1996

Control system design for robots used in simulating dynamic force and moment interaction in virtual reality applications

Christopher Lee Clover
Iowa State University

Control system design for robots used in simulating dynamic force and moment interaction in virtual reality applications

by

Christopher Lee Clover

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Mechanical Engineering

Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1996

DEDICATION

This dissertation is dedicated to the memory of my grandfather, Everett Lee Clover. His belief, which was instilled in me at an early age, that a good education is the cornerstone in leading a rewarding and successful life played no small role in motivating me to pursue a doctoral degree. For his wisdom and encouragement, I will always be grateful.

TABLE OF CONTENTS

LIST OF FIGURES v
LIST OF TABLES x
ACKNOWLEDGEMENTS xii

I. GENERAL INTRODUCTION 1
A. Overview 1
B. Dissertation Organization 4

II. THE UTILITY OF ABBREVIATED LINEAR AND NONLINEAR ROBOT MODELS 5
Abstract 5
List of Nomenclature 6
1 Introduction 6
2 Literature Survey 8
2.1 Overview 8
2.2 Linearization 9
3 Nonlinear Modeling of the PUMA 560 13
3.1 Background 13
3.2 Case Study - PUMA 560 18
4 Linear Modeling of the PUMA 560 22
4.1 Developing the Equations 22
4.2 Eigenvalue Analysis 28
5 Discussions and Conclusions 29
References 29
Appendix A -- Nonlinear Equations 52
Appendix B -- Linear Equations 53

III. THE USE OF SYMBOLIC, COMPUTER GENERATED CONTROL LAWS IN HIGH SPEED TRAJECTORY FOLLOWING: ANALYSIS AND EXPERIMENTS 55
Abstract 55
I. INTRODUCTION 55
II. LITERATURE SURVEY 56
III. A COMPUTER GENERATED CONTROLLER 62
A. Background 62
B. Nonlinear Feedforward Torques 64
C. Controller Gain Matrices 65
IV. ROBUSTNESS ANALYSIS 70
A. Stability Analysis 71
B. Numerical Analysis for the PUMA 560 73

V. SIMULATION AND EXPERIMENTS WITH A PUMA 560 78
A. Overview 78
B. Experimental setup 80
C. Results 81
VI. DISCUSSION AND CONCLUSIONS 83
REFERENCES 85
APPENDIX 106

IV. A CONTROL SYSTEM ARCHITECTURE FOR ROBOTS USED TO SIMULATE DYNAMIC FORCE AND MOMENT INTERACTION BETWEEN HUMANS AND VIRTUAL OBJECTS 111
Abstract 111
I. INTRODUCTION 112
A. Overview 112
B. Literature Survey 114
C. Motivations 118
II. CONTROL SYSTEM ARCHITECTURE 119
A. Overview 119
B. Software 121
C. Hardware 122
III. TRAJECTORY GENERATION 124
A. Virtual Object Kinematics 125
B. Virtual Object Dynamics 128
IV. INVERSE KINEMATICS 131
A. Position 132
B. Velocity 137
C. Acceleration 139
V. TEST RESULTS 143
A. Translational Motion 143
B. Rotational Motion 145
C. Collision Simulation 146
VI. DISCUSSION AND CONCLUSIONS 147
REFERENCES 148
APPENDIX 165

V. GENERAL CONCLUSIONS 167

LIST OF FIGURES

II. THE UTILITY OF ABBREVIATED LINEAR AND NONLINEAR ROBOT MODELS

Figure 1: D-H parameter descriptions between link *i* and link *i*-1 for (a) a revolute joint and (b) a prismatic joint 33

Figure 2: PUMA 560 coordinate axes 33

Figure 3a: Joint position specified via equation (18) 34

Figure 3b: Joint velocity specified via equation (19) 34

Figure 3c: Joint acceleration specified via equation (20) 35

Figure 4a: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a, 10a-3c 35

Figure 4b: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a, 10a-3c 36

Figure 4c: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a, 10a-3c 36

Figure 4d: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a, 10a-3c 37

Figure 4e: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a, 10a-3c 37

Figure 4f: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a, 10a-3c 38

Figure 5a: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2) 38

Figure 5b: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2) 39

Figure 5c: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2) 39

Figure 5d: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2) 40

Figure 5e: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2) 40

Figure 5f: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2) 41

Figure 6a: Root locus showing eigenvalue differences between the abbreviated and un-abbreviated linear systems given in Table V 41

Figure 6b: Root locus showing eigenvalue differences between the abbreviated and un-abbreviated linear systems given in Table VI 42

Figure 6c: Root locus showing eigenvalue differences between the abbreviated and un-abbreviated linear systems given in Table VII 42

Figure 7a: Time response of the linear sytem given in Table VII 43

Figure 7b: Time response of the linear sytem given in Table VII 43

Figure 7c: Time response of the linear sytem given in Table VII 44

Figure 7d: Time response of the linear sytem given in Table VII 44

Figure 7e: Time response of the linear sytem given in Table VII 45

Figure 7f: Time response of the linear sytem given in Table VII 45

III. THE USE OF SYMBOLIC, COMPUTER GENERATED CONTROL LAWS IN HIGH SPEED TRAJECTORY FOLLOWING: ANALYSIS AND EXPERIMENTS

Figure 1: Controller logic for a force reflective robotic system 90

Figure 2: PUMA 560 coordinate axes 91

Figure 3a: Root Locus for varying α with desired poles at -4.5±4.5j (imaginary axis crossings at α=-0.4,2.6) 91

Figure 3b: Root Locus for varying α with desired poles at -45±45j (imaginary axis crossings at α=-127.5,160.0) 92

Figure 4a: Root Locus for varying β with desired poles at -4.5±4.5j (imaginary axis crossing at β=4.8) 92

Figure 4b: Root Locus for varying β with desired poles at -45±45j (imaginary axis crossing at β=39.0) 93

Figure 5a: Root Locus for varying γ with desired poles at -4.5±4.5j (just touches imaginary axis at γ=0.0) 93

Figure 5b: Root Locus for varying γ with desired poles at -45±45j (just touches imaginary axis at γ=0.0) 94

Figure 6: Root Locus for varying γ with desired poles at -45±45j assuming a 200 Hz sampling rate (imaginary axis crossings at γ=4.6 and γ<1) 94

Figure 7a: Nominal trajectory for joints 1 and 4 (rad,rad/sec,rad/sec^2) 95

Figure 7b: Nominal trajectory for joint 2 (rad,rad/sec,rad/sec^2) 95

Figure 7c: Nominal trajectory for joints 3 and 6 (rad,rad/sec,rad/sec^2) 96

Figure 7d: Nominal trajectory for joint 5 (rad,rad/sec,rad/sec^2) 96

Figure 8a: Position error profiles for joint 1 (range of motion: 1.57 rad) 97

Figure 8b: Position error profiles for joint 2 (range of motion: 0.873 rad) 97

Figure 8c: Position error profiles for joint 3 (range of motion: 1.83 rad) 98

Figure 8d: Position error profiles for joint 4 (range of motion: 1.57 rad) 98

Figure 8e: Position error profiles for joint 5 (range of motion: 0.873 rad) 99

Figure 8f: Position error profiles for joint 6 (range of motion: 1.83 rad) 99

Figure 9: Cartesian position error profiles (range of motion: 0.52 meters) 100

Figure 10a: Acceleration error profiles for joint 1 (peak acceleration: 4.03 rad/sec^2) 100

Figure 10b: Acceleration error profiles for joint 2 (peak acceleration: 2.24 rad/sec^2) 101

Figure 10c: Acceleration error profiles for joint 3 (peak acceleration: 4.70 rad/sec^2) 101

Figure 10d: Acceleration error profiles for joint 4 (peak acceleration: 4.03 rad/sec^2) 102

Figure 10e: Acceleration error profiles for joint 5 (peak acceleration: 2.24 rad/sec^2) 102

Figure 10f: Acceleration error profiles for joint 6 (peak acceleration: 4.70 rad/sec^2) 103

IV. A CONTROL SYSTEM ARCHITECTURE FOR ROBOTS USED TO SIMULATE DYNAMIC FORCE AND MOMENT INTERACTION BETWEEN HUMANS AND VIRTUAL OBJECTS

Figure 1: Controller logic for a force reflective robotic system 153

Figure 2: D-H parameter descriptions between link *i* and link *i*-1 for (a) a revolute joint and (b) a prismatic joint 154

Figure 3: PUMA 560 coordinate axes 154

Figure 4: The PUMA 560 mounted with force/moment transducer and gripping fixture 155

Figure 5: Reference and experimental Cartesian X and Y positions with respect to the global origin 155

Figure 6: Reference and experimental results for the velocity component in the body fixed y-axis direction 156

Figure 7: Reference and experimental results for the acceleration component in the body fixed y-axis direction 156

Figure 8: Force component in the body fixed y-axis direction 157

Figure 9: Total acceleration magnitudes 157

Figure 10: Total force magnitude 158

Figure 11: The magnitude of the force vector divided by the magnitude of the acceleration vector for the 5 kg case 158

Figure 12: The magnitude of the force vector divided by the magnitude of the acceleration vector for the 150 kg case 159

Figure 13: Reference and experimental results for the angular velocity component in the body fixed z-axis direction 159

Figure 14: Reference and experimental results for the angular acceleration component in the body fixed z-axis direction 160

Figure 15: Moment component in the body fixed Z-axis direction 160

Figure 16: Total angular acceleration magnitude 161

Figure 17: Total moment magnitude 161

Figure 18: The magnitude of the moment vector divided by the magnitude of the angular acceleration vector for the 0.83 kg-m^2 case 162

Figure 19: The magnitude of the moment vector divided by the magnitude of the angular acceleration vector for the 12.5 kg-m^2 case 162

Figure 20: Velocity profile of a 10 kg object's collision with a stiff wall 163

Figure 21: Acceleration profile of a 10 kg object's collision with a stiff wall 163

LIST OF TABLES

II. THE UTILITY OF ABBREVIATED LINEAR AND NONLINEAR ROBOT MODELS

Table I: Comparison of the computational burden of PUMA 560 inverse dynamics and linearized dynamics 46

Table II: Maximum joint torques for the PUMA 560 reported by Armstrong, et al. (1986) 46

Table III: Statistics showing differences between abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a, 10a-3c 47

Table IV: Statistics showing differences between abbreviated and un-abbreviated nonlinear models for randomly varying joint angles with joint velocities at ±10 rad/sec and joint accelerations at ±100 rad/sec^2 47

Table V: Comparison of abbreviated and un-abbreviated linear system matrices 48

Table VI: Comparison of abbreviated and un-abbreviated linear system matrices 49

Table VII: Comparison of abbreviated and un-abbreviated linear system matrices 50

Table VIII: Eigenvalues for linear systems in Tables V-VII 51

III. THE USE OF SYMBOLIC, COMPUTER GENERATED CONTROL LAWS IN HIGH SPEED TRAJECTORY FOLLOWING: ANALYSIS AND EXPERIMENTS

Table I: Linear system matrices for robustness analysis 104

Table II: Comparison of the closed loop eigenvalues for the pole placement method in this paper and traditional computed torque 105

IV. A CONTROL SYSTEM ARCHITECTURE FOR ROBOTS USED TO SIMULATE DYNAMIC FORCE AND MOMENT INTERACTION BETWEEN HUMANS AND VIRTUAL OBJECTS

Table I: Control system execution times 164

Table II: Statistics for mass and inertia results in Figures 21-22 and Figures 28-29 164

ACKNOWLEDGEMENTS

I would like to thank my advisor, mentor, and friend Jim Bernard for his patient support throughout my graduate career. I could not imagine finding a better major professor and role model to work for. He is probably the busiest person at this university, yet he always seems to find time for his students when they need it. I have learned immeasurably not only from his engineering and technical brilliance but also from his mastery of leadership and interpersonal skills.

I would also like to thank Professors Luecke, McConnell, Oliver, Seversike, Vanderploeg, and Wacker for serving on my committee. Dr. Luecke was especially helpful in the many robotics discussions that we had. A special thanks goes to Dr. McConnell for standing in on my committee on short notice.

I thank my friends and colleagues in the Carver and Visualization labs for the many technical, philosophical, and sports related talks that we've had. A special thanks to Jim Troy for our many discussions in dynamics and controls and to my basketball partner, Jim Gruening.

The work for this dissertation was funded in part by the Iowa Center for Emerging Manufacturing Technology, the Iowa Space Grant Consortium, and the NASA Graduate Student Researchers Program. A special note of thanks goes to my colleagues at NASA's Marshall Space Flight Center.

Finally, I thank my parents and my sister for their support. I especially thank my wife, Mary, for her love, patience, and encouragement throughout the course of this research.

