NASA-CR-191339

SBIR-05.01-0300A
Release Date 8/10/93

National Aeronautics and Space Administration P. 90
Small Business Innovation Research Program

Phase II Final Report

"Control Algorithm Implementation For A Redundant Degree Of Freedom Manipulator"

Prepared For:

Jet Propulsion Laboratory
ATTN: Dr. Neville Marzwell, M/S 198-219
4800 Oak Grove Drive
Pasadena, CA 91109

Prepared By:

Odetics, Inc.
1515 S. Manchester Ave.
Anaheim, CA 92802

Contract No. NAS7-1062

October 13, 1991

(NASA-CR-191339) CONTROL ALGORITHM IMPLEMENTATION FOR A REDUNDANT DEGREE OF FREEDOM MANIPULATOR Final Report, 13 Jul. 1989 - 13 Oct. 1991 (Odetics) 90 p

N94-13131

Unclas

G3/63 0186512

Project Summary

This project's purpose is to develop and implement control algorithms for a kinematically redundant robotic manipulator. The manipulator is being developed concurrently by Odetics Inc., under internal research and development funding. This SBIR contract supports algorithm conception, development, and simulation, as well as software implementation and integration with the manipulator hardware.

The Odetics Dexterous Manipulator is a lightweight, high strength, modular manipulator being developed for space and commercial applications. It has seven fully active degrees of freedom, is electrically powered, and is fully operational in 1 G. The manipulator consists of five self-contained modules. These modules join via simple quick-disconnect couplings and self-mating connectors which allow rapid assembly/disassembly for reconfiguration, transport, or servicing. Each joint incorporates a unique drivetrain design which provides zero backlash operation, is insensitive to wear, and is single fault tolerant to motor or servo amplifier failure. The sensing system is also designed to be single fault tolerant. Although the initial prototype is not space qualified, the design is well-suited to meeting space qualification requirements.

The control algorithm design approach is to develop a hierarchical system with well defined access and interfaces at each level. The high level endpoint/configuration control algorithm transforms manipulator endpoint position/orientation commands to joint angle commands, providing task space motion. At the same time, the kinematic redundancy is resolved by controlling the configuration (pose) of the manipulator, using several different optimizing criteria. The center level of the hierarchy servos the joints to their commanded trajectories using both linear feedback and model-based nonlinear control techniques. The lowest control level uses sensed joint torque to close torque servo loops, with the goal of improving the manipulator dynamic behavior. The control algorithms are subjected to a dynamic simulation before implementation.

The manipulator control hardware is a VME bus-based multiprocessor computing system. Software, which is entirely written in the C language, is developed under UNIX on a workstation host computer and executed on the embedded controller using a real time operating system.

The report discusses the control system implementation, system integration, and performance evaluation in detail.

Potential applications exist in both the space and terrestrial domains. Many of the system's sizing and fault tolerance characteristics are chosen to be consistent with space applications such as satellite servicing, refueling, and space assembly. Terrestrial applications may include handling of hazardous materials in unstructured environments. In addition, the system's modularity encourages the development of simpler reduced degree of freedom mechanisms for specific applications.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 Introduction

1.1 Background

For years, researchers in robotics have focused a large effort on the study of manipulators with redundant degrees of freedom. Spatial manipulators with more than six degrees of freedom promise improved performance over their more conventional counterparts because the additional freedom of movement can be exploited in ways beyond positioning a tool, following a path, or applying a force. Due to their superior mobility, these machines are referred to as *dexterous* manipulators. Many variations on a few basic approaches have yielded numerous control algorithms for dexterous manipulators. Joint limit avoidance, singularity avoidance, joint torque optimization, and configuration control have been extensively studied as means to utilize redundancy. Much of this work has been theoretical in nature, utilizing simulation to demonstrate results. More recently, dexterous manipulators have appeared in laboratories, providing testbeds for the theoretical work and the only truly valid means for evaluating the performance of new control methods. This experimental work will eventually reveal the best approaches, which will be adopted by manipulator manufacturers.

Currently, very few dexterous manipulators are available commercially. While manufacturers of current industrial manipulators may have R&D programs to study advanced concepts, very few have been motivated to bring such machines into production. Current "factory robotics" applications do not require (or are perceived not to require) the capabilities of dexterous manipulators. Therefore, the current conventional applications market does not justify a large R&D expenditure. It is the newer, more demanding applications in the space, defense, and nuclear industries that will drive the development of high performance dexterous manipulators. One viewpoint is that new technology which precedes a specific application or market can help create that market by showing end users a new set of capabilities that they can apply to problem solving. A manufacturer that can create such a market with its products will enjoy a significant base technology lead over the competition and will be positioned to address new applications quickly.

For the past eight years, Odetics Inc. has been developing robotic and sensor systems and technology for applications in the space, defense, nuclear, and commercial markets. Delivered systems and systems in development include:

- several six-legged walking machines, spread over three development generations
- several sophisticated laser imaging systems
- a location identification system for outdoor navigation, and an autonomous observation/reconnaissance system for military applications
- systems for navigation within cluttered environments and path planning for autonomous manipulators
- a high strength-to-weight ratio (1:1) electric manipulator

• a 155 millimeter howitzer autoloader.

In addition, the company has conducted much government and internally sponsored robotics research, particularly in manipulator dynamics and control. Given these capabilities and the potential market for dexterous manipulators, Odetics has chosen to develop an advanced dexterous manipulator with IR&D funding. This Phase II SBIR develops and implements the control system and software for the manipulator system.

1.2 Motivation

One promising application for the Odetics Dexterous Manipulator is in space telerobotics. Significant work that defines and specifies telerobotic operations in space is ongoing in both government and commercial research organizations. Space assembly and servicing is one active area. While much of the research in space telerobotics concentrates on activities such as assembly and satellite servicing, a capable space telerobot will be useful for a wider variety of tasks, serving as a general purpose space mechanism. Docking and berthing activities could employ such a mechanism. Principal motivators for space telerobots include increased safety and reduced costs through EVA reduction. Important requirements for space telerobots include dexterity, fault tolerant/redundant systems, low weight (high strength to weight ratio), and a design that can reasonably evolve from a 1 G prototype to a space qualified version.

An important near term terrestrial application area is environmental restoration and waste management. Robotics applied to these tasks can make the tasks safer via reduced worker exposure, as well as faster and cheaper through increased productivity and reduced life cycle costs. While special purpose hard automation or simple modifications of existing technology will be appropriate for many tasks, there will be particularly demanding tasks requiring dexterity, strength, and adaptability. An example is waste storage tank remediation, which requires a way to maneuvei and position various tools for sampling, mapping, and retrieving waste in constrained and hazardous surroundings.

1.3 Phase I Work

The objective during Phase I of this program was to develop an endpoint control algorithm for a seven degree of freedom manipulator, determine its performance through simulation, and verify that its computational requirements were within the bounds of embedded processor capability. The Phase I control algorithm addresses kinematics only, that is, it transforms endpoint commands into joint commands. System dynamics and servoing are not included. The algorithm resolves the redundancy via a modified pseudoinverse technique that smoothly adds and removes a singularity avoidance term, as required. As a pseudoinverse technique, it provides a differential, or "rate" solution rather than a "position" solution. Simulation results show that the algorithm's singularity avoidance feature improve manipulator performance: with the singularity avoidance active, the manipulator executes the commanded endpoint trajectories while avoiding singularities, resulting in lower joint velocities and more accurate motion. Sensitivity of the algorithm performance to various numerical parameters is discussed. The simulation model is a precursor to the Odetics Dexterous Manipulator. A computer graphics simulation depicts manipulator motion.

1.4 Phase II Proposal

The proper follow-on to the Phase I simulation is a hardware implementation. Three principal tasks were identified as required to achieve this goal: an enhancement of the Phase I algorithm to include configuration control, design and simulation of servo control algorithms to include system dynamics, and integration of the algorithms with an actual manipulator and its embedded processor hardware/software environment. While a Phase II SBIR contract is large enough to complete these tasks, it does not come close to supporting a simultaneous manipulator design, fabrication, and delivery. Some other source of funds would be required to obtain a manipulator for the control system implementation.

Odetics started an internal research and development effort on dexterous manipulators during 1988. While this effort included conceptual design of the manipulator itself, funding limitations precluded any significant control system design. Fortunately, timing worked out advantageously. Odetics submitted a Phase II proposal to NASA for a dexterous manipulator control system implementation. Odetics would fund the design and fabrication of the manipulator, while the contract would support control system design, simulation, and implementation. While no hardware would be delivered, Odetics would provide a hardware demonstration. This approach would enable Odetics to develop a complete dexterous manipulator system, despite limited IR&D funding. In turn, NASA would benefit by receiving more than a "paper study" from the contract. It would have a significant interest in a system that addresses its need for space telerobot hardware, as well as specific rights to control system technology developed during the program.

1.5 Contract Deliverables

The Phase II contract deliverables include:

- this final report, which describes in detail the project objectives, worked carried out, results obtained, and recommendations for future work.
- a software tape that contains the control system source code.

2.0 The Odetics Dexterous Manipulator

This section of the report describes the Odetics Dexterous Manipulator. The manipulator is being developed by Odetics with internal funding; its development is not part of this SBIR contract. However, the concurrent IRAD and SBIR funding enables Odetics to develop a complete manipulator and control system at tolerable cost, providing Odetics with a new product and NASA with both a space manipulator alternative (very few are available) and specific rights to the manipulator control technology.

2.1 Objectives

Odetics is developing this manipulator in order to address new space, defense, and environmental markets in which current manipulator technology is inadequate. Although these applications are embryonic and do not translate into well-defined specifications, current manipulators clearly lack the general performance capabilities these tasks will require. The general approach guiding this design is to build an advanced manipulator which uses the best ideas from existing designs and has new features that meet the general requirements for advanced applications in both the space and commercial arenas.

Another important design objective was to create a system that could operate both terrestrially and in a microgravity environment. Previous space manipulators were not operational in 1 G and required special equipment for ground testing. Within the financial scope of this effort, the immediate objective was to develop a system that is a reasonable design evolution away from becoming a space-qualified machine.

2.2 Design

Applications such as satellite servicing and environmental remediation will require autonomous and teleoperated manipulation in unstructured, dynamic environments. The capabilities of the manipulator system will ultimately determine the success or failure of these operations. As with most system developments, cost and development time requirements must balance performance and reliability goals. Since definitions of the tasks to be performed are still evolving, a reconfigurable system that could be easily adapted to various applications would be attractive.

These considerations led to the adoption of a modular manipulator architecture. A set of self-contained manipulator modules with standard interfaces provides lower cost and minimizes development time of specialized systems. In addition, modularity allows easy transportation to a remote location, fast on-site assembly, and quick repairs in-the-field. Useful configurations are not limited to manipulators. Self-contained actuator modules can be configured into various reduced degree of freedom mechanisms for highly structured tasks, or hyper-redundant mechanisms with motion capability beyond that of manipulators.

Some specific mechanical design challenges arising from the modular architecture approach include:

- Mechanical and electrical module interface design
- Component packaging and wire harness design
- Scalable actuator topologies.

More general mechanical design and engineering goals include:

- Maximum payload to weight ratio and compact design
- High dexterity
- Fault tolerant sensing and actuation
- Fully enclosed mechanisms and wiring
- Accurate joint torque sensing.

Design issues specific to the control of a high performance kinematically redundant manipulator include:

- Providing sensing for advanced control techniques
- Redundancy management, including singularity avoidance and configuration (pose) control
- Robustness and fault tolerance

Table 1 summarizes the principal performance goals.

Table 1 Manipulator Performance Goals

Attribute	Target Value	Notes
Length	55 in.	shoulder pitch to toolplate
Weight (1 G)	165 lb.	actual weight - 150 lb.
Max Endpoint Speed	> 40 in./s	for task space moves
Payload Lateral Force	50 lb. 20 lb. 135 lb.	peak - short duration continuous duty at toolplate, fully extended
Dexterity	7 active degrees of freedom	
Repeatability	0.025 in.	
End Effector Support	72 wires	72 to forearm; 40 to toolplate

Note in particular the payload to weight ratio, which is just under 1/3. For comparison, the Puma 762 is rated for a maximum dynamic payload of 44 lb., and weighs approximately 1200 lb.

2.2.1 Kinematics

Figure 1 shows the Odetics Dexterous Manipulator. This kinematic arrangement of joint modules includes two shoulder modules, an upper arm roll module, an elbow module, and a three degree of freedom wrist module. Neither the shoulder nor the wrist axes are collocated. Although, from the control viewpoint, collocated axes are highly desirable, they require many poor trade-offs in size, packaging, strength, and weight, and thus make other manipulator performance attributes unreachable. The elbow (joint 4) offset allows the lower arm to actually fold up against the upper arm, providing excellent manipulator stowage.

Figure 1 **The Odetics Dexterous Manipulator**

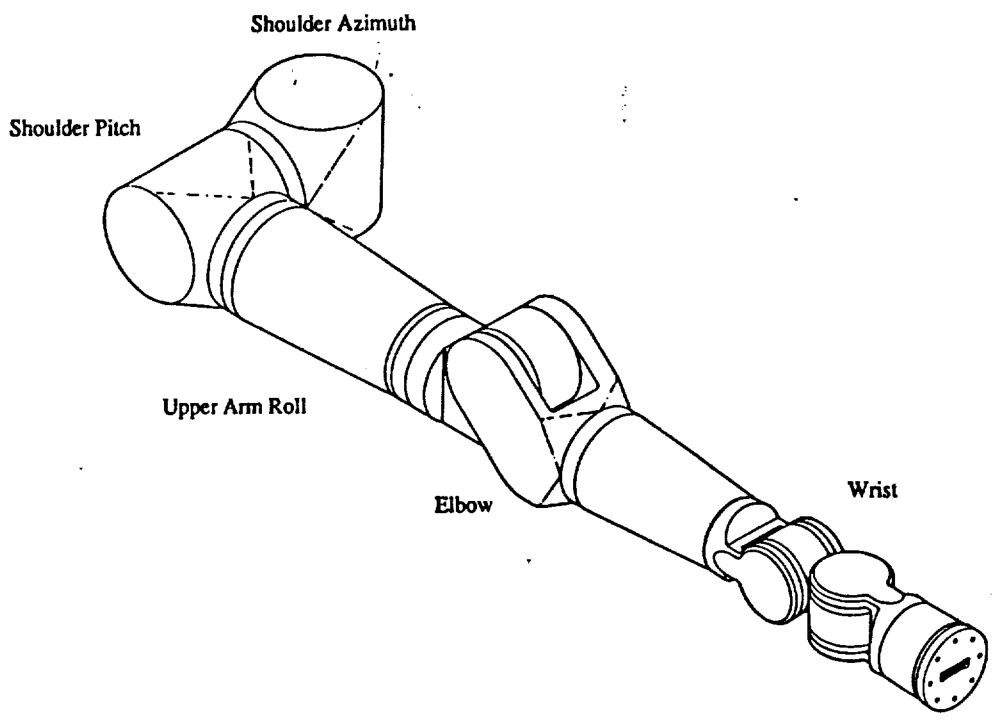


Table 2 shows the Denavit-Hartenberg parameters for the manipulator in the straight-out pose shown. Lengths are in inches and angles are in radians. The unusual parameters for links 4 and 5 result from the elbow offset.

Table 2 Denavit - Hartenberg Parameters

Link	θ	a	d	α
1	0	0	0	$\pi/2$
2	$-\pi/2$	0	8.5	$-\pi/2$
3	0	-2.75	24.125	$\pi/2$
4	1.433305	20.064350	0	0
5	0.137492	4.25	0	$-\pi/2$
6	$\pi/2$	0	0	$\pi/2$
7	0	0	6.75	0

The manipulator has two internal kinematic singularities. One occurs when the plane formed by the upper and lower arm links is vertical and the shoulder pitch (joint 2) axis lies in this plane. In this configuration, $\theta_2 = 0, \pi \quad \theta_3 = \pm\pi/2$. The second singularity occurs when the upper arm link is vertical and the wrist roll (joint 7) axis is normal to the upper arm-lower arm plane. In this configuration, $\theta_2 = 0, \pi \quad \theta_6 = 0, \pi$. While both singularities occur within the useful manipulator workspace, the second one is close to the wrist yaw (joint 6) axis joint limits, making it less problematic than the first singularity.

2.2.2 Joint Modules

Many of the innovative and unique features of the Odetics Dexterous Manipulator are apparent in the joint module design. Each module contains motors, sensors, wiring, transmission elements, and structure in a compact package. Each module uses exactly the same drivetrain concept, scaled according to that joint's torque requirements. Module interfaces consist of both positive mechanical connection and self-mating electrical connectors held together with simple clamping collars. This quick disconnect design allows the manipulator to be assembled or disassembled in approximately seven minutes.

As shown in Figure 1, there are four different types of modules. The two shoulder modules are identical. They provide the greatest output torque and finest position sensing resolution. The upper arm roll module rotates the plane formed by the upper and lower arm links, providing the ability to alter the manipulator configuration. The elbow module allows the manipulator to fold back on itself for stowage. The wrist module is a single unit containing three axes in a pitch-yaw-roll arrangement. This design is necessarily a compromise between conflicting kinematic, strength, and packaging requirements. There are approximately 40 wires brought out to the toolplate for auxiliary devices such as grippers. The modules have few fastener penetrations and provide clean surfaces that are easy to decontaminate and have no wires or protrusions to snag on the environment. Table 3 shows the pertinent characteristics of each module type.

Table 3 Module Performance Characteristics

Module	Range of Motion (rad)	Weight (lb)	Peak Torque (in-lb)	Peak Speed (rad/s)	Position Sensing Resolution (μrad)
Shoulder Azi	5.67	34.5	8000	1.25	10.9
Shoulder Ele	5.67	34.5	8000	1.25	10.9
Upper Arm Roll	12.51	27.5	4000	1.59	12.2
Elbow	4.10	24.5	4000	1.59	12.2
Wrist Pitch	4.15	27.5	1300	2.62	13.2
Wrist Yaw	3.63	27.5	1300	2.62	13.2
Wrist Roll	5.93	27.5	1300	2.62	95.9

2.2.3 Actuators and Transmission

One of the more difficult challenges in the Odetics Dexterous Manipulator design was to obtain very high torque levels while simultaneously producing a high precision mechanism, and fitting the result into as small a package as possible. The actuator transmissions use spur gear technology with special mesh geometries and materials to obtain high torque capability. These modifications conflict with the high precision requirement. In particular, the modified spur gears and planetary gear reducers used have a fairly large amount of backlash, which would make servo control problematic.

The solution to this problem is a unique transmission concept that uses two actuators connected to parallel gear trains, both of which drive a single output. This topology allows one actuator to be the "prime mover" while the second provides a small bias torque in the opposite direction to remove all backlash from both branches of the transmission. When large torques are required, the biasing actuator can "turn around" and provide torque to move the load. The algorithm for backlash management is described in Section 3.2.3.3. The additional actuator also provides tolerance to motor and motor driver failures. If one motor or its driver fails, the remaining motor is still capable of driving the joint, obviously at reduced bandwidth and torque capability. After the task at hand is completed, a fully functional module can be swapped with the degraded one, which could in turn be repaired off-line.

Each parallel drivetrain branch begins with a brushless D.C. motor. The motors are the frameless design, and are built with three phases connected in a "wye" configuration. Hall sensors are included for six step commutation. The custom windings operate at a nominal 300 VDC. Thermistors buried in the motor windings provide temperature information. Each motor is also equipped with its own fail-safe brake so that the manipulator can be stopped in any configuration. The motor shaft is geared to a planetary reducer. The reducer output pinions both drive a large internal ring gear that is connected to the joint output member.

2.2.4 Sensors

Each joint provides absolute joint position, derived joint velocity, and torque sensing for servo control, as well as motor winding temperature sensing for safety monitoring.

The joint position sensing scheme uses two sensors for each joint. The current manipulator design uses a potentiometer and a brushless resolver. Both are geared to the joint output using precision anti-backlash gears. These devices operate in a "two-speed" mode, providing much higher resolution than can be obtained from either one individually. In addition, the dual sensing scheme provides recovery from single point failures. If the resolver fails, the potentiometer can provide joint position feedback, with reduced servo bandwidth to compensate for the reduced resolution. If the potentiometer fails, the joint can continue to operate normally until the next power cycle, when the absolute joint position must be determined.

The very high position resolution makes it feasible to obtain velocity information by discrete differentiation (back differences) of the position information. Although the manipulator electronics includes circuitry to derive an analog velocity signal from the motor hall sensors, this circuit was not intended for feedback control and suffers from high ripple content at low velocities. Back differences of the position signal provides superior results at both low and high velocities. Space constraints within the joint modules make it infeasible to include velocity sensing devices such as tachometers.

The output member of each joint includes special structures instrumented with strain gauges such that joint axis torque measurements can be obtained. The strain gauge signals are amplified using a full bridge amplifier circuit that resides within the joint module. The joint torque information can be used for advanced control techniques such as force reflection or joint torque servoing. A/D conversion in the manipulator controller provides 12 bit resolution of the joint torque signals.

3.0 Control System Technical Description

The Odetics Dexterous Manipulator control system is the product of several successful embedded control system implementations for high performance robots, years of in-house research into different aspects of manipulator control, and the academic community's research results. A few principles guided the design; whenever possible, we have tried to:

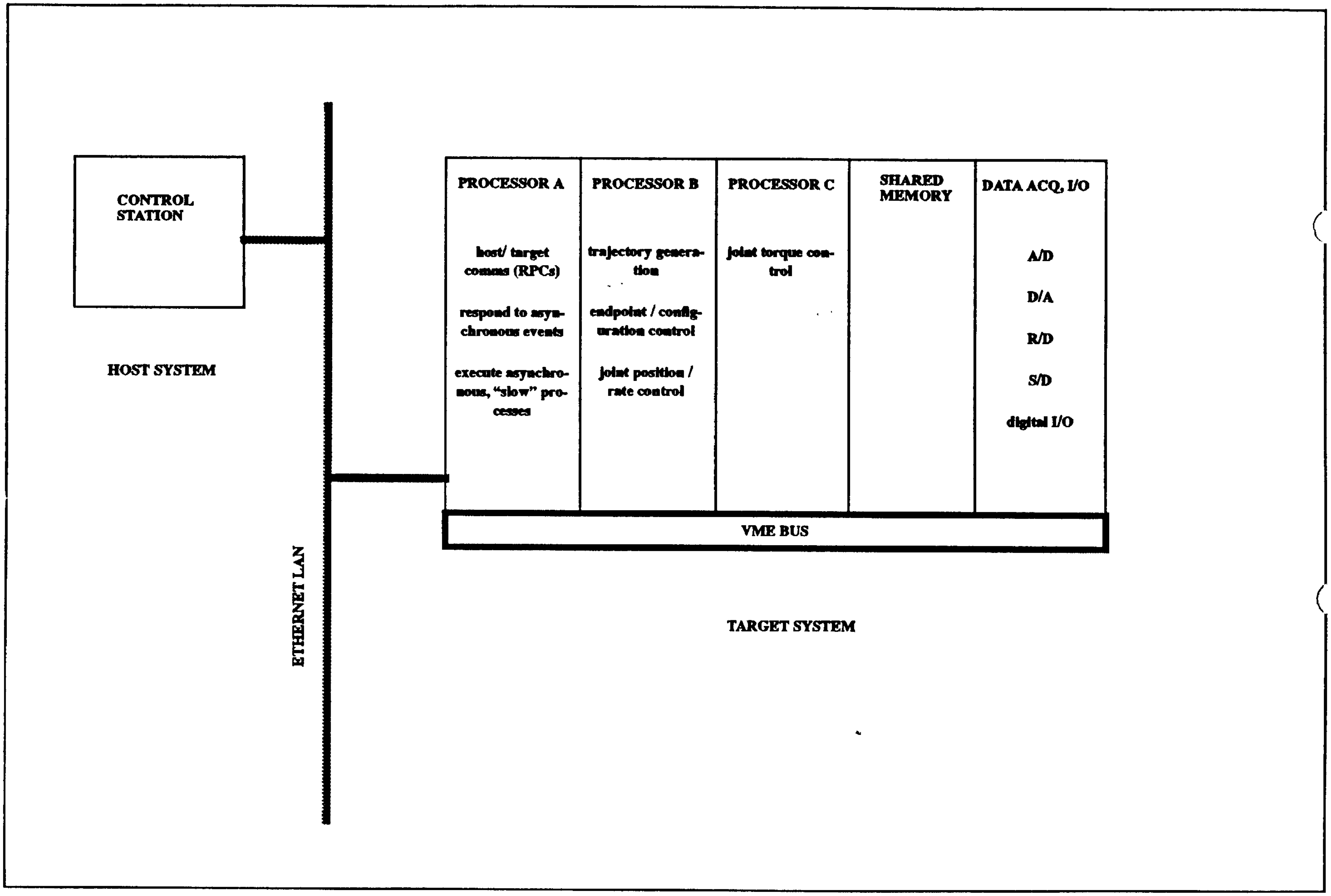
- leverage off previous work, implementing some of what has already been tested in simulation
- exploit sensor/actuator redundancy to provide a highly fault tolerant system
- use modular design and include the interfaces ("hooks") required to integrate other hardware and software subsystems for expanded capability, e.g., teleoperation and path planning
- make design choices that facilitate use of improved computer hardware, as it becomes available.

3.1 Architecture

At an abstract level, a control system architecture defines information flow between system and environment and within the system, and shows how the system takes action based on this information. The Dexterous Manipulator control system must capture information from external sources, such as an operator interface, and internal sensors, process this information, and produce physical signals to cause manipulator motion. It must perform these operations both in response to asynchronous external events and at regular, repeatable time intervals.

Figure 2 shows an overview of the system architecture. The embedded control computer, referred to as the "target" system, consists of three single board computers, data acquisition hardware, and memory, which share a VME backplane residing in a card cage. The section labeled "DATA ACQ, I/O" actually consists of several separate boards. Various processes and algorithms are allocated to the three processors. Processors B and C perform time critical control processes. These processors execute their processes synchronously: processor B executes at 50 Hz, and processor C executes at 500 Hz. Processor A executes non-critical algorithms and handles communications between the host computer and the target system. Data passes among the three target processors via shared memory, which includes each processor board's on-board memory as well as the separate memory expansion board. Data acquisition and digital I/O channels are memory mapped, and each processor accesses I/O with simple memory reads and writes. The target system is linked to a Sun workstation host computer via an Ethernet local area network. The host computer is used for development and to run the graphical user interface.

