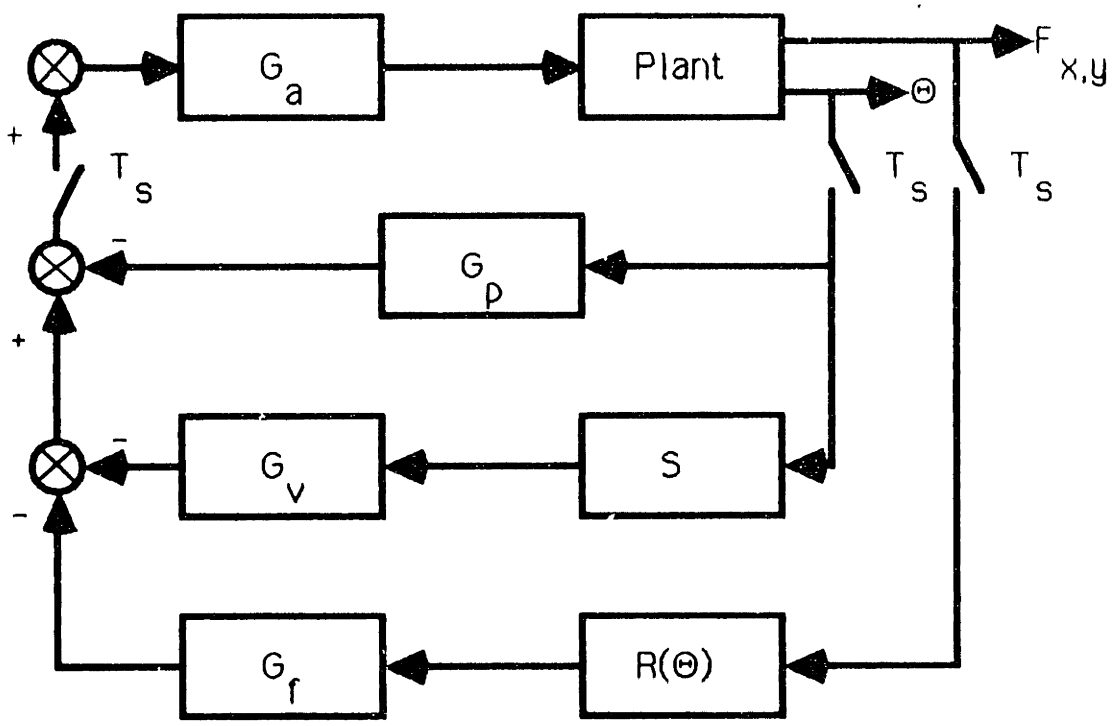
V should be seen from both joint potentiometers at 45 degress. It was found that one block held, as in figure 5.5, insured that both links were at 45 degrees. Any deviation from 5V could be adjusted by turning gain adjustments on the signal conditioning unit. Second, a block placed along an edge of the housing for motor #1 insured that link #1 projected straight out. Link #2 could be then made orthognal by lining up the square ends of each link at joint #2. Now the apparatus was fairly accurately oriented in the start position.

Early on it was noted that potentiometer calibration was very reliable but force transducer voltage drifted. Over approximately a ten minute period the force sensor varied its output by ±.01 V, well within the resolution of the A/D's. The frequency depended on environmental conditions at the mechanism site. Force sensor drift could result in a torque bias at the actuators causing deviations from ideal behavior.

Force transducer voltage drift was dealt with by incorporating a software offset null feature. Just before the control algorithm starts, a preliminary sample of force sensor voltage is stored. Subsequent samples are then reduced by the amount of the initial offset. The structure of the FORTRAN supervisory program was such that a quick cycling back to control, once control was stopped, meant that the force voltage offsets could be updated often.

5.5 Digital linear position, velocity, and force feedback.

Figure 5.6 is a block diagram of this controller. No provision was made to account for the dynamic coupling between links. The torque control law for this approach was:



Digital linear position, velocity, and force feedback control.

Figure 5.6

$$\Gamma_C = G_p \Theta + G_v \dot{\Theta} + G_f R F \qquad \text{eq. 5.12}$$

No attempt at solving for joint angles given a desired endpoint position was made. This torque control law was intended to regulate at a predetermined set point, nominally the mechanism's start position. The matrix "R" was the same as defined in the previous section.

The computer code to run this torque control law was written in three parts. A FORTRAN user interface called a MACRO-11 subprogram which computed the actual torque law in real time. A FORTRAN subroutine called by the MACRO-11 program computed sine and cosine of the current sample joint angles. The names of these three programs are PVFMNP.FOR, PVFMAC.MAC, and PVFTRG.FOR respectively. An effort was made to document the code to such an extent that the reader is encouraged to view the program listings, in appendix III.

The FORTRAN main program queried the user for parameters used to compute the torque control law. Some features of the main program, in roughly the order they appeared, were:

- safe default parameter values were provided
- the user was told what the program does and how to use it when the program set was run
- the mechanism start position was shown for sensor initialization purposes
- position, velocity and force feedback gains for both axis were requested
- the user was prompted to define a sampling rate, controlled by the computer's real time clock

- the user was allowed to start the control algorithm or return to the beginning of the program
- inputs were stored and did not need to be reentered when the main program was rerun.

The MACRO-11 subroutine in appendix III was the real time implementation of equation 5.12. All additions and subractions are within signed 16 bit ($32,767_{10}$) words. The hardware multiply and divide allows 32 bit ($2,147,459,171_{10}$) word lengths. Some features of this program were:

- variable sampling rate
- controller shutdown (with a message displayed on the screen) for:
    - sampling too fast
    - joint angle limits exceeded
    - arithmetic overflow
    - user keystrike
- variable servo amplifier command voltage limit (a message was sent to the screen warning the user that the limit had been reached)
- a 10 second delay before the controller was engaged.

The FORTRAN sine and cosine subroutine was written in floating point code. The values are passed back to the MACRO-11 control program as integers truncated after the third significant figure.

The routine's fastest sampling period was measured by toggling a TTL output port on the I/O panel of the 11/23. The resulting square wave could be measured by an oscilloscope. With all of the safety features in place the MACRO-11 control program sampled at 5 ms. Removing safety checks resulted in a 2 ms sampling period. The subprogram calls within the MACRO-11 program accounted for most of the 3 ms discrepancy.

Since each link was driven independently, only forces perpendicular to each link caused torques at the actuators. Sensed forces needed to be resolved into forces perpendicular to each link. Issues like this one were uncovered by the digital linear PVF torque control algorithm. These issues had to be solved for the NFFIC torque control law as well.

5.6 Impedance control.

The NFFIC torque control law was derived in general in section 3.3. As in [7], coriolis terms cancel out due to the choice of an absolute cartesian reference frame. An accurate description of the specific application of the general NFFIC torque control law to the mechanism used in this investigation is in the header of the program MACPNT.MAC in appendix IV. The actual system's and the desired system's inertia tensors deserve some further explanation.

The actual system's inertia tensor, equation 5.13, was derived. The choice of cartesian coordinates with an origin through the motor shafts greatly simplified the algebra. The addition of the four bar link inertias was made simpler by choosing the base for measuring centers of gravity as the origin of the cartesian reference frame.

$$I(\Theta) = \begin{vmatrix} I_{11} & I_{12}\cos(\Theta_{21}) \\ I_{12}\cos(\Theta_{21}) & I_{22} \end{vmatrix} \qquad \text{eq. 5.13}$$

where:

$$I_{11} \equiv I_1 + m_1 h_1^2 + m_2 L_1^2 + I_3 + m_3 h_3^2$$

$$I_{12} \equiv m_2 L_1 h_2 - m_3 L_4 h_3$$

$$I_{22} \equiv I_2 + m_2 h_2^2 + m_3 L_4^2 + I_4 + m_4 h_4^2 \quad .$$

Only the elements in equation 5.13 with a trigonometric function were evaluated every sampling period. The other terms were constant and appear near the top of the program MNPPNT.MAC in appendix IV.

The desired inertia tensor was constant, symetric, and decoupled. This represents an ideal inertia tensor for a manipulator. The elements reflected the desired system's mass in the X1 and X2 direction, M_{d11} and M_{d22} respectively:

$$M_d = \begin{vmatrix} M_{d11} & 0 \\ 0 & M_{d22} \end{vmatrix} \quad . \qquad \text{eq. 5.14}$$

The inverse of the desired inertia tensor took the form shown by equation 5.15.

$$M_d^{-1} = \frac{\begin{vmatrix} M_{d22} & 0 \\ 0 & M_{d11} \end{vmatrix}}{M_{d11} M_{d22}} \qquad \text{eq. 5.15}$$

Equation 5.15 was simplified to:

$$M_d^{-1} = \begin{vmatrix} M_{d11}^{-1} & 0 \\ 0 & M_{d22}^{-1} \end{vmatrix} . \qquad \text{eq. 5.16}$$

The elements in equation 5.16 were evaluated off line during the initial main program setup since they are constant. The tensor in equation 5.16 appears in the NFFIC program, MNPPNT.MAC, with different variable names.

The link's have rotational inertia which can be related to configuration dependent masses. The actual mass of the system in the plane of rotation needed to be derived so that desired masses could be picked relative to actual masses for some range of link geometries.

The problem was to relate the rotational and translational aspects of inertia. Starting with the definition for linear momentum:

$$\rho = M_a V \qquad \text{eq. 5.17}$$

and substituting the transforms:

$$V = J \Omega \quad ; \quad \eta = J^t \rho \qquad \text{eq. 5.18}$$

$$\rho = M_a J \Omega$$

$$\eta = J^t M_a J \Omega = (J^t M_a J) \Omega \qquad \text{eq. 5.19}$$

Equation 5.19 was now compared to the definition of angular momentum:

$$\eta = I \Omega \ . \qquad \text{eq. 5.20}$$

The terms in parenthesis are therefore equal:

$$I = J^t M J \longrightarrow M_a \equiv (J^t)^{-1} I(\Theta) J^{-1} \quad . \qquad \text{eq. 5.21}$$

The inverse of the actual mass tensor is defined as mobility in [21]:

$$W = M_a^{-1} = J I^{-1} J^t \quad . \qquad \text{eq. 5.22}$$

The terms on the right hand side of equation 5.22 were also in the NFFIC torque control law. These terms can be interpreted physically as an inertia transform between endpoint space and joint space. Substituting the specific matrices of the two link mechanism into equation 5.22 and solving for the links in the start position ($\Theta_{1a} = 0^0$, $\Theta_{2a} = 90^0$):

$$M_a = \begin{vmatrix} \dfrac{I_{22}}{L_2^2} & 0 \\ 0 & \dfrac{I_{11}}{L_1^2} \end{vmatrix} . \qquad \text{eq. 5.23}$$

The elements in equations 5.23 are the actual system mass in cartesian space. The effective translational mass in the X1 direction was .26 lb_m and in the X2 direction .96 lb_m. The relative sizes make physical sense. At the start position both links deflect for small movements along X2, but only link #2 deflects for small displacements along X1. Like the previous controller, the code for NFFIC was formed in three parts. The three parts were named MNPPNT.FOR, MACPNT.MAC, and TRGPNT.FOR. All three are documented and listed in appendix IV.

The FORTRAN main program served the same purpose as for the linear controller. There was no option however to set the sampling rate. Since maxiumum speed was desired, extending the sampling rate for this investigation was deemed unnecessary. The digital linear controller could always be slowed down when comparisons were being made. Some features of the FORTRAN main program were:

- illegal endpoint position check
- commands for up to five endpoint positions controlled by number keys 1-5 on the computer terminal keyboard
- desired stiffness, damping, and mass inputs
- screen display of status of user inputs during and after controller activation.

The MACRO-11 program presented some challenges. Steps taken to decrease the sampling rate are outlined in the header of MACPNT.MAC. Some preliminary calculations showed that high gains would surely saturate the 16 bit word size. An option is provided in the code that allows the user to pick between two or three significant figures of accuracy. A two significant figure accuracy requires scaling by 100 only. This way the arithmetic could be kept below $[32{,}767_{10}]$ for any gain set.

Three ways to write the matrix multiplications were investigated. One way leads to efficient program length but inefficient speed. This method entails calling a generic matrix multiplication subprogram whenever 2 matrices are to be multiplied. The repeated internal calls would lead to an inferior sampling rate. This lesson was learned from the numerous safety features in PVFMAC.MAC. The jumps in program flow would also lead to complex documentation for future researchers to follow.

Another way to compute the NFFIC torque law is to simulate what any person would do by hand. Matrices are multiplied out, element by element, and stored as temporary values for the next matrix multiply. This method was chosen. A third approach entailed multiplying all of the matrices in the torque control law symbolically and writing a single (long) line of code for each axis. This way was judged so distressful it was dropped immediately.

The NFFIC MACRO-11 program achieved a 9 ms sampling period. This was within the hoped for 5-10 ms range and was 5 ms quicker than a less complex impedance control algorithm in [7] compiled in FORTH and FORTRAN. Encoding in MACRO-11 was thus justified. When centripetal and acceleration terms were not computed the sampling period fell to 8 ms. These terms presented unique encoding challenges. They represented an additional 20% of code.

The FORTRAN sine and cosine subroutine used basically the same scheme as the digital linear case. Note some minor changes in scaling factors as made clear by the program listing in appendix IV.

Two other program sets were also developed. Each set had a FORTRAN main program, a MACRO-11 NFFIC torque law subroutine, a FORTRAN trigonometric calculation program, and a special function FORTRAN subroutine to calculate, in real time, a series of endpoints tracing out a straight line or a circle. The reason these options were not all part of one large NFFIC program set was to keep the MACRO-11 subroutine as trim as possible. Some form of decision making would be necessary in MACRO-11 to get all three program sets in one.

The circle and straight line trajectory tracing programs did not contain centripetal and acceleration terms for any of the tests described in the next

chapter. Some rough calculations based on maximum expected angular velocity suggested that these terms were small in comparision to the contribution to the command torque from stiffness and displacement. The effect of coriolis and acceleration terms can be large however for peak angular velocity and small feedback gains. In the least, the manipulator would be stable in the sense of Lyapanov [29].

The straight line program set built on the point to point program set. The MACRO-11 control portion essentially converged on a series of endpoint step commands sent it by the special FORTRAN subroutine. The program set written was MNPTRJ. FOR, MACTRJ. MAC, TRGTRJ. FOR, and SUBTRJ. FOR. The straight line trajectory was computed in real time using a parametric equation of a line. This formulation was fast and allowed for lines of vertical slope. In the nonparametric formula for a line, $y = mx + b$, vertical slope is infinity. Infinity is difficult to compute or even comprehend for that matter.

The user could also choose a point to point resolution and a delay between point updates. Movement could be stopped or started by striking the 0 or 1 keys respectively. The MACRO-11 control program could sample at 8 ms. Only the main program and special subroutine are listed in appendix V.

The circle program set MNPCYL.FOR, MACCYL.MAC, TRGCYL.FOR, and SUBCYL.FOR performed functions similar to the straight line set except that a circle of user definable radius and center was the intended trajectory. The two most important additions, the main program and the special subroutine are listed in appendix V. A fast, nontrigonometric form of the equation of a circle was used to compute the trajectory in real time. This control variation sampled in 8 ms.

These last two program sets were designed to facilitate testing and

presenting NFFIC test results. Examples are: mechanism impact with a rigid body, edge tracking, and free motion accuracy.

5.7 NFFIC software calibration.

Stiffness and damping gains under NFFIC were calibrated empirically. The apparatus was set in its start position with the proof mass just touching the rigid body. The rigid body was placed to constrain motion along the positive X2 direction.

With damping set to zero and desired mass equal to actual mass, a unity stiffness was commanded. Then incremental position commands into the barrier were given. Force sensor voltages were recorded at each step. This way known displacements gave rise to known forces and a stiffness was calculated. The assumed unity stiffness was then calibrated to the correct empirical stiffness.

Damping gains were calibrated indirectly using empirical observations. For a second order system :

$$B_d = 2\zeta\sqrt{K_d M_d} \quad . \qquad \text{eq. 5.24}$$

A critically damped system should exhibit no oscillations. To determine B_d emprically, one gradually increased the damping just to the point when oscillations no longer occur for various step inputs and a given desired stiffness and mass. The number in the software routine could then be calibrated by equation 5.24.

Dynamic Interaction Test Results

6.1 Scope of this chapter.

Tests to determine NFFIC behavior during phases two and three, as defined in section 1.1, of mechanical interactions were conducted in three regimes of force feedback. The first regime was for zero force information to the control algorithm. The second regime was positive force feedback; desired mass greater than actual mass. The third regime was negative force feedback; desired mass less than actual mass. For all of these experiments centripetal and acceleration terms were not calculated in the NFFIC software. The results presented here, and in subsequent chapters, where performed with the light proof mass only.

6.2 Experiment description.

The first test used the circle trajectory program set to command the mechanism to trace a 6 inch diameter circle centered at the mechanism's start endpoint position. A flat face of the rigid body barrier was placed about 2.75 inches into the right half of the perimeter of the circle. An arc resolution of one degree was chosen to trace the circle in approximately 3 seconds. This translates to a linear velocity of 6 in/s. A natural frequency of 1.11 Hz, unity damping ratio, and a 2.25 lb_f/ft stiffness was targeted for all of the tests.

The test was designed to yield several results. Impact response, phase two, could be explored as the proof mass collided with the barrier. Edge tracking, as the proof mass rolled along the surface, was another result.

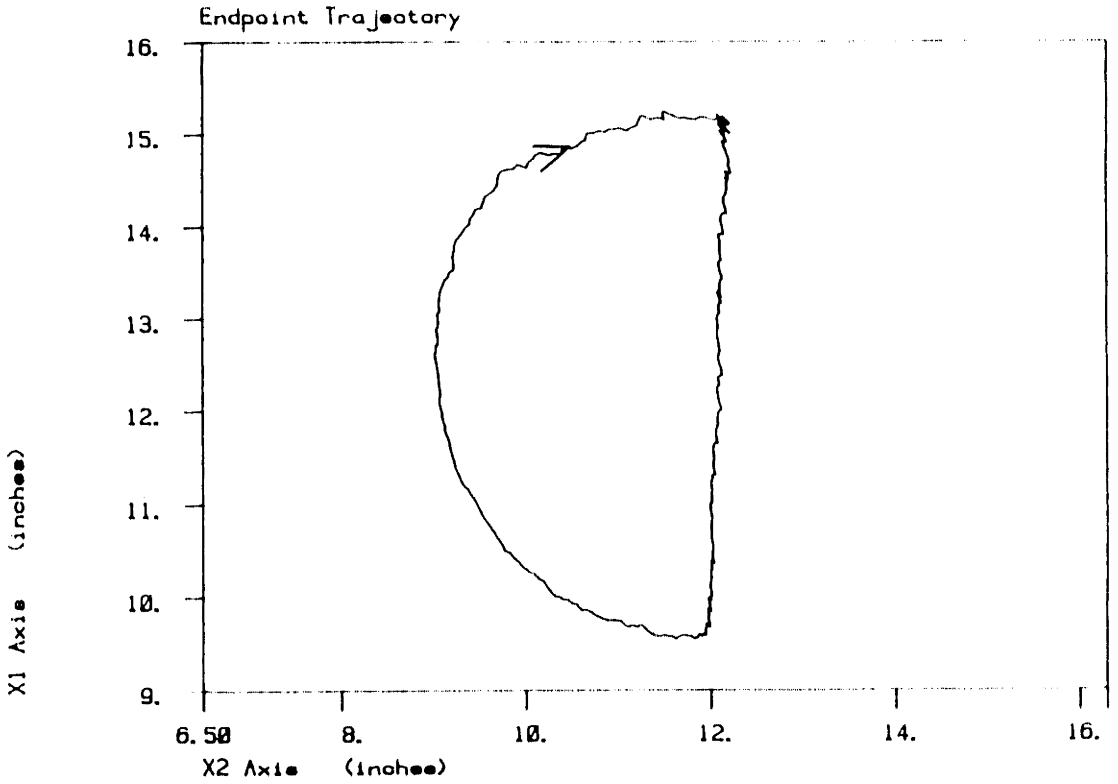
Finally, break away behavior could be analyzed as the proof mass moved away from the barrier to complete the circle. All three results were in the form of trajectory traces and total force versus time plots. Preliminary experiments were made to ensure that a large enough stiffness was chosen to give at least 25% full scale force readings.

The second test started with the barrier face perpendicular to the second link and in contact, but not forced against, the proof mass with the mechanism in its start position. This mechanism endpoint position defined point "A". Point "B" was defined as one inch perpendicular and away from the face of the barrier beginning at point "A". Letting the mechanism go from point A was a zero height impact with the surface of the rigid body. Freeing the mechanism from point B resulted in an impact with the surface The mechanism was given commanded an endpoint position two inches inside the barrier. The face of the barrier was covered with a thin sheet of aluminum and the O ring on the proof mass was removed. The intent was to simulate a metal to metal contact as might be encountered in assembly. The purpose of this setup was to test NFFIC response to impacts of different severity.

6.3 Test results for zero force feedback.

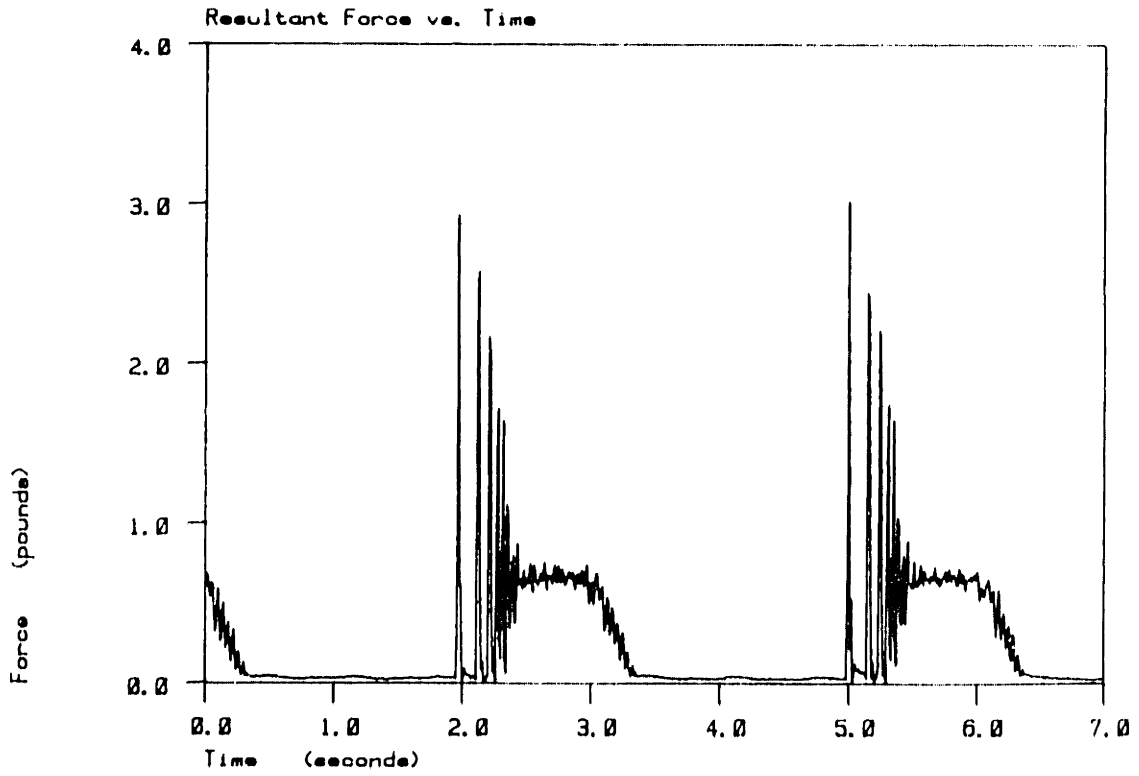
One purpose of this experiment was to compare results from the negative and positive force feedback regimes, in the following sections, to mechanism behavior without the aid of force feedback. Physically, the force feedback inputs were disconnected. In the algorithm a desired mass equal to actual mass, for the mechanism nominally in the start position, was chosen.

Figure 6.0 is an endpoint trajectory trace of the circle with barrier test. The arrowhead denotes the starting point and direction of travel. Note the



Zero force feedback, circle with barrier, endpoint trajectory.

Figure 6.0



Magnitude of the normal force for the zero force feedback test.

Figure 6.1

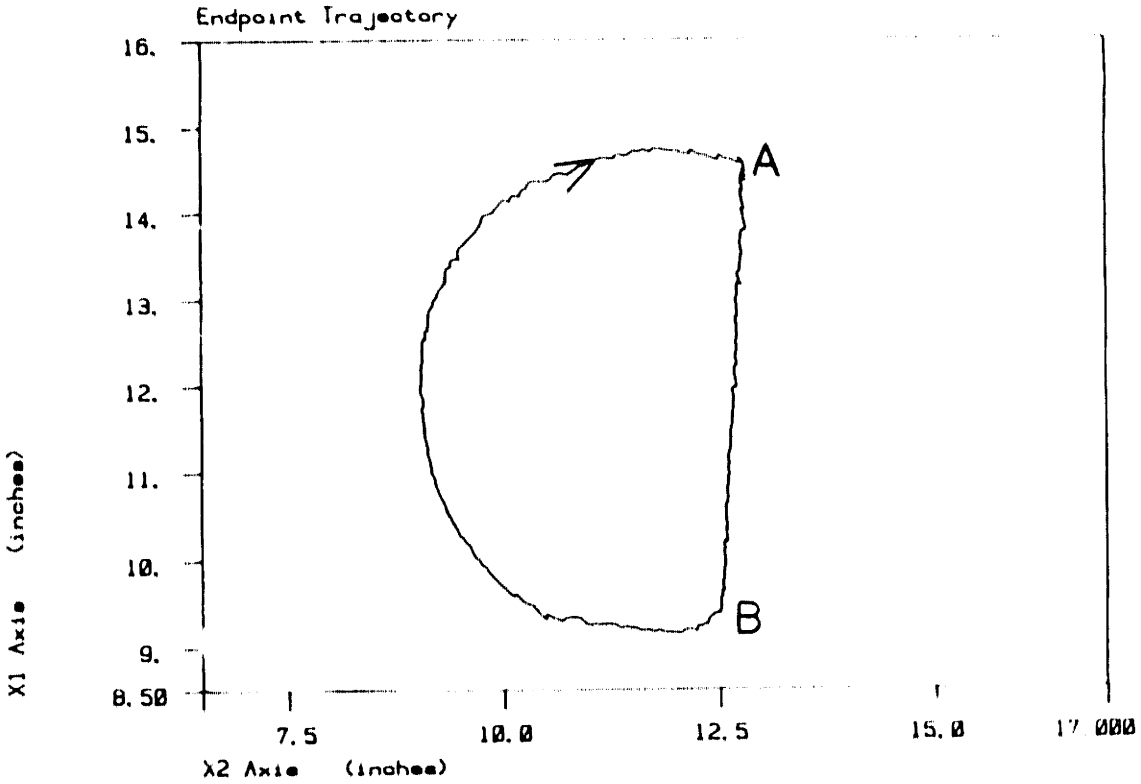
point of impact in the upper right hand corner. The movement was due to the inertial loading of the moving links straining the force transducer and causing a slight bounce. Recall that the force sensor dynamics are outside NFFIC control for this application.

The magnitude of the normal force versus time for at least two cycles is presented by figure 6.1. Note the initial impact and bounce off the barrier at the leading left hand edges. The small amplitude oscillations superimposed on the larger force signal were most likely due to the differentiator. Multiplying the command signals due to the differentiator, figure 5.4, by a scaling factor to compute the resulting force at the mechanism's endpoint, one finds force oscillations on the order of those in figure 6.1. The desired stiffness was chosen so as not to exceed the normal force limit (3.75 lbf) of the force transducer given the largest deflection from an equilibrium position (about 2.8 inches).

6.4 Test results for desired mass greater than actual mass.

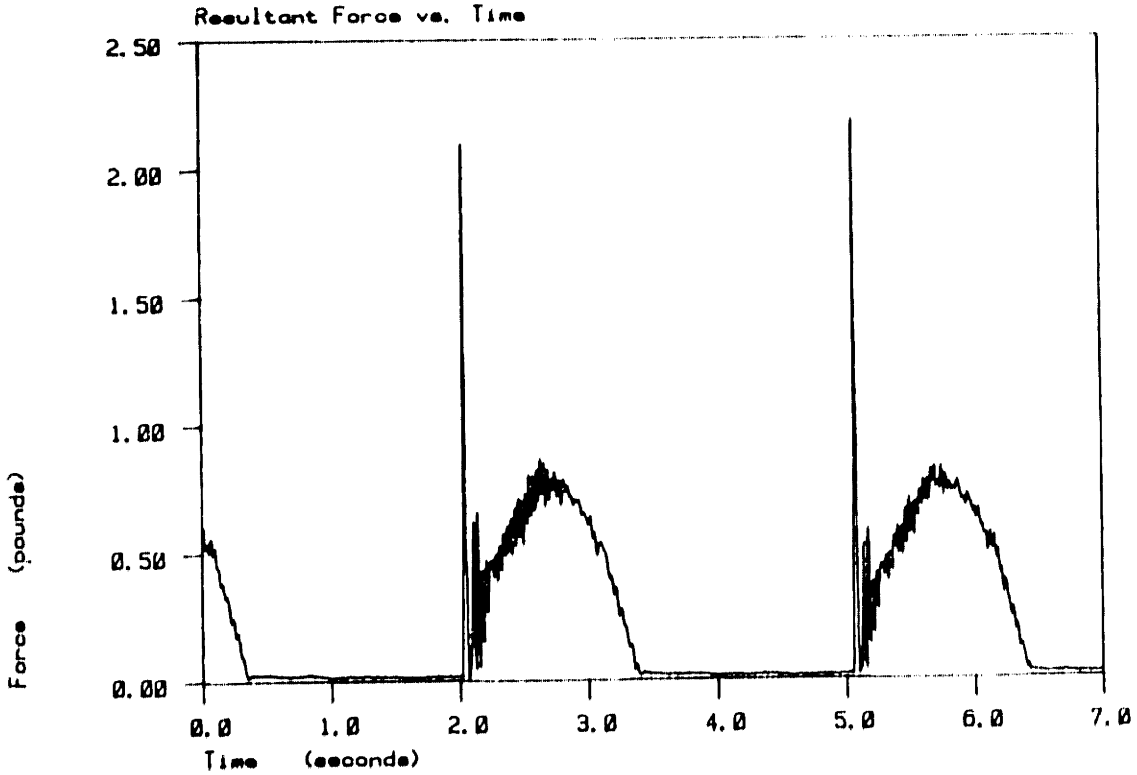
Figure 6.2 depicts the endpoint trajectory for the circle-with-barrier test for positive force feedback. The desired mass was chosen to exceed actual mass in the envelope of motion considered. Note the improved impact response in the upper right hand corner compared to figure 6.0.

Figure 6.3 is the normal force versus time trace for at least two cycles of the test shown by figure 6.2. Note the improved impact response and subsequent edge tracking. The proof mass did not bounce off the barrier as in the zero force feedback test (see figure 6.1). The normal force varies in an arc, as it should; equilibrium positions continue into the barrier and then recede.



Positive force feedback, circle with barrier, endpoint trajectory.

Figure 6.2



Magnitude of the normal force for the positive force feedback test.

Figure 6.3

The most important result from this regime is the confirmation of the stability analysis of section 3.4. Despite the force transducer being many times more stiff than the desired mechanism stiffness (700 lb_f/ft versus 4.53 lb_f/ft) the controller and mechanism remained stable.

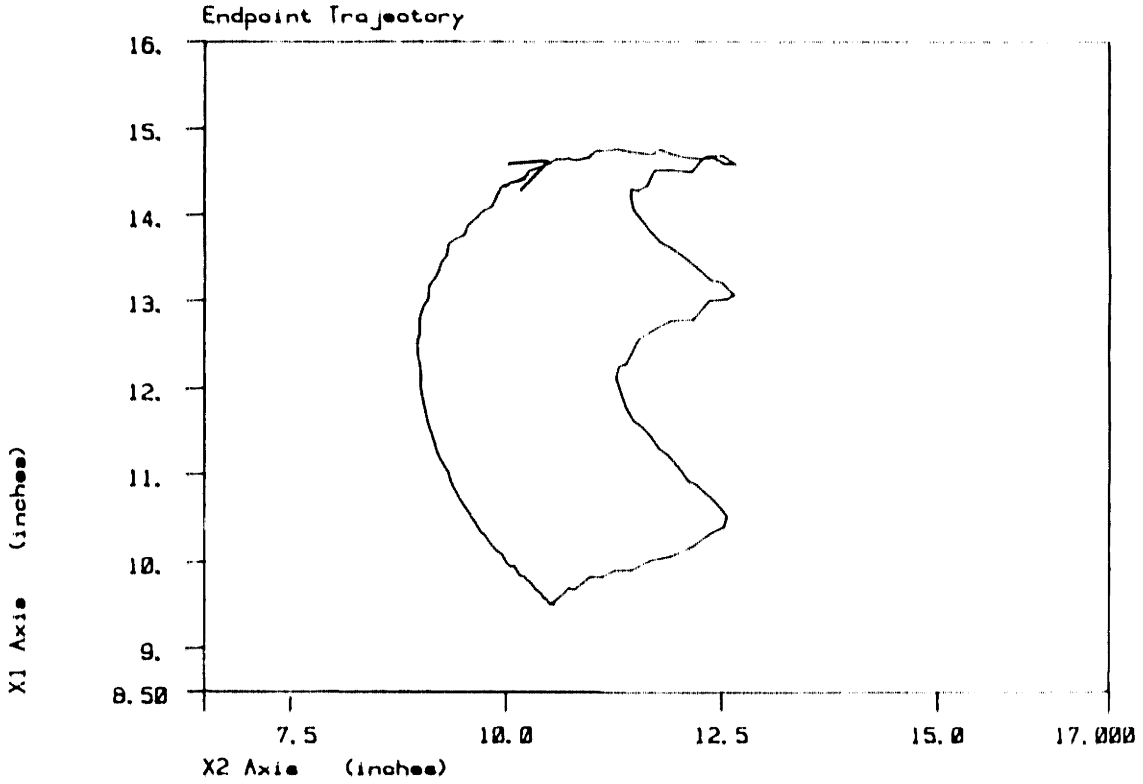
In fact, experience with this apparatus showed that stable interface force control was achieved for any combination of mechanism, sensor, and environmental stiffnesses by using desired masses greater that actual masses. In the case presented here, desired mass was approximately one and a half times the actual mass (of the X2 axis) in the start position.

6.5 Test results for desired mass less than actual mass.

This test corrosponded to negative force feedback. Current investigations into force feedback are conducted for negative force feedback [53], [50], [56], [43], [44], [46], [18], [24], [51]. Figure 6.4 shows the results of endpoint trajectory for the circle and barrier test. The magnitude of the normal force history for this same test is shown by figure 6.5. The system is stable when unconstrained but exhibits pronounced and sustained oscillations on contact with the barrier.

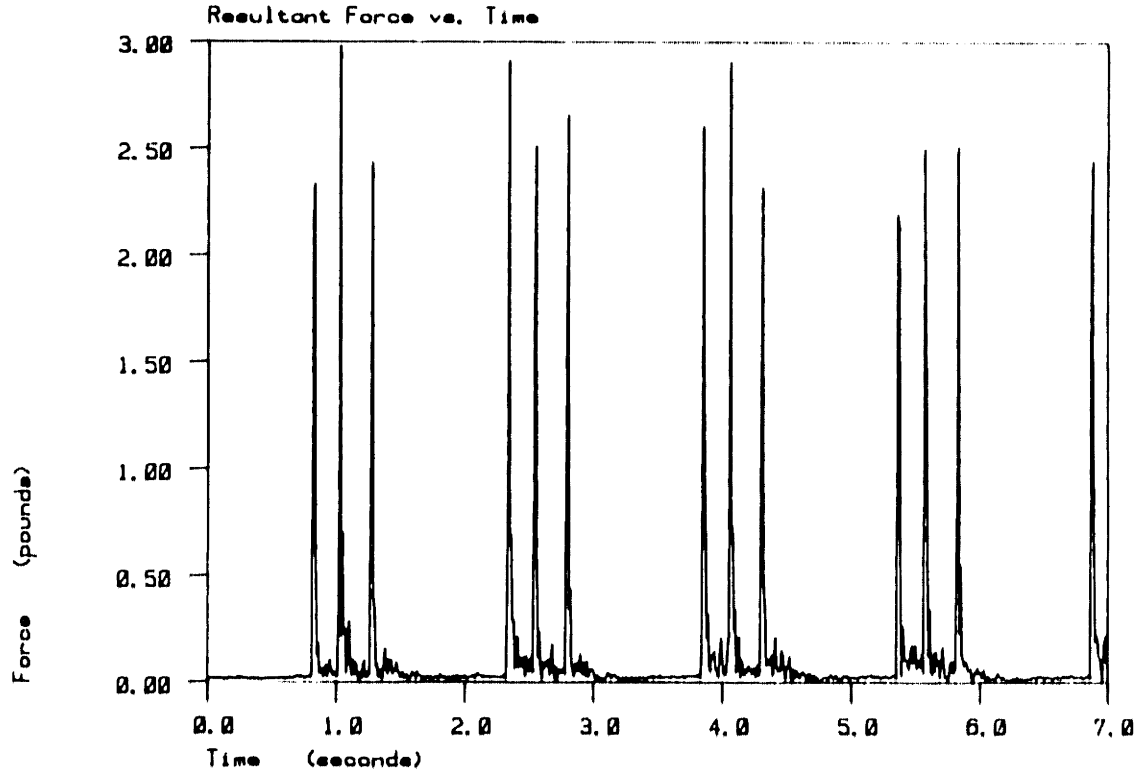
A less stringent test was performed; release point A of the barrier impact test. Was the system still unstable? For a desired mass to actual mass reduction of 10%, along X1 and X2 in the near vicinity of the start position, zero height impact was stable. However, any further reduction in desired mass led to unstable behavior (analogous to figure 6.5) for release point A.

At what frequency was the controller going unstable? Two empirical



Negative force feedback, circle with barrier, endpoint trajectory.

Figure 6.4



Magnitude of the normal force for the negative force feedback test.

Figure 6.5

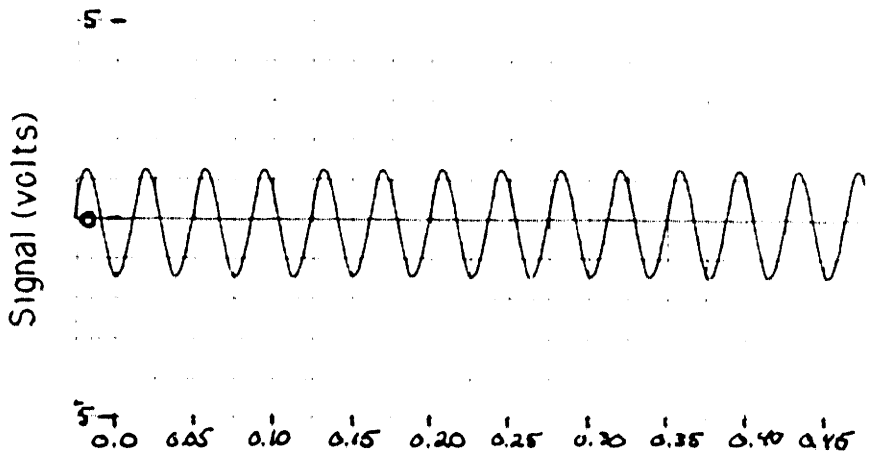
tests were performed to find out. The first required a setup where the force transducer was rigidly held against the barrier with the mechanism in the start position. The NFFIC law was then run with the same parameters as the circle with barrier test in this section. The voltage output from both axes was recorded on magnetic tape (RACAL Store 14D Tape Recorder) and played back at a slower speed for a strip chart recorder (Gould Recorder 2400). This gave a direct measurement of the frequency of instability at 27 Hz. A sample for one axis of the strip chart output is shown in figure 6.6.

The second means to determine the frequency of instability, as a check on the first, was to get a strip chart output of the analog data used to plot figure 6.5. A sample appears in figure 6.7. The large amplitude signal is the impact of the force transducer on the barrier. If one considers this signal as 1/2 of a full cycle of an unstable oscillation then the unstable frequency can be backed out. The frequency was calculated to be 26 Hz, confirming the prior test's results.

6.6 Search for the cause of negative force feedback instability.

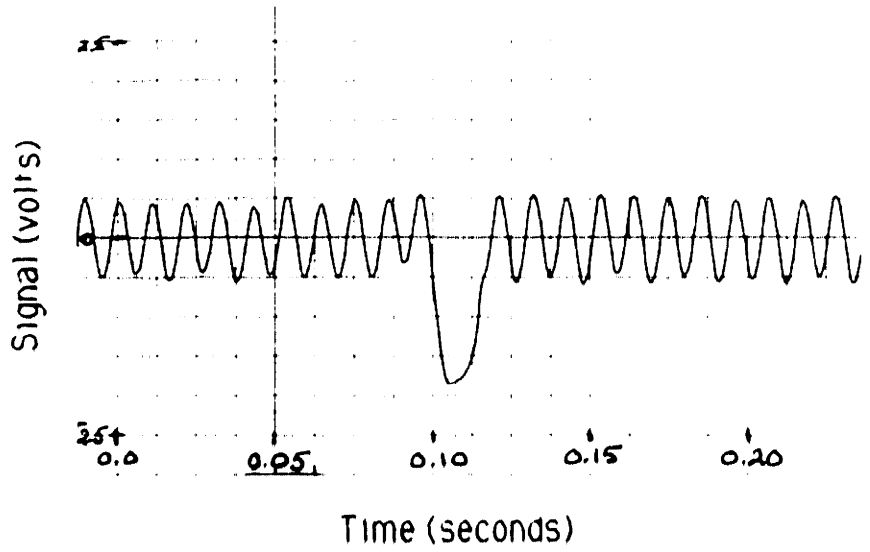
One possible cause of the instability was an error in the NFFIC code that manifested itself only for negative force feedback. A thorough inspection of the MACRO-11 control algorithm was made. No errors were found. The linear digital control algorithm was then run to repeat the test shown where the proof mass was constrained with the mechanism in the start position. Commensurate gains with the NFFIC test were assured by the procedure described in section 5.3. The instability persisted at the same frequency, 26 Hz. It was concluded that the NFFIC software was not at fault.

Was the unstable behavior due to digital phenomena? To answer this



Direct measurement of the frequency of the negative force feedback instability

Figure 6.6



One half cycle of the negative force feedback instability.