I. GENERAL INTRODUCTION

A. Overview

Virtual reality (VR) research has catapulted from its obscure beginnings among a handful of computer scientists towards a level of popularity that has reached into the mainstream media. The popularity of the words "virtual reality" has led to their misuse when describing applications which really aren't "virtual" at all. Usually, virtual reality refers to a level of sophisticated computer visualization that is so detailed that users of the technology cannot discern the computer generated environment from the "real world". In the most general case, however, a VR application should include not only visual feedback, but also interaction with other senses, including touch, sound, and smell.

The current state of the art in VR technology is still fairly crude for two reasons. First of all, most applications are graphics-only in that only one sense, that of sight, is taken into account. Secondly, even the most complex VR systems in existence today do not fully "immerse" subjects because humans are still physically aware that their surroundings are computer generated. This is due to a number of factors including computational restrictions which limit graphics complexity and by not accounting for the other senses that humans routinely use to interrogate the environments around them. Therefore, the technology required to create virtual reality systems which are indeed "virtually real" is still several years away.

This dissertation seeks to advance the state of the art in VR technology by developing tools and techniques which help augment visual systems with the sense of touch or "feel". Common phrases used in this area are "tactile force feedback" dealing with finger tip force generation, "haptic force feedback" dealing with hand force

generation, or generically as "force reflective feedback". Researchers have only recently begun to study this aspect of VR technology. Most previous work has emphasized force generation when a subject encounters a stiff virtual object such as a wall. However, a need for accurate dynamic force simulation in virtual environments has become apparent.

NASA, which is a primary contributor and consumer of VR research, has concluded that dynamic force reflective feedback in a VR system offers an effective low-cost alternative in training astronauts for space-related tasks which they may be asked to perform. Current training procedures include the use of full-scale underwater mock-ups which are very costly to operate and maintain. Because of the dramatic decrease in hardware costs associated with virtual reality technology, NASA believes computer generated training environments which make use of force reflection technology can offer effective training scenarios at a greatly reduced cost. The recent Hubble Space Telescope repair mission offers a vivid example. Astronauts were attached to the space shuttle's robotic arm and employed as human "end effectors". They were required to remove old components and insert new ones into the telescope. Maneuvering these components was often tedious, especially when the objects were translating and rotating relative to one another.

Through force reflection, astronauts could be trained to handle payloads more proficiently in zero g environments. Although weightless in space, many payloads have substantial inertia. The dynamics of handling large objects in a weightless environment is a new experience to most astronauts and requires a considerable amount of training. Furthermore, underwater mock-ups of these scenarios often do not recreate these dynamics accurately.

This dissertation provides an analysis of the components of robotic control systems for simulating the dynamic interaction between humans and virtual objects. However, many of the concepts presented here are quite general and applicable to a wide range of robotics applications, such as those in manufacturing and process automation. Chapter II presents an analysis of the utility of abbreviated linear and nonlinear dynamic models of robotic manipulators. The results presented there underpin the rest of this dissertation because they demonstrate the computational savings that can be obtained with abbreviated models while retaining a satisfactory level of complexity. Chapter III takes the results given in Chapter II to develop a computationally efficient robotic control system which is robust both in terms of stability and performance. The use of computer generated control software is shown to greatly reduce the time required for control system development over a wide range of robots. Analytical and test results are given for a PUMA 560 manipulator which demonstrate these techniques. Finally, Chapter IV utilizes the results in Chapters II and III as the foundation for developing a force reflective feedback system utilizing a PUMA 560 as the force transmitter. Equations of motion are derived for a six degree of freedom virtual object which is manipulated by a human operating in a virtual environment. A force transducer on the robot's end effector measures input forces and the control software calculates the trajectory of the virtual object subject to these forces. Inverse kinematic calculations are then performed to convert the Cartesian trajectory of the virtual object, especially its acceleration, into the joint space of the robot. Feedforward and feedback loops ensure that the PUMA end effector follows the virtual object's trajectory while maintaining the proper force balance.

B. Dissertation Organization

Chapters II, III, and IV of this dissertation are written as technical papers to be submitted to three scholarly journals. Although these chapters can stand alone, each makes use of the previous chapter's results.

Chapter II will be submitted to an ASME journal and follows the prescribed format per the ASME Manual MS-4, *An ASME Paper.* Figures and tables are kept separate from the text and appear after the references. Chapters III and IV will be submitted to two IEEE journals and follow the format given in the IEEE publication, *Information for IEEE Transactions, Journals, and Letters Authors.* Again, figures and tables are printed separately from the main text and are given after the references.

Chapters II, III, and IV of this dissertation contain independent abstracts, introductions, conclusions, bibliographies, and appendices. Each also contains a review of the literature which is relevant to that particular section. Chapter V presents the dissertation's general conclusions.

II. THE UTILITY OF ABBREVIATED LINEAR AND NONLINEAR ROBOT MODELS

A paper to be submitted to
The ASME Journal of Dynamic Systems, Measurement, and Control

C.L. Clover
Iowa State University

Abstract

This paper demonstrates the utility of abbreviated linear and nonlinear robot models conceived for use in design, sensitivity analysis, and control.

The complexity of the nonlinear and linear equations for most higher degree of freedom robots is considerable. Prior research has focused on reducing the number of computations in these models for use in real-time applications. A few researchers have analyzed abbreviated symbolic nonlinear robot models where terms in the equations which remain small in magnitude are neglected. These models have been shown to be very efficient in terms of the computational effort required. However, the closeness of these abbreviated nonlinear models to their un-abbreviated counterparts has not been evaluated satisfactorily. Furthermore, previous efforts have not addressed the utilization of abbreviated linear models. This paper seeks to explore this area by comparing abbreviated linear and nonlinear results with un-abbreviated results. Appendices provide explicit linear and nonlinear abbreviated equations for the PUMA 560.

List of Nomenclature

$A(t)$ = state space system matrix
$B(t)$ = state space input matrix
B = NxN(N-1)/2 Coriolis matrix
C = NxN joint velocity sensitivity matrix
D = NxN centrifugal matrix
F_{ext} = external force/moment vector
F_{i+1} = inertial force acting at the center of mass of link i+1
G = Nx1 gravity vector
I_{i+1} = 3x3 inertia tensor of joint i+1
J = NxN Jacobian
K = NxN joint position sensitivity matrix
M = NxN inertial matrix
N = number of degrees of freedom
N_{i+1} = inertial torque acting at the center of mass of link i+1
$P_{C_{i+1}}$ = position of the center of mass of joint i+1
P_{i+1} = position of joint i+1 w.r.t joint i
R = rotational transformation matrix
V = Nx1 friction vector
a = link length
d = link offset
f_{i+1} = force exerted on link i by link i-1
i = joint index
m_{i+1} = mass of joint i+1
n_{i+1} = torque exerted on link i by link i-1
$\dot{v}$ = joint linear acceleration
$\dot{v}_C$ = joint center of mass linear acceleration
$\hat{Z}$ = unit vector along the z-axis
q = joint coordinate (translational or rotational)
t = time
$u(t)$ = state space input vector
$x(t)$ = state variable
α = link twist angle
δ = a perturbation quantity
θ = joint angle
τ = Nx1 vector of actuator forces or torques
ω = joint angular velocity
$\dot{\omega}$ = joint angular acceleration
$*$ = a nominal quantity

1. Introduction

Because the kinematics and dynamics of multibody systems are very complex, computers have long been an essential tool in the design and control of robotic manipulators. This is due to the immense complexity involved with modeling the kinematics and dynamics of these devices. Indeed, since hand-deriving explicit equations of motion for multibody systems is a formidable task, symbolic processing software becomes the only practical option. Consequently, the robotics literature describes many useful symbolic processing programs which have been used successfully to develop closed-form equations for the kinematics,

inverse kinematics, dynamics, and linearized dynamics of robotic manipulators. In many cases, these closed-form expressions are conceived for use in real-time control applications because they generally offer a more efficient calculation process than numerical methods.

An interesting application for symbolic processing software is the linearization of manipulator equations of motion. Linearized equations of motion are even more complex and difficult to derive by hand than the associated nonlinear equations. This is because chain rule differentiation required in the linearization process leads to a formulation with many more terms than the original nonlinear formulation. Thus, symbolic processing is again the only practical alternative for obtaining a closed-form linearization of higher degree of freedom systems.

Section 2 reveals that the literature offers many justifications for deriving linearized equations of motion. Control system design, sensitivity analysis, and parameter identification are a few of the areas commonly mentioned. However, most of the closed-form examples cited in the literature are less complex mechanical systems with three or fewer degrees of freedom. This is understandable in view of the computational requirements for generating closed-form linearized equations of motion. For example, the computing power required to symbolically linearize the equations of a six degree of freedom manipulator such as the PUMA 560 has become readily available only in the last five years or so.

One question that appears yet to be answered is the utility of linearized robotics equations of motion in real-time control and analysis applications. Section 2 discusses examples of numerical algorithms that linearize equations of motion. The complexity of these algorithms is usually measured in terms of the number of floating point operations (multiplications and additions) required to compute the model. Closed-form solutions are

usually more efficient than numerical solutions in terms of the number of floating point calculations because unnecessary computations (such as multiplication by or addition of zero) are more readily eliminated. However, symbolic equations for large degree of freedom systems still require a considerable amount of computation which may be prohibitive in practical real-time applications.

The purposes of this paper are two-fold. The first purpose is to present an efficient set of nonlinear and linearized (about an arbitrary trajectory) equations of motion for the PUMA 560 manipulator including all six degrees of freedom. The second purpose of the paper is to illustrate the utility of the linear and nonlinear PUMA equations presented here with empirical analyses incorporating various trajectories and configurations.

The next section presents a literature survey outlining the salient issues associated with symbolic and numerical computation of nonlinear and linear robot dynamics. Subsequent sections present an analysis of the explicit nonlinear and linear equations for the PUMA 560 robot.

2 Literature Survey

2.1 Overview. Kircanski, et al. (1993) and Lieh (1994) give summaries of computer packages that have been presented for modeling multibody systems using either numerical or symbolic approaches. Other authors who have made contributions in the symbolic derivation of robot equations of motion include Pan and Sharp (1988) and Yin and Yuh (1989). More recent software packages that include symbolic processing of multibody systems are AutoSim (Sayers,1991), which is a general purpose multibody dynamics program, and Robotica (Nethery and Spong, 1994) which is tailored specifically for robotics systems.

A discussion of the utility of symbolic computation in the robotics field is given by

Rentia and Vira (1991). Areas where symbolic processing software has or could be utilized include forward kinematics, inverse kinematics, computer-aided design, and trajectory planning. De Jager (1993) uses the software program Maple (Char et al.,1990) to study the viability of symbolic computation in nonlinear control. This study concluded that the size and scope of the control problem has a great deal to do with the viability of symbolic processing software. Large problems may require vast amounts of computer memory to generate the necessary equations. Furthermore, general purpose software programs such as Maple often have problems with simplifying very large expressions. For example, we have seen cases where a very large expression might contain a $\cos^2(\theta)$ and a $\sin^2(\theta)$ term separated by a large number of other terms which Maple was not be able to simplify to $\cos^2(\theta)+\sin^2(\theta)=1$. Therefore, it is no surprise that de Jager (1993) calls for more research into algorithms which make the symbolic manipulation of large equations more robust, especially with respect to computer memory requirements.

2.2 Linearization. This section focusses on previous research in the area of computer-automated linearization of the equations of motion of multibody systems. Much of the analysis in this area has been numerical due to the complexity of generating closed-form linearized multibody models. This sub-section will review the relevant linearization literature in chronological order.

Early researchers who discussed the usefulness of linearized robot models include Neuman and Murray (1984). Control system design and trajectory sensitivity analysis were highlighted as important areas where linear robot models are needed. The authors also purport to be the first to introduce dynamic robot model linearization algorithms for symbolic computer implementation. The Q-matrix Lagrangian formulation (Bejczy,1974) is utilized

to symbolically acquire nonlinear and linear manipulator equations of motion. A double pendulum is given as an example.

Murray and Neuman (1986) present a recursive algorithm for computing a linearized robot model based on a first order Taylor series expansion of the well known Newton-Euler recursive formulation (Luh et al.,1980). The authors note that the linearized Newton-Euler recursion is of complexity O(N) as compared to the $O(N^5)$ complexity of their previous Q-matrix Lagrangian formulation. However, the O(N) complexity does not take into account the extra steps required to fill in values for the sensitivity matrices which becomes necessary in control system applications. Murray and Johnson (1989) then extended Murray's earlier work to include screw joints in the linearize Newton-Euler formulation. The authors comment on the importance of linearized models for optimal path planning and nonlinear decoupling control.

Balafoutis and Patel (1989) propose an algorithm for generating linearized robot models in both joint and cartesian space. This algorithm, which can be implemented either symbolically or numerically, calculates the sensitivity matrices in joint space and is of complexity $O(N^2)$. The authors state that for a general six degree of freedom manipulator (N=6), a total of 4226 multiplications and additions is required to compute a linearized model with their algorithm in joint space. They also note that a significant reduction in computation can be achieved by utilizing knowledge of the particular robot that is being analyzed. For example, a six degree of freedom manipulator with joint twist angles of either 0° or ±90° requires only 3466 additions and multiplications to compute. Implicit in the computation of Cartesian sensitivity matrices is the inverse Jacobian written with respect to the tool frame. The first and second derivatives of the Jacobian with respect to time are also required.