Figure 2 System Architecture



This architecture provides great flexibility in both hardware selection and software development. Many vendors sell processors, memory, data acquisition boards, and other special purpose hardware for VME systems. It is relatively simple to upgrade the control hardware as higher performance processors, memory, and data acquisition equipment become available. The inherent portability of the C programming language means that there is little difficulty porting the application code to the new hardware.

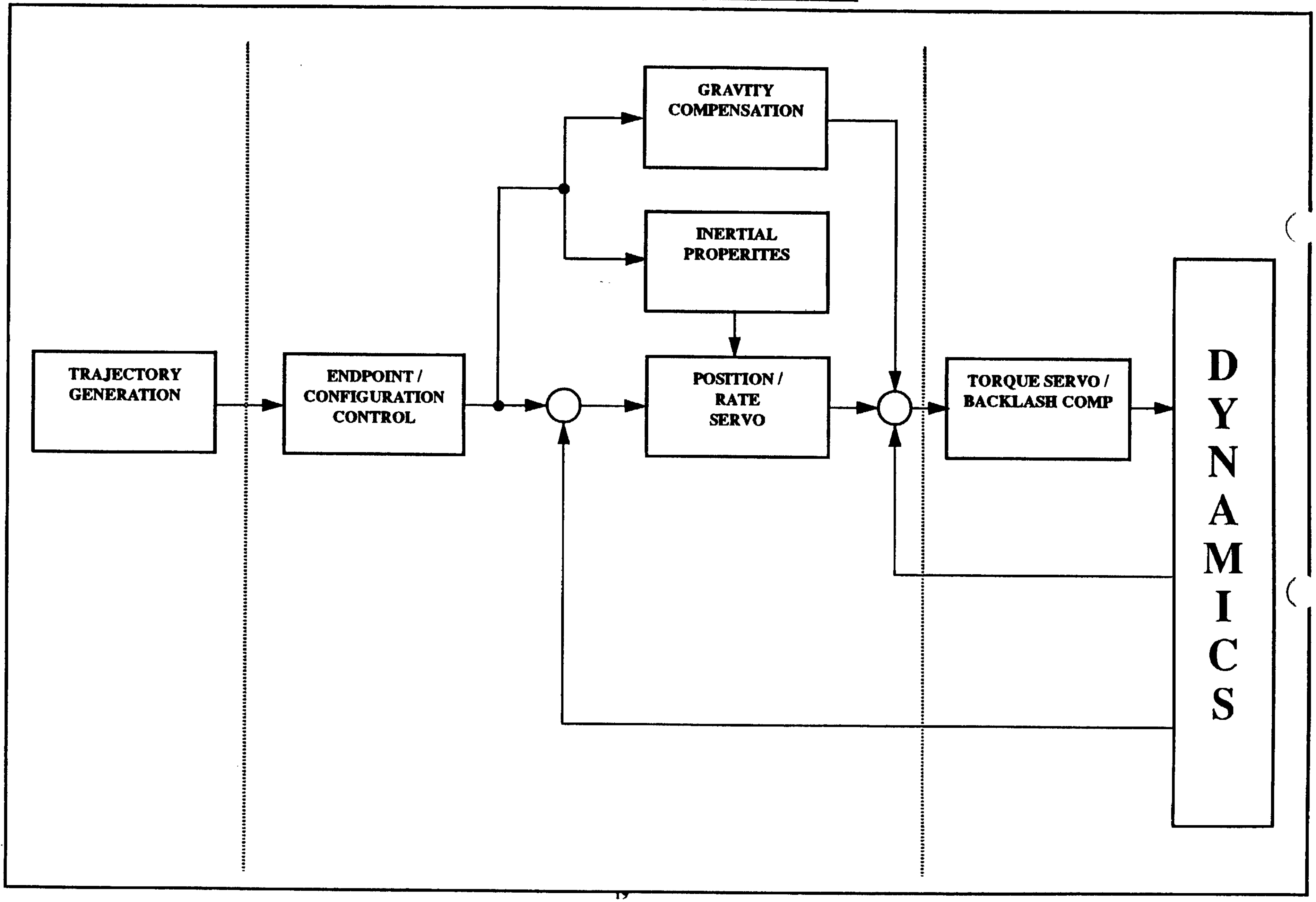
3.2 Algorithms

The control system algorithms are arranged hierarchically. Figure 3 shows the algorithm structure and information flow. The two dashed vertical lines divide the figure into three regions. The left most region contains non-real-time processes, the center region contains real time processes that execute at a 50 Hz. frequency, and the right most region contains processes that execute at a 500 Hz. frequency. The trajectory generator produces smooth endpoint/configuration trajectories, with setpoints spaced at the servo update rate. The endpoint/configuration control algorithm transforms endpoint position/orientation and arm configuration commands into joint position/rate commands. Note that this algorithm does not use feedback from the manipulator joints - it is decoupled from the servo algorithms and is thus unaffected by their dynamics. Inputs to this algorithm can come from several different sources. In the current implementation, prior to manipulator motion, the operator specifies either a set of trajectory pass-through points or a "delta" from the current arm position/orientation. The appropriate routine converts these inputs to a set of endpoint/configuration commands, discretized at the 50 Hz. position/rate servo update frequency.

The joint position/rate servo calculates and shapes position and rate errors to yield joint torque commands. The compensation is parameterized by the manipulator effective joint inertias in order to attain approximately configuration-independent dynamics. Feedforward compensation helps to reduce gravity disturbances and improve transient response. Both the inertia and gravity calculations use commanded rather than sensed joint positions.

The summed torque commands become input to the high bandwidth torque servos. These servos calculate and shape the joint torque errors to produce combined motor torque commands, discretized at the 500 Hz. torque servo update frequency. These commands are the motor torques that would be commanded in a single actuator system. Since the Dexterous Manipulator joints use dual actuators, another algorithm divides the combined motor torque commands into dual motor commands, biasing one motor against the other (when torque levels allow) to eliminate drivetrain backlash. The dual motor commands are converted to analog signals, which command the motor drivers.

Figure 3 Control Algorithm Structure



3.2.1 Trajectory Generation

The trajectory generator produces smooth endpoint commands, with setpoints spaced at the 50 Hz. servo update frequency. The trajectory generation process occurs before any motion, and thus does not execute in real time. While there are a couple of different methods to specify endpoint and configuration goals for the manipulator, the trajectory generation method is the same. A quintic polynomial, parameterized by time, is fitted to a set of pass-through points, which are spaced relatively widely in time. Setpoints spaced at 20 ms are then calculated from the quintic and stored in memory prior to actual motion.

The user can specify a trajectory by the following methods:

1. As a "delta" from the current manipulator position, i.e., as a vector $\left[\Delta X\ \Delta Y\ \Delta Z\ \delta\phi\ \delta\theta\ \delta\psi\right]$, along with a configuration command, if desired. The speed of the move is specified as a percentage of the manipulator's approximate maximum endpoint speed.
2. As a set of pass-through points, with the time between the points specified.
3. Single joint trajectories can be specified as a "delta" from the current joint position at some percentage of that joint's maximum speed. They can also be specified as sinusoids of a certain amplitude and frequency, primarily for testing purposes.
4. Joint space moves are supported as well. The user indicates the 7 desired joint angles and the percentage of maximum joint speed that he wishes the manipulator to move.

3.2.2 Endpoint / Configuration Control

The main characteristic that sets redundant manipulators apart is the ability to control manipulator configuration as well as endpoint position. As described in the introduction, configuration control provides a means to exploit the dexterity of a redundant manipulator. The manipulator configuration (pose) is altered via "self motion", which is manipulator joint motion that causes no endpoint motion. With manipulator pose control, movement in tightly constrained or obstacle strewn environments becomes tractable. In addition to specifying configuration, criteria involving proximity to joint limits or joint torques can be optimized to increase the manipulator's effective operating range.

Early approaches to redundancy resolution concentrated on optimization methods. Kinematic singularity avoidance is a goal used to formulate many optimization criteria. Many of the algorithms are based on the Moore-Penrose pseudoinverse, which yields a least squares solution for the inverse of a non-square matrix. Essentially, these algorithms yield a particular solution to:

$$dx = Jd\theta \tag{1}$$

as

$$d\theta = J^{\dagger} dx, \qquad J^{\dagger} = J^T (JJ^T)^{-1} \tag{2}$$

where dx and $d\theta$ are differential endpoint and joint motions, J is the nonsquare manipulator Jacobian matrix, and $J^{\dagger}$ is its pseudoinverse. The solution that the pseudoinverse method generates is the minimum norm solution of (1), which yields a set of joint motions that have no contribution to self-motion of the manipulator. However, self motion is required to modify manipulator configuration. To obtain self motion, a second homogeneous solution term is added to (2):

$$d\theta = J^{\dagger} dx + k(I - J^{\dagger} J) \nabla H \tag{3}$$

where I is the identity matrix, H is a function to be optimized, and k is a weighting factor. In Phase I of this project, this technique was used to control a redundant manipulator with a 4 DOF wrist [1]. The function H quantifies proximity to wrist singularity, and the weight k was a function of time and proximity to this singularity.

One difficulty with the pseudoinverse solution is that it is differential. The solution of (3) must be integrated numerically; therefore, the solution's accuracy will depend on the magnitude of the desired endpoint motion and the integration step size. An often discussed redundancy resolution algorithm characteristic is *cyclicity,* which is the property that closed endpoint trajectories in task space have closed joint space trajectories. In general, pseudoinverse solutions do not have the desirable cyclicity property because the differential solution does not yield an inverse function $\theta = F^{-1}(x)$ [2].

The endpoint/configuration control algorithm used with the Dexterous Manipulator solves both of these difficulties. It was presented in [3] and will be referred to as the "Chang" algorithm. The Chang algorithm provides an inverse kinematic solution $\theta = F^{-1}(x)$ rather than the differential solution of (3). This solution is numerical rather than analytic. As in (3), the algorithm optimizes an auxiliary function while achieving the endpoint command. However, algorithm convergence guarantees that the function H is at a local minimum for every trajectory point, which is not the case with the differential solution. Thus, the Chang algorithm provides a cyclical solution, except in the unusual case that the objective function contains separate local minima for configurations that are "close". Such behavior has not been observed in simulations of the Dexterous Manipulator.

It is important to note that this approach provides an exact endpoint position while *optimizing* an auxiliary function that specifies manipulator configuration. The algorithm will use manipulator self-motion to minimize the difference between the commanded and achieved configuration subject to attaining the exact endpoint position. If self-motion will not place the manipulator in the desired configuration, the manipulator will not reach that configuration.

An important feature of the endpoint algorithm implementation is that the endpoint algorithm is "decoupled" from the lower level servo control. Endpoint algorithm calculations that require joint coordinates use the current commanded joint angles, rather than the measured joint angles. Endpoint algorithm response is thus made independent of joint servo response.

3.2.2.1 Endpoint Algorithm Development

The following algorithm development follows that of [3]. The essence of the algorithm is to augment the underdetermined problem

$$x = f(\theta) \tag{4}$$

with additional equations to make the solution unique. For the Dexterous Manipulator, x is a 6x1 vector of desired endpoint coordinates, θ is a 7x1 vector of joint coordinates, and f is the forward kinematics transformation. In order to obtain the additional equation required, an optimization problem is posed and solved with Lagrange multipliers:

$$\text{minimize } H(\theta) \text{ subject to } F(\theta) = f(\theta) - x = 0 \tag{5}$$

Define the Lagrangian function

$$L(\theta) = \lambda^T F(\theta) + H(\theta) , \tag{6}$$

where λ is a 6x1 vector of Lagrange multipliers. The minimum of H occurs at a stationary point of L, determined as

$$\frac{\partial L}{\partial \theta} = \lambda^T \frac{\partial F}{\partial \theta} + \frac{\partial H}{\partial \theta} = 0 \tag{7}$$

Note that $\frac{\partial F}{\partial \theta} = J$, the manipulator Jacobian. Equation (7) can be rewritten as

$$\lambda^T J = -h^T, \tag{8}$$

where $h = [h_1, h_2, \ldots, h_n]^T$, $h_i = \frac{\partial H}{\partial \theta_i}, i = 1, 2, \ldots, n$, where $n = 7$ manipulator degrees of freedom. Note that Equation (8) is a system of 7 linear equations with 6 unknowns. Using Chang's notation, (8) can be re-written as

$$\begin{bmatrix} (J^1)^T \\ (J^2)^T \\ \vdots \\ (J^n)^T \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix} = - \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} , \tag{9}$$

where $(J^i)^T$ represents the transpose of column i of the Jacobian, and $m = 6$. Since this system is underdetermined, we can remove any one equation, solve the resulting system for the Lagrange multipliers, and substitute the result back into Equation (9). By removing the last row of (9), we obtain

$$\begin{bmatrix} J^{1T} \\ (J^2)^T \\ \vdots \\ (J^m)^T \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix} = -\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix} \tag{10}$$

Solving for the Lagrange multipliers yields

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix} = -\begin{bmatrix} (J^1)^T \\ (J^2)^T \\ \vdots \\ (J^m)^T \end{bmatrix}^{-1} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix}, \tag{11}$$

and substituting this result back into the remaining $n - m = 1$ equations of (9) yields

$$-\begin{bmatrix} J^{m+1T} \end{bmatrix} \begin{bmatrix} (J^1)^T \\ (J^2)^T \\ \vdots \\ (J^m)^T \end{bmatrix}^{-1} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix} = -\begin{bmatrix} h_{m+1} \end{bmatrix} . \tag{12}$$

At this point, Chang makes some notational simplifications:

$$J_m = \begin{bmatrix} (J^1)^T \\ (J^2)^T \\ \vdots \\ (J^m)^T \end{bmatrix}, \quad J_{n-m} = \begin{bmatrix} (J^{m+1})^T \end{bmatrix}, \quad h_m = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix}, \quad h_{n-m} = \begin{bmatrix} h_{m+1} \end{bmatrix} . \tag{13}$$

By using these substitutions and collecting terms, Equation (12) becomes

$$J_{n-m} J_m^{-1} h_m - h_{n-m} = 0 \tag{14}$$

Re-writing this equation in matrix form, we obtain

$$\begin{bmatrix} J_{n-m} J_m^{-1} & -I_{n-m} \end{bmatrix} \begin{bmatrix} h_m \\ h_{n-m} \end{bmatrix} = 0 \tag{15}$$

where I_{n-m} is the $n-m$ (in the 7 DOF case, one) dimensional identity matrix. In order to further simplify the equations, let

$$Z \equiv \left[J_{n-m} J_m^{-1} \; -I_{n-m} \right] \tag{16}$$

Then Equation (15) becomes

$$Zh = 0 \tag{17}$$

Note that, for the current case, J_{n-m} is 1X6 and J_m^{-1} is 6X6, so that z is 1X7. Since h is 7X1, Equation (17) is scalar. Taken together, Equations (4) and (17) provide seven equations in seven unknowns that solve Equation (5), fully specifying the 7 joint angles:

$$\begin{pmatrix} F = 0 \\ Zh = 0 \end{pmatrix} \tag{18}$$

Equation (18) is solved numerically at each endpoint trajectory point using the Newton-Raphson technique. Taking a Taylor Series expansion of Equation (18) and neglecting higher order terms, we obtain

$$\begin{aligned} 0 = F(\theta + \Delta\theta) &= F(\theta) + F'(\theta)\Delta\theta = f(\theta) - x + \frac{\partial f}{\partial \theta}\Delta\theta \\ 0 = zh(\theta + \Delta\theta) &= zh(\theta) + z\frac{\partial h}{\partial \theta}\Delta\theta \end{aligned} \tag{19}$$

Note that $\frac{\partial f}{\partial \theta} = J$, the manipulator Jacobian, and that $x - f(\theta)$ is the error between the desired and actual endpoint positions, which we call Δx. We can thus re-write Equation (19) as

$$\begin{aligned} J\Delta\theta &= \Delta x \\ z\frac{\partial h}{\partial \theta}\Delta\theta &= -zh \end{aligned} \tag{20}$$

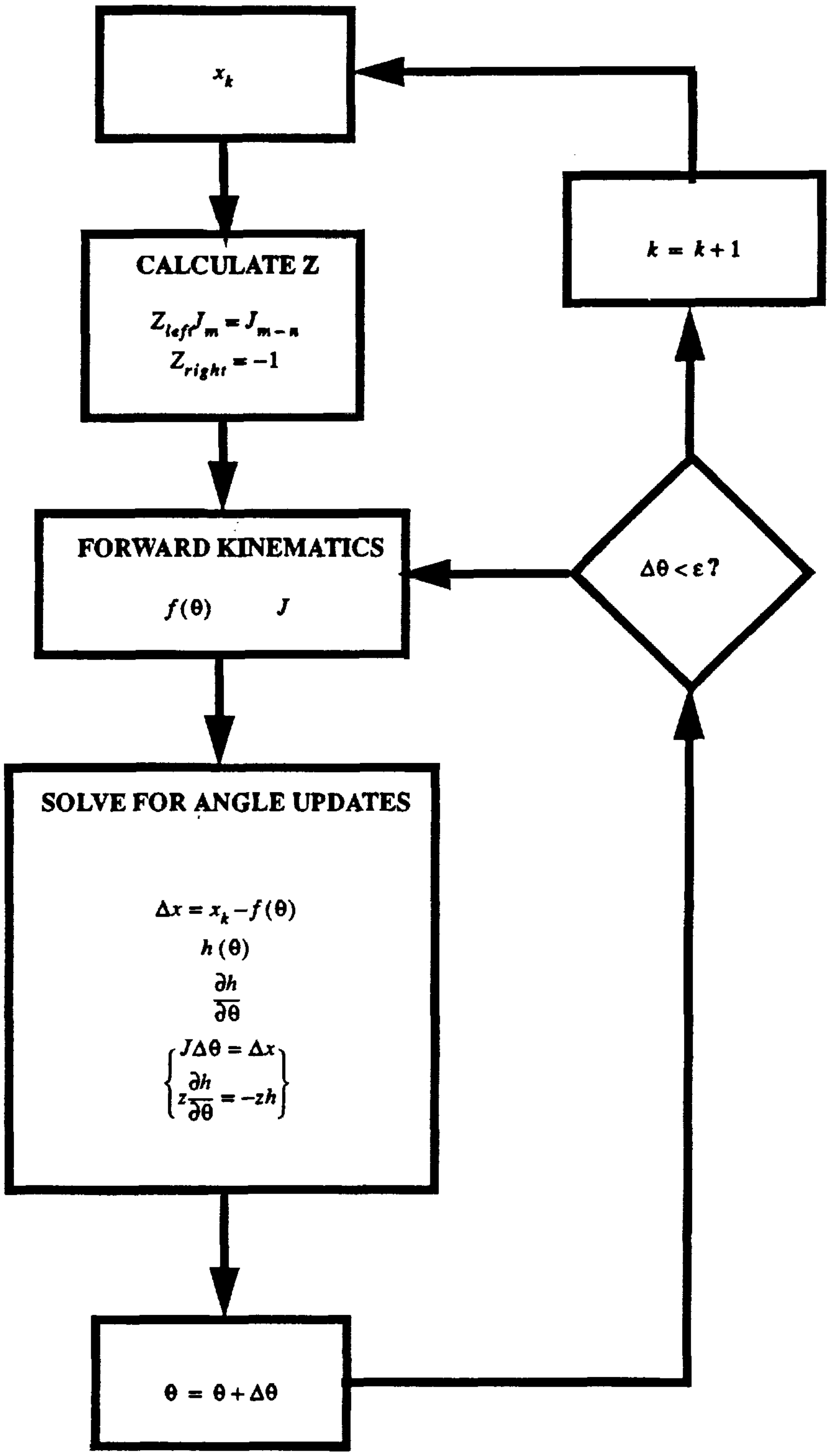
The joint angles θ are iteratively updated with the solution to the linear system (20) until (18) is satisfied to a desired tolerance. Figure 4 summarizes the procedure. First, a forward kinematics calculation provides the endpoint position/orientation and the manipulator Jacobian. The new endpoint command (x in (18)) is compared to $f(\theta)$, the endpoint position corresponding to the current set of joint angle commands, which yields an endpoint

error. Next, the functions h and $\frac{\partial h}{\partial \theta}$ are calculated. The left partition of z is calculated by solving

$$z_{left} J_m = J_{n-m}, \tag{21}$$

using gaussian elimination with partial pivoting and back substitution. For the 7 DOF case, the right partition of z is -1. The products $z\frac{\partial h}{\partial \theta}$ and zh are calculated next. Then the system (20) is formed and solved by the same gaussian elimination technique, and the joint angles are updated by $\Delta\theta$.

Figure 4 Endpoint Algorithm Structure


x_k
CALCULATE Z
$Z_{left}J_m = J_{m-n}$
$Z_{right} = -1$
FORWARD KINEMATICS
$f(\theta)$ J
SOLVE FOR ANGLE UPDATES
$\Delta x = x_k - f(\theta)$
$h(\theta)$
$\frac{\partial h}{\partial \theta}$
$\left\{ \begin{matrix} J\Delta\theta = \Delta x \\ z\frac{\partial h}{\partial \theta} = -zh \end{matrix} \right\}$
$\theta = \theta + \Delta\theta$
$\Delta\theta < \varepsilon$?
$k = k+1$

The objective function H is the sum of several functions that are designed to have minima for desired manipulator behavior and grow large during undesirable behavior. In addition, these functions have simple forms so that obtaining their first and second derivatives and calculating these derivatives in real time are tractable problems. When these functions are summed into H, it is possible for the various behaviors to conflict. Scale factors ("weights") multiply terms in each of the functions in H and allow the operator to control the contribution of each optimization criterion to the summed function.

Inverse square potential functions serve well for joint limit avoidance and velocity minimization. For joint limit avoidance, the objective function is

$$H_{lim_i} = \frac{K_{lim_i}}{\left[1 - \left(\frac{\theta_i - \theta_{bias_i}}{\theta_{max_i}}\right)^2\right]^2} , \quad i = 1 \ldots 7 , \tag{22}$$

where θ_{bias} makes the joint range of motion symmetrical, θ_{max} is the absolute value of the joint maximum (or minimum) angle, and K_{lim} is a scale factor. For joint velocity minimization, the objective function is

$$H_{vel_i} = \frac{1}{2} \frac{K_{vel_i}}{(\theta_{init_i} - \theta_i)^2}, \quad i = 1 \ldots 7 , \tag{23}$$

where θ_{init} is the joint angle at the start of an endpoint algorithm iteration, θ is the joint angle after the iteration, and K_{vel} is a scale factor.

For singularity avoidance, a trigonometric form of H is useful. Recall that the two manipulator internal kinematic singularities occur at

$$\{\theta_2, \theta_3\} = \begin{Bmatrix} 0, \pm\frac{\pi}{2} \\ \pi, \pm\frac{\pi}{2} \end{Bmatrix}$$

$$\{\theta_2, \theta_6\} = \begin{Bmatrix} 0, 0 \\ 0, \pi \\ \pi, 0 \\ \pi, \pi \end{Bmatrix}$$

The objective function should become large near these joint configurations and remain small at other configurations. For the first case, a function with the proper behavior is

$$H_{sing} = \frac{K_{sing}}{\cos^2\theta_3(\cos^2\theta_2 + 1) + \sin^2\theta_2(\sin^2\theta_3 + 1)} \quad . \tag{24}$$

It is possible to derive an even simpler function for the second case:

$$H_{sing} = \frac{K_{sing}}{\cos^2\theta_2 + \cos^2\theta_6} \quad . \tag{25}$$

In practice, the second singularity rarely occurs because joint 6 is near its limits. The objective function's joint limit avoidance component serves to keep the arm away from this singularity.

Configuration control (or more precisely, configuration optimization) takes two different approaches. One approach is to specify the orientation of the plane formed by the "upper arm" and "lower arm" links of the manipulator. Joint axis 4 (the elbow joint axis) is a normal to this plane. A simple way to specify the plane's orientation is to specify the vertical direction cosine Φ of the arm plane normal:

$$\Phi = \cos\psi = \sin\theta_2\sin\theta_3 \quad . \tag{26}$$

The vertical direction cosine Φ is an easily calculated function whose range $[-1, 1]$ provides an intuitive way to specify the arm plane as vertical or horizontal, as well as to specify on which side of the shoulder to place the arm plane. Specifying a direction cosine of the arm plane angle rather than the arm plane angle itself also eliminates the need to take derivatives of inverse trigonometric functions in the objective function. A simple objective function for this configuration optimization method is

$$H_{config} = \frac{1}{2}K_{config}(\Phi - \cos\Psi_d)^2 , \tag{27}$$

where Ψ_d is the desired arm plane angle.

An alternative method of configuration control is to construct an inverse square potential function whose center will repulse a set of points on the manipulator. An example is

$$H_{config} = \frac{K_{config}/2}{(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2} , \tag{28}$$

where (x_c,y_c,z_c) is the potential function center, and (x,y,z) is a point on the manipulator to be repulsed. Currently, this point is at the center of the elbow.

3.2.3 Joint Level Control

Conventional robotic manipulators generally use linear feedback control laws to independently servo each joint to a desired position. The performance of this method depends on various factors related to the manipulator design, control system implementation, and operation conditions. Some of these factors include:

- magnitude of nonlinear behavior, such as joint friction, motor torque disturbances, and dynamic coupling, relative to linear behavior
- bandwidth-limiting constraints, such as sensor noise, structural/actuator resonant frequencies, and other unmodeled dynamic behavior
- control computer performance, as measured by achievable sample rate for a desired control law
- expected variations in required speed, payload, and accuracy for different tasks.

Some of these factors are quantifiable during the design phase, while others cannot be determined until the hardware is built or specific tasks are defined. For example, it appeared clear early in the development that dynamic coupling between the joints would be relatively insignificant because of the large reduction ratios. The joint torque loops would help to reduce friction and improve the joint dynamics. For these reasons, it appeared that linear control laws with some nonlinear compensation for inertia variations and gravity would provide good dynamic response. Simulation would help verify this belief, and it would also provide a good means to compare the performance of a more advanced adaptive controller to the design method.