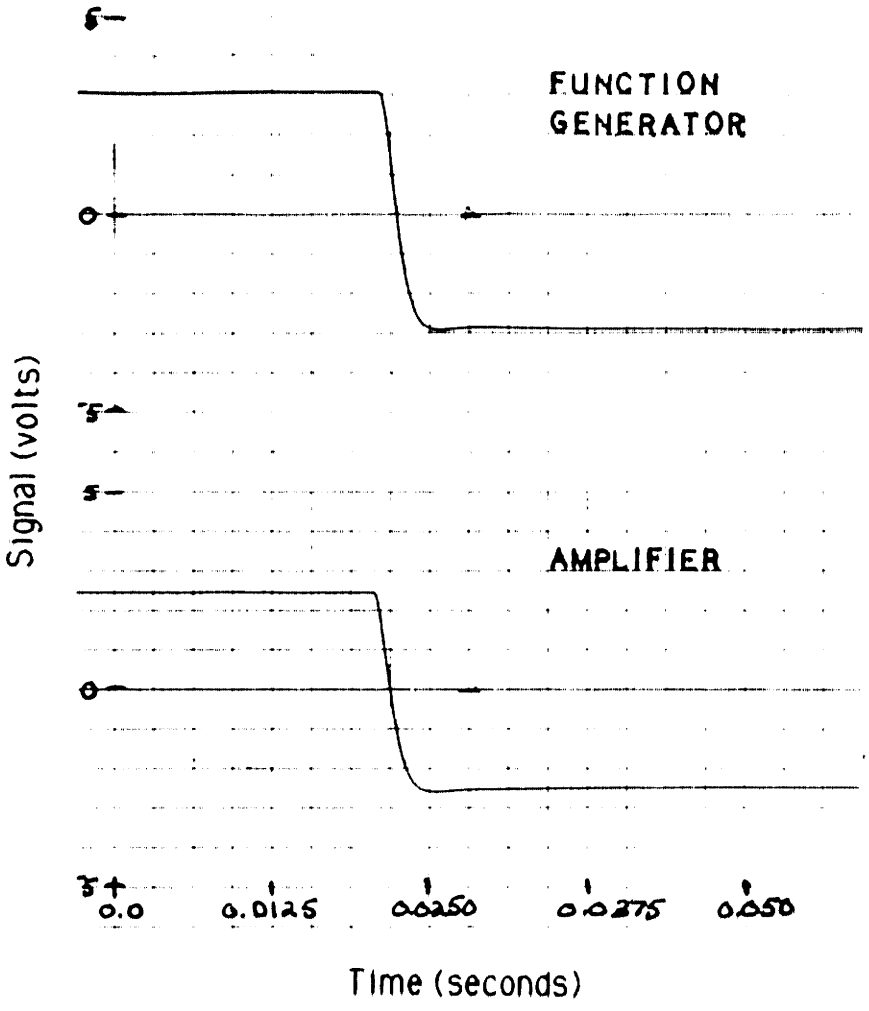
Figure 6.7

the same setup as for the linear digital controller test was made. All three control schemes were then run in this configuration. The linear PVF and NFFIC schemes were activated with zero velocity feedback so that a fair comparison with analog PF control could be made. There was no measurable difference in the frequency of instability. The strip chart output of figure 6.6 is typical of all three controllers. Digital phenomena were judged not to have caused the unstable behavior.

Could the servo amplifiers be adding extra dynamics to the system? Early on in this work the lead/lag compensation circuit in the servo amplifiers had been disabled due to nonideal behavior observed when using analog position feedback. The ideal model of a linear transconductance gain for the servo amplifier model fell into suspicion. A step input test was performed using a square wave function generator [HP 3310A]. The rotors of each motor were locked for this test. The resulting current monitor voltage provided on the servo amplifiers can be compared to the function generator's step input in figure 6.8 for servo amplifier # 1. Note that no perceptible delay or dynamics are visible in the output. The test was first recorded with a fast tape speed and then played back slowly for the strip chart recorder so as not to exceed the strip chart recorder's bandwidth (about 100 Hz). The fast tape speed assures that signals up to 10K Hz were faithfully recorded.

The step response test confirmed the assumption that the servo amplifiers provide a linear transconductance gain to at least 1K Hz as the manufacturer claims.

Could the instability have been caused by mechanical coupling between the links? With the mechanism in the start position with one link rigidly held to the motor stand at a time, analog PF control was activated. In this way the



Servo amplifier step response.

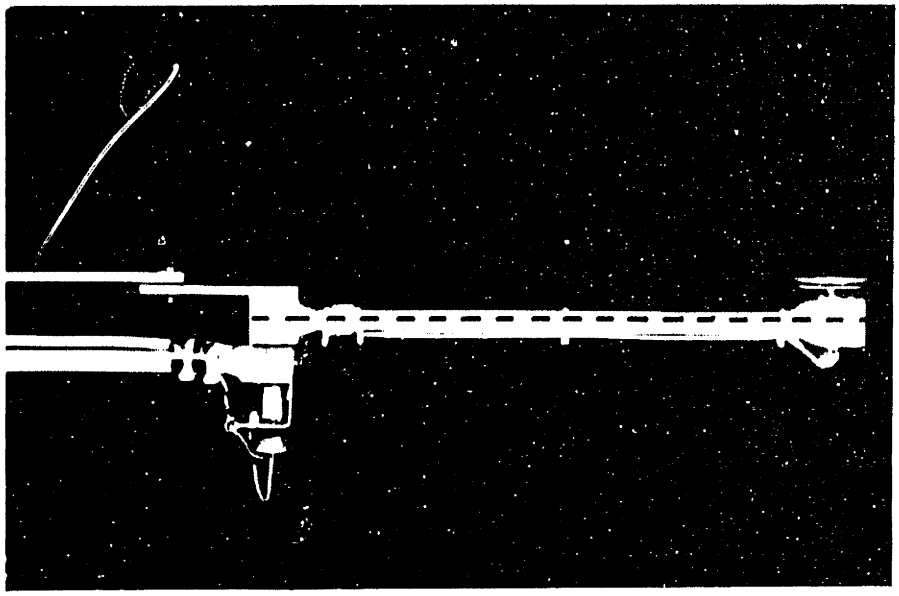
Figure 6.8

system could be viewed as having one degree of freedom since only one amplifier and motor with link was turned on. Even under these conditions the unstable behavior was manifest.

The next logical theory for what caused the unstable oscillations was system vibrations. In or out of plane oscillations are not part of the model of the mechanism used to derive the NFFIC torque law. In fact, it has been argued that a complex model of link dynamics is not necessary for NFFIC. The question asked was: is there a way that the force transducer, also not a part of the mechanism model, could be coupled with link dynamics? It should be noted that this particular mechanism, while only a testbed, visibly exhibited mechanical resonances in many of its major mechanical components.

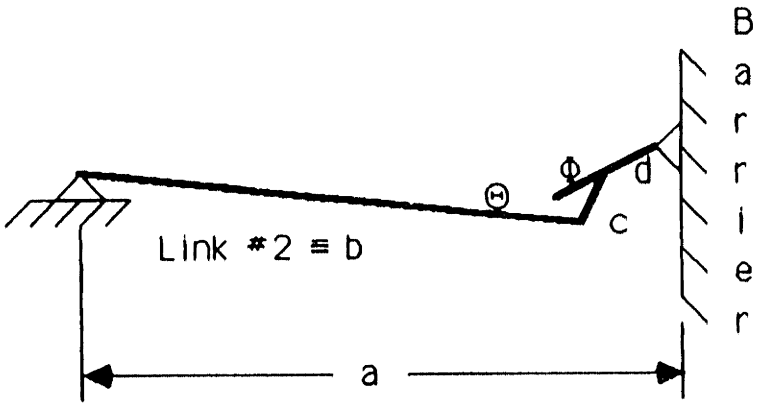
Figure 6.9 shows a side view of link #2. The dashed horizontal line passes through the center of mass of link #2. Note how the point of contact with any barrier occured at a point elevated above the horizontal line. The effect would have been an out of plane torqueing of the mechanism. Similarly, the four bar transmitted in plane torque at a level above the horizontal line giving rise to out of plane torques. A similar out of plane torque resulted from motions of link #1. Was out of plane torque coupled to force sensor deflection?

Figure 6.10 is a four bar model of the linkage shown by figure 6.10. Knowing link lengths one could relate out of plane motion, Θ, to proof mass motion, Φ. The relation between Θ and Φ for a four bar is described by equations 6.0 through 6.2 [48].



Side view of link #2.

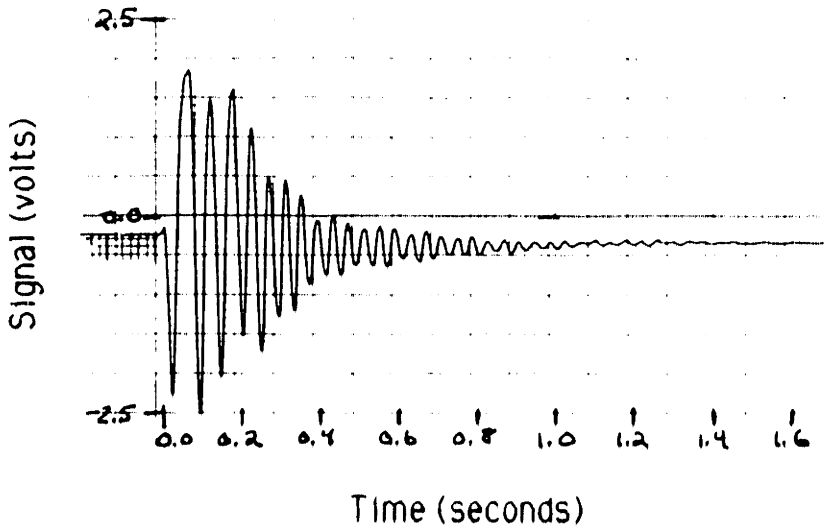
Figure 6.9



Side View

Four bar model of link #2 and force sensor.

Figure 6.10



Out of plane ascillation coupled to in plane force sensor signals.

Figure 6.11

$$\{[c^2-(a+b+d)^2]+[c^2-(b-d+a)^2]u^2\}t^2+(8bdu)t+$$

$$\{[c^2-(a-b+d)^2]+[c^2-(b+d-a)^2]u^2\}=0 \qquad \text{eq. 6.0}$$

where:

$$a = 13.1" \ , \ b = 12.0" \ , \ c = 1.42" \ , \ d = 1.0"$$

$$u = \tan(\Phi/2) \equiv \text{known} \qquad \text{eq. 6.1}$$

$$t = \tan(\Theta/2) \equiv \text{unknown} \ . \qquad \text{eq. 6.2}$$

A estimate of proof mass tilt, from experimental observations, was 5 degrees at most. When equations 6.0 through 6.2 were solved for Θ the result was .4 degrees. This means that for very small out of plane motions, many times larger proof mass tilt could be expected. Since force transducer strain is really a function of proof mass tilt and out of plane motion, one could conclude that there could be a significant coupling between out of plane oscillations and force transducer signal.

To confirm the supposition that out of plane oscillations contributed to force readings a test was devised to measure this contribution. The mechanism was placed in its start position with the barrier set to restrict motion along the positive X2 axis. A one volt signal was input to servo amplifier #1 without any controller active. This caused the proof mass to load against the barrier with .6 lb_f. The loading was intented to simulate the mechanism tracking an edge under active control. Light hammer strikes were then applied at joint #2 in the vertical direction. Figure 6.11 shows a force sensor signal for an out of plane oscillation. The peak amplitude of the force transducer output, shown by figure 6.11, was 25% of the full scale. The

frequecy of the oscillations shown by figure 6.11 was 24 Hz with a .05 damping ratio. Considering that the hammer strikes resulted in relatively small amplitude out of plane oscillations, as compared to out of plane oscillations witnessed during unstable control, this test confirmed that out of plane motions had a substantial effect on in plane force sensor readings.

6.7 Experiments to stabilize negative force feedback.

The view taken was that hardware and software were ready, why not try some well aimed experiments before continuing the analytical approach to finding the source of the instability? The problem can be considered as unstable behavior beginning at 26 Hz. If one could stabilize the negative force feedback loop with a 20 Hz bandwidth for dynamic interactions with the barrier, the result would be a very good bandwidth relative to today's robots [30].

A first cut at designing experiments to stabilize the negative force feedback loop when the mechanism was constrained by its environment was to consider the mass and spring combination of link inertia and force transducer stiffness:

$$\omega_n = \sqrt{\frac{K_{td}}{M_a}} \qquad \text{eq. 6.3}$$

Equation 6.3 solved for link #2 parameters yielded 28 Hz, very close to the observed frequency of instability. Tests of NFFIC with only force feedback active resulted in the same instability as shown by figure 6.6. These last two pieces of information lead to experiments designed to work on the force feedback signal and the force transducer itself to stabilize the negative force

feedback loop for dynamic interactions.

Any form of analog forward compensation, such as lead, lag, lead/lag are linear techniques that shape a root locus of a linear plant. The physical characteristics (the open loop poles and zeros) of the mechanism are geometry dependent. Compensators to shape a particular configuration's root locus would fail for a different configuration. Digital or digitally controlled analog compensators that adjust for changing configurations were considered too involved for the quick tests planned at this stage. Furthermore, the command signal to the servo amplifiers has position and velocity information in it that one may not wish to compensate since tests have shown the system to be stable for unconstrained motions.

Feedback compensation appeared as a way of isolating the force signal. Filtering the force feedback signal so that the unstable force feedback oscillations at 26 Hz were attenuated appeared as a plausible way to stabilize the controller for dynamic interactions with desired mass less than actual mass. The single pole filter already present in the signal conditioning unit was altered to give increasingly lower break points. There was no noticeable improvement in stability down to a 2 Hz breakpoint for the zero height impact test described earlier. Similar results for filtering are reported in [44] for this regime, negative force feedback. At such a low breakpoint the force signal was denied a great deal of information. More importantly, the system was responding to the dominant pole of the filter and was therefore quite sluggish.

A higher order filter was tried with the intent that a higher frquency breakpoint would achieve stable behavior. Two four pole Chebyshev filters, with .5 Db ripple in the passband, were constructed and inserted in series with

the force transducer outputs. The filter circuits were built using commonly available operational amplifiers (μA747). A break point of 20 Hz was tried first since this filter had a steeper rolloff and sharper knee than the single pole filter had. Again, no significant improvement in quelling the unstable behavior was noticed. It took a breakpoint of 4 Hz before behavior changed. At this point a tradeoff appeared. The system responded to impacts almost as if force feedback was not active. Then as the force signal grew a stable response, as if a step had been the input, appeared. If one was willing to accept this sluggish response (which was not as sluggish as the single pole filter at 2 Hz break frequency) then this was a solution to the instability problem. A 4 Hz bandwidth with sluggish response was not considered acceptable by the author so other avenues were investigated.

Rate force feedback was then attempted to "damp" the oscillatory force signal. The differentiation scheme was the same as the final one chosen for both digital controllers. The differentiated force signal was added to the force signal before any matrix multiplications were performed. The rate force feedback signal could also be weighted by a gain. The same tests as for the filter, namely the barrier-impact tests, were used to evaluate the effectiveness of the addition of rate force feedback. The final arbiter was to keep only force feedback active and then see if unstable behavior persisted for all three control strategies. Briefly put, within software limitations, rate force feedback had no effect whatsoever in stabilizing the behavior for negative force feedback. The greatest obtainable weighting of rate force feedback signal to force feedback signal was six to one. A more heavily weighted rate force feedback signal resulted in inconclusive results due to the increase in noise caused by the force feedback signal differentiator.

The last experiment to try to eliminate the unstable behavior in the negative force feedback regime was to add intrinsic damping to the force transducer. Modelling clay and then heavy grease were alternately tried in the gap between proof mass and upper surface of the end of link #2, see figure 4.2. Grease or clay in the gap was believed to be a plausible way of adding damping to the force sensor mechanism.

The clay so overdamped the force sensor that response took on the order of minutes to evolve. Results were therefore inconclusive. The heavy grease however had an obvious effect without slowing down the system visibly. Table 6.0 compares the lowest limits of reduced desired mass before unstable behavior began for the barrier impact test with and witnout intrinsic damping added to the force transducer. Desired mass was reduced, compared to actual mass, the same percentage for both axes with the mechanism in the neighborhood of the start position. The percentages listed are approximate since only a visual observation of stability was conducted.

In summary, only intrinsic damping resulted in improving the lower limit of achievable desired mass (tnus the negative force feedback gain) without visibly slowing down the system's response. Finding out why the filter and rate force feedback did not yield the results, as added intrinsic damping did, is the topic of the next chapter. Information on the apparatus' dynamics could also lead one to find out what was causing the instability. Knowing the cause might lead to a design constraint for mechanisms using force feedback.

	Desired mass reduction (%)	
Force Sensor Damping	Impact	Zero Height Impact
As provided by manufacturer.	2	15
With heavy grease.	7	25

Additional force sensor damping results.

Table 6.0

Dynamics Investigation

7.1 Scope of this chapter.

This chapter quantifies mechanical resonances of the two link mechanism used in this research. Two models are developed that yield insight into the constrained motion negative force feedback instability. The second model also shows why the three stabilization experiments of the previous chapter, namely filtering, rate force feedback, and added force sensor damping, had little success. The reason increased desired mass, compared to actual mass, did not exhibit the constrained unstable behavior that decreased mass did is explained. Finally, an analysis is proposed that unifies force feedback stability criterion in current literature with the results in this investigation.

7.2 Mechanical resonances in the apparatus.

Section 2.3 showed that force feedback loop sign depends on the relative magnitudes of desired mass and actual mass. Force feedback results reported in the current literature deal with negative force feedback only. Unstable oscillations were not apparent under any conditions in the positive force feedback regime.

The two links, four bar, and motor stand have several modes of virbration. Visually, first modes were apparent whenever the endpoint was struck. The dominant resonance was clearly noticed to be out of plane link oscillations. This resonance was discussed in section 6.7 and found to have significantly influenced in plane force sensor readings. Impacts and torque transmission to the outer link were shown to excite the out of plane mode.

Any model to investigate the many modes of resonance in apparatus would have to include the out of plane oscillations.

Two approaches were taken to quantify this out of plane resonance; an analytical and an empirical approach. The analytical approach started by considering link #1 to be a cantilever with a lump of mass at the free end equal to the two links plus force sensor and four bar. The first mode of oscillation was calculated from [6]:

$$f_1 = \frac{1}{2\pi}\left[\frac{3EI}{ML^3}\right] \qquad \text{eq. 7.0}$$

When solved for the parameters of the mechanism equation 7.0 yields an out of plane oscillation of 15 Hz.

The empirical approach to measuring the out of plane oscillations was in two parts. The first part was described in the previous chapter to show out of plane oscillations causing force sensor readings. The test is summed up in figure 6.6. The frequency of out of plane oscillations from figure 6.6 was 22 Hz.

The second part tried to measure out of plane vibrations with the links unconstrained in the start position. All electronics except for the force sensor were turned off. The proof mass was accelerated by out of plane motions leading to force sensor readings. Out of plane oscillations were excited by sharp hammer strikes near the end point and near joint #2. For this part of the empirical approach strip chart recordings showed 21 Hz out of plane vibration regardless of hammer strike location.

The frequency out of plane oscillations used in the models in the upcoming sections was based on empirical results, given that the analytical

approach was based on estimates. For free or semiconstrained links the result used was 21 Hz.

An approximate analysis of the remaining resonances was as follows. Components such as the rectangular piece of metal holding the mount for motor #2, were treated as cantilevers. The bench on which the apparatus was fixed did not have to be analyzed since by definition no relative motions between bench and mechanism existed. The results are shown in table 7.0. Components can be identified by reviewing the photographs in chapter 4.

Table 7.0 merely proves the existence of resonances other than the out of plane resonance. The existence of these resonances will be used in an argument later.

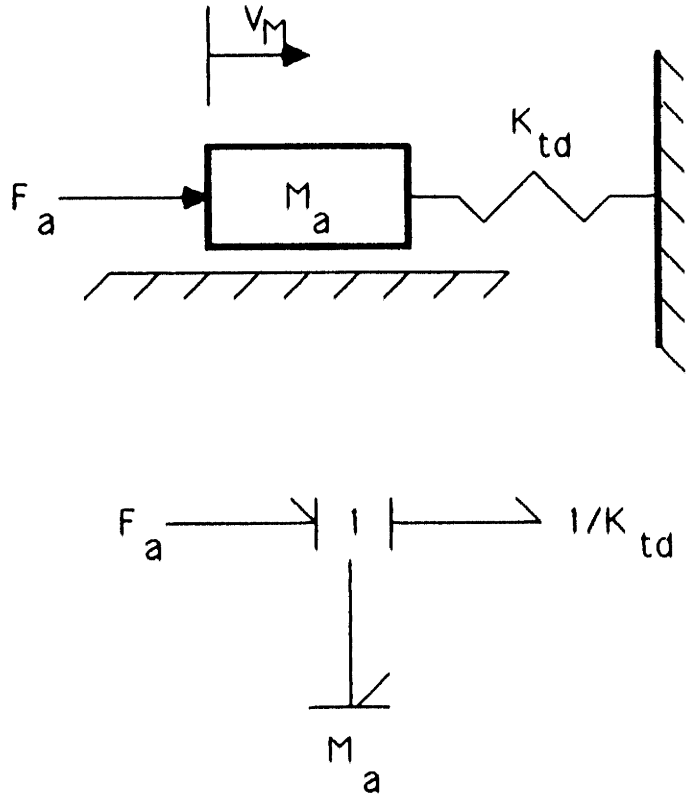
7.3 Motivation and limitations of proposed modelling.

To find the underlying reasons why filtering, rate force feedback, and added intrinsic damping at the force sensor all had different results on negative force feedback stability, models of the system were developed. The goal was to formulate a model that would at least be qualitatively correct in its prediction of real world behavior. Models were to be kept simple yet embody the dominant behavior of the mechanism.

The same 27 Hz unstable oscillations were recorded for all three control schemes. Further, when negative force feedback alone was active, the same 27 Hz unstable oscillation was manifest. The in plane force sensor deflections were shown to be heavily coupled to out of plane oscillations in the last chapter. These last three points essentially reduced the required analysis to one degree of freedom. For all of the analyses to follow, only parameters for the X2 axis were considered.

Structural Component	Natural Frequency (Hz) 1st mode	2nd mode	3rd mode
Links in bending (together)	18.0	161.0	450.0
Motor #2 cantiler in bending	425.0	–	–
Vertical back plane in bending	69.0	270.0	757.0

Some structural resonances in the system.
Table 7.0



Model #1 and bond graph.

Figure 7.0

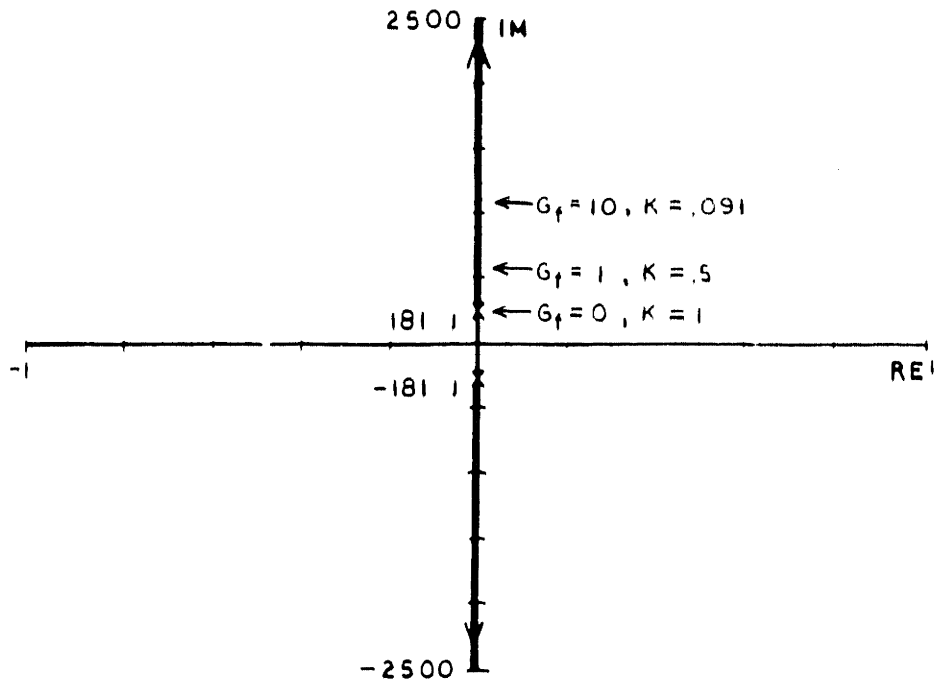
7.4 Model #1 .

The first model tried is depicted in figure 7.0 with its bond graph representation. The mass (M_a) is the effective endpoint inertia of both links in the X2 direction. The spring represents the force sensor stiffness. The control force could be considered to represent an impedance control force for a one degree of freedom system. The rigid body constrains the force sensor to represent a collision. One wants to get the relation between spring (force sensor) force and comand (impedance control) force:

$$\frac{F_{td}}{F_a} = \frac{K_{td}/M_a}{S^2 + K_{td}/M_a} \qquad \text{eq. 7.1}$$

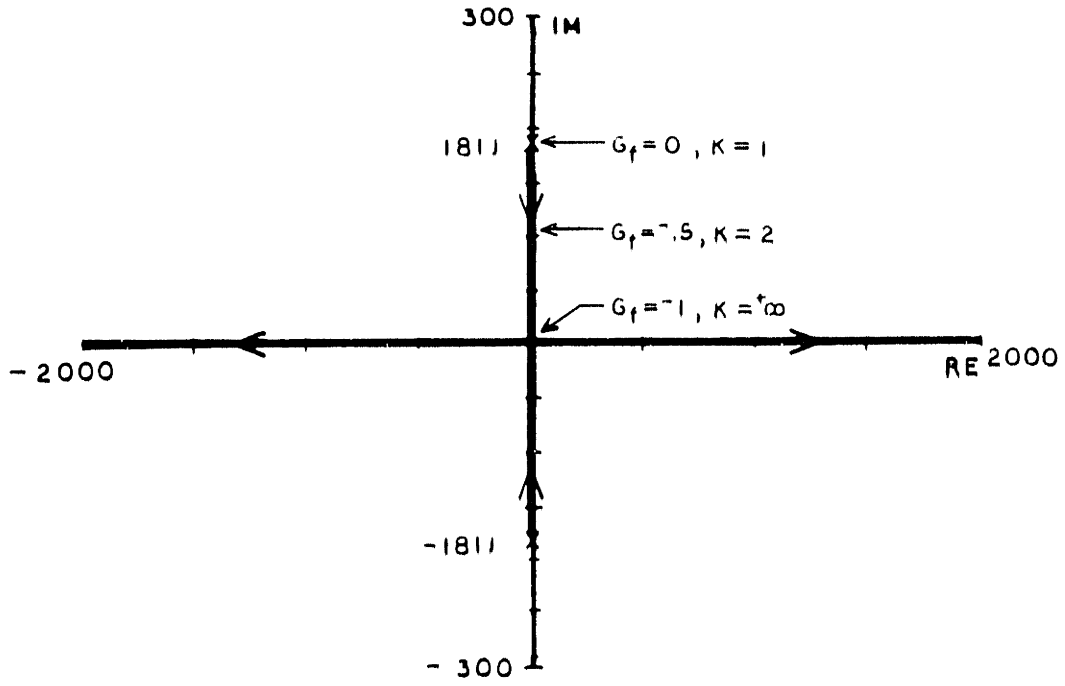
Equation 8.1 solved with actual system parameters is the open loop transfer function between feedback force and command force. The closed loop negative force feedback root locus of equation 7.1 with loop gain starting from zero is shown in figure 7.1. Recall equation 2.12 which related force feedback gain to the defined quantity K (multiples of actual mass) In figure 7.1 some values of K and negative force feedback gain (G_f) are shown. Henceforth all root loci start at the open loop poles and have values of K and G_f on them. The root locus was interpreted as meaning that desired mass can be reduced to zero without explicit instability. This prediction was not true in practice.

Some insight could be gained however from figure 7.1. Note that the open loop poles correspond to the 28 Hz first cut guess at the cause of the unstable behavior; section 6.7. These poles are due to the force sensor and effective endpoint mass. Any intrinsic damping in the model would pull the



Model # 1 negative force feedback root locus.

Figure 7.1



Model # 1 positive force feedback root locus.

Figure 7.2

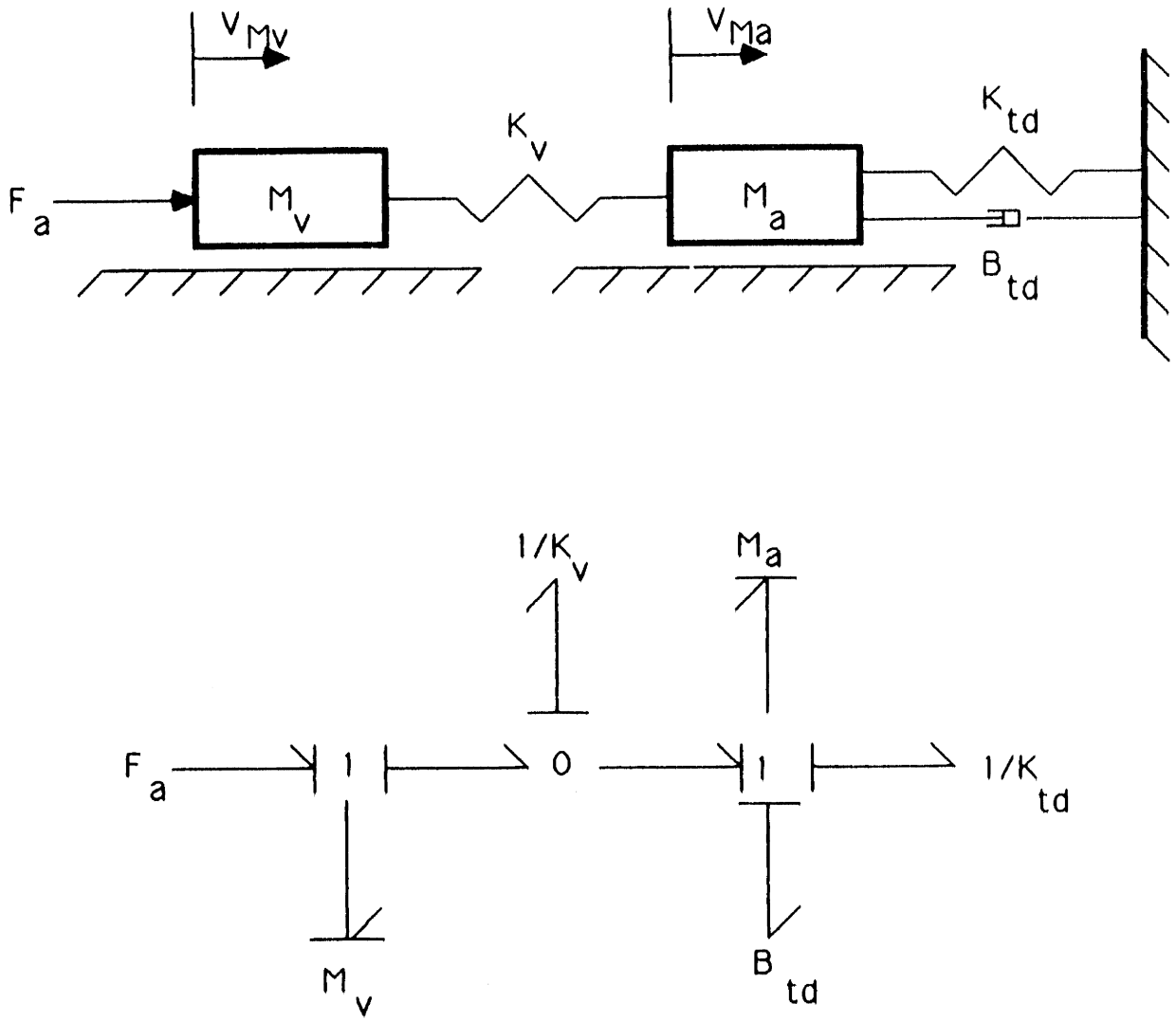
poles into the stable region left of the jω axis. Additional poles to the left of the force sensor poles would cause the force sensor poles to go unstable for even a slight negative force feedback gain. Additional poles are certainly present due to resonances proved to exist in the apparatus. The effect of these resonances and exactly what component of the mechanism is causing the instability motivates a higher order model to be developed.

The analysis so far suggests that since mechanism resonances play a part in system stability for negative force feedback loops, mechanical design criteria should include control considerations as well. This view will be discussed further later in this chapter.

The positive force feedback root locus of equation 7.1 is shown in figure 7.2. Note that the model predicts stable behavior for infinitely large desired mass. In practice, within the software limits of NFFIC, this prediction is confirmed. As long as the DC gain of force feedback transfer functions remains unity, then as a consequence of root locus construction rules the the system will always cross the imaginary axis for unity positive force feedback gain. For one degree of freedom, unity positive force feedback gain corresponds to an infinitely large desired mass.

7.5 Model #2 .

The second model developed, with bond graph, is depicted in figure 7.3. A mass and spring, which model one additional mode of vibration, are denoted by the subscript "v". A damper was included to account for any intrinsic damping in the force transducer (i.e. the grease which was intended to add intrinsic damping to the force transducer) appears in parallel with the force sensor spring.



Model #2 and bond graph.

Figure 7.3

The transfer function relating force output to commanded force was derived as:

$$\frac{F_{td}}{F_a} = \frac{\dfrac{K_t + K_v}{M_a M_v}}{s^4 + \dfrac{B_{td}}{M_a}s^3 + \left[\dfrac{K_{td}}{M_a} + \dfrac{K_v}{M_v} + \dfrac{K_v}{M_a}\right]s^2 + \dfrac{B_{td}K_v}{M_a M_v}s + \dfrac{K_{td}K_v}{M_a M_v}} \quad \text{eq. 7.2}$$

If one defines the additional vibratory mode related natural frequency as:

$$\omega_v^2 = \frac{K_v}{M_v} \quad \text{eq. 7.3}$$

and force transducer related natural frequency as:

$$\omega_{td}^2 = \frac{K_{td}}{M_a} \quad \text{eq. 7.4}$$

then equation 7.2 can be rewritten as:

$$\frac{F_{td}}{F_a} = \frac{\omega_{td}^2\,\omega_v^2}{s^4 + \dfrac{B_{td}}{M_a}s^3 + \left[\omega_{td}^2 + \omega_v^2 + \dfrac{K_v}{M_a}\right]s^2 + \omega_v^2\dfrac{B_{td}}{M_a}s + \omega_{td}^2\omega_v^2} \quad \text{eq. 7.5}$$

A discussion of the parameters in equation 7.5 is in order. Force sensor related natural frequency, ω_{td}, was calculated from empirical values. Direct measurement of the damping coefficient due to the grease was not available. A value of 2 lb_f/ft/sec (which yields a damping ratio of 0.22) was used as a reasonable estimate. The frequency of the out of plane oscillation was used

for ω_v. This was judged the best choice since out of plane oscillations are highly coupled to in plane force sensor deflections.

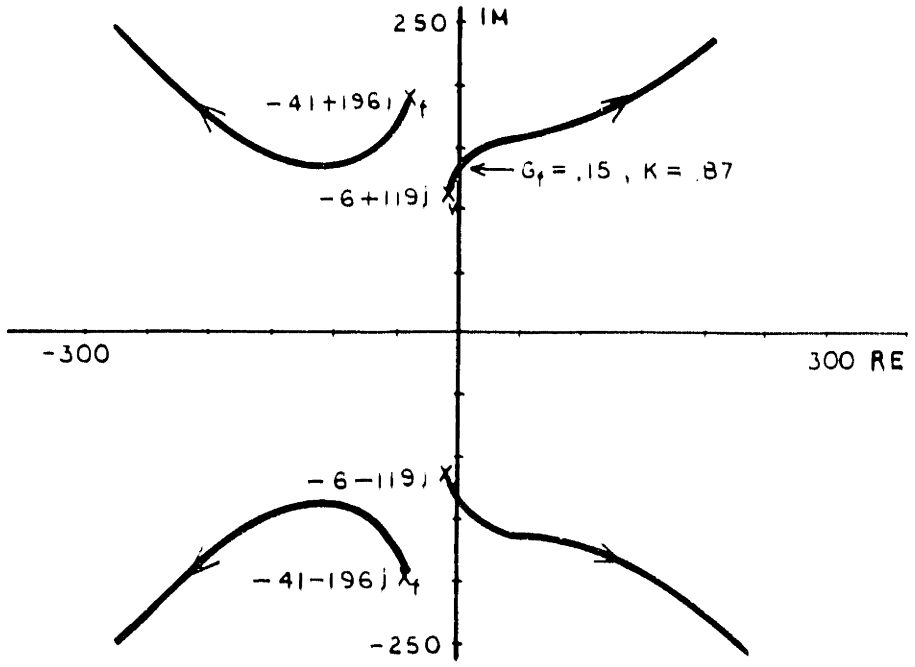
The only parameter in equation 7.5 not described so far is the out of plane vibratory stifffness, K_v. As an aside, equation 7.5 could be rewritten so that vibratory mass, M_v, is the only remaining paramter, however K_v could be empirically determined with greater reliability than M_v could. The out of plane stiffness was calculated by incrementally loading link #1 vertically at joint #2 and noting vertical displacements from the rest position for each additional load. The resulting out of plane stiffnesses were averaged to yield an out of plane stiffness of 110 lb_f/ft.

The negative force feedback root locus of eqution 7.5 appears in figure 7.4. The open loop poles related to the force sensor are identified by the subscript "f" and those related to the vibratory mode by the subscript "v". Note that model #2 is able to predict instability for the reduced mass case.

7.6 Insight into stability experiment's results.

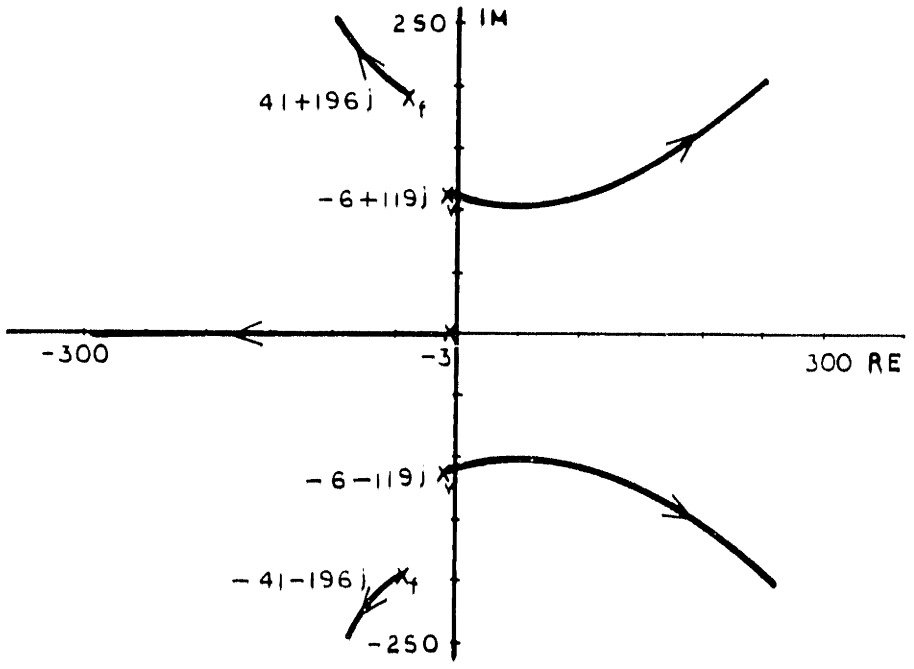
The addition of intrinsic damping at the force sensor shifts the root locus of model #2 to the left and spreads the two pole pairs apart. The result is a decreased achievable desired mass and lower frequency of instability compared to model #1 with higher frequency poles to the left of the force sensor poles already on the imaginary axis. This prediction was confirmed qualitatively by experiments in the previous chapter.

Figures 7.5 and 7.6 show negative force feedback root loci for the addition of 1 and 2 pole filters at break frequencies of .5 Hz into model #2. Note that the filters have a small, but not negligible, effect on the gain for



Model #2 negative force feedback root locus.

Figure 7.4



Model #2 negative force feedback root locus with a single pole filter.

Figure 7.5

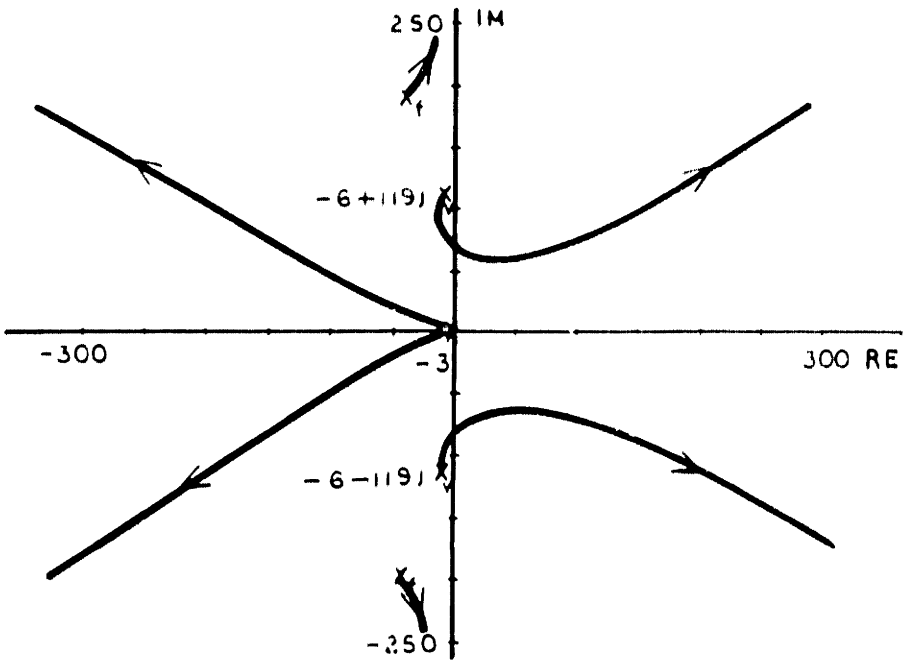
instability as seen in experimental results. Higher order filters add poles to the left of the imaginary axis which still results in a branch of the root locus going unstable.