Li (1990) introduces an efficient algorithm for deriving the sensitivity matrices of linearized robot models. This method is based on a Lagrangian formulation. Three different algorithms are presented which have $O(N^2)$ complexity. The most efficient method requires a total of 3717 additions and multiplications to calculate the linearized model of a general six degree of freedom robot. A linearized six degree of freedom model with joint twist angles of either 0° or ±90° requires 2592 additions and multiplications to compute. Off-line symbolic processing used to decrease computational requirements for a specific manipulator is also proposed.

Lynch and Vanderploeg (1990) offer a general symbolic formulation for deriving the equations of motion for any open- or closed-loop multibody system. The authors make use of Kane's dynamic equation formulation (Kane and Levinson, 1985) and velocity transformations (Kim and Vanderploeg, 1986) to define the dynamic equations in terms of a minimal set of independent coordinates. The advantages of deriving the linearized equations as explicit functions of an operating point are noted. A triple pendulum eigenvalue problem is presented as an example and the explicit linearized equations are given. The use of QR decomposition to obtain the symbolic linearization of closed-loop, constrained systems is also discussed.

Li (1991) extends his previous work (Li, 1990) to include a Cartesian space formulation. A numerical scheme to obtain the Jacobian and its first and second derivatives with respect to time is presented. These are necessary for converting the equations from joint space to Cartesian space.

Lieh (1991) and Lieh and Haque (1991) give a method for computing the dynamics of constrained, multibody systems using principles of virtual work. Maple was used as the

symbolic processor. A one link flexible robot example (Lieh, 1991), a double pendulum on a vibrating base, and a seven degree of freedom automobile suspension (Lieh and Haque, 1991) are given as examples. Symbolic linearization is presented for the double pendulum example. The nonlinear Coriolis and centrifugal terms are neglected by assuming a nominal trajectory with zero velocity. Also, small motion is assumed by setting $\cos(\theta)=1$ and $\sin(\theta)=\theta$.

Swarup and Gopal (1992,1993) compare various linearization schemes including state space linearization, system identification, rate linearization (where the Coriolis-centrifugal terms are neglected), rate and gravity linearization (where the Coriolis-centrifugal and gravity terms are neglected), and a hybrid rate-state linearization (where, in high velocity regions, a switch is made from rate linearization to state space linearization). A two link manipulator is used as an example. For the trajectory and manipulator considered, the authors assert that rate linearization leads to a satisfactory trade off between computational burden and system performance.

Trom and Vanderploeg (1994) present a numerical approach for the linearization of large multibody dynamic systems containing both open- and closed-loops. Mechanical systems such as automobiles are routinely modeled with 30 degrees of freedom or more. Current software/hardware limitations make deriving and simplifying linearized equations of motion for these models in symbolic form impractical. Typically, large systems are linearized using simple finite difference techniques. The authors note the drawbacks of this method because of the difficulties in choosing perturbation values for the independent variables. In contrast, their linearization scheme makes direct use of the nonlinear multibody formulation of Kim and Vanderploeg (1985). Eigenvalue analyses are presented for a slider-pendulum

example and a 19 degree of freedom 5-axle tractor semi-trailer to demonstrate the method.

In summary, although there has been a lot of valuable research done in this area, there remains a lack of published results discussing the viability of these methods in real-time applications. The number of calculations required to compute nonlinear and linear models of five or six degree of freedom manipulators at the sampling rates commonly found in digital control systems is considerable. One solution to this problem is to develop abbreviated models which require less computation but still retain enough complexity to be useful in control system design.

This paper demonstrates the utility of abbreviated nonlinear and linearized robotics models using the PUMA 560 manipulator as an example.

3 Nonlinear Modeling of the PUMA 560

3.1 Background. This section develops a conceptual framework for nonlinear equations of motion that we will later extend to the linear modeling of the PUMA robot. We begin by examining the basic dynamic model for an N degree of freedom robot written in configuration space as:

$$M(q)\ddot{q} + B(q)[\dot{q}\dot{q}] + D(q)[\dot{q}^2] + V(q,\dot{q}) + G(q) = \tau + J^T(q)F_{ext} \tag{1}$$

Equation (1) can be treated numerically or symbolically. In this paper we are interested primarily in symbolic calculation.

Explicit equations for higher degree of freedom mechanisms are rarely seen in the literature due to their complexity. In the past, a few researchers have presented nonlinear equations for the PUMA 560. Armstrong et al. (1986), Burdick (1986), and Neuman and Murray (1987a) each present their own formulations. These models were generated with "in-

house" software that is not readily available for outside use. The advent of general purpose symbolic processing software such as Maple has helped extend the use of symbolic modeling to the entire robotics community. However, the generality of this software brings with it the need for increased user sophistication when developing specialized robotics applications.

Because we seek to implement our models in real-time control applications, we would like to develop dynamic equations which are as computationally efficient as possible. Symbolic representation lends itself towards the creation of "customized" dynamic models. Customization includes simplification, factorization, and the elimination of unnecessary calculations (such as multiplying by one or zero). Burdick (1986), Neuman and Murray (1987a,1987b), and Toogood (1989) demonstrate substantial savings in computation by customizing robot dynamics models for specific manipulators.

A clever customization algorithm was presented by Armstrong et al. (1986). This dynamic formulation sought to simplify the amount of computation by eliminating calculations using pre-defined "fuzzy" zeroes. Whereas Murray, Neuman, and others eliminated multiplications involving zeroes within and across terms in the dynamic model, Armstrong et al. go a step further by employing significance criteria to eliminate terms that are very small compared with other terms in the model. Developing these significance criteria, however, requires that the explicit robot input parameters be determined either through parameter identification algorithms or by direct measurement.

Whereas Armstrong et al. (1986) directly measured the inertial parameters to use in their model, Izaguirre et al. (1992) use singular value decomposition to identify a minimum parameter set to be used in the inverse dynamics equations. Significant parameters were calculated by eliminating parameters whose torque contributions for a set of trajectories was

less than one percent of the maximum torque. Plots were given showing the closeness of the abbreviated solution to the un-abbreviated solution, but no error statistics were given. Lin and Zhang (1993) present a dimensionless formulation for the inverse dynamics equations and found that this aided in eliminating insignificant terms from the equations.

In the sense that customized robotics models are more efficient, customized models which apply significance criterion to eliminate small terms will always be the most efficient. However, significance methods lose generality because they require numerical parameters to be input into the model. This approach should pose no problems with the analysis or control of a particular robot but may be less desirable with manipulator designs in which explicit parameters may not be known.

Given that simplification of dynamic models based on term by term significance criteria will always reduce the total number of arithmetic operations, we ask this question - what order of complexity is required to maintain a desired level of accuracy in the nonlinear model calculations? This issue has not been satisfactorily addressed in the literature. This section will provide a framework to answer this question by extending the ideas of Armstrong et al. (1986), Izaguirre et al. (1992), and Lin and Zhang (1993).

In order to answer any model complexity questions, it is easiest if we first derive equation (1) in symbolic form. One of the most common numerical methods for deriving this equation as it applies to serial link robots is the recursive Newton-Euler dynamics algorithm (Luh et al., 1980). This method involves propagating joint velocities and accelerations (both linear and angular) outwardly from the base of the robot to the tip to obtain the local forces and moments acting at each link's center of gravity. These forces and moments are then inwardly propagated from the tip to the base to obtain local joint forces and moments. Craig

(1989) presents a slightly revised version of the recursive Newton-Euler scheme (RNE) which uses a modified Denavit-Hartenberg (D-H) parameter set which locates joint coordinate frames at the base of the link instead of at the tip. Figure 1 presents this alternative frame description. (This D-H parameter format was used by Armstrong et al. (1986) and is used throughout this paper.)

The following equations re-present, for convenience, the recursive Newton-Euler formulation using the modified D-H parameter set. For a derivation of these equations, see Craig (1989). Note that i corresponds to joint number.

$$\omega_{i+1} = R^T\omega_i + \dot{\theta}_{i+1}\hat{Z}_{i+1} \tag{2}$$

$$\dot{\omega}_{i+1} = R^T\dot{\omega}_i + R^T\omega_i\times\dot{\theta}_{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\hat{Z}_{i+1} \tag{3}$$

$$\dot{v}_{i+1} = R^T\left(\dot{\omega}_i\times P_{i+1} + \omega_i\times\left(\omega_i\times P_{i+1}\right) + \dot{v}_i\right) \tag{4}$$

$$\dot{v}_{C_{i+1}} = \dot{\omega}_{i+1}\times P_{C_{i+1}} + \omega_{i+1}\times\left(\omega_{i+1}\times P_{C_{i+1}}\right) + \dot{v}_{i+1} \tag{5}$$

$$F_{i+1} = m_{i+1}\dot{v}_{C_{i+1}} \tag{6}$$

$$N_{i+1} = I_{i+1}\dot{\omega}_{i+1} + \omega_{i+1}\times I_{i+1}\omega_{i+1} \tag{7}$$

These equations assume revolute joints. If joint i+1 is prismatic, the following modifications are made:

$$\omega_{i+1} = R^T\omega_i \tag{8}$$

$$\dot{\omega}_{i+1} = R^T\dot{\omega}_i \tag{9}$$

$$\dot{v}_{i+1} = R^T\left(\dot{\omega}_i \times P_{i+1} + \omega_i \times (\omega_i \times P_{i+1}) + \dot{v}_i\right) + 2\omega_{i+1} \times \dot{d}_{i+1}\hat{Z}_{i+1} + \ddot{d}_{i+1}\hat{Z}_{i+1} \tag{10}$$

These inertial torques and forces are then propagated inwardly in order to solve for the joint torques and forces which are required of the actuators. The inward force iterations are as follows.

$$f_i = Rf_{i+1} + F_i \tag{11}$$

$$n_i = N_i + Rn_{i+1} + P_{C_i} \times F_i + P_{i+1} \times Rf_{i+1} \tag{12}$$

Finally, the required actuator torques or forces can be calculated as

$$\tau_i = n_i^T \hat{Z}_i \tag{13}$$

for revolute joints or

$$\tau_i = f_i^T \hat{Z}_i \tag{14}$$

for prismatic joints. Note that

$$R = \begin{bmatrix} c\theta_{i+1} & -s\theta_i & 0 \\ s\theta_{i+1}c\alpha_i & c\theta_{i+1}c\alpha_i & -s\alpha_i \\ s\theta_{i+1}s\alpha_i & c\theta_{i+1}s\alpha_i & c\alpha_i \end{bmatrix} \tag{15}$$

$$P_{i+1} = \begin{bmatrix} a_i \\ -s\alpha_i d_{i+1} \\ c\alpha_i d_{i+1} \end{bmatrix} \tag{16}$$

We can also use equations (2)-(16) to derive equations in symbolic form. Wheeler (1994) has written a Maple procedure which implements these equations. Using Maple, various levels of simplification of the equations can be studied by changing the significance

criterion for which small terms are approximated as zero.

3.2 Case Study - PUMA 560. This section presents a study of the PUMA 560 equations of motion. While we select the PUMA as a case study because of its widespread use among the robotics research community, the principles outlined here are applicable to any manipulator. The PUMA 560 parameters given by Armstrong et al. (1986) are utilized with the exception of the motor two inertia. Corke and Armstrong-Helouvry (1995), in a study of PUMA parameters available in the literature, found that there is a wide discrepancy in the reported values among researchers. However, most of the parameters in Armstrong's 1986 paper were directly measured. Thus, this parameter set was deemed as reasonable as any to use in this study. The exception is an apparently spurious value for the motor two armature inertia which was corrected according to Corke and Armstrong-Helouvry (1995).

We have found that computing the explicit equations for the PUMA 560 dynamic model on a DECstation 5000/240 requires about 8 minutes. No doubt future advances in computer technology will continue to reduce this time.

To determine the amount of complexity required to maintain an acceptable level of agreement between the un-abbreviated model (no significance criteria applied) and an abbreviated model, we first examine the structure of equation (1). Note that the Coriolis, inertial, friction, and centrifugal coefficients are all matrix functions of joint position. Also, the gravity terms are a function of joint position only. It is these matrix elements which have the significance criterion applied in order to simplify the equations of motion. The absolute errors in joint torque calculations coming from neglecting small terms in equation (1) tend to grow at higher levels of joint acceleration and velocity. This implies that the character of the abbreviation will vary from application to application. For example, high speed

applications will generally require a more complex model with less abbreviation than a low speed application for which gravity compensation alone may be sufficient.