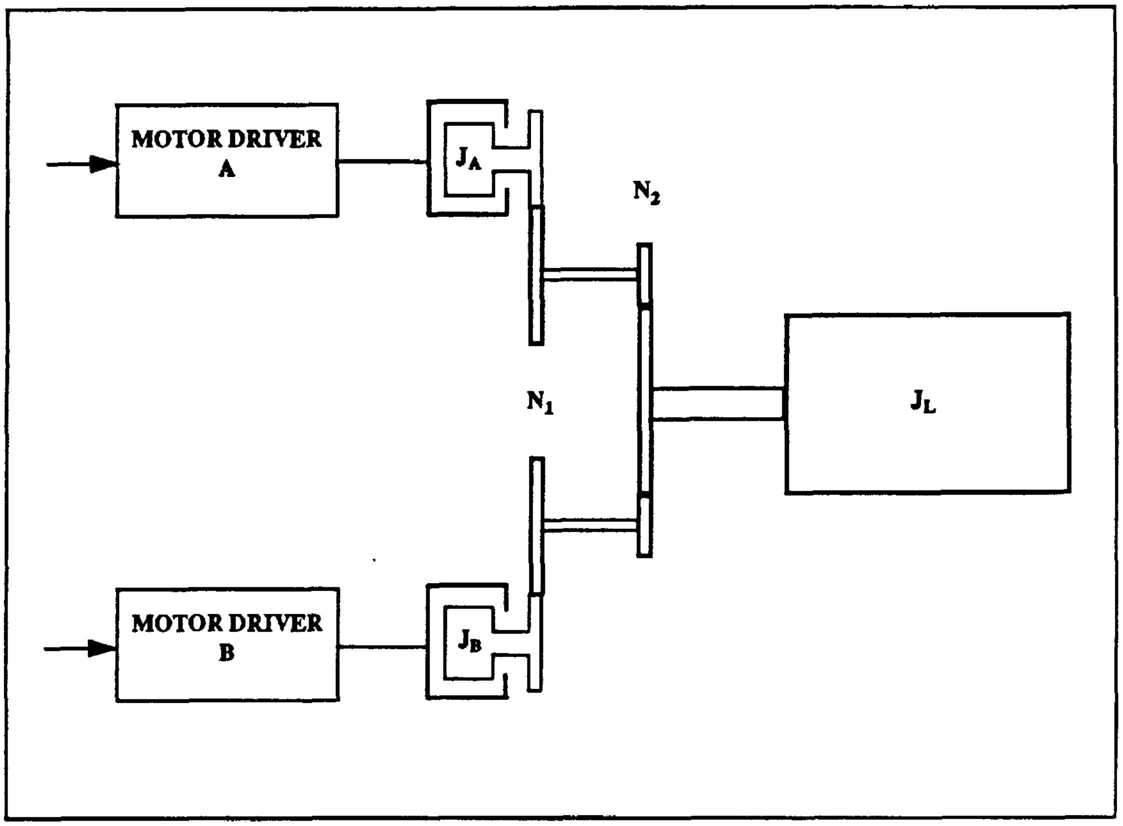
Several options for joint position/rate servoing were considered during the project's conceptual design phase. Some of the techniques considered include:

- linear feedback control
- model-based decoupling feedback control
- model-based feedforward compensation
- performance-based adaptive control.

3.2.3.1 Modelling

Figure 5 shows a simple schematic diagram of the joint actuator system. The two identical motor/reducer drivetrain branches drive a common output ring gear that is attached to the output member. The motor drivers operate in "current loop" mode: the motor winding current is proportional to the motor driver input command. Reducers N_1 are planetary spur gear reducers, while reduction N_2 consists of two pinions driving a ring gear. The two motors can drive antagonistically to remove all backlash from the drivetrain, or synergistically to provide maximum torque. Most of the backlash occurs at the output mesh; drivetrain inertia before this mesh is lumped into the two drivetrain branch inertias J_A and J_B. Joint position, rate, and torque sensing occur at the load.

Figure 5 Joint Actuator Schematic



Two simplifications were made for control design:

1. The antagonistic motor action effectively eliminates all drivetrain backlash.
2. Drivetrain component stiffnesses are high enough so that drivetrain flexibility and the ensuing resonant mode of this non-collocated actuator/sensor system are beyond the desired servo bandwidth.

The first assumption has been proven out in implementation. During the design phase, the second assumption seemed reasonable. There are no clearly "soft" components, such as harmonic drives, in the drivetrains. In implementation, this assumption has proven inappropriate. Finite drivetrain stiffness effects are discussed further in Section 5.1.1.

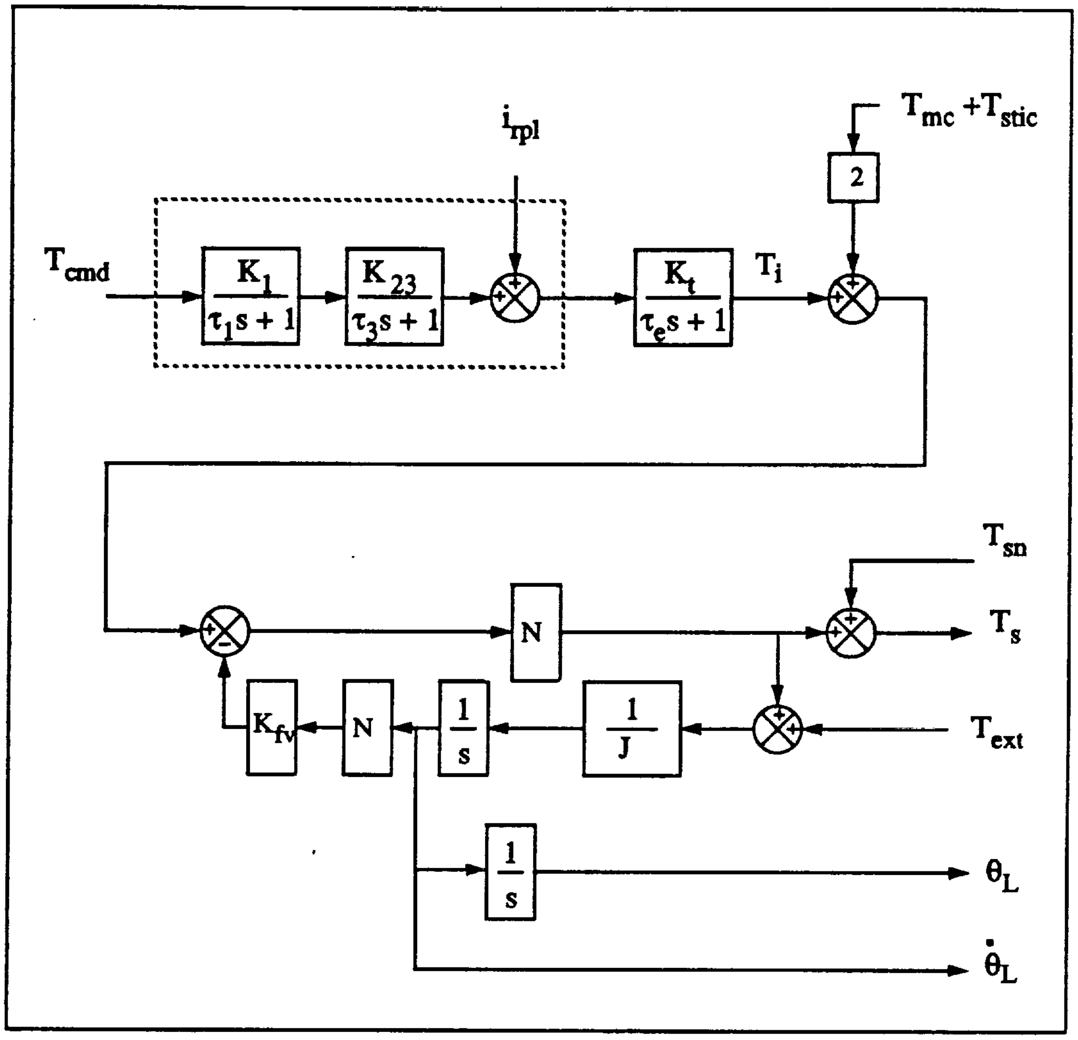
By using these two simplifications, we can develop the simplified linear model for servo synthesis shown in Figure 6. Note that the motor electrical dynamics and disturbance inputs for the two drivetrain branches are lumped into the single path at the top of the figure. Table 4 defines the model nomenclature.

Table 4 Lumped Motor / Drivetrain Model Nomenclature

Symbol	Variable	Units
T_{cmd}	motor driver command	V
τ_1, τ_3	motor driver time constants	sec
K_1, K_{23}	motor driver gains	A/V
i_{rpl}	motor driver current ripple	A
K_t	motor torque constant	in-lb/A
τ_e	motor electrical time constant	sec
T_i	motor torque	in-lb
T_{mc}	motor cogging torque	in-lb
T_{stic}	Coulomb friction and stiction	in-lb
K_{fv}	drivetrain viscous friction	in-lb-s
N	reduction from motor to output	-
J	lumped drivetrain and load inertia	in-lb-s^2
T_{sn}	torque sensor noise	in-lb
T_s	sensed torque	in-lb
T_{ext}	external load torque	in-lb
$\dot{\theta}_L$	load velocity	rad/s
θ_L	load position	rad

Note that torques T_{cmd}, T_i, T_{stic}, and T_{mc} are measured in motor coordinates, as is the viscous damping coefficient K_{fv}. Also, the product of the gains K_1 and K_{23} has the units A/V, rather than the individual gains.

Figure 6 Lumped Motor / Drivetrain Model



For servo design, we will derive a state space plant model of the form

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{29}$$

where x is the state vector, u is the input vector, and y is the output vector. For this 5th order plant, a state space model with the appropriate inputs and outputs is

$$\frac{d}{dt}\begin{bmatrix}\theta_L\\ \dot{\theta}_L\\ T_i\\ xp_1\\ xp_2\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0 & 0\\ 0 & -\frac{K_{fv}N^2}{J} & \frac{N}{J} & 0 & 0\\ 0 & 0 & -\frac{1}{\tau_e} & \frac{K_t}{\tau_e} & 0\\ 0 & 0 & 0 & -\frac{1}{\tau_3} & \frac{K_{23}}{\tau_3}\\ 0 & 0 & 0 & 0 & -\frac{1}{\tau_1}\end{bmatrix}\begin{bmatrix}\theta_L\\ \dot{\theta}_L\\ T_i\\ xp_1\\ xp_2\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & \frac{2N}{J} & \frac{1}{J} & 0\\ 0 & 0 & 0 & \frac{K_t}{\tau_e}\\ 0 & 0 & 0 & 0\\ \frac{K_1}{\tau_1} & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}T_{cmd}\\ T_{mc}\\ T_{ext}\\ i_{rpl}\end{bmatrix} \qquad (30)$$

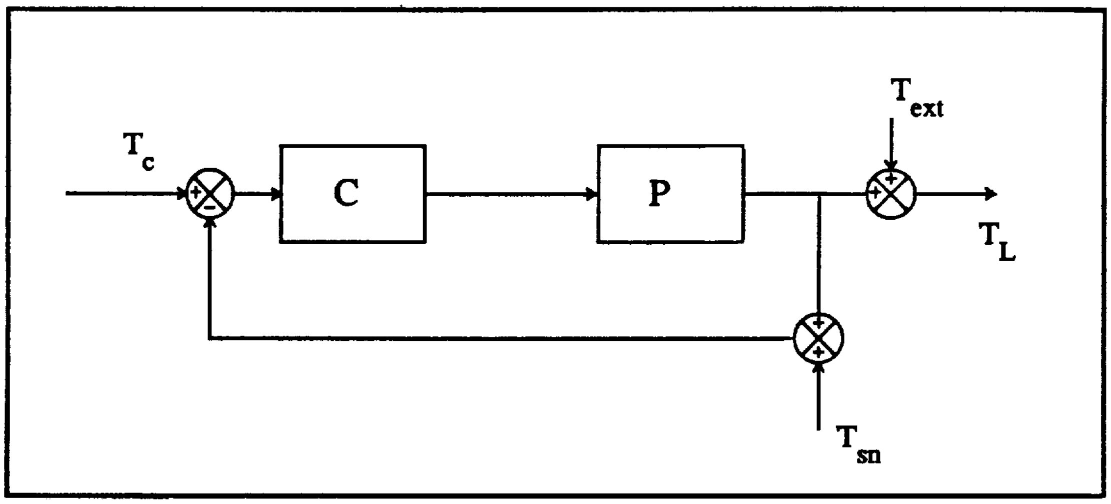
$$\begin{bmatrix}T_s\\ \theta_L\\ \dot{\theta}_L\\ T_L\end{bmatrix} = \begin{bmatrix}0 & -K_{fv}N^2 & N & 0 & 0\\ 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 0 & -K_{fv}N^2 & N & 0 & 0\end{bmatrix}\begin{bmatrix}\theta_L\\ \dot{\theta}_L\\ T_i\\ xp_1\\ xp_2\end{bmatrix} + \begin{bmatrix}0 & 2N & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 2N & 1 & 0\end{bmatrix}$$

The first two states are the load position and velocity. Motor torque is the third state. The states xp_1 and xp_2 correspond to the two motor driver electrical poles. The first input, T_{cmd}, is the motor driver command, while the latter three inputs represent disturbances. The outputs are sensed load torque, load position, load velocity, and actual load torque.

3.2.3.2 Joint Torque Servo Control

Figure 7 shows a generic block diagram of a "generic" closed loop joint torque controller. We will describe torque servo design goals by referring to the transfer functions for the various system inputs.

Figure 7 Generic Torque Servo



The figure's symbols have the following meanings:

- C = compensator
- P = plant
- T_c = torque command
- T_{ext} = external load torque disturbances
- T_L = load torque
- T_{sn} = torque sensor noise

The transfer function from commanded torque to load torque is

$$\frac{T_L}{T_c} = \frac{CP}{1+CP}$$

Ideally, this transfer function would be unity for frequencies up to the desired closed loop bandwidth, which requires that $CP \gg 1$ over this frequency range. Above the desired closed loop bandwidth, the transfer function magnitude should roll off as quickly as possible. The closed loop bandwidth should be selected so that it

- encompasses torque signals in the expected frequency range
- is high enough to make the actuator system appear as an ideal torque source to the position/rate servo loop
- is limited for good torque sensor noise rejection and to achieve a sample rate 10X higher than the bandwidth.

Based on these criteria, the design closed loop torque bandwidth was chosen as 50 Hz.

The transfer function from external load disturbances to load torque is

$$\frac{T_L}{T_{ext}} = \frac{1}{1+CP}$$

This transfer function should be small at all frequencies in order to reject this input's contribution to the load torque output, which requires that $CP \gg 1$.

Finally, the transfer function from sensor noise to load torque is

$$\frac{T_L}{T_{sn}} = -\frac{CP}{1+CP}$$

As with the previous disturbances, this transfer function should be small at all frequencies, which requires that $CP \ll 1$. Since the magnitude of this transfer function is the same as command to load torque transfer function magnitude, sensor noise cannot be rejected at frequencies below the desired closed loop bandwidth without also rejecting the command signal.

In the actual design process, the plant model was programmed into a Matlab script that calculates closed loop response and generates Bode, Nichols, and time response plots. Using nominal parameter values, a continuous time compensator design was developed and evaluated. The most important evaluation factors were stability and stability robustness. As the compensator design evolved from one design iteration to the next, the design's stability margins were evaluated first. Next, command tracking was examined, using the frequency response and step response of the closed loop system. External torque disturbances, stiction, cogging, and ripple response were examined, using their transfer functions. Nonlinear torque limiting and velocity limiting effects were ignored for this analysis. Known parameters were varied over their ranges and the design re-checked for acceptability. Once the continuous compensator design was acceptable, it was discretized, using a bilinear transformation with pre-warping, and converted to a difference equation for software implementation.

The form of the torque loop compensator is

$$k\frac{\tau_1 s + 1}{\frac{s^2}{\omega^2} + \frac{2\zeta s}{\omega} + 1} \tag{31}$$

The quadratic lag quickly rolls off the open loop gain (due to the gear ratio) at frequencies above the design open loop crossover frequency. The zero, placed below the crossover frequency, provides lead for an adequate phase margin. The gain sets the crossover frequency. This fixed gain and shaping stabilizes the design despite plant inertia variations. There is little variation in bandwidth and damping with payload or pose variation.

The following figures depict the design process. Figure 8 is a Bode plot of the continuous plant response from T_{cmd} to T_{sensed}, from Equation (30). Figure 9 shows the continuous compensator response. Figure 10 and Figure 11 show the continuous and discrete open loop compensated plant. The crossover frequency is approximately 40 Hz. Figure 12 is a Nichols plot of the compensated plant response. The design provides conservative stability margins: the phase margin is 60 degrees, and the gain margin is 25 dB. Figure 13 and Figure 14 show the continuous and discrete closed loop torque response. These plots show that the system behaves as an ideal torque source out to the closed loop bandwidth. The position response of the closed torque loop system, shown in Figure 15, also illustrates this behavior. Over the closed torque loop bandwidth, the position response rolls off at 40 dB/decade, with 180 degrees of phase lag. In essence, the torque servo loop makes the plant behave as a simple double integrator, which simplifies the position/rate servo design.

Figure 8 **Torque Servo Design Plant**

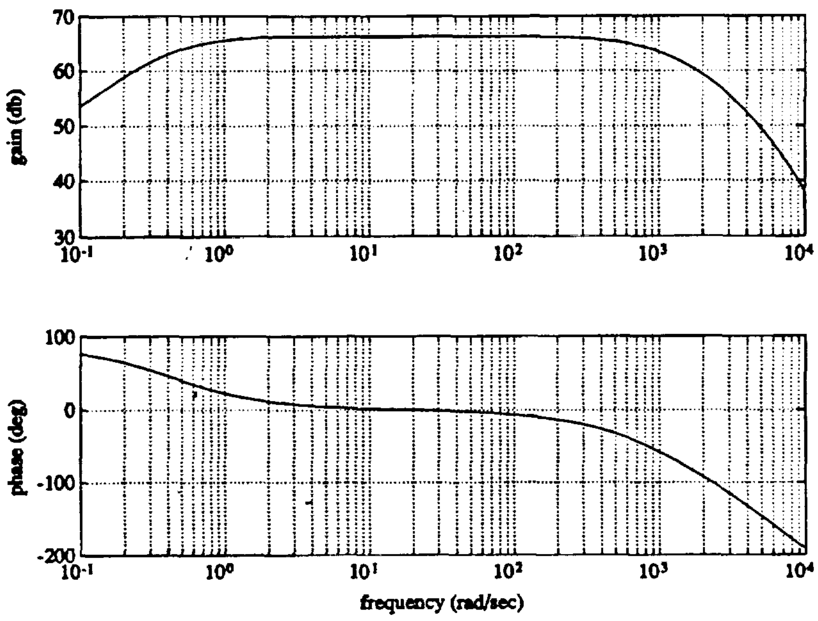


Figure 9 Torque Servo Compensator

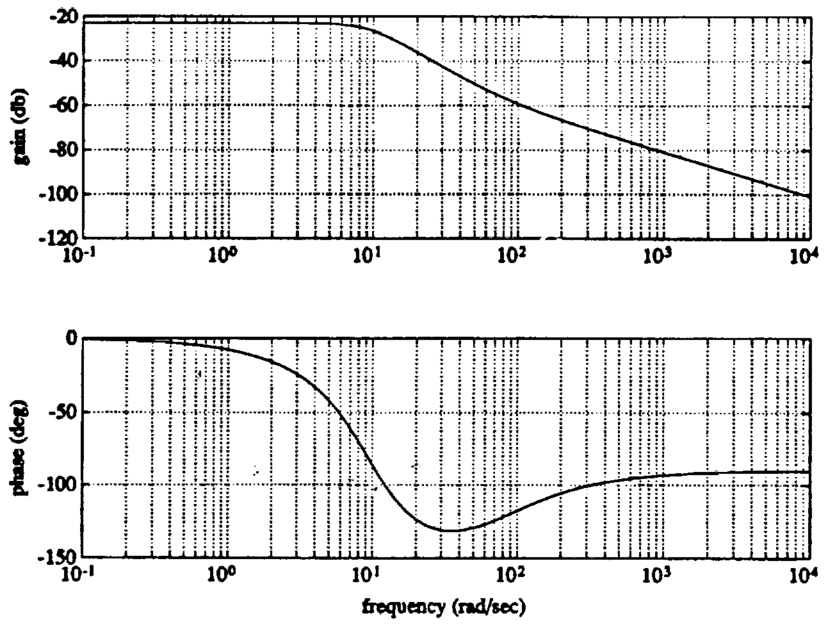


Figure 10 Continuous Compensated Plant

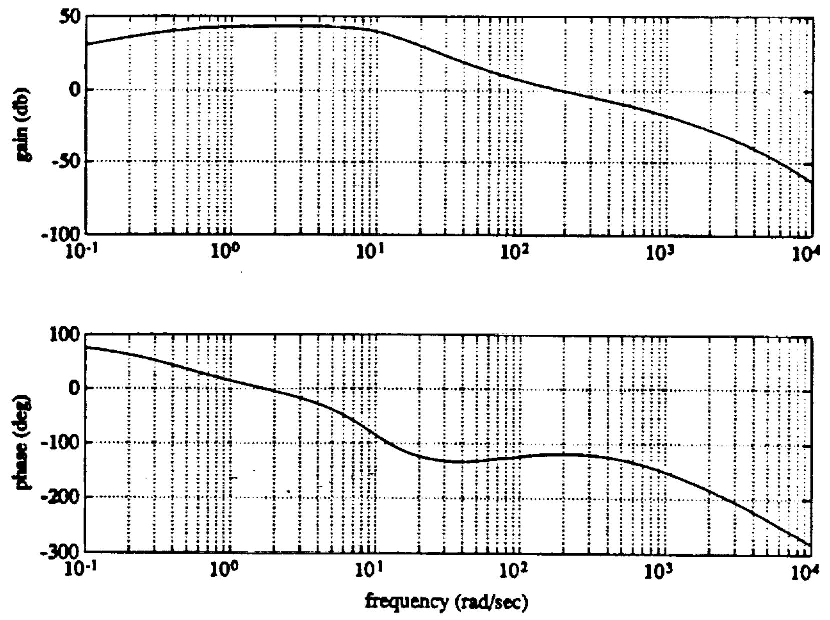


Figure 11 **Discrete Compensated Plant**

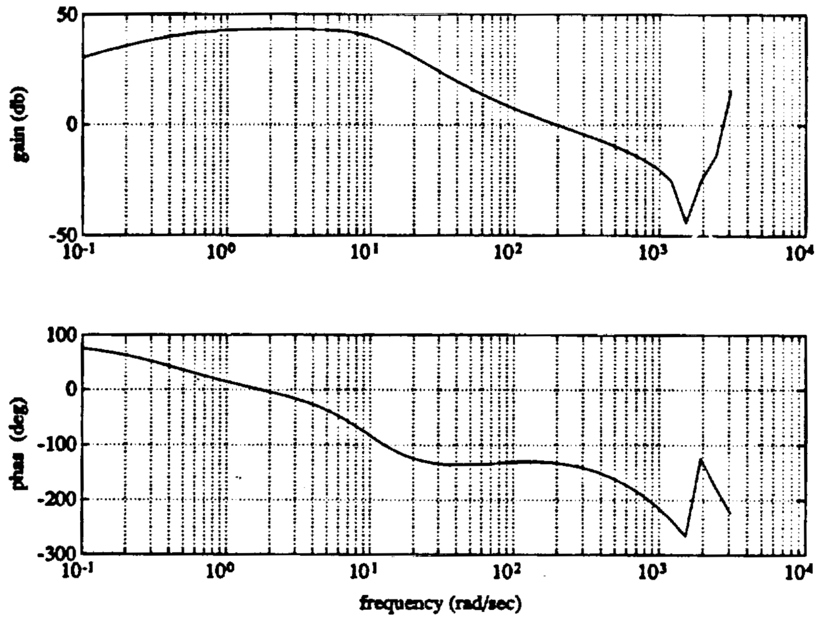


Figure 12 **Continuous Compensated Plant (Nichols Plot)**

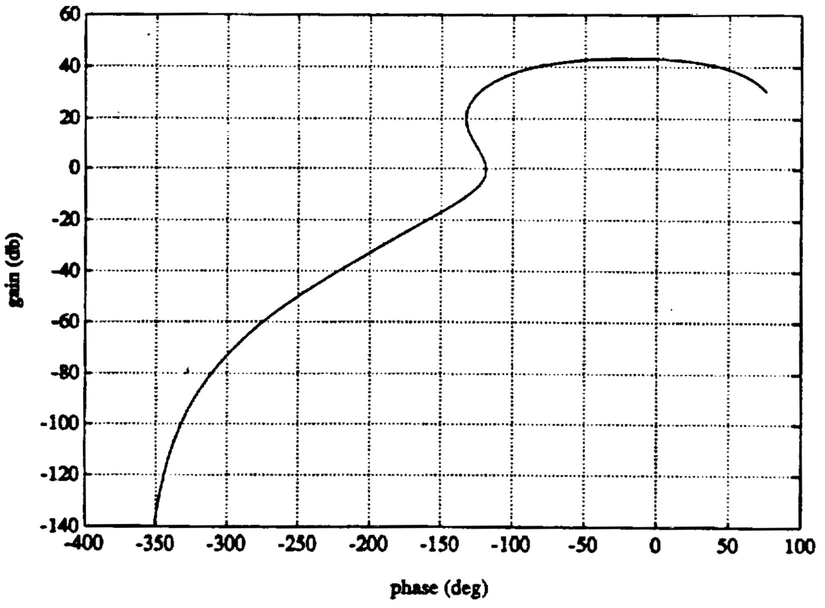


Figure 13 Continuous Closed Loop Torque Response

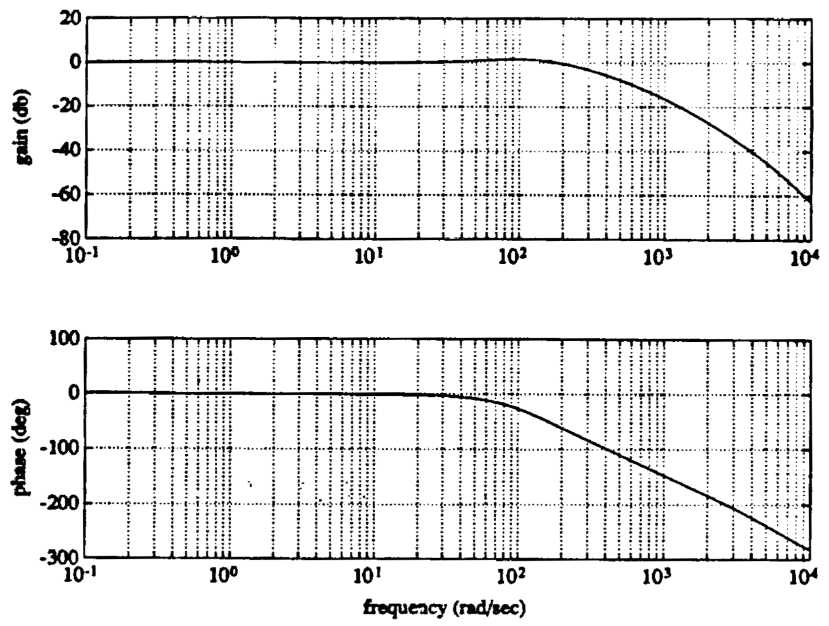


Figure 14 Discrete Closed Loop Torque Response

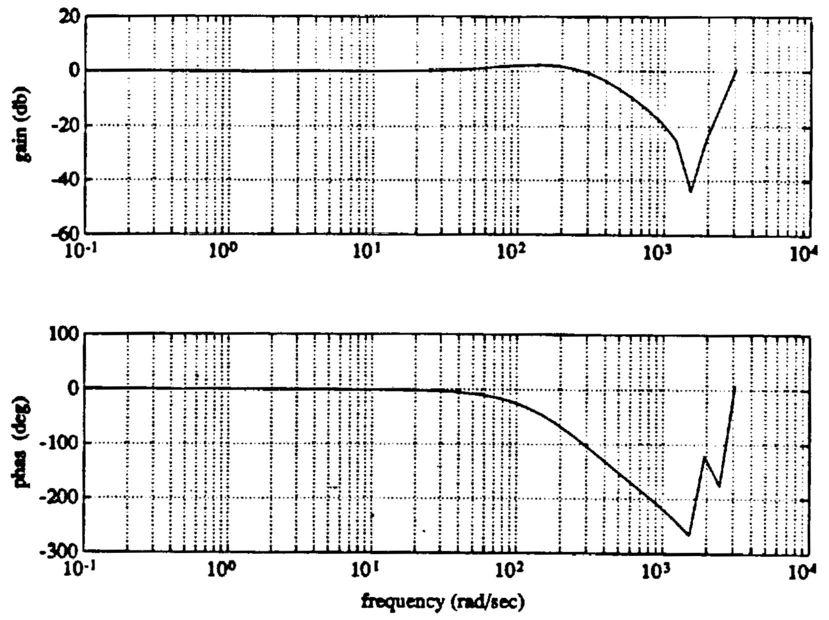
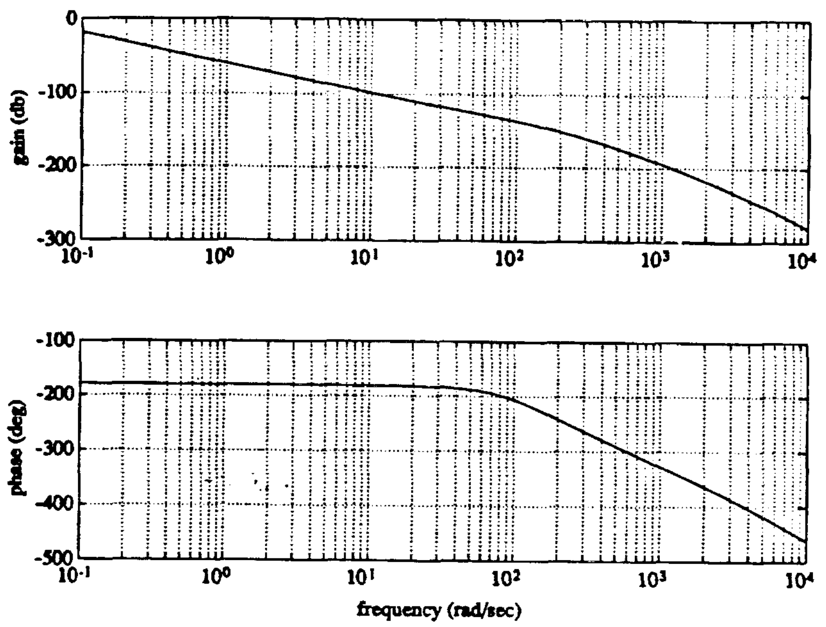