Figure 7.7 shows the effect of rate force feedback on the negative force feedback root locus of figure 7.4. Note the negligible effect on the shape of the root locus. The zero on the real axis could be shifted right to simulate a more heavily weighted rate force feedback signal; however the gain for instability does not change enough to exceed the errors inherent in this analysis. For this apparatus and software the model shows why rate force feedback failed to stabilize the system. The software limited the gain on the rate force feedback portion of the force signal to such an extent that a more general conclusion on rate force feedback efficacy cannot be prudently made.

7.7 Stability results from this investigation and from current literature compared.

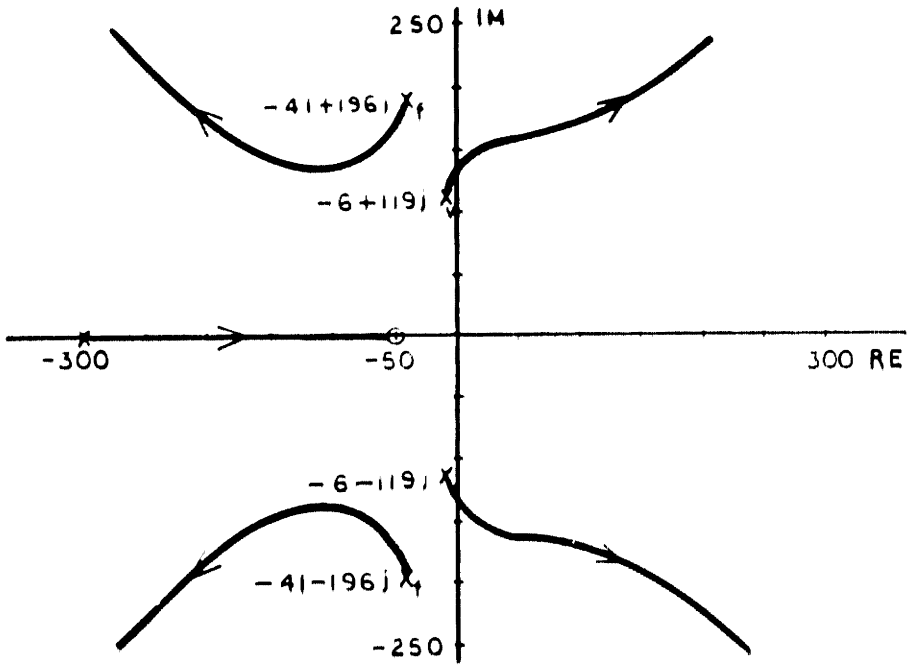
Figure 7.4 also shows that the poles derived from out of plane oscillations are the ones that go unstable and that stiffening the force sensor would increase the gain for instability. To propose that mechanism dynamics are the cause of instability in all negative force feedback systems would be premature. The apparatus and models in this work are laboratory approximations of robot manipulators in current use. A better approach is to look at the four possible combinations of force sensor poles and resonance poles. Figure 7.8 shows these four cases. Was there a trend and can any parallels with negative force feedback systems presented in current literature be proposed?

Case one is a repeat of the figure 7.4; the root locus of the model which



Model #2 negative force feedback root locus with a two pole filter.

Figure 7.6



Model #2 negative force feedback root locus with rate force feedback.

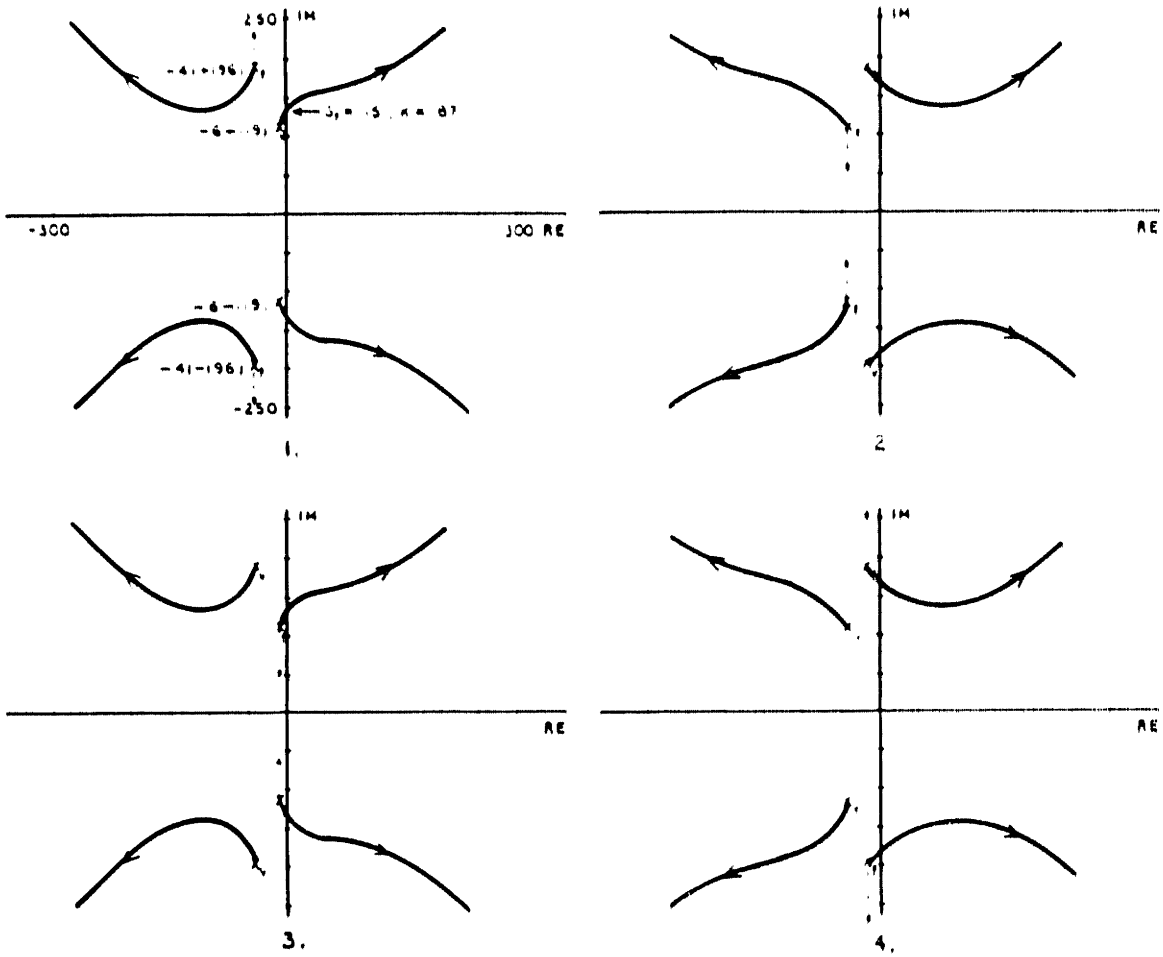
Figure 7.7

was intended to embody the apparatus of this work. Note that a hypothetically stiffer force sensor, denoted by the dashed lines originating at the force sensor poles, would increase the gain at which the resonance poles would go unstable. In this case a stiffer force sensor allows for a smaller achievable desired mass before the onset of instability than a more compliant force sensor would. This result was qualitatively opposite to the results in [44], and [53] which suggest that a compliant force sensor improves the stability margin.

Consider a mechanism desigred to be stiffer structurally (higher frequency resonances) than the apparatus used in this work. The root locus for such a mechanism might look like case 2 of figure 7.8. One might design such a mechanism to permit accurate position feedback control of endpoint motion. This is the control approach taken with most commercial robots today. Note that the structure's resonant mode poles go unstable, as in the previous case. A more compliant force sensor, denoted by the dashed lines, would now result in a greater negative force feedback gain for instability. A less stiff sensor to improve stability is qualitatively the same suggestion as in [44] and [53].

Case 3 in figure 7.8 took the hypothetical mechanism of case 2 and added damping to its structure. The force sensor poles are now the poles that go unstable. Interestingly, a more compliant force transducer would again result in an increased gain for instability, as the previous case. A compliant force sensor improves the reduced mass stability margin for this case.

The final case built on the imaginary mechanism of case 3 and replaced the force sensor with a stiffer one. Again, the force sensor poles go unstable. As in case 1, if an even stiffer sensor were tried the result would be a greater gain (thus reduced desired mass) at instability.



Four cases relating force sensor dynamics to intrinsic mechanism dynamics.

Figure 7.8

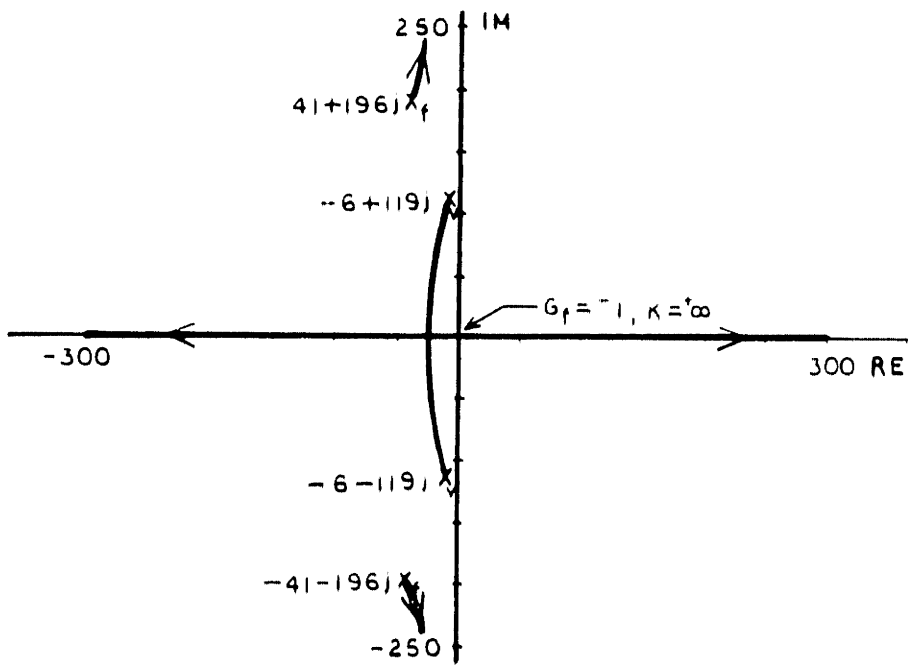
In summary, the analysis presented by figure 7.8 suggests that an intrinsically damped force sensor will always improve the achievable reduction in desired mass regardless of the relative relationship between force sensor stiffness and mechanism structural stiffness, for negative force feedback. A finer point is that reducing force sensor stiffness improves stability only if the stiffness of the mechanism exceeds that of the sensor. This last point confirms the analytical derivation and experimental results in [44], [53]. It also indicates some important alternatives. Based on both the analysis and the empirical results presented in this thesis, there are four ways to improve stability:

1. use a force sensor significantly softer than the structural stiffness of the mechanism (as recommended in [44] and [53])
2. use a force sensor significantly stiffer than the structural stiffness of the mechanism
3. increase the intrinsic damping of the force transducer
4. choose a desired apparent mass greater than the actual apparent mass of the mechanism.

A related result, worthy of noting but not fully developed here, was that dynamic interactions with primarily dissipative environments were stable for negative force feedback gains that produced instability upon rigid body collisions. Examples include when the proof mass encountered the experimenter's hand or traced the edge of a grease filled bladder. For these cases a damper exists between mechanism and environment. A more detailed model than the second one presented here is suggested to better understand

this observation.

Finally, figure 7.9 depicts the positive force feedback root locus of equation 7.5. Again, the pleasing prediction of infinite desired mass before the onset of instability results. Note that in this regime no dynamics investigation was warranted. In fact, the model predicts a modest increase in desired mass (10%) will yield a stable system regardless of sensor stiffness.



Model #2 positive force feedback root locus.

Figure 7.9

Unconstrained Test Results

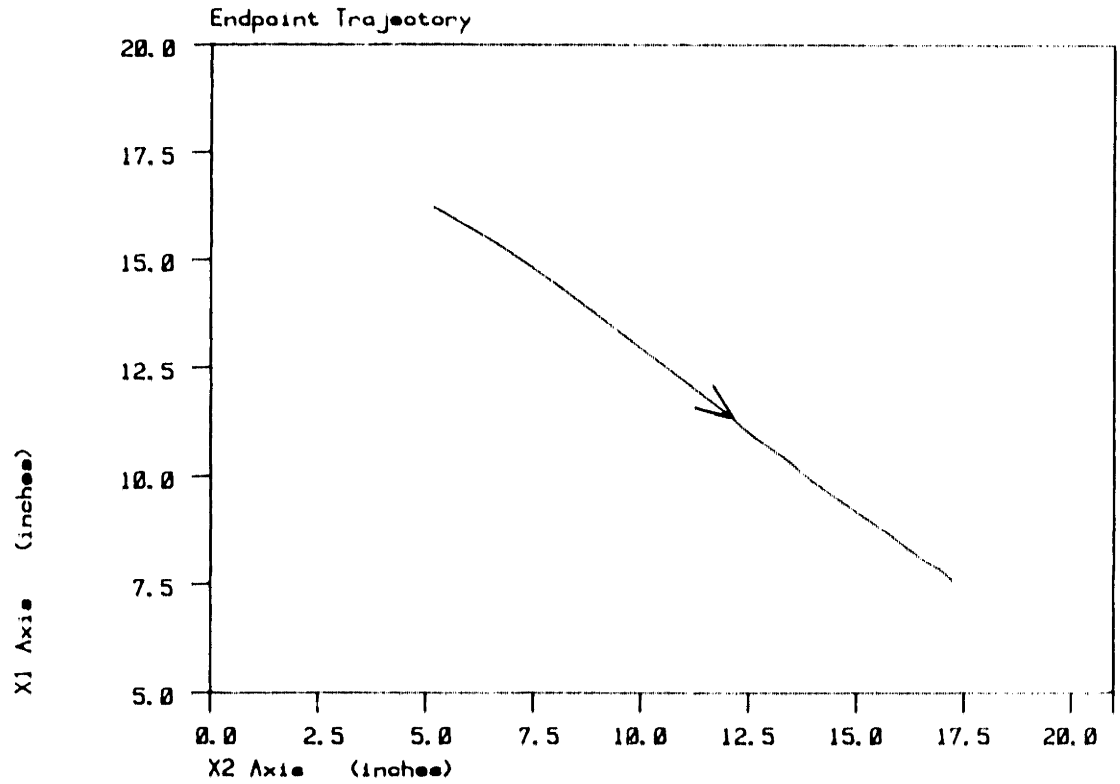
8.1 Scope of this chapter.

This chapter presents results from experiments that were intended to answer whether or not the NFFIC algorithm was achieving the desired behavior. Tests were in the time and frequency domains. The differentiation scheme is analyzed further to explain deviations from desired behavior for some of the tests. The maximum endpoint speed and acceleration of the two link mechanism used in this investigation are compared to commercially available robots and robots in laboratories at MIT.

8.2 Was the desired behavior acheived?

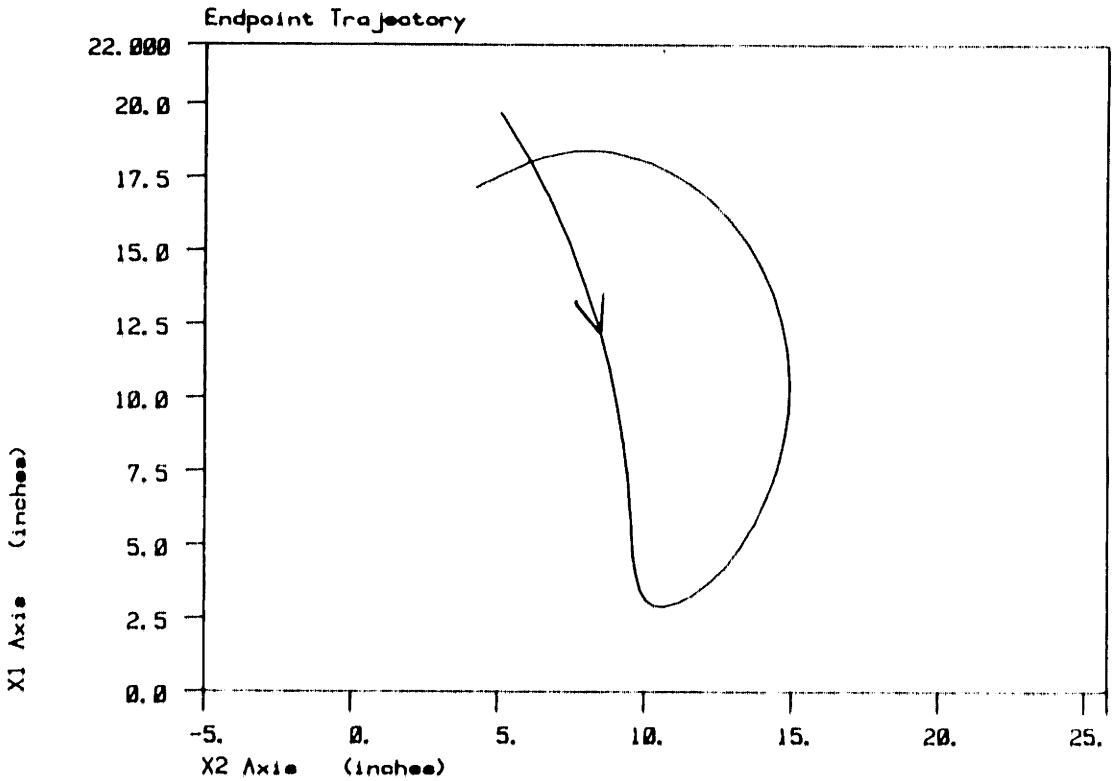
The target behavior was a linear, decoupled, second order system described in chapter 3. The endpoint trajectory for point to point moves for such a system would be straight lines. An experiment to test the hardware under NFFIC for this trajectory was devised.

The mechanism was commanded to regulate in its start position with a stiffness of 6 lb_f/ft and a desired mass of 6 lb_m. The desired damping was set to zero so that the endpoint would oscillate if disturbed. The natural frequency for this desired system was .9 Hz with a zero damping ratio. The endpoint was then pulled about six inches from its regulating point and released. Figure 8.0 shows the endpoint trajectory from a release point to a point where the mechanism changed direction. The arrowhead shows the direction of motion. Note that the desired behavior was achieved reasonably



NFFIC endpoint trajectory for the zero velocity feedback test.

Figure 8.0



Digital linear PVF endpoint trajectory for the zero velocity feedback test.

Figure 8.1

well.

A plausable explaination for the small deviation from a straight line the during the first quarter of the move was the manner in which the endpoint was released. The endpoint was held by the author so that small motions could have existed upon release. These motions would be initial conditions on the larger subsequent move of the mechanism and might have led to the small deviation from predicted straight line.

When left to oscillate, the endpoint of the mechanism began to orbit in nearly straight lines about its equilibrium position. This behavior could be expected since any stiffness and or damping intrinsic to mechanism would disturb the system. The model of the system used to derive the NFFIC algorithm, for this research, assumed zero intrinsic stiffness and damping. Friction did exist in the motors and in the bearings. A source of intrinsic stiffness was the cable used for the force sensor and potentiometer signals exiting from the rear of link #1.

A comparision was made with the linear PVF controller for the same experiment as described above. Figure 8.1 shows the endpoint trajectory for commensurate gains and release point for the PVF controller. This comparision clearly shows the action of the NFFIC algorithm.

The actual static stiffness achieved was tested for various points in the mechanism's workspace for a desired mass of 1.5 lb_m. A uniform static stiffness of 2 lb_f/ft was specified. This experiment was set up with the mechanism regulating at an equilibrium position with zero desired damping. The barrier was then brought in contact with the edge of the proof mass. One half inch blocks of aluminum were then placed between the barrier and the

edge of the proof mass to provide incremental displacements. For each displacement, force sensor signal were recorded so that an enpoint force to displacement table could be compiled. Static stiffness was then compiled from this data. Table 8.0 summarizes the results.

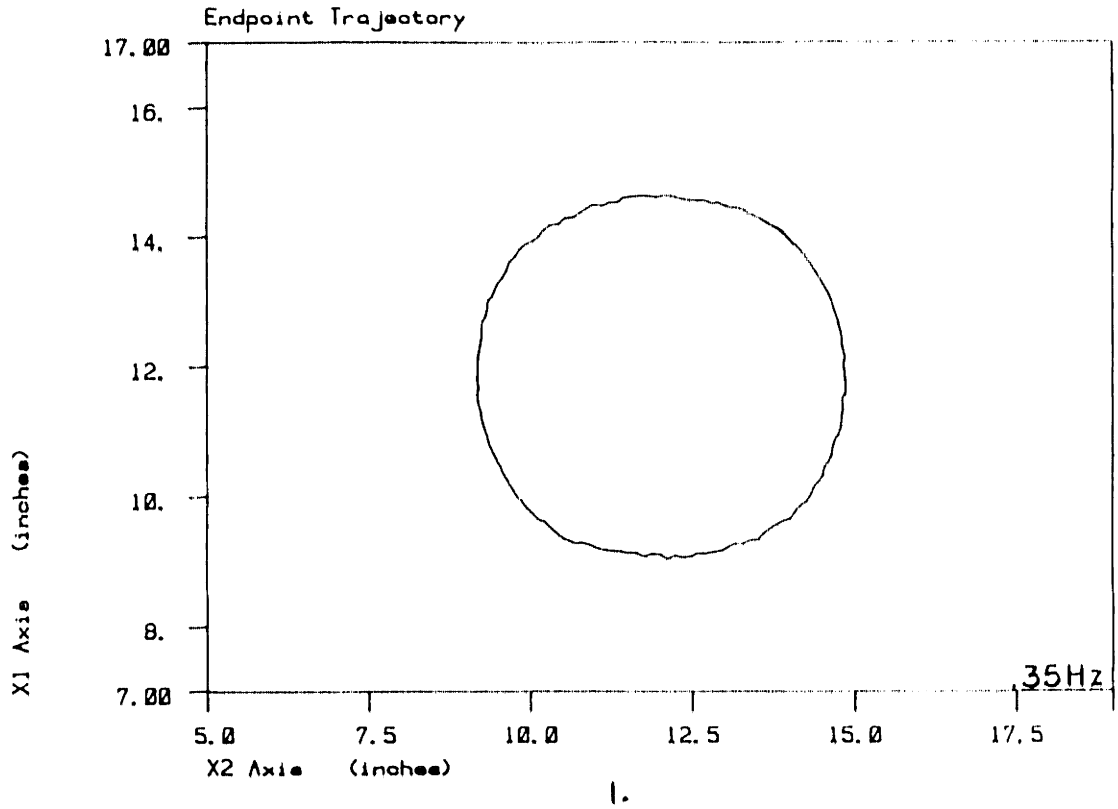
The X1 axis and X2 axis achieved stiffnesses, shown by table 8.0, were in error by a maximum of 4.5% relative to the desired stiffness. There was only 3% difference in uniformity when the stiffnesses along each axis were averaged, over the three endpoint positions, and compared. These results were judged valid proof that the algorithm acheived the desired stiffness.

The NFFIC torque control law did accomplish path tracing. The circle with barrier test in chapter 7 showed part of the commanded cicular trajectory when the mechanism was unconstrained by the barrier for desired mass equal, greater than, and less than actual mass. Case 1 of figure 8.2 is a trace of the endpoint traversing a circle with the mechanism free to move in its workspace. The desired system parameters were 3 lb_f/ft stiffness, unity damping ratio, 1.3 Hz natural frequency, and 1.5 lb_m desired mass along each axis (X1, X2). The small amplitude motions were most likely an artifact of the plotting routine, as discussed in chapter 4. The circle was closed in 2.88 seconds giving a linear speed of 6.54 in/s. Endpoint positions were commanded in absolute cartesian coordinates, (X1, X2), and executed with no inverse kinematics into relative joint angles, (Θ_1,Θ_2) by the NFFIC torque law. The circle program set, without centripetal and acceleration terms, was used to produce the trajectory.

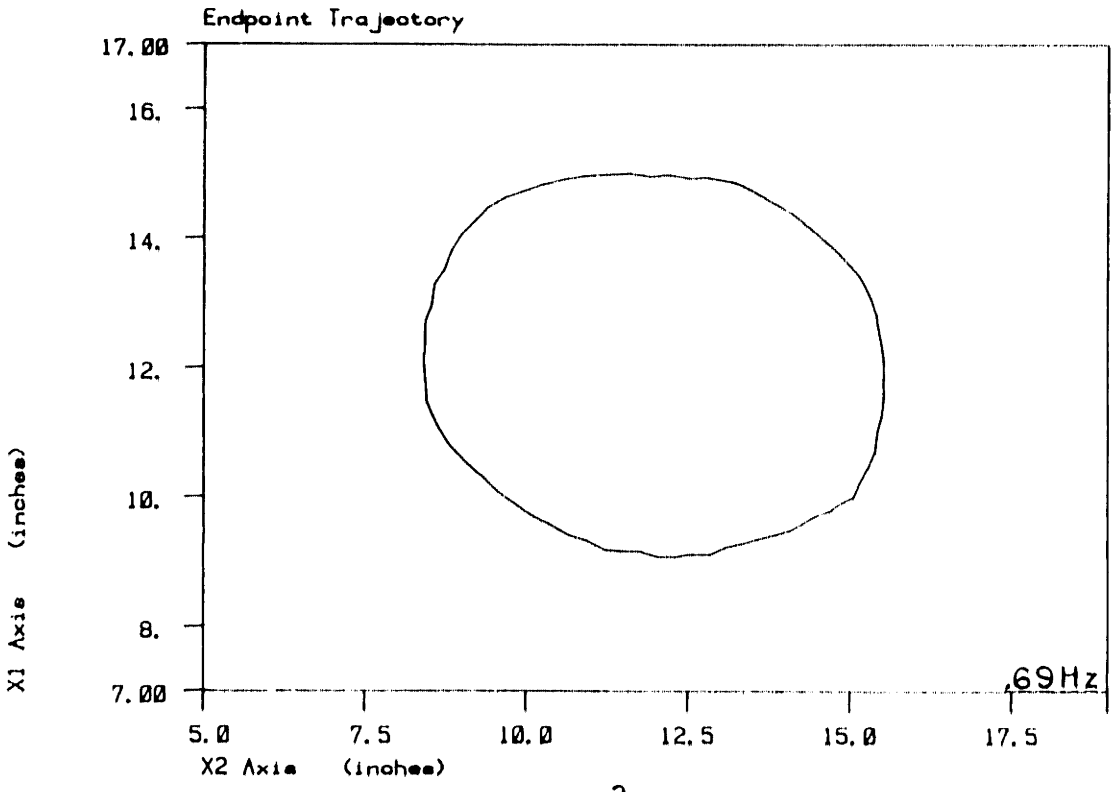
By speeding up the rate at which the circle was traversed one could test in the frequency domain the effectiveness of imposing a second order system

Endpoint Location (inches)		ActualStiffness	
X1 Axis	X2 Axis	X1 Axis	X2 Axis
12.0	12.0	1.95	2.04
10.0	6.0	2.09	2.04
10.0	15.0	1.95	2.09

Static stiffness test results.
Table 8.0



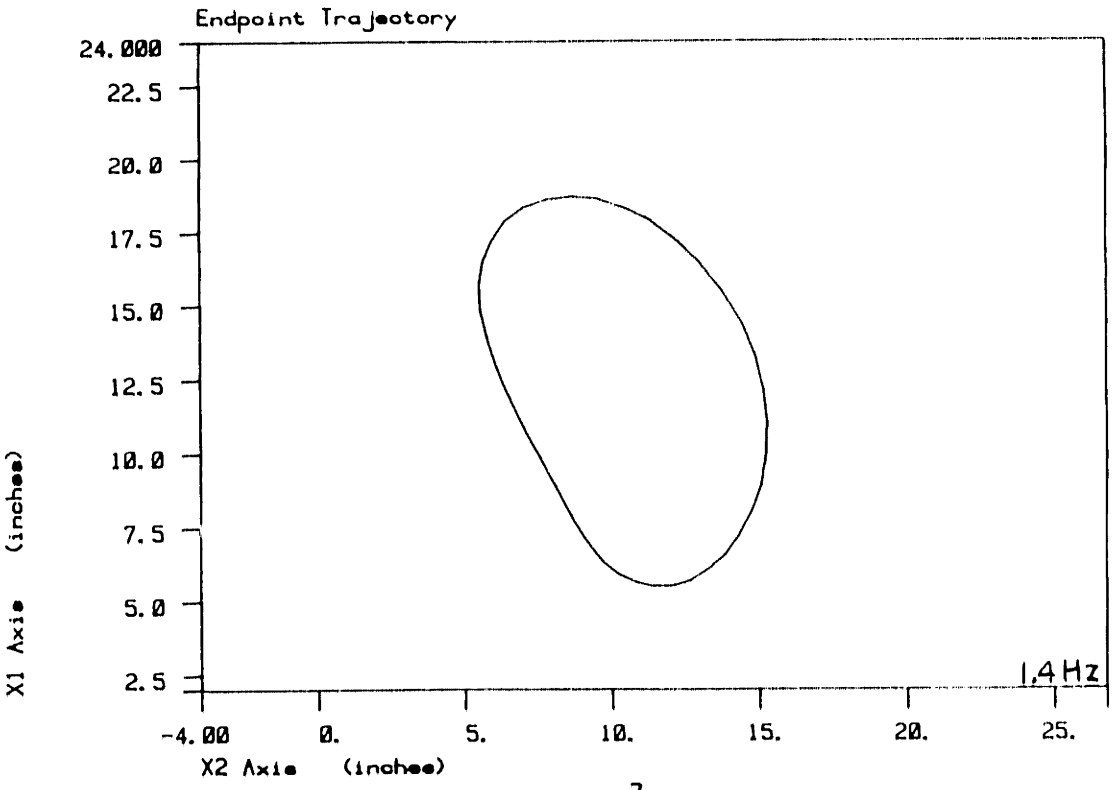
1.



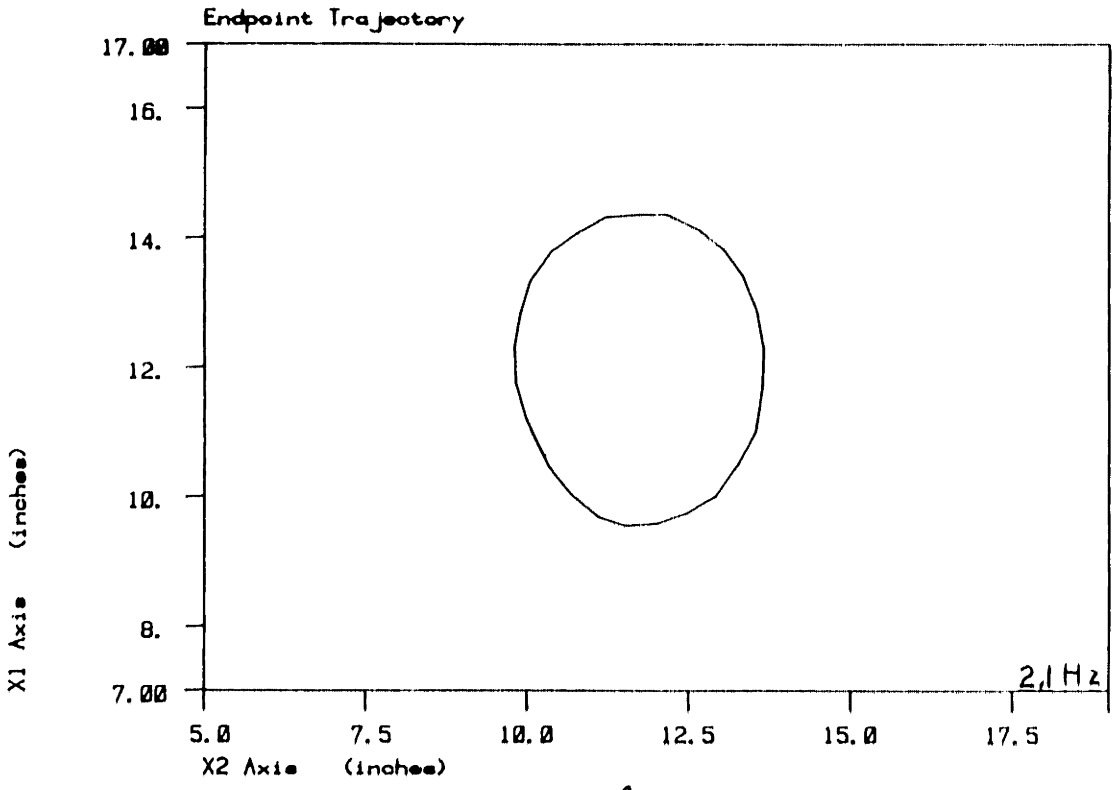
2.

Unconstrained mechanism circle trajectories.

Figure 8.2A



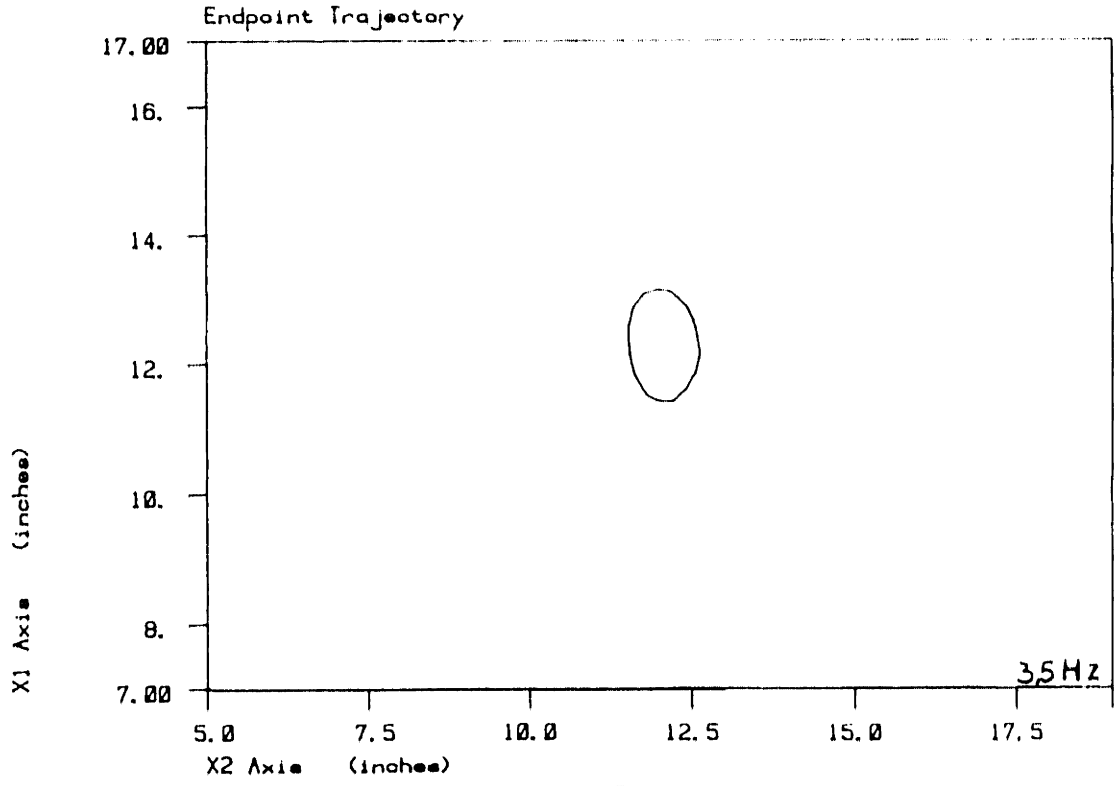
3.



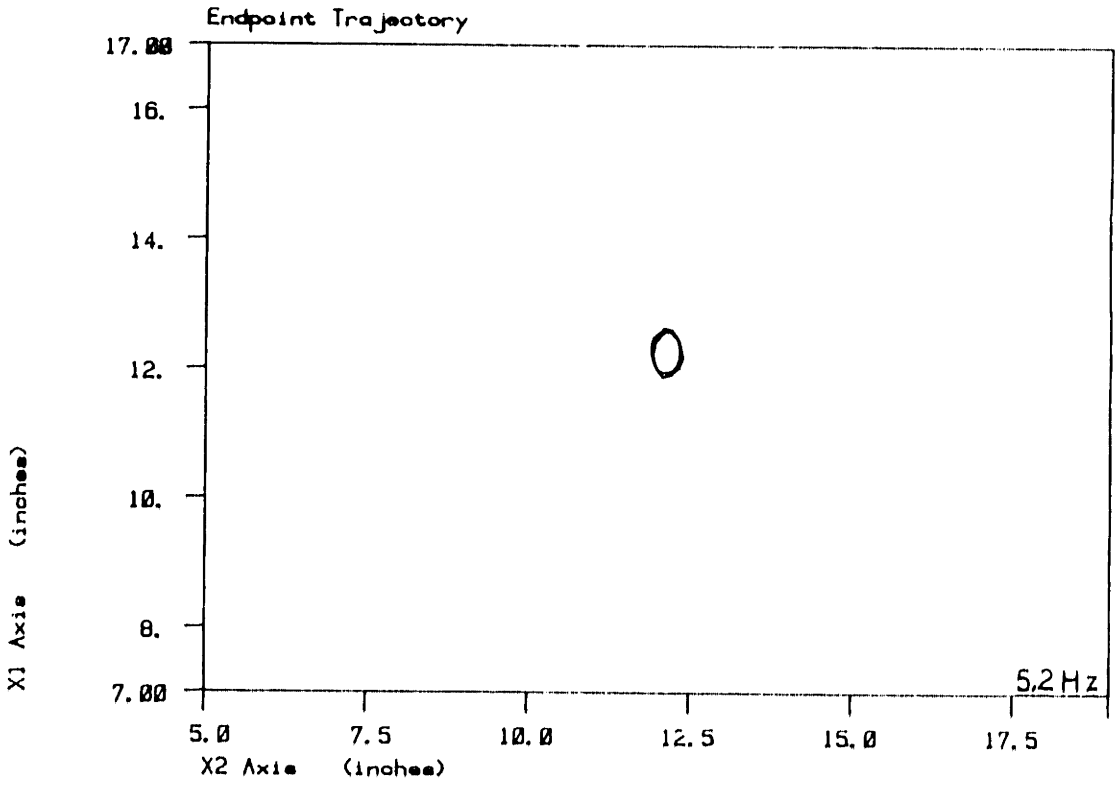
4,

Unconstrained mechanism circle trajectories.

Figure 8.2B



5.



6.

Unconstrained mechanism circle trajectories.

Figure 8.2C

on the actual system. The desired behavior, in one degree of freedom, can be written as:

$$M_d S^2 X + B_d S X + K_d X = K_d X_c \quad . \qquad \text{eq. 8.0}$$

Rewrite equation 8.1 so that a transfer function appears:

$$\frac{X}{X_c} = \frac{\omega_n^2}{S^2 + 2 \zeta \omega_n S + \omega_n^2} \quad . \qquad \text{eq. 8.1}$$

Equation 8.2 can be interpreted as saying that at low circle traversing frequency the commanded and actual positions should coincide. For high frequency the actual radius of the commanded circle should collapse to zero. For intermediate frequencies the radius of the actual circle should grow or shrink, depending on which side of the resonant peak the test frequency is at and the targeted damping ratio. By trying several frequencies one gets a Bode plot that should look like a linear second order system.

Figure 8.2 is a composite of increasing frequency circle trajectory tests. The desired system parameters for all cases were the same as case 1. The frequency of completing each circle is stated in the lower right hand corner of each plot. Case 1 was a low frequency (.35 Hz) test. The highest frequency shown (5.2 Hz) is case 6.

Ideally, all trajectories should be circular; however, small deviations of actual stiffness, damping and/or mass from the desired values would result in an elliptical shape such as is seen in case 2. However, in case 3 the trajectory is neither circular nor elliptical. This deviation from desired behavior may be attributed to either the absence of the centrifugal acceleration terms in the control algorithm or to the stiffness of the cord

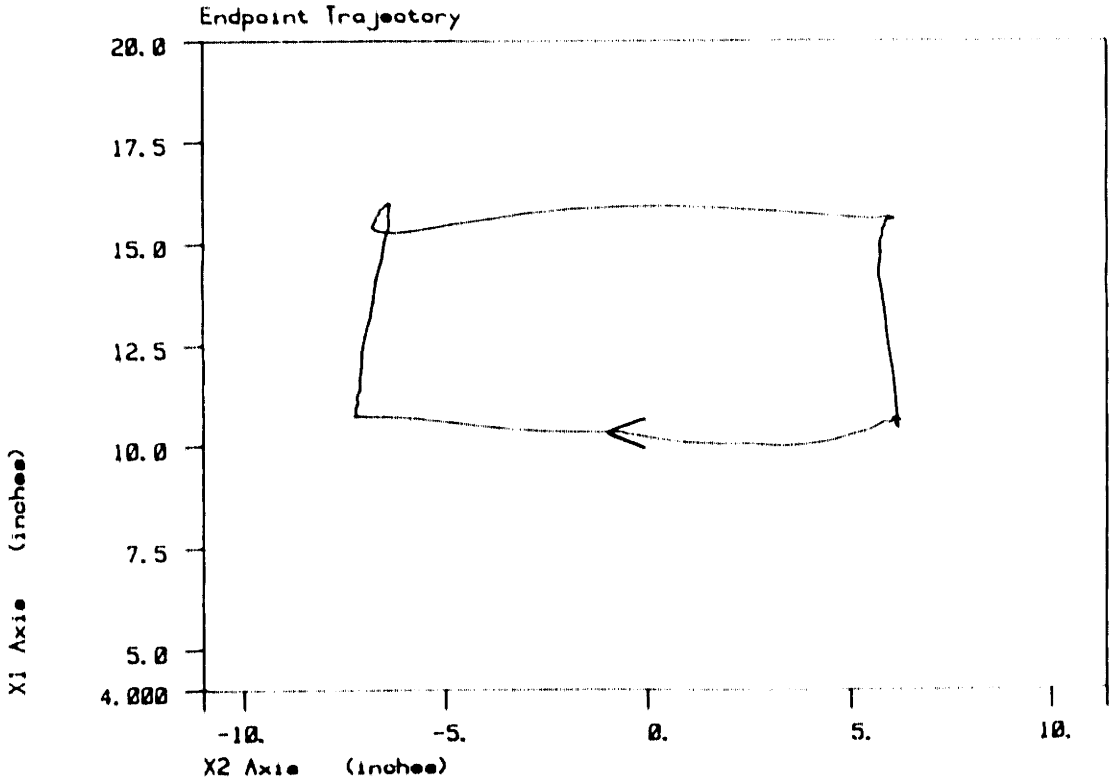
exiting from the rear of link #1. As the desired stiffness was only 3 lb_f/ft, for large moves, such as in case 3, the cord could have disturbed the system enough to account for the anomalous shape.