This paper assumes the need for accurate calculations for a robot moving at high speed. However, the amount of abbreviation of dynamic models used in high speed applications can still be substantial. Appendix A presents the nonlinear equations for an abbreviated PUMA 560 dynamic model. The external forces and moments have been set to zero for purposes of comparison with previous efforts. Note the similarities between the inertia matrix coefficients presented here and those presented by Armstrong et al. (1986). There are differences in the [1,1] and [2,2] terms however. The [1,1] element presented by Armstrong has a slight error (which the author acknowledges) and the [2,2] element is different due to the change in motor two inertia that we have implemented. The [1,1] element of the Coriolis matrix in Appendix A is also different from Armstrong's. This is due to the same error which caused the difference in the [1,1] element of the inertia matrix.

Other differences between the Appendix A equations and Armstrong's equations are due to a different significance criterion. Armstrong eliminated all terms that are less than 1% as great as the greatest term within the same equation, or 0.1% as great as the largest constant term applicable to the same joint. Implementing term by term significance criterion selection in an automated fashion is not straightforward in Maple. For the equations presented in Appendix A, a "fuzzy" zero tolerance of 0.01 is applied uniformly across all matrix elements to each trigonometric coefficient or constant. This means that all trigonometric coefficients and constants which are less then 0.01 are set equal to zero.

The computational burden of the equations in Appendix A amounts to 59 multiplications and 42 additions when optimized such that like terms (e.g., $\cos(\theta_2)$) are only computed once.

(Note the Appendix A equations are not given in optimized form so that they may be easily read.) Table I demonstrates that when these equations are used in inverse dynamics calculations, there is a substantial reduction in computation compared with other models found in the literature. However, the PUMA equations presented here have utility only if they give similar results to those derived from the un-abbreviated equations. In this paper, we present two of the many computer-based procedures that were utilized in determining the closeness of the abbreviated model to the unabbreviated model. In the first procedure, we slew each joint angle through 360 degrees in 1.5 seconds according to a pre-determined trajectory. These trajectory equations minimize a performance index in terms of jerk according to

$$J(\vec{q}) = \int_0^T \dddot{\vec{q}}^2 dt \tag{17}$$

The equations, as derived by Kyriakopoulus and Saridis (1988) are as follows:

$$q(t) = \left(6\frac{t^5}{T^2} - 15\frac{t^4}{T} + 10t^3\right)\frac{q(T)}{T^3} \tag{18}$$

$$\dot{q}(t) = \left(30\frac{t^4}{T^2} - 60\frac{t^3}{T} + 30t^2\right)\frac{q(T)}{T^3} \tag{19}$$

$$\ddot{q}(t) = \left(120\frac{t^3}{T^2} - 180\frac{t^2}{T} + 60t\right)\frac{q(T)}{T^3} \tag{20}$$

where T is the final time. Note that $q(0)=\dot{q}(0)=\ddot{q}(0)=\dot{q}(T)=\ddot{q}(T)=0$, although a non-zero offset in initial joint angle can be added to equation (18).

Figures 3a-3c show the joint positions, velocities, and accelerations given by equations (18)-(20). Figures 4a-4f show the joint torques required to cause the robot to follow the prescribed trajectory. This is a severe maneuver as the first three joints require torques which are near or exceed the maximum torque capabilities of the PUMA motors. Table II represents the maximum PUMA motor torques given by Armstrong et al. (1986).

Figures 4a-4f indicate that, for this example, the abbreviated model is essentially identical to the un-abbreviated model. Table III presents important details. Note that engineering judgment is required here. It is obvious that one could generate trajectories with joint velocities and accelerations large enough to lead to substantial disagreement between the abbreviated and un-abbreviated models. However, if these maneuvers require actuator forces and torques which the manipulator is incapable of, then these trajectories are of no consequence. In other words, because there is a finite limit to the amount of force or torque a typical motor can generate, trajectories which cause these limits to be exceeded do not lead to a comparison between abbreviated and un-abbreviated dynamic models which is reasonable. Having said this, however, we present a second procedure that can be used for comparing abbreviated and un-abbreviated models which uses trajectories that are unachievable by the PUMA.

Figures 5a-5f show a comparison between the Appendix A equations and the un-abbreviated equations. In contrast to the previous trajectory analysis, here the magnitudes of all joint velocities are set to 10 rad/sec and all joint accelerations are set to 100 rad/sec^2. The algebraic sign on the velocities and accelerations is random. Joint angles range from zero to 360 degrees and are also random. As the trajectory is randomly varied over time, problem points will be identified where the torques calculated by the un-abbreviated model and the

abbreviated model are substantially different. Table IV gives more details.

For real-time computed torque control applications, we are satisfied that the equations in Appendix A are suitably similar to the un-abbreviated equations. The reduction in computational burden is well worth the slight differences between models. These equations are also adequate for design analyses, such as bearing or motor sizing.

Finally, note that the two procedures presented here are based on an intuitive sense of the system and its potential uses. We do not claim that the model in Appendix A is appropriate for all applications. Certainly a much simpler model (e.g. gravity compensation only) could be derived for low speed applications. Likewise, more demanding applications might require further complexity. The point here is that significance criteria can have a dramatic effect on the complexity of the resulting equations and that these criteria should reflect the purposes of the model.

4 Linear Modeling of the PUMA 560

As noted in Section 2, the robotics literature discusses the utility of linear dynamic models for control system design and trajectory sensitivity analysis. It also noted that relatively little work (when compared with nonlinear modeling) has been done with linear modeling of existing "real world" robots. This section seeks to address this important issue by performing a linear analysis of the PUMA 560 as a case study. Section 4 starts by giving a set of abbreviated, linearized PUMA equations and follows with an eigenvalue analysis to determine the utility of these equations.

4.1 Developing the Equations. In order to linearize the equations of motion we utilize a standard Taylor series expansion of a nonlinear function about a nominal trajectory and neglect all but the first order terms according to

$$f(q,\dot{q},\ddot{q}) = f(q^*,\dot{q}^*,\ddot{q}^*) + \left(\frac{\partial f}{\partial q}|_{q^*}\right)\delta q + \left(\frac{\partial f}{\partial \dot{q}}|_{\dot{q}^*}\right)\delta\dot{q} + \left(\frac{\partial f}{\partial \ddot{q}}|_{\ddot{q}^*}\right)\delta\ddot{q} \qquad (21)$$

One option for generating the linear equations would be to differentiate, term by term, all the elements of the nonlinear equations with respect to the nominal trajectory according to equation (21). However, we have found that implementing this technique in Maple becomes too memory intensive for higher degree of freedom manipulators. In particular, the size of the linear model becomes such that Maple cannot adequately perform simplifications and optimization. An alternative approach is to develop the equations with recursive techniques which enable Maple to perform simplifications after each step of the process. This dramatically reduces the memory requirements for deriving the equations. Note again, however, that Maple's trigonometric simplification process is by no means perfect when asked to deal with very large expressions. But this problem is minimized when dealing with abbreviated models because so many of the terms are pre-eliminated.

The formulation we have chosen for developing linearized equations of a general manipulator is based upon linearizing the recursive Newton-Euler iterations found in equations (2)-(16) by Taylor series expansion. Murray and Neuman (1986) presented such a recursive Newton-Euler scheme which calculates linearized equations about any trajectory by algebraically canceling the nominal components of the trajectory and preserving terms which are linear with respect to the perturbation variables. However, their equations are based on a different link frame description than the one used in this paper. As described earlier, the Appendix A nonlinear equations are derived based on a modified D-H parameter set since Armstrong, at al. (1986) use this format for their parameter set. Therefore, it is more convenient to use this format for developing the linearized equations than those given

by Murray and Neuman (1986). The following equations are the linearized Newton-Euler dynamic robot model (using a first order expansion) based on the modified D-H parameter set given in Craig (1989).

$$\delta\omega_{i+1} = R^T\delta\omega_i + QR^T\omega_i\delta\theta_{i+1} + \delta\dot{\theta}_{i+1}\hat{Z}_{i+1} \tag{22}$$

$$\begin{aligned}\delta\dot{\omega}_{i+1} = {} & R^T\delta\dot{\omega}_i + R^T\delta\omega_i\times\dot{\theta}_{i+1}\hat{Z}_{i+1} + R^T\omega_i\times\delta\dot{\theta}_{i+1}\hat{Z}_{i+1} \\ & + \left(QR^T\dot{\omega}_i + QR^T\omega_i\times\dot{\theta}_{i+1}\hat{Z}_{i+1}\right)\delta\theta_{i+1} + \delta\ddot{\theta}_{i+1}\hat{Z}_{i+1}\end{aligned} \tag{23}$$

$$\begin{aligned}\delta\dot{v}_{i+1} = {} & R^T\left(\delta\dot{\omega}_i\times P_{i+1} + \delta\omega_i\times\left(\omega_i\times P_{i+1}\right) + \omega_i\times\left(\delta\omega_i\times P_{i+1}\right) + \delta\dot{v}_i\right) + \\ & QR^T\left(\dot{\omega}_i\times P_{i+1} + \omega_i\times\left(\omega_i\times P_{i+1}\right) + \dot{v}_i\right)\delta\theta_{i+1}\end{aligned} \tag{24}$$

$$\delta\dot{v}_{C_{i+1}} = \delta\dot{\omega}_{i+1}\times P_{C_{i+1}} + \delta\omega_{i+1}\times\left(\omega_{i+1}\times P_{C_{i+1}}\right) + \omega_{i+1}\times\left(\delta\omega_{i+1}\times P_{C_{i+1}}\right) + \delta\dot{v}_{i+1} \tag{25}$$

$$\delta F_{i+1} = m_{i+1}\delta\dot{v}_{C_{i+1}} \tag{26}$$

$$\delta N_{i+1} = I_{i+1}\delta\dot{\omega}_{i+1} + \delta\omega_{i+1}\times I_{i+1}\omega_{i+1} + \omega_{i+1}\times I_{i+1}\delta\omega_{i+1} \tag{27}$$

where,

$$Q = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{28}$$

such that

$$\frac{\partial R}{\partial\theta} = RQ^T = -RQ \tag{29}$$

These equations assume revolute joints. If joint $i+1$ is prismatic, the following modifications are made:

$$\delta\omega_{i+1} = R^T\delta\omega_i \tag{30}$$

$$\delta\dot{\omega}_{i+1} = R^T\delta\dot{\omega}_i \tag{31}$$

$$\begin{aligned}\delta\dot{v}_{i+1} = R^T(&\delta\dot{\omega}_i\times P_{i+1} + \dot{\omega}_i\times P_{D_{i+1}}\delta d_{i+1} + \delta\omega_i\times(\omega_i\times P_{i+1}) + \omega_i\times(\delta\omega_i\times P_{i+1}) + \omega_i\times\left(\omega_i\times P_{D_{i+1}}\right)\delta d_{i+1} \\ &+ \delta\dot{v}_i) + 2\delta\omega_{i+1}\times\dot{d}_{i+1}\hat{Z}_{i+1} + 2\omega_{i+1}\times\delta\dot{d}_{i+1}\hat{Z}_{i+1} + \delta\ddot{d}_{i+1}\hat{Z}_{i+1}\end{aligned} \tag{32}$$

where,

$$P_{D_{i+1}} = \begin{bmatrix} 0 \\ -s\alpha_i \\ c\alpha_i \end{bmatrix} \tag{33}$$

The linearized inward iterations then become

$$\delta f_i = R(\delta f_{i+1} - Qf_{i+1}\delta\theta_{i+1}) + \delta F_i \tag{34}$$

$$\begin{aligned}\delta n_i = \delta N_i + R\delta n_{i+1} - RQn_{i+1}\delta\theta_{i+1} + P_{C_i}\times\delta F_i + P_{i+1}\times R\delta f_{i+1} + \\ P_{i+1}\times(-RQf_{i+1})\delta\theta_{i+1}\end{aligned} \tag{35}$$

or for prismatic joints

$$\delta n_i = \delta N_i + R\delta n_{i+1} + P_{C_i}\times\delta F_i + P_{i+1}\times R\delta f_{i+1} + \left(P_{D_{i+1}}\times Rf_{i+1}\right)\delta d_i \tag{36}$$

such that the perturbation actuator torques and forces can be calculated respectively as

$$\delta\tau_i = \delta n_i^T\hat{Z}_i \tag{37}$$

or

$$\delta\tau_i = \delta f_i^T\hat{Z}_i \tag{38}$$

These equations can be implemented in Maple to derive linear symbolic equations of

motion. We verified the accuracy of our symbolic implementation of the Maple linearization procedures by comparing it to two different numerical formulations. First, the linear sensitivity matrices were generated by applying finite difference techniques about a series of nominal trajectories. Also, Equations (22)-(38) were numerically solved for the sensitivity matrices by utilizing an analogous approach to that presented by Walker and Orin (1982) for solving nonlinear robot dynamics. All three methods yielded nearly identical results.