Figure 15 Open Loop Position Response with Torque Loop



3.2.3.3 Backlash Elimination

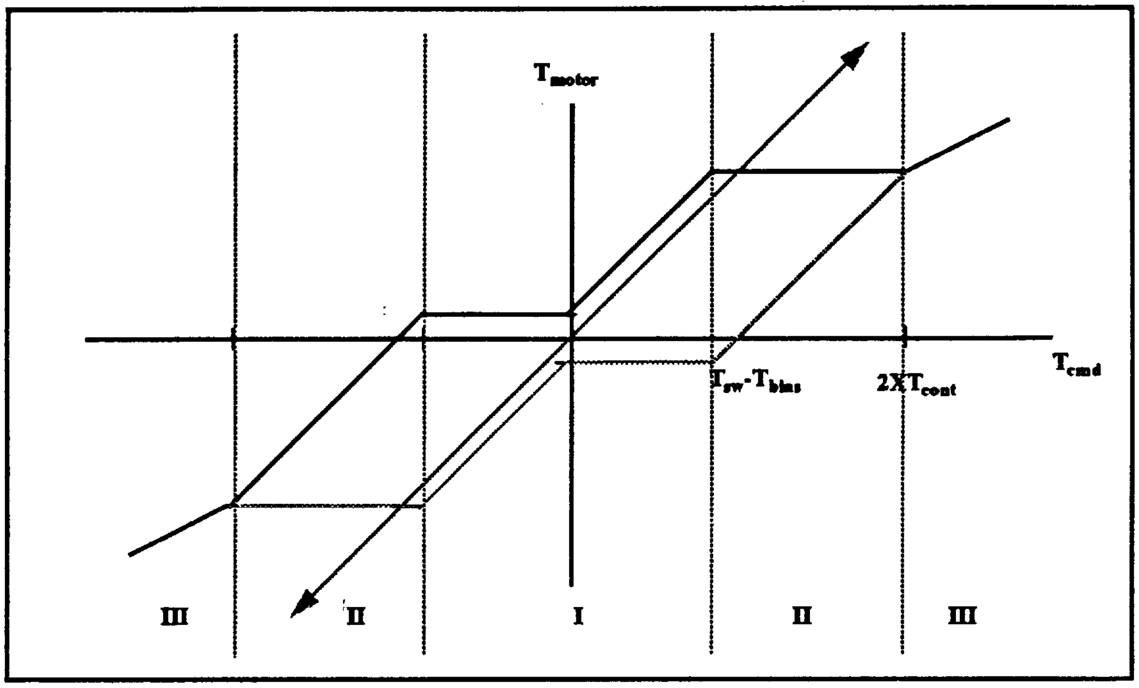
The antagonistic actuator arrangement provides the means to eliminate drivetrain backlash. In essence, the method is to use one motor as the prime mover, while the second motor exerts a small torque of opposite sense which keeps the drivetrain components on one side of the backlash. The dual motor drivetrain is neither new nor unique. *This implementation's unique feature is to switch between antagonistic operation with no backlash and cooperative operation, based on torque demand.* With this feature, each joint operates backlash-free over a large part of its range. When necessary, backlash-free operation can be traded for maximum torque. Figure 16 illustrates the method. The principal parameters are:

- T_{bias} - the antagonistic torque that keeps the drivetrain on one side of the backlash
- T_{sw} - the torque level at which the algorithm transitions from "zero backlash" mode to "maximum torque" mode.

The horizontal axis is total commanded torque (in motor coordinates), and the vertical axis is corresponding motor torques. The light and dark line segments represent the two individual motor torques. The continuous line represents the summed motor torques, which should equal the total commanded motor torque, up to the motors' limits.

In region I, the commanded motor torque magnitude is below T_{sw} - T_{bias}. One motor maintains a constant bias torque, and there is no backlash. In region II, commanded torque exceeds $T_{sw} - T_{bias}$, and the biasing motor "turns around" to help the prime motor, which continues to torque at T_{cont}. In region III, the torque command magnitude exceeds 2 X T_{cont}. Both motors torque to one half the commanded value, up to 2 X T_{peak}, where the torque command is clipped. In each case, the sum of the two motor torques always equals the torque command, up to the level 2 X T_{peak}.

Figure 16 Anti-backlash Algorithm



Parameter T_{sw} controls the trade-off between a large region of no backlash (region I) and an acceptable motor duty cycle. For the most conservative operation, T_{sw} is set equal to T_{cont}, and the motors will never overheat while operating in regions I and II. Of course, a less conservative value of T_{sw} may be chosen.

A more sophisticated method of backlash and motor duty cycle management is to make T_{sw} a function of time and commanded torque:

$$T_{sw} = e^{-\alpha s}(T_{peak} - T_{cont}) + T_{cont} \quad (32)$$

The parameter s is a counter (equivalent to time) that increments when the commanded torque exceeds the rated motor continuous torque, and decrements when the commanded torque is less than the rated motor continuous torque. The parameter α is the motor thermal time constant. When the motors are cool, the maximum joint torque can be twice the rated peak motor torque (multiplied by the reduction). As the motor duty cycle increases, the maximum torque decreases to twice the rated continuous torque. This function provides constraints on motor torque without being overly conservative, which limits the backlash free operating region.

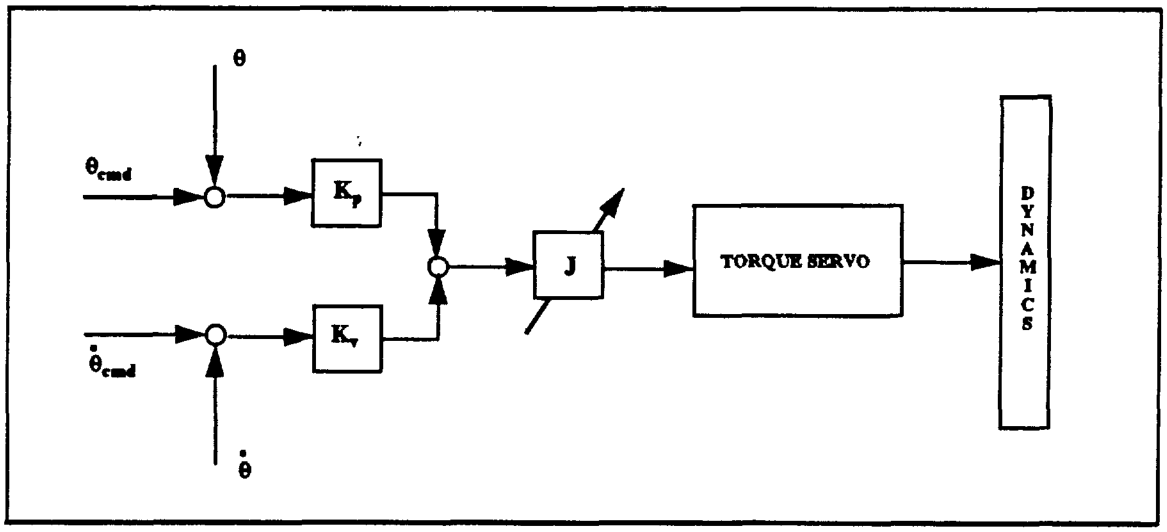
3.2.3.4 Linear Position/Rate Feedback Control

Linear position/rate feedback control drives a joint under closed loop torque control along the desired motion trajectory. This control law is implemented as

$$\tau_{fb} = K_p \varepsilon + K_v \dot{\varepsilon} \ , \tag{33}$$

where τ_{fb} is a joint torque command vector, ε is a joint position error vector, and K_p and K_v are constant diagonal feedback gain matrices. Joint velocity commands are needed to calculate joint velocity errors. The endpoint control algorithm generates velocity commands using first back differences. Since the endpoint algorithm and position/rate servo algorithm execute synchronously, the differentiation is exact and does not introduce disturbances such as sawtooth waveforms into the control loop. Figure 17 shows a block diagram of the linear position/rate feedback control loop for a single joint. Note that the joint inertia parameterizes the feedback gains, which helps to maintain configuration-independent response throughout the manipulator workspace. This parameterization is discussed further in Section 3.2.3.5.

Figure 17 Position/Rate Feedback Loop



Linear position/rate feedback compensator design begins with the closed torque loop system. The response from torque input to position output was shown in Figure 15. The plant has 180 degrees of phase lag at the desired 5 Hz. crossover frequency, which requires lead compensation. Simple PD feedback will provide this compensation. Then the gain may be adjusted to achieve the proper crossover frequency.

Figure 18 shows the compensated plant response. The crossover frequency is 30 rad/s. The Nichols plot in Figure 19 shows a 25 dB gain margin and 70 degree phase margin. Figure 20 shows the position response to an external torque input, which is a measure of the servo stiffness. Lower low frequency gain implies higher stiffness. With this compensation, the position response will have finite steady state error to external torque disturbances, such as gravity. Figure 21 and Figure 22 show the closed loop velocity and position responses. Finally, Figure 23 shows the position step response.

Figure 18 Plant with PD Position Loop Compensation

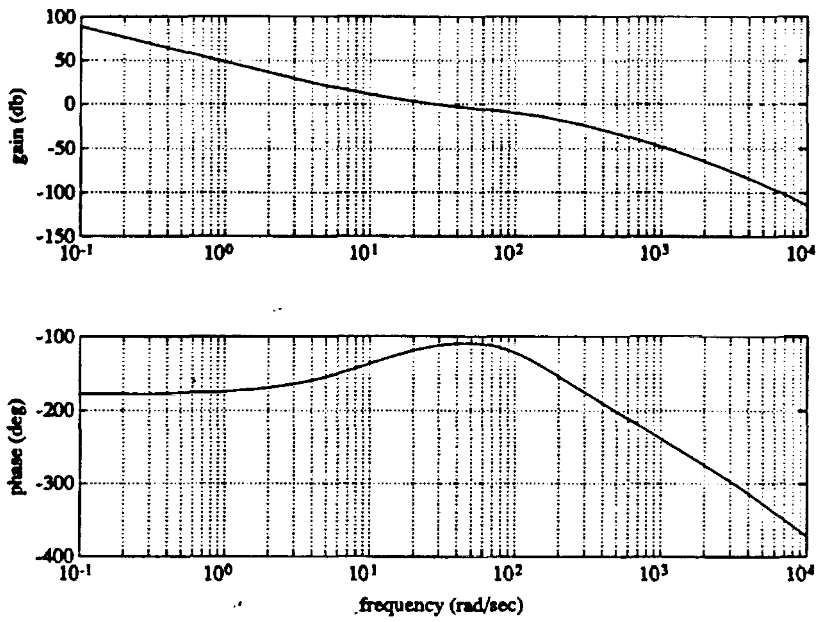


Figure 19 **Nichols Plot, Compensated Plant**

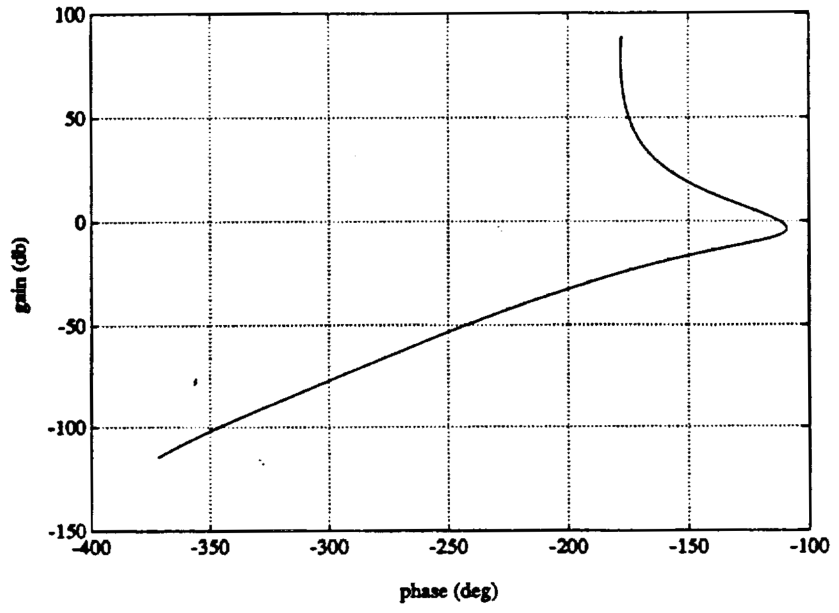


Figure 20 **Position Response to External Torque**

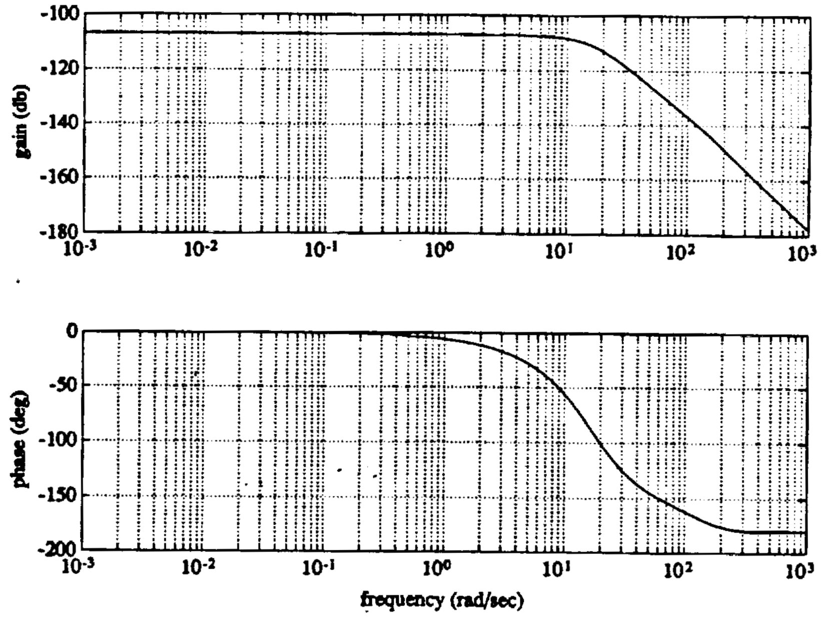


Figure 21 Closed Loop Velocity Response

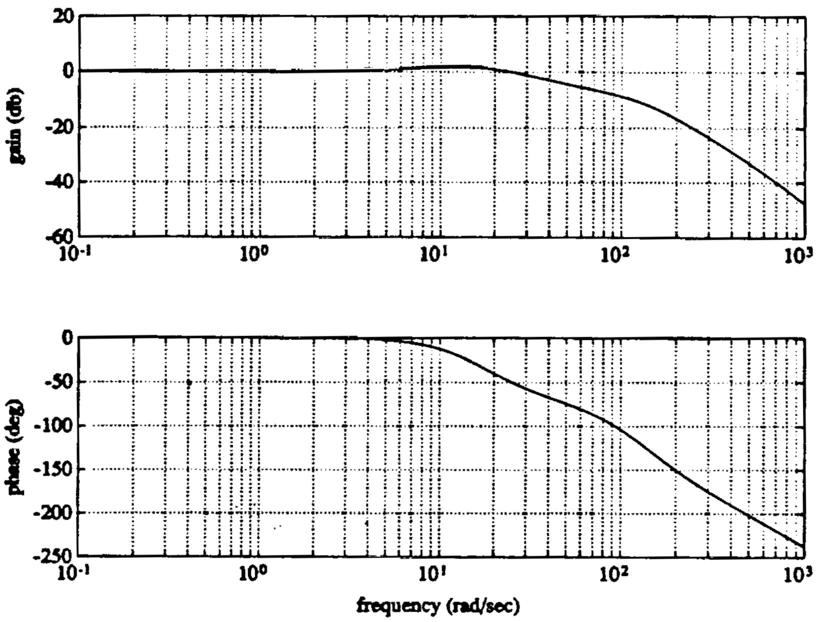


Figure 22 Closed Loop Position Response

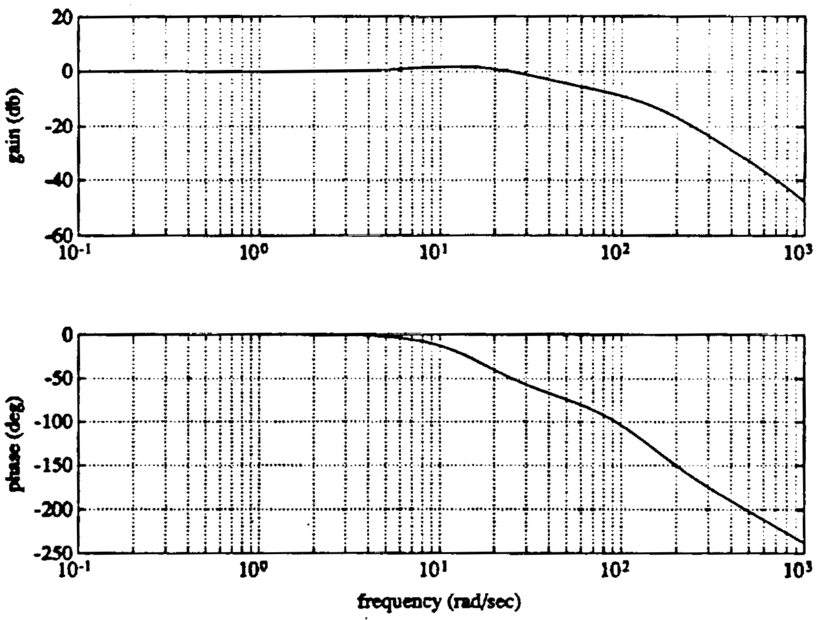
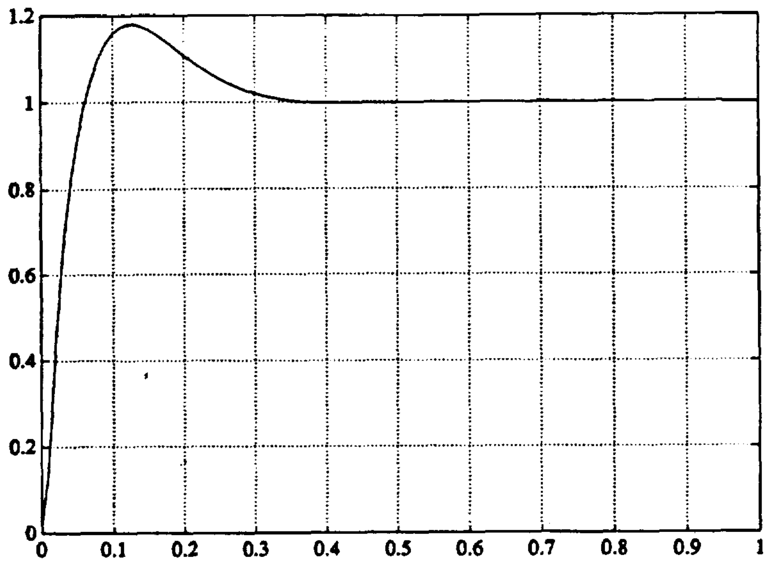


Figure 23 Position Step Response



Integral compensation will improve the joint's steady state response by nulling errors due to gravity disturbances and joint friction, with some penalty in phase loss and reduced stability margins. Figure 24 shows the PID-compensated plant response. Figure 25 shows the corresponding position response to external torque input. Finally, Figure 26 shows the closed loop position response with PID compensation.

Figure 24 Plant with PID Position Loop Compensation

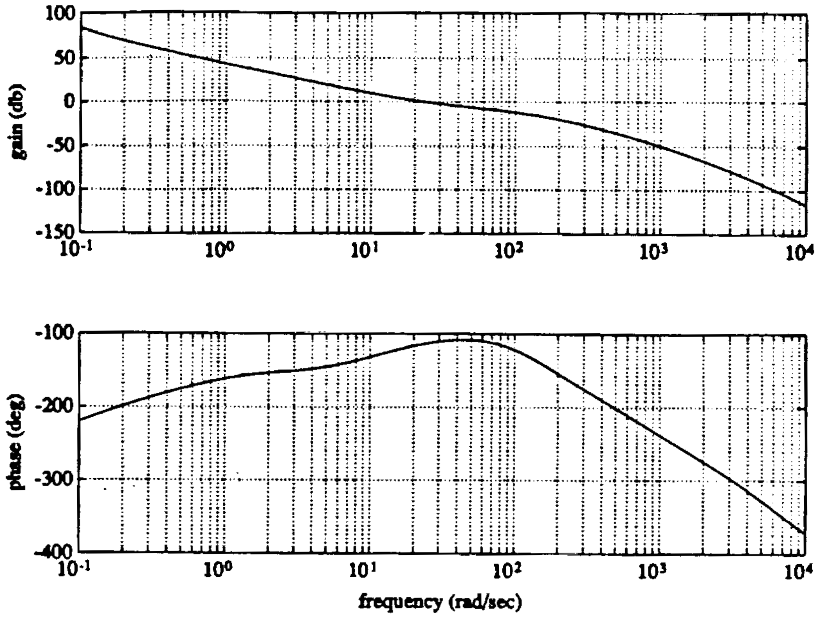


Figure 25 Position Response to External Torque with PID Comp

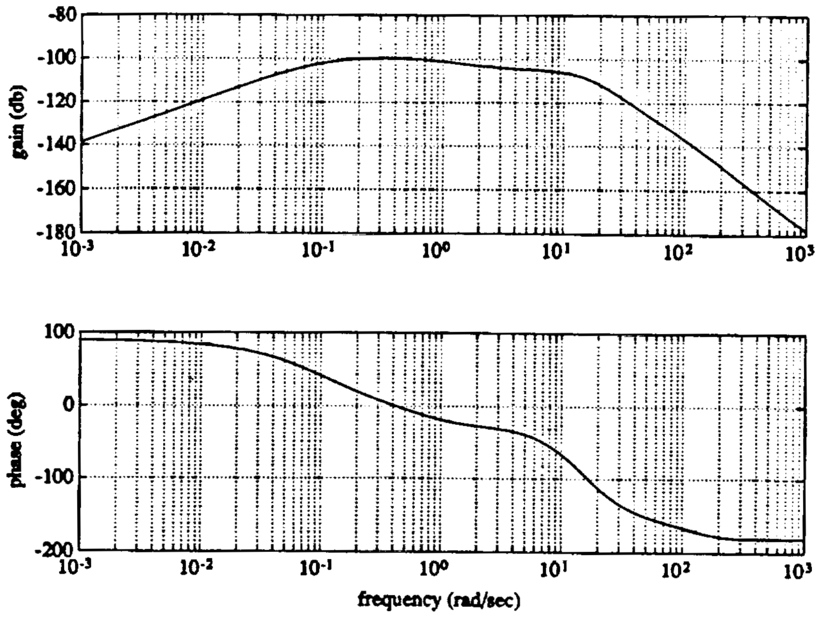
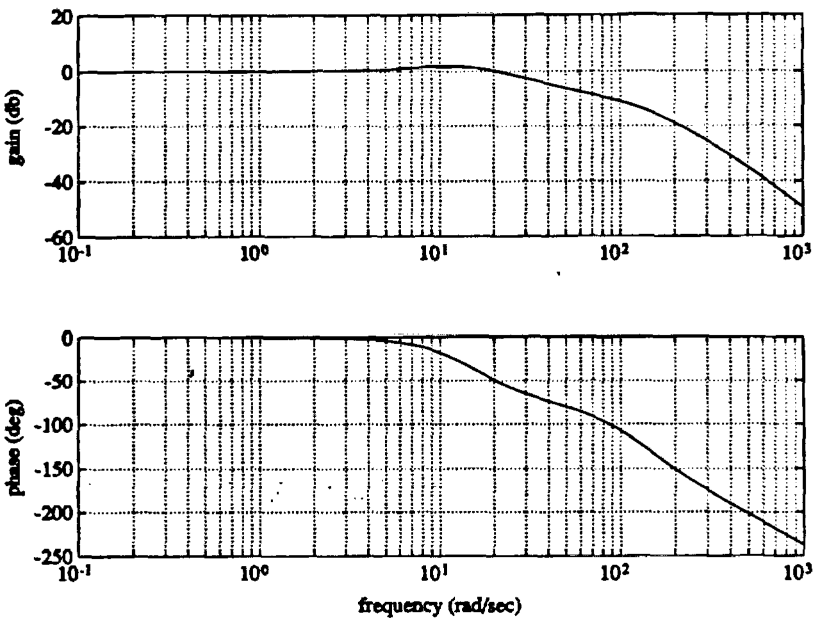


Figure 26 Closed Loop Position Response with PID Comp



3.2.3.5 Model Based Compensation

Linear feedback control (individual joint position and velocity feedback) works well to the extent that the manipulator dynamics are approximately linear, decoupled, and time-invariant. Of course, none of these ideal characteristics are true. Joint friction is a significant nonlinear effect. Properly operating torque servo loops reduce this effect to some degree. Since the joint reductions are large, dynamic coupling effects are relatively insignificant. However, some of the effective joint inertias undergo substantial variations with manipulator configuration and payload. These variations are most significant at the shoulder, and become negligible towards the wrist. This observation indicates that, while a model-based decoupling control law may be unnecessary, "effective joint inertia" control [4] could be useful. Effective joint inertia control is implemented as

$$\tau_{dec} = M(\theta_d)\,[K_p \varepsilon + K_v \dot{\varepsilon}] \,, \qquad (34)$$

where M is an estimate of the manipulator inertia matrix, obtained from the manipulator equations of motion. Using the full inertia matrix in Equation (34) provides decoupling control, while using only the diagonal elements provides effective joint inertia control.