A resonance occured near 1.4 Hz (case 3). The amplitude of motion in case 3 is larger than in any of the other cases (note the change in scales). This was not as expected since the desired damping ratio was unity. For frequencies higher than case 3 the amplitude of the motion shrinks. For higher frequencies than shown in case 6 the endpoint spiralled in to an equilibrium point at the center. The frequency domain data presented in figure 8.2 describe a second order system, though with parameters different from those desired.

Another experiment to assess the achieved system behavior was devised. In this test the mechanism was commanded to move between four endpoint positions that described a rectangle of twelve inches by six inches. The desired system parameters were 3 lb_f/ft stiffness, unity damping ratio, 1.3 Hz natural frequency, and 1.5 lb_m desired mass. The twelve inch move occurred in approximately 0.2 second. The endpoint trajectory is shown in figure 8.3.

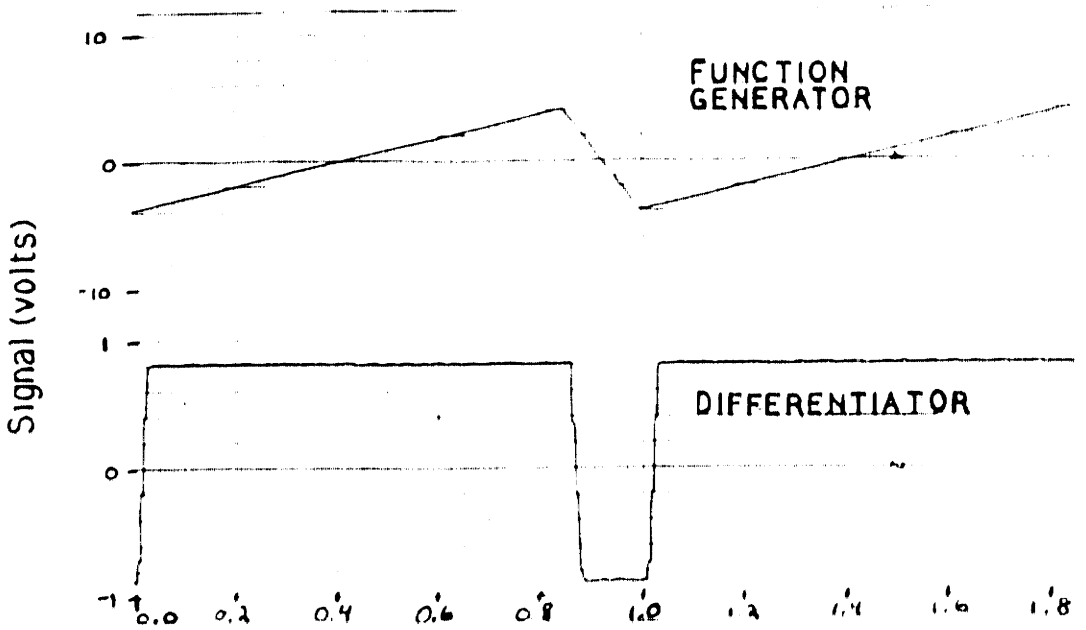
Note that just as in the circle test, overshoot occurred when there should have been none, since the desired system was critically damped. The trajectory from point to point deviates from the expected straight line. What could have caused these errors?

Cotter [7] has shown by simulation that impedance control algorithms are robust to errors in link inertia calculation. The estimate of the link inertias by the sectional method was considered accurate enough that further analysis for errors in the inertia tensor was not warranted. Since desired



NFFIC endpoint trajectory for the rectangle test.

Figure 8.3



Differentiator response to a sawtooth input.

Figure 8.4

damping and the differentiation scheme for velocity feedback were linked, an analysis of the differentiation scheme was performed.

8.3 Discrete differentiation revisted.

The discrete differentiation scheme used in the NFFIC programs consisted of two parts. The first part was a first order difference between the current position and a position from three sampling periods past. The motivation for using this particular difference equation was covered in chapter 5. The second part of the discrete differentiation scheme was smoothing of the first order difference result. The smoothing was accomplished by adding the current difference result to the difference result from one sample past, and dividing by two.

The first part of the differentiation scheme can be written in the discrete time domain as shown by equation 8.2.

$$D_k = \frac{X_k - X_{k-3}}{\Delta t} \qquad \text{eq. 8.2}$$

Equation 8.2 can be rewritten as a transfer function using Z transforms; equation 8.3.

$$\frac{D(Z)}{X(Z)} = \frac{Z-1}{Z\Delta t} \qquad \text{eq. 8.3}$$

Does equation 8.3 describe a differentiator (without smoothing)? Equation 8.3 can be rewritten for the continuous time domain (for sine inputs) and expanded as a polynomial using a Taylor series :

$$\frac{D(S)}{X(S)} = \frac{1 - e^{-S\Delta t}}{\Delta t} = \frac{1 - (1 - S\Delta t + S^2\Delta t^2/2! - + \ldots)}{\Delta t} \qquad \text{eq. 8.4}$$

which in the limit, as Δt approaches zero, is equal to S, an ideal differentiator.

The phase relationship of equation 8.4 is :

$$\Phi = \tan^{-1}\left(\frac{\sin \omega\Delta t}{1 - \cos\omega\Delta t}\right) . \qquad \text{eq. 8.5}$$

A table was generated using equation 8.5 to find the bandwidth of the first order difference for which the phase error was no more than 10 degrees. The result was that the first order difference was phase accurate to 7 Hz. Did the moves shown by figure 8.3 exceed the phase accurate bandwidth of the first order differentiator?

If one considers the time for one move as one half of a full cycle, the RMS value of the total cycle time is on the order of .29 seconds. This cycle time is a frequency of 3.5 Hz, well within the phase accurate bandwidth of the first order differentiator. The 3.5 Hz estimate made intuitive sense. Given that the controller has an approximate bandwidth of 5.6 Hz, and that the longest move in figure 8.3 was substantial compared to the largest possible workspace of the mechanism (greater than 40%), one would not expect the mechanism to exceed a 7 Hz move. So far, the differentiation scheme does not appear to account for the unexpected behavior as shown by figure 8.3.

The second part, of the overall differentiation strategy, was the smoothing of the first order difference result. This averaging scheme is shown by equation 8.6.

$$V_k = \frac{D_k + V_{k-1}}{2} \quad . \qquad \text{eq. 8.6}$$

The intent of equation 8.6 was to filter the first order difference result. The filtering was first order since the averaging took place over one time step. An equivalent continuous time filter has the form shown by equation 8.7.

$$\frac{V(S)}{D(S)} = \frac{\lambda}{S + \lambda} \quad . \qquad \text{eq. 8.7}$$

Finding the break frequency of this filter might yield more insight into the overall differentiator response. Equation 8.7 can be rewritten as shown by equations 8.8 and 8.9

$$\frac{V_k(t + \Delta t) - V_k(t)}{\Delta t} = -\lambda V_k(t) + \lambda V_{k-1}(t) \qquad \text{eq. 8.8}$$

$$V_k(t + \Delta t) = (1 - \lambda\Delta t)V_k(t) + \lambda\Delta t V_{k-1}(t) \qquad \text{eq. 8.9}$$

Comparing equation 8.9 to equation 8.6 one can solve for λ; $\lambda\Delta t = 0.5$. For a 9 ms sampling period λ is 8.8 Hz. The time constant should then be .018 seconds.

A test was devised to show empirical evidence of this time constant. The differentiator was configured to operate on a one Hz sawtooth wave from a function generator. The sawtooth and differentiator response are shown by figure 8.4. The differentiator time constant, from figure 8.4, was .020

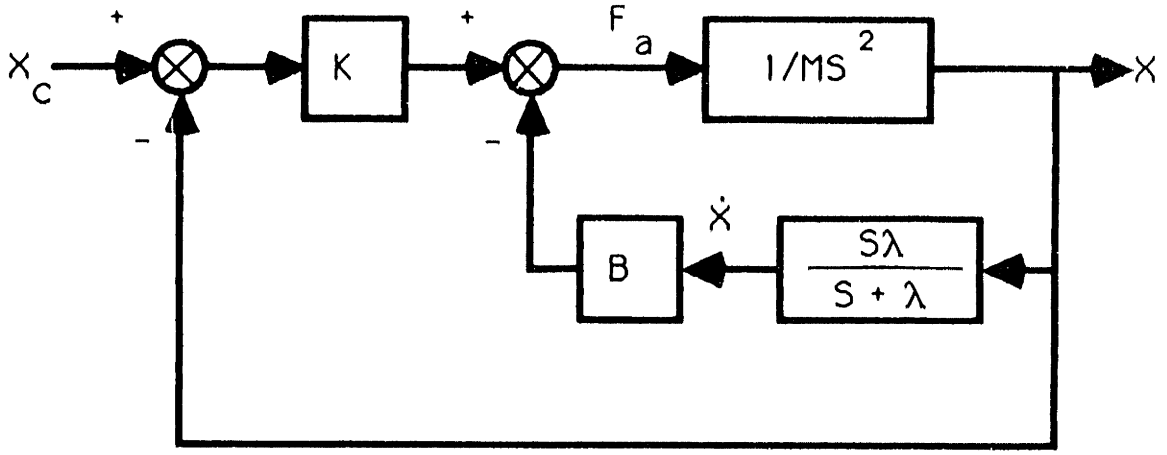
seconds which confirmed the analytical estimate.

What effect could the actual full differentiation scheme have on the controlled system given that an ideal differentiator was assumed for the NFFIC law derivation? Figure 8.5 is a block diagram of a one degree of freedom position and velocity feedback controller acting on a mass. Note the full differentiator, as described in this section, used in the velocity feedback loop. The transfer function of the inner loop takes the form shown by equation 8.10.

$$\frac{S + \lambda}{MS[\, S^2 + S\lambda + (B/M)\lambda \,]} \qquad \text{eq. 8.10}$$

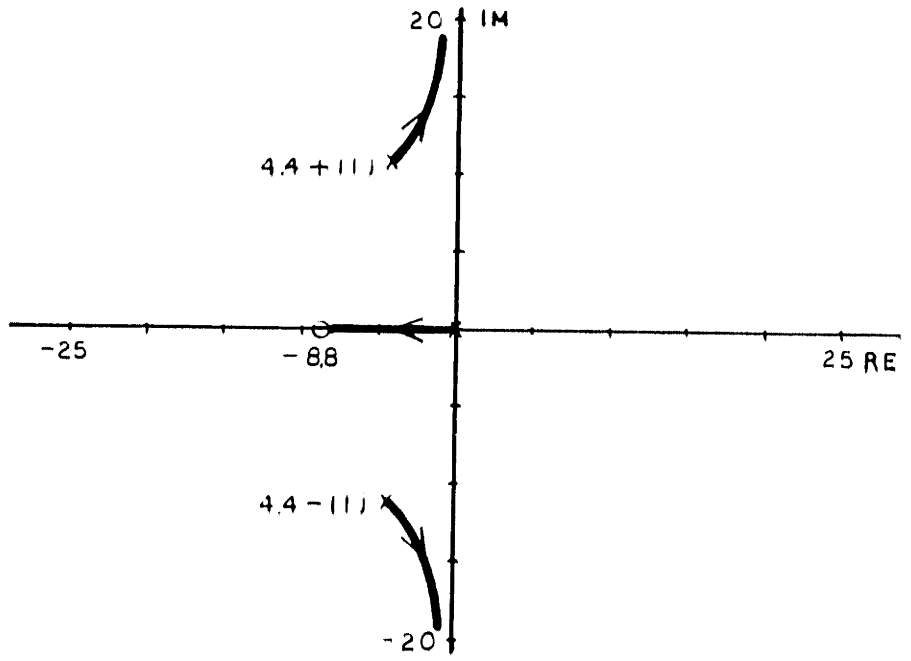
The poles and the zero of equation 8.10, when solved using the desired sustem parameters of the test shown by figure 8.3, are the "X's" and the "O" in figure 8.6. When the outer position loop is closed about equation 8.10, the resulting root locus shown by figure 8.6.

Figure 8.6 shows that whereas the nominal behavior would be characterised by two identical real poles, in fact as position feedback gain is increased the pole at the origin approaches the zero at 8.8 Hz (and the two tend to cancel one another) and the two oscillatory poles become even more oscillatory. The effective damping ratio is considerably less than the desired damping ratio of unity. This reduced damping ratio could certainly result in the overshoot and the curved trajectories of the system shown in figure 8.3. It is also consistent with the resonance seen in figure 8.2. If this was the case, then eliminating velocity feedback should eliminate deviations from desired behavior. Figure 8.0 shows that for <u>zero</u> velocity feedback the achieved behavior was close to the desired behavior.



Position and velocity feedback control using the differentiator.

Figure 8.5



Root locus showing the effect of the differentiator.

Figure 8.6

8.4 NFFIC performance evaluation.

The 111 Hz sampling rate achieved by the full point to point MACRO-11 control program yields an approximate controller bandwidth, using equation 4.6, of 5.6 Hz. Controller bandwidths of less than 10 Hz are common [30], [50]. Given that more efficient computers exist, one can conclude that the NFFIC torque control law was computationally not a great burden.

The option to switch from 3 to 2 significant figures accuracy, provided in the control program code, was only exercised once. Endpoint motions were far smoother for 3 significant figures than 2. The range of gains was great enough with the 3 significant figure option that a variety of experiments could be performed.

8.5 NFFIC system relative performance.

Limits of desired mass, stiffness, and damping were explored. In most cases the 16 bit integer arithmetic limited higher gains due to severe saturation. Saturation of the command voltage resulted in the D/A's wrapping back to 0 V. Saturated moves were typically erratic, or the mechanism seeked a position that corrosponded to a voltage that had gone past ±10 V and started over from zero. The figures in tables 8.1 and 8.2 were for maximum desired system parameters that did not result in severe saturation. A unity damping ratio was chosen so that the possible combinations of desired mass, stiffness, and damping were fewer. Moves were commanded using the point to point NFFIC software set.

Target parameter X1 and X2 axes	Value	Units
Mass	1.0	lb_m
Stiffness	6.0	lb_f/ft
Damping	0.84	lb_f s/ft
Natural frequency	2.2	Hz
Damping ratio	1.0	none

Maximum system parameters for large moves (12 in.).

Table 8.1

Target parameter X1 and X2 axes	Value	Units
Mass	1.0	lb_m
Stiffness	9.1	lb_f/ft
Damping	1.03	lb_f s/ft
Natural frequency	2.7	Hz
Damping ratio	1.0	none

Maximum system paramters for small moves (1 in.).

Table 8.2

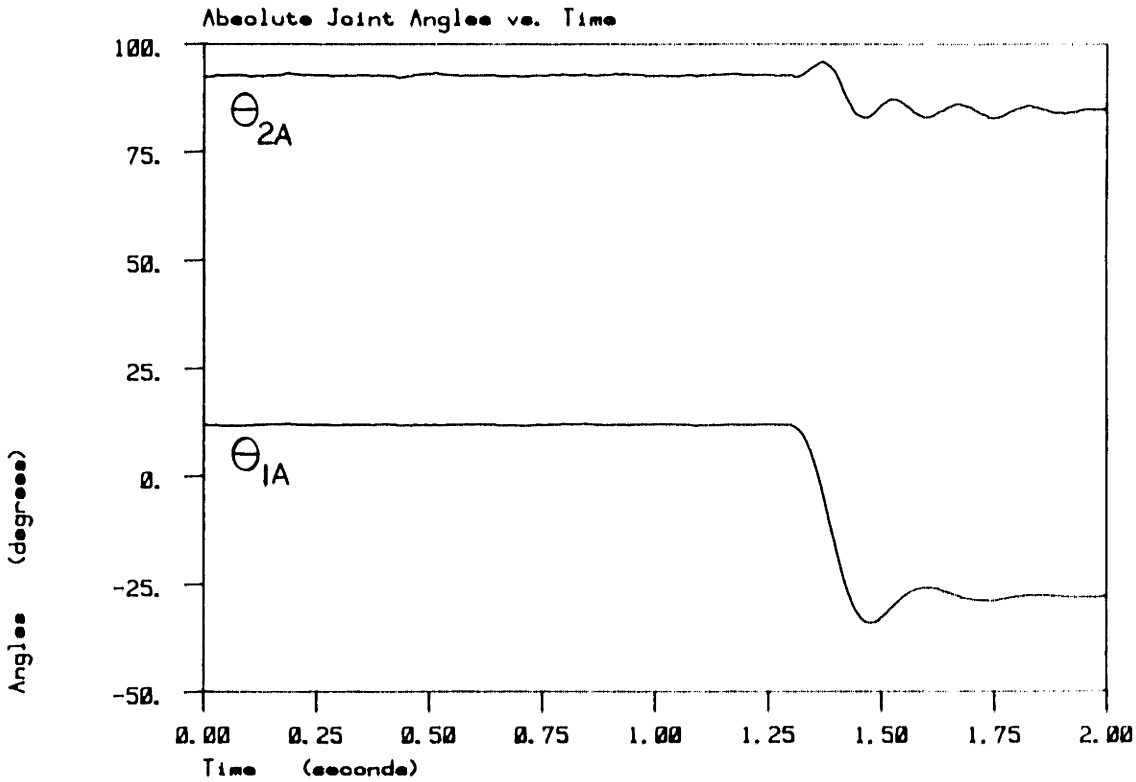
Was NFFIC tested rigorously? Figure 8.7 is a plot of the absolute joint angles for an eight inch move. The desired system parameters were 5.5 lb_f/ft, unity damping ratio, and 1 lb_m desired mass. From figure 8.7 the peak angular velocity and acceleration of the mechanism were estimated. Using a radius of twelve inches, the length of one link, an estimate of endpoint speed and acceleration was made. The two link mechanism's peak endpoint speed and acceleration was then compared with industrial robots in use today and two robots in laboratories at MIT. The result is shown in table 8.3.

The main point to showing table 8.3 is that NFFIC was being tested under conditions that come close to expected real world performance. It was difficult to assess the maximum endpoint speeds and accelerations of commercially available robots since no standard of comparision existed at the time of this writing. The figures presented here are best estimates from manufacturer's specifications and from discussions with professors in the Laboratory for Manufacturing and Productivity and the Department of Mechanical Engineering at MIT.

Manipulator	Speed (in/s)	Acceleration (g's)
Commercially available robots	40 - 60	0.1 - 0.8
Asada	390.0	5.3
Seering	200.0	4.0
NFFIC two link mechanism	100.0	3.3

Comparision of peak velocity and accelerations.

Table 8.3



Absolute joint angle time history.

Figure 8.7

Discussion and Conclusions

9.1 Scope of this chapter.

This chapter discusses the highlights of the experimental results. Suggestions for future research are proposed. The main points of this thesis are then summarized.

9.2 Discussion of the analytical and experimental results.

The meaningful result of figure 7.9 is that there exists a much larger margin of increased desired mass than decreased. A larger margin translates to greater amount of tasks that can be dealt with stably. All of the dynamic modelling and probing into scaling and inertia tensor errors was done on account of the constrained negative force feedback instability. The posititive force feedback regime appears to be more flexible.

Filtering the force feedback signal to stabilize the negative force feedback instability does not appear as good an alternartive as adding intrinsic damping to the force sensor. As the break frequency of a filter decreases the poles of the filter move right. The dominant response of the mechanism to force commands then emulates the filter whose pole configuration dictates an increasingly sluggish response. This analysis suggests that an approach to stable negative force feedback control is to weigh the benefits of being able to reduce desired mass more, at the expense of system performance, by using a force feedback filtering scheme. Adding intrinsic damping did not affect system performance as filtering the force signal did. In fact, it can be argued

that adding damping might increase system response in this case. The poles related to the force sensor would approach the real axis along an arc of radius equal to their natural frequency. This translates to a decreased system time constant; faster system response. Since this analysis treats negative force feedback in a general manner it may be applicable to the experiments in [44] that report filtering results similar to those obtained here.

Figure 8.2 demonstrates trajectory tracing without inverse kinematics computations. This is an important result since an inverse kinematics formulation is not guaranteed to exist for a particular manipulator [40].

Alternatives to digital differentiation should be pursued in light of the data presented in chapters 5 and 8. The computational complexity of the NFFIC algorithm is on the order of fast nonlinear controllers in current literature. Table 2.0 shows that the relative computational burden of NFFIC compares favorably to other control algorithms. The fact that NFFIC ran successfully in two degrees of freedom using a general purpose computer lends evidence to the claim.

In general, the results in chapters 6,7, and 8 prove the concept of impedance control. Force feedback is shown to modulate apparent mass of a low intrinsic impedance system.

9.3 Suggestions for future research.

Future impedance controller evaluations should be performed on direct drive robots or manipulators designed to do useful tasks. The two link mechanism in this work fulfilled its purpose of acting as a testbed to uncover issues in NFFIC, however it should not be used to seek performance limits of NFFIC. A suggested approach is to implement impedance control without force

feedback on a direct drive robot and concurrently run NFFIC in full on a conventional drive robot. The issue of interface force control with or without electronic inertia control (force feedback) might then be evaulated.

Experience gained in this work has shown that 16 bit integer arithmetic imposes undesirable constraints on controller experiments. A 32 bit floating point processor appears to be the correct route. A floating point A/D and D/A combination [35] would greatly improve theoretical endpoint accuracy. For quick two degree of freedom tests of impedance control variations, FORTRAN code seems adequate. The relative ease of higher level language coding versus machine code well justifies the slower sampling rate. FORTRAN code written on a PDP 11/44 with standard means of speeding up calculations could perhaps run NFFIC at 15 ms per cycle or less.

An interesting variation on equation 3.13, not implemented in this investigation, can lead to fewer calculations. If at equation 3.x one solves directly for the actuator torque command the result is:

$$\Gamma_a = [J\,I^{-1}]^{-1}\{M_d^{-1}[K_d(X_c - X) - B_d V - F_e] - H\} + C + F_e \quad . \qquad \text{eq. 9.0}$$

This NFFIC law has 516 multiplications and 468 additions for six degrees of freedom. Compare this with table 3.0. The relative complexity is nearly half that of the NFFIC law used in this investigation. Equation 9.0 could be encoded to run on the existing two link mechanism to evaluate software complexity and sampling rate.

Section 2.4.1 showed how an impedance controlled system is a type zero system. The advantage of such a system is that it is stable for step inputs. A drawback is that stiffness, and ultimately positioning accuracy, depends on

loop gain. Impedance controlled mechanisms require high stiffness for positional accuracy like any position feedback controlled device. As with manipulators today, infinite stiffness may not be required since tolerances can be specified. Alternatives such as micromanipulators might also relax loop gain requirements. The advantage of impedance control is that gains have physical meaning; mass, stiffness, and damping. A desired system can be tailored to suit the task, hence "tunable" impedance [21]. Targeting a linear second order time invariant system simplifies stability and performance calculations. Response to disturbances, the motivation behind interface force control, can be governed because the system impedance is known. All of these advantages are a direct result of viewing the interactions between a manipulator and its environment in a causal manner.

An interesting electronic yet continuous (analog) implementation of an impedance controller is reported by [1]. This approaches uses analog components to close the feedback loops for a one degree of freedom elbow prosthesis. A computer can still set gains via multiplying D/A's. This novel approach could combine the good impact response of a continuous scheme with the flexibility and potential error reduction of a discrete scheme.

One wants stability during impacts so that objects can be grasped without pausing. Humans speed up tasks naturally by picking up objects while maintaining velocity. Again, impedance control could lessen the loading on impact by specifying a compliant and moderately damped system response. Finally, the notion of combining an intrinsically variable impedance micromanipulator to a purely electronically controlled host robot appears as another alternative to the impact responce versus the steady state error problem [27].

An underlying theme throughout this work was the matching of mechanism to task. The mechanical design of mechanisms that are to be controlled by electronic means should include constraints that are due to the proposed control scheme and task. The idea of a completely multitask robot is fading, hence paint spraying "robots" and welding "robots". These devices are designed for the task at hand.

Impedance control seeks to control a manipulator such that it works in unison with its task. An analogy is the seemingly jerky, inefficient motions of today's assembly robots contrasted to the fluid motions of a human performing the same task. Often, a robot today must be large and designed to be very stiff so it can overpower the load it intends to work on. An impedance controlled manipulator could be designed to be lighter since it is the load that dictates motions while the forces arising from planned trajectory and actual trajectory are controlled.

Direct drive robots are especially suited to impedance control. Their low open loop impedance is ideal for tuning up a desired impedance. The concept of impedance control has been tested and analyzed in [7], [27], [2], and in this work that the next logical step appears to be to evaluate performance on a direct drive robot.

9.4 Conclusions.

It appears that the key to stability of negative force feedback (reduced apparent mass) control laws is the relationship between mechanism and force sensor dynamics. The very device that is used to sense force bears on the force control stability. The data suggests that dynamics do not play such a crucial role in positive force feedback (increased apparent mass) control laws.

The well documented stability enhancing effect that a compliant force sensor has on a negative force feedback loop [44], [53] was confirmed and extended.

Adding intrinsic damping to force transducers appears to improve the mass reduction stability margin for constrained negative force feedback controllers. Filtering the force signal can improve the stability margin at the cost of system performance. An experiment to study the tradeoffs between intrinsic damping, filtering, and system performance versus stability is suggested.

Future work should concentrate on evaluating NFFIC on mechanisms that are designed for accuracy. Multi degree of freedom applications of NFFIC in code is suggested for machines with 32 bit floating point capacity. Hybrid discrete and continuous control schemes ought to be pursued so as to reap benefits from both regimes.

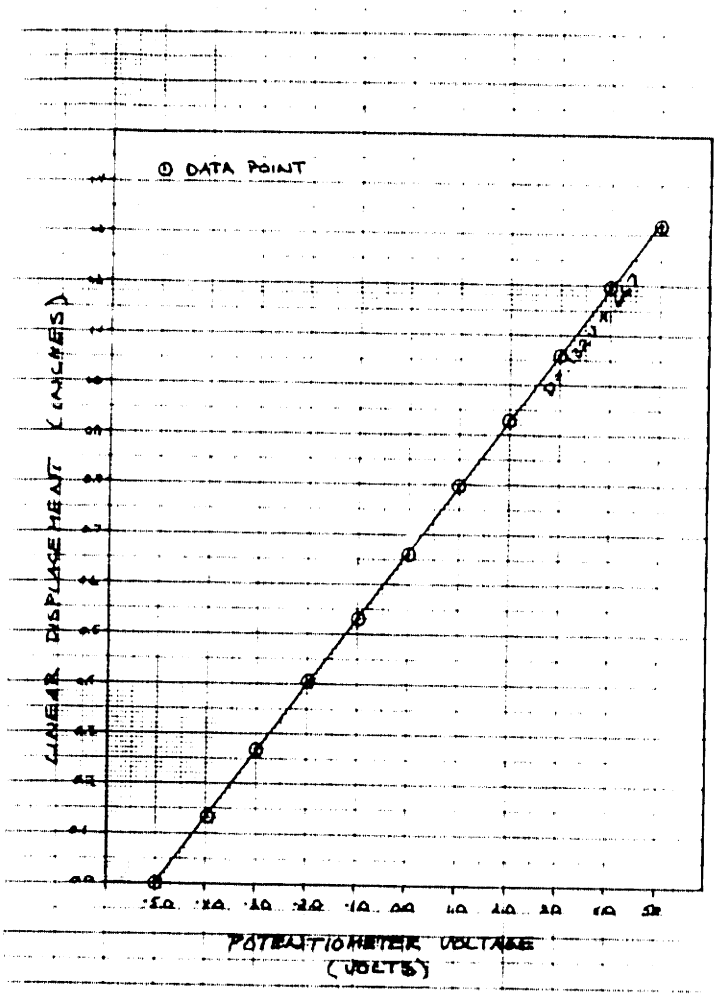
Appendix I

The intent of the three experiments listed below was to get empirical data for constants used in scaling factors. The stated linearity of the devices was also investigated.

The first experiment focused on the conductive plastic potentiometers. The manufacturer claims a deviation from linearity of no greater than .25%. Cotter [7] conjectured that this may not be the case.

A collar was connected to the shaft of the potentiometer to increase its radius. A 28 gage wire was connected at a point on the coller and kept taut horizontally by draping the wire over a pulley and hanging a weight on the end of the wire. A small piece of tape on the wire served as a flag for measurements. A piece of paper underneath the horizontal portion of the wire was used to record succesive positions of the flag as the wire wound up on the collar of the potentiometer.

The shaft of the potentiometer was turned to create 1 volt increments. At each voltage the flag on the wire pointed out a new distance on the ruler. A chart of linear distance vs. voltage was compiled and plotted. The result is figure A1.0. A least squares curve fit was performed on the data points yielding the straight line. Error bands for either axis, based on 1/2 of the smallest reading possible on the ruler and voltmeter, were too small to show up on the diagram. The curve fit and data have a correlation coefficient of .9999. From this data one can conclude that there is no evidence to support nonlinear behavior greater than .25%. The same test was performed on the other joint potentiometer with similar results.



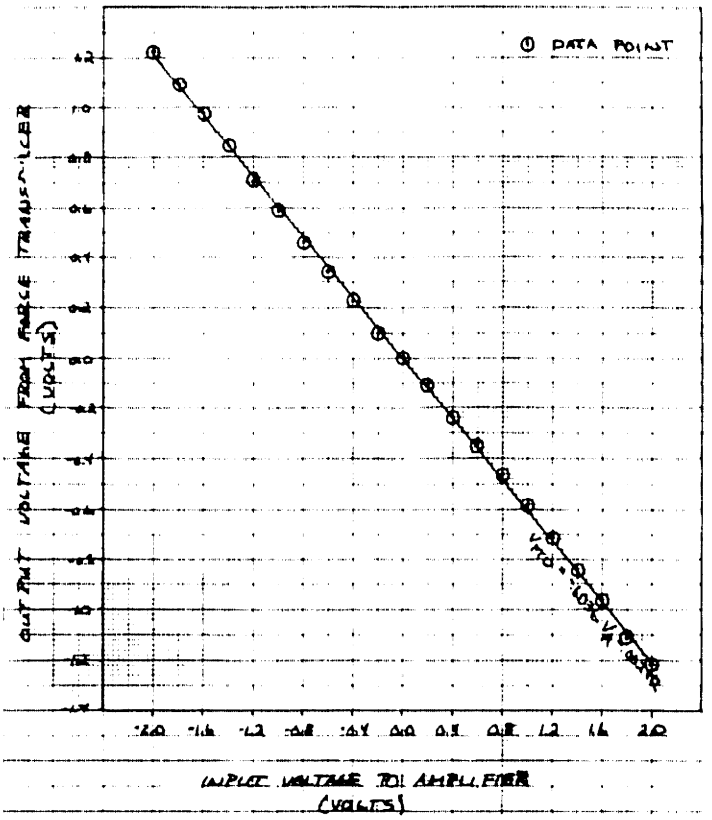
Potentiometer voltage versus linear displacement.

Figure A1.0

The next test measured the force transducer linearity and force to voltage constant. The setup consisted of the end of link #2 against a face of the barrier with the proof mass removed. A small wire knotted .05 inches from the top of the force sensor stalk was kept horizontally taut by passing the wire over a low friction pulley with a platform for weights connected to the end of the wire. The wire was kept short to minimize elongation. The pulley was mounted on the barrier near to the spot where the end of link #2 came in contact with the barrier so that relative motion between the pulley and the force sensor was kept to a minimum. Weights were added to the platform to incrementally load the force sensor. Loading the force sensor along a particular axis was achieved by moving the entire barrier with pulley to another location in the mechanism's workspace axis. Loading along only one axis, say the y axis, was done by positioning the barrier so that the x axis voltage output remained zero.

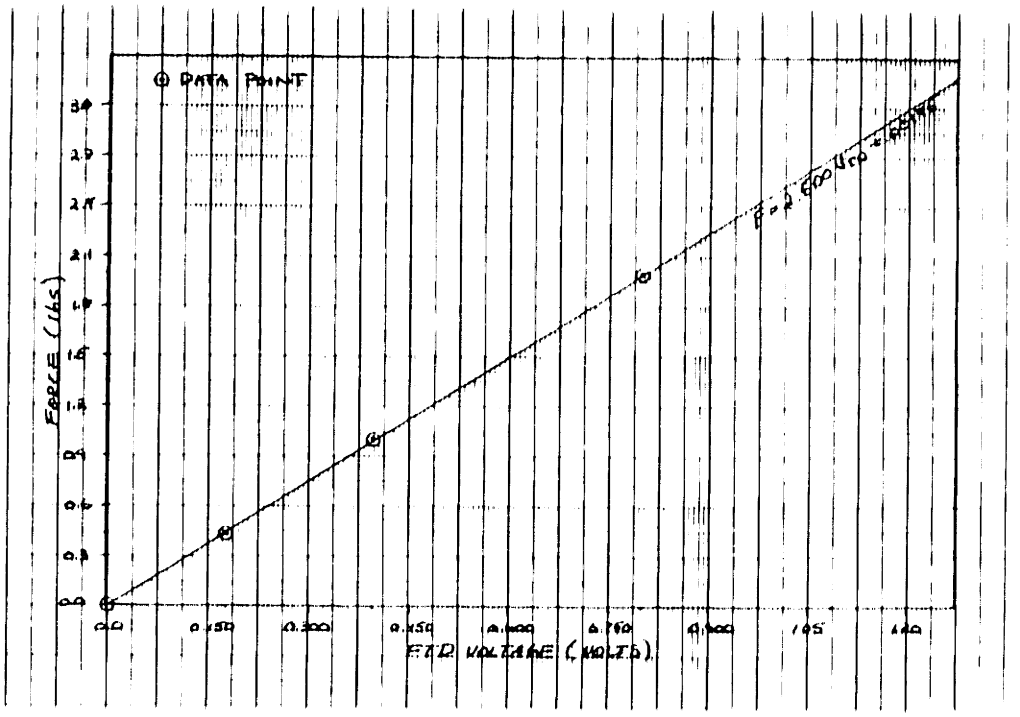
For no load the voltage was noted. Then for known increments of weight voltage readings were recorded. This was repeated for a direction orthogonal to the first. An example of the data compiled for one force sensor axis is shown by figure A1.1 A least squares curve fit of the data yields the straight line. The correlation coefficient was .9999.

To compute the sensor stiffness, linear displacement of the tip was needed. The transducer was mounted vertically on the bed of a milling machine. A short steel rod in the jaws of the mill chuck were lined up to hit the tip of the joystick and translate along an axis of the joystick. The limit of travel was determined by visually checking the gap between force sensor stalk and joystick hole. A sliver of paper at the point of contact helped to confirm the visual check by being caught when contact was made. This



Force transducer voltage versus load at the tip.

Figure A1.1



Servo amplifier input voltage versus force transducer output voltage.

Figure A1.2

procedure was repeated for the four directions of joystick travel by rotating the joystick. Linear distances were read off the mill scale. The four results were all within 3% of each other. The average of these four measurements was .040 inches.

From the empirical measurement of stroke and figure A1.1, sensor stiffness and the stop to stop force limit was calculated. The maximum voltage to a stop was known by deflecting the force sensor and recording the maximum voltage seen for one axis direction at a time. Using figure A1.1, read off the force corrosponding to the voltage at the stop using the curve fit. The result was 2.65 lb_f in the +x direction and 3.0 lb_f in the +y direction. The manufacturer specifies a uniform 3.1 lb_f limit stop force.

Since the data in figure A1.1 appears relatively linear, the calculated limit stop force and empirical limit stop deflection was used to calculate the sensor stiffness. Since the limit stop force was different for the two axes of the force sensor, the stiffness calculation resulted in two different values. In the x direction (X2 and x are parallel for the mechanism in the start position) the stiffness was computed to be 700 lb_f/ft. In the y (nominally X1) direction the stiffness was 900 lb_f.

The final constant needed was a measure of endpoint force for command voltage to the servo amplifier. Intermediate scaling factors such as servo amplifier transconductance gain and motor torque were not necessary. Finding this all encompassing constant of output force to command voltage gives greater accuracy than if intermediate values were used.

The setup consisted of the mechanism placed in the start position with the tip of the force sensor constrained from motion along one direction of one

of its axes. The tip of the force sensor was not rigidly attached but allowed to pivot when strained. The contact point on the stalk of the joystick was .05" below the tip. The manufacturer recommends this be the point of contact for force measurement.

The servo amplifier served as a transconductance gain. Only one input channel was used (channel #1). To pair up a servoamplifier and transducer axis, one link was mechanically grounded while the other swung free. The force transducer voltage was recorded at each input voltage increment of 0.2 volt. Each force sensor reading was then multiplied by the inverse of the force transducer scaling factor. A table of voltage command to force out was compiled for each servoamplifier in this manner. Since both links are a foot long each, a force of one pound at the tip of each link translates to a torque of one foot-pound at the motor. In this way the data depicted in figure A1.2 was used to derive the system torque constant.

An internal current balance potentiometer was provided by the manufacturer to zero voltages created within the servoamplifier . The success of nulling could be gauged by a currrent monitor pin provided that gave a voltage proportional to motor current. It was noted that a bias would result with time and fluctuations in temperature. Typically a bias of ±.01 amp occurred. The bias could result in torques of ±.004 lb_f/ft at the endpoint perhaps leading to some asymetry in trajectory.

Appendix II

NFFIC Software Scaling Factors

Continuous Domain

Joint angle.

Potentiometer A/D

$$\frac{\pm 10\ V}{\pm 90^0} * \frac{4096\ units}{\pm 10V}$$

Force.

Joystick A/D

$$\frac{\pm 10\ V}{6.2\ lb_f} * \frac{4096\ units}{\pm 10V}$$

Torque motor/amp #1.

motor/amp sensitivity

$$0.5926\ \frac{ft \cdot lb_f}{V}$$

Discrete Domain

Integer math Algorithm constant

$$1 * 10^5 * \frac{\pm 10V}{4096} * \frac{\pm 90^0}{\pm 10V}$$

Integer math Algorithm constant

$$1 * 10^6 * \frac{\pm 10V}{4096} * \frac{6.2\ lb_f}{\pm 10V}$$

Algorithm command Inverse motor/amp sensitivity

$$\text{1000's ft}\cdot lb_f \text{ torque} * 1.687\ \frac{V}{ft \cdot lb_f} *$$

D/A Integer math D/A offset

$$\frac{4096}{\pm 10V} \quad /1000 \quad -2048_{10}$$

Torque motor/amp #2.

Substitute 0.1748 for 0.5926, and 5.721 for 1.687 in the expression above.