The symbolic linear PUMA 560 equations require 20.8 megabytes of memory to compute. This compares with 4.7 megabytes for the nonlinear model on a DECstation 5000/240. A major obstacle in the past has been the vast amount of computer power required to generate these equations for higher degree of freedom robots.

There is usually a significant discrepancy in size among terms in the linearized equations just as there is in the nonlinear equations. This implies abbreviated linear models can be developed which offer good agreement with the un-abbreviated linear model at a fraction of the computational burden. Appendix B presents an abbreviated, linear PUMA 560 model. Table II compares the calculations required for this abbreviated model with other linear models found in the literature. These equations can be given in the typical 2nd order linear format as

$$\delta\tau = M(q^*)\delta\ddot{q} + C(q^*,\dot{q}^*)\delta\dot{q} + K(q^*,\dot{q}^*,\ddot{q}^*)\delta q \tag{39}$$

Note that the structure of the linearized inertia matrix is identical to the structure of the nonlinear model. Also, note the dependence of the damping and stiffness matrices on the nominal trajectory about which the equations are linearized.

Tables V-VII show examples of the differences between the un-abbreviated linear model

and the equations given in Appendix B at three different trajectory points. Note the damping matrix is diagonal (due to the friction terms) when the nominal joint velocities are zero. Also, note that the entries in the first and sixth columns of the stiffness matrix of the unabbreviated model are identically zero. This is because the first and sixth joints of the PUMA do no work in the gravity field. Tables V-VII reveal that as the nominal trajectory velocities and accelerations are increased, the off-diagonal terms begin to diverge. This makes intuitive sense because the nonlinear models also begin to diverge at higher speeds. Again we ask the question - what order of complexity is required in order to maintain a desired level of accuracy in the linear model calculations? This question is much harder to answer than it was in the nonlinear case. Again, the answer is application specific. But high speed vs. low speed is not the only issue. Other issues involve whether or not the linear model will be used for control system design, trajectory sensitivity analysis, or optimal path planning. Presumably, large percentage errors in very small terms are unimportant. Tables V-VII demonstrate that the Appendix B equations are in close agreement with the largest matrix elements.

A primary reason for deriving linear models has been to aid in control system design. Because the inertial matrix, $M(\theta^*)$, is non-singular for robotic manipulators, we can convert equation (39) into state space form which can be expressed in traditional form as,

$$\frac{d}{dt}\begin{bmatrix}\delta q\\ \delta\dot{q}\end{bmatrix} = \begin{bmatrix}0 & I\\ -M^{-1}K & -M^{-1}C\end{bmatrix}\begin{bmatrix}\delta q\\ \delta\dot{q}\end{bmatrix} + \begin{bmatrix}0\\ M^{-1}\end{bmatrix}\delta\tau \tag{40}$$

or

$$\delta\dot{x} = A(t)\delta x(t) + B(t)\delta u(t) \tag{41}$$

These equations form the basis of an eigenvalue analysis in the following section.

4.2 Eigenvalue Analysis. One way to compare differences between abbreviated and un-abbreviated linear models is to examine their eigenvalues. This section analyzes the differences in the open-loop eigenvalues between the un-abbreviated linear PUMA model and the abbreviated model presented in Appendix B. Figures 6a-6c, along with Table VIII, present eigenvalue results obtained from the linearized systems found in Tables V-VII. These results are representative of a number of runs that were performed. Note the differences in eigenvalues at the high speed trajectory points. Also, note that these differences occur with the eigenvalues that are close to the origin. These results represent "worst case scenarios", ie., examples are being shown where the agreement in eigenvalues between the un-abbreviated and simplified linear models is at its worst.

An obvious question from a control system analysis point of view is: What are the implications of eigenvalue mismatches between the un-abbreviated and abbreviated linearized models? Intuitively, it seems clear that the eigenvalues that are the furthest to the right will always dominate the system response characteristics. Thus, matching the dominant eigenvalues as in Figures 6a-6c and Table VIII would seem promising. We take this logic a step further by showing time response plots (Figures 7a-7f) of the second order system given in Table VII. Figures 7a-7f assume a linear time-invariant system with very small initial conditions corresponding to perturbations in the real system. Note that the responses are very similar out to 0.5 seconds. However, as the system moves away at high speed from the nominal point in the trajectory about which the linearization took place, the matrices in Tables V-VII (which are state dependent) are no longer valid and a re-linearization about a new operating point is required. In a control system application, feedback is added to the

system in an attempt to drive these perturbations to zero using fast sampling rates. Therefore, we believe the abbreviated linear model is adequate. This assertion has been further verified with simulation and experimental results (Clover, 1996).

5 Discussions and Conclusions

This paper has examined the utility of abbreviated, symbolic modeling of the PUMA 560. It demonstrated that abbreviated models yield inverse dynamic calculations that maintain a high level of agreement with the non-abbreviated model. The utility of abbreviated linear models was also studied. This turns out to be a more challenging task. Engineering judgment is required to determine the degree to which the abbreviated and un-abbreviated models should agree in areas such as trajectory sensitivity analysis or optimal path planning. These judgments are application dependent (ie, high speed vs. low speed).

Our main interest is in control system design and we have found that the abbreviated, linearized PUMA model presented here is adequate. The computational savings gained by using abbreviated models, both linear and non-linear, is substantial and important in real-time applications.

Note that the results of this paper are parameter, and therefore manipulator dependent. No claims are made for the validity of these conclusions with respect to other manipulator designs. However, intuition tells us that these conclusions probably are valid for many manipulators that are in use today. Future work will involve deriving abbreviated linear and non-linear models for other robots to verify whether or not this is the case.

References

Armstrong, B., Khatib, O., and Burdick, J., 1986, "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 1986, pp. 510-518.

Balafoutis, C.A., and Patel, R.V., 1989, "Linearized Robot Models in Joint and Cartesian Spaces," *Transactions of the CSME*, Vol 13., No. 4, pp. 103-111.

Bejczy, A.K., 1974, "Robot Arm Dynamics and Control," Technical Memo 33-669, Jet Propulsion Laboratory, February 1974.

Burdick, J.W., 1986, "An Algorithm for Generation of Efficient Manipulator Dynamic Equations," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 1986, pp. 212-218.

Char, B.W., Geddes, K.O., Gonnet, G.H., and Watt, S.M., 1990, *Maple Reference Manual*, 5th edition, Waterloo Maple Publishing, Waterloo, Canada.

Clover, C.L., 1996, "The Use of Symbolic, Computer Generated Control Laws in High Speed Trajectory Following: Analysis and Experiments," to be submitted to *IEEE Transactions on Robotics and Automation.*

Corke, P.I., and Armstrong-Helouvry, B., 1995, "A Meta-Study of PUMA 560 Dynamics: A Critical Appraisal of Literature Data," *Robotica*, Vol. 13, pp. 253-258.

Craig, John J., 1989, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, Reading, MA.

de Jager, B., 1993, "The Use of Symbolic Computation in Nonlinear Control: Is it Viable?," *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, TX, December 1993, pp. 276-281.

Izaguirre, A., Hashimoto, M., Paul, R.P., and Hayward, V., 1992, "A New Computational Structure for Real-Time Dynamics," *The International Journal of Robotics Research*, Vol. 11, No. 4, pp. 346-361.

Kane, T.R., and Levinson, D.A., *Dynamics: Theory and Applications*, Mcgraw-Hill, New York, NY.

Kim, S.S., and Vanderploeg, M.J., 1986, "A General and Efficient Method for Dynamic Analysis of Mechanical Systems Using Velocity Transformations," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 108, No. 2, pp. 176-182.

Kircanski, N., Vukobratovic, M., Timcenko, A., and Kircanski, M., 1993, "Symbolic Modeling in Robotics: Genesis, Application, and Future Prospects," *Journal of Intelligent and Robotic Systems*, Vol. 8, No. 1, pp. 1-19.

Kyriaopoulos, K.J., and Saridis, G.N., 1988, "Minimum Jerk Path Generation," *Proceedings of the IEEE 1988 International Conference on Robotics and Automation*, Philadelphia, PA, April 1988, pp. 364-369.

Li, C.J., 1990, "A new Method for Linearization of Dynamic Robot Models," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol 20., No. 1, pp. 2-17.

Li, C.J., 1991, "Linearized Dynamic Models of Robot Manipulators in Cartesian Space," *Journal of Robotic Systems*, Vol. 8, No. 1, pp. 93-115.

Lieh, J., 1991, "An Alternative Method to Formulate Closed-Form Dynamics for Elastic Manipulators Using Symbolic Process" *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 2369-2374.

Lieh, J., and Haque, I., 1991, "Symbolic Closed-Form Modeling and Linearization of Multibody Systems Subject to Control," *ASME Journal of Mechanisms, Transmissions and Automation in Design*, Vol. 113, No. 2, pp. 124-132.

Lieh, J., 1994, "Computer-Oriented Closed-Form Algorithm for Constrained Multibody Dynamics for Robotics Applications," *Mechanism and Machine Theory*, Vol. 29, No. 3, pp. 357-371.

Lin, Y.J., and Zhang, H.Y., 1993, "Simplification of Manipulator Dynamic Formulations Utilizing a Dimensionless Method," *Robotica*, Vol. 11, pp. 139-147.

Luh, J.Y.S., Walker, M.W., and Paul, R.P., 1980, "On-Line Computational Scheme for Mechanical Manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 102, No. 2, pp. 69-76.

Lynch, A.G., and Vanderploeg, M.J., 1990, "A Symbolic Formulation for Linearization of Multibody Equations of Motion," *Proceedings of the 1990 ASME International Computers in Engineering Conference and Exposition*, Boston, MA, August 1990, Vol. 1, pp. 201-207.

Murray, J.J., and Johson, D.W., 1989, "The Linearized Dynamic Robot Model: Efficient Computation and Practical Applications," *Proceedings of the 28th Conference on Decision and Control*, Tampa, FL, December 1989, pp. 1659-1664.

Murray, J.J., and Neuman, C.P., 1986, "Linearization and Sensitivity Models of the Newton-Euler Dynamic Robot Model," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 108, No. 3, pp. 272-276.

Nethery, J.F., and Spong, M.W., 1994, "Robotica: A Mathematica Package for Robot Analysis," *IEEE Robotics and Automation Magazine*, Vol. 1, No. 1, pp. 13-22.

Neuman, C.P. and Murray, J.J., 1984, "Linearization and Sensitivity Functions of Dynamic Robot Models," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 14, No. 6, pp. 805-818.

Neuman, C.P., and Murray, J.J., 1987a, "The Complete Dynamic Model and Customized Algorithms of the PUMA Robot, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 17, No. 4, pp. 635-641.

Neuman, C.P. and Murray, J.J., 1987b, "Customized Computational Robot Dynamics," *Journal of Robotic Systems*, Vol. 4, No. 4, pp. 503-526.

Pan, D., and Sharp, R.S., 1988, "Automatic Formulation of Dynamic Equations of Motion of Robot Manipulators," *Proceedings of the Institution of Mechanical Engineers: Part C, Mechanical Engineering Science*, Vol. 202, No. C6, pp. 397-403.

Rentia, N., and Vira, N., 1991, "Why Symbolic Computation in Robotics," *Computers in Industry*, Vol. 17, No. 1, pp. 49-62.

Sayers, M.W., 1991, "Symbolic Computer Language for Multibody Systems," *Journal of Guidance, Control, and Dynamics*, Vol. 15, pp. 1153-1163.

Swarup, A., and Gopal, M., 1992, "Linearized Dynamic Models for Trajectory Control of Robot Manipulators," *Journal of Robotic Systems*, Vol. 9, No. 4, pp. 455-459.

Swarup, A., and Gopal, M., 1993, "Comparative Study on Linearized Robot Models," *Journal of Intelligent and Robotic Systems*, Vol. 7, No. 3, pp 287-300.

Toogood, R.W., 1989, "Efficient Robot Inverse and Direct Dynamics Algorithms Using Micro-computer Based Symbolic Generation," *Proceedings of the 1989 IEEE Conference on Robotics and Automation*, Scottsdale, AZ, pp. 1827-1832.

Trom, J.D., and Vanderploeg, M.J., 1994, "Automated Linearization of Nonlinear Coupled Differential and Algebraic Equations, *Journal of Mechanical Design*, Vol. 113, No. 2, pp. 429-436.

Walker, M.W., and Orin, D.E., 1982, "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 104, No. 3, pp. 205-211.

Wheeler, G.T., 1994, *Maple™ procedure which derives the equations of motion for any robot given the D-H joint variables and link parameters*, Walla Walla College, College Place, WA, June 1994.

Yin, S., and Yuh, J., 1989, "Efficient Algorithm for Automatic Generation of Manipulator Dynamic Equations," *Proceedings of the 1989 IEEE Conference on Robotics and Automation*, Scottsdale, AZ, pp. 1812-1817.