Model-based feedforward compensation provides a means to generate open loop torque commands that move the payload through the desired trajectory while compensating for gravity and velocity dependent disturbance torques. The open loop feedforward commands provide most of the torque necessary to move the joints along the commanded trajectories. The feedback component of the torques serves to correct any errors due to modelling errors. Good joint tracking can thus be achieved with reduced feedback loop bandwidth, which provides better stability margins and reduced noise sensitivity. The model-based feedforward compensation is implemented as

$$\tau_{ff} = M(\theta_d)\ddot{\theta}_d + C(\theta_d, \dot{\theta}_d)\dot{\theta}_d + G(\theta_d) \quad . \tag{35}$$

M is the manipulator inertia matrix estimate, C is the centripetal and coriolis term matrix estimate, and G is the gravity term vector estimate. Since most commanded trajectories involve relatively low joint velocities, the C term is dropped to simplify the calculations for real time implementation.

3.2.3.6 Adaptive Control

Model-based approaches suffer from several well-known shortcomings. Model parameters are often inaccurate, and unmodeled portions of the system dynamics may have a significant effect. The computational cost of implementing model-based compensation is high. A potential alternative is an adaptive controller, and in particular, a performance-based controller that does not use a complex system model. A large body of research and literature on adaptive controllers exists, and it includes many implementation studies and evaluations. In particular, Seraji's work at the Jet Propulsion Laboratory [5] has some attractive properties. The algorithm does not use a complex dynamic manipulator model - in fact, its computational burden is minimal. The method uses indirect adaptation based on tracking performance; there is no parameter estimation, which again greatly simplifies the implementation. Finally, the algorithm has been tested experimentally and shown to improve manipulator performance in that particular case.

A detailed development of the algorithm can be found in [5]. The control and adaptation laws are shown here without derivation. The control law is

$$T_i(t) = f_i(t) + [k_{i0}(t)e_i(t) + k_{i1}(t)\dot{e}_i(t)] + [q_{i0}(t)\theta_{di} + q_{i1}(t)\dot{\theta}_{di} + q_{i2}(t)\ddot{\theta}_{di}] \quad . \tag{36}$$

The control law's first term is a fixed gain "auxiliary signal" which improves tracking performance. The second term is the adaptive feedback component of the control law, and the third term is the adaptive feedforward component. The feedback gain adaptation laws are

$$
\begin{aligned}
r_i(t) &= w_{pi}e_i(t) + w_{vi}\dot{e}_i(t) \\
f_i(t) &= f_i(0) + \delta_i\int_0^t r_i(t)dt + \rho_i r_i(t) \\
k_{i0}(t) &= k_{i0}(0) + \alpha_{i0}\int_0^t r_i(t)e_i(t)dt + \beta_{i0}r_i(t)e_i(t) \\
k_{i1}(t) &= k_{i1}(0) + \alpha_{i1}\int_0^t r_i(t)\dot{e}_i(t)dt + \beta_{i1}r_i(t)\dot{e}_i(t)
\end{aligned}
\tag{37}
$$

and the feedforward gain adaptation laws are

$$
\begin{aligned}
q_{i0}(t) &= q_{i0}(0) + \gamma_{i0}\int_0^t r_i(t)\theta_{di}(t)dt + \lambda_{i0}r_i(t)\theta_{di}(t) \\
q_{i1}(t) &= q_{i1}(0) + \gamma_{i1}\int_0^t r_i(t)\dot{\theta}_{di}(t)dt + \lambda_{i1}r_i(t)\dot{\theta}_{di}(t) \\
q_{i2}(t) &= q_{i2}(0) + \gamma_{i2}\int_0^t r_i(t)\ddot{\theta}_{di}(t)dt + \lambda_{i2}r_i(t)\ddot{\theta}_{di}(t)
\end{aligned}
\tag{38}
$$

Testing through simulation could indicate whether or not the adaptive control approach was likely to provide performance to the linear feedback/nonlinear feedforward approach. Section 3.3.4 discusses testing results.

3.3 Simulation

Simulation is a vital part of a complex control system development. The Odetics Dexterous Manipulator's hierarchical control system design lends itself to simulation of independent components followed by complete dynamic simulation of various components together. During the control system development, simulations verify that control designs are reasonably accurate and that modelled behavior is as anticipated. Of course, simulations cannot illustrate unmodeled behavior. There is a continual trade-off between the degree of detail desired in the system model and the cost of adding and simulating more detail.

Factors and behavior deemed most important to include in the dynamic simulation include the following:

- a rigid body model of the full seven degree of freedom manipulator
- the endpoint control algorithm
- joint position/rate servo loops
- model-based compensation algorithms, and the effects of modelling error
- adaptive control
- quantization effects, including multiple sampling rates
- actuator torque/speed and power requirements for various motions.

The most significant item missing from this list is flexible body modelling. Flexible body modelling of the manipulator was determined to be highly difficult and costly, while analysis performed during the manipulator mechanical design indicated that the manipulator would have high stiffness. In light of these two factors, it made sense to leave out flexible body modelling and simulation. The joint torque servos are not simulated either, principally because the actual hardware would be available well before the simulation could be completed.

3.3.1 Manipulator Modelling

The well-known rigid body manipulator equations of motion are

$$M'(\theta)\ddot{\theta}+C(\theta,\dot{\theta})\dot{\theta}+G(\theta) = \tau \ , \tag{39}$$

where $M(\theta)$ is the manipulator inertia matrix, $C(\theta, \dot{\theta})$ is the centripetal and coriolis term matrix, $G(\theta)$ is the gravity term vector, and τ is an applied torque vector. These equations form the core of the manipulator dynamic model. Generating these equations manually is a very costly task. Fortunately, software is available to generate and solve Equation (39) automatically. The program SD/FAST [6] takes kinematic and mass properties information for a general mechanical linkage and generates Fortran subroutines that calculate and solve Equation (39). The dynamic simulation passes the subroutines a set of joint angles θ and joint torques τ. The subroutines solve the equations of motion and return the joint angle accelerations $\ddot{\theta}$, which the simulation integrates to obtain the joint velocities and positions. SDFAST has other useful analysis capabilities that are described in the user manual.

The manipulator's kinematic description is easily generated from engineering drawings. Mass properties information is estimated from CAD modelling and engineering data.

3.3.2 Simulation Code

Rather than developing and debugging a complete dynamic simulation program manually, the simulation program ACSL [7] was used to build up the simulation. Various code modules, such as those containing the endpoint control algorithm or the equations of motion, are linked with the ACSL simulation code so that functions within these modules may be called from the simulation. This method enables the simulation to use the same code that executes critical functions in the real time embedded system. Conceptually, the simulation functions are quite simple: generate the joint torques to be applied to the manipulator (via control algorithms, disturbances, etc.), call the manipulator model subroutines (equations of motion), and integrate the resulting joint accelerations to obtain joint velocities and positions for the next simulation time step. Of course, the details of performing these tasks become quite complex. ACSL provides many features to simplify the details, such as built-in functions for modelling quantization, implementing digital filters, and including noise effects.

The full dynamic simulation provides results in the form of numerical data which can be plotted. While this data is useful for engineering analysis, it does not provide much intuitive feel for the manipulator motion during simulation runs. A simple animation program that uses the Silicon Graphics Personal Iris platform was developed to provide a better way to observe the simulated motion. The program reads a set of joint angles from a simulation run and displays a graphical manipulator model fast enough so that the motion appears continuous.

3.3.3 Manipulator Inertial Properties

One of the preliminary analysis results is the manipulator mass matrix, which is obtained from the SDFAST subroutines. The Odetics Dexterous Manipulator uses high gear reductions, so effective joint inertia variations are largely masked by reflected drivetrain inertia. However, there are some significant inertia variations for the first few joints. Two particular cases illustrate the variations. The first case is the manipulator fully outstretched and carrying the maximum payload (50 lb., on-axis). The inertia matrix is

$$\begin{bmatrix} 793.1 & & & & & & \\ -1.0 & 760.9 & & & & & \\ -2.9 & 0.0 & 104.0 & & & & \\ 5.8 & 330.8 & 0.0 & 281.7 & & & \\ 0.0 & 98.4 & 0.0 & 54.9 & 36.0 & & \\ 56.3 & 0.0 & 0.0 & 0.0 & 0.0 & 23.9 & \\ 0.0 & 0.0 & 0.3 & 0.0 & 0.0 & 0.0 & 17.3 \end{bmatrix} \text{in-lb-s}^2.$$

For the second case, the manipulator is close to the stowed position, and carries no payload. The corresponding inertia matrix is

$$\begin{bmatrix} 127.2 & & & & & & \\ -25.0 & 155.9 & & & & & \\ 14.0 & 0.0 & 116.1 & & & & \\ 18.4 & 3.6 & 0.0 & 155.7 & & & \\ 2.9 & 1.8 & 0.0 & 9.9 & 19.8 & & \\ -1.59 & 0.0 & -1.59 & 0.0 & 0.0 & 17.6 & \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 16.9 \end{bmatrix} \text{in-lb-s}^2.$$

The effective inertia at the shoulder azimuth varies by a factor of just over 6. For the shoulder elevation, the variation is nearly a factor of 5. For the remaining joints, the variations are considerably less. It is also important to note that these variations represent the maximum possible. Typically, variations occurring during manipulator motion will be less.

Another characteristic that can be determined from the manipulator inertia matrix is the degree of dynamic coupling between joints. In the first example, there is substantial dynamic coupling between the shoulder elevation joint and the elbow. In the second example, there is a lesser degree of coupling between the shoulder azimuth and shoulder elevation. In general, substantial coupling occurs mainly at high payloads, where the acceleration capability of the manipulator is limited.

3.3.4 Dynamic Simulation Results

Many simulation runs under a multitude of conditions were run during the control system development. This section contains only a few sample cases that illustrate how the simulation was used to verify control system performance and examine dynamic behavior.

One of the more important determinations from simulation is the model-based feedforward compensation performance. The manipulator, carrying an 8 lb. payload, was placed in a nominal configuration well within the workspace and driven with a rapid sinusoidal endpoint trajectory. Endpoint speed in the vertical direction reached 40 inches per second. This scenario is probably near the maximum demand that would be placed on the manipulator. Plots of the endpoint position error indicate manipulator tracking performance. In the first run, the joint servos use PD shaping with full feedforward (including velocity dependent terms) and decoupling compensation. Figure 27 shows that the maximum position errors are approximately 0.05 in. in the X and Y directions and 1.0 in. in the Z direction. The next run uses similar compensation, except that the velocity dependent terms in the feedforward compensation are zeroed. Figure 28 shows that the X and Y tracking errors have nearly doubled and the Z tracking error is about the same. In the third run, decoupling compensation is turned off. Endpoint position tracking errors are about the same as in the previous run, as shown in Figure 29.

Figure 27 **Simulation Run 1**

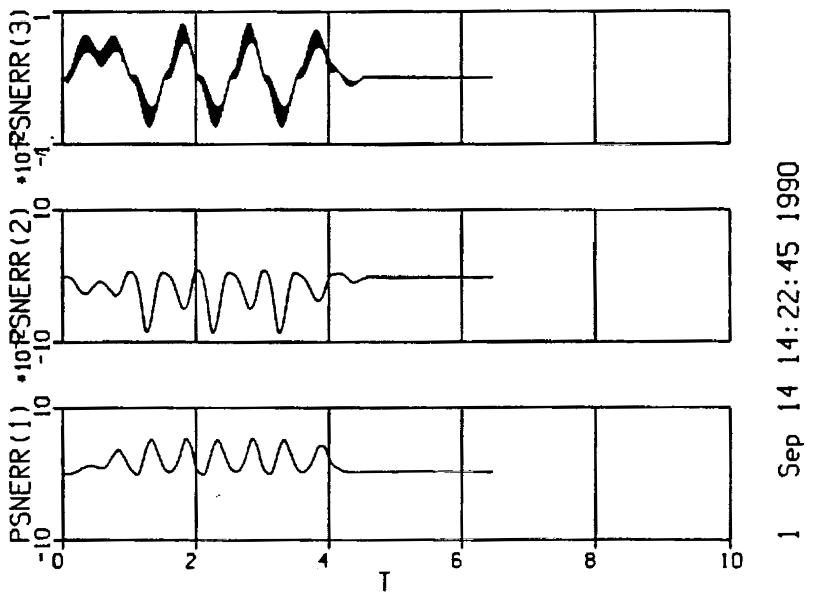


Figure 28 **Simulation Run 2**

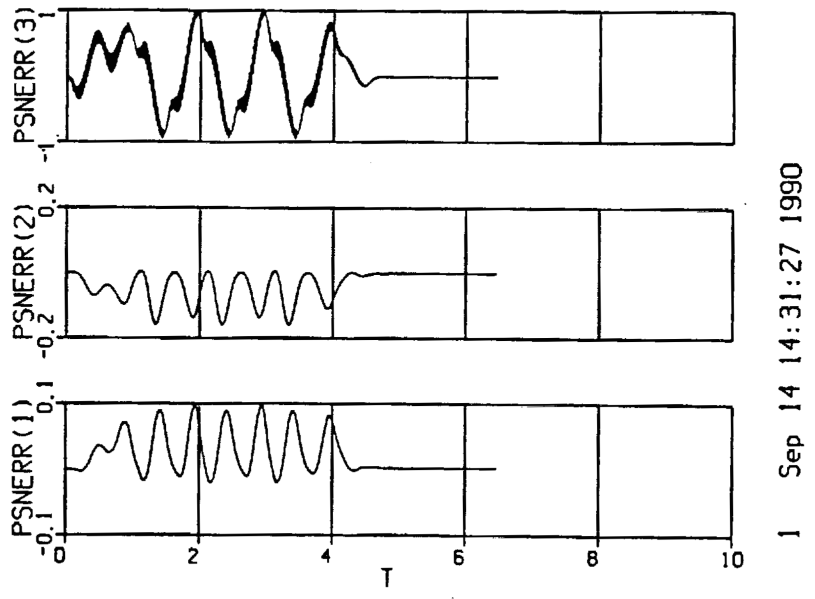
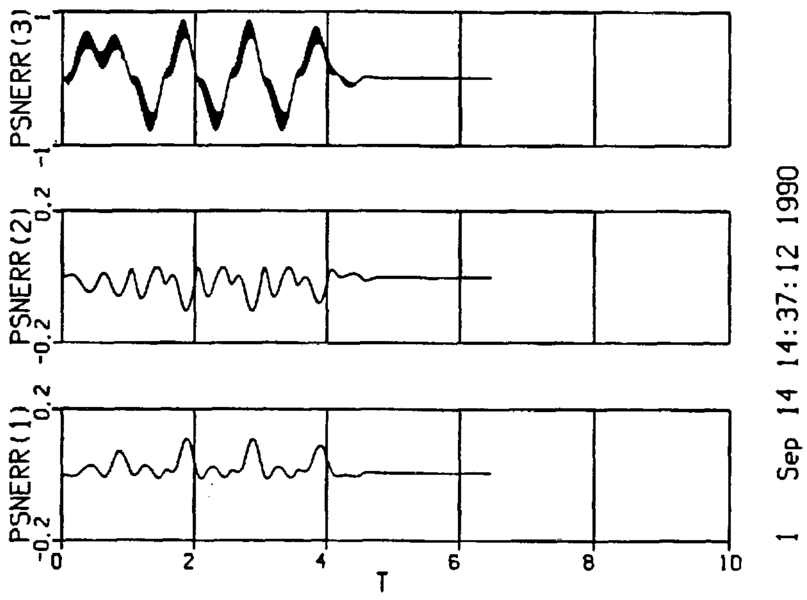


Figure 29 Simulation Run 3



In less demanding simulations, the model-based compensation had corresponding lesser effect on dynamic (transient) tracking. The results indicate that decoupling compensation will probably not be required for most motions. In contrast, the gravity compensating terms in the feedforward compensation greatly reduce steady-state gravity errors, and should be included.

Another area to be examined with simulation was the adaptive control algorithm's performance with this manipulator. The discrete version of the algorithm found in [5] was coded into the simulation. As in the PD algorithm case, the sample rate is 50 Hz., and the desired closed loop position bandwidth is approximately 5 Hz.

When the various weights are tuned properly, the algorithm works correctly and stably. Overall, good steady-state response is easy to achieve. Good transient response is more difficult. One key factor is the magnitude of the auxiliary signal. As the auxiliary signal gains δ and ρ are increased, the magnitude of the auxiliary signal becomes much greater than that of the adaptive feedback signal, and the algorithm approaches fixed gain PID feedback. The adaptive feedback gains are still able to compensate for the effective inertia variations in the system, and the large (fixed) feedback gains provide good transient response. Without large auxiliary signal gains, fast and stable transient response could not be achieved.

If an integral component is included in the auxiliary signal $(\delta > 0)$, gravity torque error is quickly eliminated. Without an integral component in the auxiliary signal, the adaptive feedback gains will rise exponentially in the steady-state to try and null this error. Large values of the adaptation gains w_p and w_v will speed up the adaptation, but tend to drive the algorithm unstable.

Three simulation runs illustrate the behavior described above. For each run, the endpoint trajectory is an orientation motion with fixed position. Data is shown for the first four manipulator joints only. Joints 2, 3, and 4 are under gravity load, while joint 1 is not.

In the first run, the adaptive control algorithm parameters are:

$$W_p = 1000$$
$$W_v = 500$$
$$\delta = \rho = 3, 3, 3, 3, 0.3, 0.3, 0.3$$
$$\alpha_p = 50$$
$$\alpha_v = 2.5$$
$$\beta_p = 10$$
$$\beta_v = 25$$

Figure 30 shows the joint angle commands and joint angles plotted together. Note the transient response shows large errors. Figure 31 and Figure 32 show the adaptive position and velocity feedback gains. The joints that are gravity-loaded show relatively long convergence times for these gains.

Figure 30 Adaptive Control, Run 1; Joint Tracking

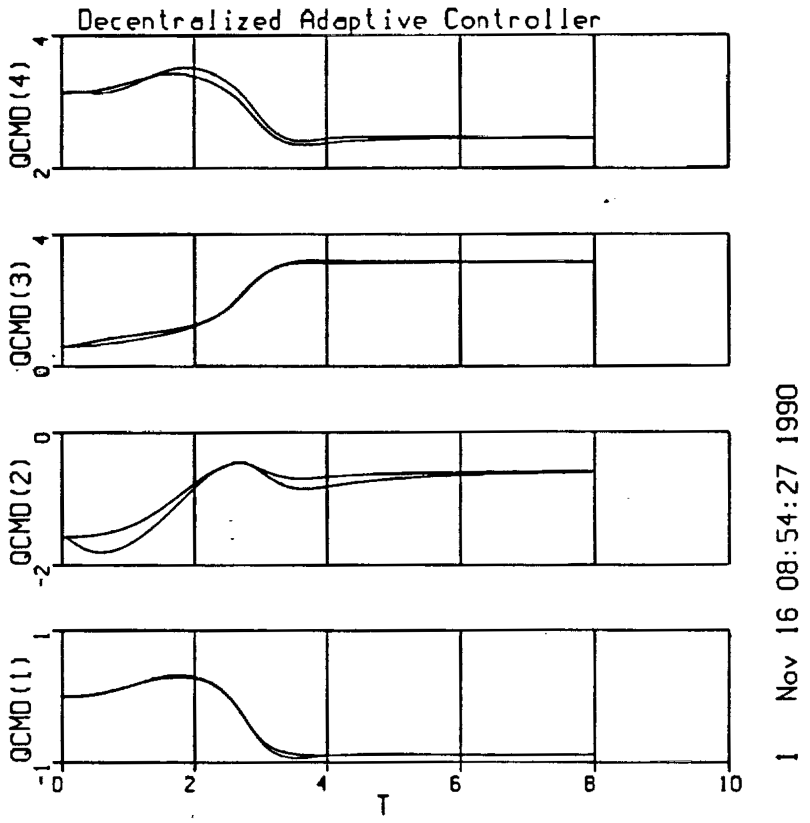


Figure 31 Adaptive Control, Run 1; Position Error Gains

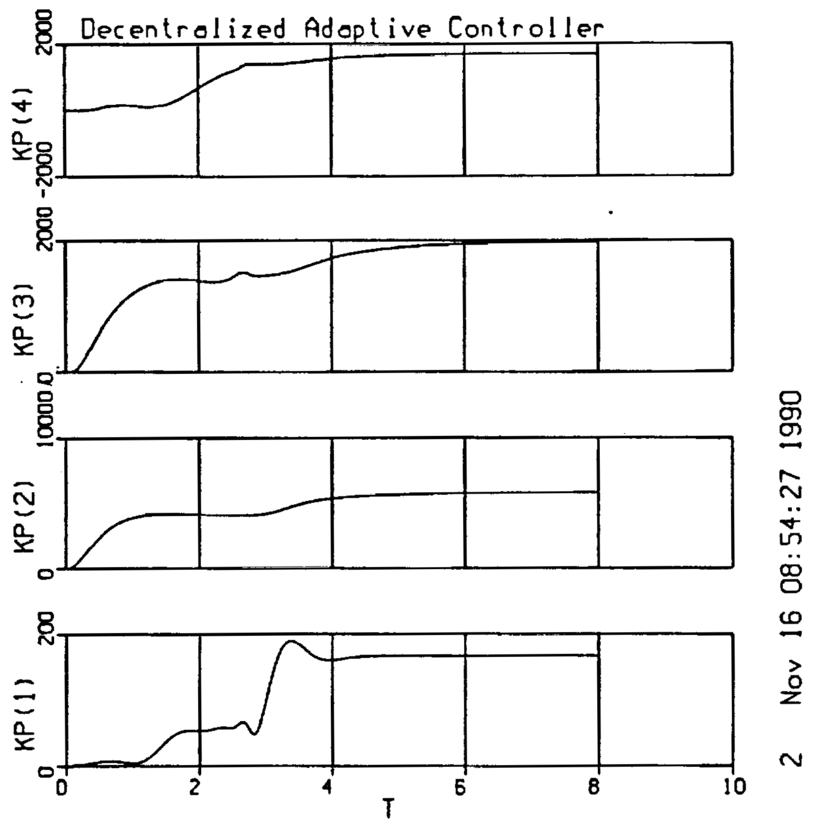
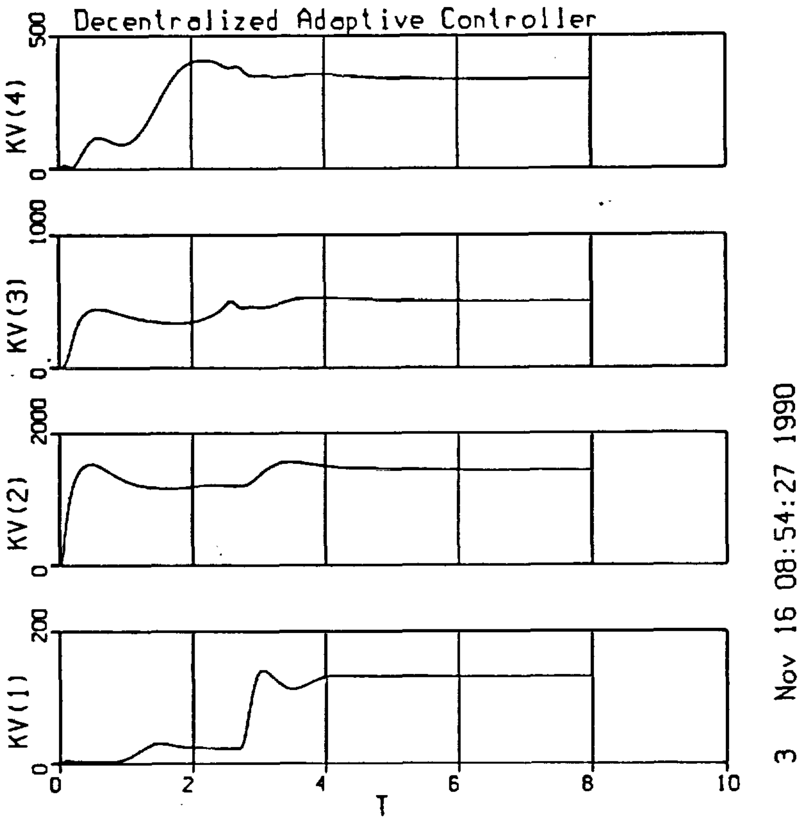


Figure 32 Adaptive Control, Run 1; Velocity Error Gains



In the second run, the auxiliary signal gains have been increased and the error weights decreased:

$$W_p = 200$$

$$W_v = 5$$

$$\delta = \rho = 1000, 1000, 1000, 1000, 100, 100, 100$$

The other gains remain the same. The transient and steady-state response are much improved - the maximum tracking error is 0.13 rad. Figure 34 and Figure 35 show that the magnitudes of K_p and K_v remain small, and therefore the adaptive component of the torque command, remain small. The torque command is almost totally composed of the auxiliary signal contribution, making the algorithm essentially fixed gain PID.