Appendix III

```
C-----------------------------------------------------------------
C
C                          PVFMNP.FOR
C                        JOHN WLASSICH
C                           3/7/85
C
C       A FORTRAN main program which calls PVFMAC.MAC, a MACRO-11
C subroutine.  PVFMAC.MAC uses PVFSUB.FOR, a FORTRAN subroutine.
C This program queries user for feedback gains, sampling rate,
C and prompts to begin or restart MACRO control program.
C
C Variable definitions follow from input queries below.
C
C-----------------------------------------------------------------
```

```
      INTEGER*4 QUEST

      COMMON/GAINS/KPOS1,KPOS2,KVEL1,KVEL2,KFF1,KFF2
      COMMON/CLOCK/IRATE,ITICKS
      COMMON/PSTN/KP1,KP2
      COMMON/FLAG/KFLAG

      DATA IRATE,ITICKS,KP1,KP2,KFLAG/4,20,3*0/
      DATA KPOS1,KPOS2,KVEL1,KVEL2,KFF1,KFF2/2*0,2*200,2*0/

3     FORMAT(/)
4     FORMAT(///)
6     FORMAT(////////)


      WRITE(5,6)
      WRITE(5,*)'-------------------------------------------------------------'
      WRITE(5,*)'This program will ask you to provide values for '
      WRITE(5,*)'variables used to run a position, velocity, and'
      WRITE(5,*)'force feedback control algorithm.  There are safe'
      WRITE(5,*)'default values if you choose not to answer the '
      WRITE(5,*)'question.  If you make a mistake, there will be a'
      WRITE(5,*)'question that allows you to start over without'
      WRITE(5,*)'running the algorithm.  Answer the literal questions'
      WRITE(5,*)'with "yes" or "no".  If you do start over, values'
      WRITE(5,*)'from your last time through have been saved and need'
      WRITE(5,*)'not be re-entered.  Do not panic, there are many'
      WRITE(5,*)'safety features within the algorithm to prevent the'
      WRITE(5,*)'hardware from damage.  Just keep out of its way.'
      WRITE(5,*)'-------------------------------------------------------------'
      WRITE(5,3)
      WRITE(5,*)'Do you wish to continue -- yes or no?'
      READ(5,200)QUEST
200   FORMAT(A4)

      IF(QUEST .EQ. 'YES')GOTO 5
              GOTO 80

5     WRITE(5,4)
      WRITE(5,*)'Do you want to pick a position -- yes or no?'
      WRITE(5,*)'Default position is:'
      WRITE(5,3)
      WRITE(5,*)'                Backboard of apparratus'
      WRITE(5,*)'-------------------------------------------------------------'
      WRITE(5,*)'                          motors'
      WRITE(5,*)'                            *'
      WRITE(5,*)'                            *'
      WRITE(5,*)'                            *'
      WRITE(5,*)'                  Link #1   *'
      WRITE(5,*)'                            *'
      WRITE(5,*)'                            *'
      WRITE(5,*)'                            *****************'
      WRITE(5,*)'                                 link #2'
      WRITE(5,3)
      WRITE(5,*)'                        (top view)'
      WRITE(5,*)'You MUST pick this position if you are running the'
      WRITE(5,*)'controller for the FIRST time!  Default is bypassed'
      WRITE(5,*)'if a nondefault option has been chosen previously.'
```

```
        READ(5,120)QUEST
120     FORMAT(A4)

        IF(QUEST .EQ. 'YES')GOTO 130
                GOTO 190


130     WRITE(5,*)'Pick a position by typing in the number in the '
        WRITE(5,*)'upper right hand corner.  Default position can be'
        WRITE(5,*)'chosen by entering a number greater than 4'
        WRITE(5,*)'----------------------          ----------------------'
        WRITE(5,*)'         *             #1                *               #2'
        WRITE(5,*)'       *                                 *'
        WRITE(5,*)'      *                                  *'
        WRITE(5,*)' L1 *                                 L1   *'
        WRITE(5,*)'    *                                      *'
        WRITE(5,*)'   ************                             ************'
        WRITE(5,*)'         L2                                        L2'
        WRITE(5,*)'----------------------          ----------------------'
        WRITE(5,*)'              *        *3                *               #4'
        WRITE(5,*)'            *                           *              *'
        WRITE(5,*)'           *                             *     L2   *'
        WRITE(5,*)'      L1 *                            L1 *       *'
        WRITE(5,*)'         *                                  *   *'
        WRITE(5,*)'       *                                      *'
        WRITE(5,*)'         *'
        WRITE(5,*)'      L2 *'
        WRITE(5,*)'           *'
        WRITE(5,*)'            *'
        READ(5,*)NUM

        IF(NUM .EQ. 1)GOTO 140
                IF(NUM .EQ. 2)GOTO 150
                        IF(NUM .EQ. 3)GOTO 160
                                IF(NUM .EQ. 4)GOTO 170
                                        GOTO 180

140     KP1 = 1593
        KP2 = -1593
        GOTO   190

150     KP1 = -1593
        KP2 =  1593
        GOTO   190

160     KP1 = 910
        KP2 = 1820
        GOTO   190

170     KP1 = -910
        KP2 = -1820
        GOTO  190

180     KP1 =  0
        KP2 =  0
```

```
190     WRITE(5,4)
        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'Do you want to set the feedback gains -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)'The actual gain used by the control algorithm is :'
        WRITE(5,3)
        WRITE(5,*)'  GAIN = (GAIN YOU TYPE IN)/100'
        WRITE(5,3)
        WRITE(5,*)'This permits fractional gains.'
        WRITE(5,*)'The default gives zero stiffness with a moderate'
        WRITE(5,*)'amount of damping and allows actual link inertias.'
        WRITE(5,*)'Default is bypassed if a nondefault option has been'
        WRITE(5,*)'chosen previously.'
        READ(5,7)QUEST
7       FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 8
                GOTO 45

8       WRITE(5,3)
        WRITE(7,10)
10      FORMAT(' Type in link #1 position gain.')
        READ(5,*)KPOS1

        WRITE(5,3)
        WRITE(7,20)
20      FORMAT(' Type in link #2 position gain.')
        READ(5,*)KPOS2

        WRITE(5,3)
        WRITE(7,30)
30      FORMAT(' Type in link #1 velocity gain.')
        READ(5,*)KVEL1

        WRITE(5,3)
        WRITE(7,40)
40      FORMAT(' Type in link #2 velocity gain.')
        READ(5,*)KVEL2

        WRITE(5,3)
        WRITE(7,42)
42      FORMAT(' Type in link #1 force-feedback gain.')
        READ(5,*)KFF1

        WRITE(5,3)
        WRITE(7,44)
44      FORMAT(' Type in link #2 force-feedback gain.')
        READ(5,*)KFF2
```

```
45      WRITE(5,3)
        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'Do you want to set the sampling rate -- yes or no?'
        WRITE(5,*)'The default gives a 20 millisecond period.'
        WRITE(5,*)'Default is bypassed if a nondefault option has been'
        WRITE(5,*)'chosen previously.'
        READ(5,47)QUEST
47      FORMAT(A4)

        IF (QUEST .EQ. 'YES') GOTO 48
              GOTO 50

48      WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'Choose a combination of tick rate (from the table'
        WRITE(5,*)'below) and number of ticks per sampling period'
        WRITE(5,*)'(your own choice, 1-1000).  The sampling rate is'
        WRITE(5,*)'computed as follows:'

        WRITE(5,3)
        WRITE(5,*)'SAMPLING RATE = (# OF TICKS) * [1/(TICK RATE)]'

        WRITE(5,3)
        WRITE(5,*)'       CHOICE                RATE'
        WRITE(5,3)
        WRITE(5,*)'         1                  1   MHZ'
        WRITE(5,*)'         2                  100 KHZ'
        WRITE(5,*)'         3                  10  KHZ'
        WRITE(5,*)'         4                  1   KHZ'
        WRITE(5,*)'         5                  100 HZ'

        WRITE(5,3)
        WRITE(5,*)'Type in your choices for clock rate (use the'
        WRITE(5,*)'numbers 1-5), and number of ticks per period.'
        WRITE(5,*)'Seperate your input with a comma.'

        READ(5,*)IRATE,ITICKS


50      WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'*************************************************'
        WRITE(5,3)
        WRITE(5,*)'Do you want to start the controller -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)'*************************************************'
        WRITE(5,3)
        READ(5,60)QUEST
60      FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 70
              GOTO 80

70      WRITE(5,6)
        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'****** Press any key to stop the controller ******'
```

```
        CALL PVF


        KFLAG = 1


80      WRITE(5,6)
        WRITE(5,6)
        WRITE(7,90)
90      FORMAT(' Do you want to start over -- yes or no?')
        READ(5,100)QUEST
100     FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 5
                WRITE(5,4)
                WRITE(5,*)' BYE'


        CALL EXIT
        END
```

```
;-----------------------------------------------------------------------
;
;                              PVFMAC.MAC
;                            JOHN WLASSICH
;                               3/7/85
;
;
;       MACRO-11 program to implement a position, velocity, and force feedback
; linear control algortim in real time.  Sensor signals are digitized via A/D
; converters.  Commands are analog signals via D/A converters. Hardware to
; test this algorithm consists of a serial link two degree of freedom
; mechanism actuated by two torque motors and two servo amplifiers. This
; program is called by PVFMNP, a user-friendly FORTRAN main program.  This
; program calls PVFSUB, a FORTRAN subroutine that calculates sine and cosine.
;
;
;-----------------------------------------------------------------------
```

```
.TITLE  PVEMAC          ;Object module name

.GLOBL  PVTSUB          ;Name of FORTRAN subroutine called by
                        ; this MACRO program.

.MCALL  .TTINR          ;Character transfer programmed request

.MCALL  .PRINT          ;Programmed request to allow messages
                        ; on CRT from MACRO program

.ENABLE LC              ;Lower case enable


ADCSR   = 170400        ;A/D control status register address
ADBUF   = 170402        ;A/D buffer address
DACA    = 170440        ;Too noisy (approx. 3* noise DACB)
DACB    = 170442        ;Link #1 digital to analog converter
DACC    = 170444        ;In reserve for possible later use
DACD    = 170446        ;Link #2 digital to analog converter
DIGOUT  = 167772        ;TTL output for motor enable/disable
RTCCSR  = 170420        ;Real-time clock CSR
RTCBUF  = 170422        ;Real-time clock buffer
SCALE   = 100.          ;Whenever a multiply by a gain, divide
                        ; by SCALE so fractional gains possible
LK1SCL  = 44.           ;Link #1 A/D units to degrees scaling
                        ; calculated by: 1000* (degrees of
                        ; movement)/ (4095., A/D units for +,-
                        ; 10V)
LK2SCL  = 44.           ;Same as LK1SCL except for link #2
DACSCL  = 2048.         ;DAC offset from A/D, 4000 octal
CH0     = 1             ;A/D channel #0, in octal
CH3     = 1401          ;A/D channel #3, in octal
CH4     = 2001          ;A/D channel #4, in octal
CH6     = 3001          ;A/D channel #6, in octal
P1MAX   = 3600          ;Octal, allows about +-42 deg. rotation
P1MIN   = -3600         ; of link #1 before a software stop.
P2MAX   = 6000          ;Octal, allows about +-42 deg. rotation
P2MIN   = -6000         ; of link #2 before a software stop.
CDMAX1  = 6000          ;Octal, allows +-5.0 volt output to
CDMIN1  = 2000          ; servo amp #1 before a software stop
CDMAX2  = 7777          ;Cctal, allows +-5.0 volt output to
CDMIN2  = 0             ; servo amp #2 before a software stop
                        ;Table showing A/D outputs:
                        ; Volts        Octal #
                        ;  -10          7777
                        ; CDMAX         see assignment
                        ;    0          4000
                        ; CDMIN         see assignment
                        ;  +10             0
```

```
;Define common blocks:
; GAINS - used in FORTRAN main program (PVFMNP.FOR).
;".BLKW 1" used instead of ".WORD 0" so that a memory space is
; reserved but not assigned to 0 when the program is "RUN".
; This allows a DATA statement in FORTRAN to initialize the variables
; to some non zero value.

                .PSECT  GAINS,RW,D,GBL,REL,OVR
        KPOS1:  .BLKW   1       ;Servo amp #1 position feedback gain.
        KPOS2:  .BLKW   1       ;Servo amp #2 position feedback gain.
        KVEL1:  .BLKW   1       ;Servo amp #1 velocity feedback gain.
        KVEL2:  .BLKW   1       ;Servo amp #2 velocity feedback gain.
        KFF1:   .BLKW   1       ;Servo amp #1 force feedback gain
        KFF2:   .BLKW   1       ;Servo amp #2 force feedback gain

; CLOCK - used in FORTRAN main program (PVFMNP.FOR).

                .PSECT  CLOCK,RW,D,GBL,REL,OVR
        RATE:   .BLKW   1       ;RTC frequency
        TICKS:  .BLKW   1       ;# of ticks per period of RTC

; PSTN - used in FORTRAN main program (PVFMNP.FOR).

                .PSECT  PSTN,RW,D,GBL,REL,OVR
        REFP1:  .BLKW   1       ;Link #1 reference position
        REFP2:  .BLKW   1       ;Link #2 reference position

; FLAG -  used in FORTRAN main program (PVFMNP.FOR).

                .PSECT  FLAG,RW,D,GBL,REL,OVR
        FLG:    .BLKW   1       ;Flag used to govern position offsets

; TRIG - used in FORTRAN subroutine (PVFSUB.FOR).

                .PSECT  TRIG,RW,D,GBL,REL,OVR
        THETA1: .WORD   0       ;Link #1 angular position in degrees
        THETA2: .WORD   0       ;Link #2 angular position in degrees
        ST1:    .WORD   0       ;Sine of THETA1
        CT1:    .WORD   0       ;Cosine of THETA1
        ST2:    .WORD   0       ;Sine of THETA2
        CT2:    .WORD   0       ;Cosine of THETA2


;This next directive tells the linker the MACRO program starts
; here. This is to prevent the COMMON blocks from overlapping
; the MACRO code.

        .PSECT  $CODE


;"PVF" marks the point from which this program begins execution when
; it is called by the fortran main program, PVFMNP.FOR.  PVF is a
; global label.
;Take negative of gains so force feedbacks are negative.
;Position and velocity feedbacks already negative feedback in hardware.

PVF::   NEG     KFF1            ;Link #1 force feedback gain
        NEG     KFF2            ;Link #2 force feedback gain
```

```
;This next section of "CLR" instructions zeros memory locations used
; to get positions of each link that have occurred 4 sampling periods
; in the past.
;The section that uses these variables is located in the
; velocity calculation portion of the servo amp command for each link.

        CLR     P12             ;Link #1 position from 1 sample past
        CLR     P13             ;Link #1 position from 2 samples past
        CLR     P14             ;Link #1 position from 3 samples past
        CLR     P15             ;Link #1 position from 4 samples past
        CLR     P22             ;Link #2 position from 1 sample past
        CLR     P23             ;Link #2 position from 2 samples past
        CLR     P24             ;Link #2 position from 3 samples past
        CLR     P25             ;Link #2 position from 4 samples past


;These next 4 "CLR" instructions zero the position offsets
; and force offsets from the last run. Saved values are not lost.
;These position values are kept when the program calls its stopping
; routine, "STOP" (located at the bottom fourth of this program).

        CLR     POFF1           ;Link #1 position
        CLR     POFF2           ;Link #2 position
        CLR     FXOFF           ;Force in X direction
        CLR     FYOFF           ;Force in Y direction


        CLR     @#RTCCSR        ;Reset real time clock


        CLRB    @#ADCSR         ;Clear A/D CSR:  1) lower byte
        CLRB    @#ADCSR+1       ;                2) upper byte
        TST     @#ADBUF         ;                3) A/D done flag


        BIS     #10000,@#44     ;JSW special mode terminal bit, monitor
                                ; will not echo character typed


;Set unused DAC's to zero volts to minimize any possible EMI.

        MOV     #DACSCL,@#DACA  ;2048 (decimal) = 0 volts
        MOV     #DACSCL,@#DACC  ;2048 (decimal) = 0 volts


;Set sampling rate.
; sampling period = number of ticks per period * (1/ RTC tick rate)
;Start up real time clock.

        NEG     TICKS           ;Clock counts up to zero
        MOV     TICKS,@#RTCBUF  ;Load RTC buffer w/ # of ticks
        MOV     RATE,R0         ;ASH requires a register
        ASH     #3,R0           ;Set clock rate (bits 3-5)
        BIS     R0,@#RTCCSR     ;Load RTC control status register
        BIS     #2,@#RTCCSR     ;Set clock in mode 1 (continuous mode)
        INCB    @#RTCCSR        ;Start clock
```

```
;Sample is the routine which gets A/D values, use it now
; to get initial sensor readings, used as offsets.

        CALL    SAMPLE          ;Located near bottom 2/3 of program


;Keep force transducer offsets generated each run.
        MOV     FX,FXOFF        ;Force in X direction
        MOV     FY,FYOFF        ;Force in Y direction


;The following section, up to the label "SKIP", uses a flag from the
; FORTRAN main program, PVFMNP, to make a decision.
;If FLG not equal to zero (set to 1 in PVFMNP):
; the algorithm has already been run at the "start" position and
; has saved potentiometer (position) offsets.

        CMP     #0,FLG          ;See if flag ("FLG") = 0
        BEQ     KEEP            ;If yes, continue at label "KEEP"
        MOV     SAVPO1,POFF1    ;Get saved link #1 position offset
        MOV     SAVPO2,POFF2    ;Get saved link #2 position offset
        BR      SKIP            ;Continue at label "SKIP"

;If FLG = 0 (set to 0 in PVFMNP):
; the algorithm is being run for the first time and needs position
; sensor offsets (potentiometer offsets) from zero.

KEEP:   MOV     POS1,POFF1      ;Get link #1 position offsets
        MOV     POS2,POFF2      ;Get link #2 position offsets


SKIP:   MOV     #0,@#DIGOUT     ;Enable motors
```

```
;This next statement starts the control algorithm by checking the
; sampling rate.  REPEAT is the upper bound of the control loop.

REPEAT: TSTB    @#RTCCSR        ;Check overflow bit:
        BPL     GOHEAD          ;If not set, algorithm w/in bounds of
                                ; specified sampling rate
        JMP     SLODWN          ;If set, sampling too fast

GOHEAD: CALL    SAMPLE          ;Get A/D values.

        CMP     POS1,#P1MAX     ;Compare link #1 position to ccw max
        BLT     1$              ; allowed, if over, jump to "EXROT1"
        JMP     EXROT1          ;Must use JMP since branch to "EXROT1"
                                ; further than 127. words away
1$:     CMP     POS1,#P1MIN     ;Compare link #1 position to cw max
        BGT     2$              ; allowed, if over, jump to "EXROT1"
        JMP     EXROT1          ;Print out "Link #2 out of bounds"

;From now on whenever JMP used instead of a branch,
; due to word branch limits.

2$:     CMP     POS2,#P2MAX     ;Compare link #2 position to ccw max
        BLT     3$              ; allowed, if over, jump to "EXROT2"
        JMP     EXROT2          ;Print out "Link #2 out of bounds"
3$:     CMP     POS2,#P2MIN     ;Compare link #2 position to cw max
        BGT     4$              ; allowed, if over, jump to "EXROT2"
        JMP     EXROT2          ;Print out "Link #2 out of bounds"
```

```
;Control algorithm built as follows:
;
; Command(servo amp #1) =
;  KPOS1*(REFP1 - POS1) + KVEL1*(POS1-P1LAST) +
;   KFF1*[(S21)*FX + (C21)*FY]
;
; Command(servo amp #2) =  KPOS2*(REFP2 - POS2) + KVEL2*(POS2-P2LAST) +
;  KFF2*FY
;
;Note :
;
;  1) The link lenghts, since they are equal, have been factored
;     out and are absorbed into the gains.
;
;  2) There is no need to divide by the sampling rate to calculate
;     the angular velocity since it can be factored out and absorbed
;     by the velocity gain.

;Servo amp #1 position feedback

4$:     MOV     POS1,R0         ;Get position of link #1 (a negative)
        ADD     REFP1,R0        ;Add in link #1 reference position
        BVC     44$             ;Checks for an arithmatic overflow
        JMP     OVRFLO          ; by seeing if "V" bit of the PSW is set
44$:    MUL     KPOS1,R0        ;Multiply by position gain #2
        DIV     #SCALE,R0       ;Cancel out gain scale
        BVC     5$              ;Checks for an arithmatic overflow
        JMP     OVRFLO          ; by seeing if "V" bit of the PSW is set
5$:     MOV     R0,R2           ;Build servo amp #1 command in R2


;Servo amp #2 position feedback

        MOV     POS2,R0         ;Get position of link #2 (a negative)
        ADD     REFP2,R0        ;Add in link #2 reference position
        BVC     55$             ;Checks for an arithmatic overflow
        JMP     OVRFLO          ; by seeing if "V" bit of the PSW is set
55$:    MUL     KPOS2,R0        ;Multiply by position gain #2
        DIV     #SCALE,R0       ;Cancel out gain scale
        BVC     6$              ;Checks for an arithmatic overflow
        JMP     OVRFLO          ; by seeing if "V" bit of the PSW is set
6$:     MOV     R0,R4           ;Build servo amp #2 command in R4
```

```
;Servo amp #1 velocity feedback
;
;Use a first order differentiator with smoothing:
; [(current link #1 position) - (link #1 position of five time steps ago)]*
; servo amplifier #1 velocity feedback gain

        MOV     POS1,R0         ;Get link #1 position
        SUB     P15,R0          ;Subtract position from 4 samples past
        MUL     KVEL1,R0        ;Multiply result by vel feedback gain
        DIV     #SCALE,R0       ;Divide out gain scale
        BVC     7$              ;Print out "arithmatic overflow"
        JMP     OVRFLO          ; message if "V" bit set of PSW
7$:     ADD     R0,R2           ;Continue building servo #1 command in R2
        BVC     8$              ;Print out "arithmatic overflow"
        JMP     OVRFLO          ; message if "V" bit set of PSW

;These next 4 instructions take the current position and saves it as a
; position of 1 sampling period past.  Each subsequent position then
; pushes the position before it into a location denoting its older status.
;When the control loop repeats, P15 will be 5 positions older than the
; most current position. Hence only 4 "MOV" instructions are needed.

8$:     MOV     P14,P15         ;Position (-4) becomes position (-5)
        MOV     P13,P14         ;Position (-3) becomes position (-4)
        MOV     P12,P13         ;Position (-2) becomes position (-3)
        MOV     POS1,P12        ;Present position becomes position (-2)


;Servo amp #2 velocity feedback
;
;Use a first order differentiator with smoothing:
; [(current link #2 position) - (link #2 position of five time steps ago)]*
; servo amplifier #2 velocity feedback gain

        MOV     POS2,R0         ;Get link #2 position
        SUB     P25,R0          ;Subtract position from 4 samples past
        MUL     KVEL2,R0        ;Multiply result by vel feedback gain
        DIV     #SCALE,R0       ;Divide out gain scale
        BVC     9$              ;Print out "arithmatic overflow"
        JMP     OVRFLO          ; message if "V" bit set of PSW
9$:     ADD     R0,R4           ;Continue building servo #2 command in R4
        BVC     10$             ;Print out "arithmatic overflow"
        JMP     OVRFLO          ; message if "V" bit set of PSW

;These next 4 instructions take the current position and saves it as a
; position of 1 sampling period past.  Each subsequent position then
; pushes the position before it into a location denoting its older status.
;When the control loop repeats, P15 will be 5 positions older than the
; most current position. Hence only 4 "MOV" instructions are needed.

10$:    MOV     P24,P25         ;Position (-4) becomes position (-5)
        MOV     P23,P24         ;Position (-3) becomes position (-4)
        MOV     P22,P23         ;Position (-2) becomes position (-3)
        MOV     POS2,P22        ;Present position becomes position (-2)
```

```
;Force feedback for link #1 and link #2
;
;Transducer outputs:
;
; FX = parallel to (positive outward) link #2
; FY = perpendicular to (positive outward) link #2
;
;Rotation of force components into absolute coordinates as follows:
;
; Torque command for link #1 =
;  {[(sin(theta 2 - theta 1))*FX]/1000. +
;    [(cos(theta 2 - theta 1))*FY]/1000.}*(force-feedback gain 1)
;
; Torque command for link #2 = (force-feedback gain 2)*FY

        CALL    TRIG            ;Get sine and cosine in abs. coords.

        MOV     SAVC1,R2        ;Get saved servo amp command #1
        MOV     SAVC2,R4        ;Get saved servo amp command #2

        MOV     ST2,R0          ;Start forming ST21
        MUL     CT1,R0          ;ST2 * CT1
        DIV     #1000.,R0       ;/ result by 1000
        MOV     R0,TEMP2        ;Save this result
        MOV     CT2,R0          ;Get CT2
        MUL     ST1,R0          ;CT2 * ST1
        DIV     #1000.,R0       ;Get result in 1000's
        NEG     R0              ;Make CT2 * ST1 negative
        ADD     TEMP2,R0        ;Sin(theta2-theta1)=ST2*CT1 - CT2*ST1
        BVC     16$             ;Check for an arithmatic overflow
        JMP     OVRFLO          ;Print out "arithmatic overflow"
16$:    MUL     FX,R0           ;Multiply above by force in X direction
        DIV     #1000.,R0       ;/1000. to cancel out 1000. in sin
        BVC     11$             ;Check for an arithmatic overflow
        JMP     OVRFLO          ;Print out "arithmatic overflow"
11$:    MOV     R0,TEMP1        ;Save this first part
```

```
        MOV     CT1,R0          ;Start forming CT21
        MUL     CT2,R0          ;CT2 * CT1
        DIV     #1000.,R0       ;/ result by 1000
        MOV     R0,TEMP2        ;Save this result temporarily
        MOV     ST1,R0          ;Get ST2
        MUL     ST2,R0          ;ST2 * ST1
        DIV     #1000.,R0       ;Get result in 1000's
        ADD     TEMP2,R0        ;Cos(theta2-theta1)= CT2*CT1 + ST2*ST1
        BVC     17$             ;Check for an arithmatic overflow
        JMP     OVRFLO          ;Print out "arithmatic overflow"
17$:    MUL     FY,R0           ;Multiply above by force in Y direction
        DIV     #1000.,R0       ;cos is scaled by 1000. in PVFSUB
        BVC     12$             ;Check for an arithmatic overflow
        JMP     OVRFLO          ;Print out "arithmatic overflow"
12$:    ADD     TEMP1,R0        ;Get saved part, add two sections above
        MUL     KFF1,R0         ;Multiply by force-feedback gain 1
        DIV     #SCALE,R0       ;Scale back KFF1
        ADD     R0,R2           ;Add in torque 1 command to P1 & V1
        BVC     13$             ;Check for an arithmatic overflow
        JMP     OVRFLO          ;Print out "arithmatic overflow"

13$:    MOV     FY,R0           ;Build torque 2 command
        MUL     KFF2,R0         ;Multiply by force-feedback gain 2
        DIV     #SCALE,R0       ;Scale back, like all gains
        ADD     R0,R4           ;Add in torque 2 command to P2 & V2
        BVC     14$             ;Check for an arithmatic overflow
        JMP     OVRFLO          ;Print out "arithmatic overflow"
```

```
14$:    ADD     #DACSCL,R2      ;Add in offset between DAC and A/D
        ADD     #DACSCL,R4      ; DAC = 4000 - A/D

        CMP     R2,#CDMAX1      ;Check if output command to servo #1 is
        BLT     OK1A            ; less than positive bound, if yes,
                                ; go to label OKIA
        CALL    STOPCL          ;Stop clock so that message can be
                                ; displayed without tripping
                                ; "sampling too fast" routine
        .PRINT  #MSG5           ;Message "servo #1 command excessive"
        CALL    STRTCL          ;Restart clock
        MOV     #CDMAX1,R2      ;Command must be > positive bound, put
                                ; positive bound into command

OK1A:   CMP     R2,#CDMIN1      ;Check if output command to servo #1 is
        BGT     OK1B            ; greater than negative bound, if yes,
                                ; go to label OKIB
        CALL    STOPCL          ;Stop clock so that message can be
                                ; displayed without tripping
                                ; "sampling too fast" routine
        .PRINT  #MSG5           ;Message "servo #1 command excessive"
        CALL    STRTCL          ;Restart clock
        MOV     #CDMIN1,R2      ;Command must be < negative bound, put
                                ; positive bound into command

OK1B:   CMP     R4,#CDMAX2      ;Check if output command to servo #2 is
        BLT     OK2A            ; less than positive bound, if yes,
                                ; go to label OK2A
        CALL    STOPCL          ;Stop clock so that message can be
                                ; displayed without tripping
                                ; "sampling too fast" routine
        .PRINT  #MSG6           ;Message "servo #2 command excessive"
        CALL    STRTCL          ;Restart clock
        MOV     #CDMAX2,R4      ;Command must be < negative bound, put
                                ; positive bound into command

OK2A:   CMP     R4,#CDMIN2      ;Check if output command to servo #2 is
        BCT     OK2B            ; less than positive bound, if yes,
                                ; go to label OK2B
        CALL    STOPCL          ;Stop clock so that message can be
                                ; displayed without tripping
                                ; "sampling too fast" routine
        .PRINT  #MSG6           ;Message "servo #21 command excessive"
        CALL    STRTCL          ;Restart clock
        MOV     #CDMIN2,R4      ;Command must be < negative bound, put
                                ; positive bound into command
```

```
;Output commands to DACs

OK2B:   MOV     R2,@#DACB       ;Servo amp #1 (link #1)
        MOV     R4,@#DACC       ;Servo amp #2 (link #2)


        .TTINR                  ;Check for key strike, no character
                                ; transferred if carry set
        BCC     STRUCK          ;If no key strike, continue with the
        JMP     REPEAT          ; control algorithm loop
STRUCK: JMP     STOP            ;If a key struck, stop program and
                                ; hardware


;Sampling routine for link #1 and link#2 positions,
; and force transducer outputs in X and Y
;No "CALLS" used here since speed is critical

SAMPLE: TSTB    @#RTCCSR        ;Check overflow bit
        BPL     SAMPLE          ; if not set,wait
        BIC     #200,@#RTCCSR   ;Clear overflow bit

;Channel #0 for link #1 position
        MOV     #CH0,@#ADCSR    ;Convert analog voltage on channel #0
WAIT1:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT1           ;If clear, wait for conversion
        MOV     @#ADBUF,POS1    ;Get digital value from A/D buffer
        SUB     POFF1,POS1      ;Subtract off initial position offset

;Channel #3 for link #3 position
        MOV     #CH3,@#ADCSR    ;Convert analog voltage on channel #3
WAIT2:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT2           ;If clear, wait for conversion
        MOV     @#ADBUF,POS2    ;Get digital value from A/D buffer
        SUB     POFF2,POS2      ;Subtract off initial position offset

;Channel #4 for force in the X direction
        MOV     #CH4,@#ADCSR    ;Convert analog voltage on channel #4
WAIT3:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT3           ;If clear, wait for conversion
        MOV     @#ADBUF,FX      ;Get digital value from A/D buffer
        SUB     FXOFF,FX        ;Subtract off initial force offset

;Channel #6 for force in the Y direction
        MOV     #CH6,@#ADCSR    ;Convert analog voltage on channel #6
WAIT4:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT4           ;If clear, wait for conversion
        MOV     @#ADBUF,FY      ;Get digital value from A/D buffer
        SUB     FYOFF,FY        ;Subtract off initial force offset
        RETURN                  ;Go back to statement after last CALL
```

```
;This next routine scales the A/D position inputs to angles
; in degrees in absolute (backboard of apparratus) coordinates
;Computation is then sent to a fortran subroutine (PVFSUB)
; where the sine and cosine of each angle is found
;The last step returns computation to the statement just after
; the last CALL TRIG

TRIG:   MOV     POS1,R0         ;Get link #1 A/D position value
        MUL     #LK1SCL,R0      ;Scale A/D units to 1000's of degrees
        DIV     #1000.,R0       ;Cancel out 1000 in LK1SCL
        MOV     R0,THETA1       ;Save angle 1

        MOV     POS2,R0         ;Get link #2 A/D position value
        MUL     #LK2SCL,R0      ;Scale A/D units to 1000's of degrees
        DIV     #1000.,R0       ;Left with 1 degree of accuracy
        MOV     THETA1,R1       ;To get angle #2 in absolute coords:
        ADD     R1,R0           ; add in angle #1
        ADD     #90.,R0         ; and add in 90 degrees
        MOV     R0,THETA2       ;Save angle 2

        MOV     R2,SAVC1        ;Save servo amp #1 command
        MOV     R4,SAVC2        ;Save servo amp #2 command

        JSR     PC,PVFSUB       ;Go compute sine and cosine
        RETURN                  ;Go to 1 line after last CALL


;"STOPCL" and "STRTCL" used to allow time for the "servo amp command
; excessive" warnings to be printed without tripping the "sampling too
; fast routine"

STOPCL: CLR     @#RTCCSR        ;Reset RTC control status register
        RETURN                  ;Go back to point just after last CALL

STRTCL: MOV     TICKS,@#RTCBUF  ;Load RTC buffer w/ # of ticks
        MOV     RATE,R0         ;ASH requires a register
        ASH     #3,R0           ;Set clock rate (bits 3-5)
        BIS     R0,@#RTCCSR     ;Load RTC control status register
        BIS     #2,@#RTCCSR     ;Set clock in mode 1 (continuous mode)
        INCB    @#RTCCSR        ;Start clock
        RETURN                  ;Go back to point just after last CALL
```

```
;This next section shuts down all hardware and returns control
; to the FORTRAN main program, PVFMNP.FOR.

STOP:   MOV     #1,@#DIGOUT     ;Disable motors
        MOV     #DACSCL,@#DACA  ;Clear DAC A output, 2048 = 0 volts
        MOV     #DACSCL,@#DACB  ;Clear DAC B output, 2048 = 0 volts
        MOV     #DACSCL,@#DACC  ;Clear DAC C output, 2048 = 0 volts
        MOV     #DACSCL,@#DACD  ;Clear DAC D output, 2048 = 0 volts
        BIC     #10000,@#44     ;Reset JSW
        CLR     @#RTCCSR        ;Reset RTC control status register
        CLRB    @#ADCSR         ;Clear A/D CSR:  1) lower byte
        CLRB    @#ADCSR+1       ;                2) upper byte
        TST     @#ADBUF         ;                3) A/D done flag
        MOV     POFF1,SAVPO1    ;Save link #1 position offset
        MOV     POFF2,SAVPO2    ;Save link #2 position offset
        NEG     TICKS           ;Reset # of ticks per sampling period
        NEG     KFF1            ;Reset link #1 force feedback gain
        NEG     KFF2            ;Reset link #2 force feedback gain
        RETURN                  ;Go back to main program


;This next section has the routines that are called when a error or
; fault has been encountered in the program.

EXROT1: CALL    STOP            ;This is called when a rotation out of
        .PRINT  #MSG2           ; bounds occurs for link#1
        RETURN                  ;Control is returned to the main program

EXROT2: CALL    STOP            ;Same as comment above but for link#2
        .PRINT  #MSG3
        RETURN

OVRFLO: CALL    STOP            ;This is called when an arithmatic
        .PRINT  #MSG4           ; overflow is detected
        RETURN                  ;Control is returned to the main program

SLODWN: CALL    STOP            ;This is called when the sampling rate
        .PRINT  #MSG7           ; exceeds the time the algorithm needs
        RETURN                  ; to be computed
                                ;Control is returned to the main program
```

```
;The following section defines the error messages.
;
;Called by "EXROT1"

MSG2:   .ASCIZ  /LINK #1 OUT OF BOUNDS./
        .EVEN

;Called by "EXROT2"

MSG3:   .ASCIZ  /LINK #2 OUT OF BOUNDS./
        .EVEN

;Called by "OVRFLO"

MSG4:   .ASCIZ  /ARITHMATIC OVERFLOW - BARF! BARF! BARF!/
        .EVEN                                                    !

;These next two messages are printed by the program. Control
; stays in the MACRO program.

MSG5:   .ASCIZ  /LINK #1 SERVO AMP COMMAND EXCESSIVE./
        .EVEN

MSG6:   .ASCIZ  /LINK #2 SERVO AMP COMMAND EXCESSIVE./
        .EVEN

;Called by "SLODWN"

MSG7:   .ASCIZ  /SAMPLING TOO FAST./
        .EVEN


;Variables not already defined :

        FX:     .WORD   0       ;Force in X direction
        FY:     .WORD   0       ;Force in Y direction
        FXOFF:  .WORD   0       ;Initial force transducer X offset
        FYOFF:  .WORD   0       ;Initial force transducer Y offset
        P12:    .WORD   0       ;Link #1 position from 1 sample past
        P13:    .WORD   0       ;Link #1 position from 2 samples past
        P14:    .WORD   0       ;Link #1 position from 3 samples past
        P15:    .WORD   0       ;Link #1 position from 4 samples past
        P22:    .WORD   0       ;Link #2 position from 1 sample past
        P23:    .WORD   0       ;Link #2 position from 2 samples past
        P24:    .WORD   0       ;Link #2 position from 3 samples past
        P25:    .WORD   0       ;Link #2 position from 4 samples past
        POFF1:  .WORD   0       ;Initial link #1 zero position offset
        POFF2:  .WORD   0       ;Initial link #2 zero position offset
        POS1:   .WORD   0       ;Link #1 current position
        POS2:   .WORD   0       ;Link #2 current position
        SAVC1:  .WORD   0       ;Temporary location, servo #1 command
        SAVC2:  .WORD   0       ;Temporary location, servo #2 command
        SAVPO1: .WORD   0       ;Saved link #1 zero position offset
        SAVPO2: .WORD   0       ;Saved link #2 zero position offset
        TEMP1:  .WORD   0       ;Temporary memory location #1
        TEMP2:  .WORD   0       ;Temporary memory location #2

        .END
```

```
C-----------------------------------------------------------------------
C
C                              PVFTRG.FOR
C                            JOHN WLASSICH
C                               2/27/85
C
C
C       A FORTRAN subroutine which calculates sine and cosine.  Called
C   by PVFMAC.
C
C       Variables defined as follows:
C
C               KCT1    = integer cosine of theta 1
C               KCT2    = integer cosine of theta 2
C               KST1    = integer sine of theta 2
C               KST2    = integer sine of theta 2
C               KTHET1  = integer theta 1 in absolute coordinates
C               KTHET2  = integer theta 2 in absolute coordinates.
C               THETA1  = real of KTHET1
C               THETA2  = real of KTHET2
C
C       KTHET1 and KTHET2 are sent over from PVFMAC as integers.
C They represent the link positions, in decimal degrees, multiplyed
C by 1000 to keep 3 significant figures in integer arithmatic.
C KCT1, KCT2, KST1, KST2, are all sent back to PVFMAC as integers,
C and as such, must be multiplyed by 1000 to keep 3 significant
C figures accuracy.
C
C-----------------------------------------------------------------------
```

```
      SUBROUTINE      PVFTRG

      COMMON/TRIG/KTHET1,KTHET2,KST1,KCT1,KST2,KCT2

C     KTHET1 and KTHET2 are calculated in PVFMAC.MAC to whole
C  number degrees.  The next line converts degrees to radians
C  to greater than the 3 significant figure accuracy of PVFMAC.MAC.
C
C           PI/180 = .017543
C

      THETA1  = FLOAT(KTHET1) * .017453
      THETA2  = FLOAT(KTHET2) * .017453


C     The next four lines compute sine and cosine.  The results are
C  each multiplied by 1000 to prepare them for integer arithmatic in
C  PVFMAC.MAC
C

      KST1    = IFIX(SIN(THETA1) * 1000)
      KCT1    = IFIX(COS(THETA1) * 1000)

      KST2    = IFIX(SIN(THETA2) * 1000)
      KCT2    = IFIX(COS(THETA2) * 1000)


      RETURN

      END
```