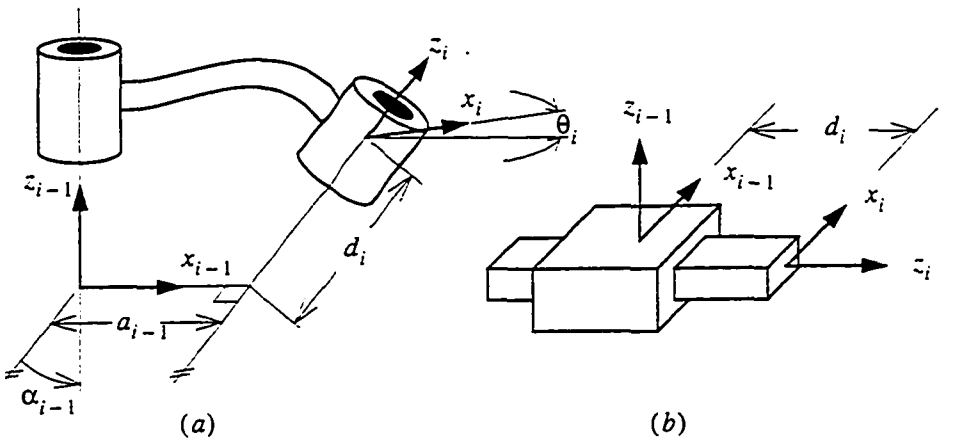

Figure 1: D-H parameter descriptions between link i and link i-1 for (a) a revolute joint and (b) a prismatic joint

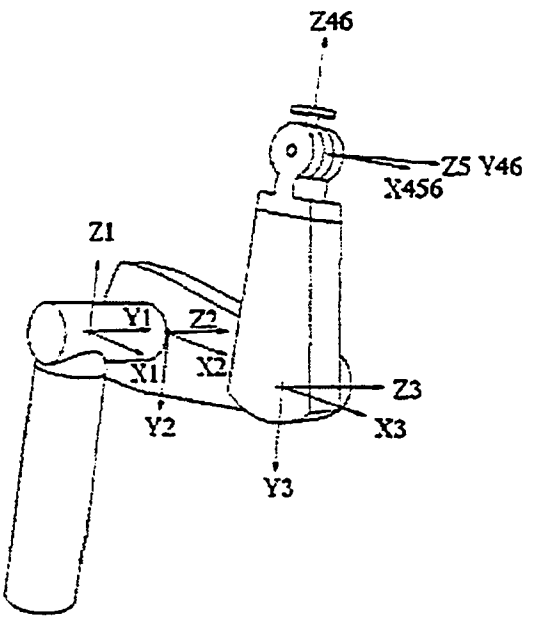

Figure 2: PUMA 560 coordinate axes

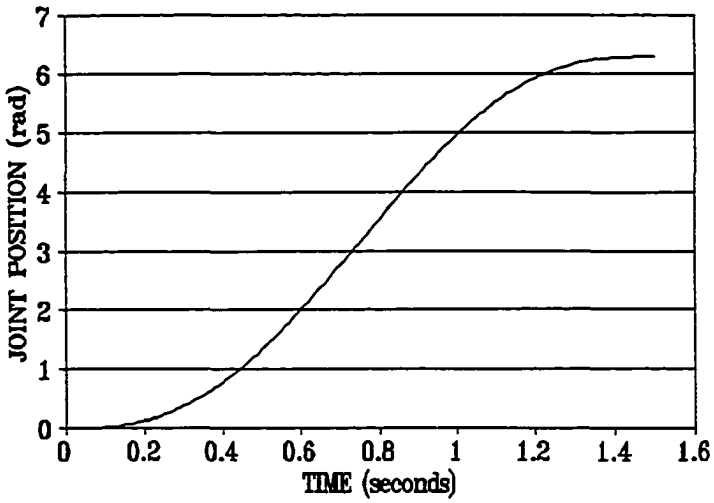

Figure 3a: Joint position specified via equation (18)

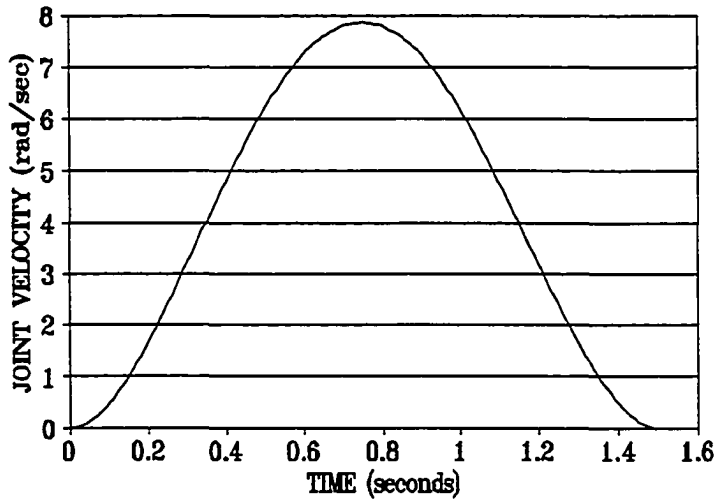

Figure 3b: Joint velocity specified via equation (19)

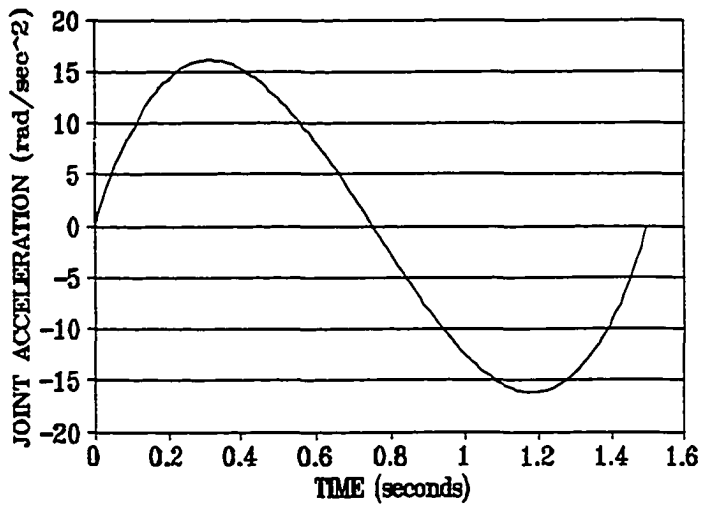

Figure 3c: Joint acceleration specified via equation (20)

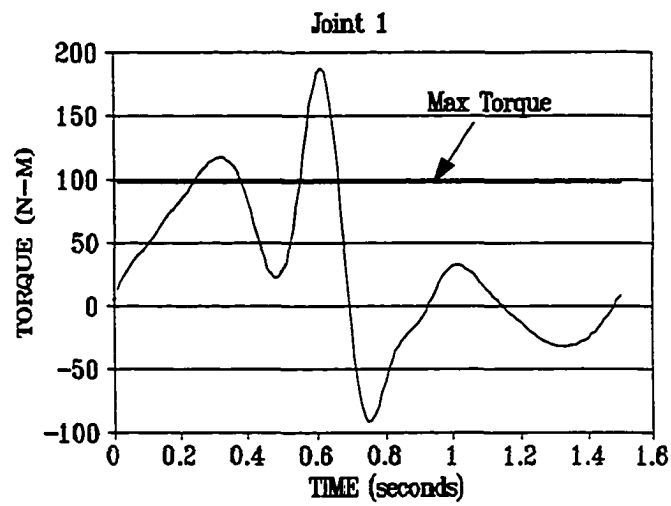

Figure 4a: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a-3c

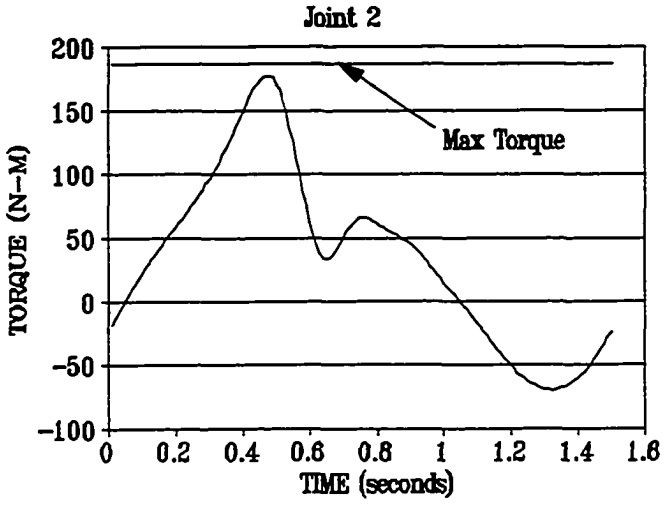

Figure 4b: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a-3c

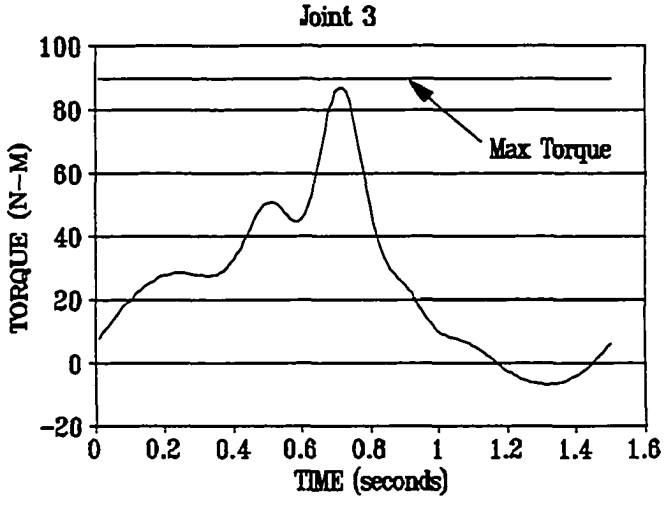

Figure 4c: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a-3c

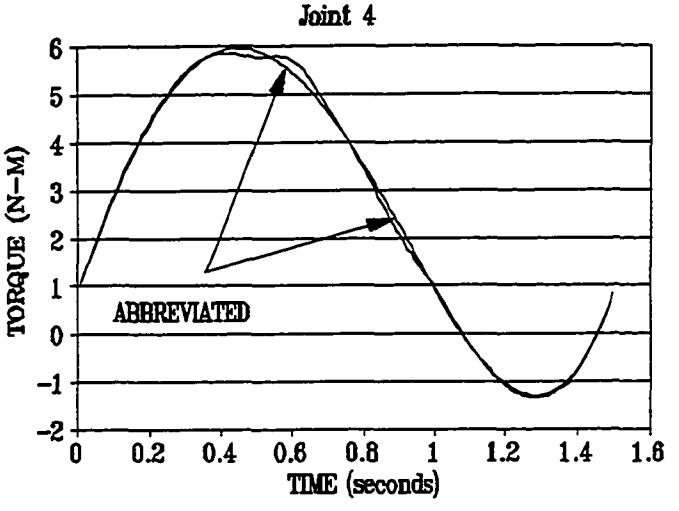

Figure 4d: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a-3c

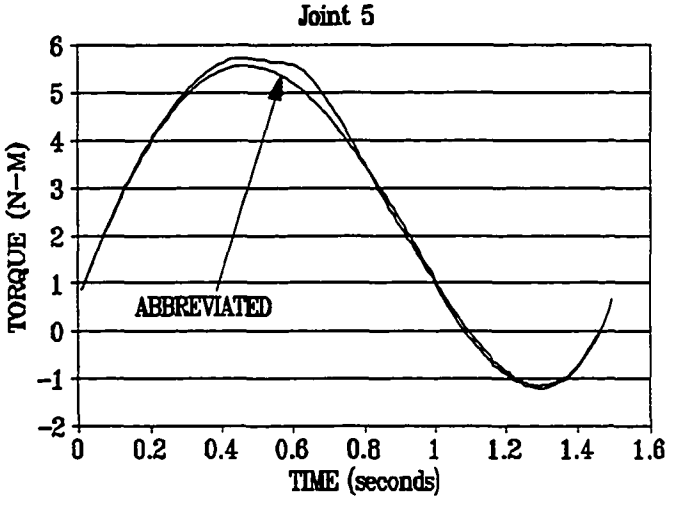

Figure 4e: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a-3c

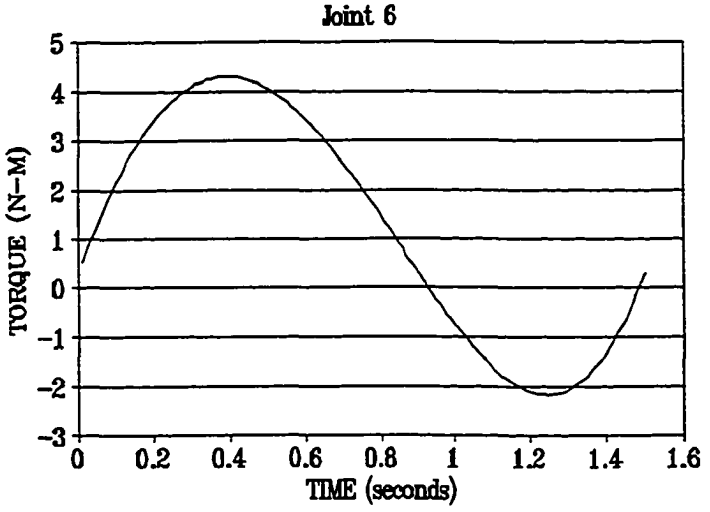

Figure 4f: Comparison of abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a-3c