Figure 33 Adaptive Control, Run 2; Joint Tracking

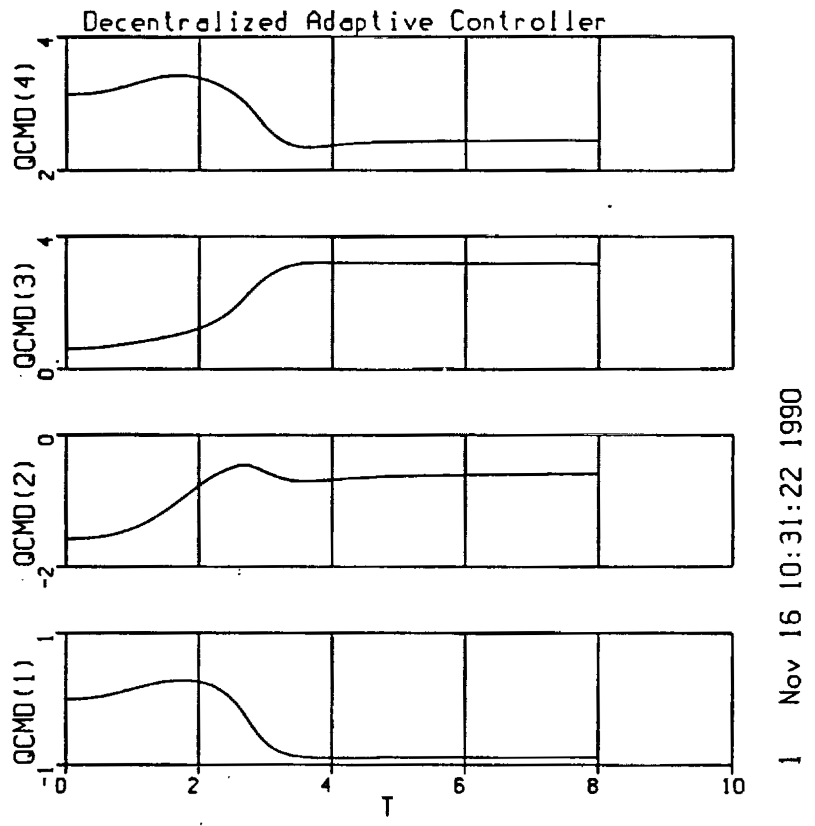


Figure 34 Adaptive Control, Run 2; Position Error Gains

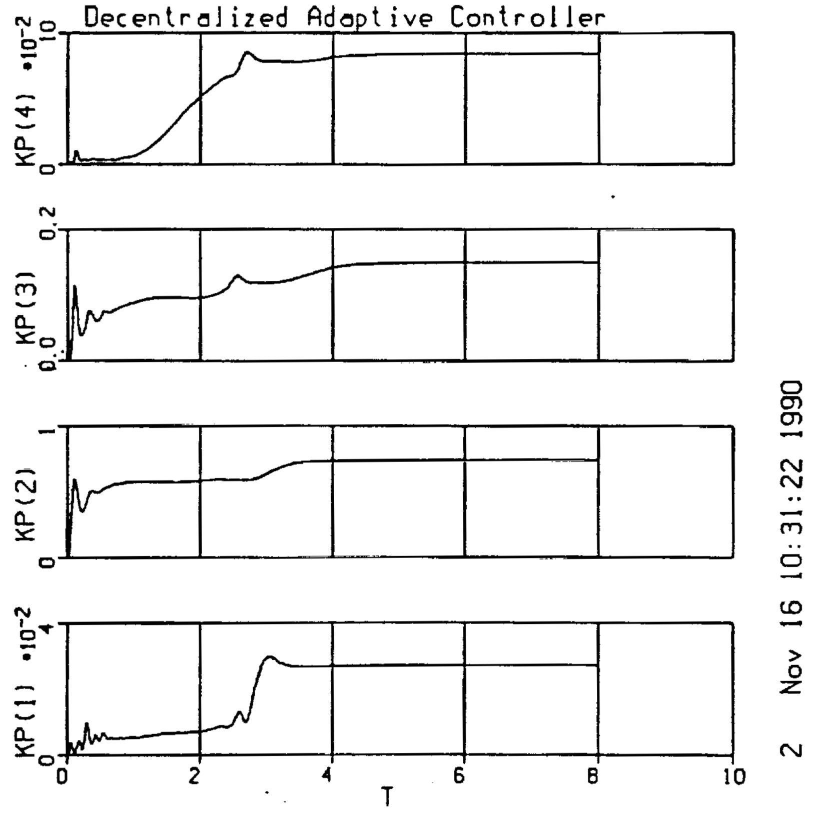
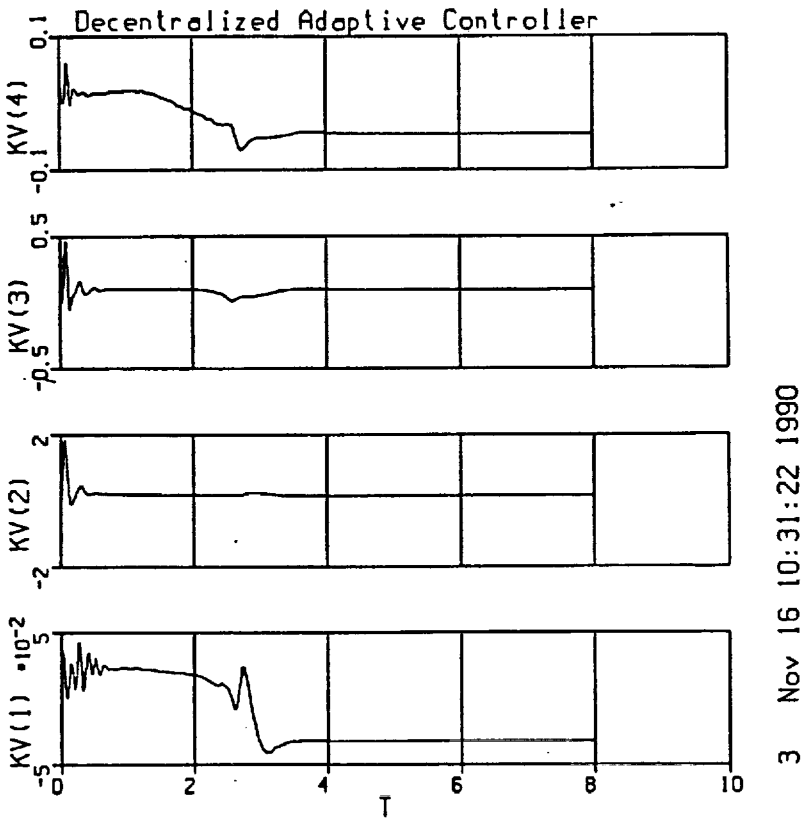
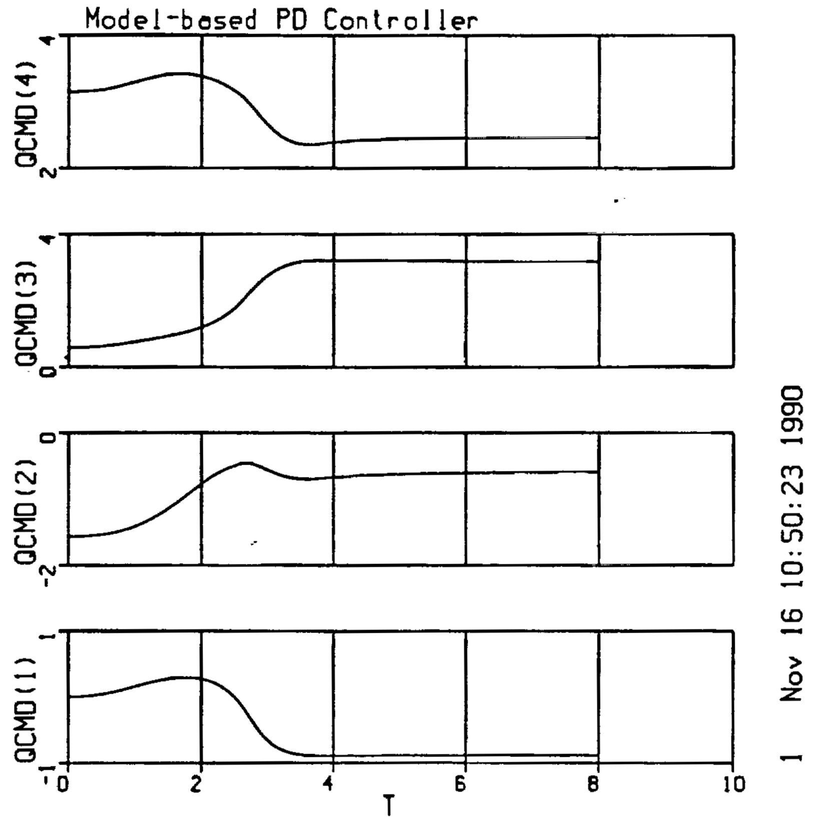


Figure 35 Adaptive Control, Run 2; Velocity Error Gains



The last run shows the same motion, using a model-based PD controller. An error of 50% has purposely been introduced into the mass properties of the payload. Figure 36 shows the The maximum transient error is 0.005 rad, on joint 3. Joint 2 has a steady-state gravity error of 0.001 rad.

Figure 36 Same Motion, Model-Based PD Control; Joint Tracking



4.0 Control System Implementation

Control system implementation involves providing computer and electronics hardware with sufficient performance to execute control algorithms, and developing software to implement those algorithms in a correct, efficient, and maintainable manner. As with any engineering design, the implementation is a compromise of performance, cost, flexibility, and other attributes. For the Odetics Dexterous Manipulator control system implementation, several of important trade-off considerations were:

- Use commercially available (rather than custom) processors, data acquisition, and control hardware. Although custom hardware has advantages in size and reduced number of interfaces, the cost, development time, and difficulty in making configuration changes are major disadvantages for a prototype system.
- Use the VME bus and Motorola 680x0 family processors. The wealth of vendors that provide hardware for VME systems increases design options and reduces risk. Similarly, there is a friendly software development environment with many software development tools (compilers, real time operating systems, etc.) available for the 680x0 architecture. Although other processor architectures such as SPARC and DSP's may provide higher performance, they do not have the 680x0 family's rich set of development tools.
- Adopt a host/target strategy for real time code development and implementation. In this strategy, software development takes place on a host computer rather than on the embedded control computer. Ideally, the host computer has very good facilities for software development, debugging, testing, and documentation. Working code is then loaded into the embedded control computer, known as the "target", for actual system control. The target is optimized for real time processing, and typically has rather poor facilities for software development. With the host/target strategy, both development and execution take place in a near optimal environment, rather than in some compromised environment that works sub-optimally.

4.1 Hardware

The control system hardware is a multiprocessor computer system designed for mechanical system control. At the heart of the system is the VME bus, which provides high bandwidth communications, shared memory, and a large set of readily available hardware and software tools for system development. Figure 37 shows the physical layout. A VME cardcage with a 22 slot backplane holds the various processor, data acquisition, and I/O boards. The cardcage is rack mounted in a large cabinet, which also holds power supplies and an interface electronics enclosure. A second cabinet contains servo amplifiers. Both cabinets are connected to the manipulator via an umbilical cable. An emergency "E-stop" button is mounted in the cabinet containing the card cage, and there are additional connectors for connecting remote E-stop buttons to the system. A person entering the manipulator workspace while the manipulator is active carries one of the remote buttons with him so that he can quickly disable the manipulator if required.

Figure 37 Control System Physical Layout

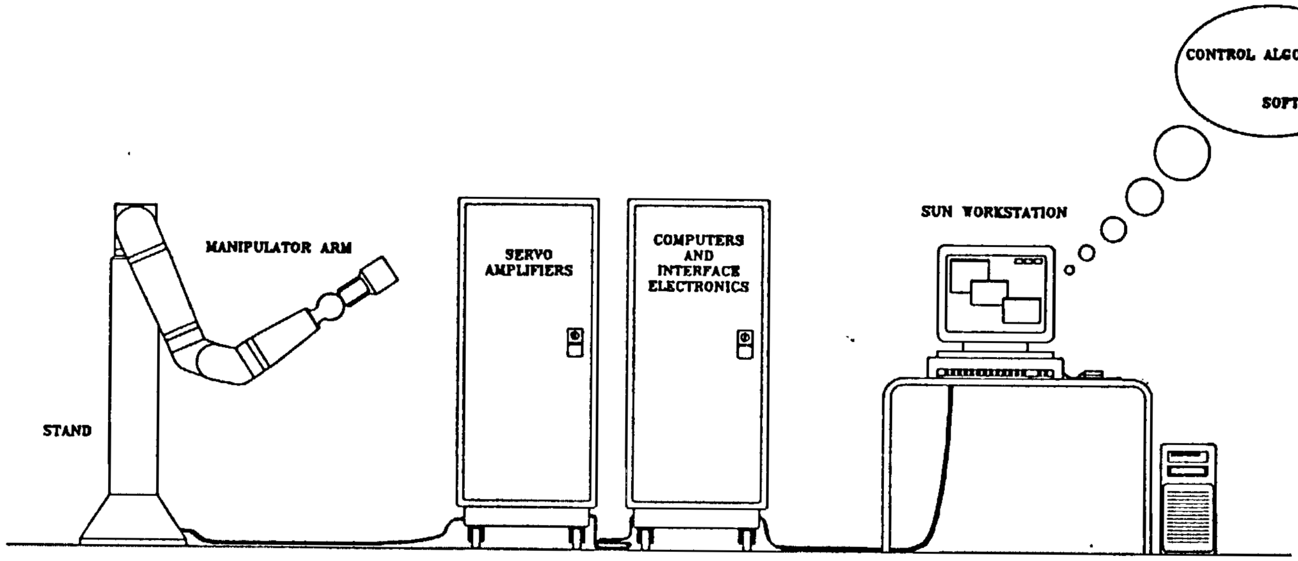
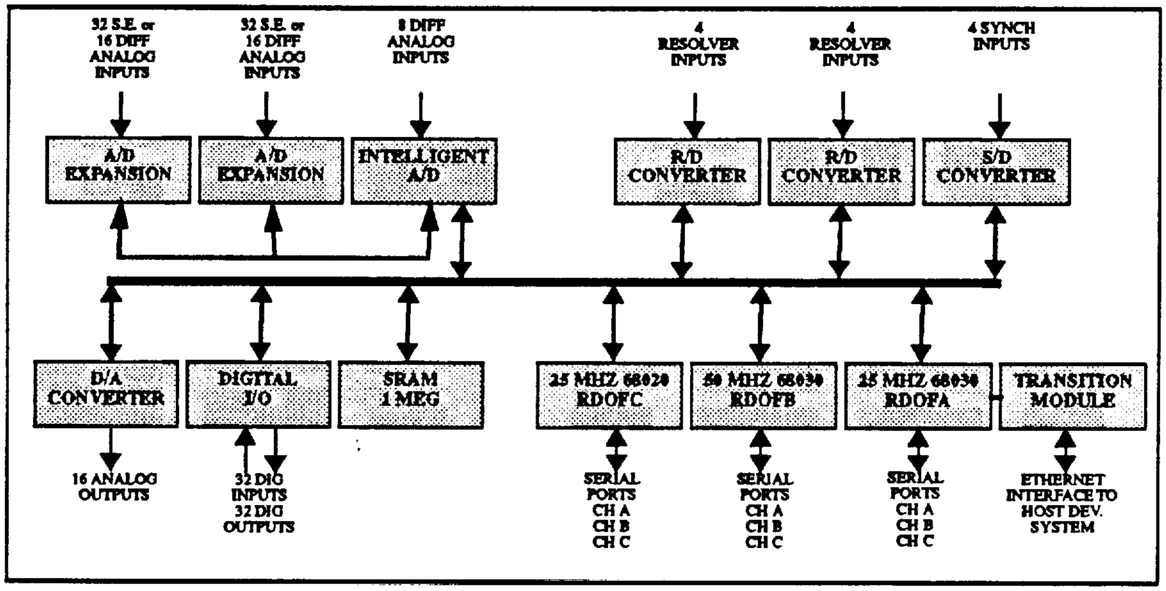


Figure 38 shows the control hardware architecture in detail. Three single board computers execute algorithms, control hardware devices, and perform executive functions. There is an additional 1 Meg of battery backed SRAM for programs and data. A programmable (68010-based) A/D converter with expansion units processes analog signals. Resolver to digital and Synchro to digital converters handle joint position sensor signals. Digital I/O is available for joint brakes as well as enable, fault, and reset operations. A D/A converter provides the analog command signals for the servo amplifiers. The Transition Module provides connectors for Ethernet and serial interfaces used to connect the target system to other systems sharing the same network.

Figure 38 Control Hardware Architecture



4.1.1 Processors

Each of the three system processors is a 680x0 family single board computer that is dedicated to serving at one of the three control hierarchy levels. The first processor, known as RDOFA, is a Motorola MV-147 single board computer. It uses a 25 MHz MC68030 CPU with a 25 MHz MC68882 floating point coprocessor. It also includes an Ethernet transceiver interface for communications with other computers on a network, as well as serial ports. This processor primarily serves as the system executive. It provides the interface to the network for loading code and communicating with the host computer, which runs the user interface.

The second processor, referred to as RDOFB, is a Synergy Microsystems SV31S single board computer. This is a very high performance computer that uses a 50 MHz MC68030 CPU with a 50 MHz MC68882 floating point coprocessor. RDOFB performs all algorithm calculations at the middle level of the control hierarchy. These include trajectory generation, endpoint control, joint position/rate servo control, and model based feedforward compensation. In addition to algorithm calculations, RDOFB monitors sensor values and internal variables and disables the manipulator if they leave safe ranges.

The third processor, referred to as RDOFC, is a Motorola MV-133XT single board computer that uses a 25 MHz MC68020 CPU with a 25 MHz 68881 floating point coprocessor. RDOFC's primary function is to execute the joint torque servo loop algorithm. It also generates a VME interrupt every 20 milliseconds for RDOFB execution timing.

4.1.2 Data Acquisition and Interface Electronics

Data conversion devices in the control system include an A/D converter, resolver to digital (R/D) and synchro to digital (S/D) converters, and discrete input/output. Each of these is a board or board set that resides in the VME card cage. An analog output board and the discrete outputs provide hardware control signals. A separate enclosure houses the interface and signal conditioning electronics.

The A/D converter is a Datel DVME-601 "smart" A/D board. It includes a 68010 processor to control A/D conversion, relieving the host processor of that task. The board features 16 single ended inputs with 12 bit conversions down to 2 μs. The board includes numerous other features, including an on-board timer and VME bus interrupt capability that are used to synchronize algorithm execution. Two Datel DVME-641 expansion boards provide the additional input channels required for the complete manipulator system. The 641 boards interface to the 601 board via a channel expansion bus that is separate from the VME bus.

The R/D and S/D converters are Transmagnetics 5410C-8-12 boards that include three separate 16 bit tracking converters. The boards provide 45 arc-second accuracy. While some R/D converters provide an analog velocity output, high accuracy units such as this one typically do not due to the difficulty in producing a usable signal with reasonable ripple at low velocities.

Discrete I/O is provided by a VME Microsystems VMIVME 2510B digital I/O board, a 64 channel model. The discrete inputs connect to the servo amplifier fault lines, which show conditions such as a shorted motor winding or AC power loss. The discrete outputs control the servo amplifier enables and the joint brakes. A VME Microsystems VMIVME 4100 16-channel D/A board provides analog commands for the servo amplifiers.

The interface electronics provides many functions and circuits:

- routing of signals between the manipulator, card cage, servo amplifier enclosure, and host computer
- reference voltages for various circuits, and level shifting as required for discrete signals
- anti-aliasing filtering of analog inputs, such as joint torque sensor signals

- latching for critical discrete signals, such as E-stops and amplifier faults
- indicator LEDs to display the system state
- brake driver circuitry.

Most of this circuitry is modular, e.g., individual boards exist for each brake driver, so that spares can be added quickly in the event of a failure. The design gives primary consideration to low cost implementation and simple debugging and modifications, and little consideration to compactness and interface minimization. This design approach is correct for a prototype system, and would certainly be modified for a production system.

4.2 Software

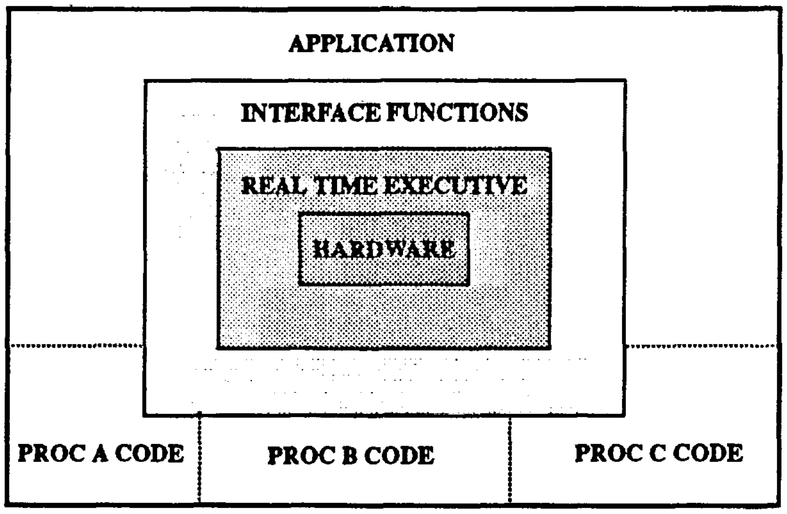
Various goals and constraints have guided the software design and implementation for the Dexterous Manipulator control system. Principal among these criteria are:

- provide good real time performance (high sample rates and determinacy)
- insure robustness to error conditions and failure modes; protect hardware and degrade in a controlled manner
- finish the project with structured, maintainable code that can be extended to test new algorithms and hardware as they become available
- maintain an efficient development environment, using available software tools to generate and maintain code when possible.

4.2.1 Architecture

The software architecture integrates the three level algorithm hierarchy described in Section 3.1 with a set of real time executive and hardware interface (driver) functions. Together, these components provide the embedded control software. The real time executive and interface functions may be thought of as the two innermost levels of the software architecture, while the application code forms the outer levels, as shown in Figure 39.

Figure 39 Software Architecture



The application code reflects the three level algorithm hierarchy. Functions and data for each level are linked into single modules, which are loaded and executed on the appropriate processor. Data used by more than one processor is shared over the VME bus, using dual ported RAM controlled by a double buffering scheme. Code running on processor A is primarily responsible for communication with the host workstation, which controls the user interface. The design uses UNIX Remote Procedure Calls to recognize events at the user interface and transfer the appropriate control signals and data from the host to the target system, where the target processors have access to the data via shared memory. This processor executes its code asynchronously. Processors B and C run synchronously, executing time-critical control and servo functions. Functions running on these processors use the lower level interface functions for data acquisition and I/O and the real time operating system services for synchronization and execution control.

4.2.2 Execution Control

Process execution in the control system can be divided into real time and non-real time processes. Real time processes include data acquisition, algorithm execution, and data logging, while non-real time processes include communication with the host computer, trajectory generation, and parameter modifications. A reasonable way to describe the system is to describe the three execution levels separately.

4.2.2.1 Asynchronous Processes

As described in the software architecture, processor RDOFA executes asynchronously, and is responsible for communications with the host computer and calculating the manipulator mass matrix. Communications with the host computer is implemented with Unix Remote Procedure Calls. It is a one-way path: as implemented, the RPC interface passes data from the host system to the target system, but not the other way. Currently, messages from the target system to the host are sent over a serial line and appear on the host computer in a window running a simple serial communication program. Information that can be passed to the target system includes motion parameters and commands, algorithm parameters, and feature enabling/disabling switches for debugging. The data structure that contains all of this information is a union of C structures, each of which contains structure elements for the various system parameters and commands. Once the RDOFA processor has obtained the pertinent data from the control station, it makes the necessary function calls to place the data into shared memory, where the other processors have access to it.

The mass matrix function calculates the manipulator mass matrix approximately 30 times per second. It retrieves the manipulator joint angle commands from shared memory, calculates the mass matrix entries, and places the results into shared memory. The C code that performs the calculations is generated automatically using the Mathematica programming language. The Mathematica mass matrix script uses a recursive Newton-Euler method to generate the mass matrix.