Appendix IV

```
C-----------------------------------------------------------------------
C
C                        MNPPNT.FOR
C                      JOHN WLASSICH
C                         5/20/85
C
C       A FORTRAN main program which calls MACPNT.MAC, a
C MACRO - 11 subroutine.  MACPNT uses TRGPNT, a FORTRAN subroutine,
C which calculates the sine and cosine of link joint angles.
C
C       This program prompts the user to initialize the controller,
C choose up to 5 cartesian endpoint reference positons, feedback
C gains, apparent endpoint inertia, and prompts to begin or restart the
C MACRO - 11 control program.  The objective of this FORTRAN program
C is be a user friendly interface to the MACRO - 11 control program.
C
C       Variable definitions follow from input queries below.
C
C-----------------------------------------------------------------------
```

```
      INTEGER*4 QUEST,I
      REAL*4 RK1,RK2,RKV1,RKV2

      DIMENSION KX1R(5),KX2R(5)
      DIMENSION DX1R(5),DX2R(5)

      COMMON/GAINS/K1,K2,KV1,KV2
      COMMON/X1CORD/KX1R
      COMMON/X2CORD/KX2R
      COMMON/DESMAS/M11INV,M22INV
      COMMON/TEST/KTEST1,KTEST2,KTEST3

      DATA KX1R,KX2R/10*1000/
      DATA K1,K2,KV1,KV2/2*0,2*5000/
      DATA M11INV,M22INV/2*1000/

      DATA DX1R,DX2R/10*12.0/
      DATA RK1,RK2,RKV1,RKV2/2*0.,2*5./
      DATA DM11,DM22/2*1./

3     FORMAT(/)
4     FORMAT(///)
6     FORMAT(/////////)
```

```
      WRITE(5,4)
      WRITE(5,*)'-------------------------------------------------'
      WRITE(5,*)'  This program will ask you to provide values for '
      WRITE(5,*)' variables used to run a nonlinear force - feedback'
      WRITE(5,*)' impedence control algorithm.  There are safe'
      WRITE(5,*)' default values if you choose not to answer the '
      WRITE(5,*)' question.  If you make a mistake, there will be a'
      WRITE(5,*)' question that allows you to start over without'
      WRITE(5,*)' running the algorithm.  Answer the literal questions'
      WRITE(5,*)' with "yes" or "no".  If you do start over, values'
      WRITE(5,*)' from your last time through have been saved and need'
      WRITE(5,*)' not be re-entered.  To stop the controller once you '
      WRITE(5,*)' have started it, strike any key.  Disable the servo'
      WRITE(5,*)' amps by toggling your belt switch.  This program does'
      WRITE(5,*)' not have the software safety features of PVFMAC.'
      WRITE(5,*)' STAY CLEAR OF THE YELLOW LINE.'
      WRITE(5,*)'-------------------------------------------------'
      WRITE(5,3)
      WRITE(5,*)' Do you wish to continue -- yes or no?'
      READ(5,205)QUEST
205   FORMAT(A4)

      IF(QUEST .EQ. 'YES')GOTO 1
            GOTO 500


1     WRITE(5,4)
      WRITE(5,*)'-------------------------------------------------'
      WRITE(5,*)'  Your transducer signal inputs should be connected '
      WRITE(5,*)' to the I/O board in the following manner:'
      WRITE(5,3)
      WRITE(5,*)'   1) Link #1 potentiometer --- A/D channel #0'
      WRITE(5,*)'   2) Link #2 potentiometer --- A/D channel #2'
      WRITE(5,*)'   3) Force transducer x output --- A/D channel #4'
      WRITE(5,*)'   4) Force transducer y output --- A/D channel #6'
      WRITE(5,3)
      WRITE(5,*)' Your command outputs from the I/O board should be'
      WRITE(5,*)' in the following manner:'
      WRITE(5,3)
      WRITE(5,*)'   1) Servo amp #1 (motor, link #1) --- DAC B'
      WRITE(5,*)'   2) Servo amp #2 (motor, link #2) --- DAC D'
      WRITE(5,*)'-------------------------------------------------'
      WRITE(5,3)
      WRITE(5,*)' Do you wish to continue -- yes or no?'
      READ(5,200)QUEST
200   FORMAT(A4)

      IF(QUEST .EQ. 'YES')GOTO 128
            GOTO 500
```

```
128     WRITE(5,4)
        WRITE(5,*)'  To initialize the controller, put the apparatus in'
        WRITE(5,*)' the "START" position as illustrated below.'
        WRITE(5,3)
        WRITE(5,*)'              Backboard of apparratus'
        WRITE(5,*)'-----------------------------------------------'
        WRITE(5,*)'                        motors'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                Link #1   *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          ******************'
        WRITE(5,*)'                               link #2'
        WRITE(5,3)
        WRITE(5,*)'                      (top view)'
        WRITE(5,*)' Now stay clear of the yellow line and type in "go"'
        READ(5,122)QUEST
122     FORMAT(A4)

        IF(QUEST .EQ. 'GO')GOTO 132
                WRITE(5,6)
                WRITE(5,6)
                WRITE(5,6)
                WRITE(5,*)' ********  ILLEGAL INPUT  ********'
                WRITE(5,3)
                WRITE(5,*)' Do you want to quit -- yes or no?'
                WRITE(5,6)
                READ(5,124)QUEST
124             FORMAT(A4)

                IF(QUEST .EQ. 'YES')GOTO 500
                        GOTO 128


132     CALL FFI1


        WRITE(5,6)
        WRITE(5,*)'************* INITIALIZATION COMPLETED *************'

5       WRITE(5,6)
        WRITE(5,*)'  Do you want to input endpoint positions '
        WRITE(5,*)' -- yes or no?'
        WRITE(5,*)' Default positions are all the START position '
        WRITE(5,*)' (X1 = 12 in., X2 = 12 in.)'
        WRITE(5,*)' A position default is bypassed if a nondefault'
        WRITE(5,*)' position has been chosen previously.'
        READ(5,120)QUEST
120     FORMAT(A4)
```

```
        IF(QUEST .EQ. 'YES')GOTO 130
                GOTO 190

130     I = 1
        WRITE(5,6)
        WRITE(5,*)'  Input a position by typing in the (X1,X2)'
        WRITE(5,*)' coordinates in inches (to .01 in.).  You will be'
        WRITE(5,*)' notified if your choice is out of the mechanisms'
        WRITE(5,*)' bounds.  You will be able to direct the mechanism'
        WRITE(5,*)' to these points in any order'
        WRITE(5,*)'  Suggestion; make your first point the START '
        WRITE(5,*)' position to avoid any unexpected movements.'
        WRITE(5,4)
138     WRITE(5,131)I
131     FORMAT('0','Type in point number',I2,'.')
        READ(5,*)DX1R(I),DX2R(I)

C These next four lines check that the reference position specified by
C the user is within the mechanism's bounds.

        IF(DX1R(I) .GT. 18  .OR.  DX1R(I) .LT. 7.9)GOTO 140
                IF(DX2R(I) .GT. 16  .OR. DX2R(I) .LT. -12.1)GOTO 140

        BOUND = -3. * DX1R(I) + 24
        IF(DX2R(I) .LT. BOUND)GOTO 140

C These next two lines convert the desired endpoint position from
C  inches to (feet *1000).

                KX1R(I) = IFIX((DX1R(I)/12.) * 1000)
                KX2R(I) = IFIX((DX2R(I)/12.) * 1000)
                GOTO 180

140     WRITE(5,6)
        WRITE(5,*)' ILLEGAL POSTION ENTRY ---  TRY AGAIN.'
        GOTO 138

180     IF(I .GE. 5) GOTO 190
        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'Do you wish to input another point -- yes or no?'
        READ(5,134)QUEST
134     FORMAT(A4)

        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,6)
        IF(QUEST .NE. 'YES')GOTO 190
                I = I + 1
                GOTO 138
```

```
190     WRITE(5,6)
        WRITE(5,*)'  Do you want to set the feedback gains -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)' The permissable range is :'
        WRITE(5,3)
        WRITE(5,*)'       -32 <= STIFFNESS (lb/ft) <= 32 '
        WRITE(5,3)
        WRITE(5,*)'       -1.5 <= DAMPING (lb-sec/ft) <= 1.5, '
        WRITE(5,3)
        WRITE(5,*)'       where the smallest fraction allowed is .001 .'
        WRITE(5,3)
        WRITE(5,*)'  The default gives zero stiffness with a moderate'
        WRITE(5,*)' amount of damping.'
        WRITE(5,*)' Default is bypassed if a nondefault option has been'
        WRITE(5,*)' chosen previously.'
        READ(5,7)QUEST
7       FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 8
                GOTO 45

8       WRITE(5,3)
        WRITE(7,10)
10      FORMAT(' Type in X1 direction stiffness gain -- lb./ft.')
        READ(5,*)RK1

C This next line checks for an illegal gain entry.  The line following
C calibrates and prepares K1 for integer math in MACPNT.  The ".5" guards
C against decimal number truncation, it does not round up the input.

        IF(RK1 .LT. -32  .OR.  RK1 .GT. 32) GOTO 35
                K1 = IFIX((RK1 * .662 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,20)
20      FORMAT(' Type in X2 direction stiffness gain -- lb./ft.')
        READ(5,*)RK2

        IF(RK2 .LT. -32  .OR.  RK2 .GT. 32) GOTO 35
                K2 = IFIX((RK2 * .662 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,30)
30      FORMAT(' Type in X1 direction damping gain -- (lb. sec.)/ft.')
        READ(5,*)RKV1

        IF(RKV1 .LT. -1.5  .OR.  RKV1 .GT. 1.5) GOTO 35

C  The following line adjusts the desired damping gain to a value
C that is correct for the code.  This adjustment is necessary to
C account for the velocity being built from every 3 position samples with
C smooothing while the position is derived from each sensed position sample.
C This line is repeated for the next desired damping gain.

        KV1 = IFIX(RKV1 * 21.3 * 1000. + .5)

        WRITE(5,3)
        WRITE(7,40)
40      FORMAT(' Type in X2 direction damping gain -- (lb. sec.)/ft.')
```

```
      READ(5,*)RKV2

      IF(RKV2 .LT. -1.5  .OR.  RKV2 .GT. 1.5) GOTO 35
      KV2 = IFIX(RKV2 * 21.3 * 1000. + .5)
            GOTO 45

35    WRITE(5,4)
      WRITE(5,*)' ILLEGAL GAIN ENTRY --- START OVER.'
      WRITE(5,4)
      GOTO 8


45    WRITE(5,4)
      WRITE(5,6)
      WRITE(5,6)
      WRITE(5,*)'  Do you wish to specify apparent link inertias '
      WRITE(5,*)' -- yes or no?'
      WRITE(5,3)
      WRITE(5,*)'  The default values are 1 lb. along X1 and X2 axes.'
      WRITE(5,*)' Default is bypassed if a nondefault option has been'
      WRITE(5,*)' chosen previously.'
      READ(5,150)QUEST
150   FORMAT(A4)

      IF(QUEST .EQ. 'YES')GOTO 155
            GOTO 160

155   WRITE(5,6)
      WRITE(5,6)
      WRITE(5,*)'  Type in your choices for apparent link inertias'
      WRITE(5,*)' in pounds (to .001 pound).  Enter your numbers'
      WRITE(5,*)' accordind to the following format:'
      WRITE(5,3)
      WRITE(5,*)'   Mass in X1 direction,Mass in X2 direction.'
      WRITE(5,3)
      WRITE(5,*)' Your values should not less than .05 pound.'
      WRITE(5,3)
      READ(5,*)DM11,DM22

      IF(DM11 .LT. .049) GOTO 170                              !
            IF(DM22 .LT. .049) GOTO 170

C These next two lines invert DM11 and DM22.  These values are used
C  in the MACRO program in the inverted desired mass matrix.  These
C  calculations are performed here since it is easier code in
C  FORTRAN and the calculations need be performed only once.
C  The result is rounded to 3 significant figures and made an
C  integer value to prepare it for FFIMAC.

      M11INV = IFIX(((1/DM11) * 1000.) + .5)
      M22INV = IFIX(((1/DM22) * 1000.) + .5)
      GOTO 160

170   WRITE(5,4)
      WRITE(5,*)' ILLEGAL MASS VALUE -- TRY AGAIN.'
      GOTO 155
```

```
160     WRITE(5,4)
        WRITE(5,4)
        WRITE(5,4)
        WRITE(5,*)'*******************************************************'
        WRITE(5,3)
        WRITE(5,*)' Do you want to start the controller -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)'  There is a 5 second pause after you hit the '
        WRITE(5,*)' "RETURN" key before the program takes over.'
        WRITE(5,3)
        WRITE(5,*)' STAY CLEAR OF THE YELLOW LINE.'
        WRITE(5,3)
        WRITE(5,*)'*******************************************************'
        READ(5,60)QUEST
60      FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 70
                GOTO 80

70      WRITE(5,6)
        WRITE(5,*)'* PRESS ANY NON-NUMBER KEY TO STOP THE CONTROLLER *'
        WRITE(5,3)
        WRITE(5,*)'*** TOGGLE BELT OR WALL SWITCH TO DISABLE AMPS ***'
        WRITE(5,3)
        WRITE(5,62)
62      FORMAT('0',78('-'))
        WRITE(5,*)'The endpoint positions you selected were:'
        WRITE(5,64)
64      FORMAT('0',7X,'#1',13X,'#2',13X,'#3',13X,'#4',13X,'#5')
        WRITE(5,66)DX1R(1),DX2R(1),DX1R(2),DX2R(2),DX1R(3),DX2R(3),
     *DX1R(4),DX2R(4),DX1R(5),DX2R(5)
66      FORMAT('0',2X,F6.2,',',F6.2,2X,F6.2,',',F6.2,2X,F6.2,',',F6.2,
     *2X,F6.2,',',F6.2,2X,F6.2,',',F6.2)
        WRITE(5,3)
        WRITE(5,3)
        WRITE(5,*)' Command a desired endpoint position by pressing the'
        WRITE(5,*)'number it corresponds to.  Any order is valid.'
        WRITE(5,68)
68      FORMAT('0',78('-'))

        CALL FFI2
```

```
C Print out a table of the user input values.

80      WRITE(5,6)
        WRITE(5,74)
74      FORMAT('0',78('-'))
        WRITE(5,*)'The endpoint positions you selected were:'
        WRITE(5,76)
76      FORMAT('0',7X,'#1',13X,'#2',13X,'#3',13X,'#4',13X,'#5')
        WRITE(5,78)DX1R(1),DX2R(1),DX1R(2),DX2R(2),DX1R(3),DX2R(3),
     *DX1R(4),DX2R(4),DX1R(5),DX2R(5)
78      FORMAT('0',2X,F6.2,',',F6.2,2X,F6.2,',',F6.2,2X,F6.2,',',F6.2,
     *2X,F6.2,',',F6.2,2X,F6.2,',',F6.2)
        WRITE(5,*)'The gains you selected were:'
        WRITE(5,82)
82      FORMAT('0',16X,'K1',7X,'K2',7X,'B1',7X,'B2')
        WRITE(5,84)RK1,RK2,RKV1,RKV2
84      FORMAT('0',14X,F6.3,3X,F6.3,3X,F6.3,3X,F6.3)
        WRITE(5,*)'The desired apparent endpoint mass you selected was:'
        WRITE(5,86)
86      FORMAT('0',16X,'M1',9X,'M2')
        WRITE(5,88)DM11,DM22
88      FORMAT('0',13X,F6.3,5X,F6.3)
        WRITE(5,89)
89      FORMAT('0',78('-'))


85      WRITE(5,3)
        WRITE(7,90)                                              !
90      FORMAT(' Do you want to start over -- yes or no?')
        READ(5,100)QUEST
        WRITE(5,6)
100     FORMAT(A4)


        WRITE(5,6)
        IF(QUEST .EQ. 'YES') GOTO 5
500             WRITE(5,6)
                WRITE(5,6)
                WRITE(5,*)'****************************************'
                WRITE(5,*)'*                                      *'
                WRITE(5,*)'*  Another JJW software production.    *'
                WRITE(5,*)'*                                      *'
                WRITE(5,*)'****************************************'
                WRITE(5,6)


        STOP
        END
```

```
;
;                                MNPPNT.MAC
;                              JOHN WLASSICH
;                                05/20/85
;                            REVISED: 11/20/85
;
;
;       MACRO-11 program to implement a nonlinear force-feedback impedence
; control algorthim in real time.  Sensor signals are digitized via A/D
; converters.  Commands are analog signals via D/A converters. Hardware to
; test this algorithm consists of a serial link two degree of freedom
; mechanism actuated by two torque motors and two servo amplifiers. This
; program is called by MNPPNT, a user-friendly FORTRAN main program.  This
; program calls TRGPNT, a FORTRAN subroutine that calculates sine and cosine.
;
;
;       A/D signal inputs should be as follows:
;
;       1) A/D channel #0 ---- Link #1 potentiometer
;
;       2) A/D channel #2 ---- Link #2 potentiometer
;
;       3) A/D channel #4 ---- Force transducer X output
;
;       4) A/D channel #6 ---- Force transducer Y output
;
;
;       Command outputs should be as follows:
;
;       1) DAC "B" --- Servo amplifier #1 (motor, link #1)
;
;       2) DAC "D" --- Servo amplifier #2 (motor, link #2)
;
;
;       This program is "time optimal" in that every scheme or method I know
; of to reduce the sampling period has been utilized.  Four of the biggest
; time savers are listed below:
;
;                  1) No "CALL's", branches or jumps. Code has been duplicated
;                     where necessary.
;                  2) Elimination of software safety features as exemplified
;                     in PVFMAC.  The level of hardware safety features are
;                     still present                                    !
;                  3) Complete register arithmatic.
;                  4) Control algorithm loop runs "flat out", limited only
;                     only by the time it takes the A/D's to convert and the
;                     time needed for the algorithm to number crunch.
;                  5) Immediate mode for constants and addresses.  Local ($)
;                     labels whenever possible.
;
; The section preceding the control loop does not need these features.
;
;
;  The format used for register usage in this program is :
;
;       R0      =       nonconstant multipliers and divisors
;
;       R1      =       constant scaling factors
;
```

```
;       R2,R3    =       nonconstant multiplicands and dividands
;                        (products and quotients+remainders)
;
;       R4       =       more constant scaling factors or a
;                        temporary memory location (for running sums)
;
;
; The computed command torques are calculated as follows.
;
;     T(command torques) =
;
;       J(transpose) * {[J * I(inverse) * J(transpose)](inverse)} *
;       { M(inverse) * [ (K * (XR-XF) - B * VF) +  (R * FM)]
;       + J * I(inverse) * C - H} - T * FM
;
; The matrices written out are:
;
;
;                              * T1 *
;     T(command torques) =     *    *
;                              * T2 *
;
;
;          * -S1     -S2 *
;     J =  *               *
;          *  C1      C2 *
;
;
;                        * -S1      C1 *
;     J(transpose) =     *               *
;                        * -S2      C2 *
;
;
;                       * I22       -I12C21 *
;                       *                   *
;     I(inverse) =      * -I12C21       I11 *
;                       ---------------------
;                       I11I22 - (I12C21)**2
;
;
;                       * M22    0   *
;                       *            *   * M11INV       0    *
;     M(inverse) =      *  0    M11  * = *                     *
;                       -------------    *    0       M22INV *
;                          M11M22
;
;
;     K = * K1    K2 *
;
;
;                     * X1R    X1F1 *
;     (XR - XF) =     *               *
;                     * X2R    X2F1 *
;
;
;     B = * B1    B2 *
;
;
;          * V1F *
;     VF = *     *
```

```
;          * V2F *
;
;
;       * C2   -S2 *
;    R = *          *
;       * S2    C2 *
;
;
;        * FXMS *
;    FM = *      *
;        * FYMS *
;
;
;       *    0      -I12S21 *      * W1FS *    * CEN1 *
;    C = *                   *  *  *       * = *      *
;       *  I12S21       0    *      * W2FS *    * CEN2 *
;
;
;       * -C1  -C2 *      * W1FS *    * H1 *
;    H = *          *  *  *       * = *    *
;       * -S1  -S2 *      * W2FS *    * H2 *
;
;
;       * S21   C21 *
;    T = *           *                                    !
;       *  0     1  *
;
;
; The (X1,X2) endpoint positions are calculated from:
;
;     X1F1 = C1 + C2
;
;     X2F1 = S1 + S2
;
;     where the ending "1" in X1F1 and X2F1 means "current" sample.
;
;
; The (X1,X2) endpoint velocity is calculated from the endpoint
;  position calculation (itself a function of joint angles which
;  are observed via potentiometers) by a first order, three time step
;  difference equation with smoothing:
;
;     Difference in X1 position = [(current X1 endpoint position) -
;           (X1 position of 3 sampling periods ago)]
;
;     Difference in X2 position = [(current X2 endpoint position) -
;           (X2 position of 3 sampling periodds ago)]
;
;     Current X1 velocity = ( difference in X1 + average velocity from
;           one sampling period past )/2
;
;     Current X2 velocity = ( difference in X2 + average velocity from
;           one sampling period past )/2
;
;  In symbols used by this program:
;
;     V1F1 = [(X1F1 - X1F4) + V1F2]/2
;
;     V2F1 = [(X2F1 - X2F4) + V2F2]/2
;
```

```
; The sampling period is not needed (as in (X1F1-X1F4)/sampling period),
;  it has been factored out and absorbed into the velocity feedback gain.
; The division by two is necessary and can not be factored out because
;  the average velocity from the prior sampling period is also divided
;  by two.  Thus all prior average velocities are divided by increasing
;  multiples of two.
;
;
; Link lengths do not appear anywhere since each link = 1 (ft.).
;
;
; Whenever an element of a matrix is referred to by number (ie. #3),
;  the following format is being used:
;
;                      * #1   #2 *
;     2 by 2 matrix =  *         *
;                      * #3   #4 *
;
;
;                            * #1 *
;     1 by 2 column matrix = *    *
;                            * #2 *
;
;
;     2 by 1 row matrix = * #1 #2 *
;
;
; Whenever two matrices are multiplied, the following method is
;  adhered to :
;
;       * EL1   EL2 *   * EL5   EL6 *      * EL1*EL5+EL2*EL7   EL1*EL6+EL2*EL8 *
;       *           *   *           *  =   *                                    *
;       * EL3   EL4 *   * EL7   EL8 *      * EL3*EL5+EL4*EL7   EL3*EL6+EL4*EL8 *
;
;
;  where the order is:
;
;       EL1*EL5+EL2*EL7 --- evaluated first,
;
;       EL1*EL6+EL2*EL8 --- evaluated second,
;
;       EL3*EL5+EL4*EL7 --- evaluated third,
;
;       EL3*EL6+EL4*EL8 --- evaluated fourth.
;
;  The evaluated element shows up as blocks of code seperated by blank
;  spaces.  The order in which these blocks appear is always the same.
;
;-------------------------------------------------------------------------
```

```
        .TITLE  PNTHAC          ;Object module name

        .GLOBL  TRGPNT          ;Name of FORTRAN subroutine called by
                                ; this MACRO program.

        .MCALL  .TTINR          ;Character transfer programmed request

        .MCALL  .PRINT          ;Programmed request to allow messages
                                ; on CRT from MACRO program

        .ENABLE LC              ;Lower case enable


ADBUF   = 170402        ;A/D buffer address
ADCSR   = 170400        ;A/D control status register address
CH0     = 1             ;A/D channel #0, in octal
CH2     = 1001          ;A/D channel #2, in octal
CH4     = 2001          ;A/D channel #4, in octal
CH6     = 3001          ;A/D channel #6, in octal
DACA    = 170440        ;Too noisy (approx. 3* noise DACB)
DACB    = 170442        ;Link #1 digital to analog converter
DACC    = 170444        ;In reserve for possible later use
DACD    = 170446        ;Link #2 digital to analog converter
DACSCL  = 2048.         ;DAC offset from A/D, 4000 octal
DIGOUT  = 167772        ;TTL output on port #0
DRSCL   = 175.          ;Degrees to 10,000's of radians scaling
                        ; factor.
DTI     = 1111.         ;10 * the inverse of the sampling period
                        ; used to compute joint angular velocity
FSCL1   = 1514.         ;Force transducer scaling factor X1 axis
                        ;Calculated by:
                        ; [(6.2 Pounds)/(4096., A/D units for
                        ; +-10 volts)] * 10E6
FSCL2   = 1514.         ;Same as FSCL1 except for X2 axis
I11     = 687.          ;Actual system, principle axis inertia
                        ; tensor element. In (lbs.*in.**2)*1000.
I22     = 409.          ;Bottom right hand corner element.
I12     = 299.          ;Cross coupling inertia tensor elements.
LK1SCL  = 4395.         ;Link #1 A/D units to degrees * 10E5.
                        ;Calculated by:
                        ; 10E5 * (180 degrees of movement)/
                        ;  (4096., A/D units for +-10V)
LK2SCL  = 4395.         ;Same as LK1SCL except for link #2
RATE    = 5.            ;100 Hz RTC rate.
RTCBUF  = 170422        ;Real-time clock buffer
RTCCSR  = 170420        ;Real-time clock CSR
TICKS   = 1000.         ;No. of ticks per RTC clock cycle.
TOR1SC  = 346.          ;Computed torque to A/D factor, servo #1
TOR2SC  = 1170.         ;Computed torque to A/D factor, servo #2
```

```
;Define common blocks:
; GAINS - used in FORTRAN main program (MNPPNT.FOR).
; ".BLKW 1" used instead of ".WORD 0" so that a memory space is
; reserved but not assigned to 0 when the program is "RUN".
; This allows a DATA statement in FORTRAN to initialize the variables
; to some non zero value.

                .PSECT  GAINS,RW,D,GBL,REL,OVR
        K1:     .BLKW   1       ;X1 axis position feedback gain.
        K2:     .BLKW   1       ;X2 axis position feedback gain.
        B1:     .BLKW   1       ;X1 axis velocity feedback gain.
        B2:     .BLKW   1       ;X2 axis velocity feedback gain.

; X1CORD - user defined in FORTRAN main program (MNPPNT.FOR).

                .PSECT  X1CORD,RW,D,GBL,REL,OVR
        X1R1:   .BLKW   1       ;Reference point #1, X1 coordinate
        X1R2:   .BLKW   1       ;Reference point #2, X1 coordinate
        X1R3:   .BLKW   1       ;Reference point #3, X1 coordinate
        X1R4:   .BLKW   1       ;Reference point #4, X1 coordinate
        X1R5:   .BLKW   1       ;Reference point #5, X1 coordinate

; X2CORD - user defined in FORTRAN main program (MNPPNT.FOR).

                .PSECT  X2CORD,RW,D,GBL,REL,OVR
        X2R1:   .BLKW   1       ;Reference point #1, X2 coordinate
        X2R2:   .BLKW   1       ;Reference point #2, X2 coordinate
        X2R3:   .BLKW   1       ;Reference point #3, X2 coordinate
        X2R4:   .BLKW   1       ;Reference point #4, X2 coordinate
        X2R5:   .BLKW   1       ;Reference point #5, X2 coordinate

; TRIG - used by FORTRAN subroutine (TRGPNT.FOR).

                .PSECT  TRIG,RW,D,GBL,REL,OVR
        T1A:    .WORD   0       ;Link #1 absolute angular position
                                ; in 100's of degrees.
        T2A:    .WORD   0       ;Link #2 absolute angular position
                                ; in 100's of degrees.
        S1:     .WORD   0       ;[Sine of (T1A/100)]*1000
        C1:     .WORD   0       ;[Cosine of (T1A/100)]*1000
        S2:     .WORD   0       ;[Sine of (T2A/100)]*1000
        C2:     .WORD   0       ;[Cosine of (T2A/100)]*1000

; DESMAS - user defined in FORTRAN main program (FFIMNP.FOR).

                .PSECT  DESMAS,RW,D,GBL,REL,OVR
        M11INV: .BLKW   1       ;Inverse of apparent system
                                ; inertia along X1 axis.
        M22INV: .BLKW   1       ;Inverse of apparent system
                                ; inertia along X2 axis.

; TEST - dummy variables to monitor program variables during execution.
;  COMMON with both FORTRAN programs so that values can appear on CRT
;  as an aid to debugging.

                .PSECT  TEST,RW,D,GBL,REL,OVR
        TEST1:  .WORD   0
        TEST2:  .WORD   0
        TEST3:  .WORD   0
```

```
; This next directive tells the linker the MACRO program starts
;  here. This is to prevent the COMMON blocks from overlapping
;  the MACRO code.

        .PSECT  $CODE


; "FFI1" marks the point from which this program begins execution when
;  it is called by the FORTRAN main program, MNPPNT.FOR.  FFI1 is a
;  global label.

FFI1::  CLR     @#DIGOUT        ;Set TTL output to 0 volts


; These next four instructions send a 0 voltage output to the DAC's
;  with BNC connections on the I/O board.  This is a safety feature
;  so that the servo amps do not receive a command voltage before
;  the control algorithm takes over.

        MOV     #DACSCL,@#DACA  ;Clear DAC A output, 2048 = 0 volts
        MOV     #DACSCL,@#DACB  ;Clear DAC B output, 2048 = 0 volts
        MOV     #DACSCL,@#DACC  ;Clear DAC C output, 2048 = 0 volts
        MOV     #DACSCL,@#DACD  ;Clear DAC D output, 2048 = 0 volts


; Reset the RTC from the last run.

        CLR     @#RTCCSR        ;Reset real time clock


        CLRB    @#ADCSR         ;Clear A/D CSR:  1) lower byte
        CLRB    @#ADCSR+1       ;                2) upper byte
        TST     @#ADBUF         ;                3) A/D done flag


; Do some calculations now that determine constant values for
;  use by the control algorithm.

        MOV     #I11,R0         ;Get mechanism inertia tensor element 11
        MOV     #I22,R2         ;Get mechanism inertia tensor element 22
        MUL     R0,R2           ;I11 * I22 (I11 & I22 in 1000's)
        DIV     #1000.,R2       ;Divide product by 1000
        MOV     R2,I11I22       :Save result


;The routine below gets A/D values, use it now to get initial
; potentiometer readings, used as offsets.  Potentiometers are only
; initialized each time MNPPNT.FOR is run from the start since they
; are unlikely to drift during an experimental session like the
; force transducer might.  The user is freed from have to put the
; mechanism in the START position for every control loop run.
```

```
;Channel #0 for link #1 joint angle
        MOV     #CH0,@#ADCSR    ;Convert analog voltage on channel #0
WAIT1:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT1           ;If clear, wait for conversion
        MOV     @#ADBUF,T1ROFF  ;Get digital value from A/D buffer

;Channel #2 for link #2 joint angle
        MOV     #CH2,@#ADCSR    ;Convert analog voltage on channel #2
WAIT2:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT2           ;If clear, wait for conversion
        MOV     @#ADBUF,T2ROFF  ;Get digital value from A/D buffer


; Go back to the main program now that initialization is complete
;  and get user input parameters.

        RETURN


;"FFI2" marks the point from which this program continues execution when
; it is called by the FORTRAN main program, MNPPNT.FOR.  FFI2 is a
; global label.

FFI2::  BIS     #10000,@#44     ;JSW special mode terminal bit, monitor
                                ; will not echo character typed


; "PAUSE" is used later in the program.  This is to allow the
;  storage of three endpoint positions so that the velocity command is
;  made up of deviations from the current mechanism position.

        MOV     #5.,PAUSE       ;Number of loops to count down


;  Let reference position be first user specified position.  New
; reference positions are user chosen in control loop.

        MOV     X1R1,X1R        ;User specified X1 #1 into generic X1
        MOV     X2R1,X2R        ;User specified X2 #1 into generic X2


; Now get force tranducer offsets.  Force transducer reinitialized
;  every run since it has a tendency to drift.

;Channel #4 for measured force in the X direction
        MOV     #CH4,@#ADCSR    ;Convert analog voltage on channel #4
WAIT3:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT3           ;If clear, wait for conversion
        MOV     @#ADBUF,FXMOFF  ;Get digital value from A/D buffer

;Channel #6 for measured force in the Y direction
        MOV     #CH6,@#ADCSR    ;Convert analog voltage on channel #6
WAIT4:  TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     WAIT4           ;If clear, wait for conversion
        MOV     @#ADBUF,FYMOFF  ;Get digital value from A/D buffer
```

```
;Start ten second countdown before control algorithm takes over.
; Clock period = number of ticks per period * (1/ RTC tick rate).

        NEG     TICKS           ;Clock counts up to zero
        MOV     TICKS,@#RTCBUF  ;Load RTC buffer w/ # of ticks
        MOV     RATE,R0         ;ASH requires a register
        ASH     #3,R0           ;Set clock rate (bits 3-5)
        BIS     R0,@#RTCCSR     ;Load RTC control status register
        BIS     #0,@#RTCCSR     ;Set clock in mode 0 (single interval)
        INCB    @#RTCCSR        ;Start clock


;This next loop waits until the RTC is done with its countdown.

COUNT:  TSTB    @#RTCCSR        ;Check overflow bit
        BPL     COUNT           ;If overflow bit not set, wait


;The label REPEAT signifies the top of the control loop.  The only way
; stop the algorithm under normal circumstances is to press any key.
;The digital output is used to measure the sampling period.

REPEAT: INC     @#DIGOUT        ;Toggle a TTL voltage to port #0


;The section below repeats the four channel sampling routine.
;Use it now to get present joint angles and force transducer readings.

;Channel #0 for link #1 joint angle
        MOV     #CH0,@#ADCSR    ;Convert analog voltage on channel #0
1$:     TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     1$              ;If clear, wait for conversion
        MOV     @#ADBUF,T1R     ;Get digital value from A/D buffer
        SUB     T1ROFF,T1R      ;Subtract off initial angle offset

;Channel #2 for link #2 joint angle
        MOV     #CH2,@#ADCSR    ;Convert analog voltage on channel #2
2$:     TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     2$              ;If clear, wait for conversion
        MOV     @#ADBUF,T2R     ;Get digital value from A/D buffer
        SUB     T2ROFF,T2R      ;Subtract off initial angle offset

;Channel #4 for measured force in the X direction
        MOV     #CH4,@#ADCSR    ;Convert analog voltage on channel #4
3$:     TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     3$              ;If clear, wait for conversion
        MOV     @#ADBUF,FXM     ;Get digital value from A/D buffer
        SUB     FXMOFF,FXM      ;Subtract off initial force offset

;Channel #6 for measured force in the Y direction
        MOV     #CH6,@#ADCSR    ;Convert analog voltage on channel #6
4$:     TSTB    @#ADCSR         ;If bit #8 set, conversion done
        BPL     4$              ;If clear, wait for conversion
        MOV     @#ADBUF,FYM     ;Get digital value from A/D buffer
        SUB     FYMOFF,FYM      ;Subtract off initial force offset
```

```
;This next section scales the raw joint angles as they come from
; the A/D's into absolute joint angles in 100's of degrees.

        MOV     #1000.,R1       ;Used repeatedly later

        MOV     #LK1SCL,R0      ;A/D units to 10E5's of degrees, factor
        MOV     T1R,R2          ;Get link #1 A/D joint angle value
        MUL     R0,R2           ;LK1SCL * T1R
        DIV     R1,R2           ;Cancel out a 1000 in LK1SCL
        MOV     R2,T1A          ;Absolute joint angle 1 in 100 * degs.

        MOV     T2R,R2          ;Get link #2 A/D joint angle value
        MUL     R0,R2           ;LK1SCL * T2R
        DIV     R1,R2           ;T2R now in 100 * degrees
        MOV     T1A,R0          ;Get angle #1 in absolute coords:
        ADD     R0,R2           ; add in angle #1
        MOV     #9000.,R0       ;(90 degrees) * 100
        ADD     R0,R2           ; and add in 90 degrees
        MOV     R2,T2A          ;Save absolute joint angle 2


; Now send the absolute joint angles to TRGPNT to get their sine
;  and cosine.  The values sent back are rounded to 3 significant
;  figures and multiplied by 1000 so they can be kept as integers.
; Any values in registers 0 thru 4 must be saved if they are of
;  use after calling TRGPNT.  This is because the FORTRAN subroutine
;  will generate code which uses these registers.

        JSR     PC,TRGPNT       ;Go compute sine and cosine


; These next two lines appear again in this program.  By assigning
;  constants that are used repeatedly to registers, MUL's and
;  DIV's are speeded up.

        MOV     #1.,R1          ;Will be used repeatedly later
        MOV     #1000.,R4       ;Will be used repeatedly later


; Calculate values now that are used repeatedly in the
;  control loop to save time.  Save these values for future use.

        MOV     S2,R2           ;Get sine of T2A
        MOV     C1,R0           ;Get cosine of T1A
        MUL     R0,R2           ;S2 * C1
        DIV     R4,R2           ;/ result by 1000
        MOV     R2,TEMP1        ;Save this result temporarily
        MOV     C2,R2           ;Get cosine of T2A
        MOV     S1,R0           ;Get sine of T1A
        MUL     R0,R2           ;C2 * S1
        DIV     R4,R2           ;/ result by 1000
        MOV     TEMP1,R3        ;Get S2 * C1
        SUB     R2,R3           ;S2 * C1 - C2 * S1 = SIN(T2A-T1A)
        MOV     R3,S21          ;T matrix element #1
```

```
MOV     C2,R2           ;Get C2
MOV     C1,R0           ;Get C1
MUL     R0,R2           ;C2 * C1
DIV     R4,R2           ;/ result by 1000
MOV     R2,TEMP1        ;Save this result temporarily
MOV     S2,R2           ;Get S2
MOV     S1,R0           ;Get S1
MUL     R0,R2           ;S1 * S2
DIV     R4,R2           ;/ result by 1000
MOV     TEMP1,R3        ;Get C2 * C1
ADD     R3,R2           ;C2 * C1 + S2 * S1 = cos(T2A-T1A)
MOV     R2,C21          ;Save the result

MOV     #I12,R0         ;Get coupling inertia tensor element
MOV     C21,R2          ;Get cos(T2A-T1A)
MUL     R0,R2           ;I12 * C21
DIV     R4,R2           ;Leave result in 1000's
MOV     R2,I12C21       ;Save this result for use later
MOV     R2,R0           ;Duplicate this result
MUL     R0,R2           ;Multiply result by itself (square it)
DIV     R4,R2           ;Leave result in 1000's
MOV     I11I22,R3       ;Get I11I22 calculated previously
SUB     R2,R3           ;I11I22 - (I12 * C21)**2
MOV     R3,DI           ;Determinent of matrix I

MOV     C1,R2           ;Get C1
MOV     C2,R0           ;Get C2
ADD     R0,R2           ;C2 + C1 (in 1000's)
MOV     R2,X1F1         ;X1 endpoint position (link length = 1)

MOV     S1,R2           ;Get S1
MOV     S2,R0           ;Get S2
ADD     R0,R2           ;S1 + S2 (in 1000's)
MOV     R2,X2F1         ;X2 endpoint position (link length = 1)

MOV     FXM,R2          ;Get measured force in X direction
MOV     #FSCL1,R0       ;Get force transducer scaling factor
MUL     R0,R2           ;FSCL * FXM
DIV     R4,R2           ;Pounds force in 1000's
MOV     R2,FXMS         ;Save this scaled result

MOV     FYM,R2          ;Get measured force in Y direction
MUL     R0,R2           ;FSCL * FYM
DIV     R4,R2           ;Pounds force in 1000's
MOV     R2,FYMS         ;Save this scaled result
```

```
; Begin computing algorithm.
; Compute  M(inverse)*[ (K * (XR-XF) - B * VF) + (R * FM)].
;
; Note that R uses its top row when multiplying by FM.
;
; Start with K * (XR - XF) for servo amplifier #1 (torque command #1).

        MOV     #1000.,R1       ;Used repeatedly later

        MOV     X1F1,R0         ;Get current X1 endpoint position
        MOV     X1R,R2          ;Get reference position along X1 axis
        SUB     R0,R2           ;X1R - X1F1
        MOV     K1,R0           ;Get position feedback gain for X1 axis
        MUL     R0,R2           ;K1 * (X1R - X1F1)
        DIV     R1,R2           ;Divide out gain scale
        MOV     R2,R4           ;Build { above } in R4

; Now compute B * VF for servo amplifier #1 (torque command #1) and
;  subtract from K * (X1R - X1F1).
;
; First compute position difference from one sample ago.

        MOV     X1F1,R2         ;Get X1F1
        MOV     X1F4,R0         ;X1 endpoint position, 3 samples ago
        SUB     R0,R2           ;X1F1 - X1F4

; Add in average velocity from prior sampling period.

        MOV     V1F2,R0         ;Get V1F2
        ADD     R0,R2           ;(X1F1 - X1F4) + V1F2

;  Smooth, save current average velocity (the next sampling period
; will use it as the average velocity from the prior sampling period)
; and multiply by velocity feedback gain.

        ASR     R2              ;Divide by two
        MOV     R2,V1F2         ;Save current average velocity
        MOV     B1,R0           ;Get velocity gain, along X1 axis
        MUL     R0,R2           ;Multiply average velocity by B1
        DIV     R1,R2           ;Above result scaled back to 1000's
        SUB     R2,R4           ;[K1 * (X1R - X1F1)] - B1 * V1F

; Compute (R * FXMS) and add to [K1*(X1R - X1F1)] + B1*V1F.

        MOV     FXMS,R2         ;Scaled measured force along x axis
        MOV     C2,R0           ;Get C2
        MUL     R0,R2           ;C2 * FXMS
        DIV     R1,R2           ;Divide out one of the 1000 scales
        MOV     R2,TEMP1        ;Save result temporarily
        MOV     FYMS,R2         ;Scaled measured force along y axis
        MOV     S2,R0           ;Get S2
        MUL     R0,R2           ;S2 * FYMS
        DIV     R1,R2           ;Leave result in 1000's
        MOV     TEMP1,R3        ;Get C2 * FXMS
        SUB     R2,R3           ;C2 * FXMS - S2 * FYMS
        ADD     R3,R4           ;Final result is { above } in 1000's
```

```
; These next 5 lines are optional depending on the user's needs.
; Make them inoperative (branch over them or "comment" them away) and
;  the program keeps the final torque command accurate to 3 significant
;  figures however the gains input to MNPPNT are limited (or else
;  the 11/23 will overflow its capacity).  The simultaneous limits are:
;
;       K1 <= 10
;
;       B1 <= 20
;
;       AND : M11 = M22 >= .5
;
; Keep these next 5 lines operative and any gain and desired apparent
;  mass (M11,M22) combination that makes physical sense is permissable.
;  The penalty associated with this choice is the torque commands are
;  accurate to two significant figures only.
; In the last instruction before the next section, "MOV  R2,CE1", R2
;  must be changed to R4.
; Finally, see further instructions at the end of the next section that
;  is essentially a repeat of this section except for servo amp #2.

;       MOV     R4,R2           MUL needs 2 registers, R5 reserved
;       MOV     #1.,R4          Put 1 in R4 to keep arithmatic in R's
;       MUL     R4,R2           Get { above } in 2 register format
;       MOV     #10.,R1         Need 10 for scaling back
;       DIV     R1,R2           Leaves { above } in 100's

        MOV     R4,CE1          ;Save result for later
```

```
; Essentially repeat the entire section above where :
;  compute { M(inverse)*[ (K * (XR-XF) - B * VF) + (R * FM)]}
;  is repeated for servo amplifier #2.
;
; Note that R uses its bottom row when multiplying by FM.
;
; Start with K * (XR - XF) for servo amplifier #2 (torque command #2).

        MOV     #1000.,R1       ;Used repeatedly later

        MOV     X2F1,R0         ;Get current X2 endpoint position
        MOV     X2R,R2          ;Get reference position along X2 axis
        SUB     R0,R2           ;X2R - X2F1
        MOV     K2,R0           ;Get position feedback gain for X2 axis
        MUL     R0,R2           ;K2 * (X2R - X2F1)
        DIV     R1,R2           ;Divide out gain scale
        MOV     R2,R4           ;Build { above } in R4

; Now compute B * VF for servo amplifier #2 (torque command #2) and
;  subtract from K2 * (X2R - X2F1).
;
; First compute position difference from one sample ago.

        MOV     X2F1,R2         ;Get X2F1
        MOV     X2F4,R0         ;X2 endpoint position, 3 samples ago
        SUB     R0,R2           ;X2F1 - X2F4

; Add in average velocity from prior sampling period.

        MOV     V2F2,R0         ;Get V2F2
        ADD     R0,R2           ;(X2F1 - X2F4) + V2F2

;  Smooth, save current average velocity (the next sampling period
; will use it as the average velocity from three sampling periods
; past), and multiply by velocity feedback gain.

        ASR     R2              ;Divide by two
        MOV     R2,V2F2         ;Save current average velocity
        MOV     B2,R0           ;Get velocity gain, along X2 axis
        MUL     R0,R2           ;Multiply average velocity by B2
        DIV     R1,R2           ;Above result scaled back to 1000's
        SUB     R2,R4           ;[K2 * (X2R - X2F1)] - B2 * V2F

; Compute (R * FYMS) and add to [K2 * (X2R - X2F1)] + B2 * V2F.

        MOV     FXMS,R2         ;Scaled measured force along x axis
        MOV     S2,R0           ;Get S2
        MUL     R0,R2           ;S2 * FXMS
        DIV     R1,R2           ;Divide out one of the 1000 scales
        MOV     R2,TEMP1        ;Save result temporarily
        MOV     FYMS,R2         ;Scaled measured force along y axis
        MOV     C2,R0           ;Get C2
        MUL     R0,R2           ;C2 * FYMS
        DIV     R1,R2           ;Leave result in 1000's
        MOV     TEMP1,R3        ;Get CS2 * FXMS
        ADD     R3,R2           ;S2 * FXMS + C2 * FYMS
        ADD     R2,R4           ;Final result is { above } in 1000's
```

```
; These next 5 lines are optional depending on the user's needs.
; Make them inoperative (branch over them or "comment" them away) and
;  the program keeps the final torque command accurate to .001,
;  however the gains input to MNPPNT are limited (or else the 11/23
;  will overflow its capacity).  The limits are:
;
;       K2 <= 10
;
;       B2 <= 20
;
;       AND : M11 = M22 >= .5
;
; Keep these next 5 lines operative and any gain and desired apparent
;  mass (M11,M22) combination that makes physical sense is permissable.
;  The penalty associated with this choice is the torque commands are
;  accurate to two significant figures only.
; In the last instruction before the next section, "MOV  R2,CE2", R2
;  must be changed to R4.
; Finally, see further instructions at the "T * FMS" section.

;       MOV     R4,R2           MUL needs 2 registers, R5 reserved
;       MOV     #1.,R4          Put 1 in R4 to keep arithmatic in R's
;       MUL     R4,R2           Get { above } in 2 register format
;       MOV     #10.,R1         Need 10 to scale back
;       DIV     R1,R2           Leaves { above } in 100's

        MOV     R4,CE2          ;Save result for later

; Now multiply by M(inverse).

        MOV     #1.,R1          ;Used later
        MOV     #1000.,R4       ;Used later

; For torque command #1.

        MOV     CE1,R2          ;Get servo #1 { } result
        MOV     M11INV,R0       ;Get 1/M11 in 1000's
        MUL     R0,R2           ;CE1 * M11INV
        DIV     R4,R2           ;Leave result in 100's or 1000's
        MOV     R2,CE1          ;Reuse CE1 as a memory location

; For torque command #2.

        MOV     CE2,R2          ;Get servo #2 { } result
        MOV     M22INV,R0       ;Get 1/M22 in 1000's
        MUL     R0,R2           ;CE2 * M22INV
        DIV     R4,R2           ;Leave result in 100's or 1000's
        MOV     R2,CE2          ;Reuse CE2 as a memory location
```

```
; Calculate J(transpose) * {[J * I(inverse) * J(transpose)](inverse)}
;
; Start by calculating J * I(inverse).  Here, the calculations are not
;  decoupled into a section each for torque command #1 and torque
;  command #2.
; The resulting 2 by 2 matrix is given the following element names:
;
;
;                        * EL1   EL2 *
;       J * I(inverse) = *           *
;                        * EL3   EL4 *
;

        MOV     #1000.,R1       ;Used later

        MOV     S1,R2           ;Get S1
        MOV     #I22,R0         ;Inverse inertia tensor element #4
        MUL     R0,R2           ;I22 * S1
        DIV     R1,R2           ;Leave result in 1000's
        MOV     R2,R4           ;Save result temporarily
        MOV     S2,R2           ;Get S2
        MOV     I12C21,R0       ;Inverse inertia tensor element #3
        MUL     R0,R2           ;S2 * I12C21
        DIV     R1,R2           ;Leave result in 1000's
        MOV     R4,R3           ;Get I22 * S1
        SUB     R3,R2           ;S2 * I12C21 - I22 * S1
        MOV     R2,EL1          ;Save result

        MOV     S1,R2           ;Get S1
        MUL     R0,R2           ;I12C21 * S1
        DIV     R1,R2           ;Leave answer in 1000's
        MOV     R2,R4           ;Save result in temporary location
        MOV     S2,R2           ;Get S2
        MOV     #I11,R0         ;Inverse inertia tensor element #4
        MUL     R0,R2           ;S2 * I11
        DIV     R1,R2           ;Divide out 1000
        MOV     R4,R3           ;Get I12C21 * S1
        SUB     R2,R3           ;I12C21 * S1 - S2 * I11
        MOV     R3,EL2          ;Save result

        MOV     C1,R2           ;Get C1
        MOV     #I22,R0         ;Get I22
        MUL     R0,R2           ;C1 * I22
        DIV     R1,R2           ;Leave result in 1000's
        MOV     R2,R4           ;Save result temporarily
        MOV     C2,R2           ;Get C2
        MOV     I12C21,R0       ;Get I12C21
        MUL     R0,R2           ;C2 * I12C21
        DIV     R1,R2           ;This is jjw's code
        MOV     R4,R3           ;Get C1 * I22
        SUB     R2,R3           ;C1 * I22 - C2 * I12C21
        MOV     R3,EL3          ;Save result
```

```
        MOV     C1,R2           ;Get C1
        MUL     R0,R2           ;C1 * I12C21
        DIV     R1,R2           ;Get result in 1000's
        MOV     R2,R4           ;Save temporarily
        MOV     C2,R2           ;Get C2
        MOV     #I11,R0         ;Get I11
        MUL     R0,R2           ;C2 * I11
        DIV     R1,R2           ;Leave result in 1000's
        MOV     R4,R3           ;Get C1 * I12C21
        SUB     R3,R2           ;C2 * I11 - C1 * I12C21
        MOV     R2,EL4          ;Save result


; Calculate [J * I(inverse)] * J(transpose).
;  The resulting 2 by 2 matrix is given the element names:
;
;
;                                                * EL5   EL6 *
;       [J * I(inverse)] * J(transpose) =        *           *
;                                                * EL7   EL8 *
;

        MOV     S1,R2           ;Put S1 into multiplicand register
        MOV     EL1,R0          ;Get EL1 from J * I(inverse)
        MUL     R0,R2           ;S1 * EL1
        DIV     R1,R2           ;Divide by 1000 leaving result in 1000's
        MOV     R2,R4           ;Save in temporary location register
        MOV     S2,R2           ;Get S2
        MOV     EL2,R0          ;Get EL2 from J * I(inverse)
        MUL     R0,R2           ;EL2 * S2
        DIV     R1,R2           ;Divide out one of the scaling 1000's
        ADD     R4,R2           ;S1 * EL1 + EL2 * S2
        NEG     R2              ;-(S1 * EL1 + EL2 * S2)
        MOV     R2,EL5          ;Save result

        MOV     C2,R2           ;Get C2
        MUL     R0,R2           ;EL2 * C2
        DIV     R1,R2           ;Divide out a 1000
        MOV     R2,R4           ;Save result temporarily
        MOV     C1,R2           ;Get C1
        MOV     EL1,R0          ;Put EL1 into multiplyer register
        MUL     R0,R2           ;C1 * EL1
        DIV     R1,R2           ;Divide out a 1000
        ADD     R4,R2           ;C1 * EL1 + EL2 * C2
        MOV     R2,EL6          ;Save result

        MOV     S1,R2           ;Put S1 into multiplicand register
        MOV     EL3,R0          ;Get EL3 from J * I(inverse)
        MUL     R0,R2           ;S1 * EL3
        DIV     R1,R2           ;Divide by 1000 leaving result in 1000's
        MOV     R2,R4           ;Save in temporary location register
        MOV     S2,R2           ;Get S2
        MOV     EL4,R0          ;Get EL4 from J * I(inverse)
        MUL     R0,R2           ;EL4 * S2
        DIV     R1,R2           ;Divide out one of the scaling 1000's
        ADD     R4,R2           ;S1 * EL3 + EL4 * S2
        NEG     R2              ;-(S1 * EL3 + EL4 * S2)
        MOV     R2,EL7          ;Save result
```

```
        MOV     C2,R2           ;Get C2
        MUL     R0,R2           ;EL4 * C2
        DIV     R1,R2           ;Divide out a 1000
        MOV     R2,R4           ;Save result temporarily
        MOV     C1,R2           ;Get C1
        MOV     EL3,R0          ;Put EL3 into multiplier register
        MUL     R0,R2           ;C1 * EL3
        DIV     R1,R2           ;Divide out a 1000
        ADD     R4,R2           ;C1 * EL3 + EL4 * C2
        MOV     R2,EL8          ;Save result

; Invert the last matrix :
;
;                                 *  EL8   -EL6 *
;       * EL5   EL6 * -1          *              *
;       *           *     =     * -EL7    EL5 *, and this matrix is renamed:
;       * EL7   EL8 *            ---------------
;                              EL5*EL8-(EL6*EL7)
;
;
;       * EL5   EL6 *
;       *           *
;       * EL7   EL8 *
;       -------------
;            DMT
;
;  just so variable names that are temporary, are used over.
;

        MOV     EL6,R0          ;Put EL6 from [J*I(inv)*J(trans)] in R0
        MOV     EL7,R2          ;Put EL7 from [J*I(inv)*J(trans)] in R2
        MUL     R0,R2           ;EL6 * EL7
        DIV     R1,R2           ;Divide out one of the scaling 1000
        MOV     R2,R4           ;Save result temporarily
        MOV     EL5,R0          ;Put EL5 from [J*I(inv)*J(trans)] in R0
        MOV     EL8,R2          ;Put EL8 from [J*I(inv)*J(trans)] in R2
        MUL     R0,R2           ;EL5 * EL8
        DIV     R1,R2           ;Divide out one of the scaling 1000
        SUB     R4,R2           ;EL5 * EL8 - EL6 * EL7
        MOV     R2,DMT          ;Save result

        MOV     EL5,R4          ;Put EL5 in temporary register
        MOV     EL8,EL5         ;The name "EL5" becomes the value EL8
        MOV     R4,EL8          ;The name "EL8" becomes the value EL5
        NEG     EL6             ;"EL6" = -EL6
        NEG     EL7             ;"EL7" = -EL7
```

```
; Calculate: J(transpose) * (the prior inverted matrix).
;  The resulting 2 by 2 matrix element variables are :
;
;
;                        * EL5   EL6 *    * EL9    EL10 *
;      J(transpose)   *  *           * = *             *
;                        * EL7   EL8 *    * EL11   EL12 *
;

        MOV     EL5,R2          ;Put EL5 in multiplicand register
        MOV     S1,R0           ;Put S1 in multiplier register
        MUL     R0,R2           ;EL5 * S1
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        MOV     R2,R4           ;Save result in temporary register
        MOV     EL7,R2          ;Get EL7
        MOV     C1,R0           ;Get C1
        MUL     R0,R2           ;EL7 * C1
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        SUB     R4,R2           ;C1 * EL7 - S1 * EL5
        MOV     R2,EL9          ;Save result

        MOV     EL8,R2          ;Get EL8
        MUL     R0,R2           ;EL8 * C1
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        MOV     R2,R4           ;Save temporarily
        MOV     EL6,R2          ;Get EL6
        MOV     S1,R0           ;Get S1
        MUL     R0,R2           ;EL6 * S1
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        SUB     R2,R4           ;EL8 * C1 - EL6 * S1
        MOV     R4,EL10         ;Save result

        MOV     EL5,R2          ;Put EL5 in multiplicand register
        MOV     S2,R0           ;Put S2 in multiplier register
        MUL     R0,R2           ;EL5 * S2
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        MOV     R2,R4           ;Save result in temporary register
        MOV     EL7,R2          ;Get EL7
        MOV     C2,R0           ;Get C2
        MUL     R0,R2           ;EL7 * C2
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        SUB     R4,R2           ;C2 * EL7 - S2 * EL5
        MOV     R2,EL11         ;Save result

        MOV     EL8,R2          ;Get EL8
        MUL     R0,R2           ;EL8 * C2
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        MOV     R2,R4           ;Save temporarily
        MOV     EL6,R2          ;Get EL6
        MOV     S2,R0           ;Get S2
        MUL     R0,R2           ;EL6 * S2
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        SUB     R2,R4           ;EL8 * C2 - EL6 * S2
        MOV     R4,EL12         ;Save result
```

```
;  Now that J(transpose) * {[J * I(inverse) * J(transpose)](inverse)} has
; been calculated, J * I(inverse) * C can be calculated and along with H
; added to M(inverse)*[ (K * (XR-XF) - B * VF) + (R * FM)].
;
;  First calculate the elements that make up C and H. The only values not
; already calculated are the joint angular velocities, W1F and W2F.  Get
; these values much in the same way as V1F and V2F were calculated.
; Approximate the differential of T1A and T2A with a difference equation
; and smooth.
;

;  T1A and T2A must be converted from 100's of degrees to 10,000's of
; radians.  The 10E5 is necessary to keep three significant
; figures of accuracy.
;

        MOV     T1A,R2          ;Get T1A in 100's of degrees
        MOV     #DRSCL,R0       ;Get deg to 10E5*rad scaling factor
        MUL     R0,R2           ;Convert to 10,000's of radians
        DIV     #100.,R2        ;Divide out 100 from 100's of degrees
        MOV     R2,T1AR         ;Save absolute joint angle 1 (radians)

        MOV     T2A,R2          ;Get T2A in 100's of degrees
        MUL     R0,R2           ;Convert to 10,000's of radians
        DIV     #100.,R2        ;Divide out 100 from 100's of degrees
        MOV     R2,T2AR         ;Save absolute joint angle 1 (radians)

; Compute absolute joint angle 1 difference from one sample ago.

        MOV     T1AR,R2         ;Get T1AR
        MOV     T1AR2,R0        ;Absolute joint angle 1, 1 sample ago
        SUB     R0,R2           ;T1AR - T1AR2
        MOV     #DTI,R4         ;Get inverse of time T1AR to T1AR2
        MUL     R4,R2           ;DTI * (T1AR - T1AR2)
        DIV     #10.,R2         ;Divide out the factor of 10 in DTI

; Add in average angular velocity from prior sampling period.

        MOV     W1F2,R0         ;Get W1F2
        ADD     R0,R2           ;(T1AR - T1AR2) + W1F2

;  Smooth, save current average angular velocity (the next sampling
; period will use it as the average velocity from the prior sampling
; period) and square it.

        ASR     R2              ;Divide by two
        MOV     R2,W1F2         ;Save current average angular velocity
        MOV     R2,R0           ;Duplicate W1F2 for squaring
        MUL     R0,R2           ;W1F2 * W1F2
        DIV     #10000.,R2      ;Above result scaled back to 10000's
        MOV     R2,W1FS         ;Save W1F2(squared)

; Compute absolute joint angle 2 difference from one sample ago.

        MOV     T2AR,R2         ;Get T2AR
        MOV     T2AR2,R0        ;Absolute joint angle 2, 1 sample ago
        SUB     R0,R2           ;T2AR - T2AR2
        MUL     R4,R2           ;DTI * (T2AR - T2AR2)
        DIV     #10.,R2         ;Divide out the factor of 10 in DTI
```

```
; Add in average angular velocity from prior sampling period.

        MOV     W2F2,R0         ;Get W2F2
        ADD     R0,R2           ;(T2AR - T2AR2) + W2F2

;  Smooth, save current average angular velocity (the next sampling
; period will use it as the average velocity from the prior sampling
; period) and square it.

        ASR     R2              ;Divide by two
        MOV     R2,W2F2         ;Save current average angular velocity
        MOV     R2,R0           ;Duplicate W2F2 for squaring
        MUL     R0,R2           ;W2F2 * W2F2
        DIV     #10000.,R2      ;Above result scaled back to 10000's
        MOV     R2,W2FS         ;Save W2F2(squared)


;  Form C :
;
;       * CEN1 *
;       *      *
;       * CEN2 *
;

        MOV     #I12,R0         ;Get I12
        MOV     S21,R2          ;Get S21
        MUL     R0,R2           ;I12 * S21
        DIV     R1,R2           ;Scale back to 1000's
        MOV     R2,R4           ;Save this result temporarily
        MOV     W2FS,R0         ;Get W2FS
        MUL     R0,R2           ;(I12 * S21) * W2FS
        DIV     #10000.,R2      ;Scale back to 1000's
        NEG     R2              ;-(I12 * S21) * W2S
        MOV     R2,CEN1         ;Save this result

        MOV     R4,R2           ;Get (I12 * S21)
        MOV     W1FS,R0         ;Get W1FS
        MUL     R0,R2           ;(I12 * S21) * W1FS
        DIV     #10000.,R2      ;Scale back to 1000's
        MOV     R2,CEN2         ;Save this result


;  Compute J * I(inverse) * C :
;
;       * EL1  EL2 *    * CEN1 *    * CENP1 *
;       *          * * *       * = *       *
;       * EL3  EL4 *    * CEN2 *    * CENP2 *
;
; Divide through by the determinent of I (DI) later.

        MOV     EL1,R0          ;Get EL1
        MOV     CEN1,R2         ;Get CEN1
        MUL     R0,R2           ;EL1 * CEN1
        DIV     R1,R2           ;Scale back to 1000's
        MOV     R2,R4           ;Save this result temporarily
        MOV     EL2,R0          ;Get EL2
        MOV     CEN2,R2         ;Get CEN2
        MUL     R0,R2           ;EL2 * CEN2
        DIV     R1,R2           ;Scale back to 1000's
```

```
        ADD     R2,R4           ;(EL1 * CEN1) + (EL2 * CEN2)
        MOV     R4,CENP1        ;Save this result for later

        MOV     EL3,R0          ;Get EL3
        MOV     CEN1,R2         ;Get CEN1
        MUL     R0,R2           ;EL3 * CEN1
        DIV     R1,R2           ;Scale back to 1000's
        MOV     R2,R4           ;Save this result temporarily
        MOV     EL4,R0          ;Get EL4
        MOV     CEN2,R2         ;Get CEN2
        MUL     R0,R2           ;EL4 * CEN2
        DIV     R1,R2           ;Scale back to 1000's
        ADD     R2,R4           ;(EL3 * CEN1) + (EL3 * CEN2)
        MOV     R4,CENP2        ;Save this result for later

;  Divide through by the determinent of I (DI) :
;
;       1000      * CENP1 *    * CENP1 *
;     -------- * *         * = *         *
;        DI       * CENP2 *    * CENP2 *
;

        MOV     DI,R0           ;Get DI
        MOV     CENP1,R2        ;Get CENP1
        MUL     R1,R2           ;1000 * CENP1
        DIV     R0,R2           ;(1000 * CENP1)/DI
        MOV     R2,CENP1        ;Save this result for later

        MOV     CENP2,R2        ;Get CENP2
        MUL     R1,R2           ;1000 * CENP2
        DIV     R0,R2           ;(1000 * CENP2)/DI
        MOV     R2,CENP2        ;Save this result for later


;  Compute the elements of H and save :
;
;       * (C1 * W1FS + C2 * W2FS) *    * H1 *
;       *                         * = *    *
;       * (S1 * W1FS + S2 * W2FS) *    * H2 *
;
; Notice that the elements of H should be all negative, but since
; H is to be subtracted later on, the product of the negatives result
; in an add.  By not taking the negatives twice, computation'time
; is saved.

        MOV     C1,R0           ;Get C1
        MOV     W1FS,R2         ;Get W1FS
        MUL     R0,R2           ;C1 * W1FS
        DIV     #10000.,R2      ;Scale back to 1000's
        MOV     R2,R4           ;Save this result temporarily
        MOV     C2,R0           ;Get C2
        MOV     W2FS,R2         ;Get W2FS
        MUL     R0,R2           ;C2 * W2FS
        DIV     #10000.,R2      ;Scale back to 1000's
        ADD     R2,R4           ;(C1 * W1FS) + (C2 * W2FS)
        MOV     R4,H1           ;Save this result for later
```

```
        MOV     S1,R0           ;Get S1
        MOV     W1FS,R2         ;Get W1FS
        MUL     R0,R2           ;S1 * W1FS
        DIV     #10000.,R2      ;Scale back to 1000's
        MOV     R2,R4           ;Save this result temporarily
        MOV     S2,R0           ;Get S2
        MOV     W2FS,R2         ;Get W2FS
        MUL     R0,R2           ;S2 * W2FS
        DIV     #10000.,R2      ;Scale back to 1000's
        ADD     R2,R4           ;(S1 * W1FS) + (S2 * W2FS)
        MOV     R4,H2           ;Save this result for later


;  Now add in [J * I(inverse) * C] and H to
; M(inverse)*[ (K * (XR-XF) - B * VF) + (R * FM)] :
;
;       * CENP1 *   * H1 *   * CE1 *   * CE1 *
;       *       * + *    * + *     * = *     *
;       * CENP2 *   * H2 *   * CE2 *   * CE2 *
;

        MOV     CENP1,R0        ;Get CENP1
        MOV     H1,R2           ;Get H1
        MOV     CE1,R4          ;Get CE1
        ADD     R0,R2           ;CENP1 + H1
        ADD     R2,R4           ;(CENP1 + H1) + CE1
        MOV     R4,CE1          ;Save result as new CE1

        MOV     CENP2,R0        ;Get CENP2
        MOV     H2,R2           ;Get H2
        MOV     CE2,R4          ;Get CE2
        ADD     R0,R2           ;CENP2 + H2
        ADD     R2,R4           ;(CENP2 + H2) + CE2
        MOV     R4,CE2          ;Save result as new CE2
```

```
; Calculate: (prior 2 by 2 square matrix)*(prior 1 by 2 column matrix).
;
;
;       * EL9    EL10 *     * CE1 *   * CEL1 *
;       *              *  *  *     * = *      *
;       * EL11   EL12 *     * CE2 *   * CEL2 *
;

        MOV     EL9,R2          ;Get EL9
        MOV     CE1,R0          ;Get CE1
        MUL     R0,R2           ;EL9 * CE1
        DIV     R1,R2           ;Divide out the scaling 1000
        MOV     R2,R4           ;Save temporarily
        MOV     EL10,R2         ;Get EL10
        MOV     CE2,R0          ;Get CE2
        MUL     R0,R2           ;EL10 * CE2                    !
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        ADD     R4,R2           ;EL9 * CE1 + EL10 * CE2
        MOV     R2,CEL1         ;Save result

        MOV     EL12,R2         ;Get EL12
        MUL     R0,R2           ;EL12 * CE2
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        MOV     R2,R4           ;Save temporarily
        MOV     EL11,R2         ;Get EL11
        MOV     CE1,R0          ;Get CE1
        MUL     R0,R2           ;EL11 * CE1
        DIV     R1,R2           ;Divide out one of the scaled 1000's
        ADD     R4,R2           ;EL21 * CE2 + EL11 * CE1
        MOV     R2,CEL2         ;Save result


; Calculate DI/DMT.

        MOV     DI,R2           ;Get determinent of I
        MUL     R1,R2           ;1000 * DI
        MOV     DMT,R0          ;Determinent of {J * I(inv) * J(trans)}
        DIV     R0,R2           ;(1000 * DI)/DMT in 1000's
        MOV     R2,R0           ;Put result into multiplier register

; Calculate :
;
;
;       1000 * DI       * CEL1 *    * SUM1 *
;       ---------   *   *      * = *      *
;          DMT          * CEL2 *    * SUM2 *
;

        MOV     CEL1,R2         ;Get CEL1
        MUL     R0,R2           ;(1000*DI/DMT)*CEL1
        DIV     R1,R2           ;Divide out the scaling 1000
        MOV     R2,SUM1         ;Save result (in 100's)

        MOV     CEL2,R2         ;Get CEL2
        MUL     R0,R2           ;(1000*DI/DMT)*CEL2
        DIV     R1,R2           ;Divide out the scaled 1000
        MOV     R2,SUM2         ;Save result (in 100's)
```

```
; Calculate T * FM :
;
;                    * SUM3 *
;         T  * FMS = *      *
;                    * SUM4 *
;
; For the three significant figures option, the following changes
;  must be made:
;
;       1) In the next instruction, "MOV  #10000.,R1", #10000. must
;          be changed to #1000..
;
;       2) The 4 instructions beginning with "MOV  #10.,R3" and
;          ending with "DIV  R3,R0" inoperative.
;
;       3) The first instruction of the next section, "MOV  #100.,R1",
;          #100. must be changed to #1000..

        MOV     #1000.,R1       ;Used later

        MOV     S21,R2          ;Sin(T2R-T1R) into multiplicand register
        MOV     FXMS,R0         ;Scaled measured force, x direction
        MUL     R0,R2           ;FXMS * S21
        DIV     R1,R2           ;Scale result back to same as SUM1
        MOV     R2,R4           ;Save temporarily
        MOV     C21,R2          ;Get cos(T2E-T1R)
        MOV     FYMS,R0         ;Scaled measured force, y direction
        MUL     R0,R2           ;C21 * FYMS
        DIV     R1,R2           ;Scale result back to same as SUM1
        ADD     R4,R2           ;FXMS * S21 + C21 * FYMS
        MOV     R2,SUM3         ;Save result

;       MOV     #10.,R3         Use later
;       MOV     #1.,R4          Keep arithmatic in registers
;       MUL     R4,R0           Set up so FYMS in R0,R1 format
;       DIV     R3,R0           FYMS now in 100's
        MOV     R0,SUM4         ;FYMS * 1 = FYMS


; Calculate :
;
;       * SUM1 *    * SUM3 *    * CMD1 *
;       *      * - *      * = *      *
;       * SUM2 *    * SUM4 *    * CMD2 *
;

        MOV     #1000.,R1       ;Use later

        MOV     SUM1,R2         ;Get SUM1
        MOV     SUM3,R0         ;Get SUM3
        SUB     R0,R2           ;SUM1 - SUM3
        MOV     #TOR1SC,R0      ;Get torque to DAC scaling for servo #1
        MUL     R0,R2           ;#TOR1SC * (SUM1 - SUM3)
        DIV     R1,R2           ;#TOR1SC * (SUM1 - SUM3) in 1's
        MOV     R2,CMD1         ;Save result
```

```
        .:OV    SUM2,R2         ;Get SUM2
        MOV     SUM4,R0         ;Get SUM4
        SUB     R0,R2           ;SUM2 - SUM4
        MOV     #TOR2SC,R0      ;Get torque to DAC scaling for servo #2
        MUL     R0,R2           ;#TOR2SC * (SUM2 - SUM4)
        DIV     R1,R2           ;#TOR2SC * (SUM2 - SUM4) in 1's
        MOV     R2,CMD2         ;Save result


; This next short section lets the program loop 3 times before sending
;  out a command, at the start of every run.

        MOV     PAUSE,R0        ;Get current value of countdown
        TST     R0              ;Compare PAUSE to zero
        BEQ     10$             ;If = 0, countdown done, fire commands
        DEC     R0              ;If not equal to zero, decrement PAUSE
        MOV     R0,PAUSE        ;Save decremented value of PAUSE
        BR      11$             ;Go shift down X1,X2 position values


; Before firing command to DAC's, scale to DAC units.
;   DAC = 4000(octal) - A/D

10$:    MOV     #DACSCL,R1      ;Get A/D to DAC offset
        MOV     CMD1,R2         ;Get servo command #1
        NEG     R2              ;Negative of command #1 in A/D units
        ADD     R1,R2           ;Add offset
        MOV     R2,@#DACB       ;Fire out command (#1 to DAC B)

        MOV     CMD2,R2         ;Get servo command #2
        NEG     R2              ;Negative of command #2 in A/D units
        ADD     R1,R2           ;Add offset
        MOV     R2,@#DACD       ;Fire out command (#2 to DAC D)


; The next instructions take the current X1 endpoint position
;  and saves it as a X1 position of 3 sampling periods past.

11$:    MOV     X1F3,X1F4       ;X1(2 past) becomes X1(3 past)
        MOV     X1F2,X1F3       ;X1(1 past) becomes X1(2 past)
        MOV     X1F1,X1F2       ;X1(present) becomes X1(1 sample past)

; The next instructions take the current X2 endpoint position
;  and saves it as a X2 position of 3 sampling periods past.

        MOV     X2F3,X2F4       ;X2(2 past) becomes X2(3 past)
        MOV     X2F2,X2F3       ;X2(1 past) becomes X2(2 past)
        MOV     X2F1,X2F2       ;X2(present) becomes X2(1 sample past)

; The next instructions take the current absolute joint angle 1
;  and saves it as an angle of 1 sampling period past.

        MOV     T1AR,T1AR2      ;T1AR(present) to T1AR(1 sample past)

; The next instructions take the current absolute joint angle 2
;  and saves it as an angle of 1 sampling period past.

        MOV     T2AR,T2AR2      ;T2AR(present) becomes T2AR(1 sample past)
```

```
        .TTINR                  ;Check for key strike, no character
                                ; transferred if carry set
        BCC     STRUCK          ;If no key strike, continue with the
        JMP     REPEAT          ; control algorithm loop


STRUCK: CMP     R0,#61          ;See if key struck is #1
        BNE     20$             ;If key struck not #1, go to 20$
        MOV     X1R1,X1R        ;User specified X1 into generic X1
        MOV     X2R1,X2R        ;User specified X2 into generic X2
        BR      40$             ;Search no further, go to 40$

20$:    CMP     R0,#62          ;See if key struck is #2
        BNE     21$             ;If key struck not #2, go to 21$
        MOV     X1R2,X1R        ;User specified X1 into generic X1
        MOV     X2R2,X2R        ;User specified X2 into generic X2
        BR      40$             ;Search no further, go to 40$

21$:    CMP     R0,#63          ;See if key struck is #3
        BNE     22$             ;If key struck not #3, go to 22$
        MOV     X1R3,X1R        ;User specified X1 into generic X1
        MOV     X2R3,X2R        ;User specified X2 into generic X2
        BR      40$             ;Search no further, go to 40$

22$:    CMP     R0,#64          ;See if key struck is #4
        BNE     23$             ;If key struck not #4, go to 23$
        MOV     X1R4,X1R        ;User specified X1 into generic X1
        MOV     X2R4,X2R        ;User specified X2 into generic X2
        BR      40$             ;Search no further, go to 40$

23$:    CMP     R0,#65          ;See if key struck is #5
        BNE     STOP            ;If key struck not #1, go to STOP
        MOV     X1R5,X1R        ;User specified X1 into generic X1
        MOV     X2R5,X2R        ;User specified X2 into generic X2
        BR      40$             ;Search no further, go to 40$


40$:    CLC                     ;Clear carry, set when a key struck
        JMP     REPEAT          ;Continue with control loop


;This next section shuts down all hardware and returns control
; to the FORTRAN main program, MNPPNT.FOR.

STOP:   CLR     @#DIGOUT        ;Clear sampling meter
        MOV     #DACSCL,@#DACA  ;Clear DAC A output, 2048 = 0 volts
        MOV     #DACSCL,@#DACB  ;Clear DAC B output, 2048 = 0 volts
        MOV     #DACSCL,@#DACC  ;Clear DAC C output, 2048 = 0 volts
        MOV     #DACSCL,@#DACD  ;Clear DAC D output, 2048 = 0 volts
        BIC     #10000,@#44     ;Reset JSW
        CLR     @#RTCCSR        ;Reset RTC control status register
        CLRB    @#ADCSR         ;Clear A/D CSR:  1) lower byte
        CLRB    @#ADCSR+1       ;                2) upper byte
        TST     @#ADBUF         ;                3) A/D done flag
        NEG     TICKS           ;Reset # of ticks for clock cycle
        RETURN                  ;Go back to main program
```

```
;Variables not already defined :

C21:      .WORD   0       ;Cos(T2A -T1A)
CE1:      .WORD   0       ;Temporary 1 * 2 matrix element
CE2:      .WORD   0       ;Temporary 1 * 2 matrix element
CEL1:     .WORD   0       ;Temporary 1 * 2 matrix element
CEL2:     .WORD   0       ;Temporary 1 * 2 matrix element
CEN1:     .WORD   0       ;Top row of centripetal matrix
CEN2:     .WORD   0       ;Bottom row of centripetal matrix
CENP1:    .WORD   0       ;Top row, J*I(inv)*C matrix
CENP2:    .WORD   0       ;Bottom row, J*I(inv)*C matrix
CMD1:     .WORD   0       ;Servo amp #1 command before DAC offset
CMD2:     .WORD   0       ;Servo amp #2 command before DAC offset
DI:       .WORD   0       ;Determinent rotational inertia tensor
DMT:      .WORD   0       ;Determinent of {J*I(inverse)*P}
EL1:      .WORD   0       ;Temporary 2 * 2 matrix element
EL2:      .WORD   0       ;Temporary 2 * 2 matrix element
EL3:      .WORD   0       ;Temporary 2 * 2 matrix element
EL4:      .WORD   0       ;Temporary 2 * 2 matrix element
EL5:      .WORD   0       ;Temporary 2 * 2 matrix element
EL6:      .WORD   0       ;Temporary 2 * 2 matrix element
EL7:      .WORD   0       ;Temporary 2 * 2 matrix element
EL8:      .WORD   0       ;Temporary 2 * 2 matrix element
EL9:      .WORD   0       ;Temporary 2 * 2 matrix element
EL10:     .WORD   0       ;Temporary 2 * 2 matrix element
EL11:     .WORD   0       ;Temporary 2 * 2 matrix element
EL12:     .WORD   0       ;Temporary 2 * 2 matrix element
FXM:      .WORD   0       ;Force transducer output in X direction
FYM:      .WORD   0       ;Force transducer output in Y direction
FXMOFF:   .WORD   0       ;Initial force transducer X offset
FYMOFF:   .WORD   0       ;Initial force transducer Y offset
FXMS:     .WORD   0       ;Scaled force output in X direction
FYMS:     .WORD   0       ;Scaled force output in Y direction
H1:       .WORD   0       ;Top row of H column H matrix
H2:       .WORD   0       ;Bottom row of column H matrix
I11I22:   .WORD   0       ;I11 * I22 * 1000
I12C21:   .WORD   0       ;I12 * C21 * 1000
PAUSE:    .WORD   0       ;Number of noncommand loops
S21:      .WORD   0       ;Sin(T2A - T1A) * 1000
SUM1:     .WORD   0       ;Temporary 1 * 2 matrix element
SUM2:     .WORD   0       ;Temporary 1 * 2 matrix element
SUM3:     .WORD   0       ;Temporary 1 * 2 matrix element
SUM4:     .WORD   0       ;Temporary 1 * 2 matrix element
TEMP1:    .WORD   0       ;Temporary memory location #1
T1AR:     .WORD   0       ;Joint angle 1 in 10000's of radians
T1AR2:    .WORD   0       ;Joint angle 1, 1 sample old, radians
T1AR3:    .WORD   0       ;Joint angle 1, 2 samples old, radians
T1AR4:    .WORD   0       ;Joint angle 1, 3 samples old, radians
T1R:      .WORD   0       ;Link #1 potentiometer signal
T1ROFF:   .WORD   0       ;Initial link #1 potentiometer offset
T2AR:     .WORD   0       ;Joint angle 2 in 10000's of radians
T2AR2:    .WORD   0       ;Joint angle 2, 1 sample old, radians
T2AR3:    .WORD   0       ;Joint angle 2, 2 samples old, radians
T2AR4:    .WORD   0       ;Joint angle 2, 3 sample old, radians
T2R:      .WORD   0       ;Link #2 potentiometer signal
T2ROFF:   .WORD   0       ;Initial link #2 potentiometer offset
V1F2:     .WORD   0       ;Endpoint X1 velocity, 1 sample old
V2F2:     .WORD   0       ;Endpoint X2 velocity, 1 sample old
W1FS:     .WORD   0       ;Joint 1 angular velocity squared
```

```
W1F2:    .WORD   0       ;Joint 1 angular velocity
W2FS:    .WORD   0       ;Joint 2 angular velocity squared
W2F2:    .WORD   0       ;Joint 2 angular velocity
X1F1:    .WORD   0       ;X1 present position
X1F2:    .WORD   0       ;X1 position from 1 sample past
X1F3:    .WORD   0       ;X1 position from 2 samples past
X1F4:    .WORD   0       ;X1 position from 3 samples past
X2F1:    .WORD   0       ;X2 present position
X2F2:    .WORD   0       ;X2 position from 1 sample past
X2F3:    .WORD   0       ;X1 position from 2 samples past
X2F4:    .WORD   0       ;X1 position from 3 samples past
X1R:     .WORD   0       ;Generic reference X1 coordinate
X2R:     .WORD   0       ;Generic reference X2 coordinate

.END
```

```
C-----------------------------------------------------------------------
C
C                              TRGPNT.FOR
C                            JOHN WLASSICH
C                               2/27/85
C
C       A FORTRAN subroutine which calculates SIN and COS. Linked with
C  MACPNT.
C
C       Variables defined as follows:
C
C               KCT1    = integer cosine of theta 1
C               KCT2    = integer cosine of theta 2
C               KST1    = integer sine of theta 2
C               KST2    = integer sine of theta 2
C               KTHET1  = integer theta 1 in absolute coordinates
C               KTHET2  = integer theta 2 in absolute coordinates.
C               THETA1  = real of KTHET1
C               THETA2  = real of KTHET2
C
C       KTHET1 and KTHET2 are sent over from MACPNT as integers.
C They represent the link positions, in 100's of degrees.
C to keep 3 significant figures in integer arithmatic.
C KCT1, KCT2, KST1, KST2, are all sent back to MACPNT as
C integers, and as such, must be multiplyed by 1000 and
C to keep 3 significant figures accuracy.
C
C-----------------------------------------------------------------------
```

```
      SUBROUTINE      TRGPNT

      REAL*4 THETA1,THETA2

      COMMON/TRIG/KTHET1,KTHET2,KST1,KCT1,KST2,KCT2
      common/test/ktest1,ktest2,ktest3

C   KTHET1 and KTHET2 are calculated in MACPNT to 100 * actual degrees.
C In the next two lines KTHET1 and KTHET2 are multiplied by .01 to
C get them in degrees accurate to .01 and then multiplied by
C (PI/180) convert to radians.  These two steps are combined in the
C number .00017453.   (PI/180) is accurate to 5 significant figures
C to keep roundoff errors to a minimum.
C The result is assigned to a REAL variable.

      THETA1  = KTHET1 * 1.7453E-4
      THETA2  = KTHET2 * 1.7453E-4


C   The next four lines compute sine and cosine.  The results are
C  multiplied by 1000 to prepare them for integer arithmatic in MACPNT.

      KST1    = SIN(THETA1) * 1000
      KCT1    = COS(THETA1) * 1000

      KST2    = SIN(THETA2) * 1000
      KCT2    = COS(THETA2) * 1000


C Go back to MACPNT and continue execution there.

      RETURN

      END
```