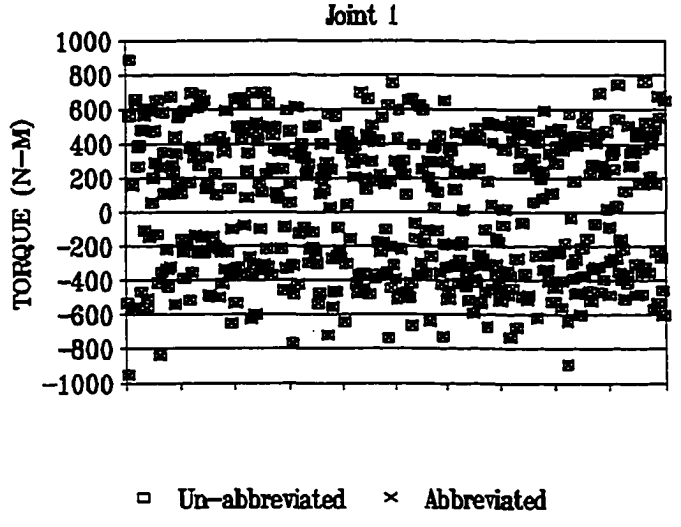

Figure 5a: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2)

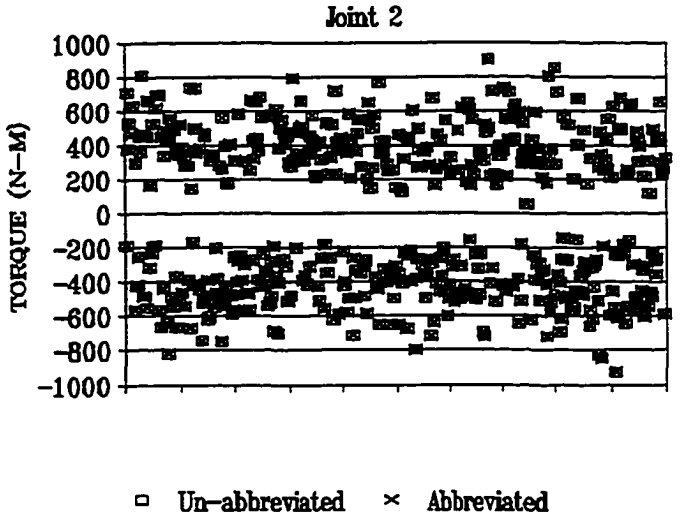

Figure 5b: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2)

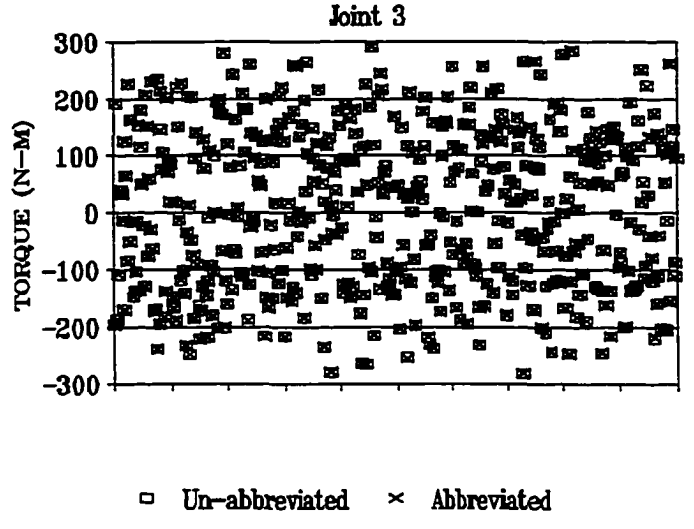

Figure 5c: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2)

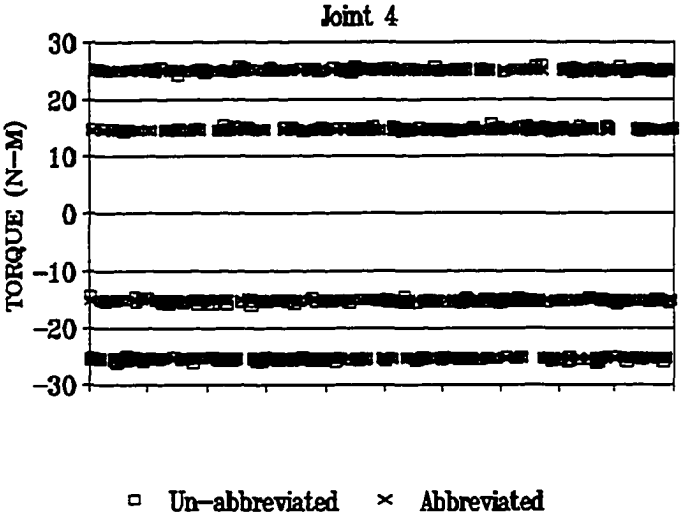

Figure 5d: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2)

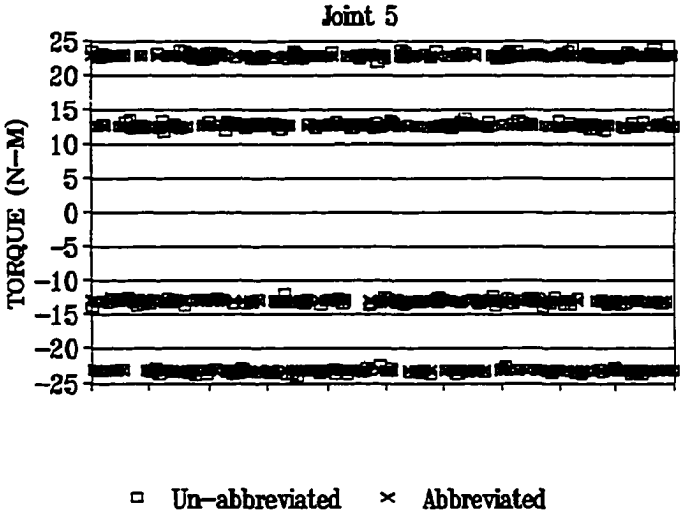

Figure 5e: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2)

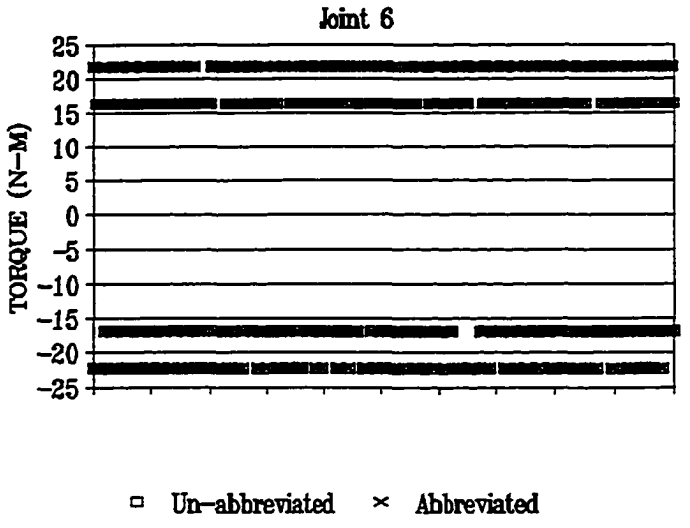

Figure 5f: Comparison of abbreviated and un-abbreviated nonlinear models for randomly varying joint angles, joint velocities (±10 rad/sec), and joint accelerations (±100 rad/sec^2)

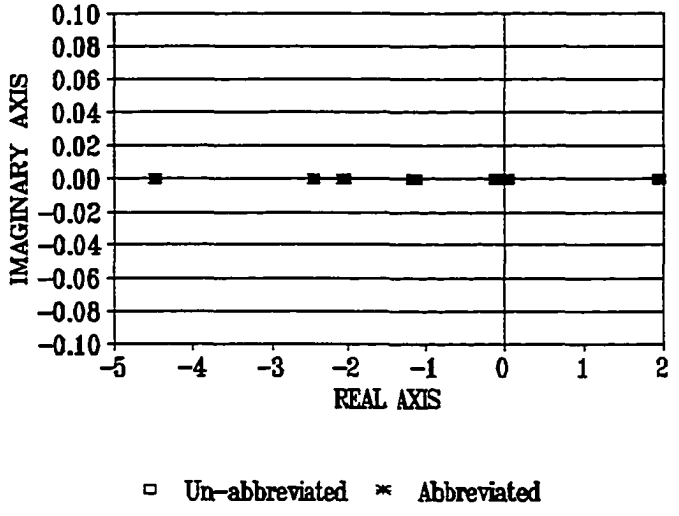

Figure 6a: Root locus showing eigenvalue differences between the abbreviated and un-abbreviated linear systems given in Table V

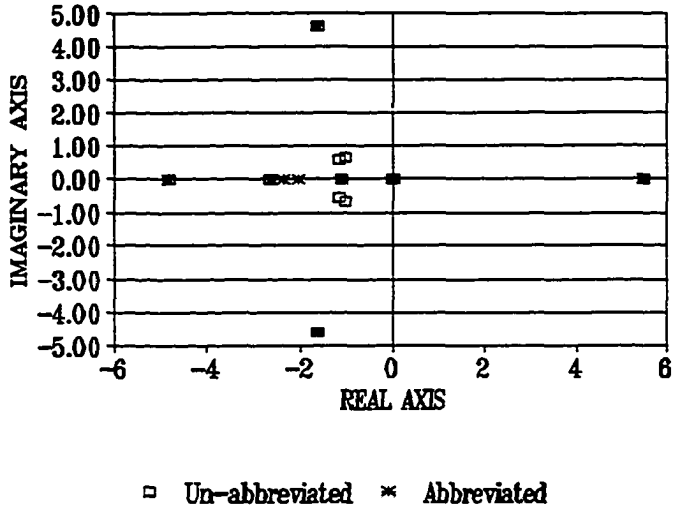

Figure 6b: Root locus showing eigenvalue differences between the abbreviated and un-abbreviated linear systems given in Table VI

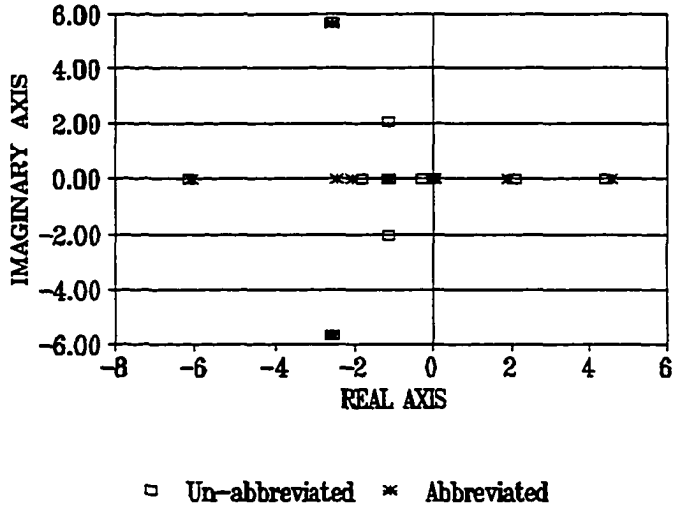

Figure 6c: Root locus showing eigenvalue differences between the abbreviated and un-abbreviated linear systems given in Table VII