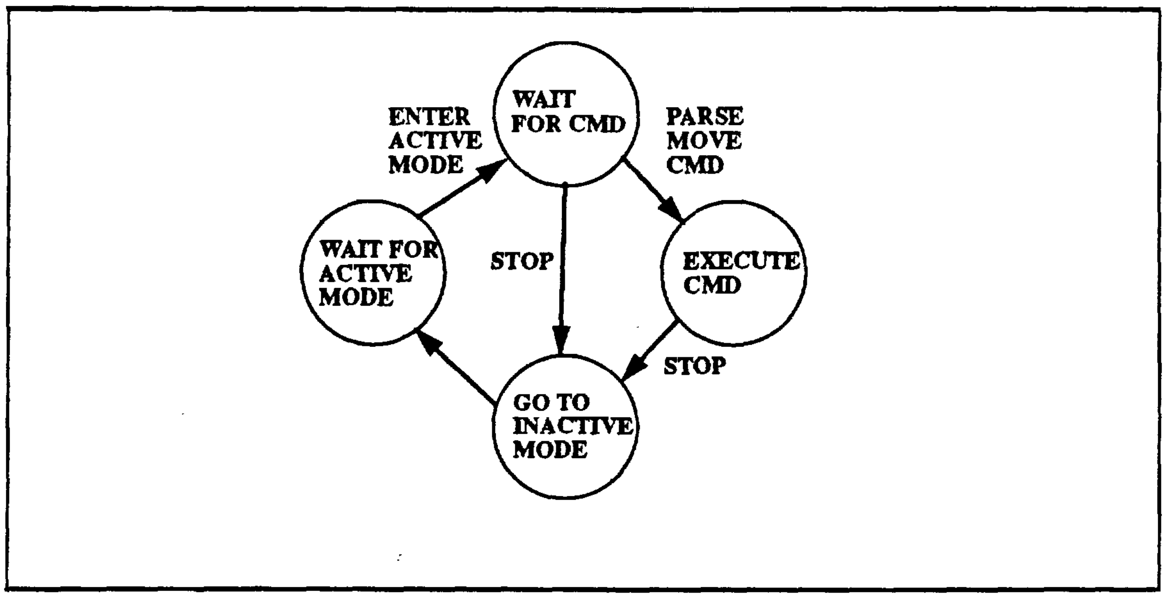
Both the mass matrix process and the RPC communications processes are spawned after initialization of the RDOFA processor, and both continue to execute asynchronously forever, sharing the processor resources.

4.2.2.2 Endpoint and Joint Position Control

Most of the time-critical and computation-intensive processes occur at the RDOFB processor level. There are also some non-real time tasks, such as trajectory generation and loading a stored trajectory from a file. The real time tasks are organized as a state machine, and implemented as an interrupt handler. This interrupt handler services a VME interrupt generated every 20 ms by the RDOFC processor. All endpoint control, model-based compensation, and joint position servo algorithms execute in this time period.

Figure 40 shows the organization of the RDOFB state machine. The system starts up in the "wait for active mode" state, where the brakes are set and servos de-energized. When the user switches the system to "active mode", the servos are energized, the brakes released, and the manipulator servoed to its current position. The system is now in the "wait for command" state. When the user chooses a manipulator motion mode (endpoint, joint, playback) and presses the "move" button, the motion command is parsed, and the system enters the "execute command" state. In addition to retrieving motion parameters, the parsing function sets a pointer to a function to point at the particular function that implements the motion mode desired. As the manipulator moves, this function is called repeatedly, until its return value indicates that the motion is complete. The system then enters the "go to inactive mode" state, in which the servos are de-energized and brakes set. The system can also go directly to this state if the user presses the abort button on the control panel, or the watchdog process detects an error condition.

Figure 40 RDOFB State Diagram



There are several function that are called from within the interrupt handler regardless of the state. If the system is not in the execute command state on entry to the handler, the non-real time tasks are serviced. The manipulator joint angles are sampled, either for current joint servoing or so that joint servoing can begin at the manipulator's current position. After executing code corresponding to the current manipulator state, the watchdog and data logging functions are called, and the new set of joint torque setpoints produced by the joint position servo algorithm are placed into global memory so that the torque servo control level has access to them.

4.2.2.3 Joint Torque Control

The RDOFC processor executes the code for joint torque servo control. Since torque servoing is this processor's only real time task, there is no need for a state machine. The torque servo algorithm code is called from an interrupt handler that services the A/D board's end of scan interrupt, which occurs every 2 ms. At every tenth entry to the handler, it generates a VME interrupt for the RDOFB level interrupt handler to service.

The processor's non-real time function is to initialize the data acquisition at system start-up.

4.2.2.4 Watchdog Process

The watchdog process performs several checks to insure that the manipulator is within its operating limits and that the control system is functioning properly. The first check is a joint oriented limit test that is coded as a C macro. A C structure defines the checks for each joint; an array of these structures defines the numerical limits. If any of the checked quantities is outside of the watchdog limits, the manipulator servos are disabled and an explanatory message is printed. Checked quantities include joint position, velocity, sensed torque, motor temperature, and motor current. One joint is completely tested each time the limit check function is called, so that test frequency for a joint is 7.14 Hz.

A second set of checks verify that the three processors are alive and functional. Processors RDOFA and RDOFC are deemed functional if the global variables that their processes update are being refreshed in shared memory, as indicated by flags. If these flags indicate that this critical data is not being updated, the servos are disabled and the RDOFB processor is halted. Processor RDOFB is monitored by verifying that its synchronization interrupt is alive. If it is not, the servos are disabled and the RDOFB processor is halted.

4.2.3 Shared Memory Interface

The shared memory interface enables the three processors to read and write data to the VME bus memory space. Data structures that pass through the interface include single items (C types int, short, double), structures and arrays, and double buffered data. Data that requires double buffering includes the manipulator mass matrix, joint angles, and torque commands for the joint torque servos. The interface is implemented as a set of source code modules whose objects are linked and loaded into each of the three processors. The interface design goals were:

- cleanly encapsulate the interface implementation
- hide specific VME memory locations from interface users
- allow different sections of the interface to be located in different VME memory spaces
- provide standard access functions for interface data items, and allow no data items to be visible outside of the code modules implementing the interface; only function interfaces are visible
- hide the implementation details of double buffered data from interface users.

Sections of code that need data available through the interface simply make the appropriate function calls, which return either the data itself (for single items) or a pointer to the data. A section of code that uses double buffered data could look something like the following:

```
if (mat_avail() == NO)
        mat_not_ready++;
else {
        get_mat(&mat_local[0][0]);
        mat_not_ready = 0;
}
if (mat_not_ready > MAX_NO_CYCLES)
        shutdown_processor();
```

Double buffering insures that critical sections of code use only fresh data that has not been previously used.

4.2.4 User Interface

The user interface program executes on the host computer, a Sun workstation. It manages a set of windows that provide manipulator control, parameter entry, and status display functions. As the user chooses different operating modes or makes requests to set parameters, the appropriate windows are displayed. Controls within these windows are enabled and disabled depending on the manipulator and control system state.

There are two distinct parts of the user interface code. One part is a set of modules that control the actual window configuration and display. Functions in these modules display and hide windows, control their size and placement, and operate the graphical devices within the windows (buttons, switches, and slider bars). This part of the interface is written using the Sunview user interface toolkit, which is a library of C functions for graphical applications. A code-generating program called Autocode is used in conjunction with Sunview. Autocode enables a programmer to design user interfaces graphically. The programmer arranges windows, panels, and graphical devices as desired, and then Autocode generates modules of Sunview code that may be compiled, linked with application code, and executed. Automatic Sunview code generation reduces graphical interface development and maintenance time and effort dramatically.

The second part of the user interface code captures the control signals and data from the user interface running on the host system and transfers it to processor RDOFA in the target system. These functions are called when the user presses a button or sets a parameter value. When such an event occurs, the target must be notified that something has happened and take action to service the event. UNIX remote procedure calls provide the means to do both these tasks. Once RDOFA receives the data, it makes the necessary function calls to place the data into shared memory so that the other processors have access to it.

4.2.5 Data Logging and Plotting

Originally, custom software to perform data logging and plotting functions was to be developed. During code development, a commercially available software package for real time data capture and presentation was discovered. The program, called StethoScope, is designed to work the host/target system architecture, and is compatible with Sun workstation hosts and target systems running the VxWorks operating system. These features, and the software's low cost compared to the cost of a software development effort to achieve the same functionality, make it an ideal choice for this control system.

The program is divided into two parts. One part executes on the target system and performs the real time data capture, with the goal of minimizing time impact on the critical real time application. The other part of the program executes on the workstation. It receives the captured data from the target system over the local area network (Ethernet) and displays it in near-real time. The software has many additional capabilities and features that are described in its manual [8].

4.3 Operating the Manipulator

To operate the manipulator, the user logs into the Sun workstation host and powers up the electronics cabinet. When the cabinet is powered up, the three processors boot VxWorks, load their application code, and start executing it. The manipulator is initialized to the disabled state, of course. Once the booting process is complete, the user types a single command at the workstation which starts the user interface and StethoScope.

Figure 41 Control Station Main Panel

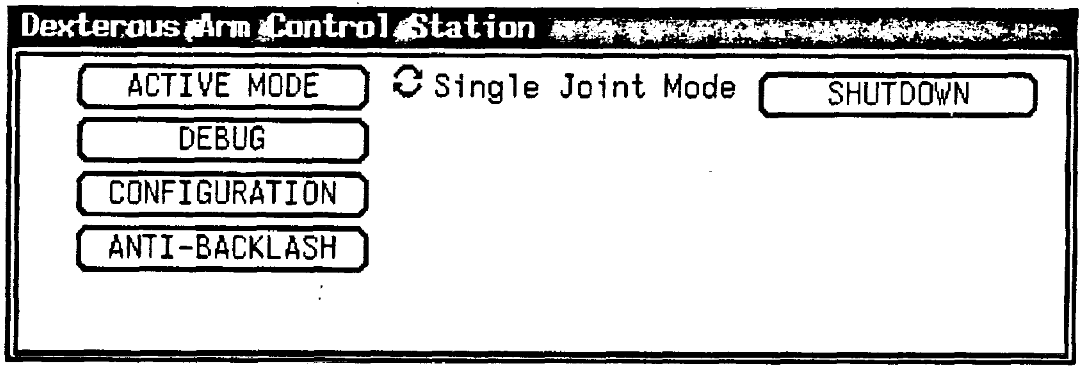
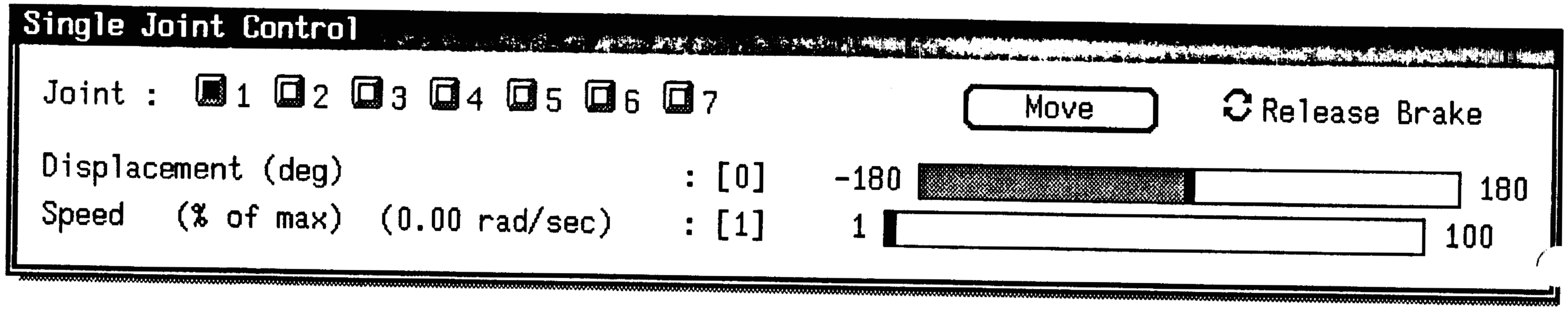

Dexterous Arm Control Station

Figure 41 shows the control station main panel. The user selects the operating mode with the cycle switch in the center, clicking on it until reaching the desired motion mode. The "active mode" button enables the manipulator servos and prepares it to move. The lower three buttons allow the user to set control system parameters.

Figure 42 shows the "Single Joint Mode" panel. The user selects which joint to move, how far to move it from its current position, and how fast to move it, as a percentage of its maximum speed. Pressing the "move" button when the system is in active mode starts the joint's motion. The "Release Brake" button allows the joint to be moved manually. Similar panels exist for sinusoidal single joint motion (for testing), coordinated joint space moves, and endpoint moves. The user can immediately stop manipulator motion by pressing the "abort" button, which appears on the main panel when the system enters active mode.

Figure 42 Single Joint Mode Panel



Single Joint Control

Joint : ■ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7

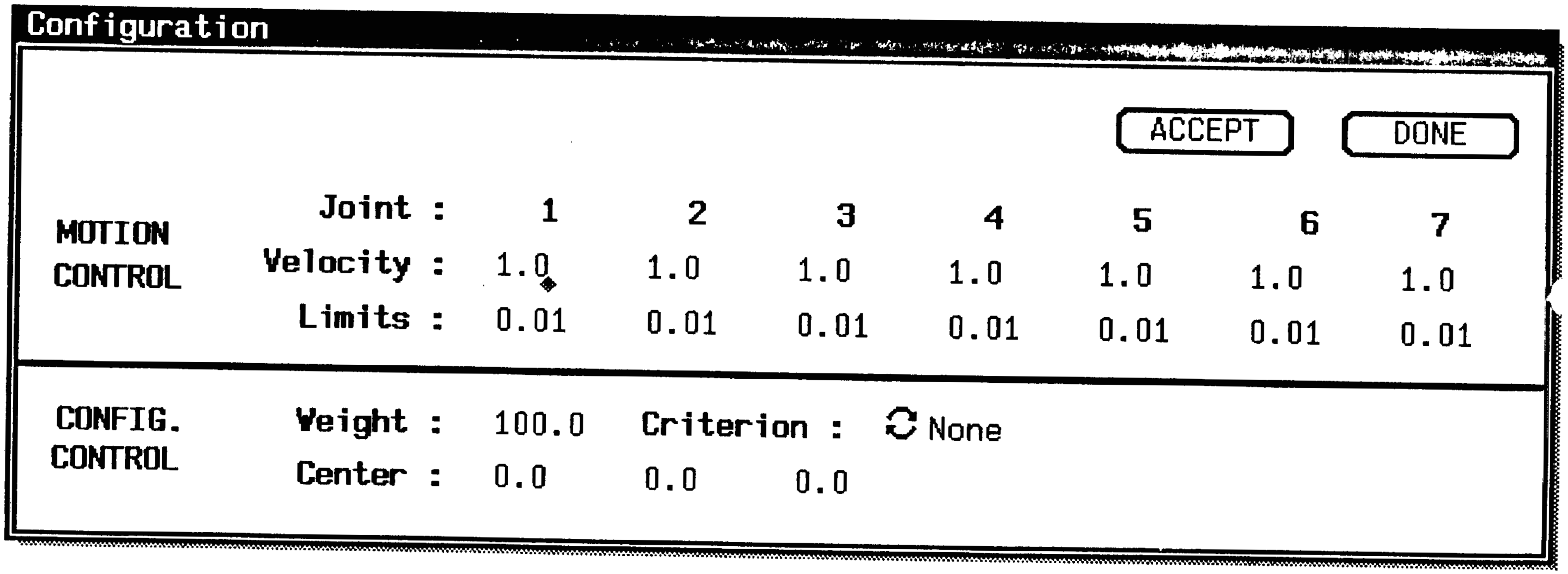
Move

Release Brake

Displacement (deg) : [0] -180 180

Speed (% of max) (0.00 rad/sec) : [1] 1 100

Figure 43 Configuration Control Parameter Panel



Configuration

ACCEPT DONE

MOTION CONTROL

Joint :	1	2	3	4	5	6	7
Velocity :	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Limits :	0.01	0.01	0.01	0.01	0.01	0.01	0.01

CONFIG. CONTROL

Weight : 100.0 Criterion : None

Center : 0.0 0.0 0.0

Figure 43 shows the algorithm parameter panel that is displayed when the "configuration" button on the main panel is pressed. This panel enables the user to set endpoint/ configuration control parameters. New values can be typed into the numerical fields and the cycle switch set to choose the configuration control method. When the settings are as desired, the "accept" button initiates the RPC that sends the new settings from the panel to processor RDOFA in the target system, which in turn writes them to global memory for access by the other processors. A Similar panels exists for the anti-backlash algorithm parameter settings.

Other panels include the "Debug" panel, which allows the user to selectively disable and enable joint torque servoing, the anti-backlash algorithm, and feedforward gravity and acceleration compensation on a joint-by-joint basis. A "Playback" panel provides a means to execute a joint or endpoint trajectory that has been stored on disk.

5.0 Control System Evaluation

The primary system evaluation is a comparison of the performance goals listed in Table 1 with the actual manipulator performance. Schedule and cost constraints limited the amount of testing that could be performed during this Phase II contract. Although contract support has been exhausted, work to characterize the system's performance and implement improvements continues.

5.1 Subsystem Performance

This section describes the joint torque servo, joint position servo, backlash elimination, and endpoint algorithm performance at the subsystem level. Section 3.2 of the report details the design and analysis of these subsystems.

5.1.1 Joint Torque Servoing

The high payload to weight ratio and compactness design goals and subsequent achievements have a significant impact on the manipulator's structural dynamics and control system performance. This impact first became clear during torque servo implementation.

After assembling the first shoulder module and testing its actuator, brake, and sensor subsystems, the first attempt at joint torque servoing was made. The module was mounted to a test stand and fitted with a large hollow tube to approximate the load corresponding to an intermediate inertia configuration of the manipulator. Typical torque sensor data collected while driving the system open loop is shown in Figure 44. Figure 45 shows the power spectrum of this data. Clearly, the sensor is picking up vibration from lightly damped flexible modes in the system. For this test inertia, the lowest resonant frequency is at approximately 16 Hz. Since the torque loop compensation was not designed for this flexible mode within its bandwidth, the torque loop is unstable. The shoulder resonant frequency varies between 40 Hz. at minimum inertia and 6 Hz. at maximum inertia.

Figure 44 **Torque Sensor Data, Shoulder Module**

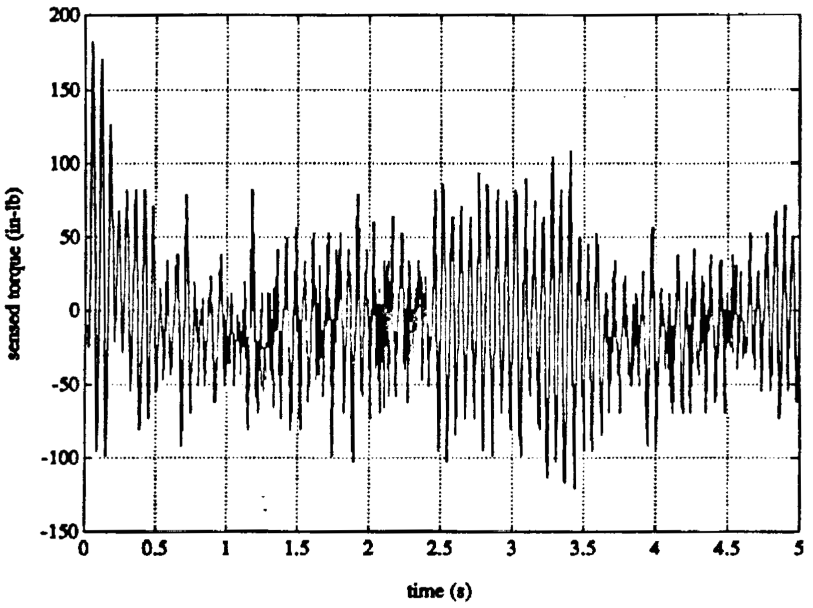
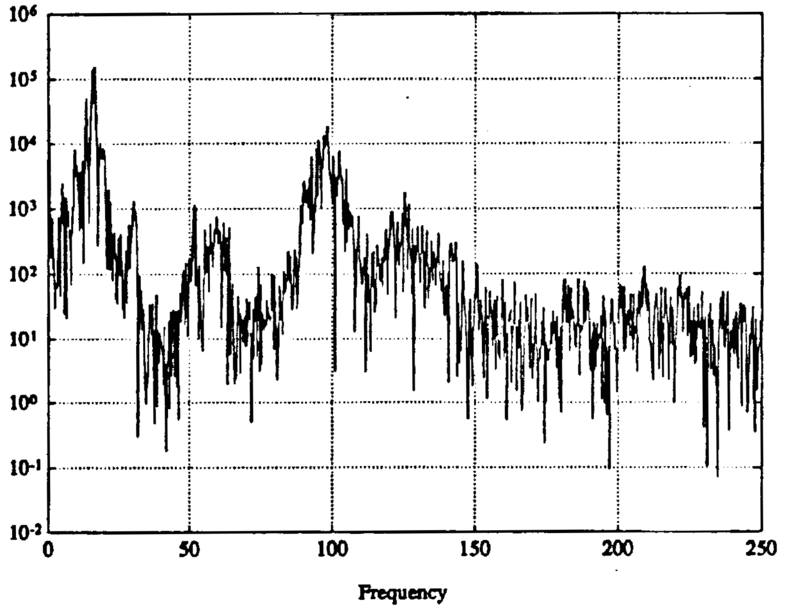


Figure 45 **Power Spectrum, Shoulder Module**



A quick way to determine if the torque loop works properly apart from the unmodeled dynamics is to lower the closed loop bandwidth well below the resonant frequency. The loop compensation was redesigned for a 15 Hz. bandwidth and the joint tested with a low inertia load. The resonant mode for this low inertia load is at 40 Hz. The torque loop response was stabilized. With the torque loop operational, apparent joint friction is greatly reduced; when commanded to zero torque, a very light push on the joint will move it through most of its range of motion. Much of the joint's friction is servoed out, and the resulting dynamic response is close to a $1/s^2$ rigid body, over the torque loop bandwidth.

Non-collocated control through a flexible structure is a well known problem [9]. This problem motivated joint torque feedback, which could (to some degree) actively damp the flexible mode while servoing out friction. The method has been detailed analytically for a single degree of freedom testbed [10]. The approach is to use lead compensation to damp the resonant mode. In subsequent experimental work by the same researchers, the testbed resonant frequency was above the torque loop bandwidth, so the active damping compensation was not tested [11].

There are several difficulties with this solution to the problem. The first is that the frequency of this flexible mode is load-dependent. For outstretched manipulator configurations and large payloads, this frequency will drop substantially. For example, the Dexterous Manipulator's shoulder joint resonant frequency varies from 40 Hz. in the unloaded state down to 6 Hz. for the maximum joint inertia. The load dependence makes strategies such as lead compensation or notch filtering at the resonant frequency non-robust.

A second problem with the reduced bandwidth/notch filtering approach involves phase loss. The original torque loop compensation bandwidth insured that the system approximated an undamped rigid body out to the 50 Hz. closed loop bandwidth. The torque to position frequency response has 180 degree phase shift out to this frequency, as shown in Figure 15. With reduced torque loop bandwidth or extra attenuation from filtering, the phase loss is much faster in the torque to position response, requiring additional lead in the position/rate loop compensation design. Depending on the closed loop bandwidth and/or filter order, the position loop may have to compensate over 100 degrees more of phase loss to achieve a position bandwidth close to the original 5 Hz. design, which is impractical.

The second shoulder joint module was assembled next. It exhibited nearly identical behavior. The upper arm roll joint was assembled and tested next. Torque sensor data collected while the joint was driven open loop show a similar resonance at approximately 25 Hz., using a load corresponding to a relatively low joint inertia. Clearly, the problem is not limited to the shoulder joints. The next logical step was to determine if any particular drivetrain members contribute unduly to the low stiffness and damping.

One of the initial suspects was the instrumented ring gear that serves as output member and torque sensor in each of the joints. Although this gear is designed to be exceedingly stiff, the actual part may not have the high design stiffness, or it may contribute to the flexibility in some unknown manner. A spare solid ring gear that did not have webs and strain gauges was available for testing. Testing the joint with this ring gear required an alternative means of torque sensing. A JR3 Inc. six degree of freedom force/moment sensor of compatible diameter was available. Two adapters were fabricated: one to mount the sensor to the roll module output, and the second to mount the test load to the sensor. Figure 46 shows the force/moment sensor signal (axial axis moment component) with the instrumented ring gear, and Figure 47 shows its power spectrum. These plots verify that the external force/moment sensor can observe the flexible mode.