Appendix V

```
C-----------------------------------------------------------------------
C
C                        MNPCYL.FOR
C                      JOHN WLASSICH
C                        5/20/85
C
C       A FORTRAN main program which calls MACCYL.MAC, a
C MACRO - 11 subroutine.  MACCYL uses TRGCYL, a FORTRAN subroutine
C which calculates the sine and cosine of link joint angles.
C
C       This program prompts the user to initialize the controller,
C choose parameteres that determine the circumference of a circular
C path and the speed and resolution of the endpoint, feedback
C gains, apparent endpoint inertia, and prompts to begin or restart the
C MACRO - 11 control program.  The objective of this FORTRAN program
C is be a user friendly interface to the MACRO - 11 control program.
C
C       Variable definitions follow from input queries below.
C
C-----------------------------------------------------------------------
```

```
      INTEGER*4 QUEST

      INTEGER J,DELAY,KX1R,KX2R,INDEX

      REAL RK1,RK2,RKV1,RKV2,DX1RO,DX2RO,R,X1,X2

      REAL DTHET,DT,SINDT,COSDT,X1RS,X2RS,X1RC,X2RC

      COMMON/GAINS/K1,K2,KV1,KV2
      COMMON/DESMAS/M11INV,M22INV
      COMMON/PSTN/KX1R,KX2R
      COMMON/CIRC/SINDT,COSDT,X1RC,X2RC,X1RS,X2RS
      COMMON/HOLD/DELAY

      DATA R,DTHET,DELAY/3.0,10.0,13/
      DATA K1,K2,KV1,KV2/2*0,2*5000/
      DATA M11INV,M22INV/2*1000/

      DATA DX1RO,DX2RO/2*12.0/
      DATA RK1,RK2,RKV1,RKV2/2*0.,2*5./
      DATA DM11,DM22/2*1./

3     FORMAT(/)
4     FORMAT(///)
6     FORMAT(/////////)
```

```
      WRITE(5,4)
      WRITE(5,*)'-------------------------------------------------------'
      WRITE(5,*)'  This program will ask you to provide values for '
      WRITE(5,*)' variables used to run a nonlinear force - feedback'
      WRITE(5,*)' impedence control algorithm.  There are safe'
      WRITE(5,*)' default values if you choose not to answer the '
      WRITE(5,*)' question.  If you make a mistake, there will be a'
      WRITE(5,*)' question that allows you to start over without'
      WRITE(5,*)' running the algorithm.  Answer the literal questions'
      WRITE(5,*)' with "yes" or "no".  If you do start over, values'
      WRITE(5,*)' from your last time through have been saved and need'
      WRITE(5,*)' not be re-entered.  To stop the controller once you '
      WRITE(5,*)' have started it, strike any key.  Disable the servo'
      WRITE(5,*)' amps by toggling your belt switch.  This program does'
      WRITE(5,*)' not have the software safety features of PVFMAC.'
      WRITE(5,*)' STAY CLEAR OF THE YELLOW LINE.'
      WRITE(5,*)'-------------------------------------------------------'
      WRITE(5,3)
      WRITE(5,*)' Do you wish to continue -- yes or no?'
      READ(5,205)QUEST
205   FORMAT(A4)

      IF(QUEST .EQ. 'YES')GOTO 1
            GOTO 500


1     WRITE(5,4)
      WRITE(5,*)'-------------------------------------------------------'
      WRITE(5,*)'  Your transducer signal inputs should be connected '
      WRITE(5,*)' to the I/O board in the following manner:'
      WRITE(5,3)
      WRITE(5,*)'   1) Link #1 potentiometer --- A/D channel #0'
      WRITE(5,*)'   2) Link #2 potentiometer --- A/D channel #2'
      WRITE(5,*)'   3) Force transducer x output --- A/D channel #4'
      WRITE(5,*)'   4) Force transducer y output     A/D channel #6'
      WRITE(5,3)
      WRITE(5,*)' Your command outputs from the I/O board should be'
      WRITE(5,*)' in the following manner:'
      WRITE(5,3)
      WRITE(5,*)'   1) Servo amp #1 (motor, link #1) --- DAC B'
      WRITE(5,*)'   2) Servo amp #2 (motor, link #2) --- DAC D'
      WRITE(5,*)'-------------------------------------------------------'
      WRITE(5,3)
      WRITE(5,*)' Do you wish to continue -- yes or no?'
      READ(5,200)QUEST
200   FORMAT(A4)

      IF(QUEST .EQ. 'YES')GOTO 128
            GOTO 500
```

```
128     WRITE(5,4)
        WRITE(5,*)'  To initialize the controller, put the apparatus in'
        WRITE(5,*)' the "START" position as illustrated below.'
        WRITE(5,3)
        WRITE(5,*)'                Backboard of apparratus'
        WRITE(5,*)'-----------------------------------------------------'
        WRITE(5,*)'                        motors'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                 Link #1  *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          ******************'
        WRITE(5,*)'                               link #2'
        WRITE(5,3)
        WRITE(5,*)'                      (top view)'
        WRITE(5,*)' Now stay clear of the yellow line and type in "go"'
        READ(5,122)QUEST
122     FORMAT(A4)

        IF(QUEST .EQ. 'GO')GOTO 132
                WRITE(5,6)
                WRITE(5,6)
                WRITE(5,6)
                WRITE(5,*)' ********  ILLEGAL INPUT  ********'
                WRITE(5,3)
                WRITE(5,*)' Do you want to quit -- yes or no?'
                WRITE(5,6)
                READ(5,124)QUEST
124             FORMAT(A4)

                IF(QUEST .EQ. 'YES')GOTO 500
                        GOTO 128


132     CALL FFI1


        WRITE(5,6)
        WRITE(5,*)'************* INITIALIZATION COMPLETED *************'


5       WRITE(5,6)
        WRITE(5,*)'  Do you wish to have the mechanism endpoint follow'
        WRITE(5,*)' a circular path -- yes or no?'
        WRITE(5,*)' Default is a 3 inch radius circle centered at the'
        WRITE(5,*)' START position'
        WRITE(5,*)' Default is bypassed if a nondefault option has'
        WRITE(5,*)' been chosen previously.'
        WRITE(5,3)
        READ(5,120)QUEST
120     FORMAT(A4)

        IF(QUEST .EQ. 'YES')GOTO 130
                GOTO 184
```

```
130     WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Type in your desired circle center in (X1,X2)'
        WRITE(5,*)' coordinates.  You will be notified if your choice'
        WRITE(5,*)' is out of the mechanisms bounds.'
        WRITE(5,3)
        READ(5,*)DX1RO,DX2RO

        INDEX = 1

        X1 = DX1RO
        X2 = DX2RO

        GOTO 131


180     WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Type in your radius choice, 1 to 3 inches to the'
        WRITE(5,*)' .01 inch.'
        WRITE(5,3)
        READ(5,*)R

        INDEX = 2


C  These next two lines, and the two further down just like them, check
C if the four extreems of the circle boundry ever cross out of the
C mechanism's bounds.

        X1 = DX1RO + R
        X2 = DX2RO + R

        GOTO 131


182     INDEX = 3

        X1 = DX1RO - R
        X2 = DX2RO - R                                          !


C  The next four lines test if circle perimeter is within mechanisms
C bounds.  See the documentation in MNPTRJ.FOR for an explaination
C of the same code.

131     IF(X1 .GT. 15  .OR.  X1 .LT. 7.9) GOTO 140
                IF(X2 .GT. 16  .OR.  X2 .LT. -12.1) GOTO 140

        BOUND = -3. * X1 + 24

        IF(X2 .LT. BOUND) GOTO 140
                GOTO(180,182,184),INDEX

140     WRITE(5,6)
        WRITE(5,*)' Mechanism out of bounds - try again'
        GOTO(130,180,180),INDEX
```

```
C  Set up the perimeter starting point in feet.  This value is relative
C to the center of the circle.  Send this value to SUBCYL.

184     X1RS = R/12.
        X2RS = 0.


C  Calculate the center point of the circle in feet. Send this to
C SUBCYL to be used as an offset.

        X1RC = DX1R0/12.
        X2RC = DX2R0/12.


C  Ready the starting point value for MACCYL by adding in the center
C coordinates, scaling by 1000, and assigning the values to integer
C variables.

        KX1R = (X1RC + X1RS) * 1000.
        KX2R = (X2RC + X2RS) * 1000.


        WRITE(5,6)
        WRITE(5,6)                                                  !
        WRITE(5,*)'   Do you wish to specify a path point-to-point'
        WRITE(5,*)' resolution -- yes or no?'
        WRITE(5,*)' Default is 10 degrees of arc.  Default is bypassed'
        WRITE(5,*)' if a nondefault option has been chosen previously'
        WRITE(5,3)
        READ(5,22)QUEST
22      FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 24
                GOTO 26


24      WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)' Type in your choice of path point resolution. This'
        WRITE(5,*)'is accomplished by choosing the angle between points'
        WRITE(5,*)'on the circular path.  Type in your angle choice in'
        WRITE(5,*)'whole divisions of 360 degrees.  Examples are:'
        WRITE(5,3)
        WRITE(5,*)'      angle in degrees       # points to circle'
        WRITE(5,*)'             2                      180'
        WRITE(5,*)'             5                       72'
        WRITE(5,*)'            10                       36'
        WRITE(5,*)'            20                       18'
        WRITE(5,*)'            60                        6'
        WRITE(5,*)'           120                        3'
        WRITE(5,3)
        READ(5,*)DTHET

        DT = (3.1415926 * DTHET)/180.

        SINDT = SIN(DT)
        COSDT = COS(DT)
```

```
26      WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'   Do you wish to specify a delay between points'
        WRITE(5,*)' -- yes or no?'
        WRITE(5,*)' Default is .1 second.  Default is bypassed'
        WRITE(5,*)' if a nondefault option has been chosen previously'
        WRITE(5,3)
        READ(5,28)QUEST
28      FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 32
                GOTO 198

                                                                    !
32      WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Type in your choice of delay between path points.'
        WRITE(5,*)' Choose from the table below and type in the number'
        WRITE(5,*)' of your choice.'
        WRITE(5,3)
        WRITE(5,*)'            choice #           delay(seconds)'
        WRITE(5,*)'               1                    1.0'
        WRITE(5,*)'               2                    0.5'
        WRITE(5,*)'               3                    0.1'
        WRITE(5,*)'               4                limit of program'
        WRITE(5,3)
        READ(5,*)J

        IF(J .EQ. 1)GOTO 192
                IF(J .EQ. 2)GOTO 194
                        IF(J .EQ. 3)GOTO 196
                                DELAY = 0
                                GOTO 198

192     DELAY = 135
        GOTO 198

194     DELAY = 66
        GOTO 198

196     DELAY = 13


198     WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Do you want to set the feedback gains -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)' The permissable range is :'
        WRITE(5,3)
        WRITE(5,*)'       -32 <= STIFFNESS (lb/ft) <= 32'
        WRITE(5,3)
        WRITE(5,*)'       -1.5 <= DAMPING (lb-sec/ft) <= 1.5,'
        WRITE(5,3)
        WRITE(5,*)'       where the smallest fraction allowed is .001 .'
        WRITE(5,3)
        WRITE(5,*)'  The default gives zero stiffness with a moderate'
        WRITE(5,*)' amount of damping.'
        WRITE(5,*)' Default is bypassed if a nondefault option has been'
        WRITE(5,*)' chosen previously.'
```

```
        READ(5,7)QUEST
7       FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 8                                    !
                GOTO 45

8       WRITE(5,3)
        WRITE(7,10)
10      FORMAT(' Type in X1 direction stiffness gain -- lb./ft.')
        READ(5,*)RK1

C This next line checks for an illegal gain entry.  The line following
C calibrates and prepares K1 for integer math in MACCYL.  The ".5" guards
C against decimal number truncation, it does not round up the input.

        IF(RK1 .LT. -32  .OR.  RK1 .GT. 32) GOTO 35
                K1 = IFIX((RK1 * .662 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,20)
20      FORMAT(' Type in X2 direction stiffness gain -- lb./ft.')
        READ(5,*)RK2

        IF(RK2 .LT. -32  .OR.  RK2 .GT. 32) GOTO 35
                K2 = IFIX((RK2 * .662 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,30)
30      FORMAT(' Type in X1 direction damping gain -- (lb. sec.)/ft.')
        READ(5,*)RKV1

        IF(RKV1 .LT. -1.5  .OR.  RKV1 .GT. 1.5) GOTO 35

C See MNPPNT for an explaination of this next line.

                KV1 = IFIX((RKV1 * 21.3 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,40)
40      FORMAT(' Type in X2 direction damping gain -- (lb. sec.)/ft.')
        READ(5,*)RKV2

        IF(RKV2 .LT. -1.5  .OR.  RKV2 .GT. 1.5) GOTO 35
                KV2 = IFIX((RKV2 * 21.3 *1000.) + .5)
                GOTO 45

35      WRITE(5,4)
        WRITE(5,*)' ILLEGAL GAIN ENTRY --- START OVER.'
        WRITE(5,4)
        GOTO 8
```

```
45      WRITE(5,4)
        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Do you wish to specify apparent link inertias '
        WRITE(5,*)' -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)'  The default values are 1 lb. along X1 and X2 axes.'
        WRITE(5,*)' Default is bypassed if a nondefault option has been'
        WRITE(5,*)' chosen previously.'
        READ(5,150)QUEST
150     FORMAT(A4)

        IF(QUEST .EQ. 'YES')GOTO 155
                GOTO 160

155     WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Type in your choices for apparent link inertias'
        WRITE(5,*)' in pounds (to .001 pound).  Enter your numbers'
        WRITE(5,*)' accordind to the following format:'
        WRITE(5,3)
        WRITE(5,*)'   Mass in X1 direction,Mass in X2 direction.'
        WRITE(5,3)
        WRITE(5,*)' Your values should not be larger than 22 pounds or'
        WRITE(5,*)' less than .3 pound.'
        READ(5,*)DM11,DM22

        IF(DM11 .GT. 22.1 .OR.  DM11 .LT. .29) GOTO 170
                IF(DM22 .GT. 22.1 .OR.  DM22 .LT. .29) GOTO 170

C These next two lines invert DM11 and DM22.  These values are used
C  in the MACRO program in the inverted desired mass matrix.  These
C  calculations are performed here since it is easier code in
C  FORTRAN and the calculations need be performed only once.
C  The result is rounded to 3 significant figures and made an
C  integer value to prepare it for FEIMAC.

        M11INV = IFIX(((1/DM11) * 1000) + .5)
        M22INV = IFIX(((1/DM22) * 1000) + .5)
        GOTO 160

170     WRITE(5,4)
        WRITE(5,*)' ILLEGAL MASS VALUE -- TRY AGAIN.'
        GOTO 155
```

```
160     WRITE(5,4)
        WRITE(5,4)
        WRITE(5,4)
        WRITE(5,*)'*******************************************************'
        WRITE(5,3)
        WRITE(5,*)' STAY CLEAR OF THE YELLOW LINE.'
        WRITE(5,3)
        WRITE(5,*)'  Press "1" to begin traversing the circle, after '
        WRITE(5,*)' the mechanism has settled at the starting point.'
        WRITE(5,3)
        WRITE(5,*)' Do you want to start the controller -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)'  There is a 5 second pause after you hit the '
        WRITE(5,*)' "RETURN" key before the program takes over.'
        WRITE(5,3)
        WRITE(5,*)'*******************************************************'
        READ(5,60)QUEST
60      FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 70
                GOTO 80

70      WRITE(5,6)
        WRITE(5,*)'* PRESS 0 TO STOP AND MAINTAIN A POSITION    *'
        WRITE(5,*)'*                                            *'
        WRITE(5,*)'* PRESS 1 TO TRAVERSE THE CIRCLE PERIMETER   *'
        WRITE(5,*)'*                                            *'
        WRITE(5,*)'* PRESS ANY OTHER KEY TO STOP THE CONTROLLER *'
        WRITE(5,*)'*                                            *'
        WRITE(5,*)'* TOGGLE BELT OR WALL SWITCH TO DISABLE AMPS *'
        WRITE(5,3)
        WRITE(5,62)
62      FORMAT('0',78('-'))
        WRITE(5,64)DX1R0,DX2R0
64      FORMAT('0','Your circle center is ',F6.3,',',F6.3,' inches.')
        WRITE(5,66)R
66      FORMAT('0','Your circle radius is ',F4.2,' inches.')
        WRITE(5,67)DTHET
67      FORMAT('0','Your resolution is ',F3.0,' degrees.')
        WRITE(5,63)J
63      FORMAT('0','Your delay choice is number',I2,'.')
        WRITE(5,68)
68      FORMAT('0',78('-'))

        CALL FFI2
```

```
C Print out a table of the user input values.

80      WRITE(5,6)
        WRITE(5,74)
74      FORMAT('0',78('-'))
        WRITE(5,92)DX1RO,DX2RO
92      FORMAT('0','Your circle center was ',F6.3,',',F6.3,' inches.')
        WRITE(5,94)R
94      FORMAT('0','Your circle radius was ',F4.2,' inches.')
        WRITE(5,96)DTHET
96      FORMAT('0','Your resolution was ',F3.0,' degrees.')
        WRITE(5,93)J
93      FORMAT('0','Your delay choice is number',I2,'.')
        WRITE(5,*)'The gains you selected were:'
        WRITE(5,82)
82      FORMAT('0',16X,'K1',7X,'K2',7X,'B1',7X,'B2')
        WRITE(5,84)RK1,RK2,RKV1,RKV2
84      FORMAT('0',14X,F6.3,3X,F6.3,3X,F6.3,3X,F6.3)
        WRITE(5,*)'The desired apparent endpoint mass you selected was:'
        WRITE(5,86)
86      FORMAT('0',16X,'M1',9X,'M2')
        WRITE(5,88)DM11,DM22
88      FORMAT('0',13X,F6.3,5X,F6.3)
        WRITE(5,89)
89      FORMAT('0',78('-'))


85      WRITE(7,90)
90      FORMAT(' Do you want to start over -- yes or no?')
        READ(5,100)QUEST
        WRITE(5,6)
100     FORMAT(A4)


        WRITE(5,6)
        IF(QUEST .EQ. 'YES') GOTO 5
500             WRITE(5,6)
                WRITE(5,6)
                WRITE(5,*)'****************************************'
                WRITE(5,*)'*                                      *'
                WRITE(5,*)'*  Another JJW software production.    *'
                WRITE(5,*)'*                                      *'
                WRITE(5,*)'****************************************'
                WRITE(5,6)


        STOP
        END
```

```
C-----------------------------------------------------------------------
C
C                              SUBCYL.FOR
C                               5/22/85
C                              JOHN WLASSICH
C
C      This FORTRAN subroutine generates points about a circle.  The
C circle center, radius, and point to point resolution about the
C perimeter is user specified in MNPCYL.FOR.  The speed with which the
C mechanism endpoint traverses the circle's circumference is also
C user specified in MNPCYL.
C
C      The points about the circle circumference are calculated in
C real time (versus using a look up table).  The equations used are
C time optimal and have the form below.
C
C Off line :
C
C       X1(start) = X1(center) + R * cos(0)
C                 = X1(center) + R
C
C       X2(start) = X2(center) + R * sin(0)
C                 = X2(center)
C
C where a 0 degree aszimuth was picked arbitrarily.
C
C On line :
C
C       X1(new) = X1(old) * COSDT - X2(old) * SINDT
C
C       X2(new) = X1(old) * SINDT + X2(old) * COSDT
C
C       X1(endpoint position) = X1(center offset) + X1(new)
C
C       X2(endpoint position) = X2(center offset) + X2(new)
C
C where SINDT and COSDT are the sine and cosine of the angular
C resolution, specified by the user and calculated off line in MNPCYL.
C After each time thru these equations, the "new" values become the
C "old" values in the formulas.
C
C      Note that sine and cosine are called only once, off line.  To
C generate the next point only four multiplications and four memory
C locations are needed.
C
C-----------------------------------------------------------------------
```

```
      SUBROUTINE SUBCYL

      REAL SINDT,COSDT,X1RO,X2RO,X1RN,X2RN,X1RC,X2RC

      INTEGER DELAY,KX1R,KX2R

      COMMON/CIRC/SINDT,COSDT,X1RC,X2RC,X1RO,X2RO
      COMMON/HOLD/DELAY
      COMMON/PSTN/KX1R,KX2R


      IF(WAIT .LT. DELAY)GOTO 20

C  The calculations are done with floating point arithmatic for
C accuracy.

            X1RN = X1RO * COSDT - X2RO * SINDT

            X2RN = X1RO * SINDT + X2RO * COSDT

            X1RO = X1RN

            X2RO = X2RN


C  Here, the endpoint position is readied for the MACRO -11 routine
C by adding the current perimeter ccoridinates (above) to the
C coordinates of the center of the circle (user specified). The results
C are scaled by 1000 and assigned to integer variables.

            KX1R = (X1RC + X1RN) * 1000.

            KX2R = (X2RC + X2RN) * 1000.

            WAIT = 0

            GOTO 30


20    WAIT = WAIT + 1


30    RETURN
      END
```

```
C-----------------------------------------------------------------------
C
C                        MNPTRJ.FOR
C                       JOHN WLASSICH
C                          5/20/85
C
C       A FORTRAN main program which calls MACTRJ.MAC, a
C MACRO - 11 subroutine.  MACTRJ uses TRGTRJ, a FORTRAN subroutine
C which calculates the sine and cosine of link joint angles.
C
C       This program prompts the user to initialize the controller,
C choose a straight line path for the manipulator to follow, feedback
C gains, apparent endpoint inertia, and prompts to begin or restart the
C MACRO - 11 control program.  The objective of this FORTRAN program
C is be a user friendly interface to the MACRO - 11 control program.
C
C       Variable definitions follow from input queries below.
C
C-----------------------------------------------------------------------
```

```
      INTEGER*4 QUEST
      INTEGER INDEX,DELAY,J1,J2,KX1R,KX2R
      REAL RK1,RK2,RKV1,RKV2,DX1RB,DX2RB,DX1RE,DX2RE,X1,X2
      REAL BOUND,DELX1,DELX2,X1RB,X2RB,FX1,FX2,L,T,TSUM


      COMMON/GAINS/K1,K2,KV1,KV2
      COMMON/LINE/X1RB,X2RB,L,T,FX1,FX2,TSUM
      COMMON/HOLD/DELAY
      COMMON/DESMAS/M11INV,M22INV
      COMMON/TEST/KTEST1,KTEST2,KTEST3
      COMMON/PSTN/KX1R,KX2R


      DATA X1RB,X2RB,FX1,FX2/3*1.0,0.0/
      DATA T,L,DELAY,J1,J2/8.333E-3,0.25,13,4,3/
      DATA K1,K2,KV1,KV2/2*0,2*5000/
      DATA M11INV,M22INV/2*1000/


      DATA DX1RB,DX2RB,DX1RE,DX2RE/2*12.0,15.0,12.0/
      DATA RK1,RK2,RKV1,RKV2/2*0.,2*5./
      DATA DM11,DM22/2*1./


3     FORMAT(/)
4     FORMAT(///)
6     FORMAT(/////////)
```

```
        WRITE(5,4)
        WRITE(5,*)'-------------------------------------------------'
        WRITE(5,*)'  This program will ask you to provide values for '
        WRITE(5,*)' variables used to run a nonlinear force - feedback'
        WRITE(5,*)' impedence control algorithm.  There are safe'
        WRITE(5,*)' default values if you choose not to answer the '
        WRITE(5,*)' question.  If you make a mistake, there will be a'
        WRITE(5,*)' question that allows you to start over without'
        WRITE(5,*)' running the algorithm.  Answer the literal questions'
        WRITE(5,*)' with "yes" or "no".  If you do start over, values'
        WRITE(5,*)' from your last time through have been saved and need'
        WRITE(5,*)' not be re-entered.  To stop the controller once you '
        WRITE(5,*)' have started it, strike any key.  Disable the servo'
        WRITE(5,*)' amps by toggling your belt switch.  This program does'
        WRITE(5,*)' not have the software safety features of PVFMAC.'
        WRITE(5,*)' STAY CLEAR OF THE YELLOW LINE.'
        WRITE(5,*)'-------------------------------------------------'
        WRITE(5,3)
        WRITE(5,*)' Do you wish to continue -- yes or no?'
        READ(5,205)QUEST
205     FORMAT(A4)

        IF(QUEST .EQ. 'YES')GOTO 1
                GOTO 85


1       WRITE(5,4)
        WRITE(5,*)'-------------------------------------------------'
        WRITE(5,*)'  Your transducer signal inputs should be connected '
        WRITE(5,*)' to the I/O board in the following manner:'
        WRITE(5,3)
        WRITE(5,*)'    1) Link #1 potentiometer --- A/D channel #0'
        WRITE(5,*)'    2) Link #2 potentiometer --- A/D channel #2'
        WRITE(5,*)'    3) Force transducer x output --- A/D channel #4'
        WRITE(5,*)'    4) Force transducer y output --- A/D channel #6'
        WRITE(5,3)
        WRITE(5,*)' Your command outputs from the I/O board should be'
        WRITE(5,*)' in the following manner:'
        WRITE(5,3)
        WRITE(5,*)'    1) Servo amp #1 (motor, link #1) --- DAC B'
        WRITE(5,*)'    2) Servo amp #2 (motor, link #2) --- DAC D'
        WRITE(5,*)'-------------------------------------------------'
        WRITE(5,3)
        WRITE(5,*)' Do you wish to continue -- yes or no?'
        READ(5,200)QUEST
200     FORMAT(A4)

        IF(QUEST .EQ. 'YES')GOTO 128
                GOTO 85
```

```
128     WRITE(5,4)
        WRITE(5,*)'  To initialize the controller, put the apparatus in'
        WRITE(5,*)' the "START" position as illustrated below.'
        WRITE(5,3)
        WRITE(5,*)'              Backboard of apparratus'
        WRITE(5,*)'-----------------------------------------------------'
        WRITE(5,*)'                        motors'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                 Link #1  *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          *'
        WRITE(5,*)'                          ******************'
        WRITE(5,*)'                               link #2'
        WRITE(5,3)
        WRITE(5,*)'                     (top view)'
        WRITE(5,*)' Now stay clear of the yellow line and type in "go"'
        WRITE(5,3)
        READ(5,122)QUEST
122     FORMAT(A4)

        IF(QUEST .EQ. 'GO')GOTO 132
                WRITE(5,6)
                WRITE(5,6)
                WRITE(5,6)
                WRITE(5,*)' ********  ILLEGAL INPUT  ********'
                WRITE(5,3)
                WRITE(5,*)' Do you want to quit -- yes or no?'
                WRITE(5,6)
                READ(5,124)QUEST
124             FORMAT(A4)

                IF(QUEST .EQ. 'YES')GOTO 500
                        GOTO 128


132     CALL FFI1


        WRITE(5,6)
        WRITE(5,*)'************** INITIALIZATION COMPLETED **************'
```

```
C  Reset parameter used in SUBTRJ.  See documentation in SUBTRJ for
C further detail.

5       TSUM = 0
        WRITE(5,6)
        WRITE(5,*)'  Do you wish to have the mechanism endpoint'
        WRITE(5,*)' traverse a straight line -- yes or no?'
        WRITE(5,*)' Default is a 3 inch movement along the X1 axis'
        WRITE(5,*)' begining at the START position.'
        WRITE(5,*)' Default is bypassed if a nondefault option has'
        WRITE(5,*)' has been chosen previously.'
        WRITE(5,3)
        READ(5,120)QUEST
120     FORMAT(A4)

        IF(QUEST .EQ. 'YES')GOTO 133
                GOTO 26

133     WRITE(5,6)
130     WRITE(5,6)
        WRITE(5,*)'  Type in your desired starting point in (X1,X2)'
        WRITE(5,*)' coordinates in inches (to .01 in.).  You will be'
        WRITE(5,*)' notified if your choice is out of the mechanisms'
        WRITE(5,*)' bounds.'
        WRITE(5,*)'  Suggestion; make your first run starting point the'
        WRITE(5,*)' START position to avoid any unexpected movements.'
        WRITE(5,4)
138     READ(5,*)DX1RB,DX2RB

        INDEX = 1

C  Let points specified by user be checked for legal mechanism bounds.
C INDEX is used to return execution to the correct location via a
C computed GOTO.  A subroutine or function calls could have been used,
C however this is a viable alternative that I wanted to try.

        X1 = DX1RB
        X2 = DX2RB

        GOTO 131

143     WRITE(5,6)
142     WRITE(5,6)
        WRITE(5,*)'  Type in your desired ending point in (X1,X2)'
        WRITE(5,*)' coordinates in inches (to .01 in.).  You will be'
        WRITE(5,*)' notified if your choice is out of the mechanisms'
        WRITE(5,*)' bounds.'
        WRITE(5,*)'  Suggestion; make your first run ending point 2 '
        WRITE(5,*)' or 3 inches to avoid any unexpected movements.'
        WRITE(5,4)
        READ(5,*)DX1RE,DX2RE

        INDEX = 2

C  See comments above on the computed GOTO

        X1 = DX1RE
        X2 = DX2RE
```

```
C  These next five lines check that the reference position specified by
C the user is within the mechanism's bounds.  The limits are a rectangle
C (viewing the mechanism from above) with a cut-off rear left hand
C corner.  The corner is clipped so that the 4 bar does not collide
C with the backboard of the mechanism for some endpoint positions.
C BOUND defines the line that cuts the corner off the rectangle.

131     IF(X1 .GT. 18  .OR.  X1 .LT. 7.9) GOTO 140
              IF(X2 .GT. 16  .OR. X2 .LT. -12.1) GOTO 140

        BOUND = -3. * X1 + 24
        IF(X2 .LT. BOUND) GOTO 140
              GOTO (143,180),INDEX

140     WRITE(5,6)
        WRITE(5,*)' ILLEGAL POSTION ENTRY ---  TRY AGAIN.'
        GOTO (130,142),INDEX


C These next two lines convert the desired start position from
C inches to feet.

180     X1RB = DX1RB/12.0
        X2RB = DX2RB/12.0


C  Compute X1(end) - X1(start) and X2(end) - X2(start) for use in
C SUBTRJ, see this subroutine for further explaination.

        DELX1 = DX1RE - DX1RB
        DELX2 = DX2RE - DX2RB


C  Compute the length of the line for use in SUBTRJ.

        L = SQRT(DELX1**2 + DELX2**2)


C These "factors" are used by SUBTRJ.

        FX1 = DELX1/L
        FX2 = DELX2/L


C  Change "L" TO units of feet from inches for use in SUBTRJ.  Used to
C decide when ending point of line has been reached.

        L = L/12.


C  These next two lines ready the starting position for used by MACTRJ
C (integer math).  The starting position is wanted so that the mechanism
C will go to it and hold until the user strikes the "1" key.

26      KX1R = X1RB * 1000.
        KX2R = X2RB * 1000.

        WRITE(5,6)
```

```
        WRITE(5,6)
        WRITE(5,*)'  Do you wish to specify a point to point resolution'
        WRITE(5,*)' and a time delay between points -- yes or no'
        WRITE(5,*)' Default gives a .1 inch resolution and .1 second '
        WRITE(5,*)' delay.  Default is bypassed if a nondefault option'
        WRITE(5,*)' has been chosen previously.'
        WRITE(5,3)
        READ(5,22)QUEST
22      FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 24
                GOTO 190

24      WRITE(5,6)
        WRITE(5,*)'  Pick a combination of point-path resolution and'
        WRITE(5,*)' delay time, between points, from the table below.'
        WRITE(5,*)' ------------------------------------------------'
        WRITE(5,*)'   RESOLUTION (INCHES)     DELAY (SECONDS)'
        WRITE(5,3)
        WRITE(5,*)'   1)  0.4                 1)  1.0'
        WRITE(5,*)'   2)  0.3                 2)  0.5'
        WRITE(5,*)'   3)  0.2                 3)  0.1'
        WRITE(5,*)'   4)  0.1                 4)  Limit of controller'
        WRITE(5,*)'------------------------------------------------'
        WRITE(5,*)'  Type in your choices by using the numbers '
        WRITE(5,*)' preceding your resolution and delay.'
        WRITE(5,3)
        READ(5,*)J1,J2

C  Follow the decision tree below to find the resolution "J1" in feet,
C and the delay "J2" in number of MACTRJ control loop sampling periods.
C A sampling period is about 7.6 milliseconds.

        IF(J1 .EQ. 1) GOTO 182
                IF(J1 .EQ. 2) GOTO 184
                        IF(J1 .EQ. 3) GOTO 186
                                T = 8.333E-3
                                GOTO 188

182     T = 33.3E-3
        GOTO 188

184     T = 25.0E-3
        GOTO 188

186     T = 16.7E-3

188     IF(J2 .EQ. 1) GOTO 192
                IF(J2 .EQ. 2) GOTO 194
                        IF(J2 .EQ. 3) GOTO 196
                                DELAY = 0
                                GOTO 190

192     DELAY = 131
        GOTO 190
194     DELAY = 66
        GOTO 190
196     DELAY = 13
```

```
190     WRITE(5,4)
        WRITE(5,4)
        WRITE(5,*)'  Do you want to set the feedback gains -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)' The permissable range is :'
        WRITE(5,3)
        WRITE(5,*)'       -32 <= STIFFNESS (lb/ft) <= 32 '
        WRITE(5,3)
        WRITE(5,*)'       -1.5 <= DAMPING (lb-sec/ft) <= 1.5,'
        WRITE(5,3)
        WRITE(5,*)'       where the smallest fraction allowed is .001 .'
        WRITE(5,3)
        WRITE(5,*)'  The default gives zero stiffness with a moderate'
        WRITE(5,*)' amount of damping.'
        WRITE(5,*)' Default is bypassed if a nondefault option has been'
        WRITE(5,*)' chosen previously.'
        READ(5,7)QUEST
7       FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 8
                GOTO 45

8       WRITE(5,3)
        WRITE(7,10)
10      FORMAT(' Type in X1 direction stiffness gain -- lb./ft.')
        READ(5,*)RK1

C This next line checks for an illegal gain entry.  The line following
C calibrates and prepares K1 for integer math in MACTRJ.  The ".5" guards
C against decimal number truncation, it does not round up the input.

        IF(RK1 .LT. -32  .OR.  RK1 .GT. 32) GOTO 35
                K1 = IFIX((RK1 * .662 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,20)
20      FORMAT(' Type in X2 direction stiffness gain -- lb./ft.')
        READ(5,*)RK2

        IF(RK2 .LT. -32  .OR.  RK2 .GT. 32) GOTO 35
                K2 = IFIX((RK2 * .662 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,30)
30      FORMAT(' Type in X1 direction damping gain -- (lb. sec.)/ft.')
        READ(5,*)RKV1

        IF(RKV1 .LT. -1.5  .OR.  RKV1 .GT. 1.5) GOTO 35
                KV1 = IFIX((RKV1 * 21.3 * 1000.) + .5)

        WRITE(5,3)
        WRITE(7,40)
40      FORMAT(' Type in X2 direction damping gain -- (lb. sec.)/ft.')
        READ(5,*)RKV2

        IF(RKV2 .LT. -1.5  .OR.  RKV2 .GT. 1.5) GOTO 35
                KV2 = IFIX((RKV2 * 21.3 * 1000.) + .5)
                GOTO 45
```

```
35      WRITE(5,4)
        WRITE(5,*)' ILLEGAL GAIN ENTRY --- START OVER.'
        WRITE(5,4)
        GOTO 8


45      WRITE(5,4)
        WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Do you wish to specify apparent link inertias '
        WRITE(5,*)' -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)'  The default values are 1 lb. along X1 and X2 axes.'
        WRITE(5,*)' Default is bypassed if a nondefault option has been'
        WRITE(5,*)' chosen previously.'
        READ(5,150)QUEST
150     FORMAT(A4)

        IF(QUEST .EQ. 'YES')GOTO 155
                GOTO 160

155     WRITE(5,6)
        WRITE(5,6)
        WRITE(5,*)'  Type in your choices for apparent link inertias'
        WRITE(5,*)' in pounds (to .001 pound).  Enter your numbers'
        WRITE(5,*)' accordind to the following format:'
        WRITE(5,3)
        WRITE(5,*)'   Mass in X1 direction,Mass in X2 direction.'
        WRITE(5,3)
        WRITE(5,*)' Your values should not be larger than 22 pounds or'
        WRITE(5,*)' less than .3 pound.'
        READ(5,*)DM11,DM22

        IF(DM11 .GT. 22.1 .OR.  DM11 .LT. .29) GOTO 170
                IF(DM22 .GT. 22.1 .OR.  DM22 .LT. .29) GOTO 170

C These next two lines invert DM11 and DM22.  These values are used
C  in the MACRO program in the inverted desired mass matrix.  These
C  calculations are performed here since it is easier code in
C  FORTRAN and the calculations need be performed only once.
C  The result is rounded to 3 significant figures and made an
C  integer value to prepare it for FFIMAC.

        M11INV = IFIX(((1/DM11) * 1000) + .5)
        M22INV = IFIX(((1/DM22) * 1000) + .5)
        GOTO 160

170     WRITE(5,4)
        WRITE(5,*)' ILLEGAL MASS VALUE -- TRY AGAIN.'
        GOTO 155
```

```
160     WRITE(5,4)
        WRITE(5,4)
        WRITE(5,4)
        WRITE(5,*)'*****************************************************'
        WRITE(5,3)
        WRITE(5,*)' STAY CLEAR OF THE YELLOW LINE.'
        WRITE(5,3)
        WRITE(5,*)'  There is a 5 second pause after you hit the '
        WRITE(5,*)' "RETURN" key before the program takes over.'
        WRITE(5,3)
        WRITE(5,*)'  Press the "1" key, once the mechanism has settled'
        WRITE(5,*)' at the start of the specified line, to begin '
        WRITE(5,*)' traversing the line.'
        WRITE(5,3)
        WRITE(5,*)' Do you want to start the controller -- yes or no?'
        WRITE(5,3)
        WRITE(5,*)'*****************************************************'
        READ(5,60)QUEST                                          !
60      FORMAT(A4)

        IF(QUEST .EQ. 'YES') GOTO 70
                GOTO 80

70      WRITE(5,6)
        WRITE(5,*)'*  PRESS "0" TO STOP AND MAINTAIN A POSITION    *'
        WRITE(5,*)'*                                               *'
        WRITE(5,*)'*  PRESS "1" TO TRAVERSE THE SPECIFIED LINE     *'
        WRITE(5,*)'*                                               *'
        WRITE(5,*)'*  PRESS ANY OTHER KEY TO STOP THE CONTROLLER   *'
        WRITE(5,*)'*                                               *'
        WRITE(5,*)'*  TOGGLE BELT OR WALL SWITCH TO DISABLE AMPS   *'
        WRITE(5,4)
        WRITE(5,62)
62      FORMAT('0',78('-'))
        WRITE(5,*)'The starting and ending positions you selected were:'
        WRITE(5,66)DX1RB,DX2RB,DX1RE,DX2RE
66      FORMAT('0',8X,F6.2,',',F6.2,14X,F6.2,',',F6.2)
        WRITE(5,64)J1,J2
64      FORMAT('0','The resolution and delay choices you made were
     *numbers',I2,' and',I2,'.')
        WRITE(5,68)
68      FORMAT('0',78('-'))

        CALL FFI2
```

```
C Print out a table of the user input values.

80      WRITE(5,6)
        WRITE(5,74)
74      FORMAT('0',78('-'))
        WRITE(5,*)'The starting and ending positions you selected were:'
        WRITE(5,92)DX1RB,DX2RB,DX1RE,DX2RE                    !
92      FORMAT('0',8X,F6.2,',',F6.2,14X,F6.2,',',F6.2)
        WRITE(5,94)J1,J2
        WRITE(5,3)
94      FORMAT('0','The resolution and delay choices you made were
     *numbers',I2,' and',I2,'.')
        WRITE(5,*)'The gains you selected were:'
        WRITE(5,82)
82      FORMAT('0',16X,'K1',7X,'K2',7X,'B1',7X,'B2')
        WRITE(5,84)RK1,RK2,RKV1,RKV2
84      FORMAT('0',14X,F6.3,3X,F6.3,3X,F6.3,3X,F6.3)
        WRITE(5,*)'The desired apparent endpoint mass you selected was:'
        WRITE(5,86)
86      FORMAT('0',16X,'M1',9X,'M2')
        WRITE(5,88)DM11,DM22
88      FORMAT('0',13X,F6.3,5X,F6.3)
        WRITE(5,89)
89      FORMAT('0',78('-'))


85      WRITE(5,3)
        WRITE(7,90)
90      FORMAT(' Do you want to start over -- yes or no?')
        READ(5,100)QUEST
        WRITE(5,6)
100     FORMAT(A4)


        WRITE(5,6)
        IF(QUEST .EQ. 'YES') GOTO 5
                WRITE(5,6)
                WRITE(5,6)
500             WRITE(5,*)'****************************************'
                WRITE(5,*)'*                                      *'
                WRITE(5,*)'*  Another JJW software production.    *'
                WRITE(5,*)'*                                      *'
                WRITE(5,*)'****************************************'
                WRITE(5,6)


        STOP
        END
```

```
C-----------------------------------------------------------------------
C
C                                SUBTRJ.FOR
C                                 5/22/85
C                                JOHN WLASSICH
C
C
C       This FORTRAN subroutine calculates the next point in a series
C  of points that define a straight line for the endpoint of the
C  mechanism.  User defined parameters come from MNPTRJ and SUBTRJ is
C  called by MACTRJ.
C
C       The points defining a straight line are computed by a parametric
C  equation of a straight line.  The form is :
C
C   X1(new) = X1(start) + ((X1(end) - X1(start))/(length of line))*TSUM
C
C   X2(new) = X2(start) + ((X2(end) - X2(start))/(length of line))*TSUM
C
C  where "TSUM" is the paremter being incremented (resolution is user
C  defined, "T") until "TSUM" equals the length of the line.
C
C-----------------------------------------------------------------------
```

```
      SUBROUTINE SUBTRJ

      INTEGER WAIT,DELAY,KX1R,KX2R

      REAL X1RB,X2RB,L,T,FX1,FX2,TSUM

      COMMON/LINE/X1RB,X2RB,L,T,FX1,FX2,TSUM
      COMMON/HOLD/DELAY
      COMMON/PSTN/KX1R,KX2R

      IF(TSUM .GT. L) GOTO 30
              IF(WAIT .LT. DELAY) GOTO 20
                      KX1R = (X1RB + (FX1 * TSUM)) * 1000.
                      KX2R = (X2RB + (FX2 * TSUM)) * 1000.
                      TSUM = TSUM + T
                      WAIT = 0
                      GOTO 30


20    WAIT = WAIT + 1


30    RETURN
      END
```

References

1. Abul-Haj C.
   An Emulation Technique for Developing Improved Elbow-Prosthesis Designs.
   Document in preparation, 1986.

2. Andrews R., Hogan N.
   Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator.
   Control of Manufacturing Processes and Robotic Systems, pp.243-251, ASME, New York, 1983.

3. Arimoto S., Takegaki M.
   An Adapative Method for Trajectory Control of Manipulators.
   Proceedings Eight Triennial IFAC, 1982.

4. Asada H., Youcef-Toumi K.
   Analysis and Design of a Direct-Drive Arm With a Five-Bar-Link Mechanism.
   Journal of Dynamic Systems, Measurement, and Control, pp. 225-230, ASME, September 1984.

5. Astrom K., Wittenmark B.
   Computer Controlled Systems.
   Prentice-Hall, Englewood Cliffs, NJ., 1984.

6. Blevins R.
   Formulas for Natural Frequency and Mode Shape.
   Van Nostrand Reinhold, New York, NY., 1979.

7. Cotter S.
   Nonlinear Feedback Control of Manipulator Endpoint Impedance.
   SMME Thesis, MIT, Mechanical Engineering Department, July 1982.

8. Crandall S., Karnop D., Kurtz E., Pridmore-Brown D.
Dynamics of Mechanical and Electromechanical Systems.
Krieger, Malabar, FL., 1982.

9. D'Azzo J., Houpis C.
Linear control Systems Aanlysis and Design.
McGraw-Hill, New York, NY., 1981.

10. Dean D.
Design of a Robotic End-Effector for Automatic Bolting.
SMME Thesis, MIT, Mechanical Engineering Department, May 1985.

11. Drake S.
Using Compliance in Lieu of Sensory Feedback for Automatic Assembly.
Charles Stark Draper Laboratory Report T-657, September 1977.

12. Dubowsky S., DesForges D.
The Application of Model-Referenced Adaptive Control to Robotic Manipulators.
Journal of Dynamic Systems, Measurement, and Control, pp. 193-200, ASME, September 1979.

13. EOA Systems Inc.
Electronic Feedback Nutrunner.
EOA, Dallas , TX. 1985.

14. Faires V.
Design of Machine Elements.
Macmillan, New York, NY., 1965.

15. Gelb A., Vandervelde W.
Multiple Input-Describing Functions and Nonlinear Systems Design.
McGraw-Hill, New York, NY., 1968.

16. Harrison H., Bollinger J.
Introduction to Automatic Controls.
Harper and Row, New York, NY., 1969.

17. Hibbler R.
Engineering Mechanics: Dynamics.
Macmillan, New York, NY., 1978.

18. Hirzinger G.
Direct Digital Robot Control Using a Force-Torque Sensor.
IFAC Symposium on Real Time Digital Control Applications, 1983.

19. Hollerbach J.
A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity.
IEEE Transactions on Systems, Man, and Cybernetics, November 1980.

20. Hogan N.
Control Strategies for Complex Movements Derived from Physical Systems Theory.
International Symposium on Synergetics, Bavaria, May 1985.

21. Hogan N.
Impedance Control: An Approach to Manipulation: Part I - Theory, Part II - Implementation, Part III - Applications.
Journal of Dynamic Systems, Measurement, and Control, pp. 1-24, ASME, March 1985.

22. Hogan N.
Robotics: An Assembly Machine for Putting Bolts in Threaded Holes.
A Continuation Proposal to Daewoo Heavy Industries, Ltd., June 1983.

23. Horowitz P., Hill W.
The Art of Electronics.
Cambridge University Press, Cambridge, England, 1980.

24. Ishida T.
Force Control in Coordination of Manipulator Fine Motions.
Fifth International Conference on Artificial Intelligence, 1977.

25. Kazerooni H.
A Robust Design Method for Impedance Control of Constrained Dynamic Systems.
Doctor of Science Thesis, MIT, Mechanical Engineering Department, February 1985.

26. Khaboza J.
Numerical Analysis.
Maxwell, Los Angeles, CA., 1965.

27. Kleidon M.
Modelling and Performance of a Pneumatic/Hydralic Hybrid Actuator with Tunable Mechanical Impedance.
SM Thesis, MIT, Mechanical Engineering Department, September 1983.

28. Kreyszig E.
Advanced Engineering Mathematics.
Wiley, New York, NY., 1979.

29. Kwakernaak H., Sivan R.
Linear Optimal Control Systems.
Wiley-Interscience, New York, NY. 1972.

30. Luh J., Walker M., Paul R.
On-line Computation Scheme for Mechanical Manipulators.
Journal of Dynamic Systems, Measurement , and Control, June 1980.

31. Luh J., Walker M., Paul R.
Resolved-Acceleration Control of Mechanical Manipulators.
IEEE Transactions on Automatic Control, June 1980.

32. Makino H., Furuya N., Soma K., Chin E.
Research and Development of the SCARA Robot.
Proceedings of the Fourth Annual International Conference on Productio Engineering, Tokyo, 1980.

33. Mason M.
Compliance and Force Control for Computer Controlled Manipulators.
IEEE Transactions, Systems, Man and Cybernetics, pp. 418-432, 1981.

34. McCormick J., Salvadori M.
Numerical Methods in FORTRAN.
Prentice-Hall, Englewood Cliffs, NJ., 1964.

35. Micro Networks.
New Product Review and *MN5420 Preliminary.*
Unitrode Corporation, Worcester, MA, 1985.

36. Nevins J., Whitney D.
The Force Vector Assembler Concept.
Proceedings, First CISM-IFTOMM Symposium on Theory and Practice o
Robots and Manipulators, Udine, Italy September 1973.

37. Oberg E., Jones F., Horton H.
Machinery's Handbook.
Industrial Press, New York, NY., 1984.

38. Ogata K.
Modern Control Engineering.
Prentice-Hall, Englewood Cliffs, NJ., 1970.

39. PMI Motors.
Switching Servo Amplifier.
Kollmorgen Corp., Syosset, NY., 1981.

40. Paul R.
Robot Manipulators: Mathematics, Programming, and Control.
M.I.T. Press, Cambridge, MA., 1981.

41. Paynter H.
Analysis and Design of Engineering Systems.
M.I.T. Press, Cambridge, MA., 1961.

42. Raibert M., Craig J.
Hybrid Poosition/Force Control of Manipulators.
ASME Transactions, Journal of Dynamic Systems, Measurement , an
Control, pp. 126-133, 1981.

43. Raibert M., Horn B.
Manipulator Control Using the Configuration Space Method.
The Industrial Robot, pp. 69-73, June 1978.

44. Roberts R., Paul R., Hillberry B.
The Effect of Wrist Force Sensor Stiffness on the Control of Robot Manipulators.
IEEE Proceedings, Conference on Robotics and Automation, 1985.

45. Rosenberg R., Karnopp D.
Introduction to Physical System Dynamics.
McGraw-Hill, New York, NY., 1983.

46. Salisbury J.
Active Stiffness Control of a Manipulator in Cartesian Coordinates.
IEEE Proceedings, Conference on Decision and Control, 1980.

47. Semiconductor Group of Texas Instruments
The TTL Data Book for Design Engineers.
Texas Instruments Inc., Dallas, TX., 1981.

48. Shigley J.
Kinematic Analysis of Mechanisms.
McGraw-Hill, New York, NY., 1969.

49. Shimano B.
Force Control
Stanford University Artificial Intelligence Laboratory Memo 285.4, June 1977.

50. Stepien T., Sweet M., Good M.
Control of Tool/Workpiece Contact Force with Application to Robotic Deburring.
IEEE Proceedings, Conference on Robotics and Automation, 1985.

51. van Brussel H., Simons J.
Robot Assembly by Force Feedback Accommodation.
Annual CIRP, Great Britain, Vol. 28 No. 1, 1979.

52. Whitney D.
Force Feedback Control of Manipulator Fine Motions.
ASME Transactions, Journal of Dynamic Systems, Measurement , and Control, pp. 91-97, 1976.

53. Whitney D.
Historical Perspective and State of the Art in Robot Force Control.
IEEE Conference, Robotics and Automation, 1985.

54. Whitney D., Nevins J.
What is Remote Center Compliance and What Can It Do?
Ninth ISIR Proceedings, Washington DC., March 1979.

55. Wu C., Paul R.
Manipulator Compliance Based on Joint Torque Control.
IEEE Proceedings, Conference on Decision and Control, pp. 205-270, December 1981.

56. Wu C., Paul R.
Resolved Motion Force Control of Robot Manipulators.
IEEE Transactions on Systems, Man, and Cybernetics, June 1982.