Figure 46 Torque Sensor Data, Upper Arm Roll Module, Instru. Ring Gear

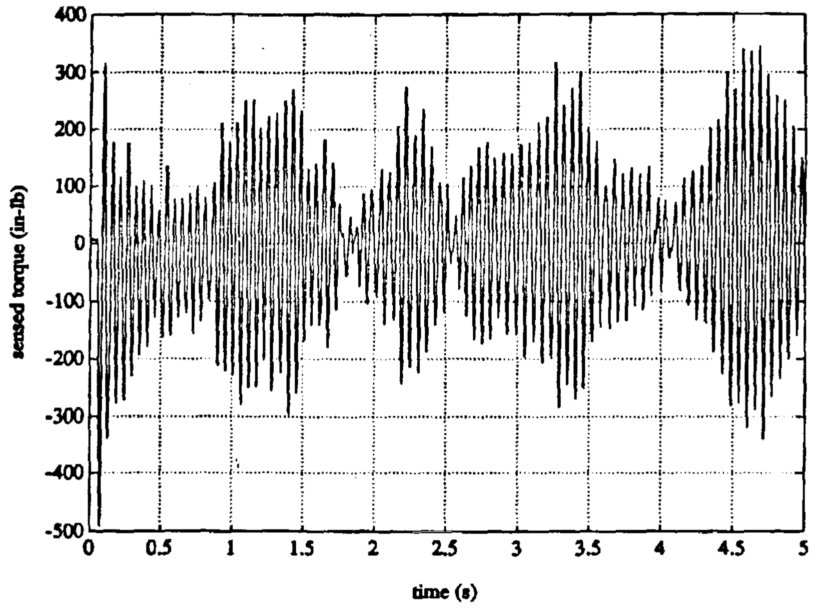
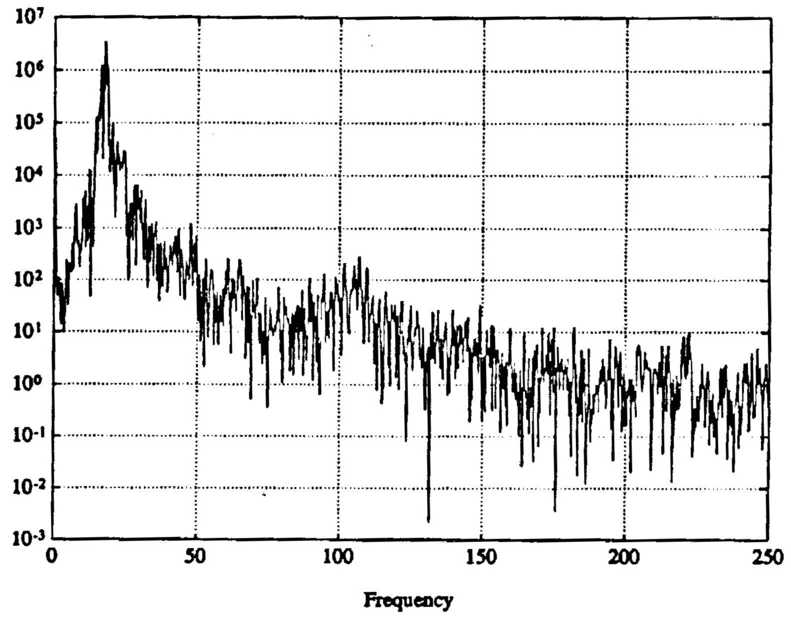
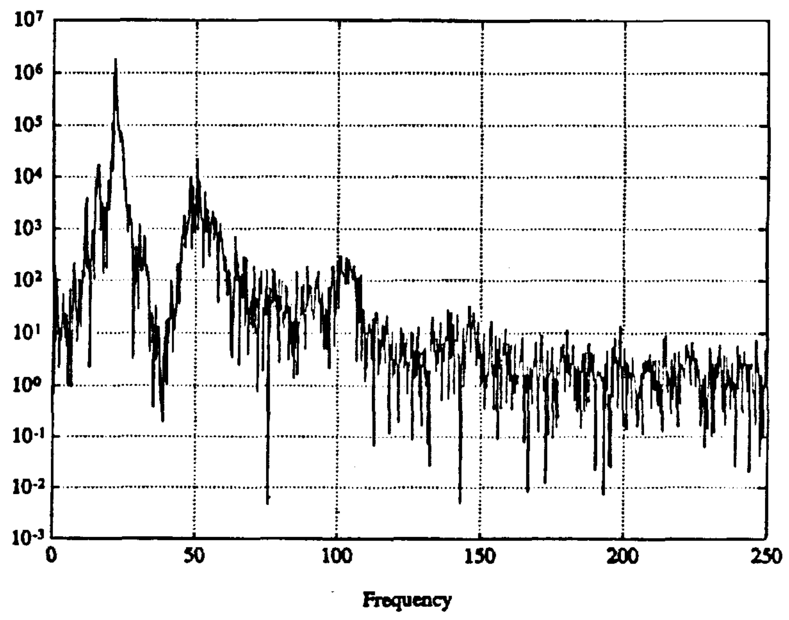


Figure 47 Power Spectrum, Upper Arm Roll Module, Instrumented Ring Gear



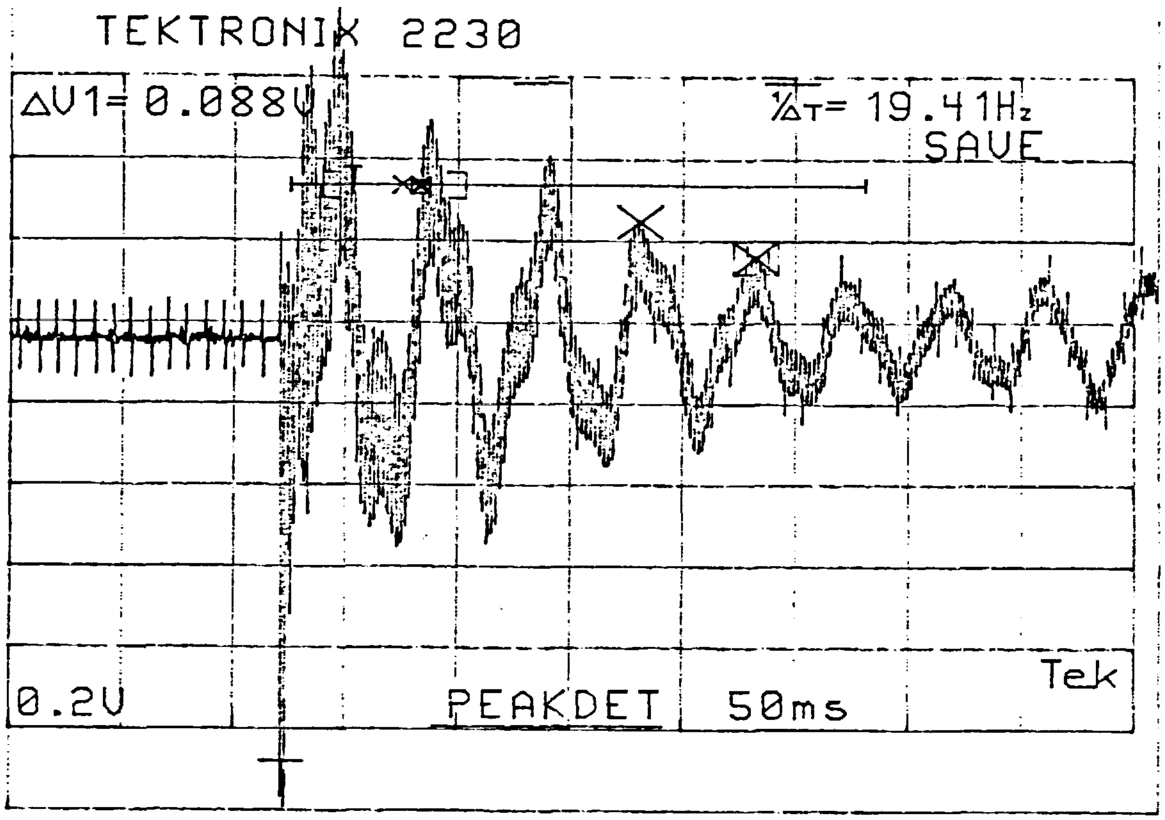
The instrumented ring gear was replaced with the solid ring gear, and the measurements repeated. Figure 48 shows the corresponding power spectrum. Note that the results are quite similar.

Figure 48 Power Spectrum, Upper Arm Roll Module, Solid Ring Gear



Another hypothesis was that flexibility in the test stand was contributing to the flexible mode. The test stand's lowest structural resonant frequency is approximately 80 Hz. In order to test this hypothesis, the upper arm roll module was mounted to a large 1 inch steel plate, which was in turn clamped to a large welding table, which provided a very rigid base for testing. This equipment is located in a different part of the facility, so the manipulator controller was unavailable for open loop testing. An alternative test setup using an inductive velocity sensor was devised. A small magnet was mounted to the test load at a sufficient radius from the joint axis so that joint vibration would cause the magnet to move. A coil was positioned close to the magnet so that as the magnet moved, a measurable voltage would be induced in the coil. This voltage is proportional to the velocity of the magnet, and it can be captured and measured using a storage oscilloscope. Although this method is crude, it provides good time/frequency data on the joint vibration. Figure 49 shows the measured voltage in response to a mallet tap on the side of the test load. The waveform has two frequency components; the lower component is just over 19 Hz. The much higher second component probably represents the audible ring of the tubular test load. The plot also clearly shows the light damping associated with this vibrational mode. This result shows that mounting the joint to a very rigid base has little effect on the resonant mode.

Figure 49 Upper Arm Roll Module Vibration; Bench Test

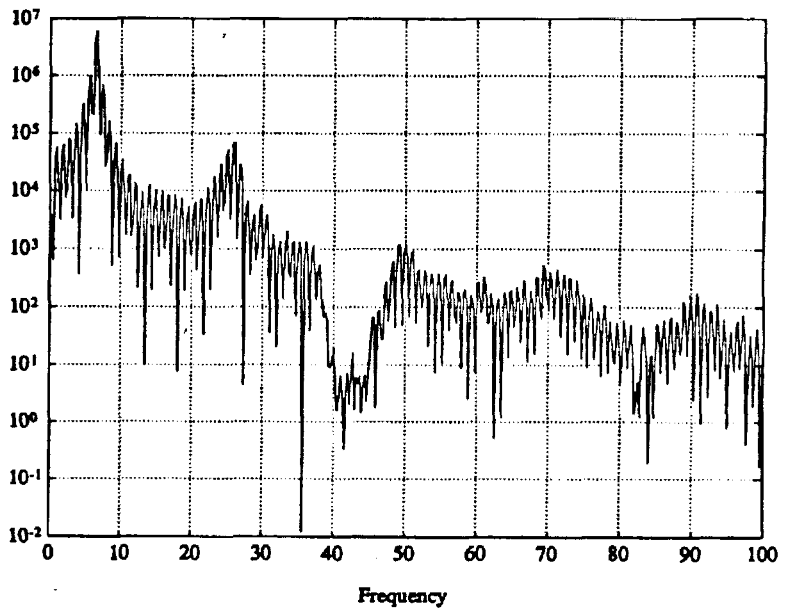


Elbow and wrist module testing yielded similar results. In each case, the joint has a very lightly damped flexible mode whose frequency varies between approximately 6 and 40 Hz., depending on loading. Strategies for ameliorating the effects of the low frequency joint vibrational mode include raising the joint stiffness, increasing the damping, or both. The flexible mode frequency increases only as the square root of the stiffness; thus, great efforts to increase drivetrain stiffness appear unattractive. As a first attempt to increase damping, one of the shoulder module transmissions was packed with viscous grease, and the previous measurements were repeated. The measurements showed practically no difference in the joint dynamics. At this point in the project, great effort had been applied to identifying and understanding the resonance problem. The strategy that made the most sense was to leave the torque loops disabled and to include additional position/rate loop shaping to compensate the resulting joint dynamics.

5.1.2 Position / Rate Servoing

Developing position loop shaping for the individual joints (without torque loops) is relatively straightforward. Typically, the open loop data shows that, at the desired crossover frequency, the non-torque servoed joint has somewhat more attenuation and phase loss than that in the original torque servoed case. When the flexible mode frequencies are in the upper part of their range, it is not difficult to obtain good stability margins, bandwidth, and low frequency gain (integral compensation) for each joint. The compensator design yielded closed loop dynamics similar to the example in Section 3.2.3.4. The real difficulty arises when the individual modules are joined into a seven degree of freedom manipulator, and the resonant frequencies drop close to the servo bandwidths. The vibrational dynamics of the manipulator as a whole is a combination of the actuator structural dynamics. Further measurements were taken in order to characterize the complete mechanism's flexibility. The manipulator, carrying no payload, was extended to its full reach, which would produce the lowest resonant frequency. The brakes were set, and the toolplate was tapped with a mallet. The response was measured with the shoulder elevation joint torque sensor. Figure 50 shows the power spectrum of such a measurement. The lowest frequency mode is at about 7 Hz., and the second mode is at about 25 Hz.

Figure 50 Manipulator Vibration Power Spectrum



Again, the vibrational modes are very lightly damped and only slightly above the desired position loop bandwidth. The initial attempts to use the position loop shaping that worked with the joints as individuals proved unstable for the manipulator as a whole. The manipulator vibrational modes were almost immediately excited by the servos. Once again, the resonant frequency is a function of manipulator configuration and load. In order to drive the manipulator stably, the position loop bandwidth of the joints was reduced down to approximately 1-2 Hz. The manipulator operates stably at this bandwidth, but tracks rather poorly due to the low authority control. In particular, the steady state tracking is well below the performance required to achieve the repeatability goals, primarily due to high joint friction combined with low control authority.

The model-based compensation's performance was examined somewhat more quantitatively, although only rudimentary measurements were made. Ideally, this compensation maintains even servo response throughout the workspace and under the full range of payloads. In order to evaluate variations in position servo response, the major joints' position servo bandwidth was tested for three payload/configuration conditions:

1. No payload, minimum joint inertia pose.
2. No payload, maximum joint inertia pose.
3. 35 lb. payload, intermediate joint inertia pose.

The elbow and wrist joints see little configuration-dependent inertia variations, so they were tested in an intermediate configuration with no payload and 35 lb. payload. Once again, high joint friction, uncompensated due to the lack of torque loops, made the measurements very difficult, particularly for the heavily-loaded shoulder joints. Table 5 summarizes the results.

Table 5 Position Servo Bandwidth Testing

Joint	Position Bandwidth (Hz)		
	Test 1	Test 2	Test 3
Shldr Azi	x	x	(0.5)
Shldr Ele	(0.5)	(0.5)	(0.5)
UA Roll	2.0	2.0	1.5
Elbow		1.6	1.4
Wrist Yaw		2.0	1.0
Wrist Pitch		2.0	1.5
Wrist Roll		2.2	2.0

The table entries marked "x" indicate that the friction effects dominated the test, making the measurement invalid. The parenthesized measurements indicate that high friction tends to "clip" the response and make the apparent bandwidth lower than the servo design indicates.

These results show that there is some variation in the servo response over the various joint inertia conditions. It is important to note that the current servo tuning yields much lower bandwidths than the original design called for (~ 5 Hz). In addition, the mass properties used in the feedforward compensation are estimates that are somewhat in error with the manipulator's true mass properties (the manipulator is 15 lb. lighter than the design goal).

5.1.3 Backlash Elimination

On the positive side, the backlash elimination technique's performance is superb. As assembled, the joints have a fairly large amount of backlash. The elbow, for example, has about 0.5 degrees. Using the debugging switches provided in the user interface, the user can enable/disable the joint torque biasing "on the fly" and watch the response. Once disabled, a motion command or external disturbance will cause the joint to vibrate; enabling the biasing immediately stops the vibration.

5.1.4 Endpoint Algorithm

Since the endpoint algorithm is decoupled from the lower levels of servo control and thus unaffected by their dynamics (refer to Section 3.2.2, page 21), its behavior is not compromised by the difficulties described earlier. This fact means that kinematic simulations provide a good measure of the algorithm's performance in implementation.

The factors that actually determine the implemented algorithm's performance involve hardware and software implementation. The algorithm solves the manipulator kinematics using an iterative numerical solution. Iteration continues until the algorithm achieves a solution within a specified tolerance, or reaches the maximum number of iterations allowed. The algorithm update rate limits the maximum number of iterations allowed for each trajectory point. Clearly, fast hardware and efficient software implementation will increase the number of allowable iterations. The important question is, "How many algorithm iterations are required for good algorithm performance?"

The two principal factors that determine what this maximum should be are:

1. The commanded endpoint speed
2. The condition of Equation (20).

For relatively low commanded endpoint speeds, Δx in (20) is small and the numerical solution will be close to exact on the first iteration. At higher speeds, Δx will be larger, requiring more iterations for solution. As the manipulator approaches kinematic singularity, (20) becomes ill-conditioned. Again, more iterations are required to achieve convergence to a given tolerance.

The current computer hardware and algorithm software implementation yield an endpoint algorithm iteration time of approximately 5 ms. In order to execute the endpoint algorithm, joint position servo, and gravity feedforward compensation synchronously with 20 ms updates, the maximum number of iterations is set to 3. The corresponding tolerance on endpoint position and orientation error is 0.005 inches and 0.005 radians. Note that this tolerance is held throughout trajectory tracking, including the fastest, most demanding sections of a move. If desired, a much tighter end of move tolerance could be specified without increasing the required number of iterations.

5.2 Manipulator Performance

There is a wide spectrum of performance criteria and testing applicable to robotic manipulators. Examples include accuracy, repeatability, speed, payload capability, force capability, efficiency, and power consumption. Contract cost constraints prohibit extensive testing of each of these criteria. In addition, the current system performance level is tainted by the unresolved low frequency structural dynamics problem and the corresponding servo bandwidth reduction forced on the system. Much of the effort near the contract's end was applied toward solving this problem and improving the manipulator performance, rather than toward extensive testing. This effort is continuing, although without contract support. Once the problem is relieved, the subsystem performance described in the previous sections will improved substantially, and the system will achieve the design performance level.

5.2.1 Mechanical Design Goals

The design-oriented goals outlined in Table 1 were, for the most part, achieved or exceeded. The manipulator length to toolplate is 55 inches. The actual 150 lb. weight is 9% less than the design weight. Difficulty in routing the wire harness reduced the wire count emerging at the toolplate connector to 40 wires. The remaining 72 wires terminate in the hollow forearm tube.

5.2.2 Payload Capability

The maximum payload that the Dexterous Manipulator can statically support is related more to the manipulator's mechanical design than to the control system. However, the ability to move this large payload smoothly throughout the workspace is a function of both mechanical properties and control system robustness. The manipulator's flexible dynamics complicates control at higher payloads, where the resonant mode frequency is close to the servo bandwidth.

Basic qualitative testing characterized the payload capability. The manipulator was loaded to 35 lb. and moved around the workspace in endpoint control mode. With the current 1-2 Hz. joint position servo bandwidth, the manipulator moved the load in a stable manner through most of the workspace, but became unstable near the outer workspace border, where the resonant frequency is lowest. This behavior indicates that, even at this low servo bandwidth, the current servo tuning does not shape the response such that the manipulator can operate stably at high load near the workspace edge. While "rolling off" the response at a somewhat lower frequency would stabilize the manipulator response under the fully loaded condition, the lower bandwidth would reduce performance in the remainder of the workspace unacceptably. Once again, the proper course of action is to resolve the structural dynamics problem.

5.2.3 Endpoint Speed

Maximum endpoint speed depends on both the maximum achievable joint speeds and the particular manipulator pose. From a theoretical standpoint, the maximum endpoint speed occurs when the manipulator is outstretched (for maximum radius), and all joints with parallel axes are moving at their maximum angular speed. Of course, this is a singular manipulator configuration at the workspace edge and is thus useless for manipulation. The design goal in Table 1 is obtained by considering each major joint alone. Multiplying the joint's maximum speed by its distance to the toolplate yields approximately 40 in./s. This method is more reasonable, but considers joint space moves rather than coordinated endpoint moves. A useful endpoint speed measurement considers coordinated endpoint moves and thus exhibits both physical and control capabilities.

Limited endpoint speed testing consisted of moving the manipulator and a 10 lb. payload to various positions in the workspace, commanding linear endpoint moves, and measuring the achieved endpoint velocity. The endpoint trajectories are quintic polynomials; the peak speed occurs around the trajectory center. Table 6 shows some of the results.

Table 6 Endpoint Speed Testing Results

Test Case	Peak Endpoint Speed
Outer section of workspace; move in y-direction w/ fixed orientation	66.0 in./s
Interior workspace; move in x-direction w/ fixed orientation	37.2 in./s
Interior workspace; rotate about y-axis w/ fixed position	1.44 rad/s (82.5 deg/s)

6.0 Conclusions and Recommendations

6.1 Observations

The Odetics Dexterous Manipulator has extended the state of the art in manipulator design in several significant areas:

- The manipulator achieves **extensive modularity** with its simple mechanical and electrical interfaces, clean exterior, and totally internal wire harness from base to toolplate.
- The **high strength to weight ratio** will enable the manipulator to dexterously manipulate significant payloads for its size.
- Careful attention to packaging has yielded **compactness** that allows the mechanism to stow into a small volume, which is an important requirement for space applications.
- **Redundant actuation and sensing** provide fault tolerance that is important to any application and crucial to space operations.

Significant accomplishments in algorithms and control have also been made during the project:

- An **advanced endpoint control / redundancy resolution algorithm** has been successfully demonstrated on real hardware.
- The dual motor drives combined with the anti-backlash technique has been proven to **eliminate backlash** that would otherwise destabilize a closed loop position servo.
- A multiprocessor-based hierarchical control system has been successfully integrated with the manipulator mechanism. The **open control system implementation** uses commercially available hardware and structured software that enables users to tailor the system to specific applications, test new advances in control algorithms, and upgrade hardware as more powerful computers become available.

Another important observation is that the system's modularity will allow it to be configured in kinematic arrangements other than a 7 degree of freedom manipulator. Applications that don't require dexterous motion could still benefit from the modules' high strength and fault tolerance by using them in a simpler system. One example is a very high performance pointing (pan and tilt) unit.

6.2 Further Development Activities

The salient fact to bear in mind when considering further development is that the starting point is a working system that includes the tools required to support evolution to a product. Ideally, the space and commercial communities' needs will drive further design and implementation choices for the system's form and function. How expediently a product evolves depends primarily on funding. Whatever entity funds continuing development will be able to leverage its dollars off the solid accomplishments obtained during this research.

6.2.1 Noncollocated Flexible Dynamics Compensation

The final step to finish this stage of the development is to enhance the control algorithms to compensate for the combined actuator flexibility and noncollocation. Once properly compensated, the system's true performance level can be achieved and measured. While work in this area is currently ongoing, funding limitations keep the level of effort low. A few principal activities define the continuing work:

- Properly characterize the system's flexible modes, using experimental techniques.
- Analyze the experimental data and design control compensation to actively damp the flexibility.
- Test the compensation on the manipulator hardware.
- Measure the performance and report the results.

The specific methodology to accomplish this work is outlined as follows. First, a single joint's flexible dynamics would be characterized, using spectral analysis techniques. A two body joint model that represents the noncollocated system would be tuned to reflect the measured dynamics. Torque loop compensation that properly damps the flexible mode over some nominal frequency range would be developed. A simple single joint nonlinear simulation would verify that this compensation works when torque ripple and stiction effects are present. This nominal compensation must then be generalized to damp the resonant mode over its entire frequency range, that is, for the full range of manipulator payloads and configurations. Once the compensation is verified, it would be implemented and tested on the actual manipulator. When the compensation is properly tuned, the position/rate servo bandwidth can be increased to the design level, improving the manipulator's tracking capability. Finally, the improved manipulator performance would be measured and the results reported.

6.2.2 Unilateral and Bilateral Teleoperation

Odetics has previously implemented teleoperator systems and continues active development in the area. Since, by definition, space telerobotics will utilize teleoperation as a means of manipulator control, it makes sense to add teleoperator control capability to the Dexterous Manipulator system. The control system includes hardware and software provisions to support teleoperation. Unilateral control (no force feedback) can be added to the current system with a relatively limited effort. Bilateral teleoperation could also be implemented, with substantially greater effort.

The master controller would normally be the most costly system hardware component. Fortunately, one master controller is already available, and a more sophisticated one is under development. As part of another project, a JPL Model C Force Reflecting Hand Controller was fabricated and tested. This hand controller is currently available for integration with the Dexterous Manipulator System. The company is also developing a 7 degree of freedom universal Exoskeleton Control under an Air Force SBIR contract [13]. While the prototype exoskeleton will be delivered to the customer, another could be fabricated if adequate funding was available. This kinematically redundant exoskeleton would provide valuable control capabilities that are relevant to a kinematically redundant manipulator and unavailable from a 6 degree of freedom hand controller.

Teleoperation would be added to the system using a phased approach. The initial activity, which could be called phase 0, would center on carefully defining system requirements and developing a concept for implementation based on both previous work at Odetics and the extensive research in teleoperator system implementation conducted at NASA ([12], for example). In the first phase, unilateral control would be added. After unilateral control is successfully demonstrated, the more challenging bilateral control would be implemented and tested.

6.2.3 Path Planner Integration

Odetics is currently working on another Phase II SBIR contract, also sponsored by JPL, to develop path planning and trajectory generation algorithms for the Dexterous Manipulator [14]. The path planning algorithms will find the shortest path around obstacles in the manipulator workspace to a goal position for the manipulator end effector. The trajectory generation algorithms use a potential field approach to guide the end effector along this path while simultaneously avoiding collisions between the end effector, the links of the manipulator, and obstacles in the manipulator workspace. The resulting trajectory can be converted to joint angle commands and input to the joint servo control algorithms. This project includes an animation of the Dexterous Manipulator being driven by the Path Planner algorithms.

Some of the groundwork is already in place for integration of the Path Planner algorithms into the Dexterous Manipulator control system. These algorithms fit cleanly into the control hierarchy. They would reside at the same level as the endpoint control algorithm and would serve roughly the same purpose, which is to produce commands for the joint servo algorithms. Once again, with some initial conceptual work, this integration could be encapsulated into a well-defined project scope that could be completed with a high probability of success, once funding becomes available.

7.0 References

[1] Odetics, Inc., "Phase I Final Report - Control Algorithm for a Redundant Degree of Freedom Manipulator", NASA Contract NAS7-1006, September 1988

[2] Klein, C.A., and Huang, C.H., "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 3, March/April 1983, pp. 245-250

[3] Chang, P.H., "A Closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundancy", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 5, October 1987, pp. 393-403

[4] Craig, J.J., Introduction to Robotics: Mechanics and Control, Addison-Wesley, 1986

[5] Seraji, H., "Decentralized Adaptive Control of Manipulators: Theory, Simulation, and Experimentation", IEEE Transactions on Robotics and Automation, Vol. 5, No. 2, April 1989, pp. 183-201

[6] SD/FAST User's Manual, Symbolic Dynamics, Inc., Mountain View, CA

[7] Advanced Continuous Simulation Language (ACSL) Reference Manual, Mitchell and Gauthier Associates, Concord, MA

[8] StethoScope User's Manual, Real Time Innovations, Sunnyvale, CA

[9] Franklin, G.F., nad Powell, J.D., Digital Control of Dynamic Systems, Addison-Wesley, 1980

[10] Tilley, S.W., Francis, C.W., Emerick, K., Hollars, M.G., "Preliminary Results On Non-Collocated Torque Control of Space Robot Actuators", Proceedings of the NASA Conference on Space Telerobotics, January 1989, Vol. II, pp. 143-152

[11] Tilley, S.W., Hollars, M.G., Emerick, K. S., "Experimental Control Results In A Compact Space Robot Actuator", Proceedings of the ASME Winter Annual Meeting, Anaheim, CA, December, 1989

[12] Lee, T.S., "Implementation and Design of a Teleoperation System Based on a VMEbus/68020 Pipelined Architecture", Proceedings of the NASA Conference on Space Telerobotics, January 1989, Vol. II, pp. 97-107

[13] Odetics, Inc., "Phase II Interim Report - Exoskeleton Master Arm, Wrist, And End Effector Controller With Force Reflecting Telepresence", Dept. Of Defense Contract F33615-89-C-0587, April 1991

[14] Odetics, Inc., "Phase II Final Report - Path Planner for Redundant Degree of Freedom Manipulator", NASA Contract NAS7-1108, January,1992

NASA

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Control Algorithm Implementation for a Redundant Degree-of-Freedom Manipulator		5. Report Date October 13, 1991
		6. Performing Organization Code
7. Author(s) Steve Cohan		8. Performing Organization Report No.
		10. Work Unit No.
9. Performing Organization Name and Address Odetics, Inc. 1515 S. Manchester Avenue Anaheim, CA 92802		11. Contract or Grant No. NAS7-1062
12. Sponsoring Agency Name and Address National Aeronautics & Space Administration Washington, DC 20546-0001 NASA JPL, 4800 Oak Grove Dr., Pasadena, CA 91109		13. Type of Report and Period Covered Final Report, 13 July-89 - 13 Oct 91
		14. Sponsoring Agency Code
15. Supplementary Notes		
16. Abstract Not applicable		
17. Key Words (Suggested by Author(s)) Not applicable	18. Distribution Statement [redacted]	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 90 / 22. Price

NASA FORM 1626 OCT 86