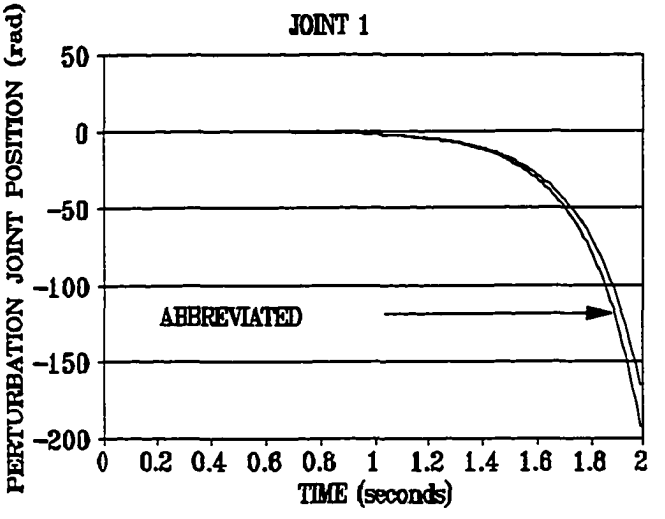

Figure 7a: Time response of the linear sytem given in Table VII

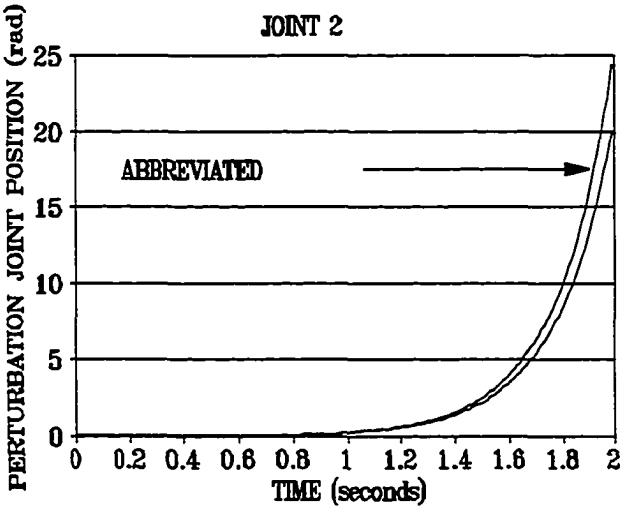

Figure 7b: Time response of the linear sytem given in Table VII

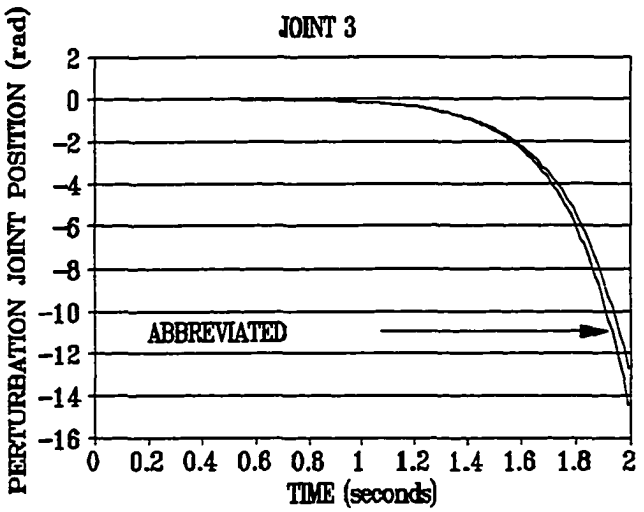

Figure 7c: Time response of the linear sytem given in Table VII

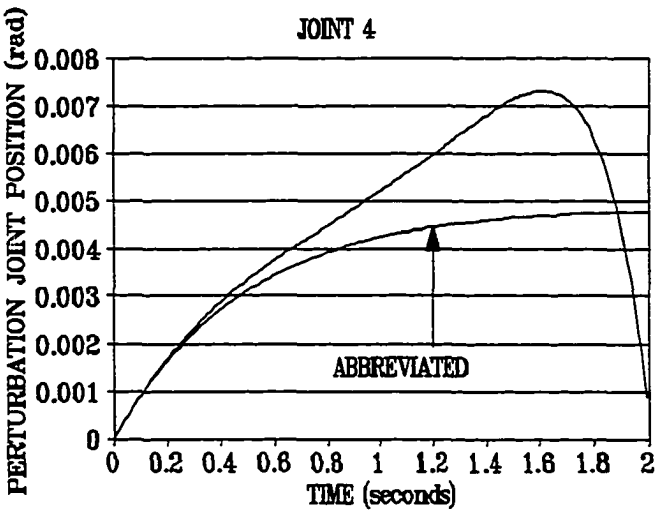

Figure 7d: Time response of the linear sytem given in Table VII

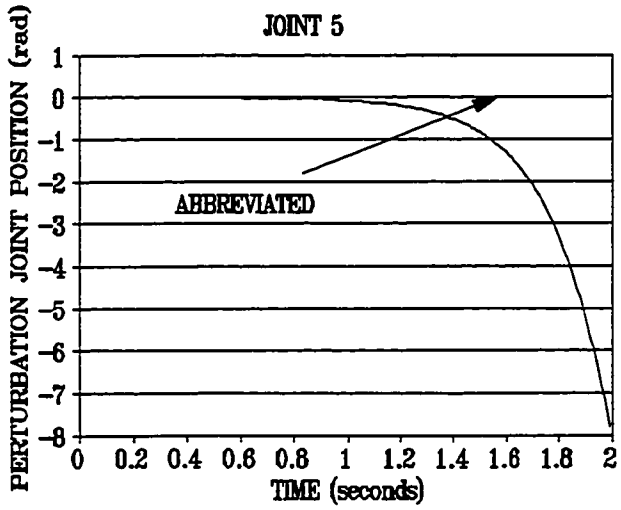

Figure 7e: Time response of the linear sytem given in Table VII

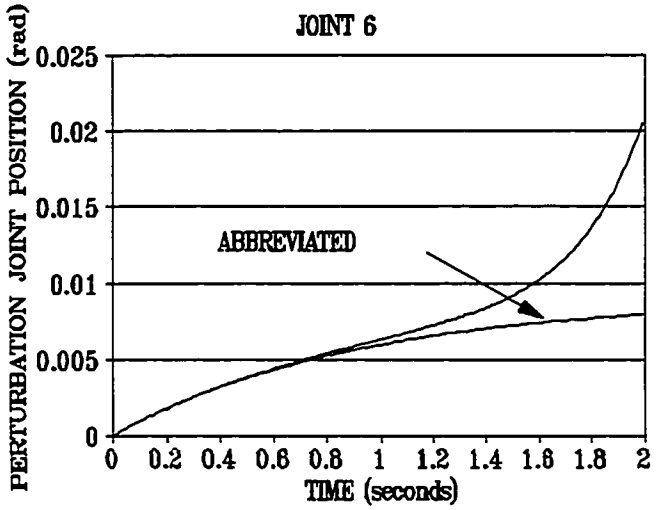

Figure 7f: Time response of the linear sytem given in Table VII

Table I: Comparison of the computational burden of PUMA 560 inverse dynamics and linearized dynamics

Nonlinear model (inverse dynamics)		
Non-Abbreviated Models	Multiplications	Additions
Burdick (1986)	401	254
Neuman and Murray (1987)	473	358
Toogood (1989)	814	630
Abbreviated Models		
Armstrong, et al. (1986)	201	102
Present model	87	60
Linear model (three sensitivity matrices)		
Non-Abbreviated Models	Multiplications	Additions
Balafoutis and Patel (1989)	1726	1740
Li (1990)	1364	1228
Abbreviated Models		
Present model	257	158

Table II: Maximum joint torques for the PUMA 560 reported by Armstrong, et al. (1986)

Joint	Maximum torque (N-M)
1	97.6
2	186.4
3	89.4
4	24.2
5	20.1
6	21.3

Table III: Statistics showing differences between abbreviated and un-abbreviated nonlinear models for the trajectory in Figures 3a-3c

Joint Number	Average Error (N-M)	Max Absolute Error (N-M)	Average Absolute Error (N-M)	Standard Deviation (N-M)
1	-0.026	0.99	0.27	0.28
2	-0.032	0.75	0.19	0.19
3	-0.035	0.80	0.21	0.20
4	-0.002	0.27	0.07	0.07
5	0.047	0.43	0.11	0.10
6	0.0	0.009	0.003	0.003

Table IV: Statistics showing differences between abbreviated and un-abbreviated nonlinear models for randomly varying joint angles with joint velocities at ±10 rad/sec and joint accelerations at ±100 rad/sec^2

Joint Number	Average Error (N-M)	Max Absolute Error (N-M)	Average Absolute Error (N-M)	Standard Deviation (N-M)
1	0.076	3.74	0.9	0.68
2	-0.022	2.58	0.7	0.51
3	0.037	2.53	0.64	0.49
4	-0.046	1.14	0.29	0.23
5	0.008	1.24	0.30	0.24
6	0.0002	0.02	0.007	0.005

Table V: Comparison of abbreviated and un-abbreviated linear system matrices

Trajectory: joint positions=(0,0,0,0,0,0) rad, joint velocities (0,0,0,0,0,0) rad/sec, and joint accelerations (0,0,0,0,0,0) rad/sec^2

Un-abbreviated M matrix

4.2732	-0.1107	-0.1345	0.0016	-0.0004	0.0000
-0.1107	4.3824	0.3253	0.0000	0.0019	0.0000
-0.1345	0.3253	1.1662	0.0000	0.0019	0.0000
0.0016	0.0000	0.0000	0.2016	0.0000	0.0000
-0.0004	0.0019	0.0019	0.0000	0.1796	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.1930

Abbreviated M matrix

4.2728	-0.1104	-0.1341	0.0000	0.0000	0.0000
-0.1104	4.3801	0.3226	0.0000	0.0000	0.0000
-0.1341	0.3226	1.1636	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.2000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.1790	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.1930

Un-abbreviated C matrix

4.9400	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	7.6700	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	3.2700	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.4116	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.4270	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.2160

Abbreviated C matrix

4.9400	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	7.6700	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	3.2700	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.4116	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.4270	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.2160

Un-abbreviated K matrix

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	-7.4583	-8.4825	0.0000	-0.0283	0.0000
0.0000	-8.4825	-8.4825	0.0000	-0.0283	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	-0.0283	-0.0283	0.0000	-0.0283	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Abbreviated K matrix

0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	-7.4583	-8.4823	0.0000	0.0000	0.0000
0.0000	-8.4823	-8.4823	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	-0.0283	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table VI: Comparison of abbreviated and un-abbreviated linear system matrices

Trajectory: joint positions=(1.51,1.51,1.51,1.51,1.51,1.51) rad,
joint velocities (6.56,6.56,6.56,6.56,6.56,6.56) rad/sec,
and joint accelerations (11.23,11.23,11.23,11.23,11.23,11.23) rad/sec^2

Un-abbreviated M matrix

2.7130	0.8256	0.1339	-0.0022	0.0001	0.0000
0.8256	5.1434	0.7045	-0.0025	0.0001	0.0000
0.1339	0.7045	1.1636	-0.0013	-0.0000	0.0000
-0.0022	-0.0025	-0.0013	0.2018	0.0000	0.0000
0.0001	0.0001	-0.0000	0.0000	0.1796	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.1930

Abbreviated M matrix

2.7124	0.8236	0.1331	0.0000	0.0000	0.0000
0.8236	5.1435	0.7044	0.0000	0.0000	0.0000
0.1331	0.7044	1.1636	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.2000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.1790	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.1930

Un-abbreviated C matrix

1.2440	-2.0506	-0.3818	-0.0052	-0.0263	0.0003
2.8113	8.0747	0.8426	-0.0046	-0.0675	0.0000
0.9028	-0.4539	3.2540	-0.0016	-0.0340	0.0000
-0.0097	0.0022	0.0022	0.4118	0.0107	-0.0003
-0.0088	0.0389	0.0236	-0.0107	0.4270	-0.0000
0.0003	0.0001	0.0001	-0.0002	0.0000	0.2160

Abbreviated C matrix

1.2377	-2.1432	-0.4726	0.0000	0.0000	0.0000
2.8049	8.1088	0.8776	0.0000	0.0000	0.0000
0.8974	-0.4388	3.2700	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.4116	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.4270	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.2160

Un-abbreviated K matrix

0.0000	223.3854	56.1990	-0.1961	0.0466	0.0000
0.0000	-66.1524	-65.9800	0.2769	-0.1303	0.0000
0.0000	-19.3987	12.7551	0.0152	-0.0637	0.0000
0.0000	0.2047	0.0990	0.2929	0.0063	0.0000
0.0000	-0.0443	-0.0469	-0.0289	0.3165	0.0000
0.0000	-0.0012	-0.0012	-0.0067	-0.0029	0.0000

Abbreviated K matrix

0.0000	223.1539	55.9517	0.0000	0.0000	0.0000
0.0000	-66.1312	-66.1162	0.0000	0.0034	0.0000
0.0000	-19.3700	12.7473	0.0000	0.0034	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0034	0.0034	0.0000	0.0017	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table VII: Comparison of abbreviated and un-abbreviated linear system matrices

Trajectory: joint positions=(2.44,4.90,1.06,1.60,5.80,4.49) rad,
joint velocities (-10,-10,-10,10,10,-10) rad/sec, and
joint accelerations (-100,-100,-100,-100,100,100) rad/sec^2

Un-abbreviated M matrix

2.7405	-0.7987	-0.1257	0.0014	-0.0003	0.0000
-0.7987	5.0443	0.6560	0.0012	-0.0003	-0.0000
-0.1257	0.6560	1.1658	0.0007	-0.0000	-0.0000
0.0014	0.0012	0.0007	0.2017	0.0000	0.0000
-0.0003	-0.0003	-0.0000	0.0000	0.1796	0.0000
0.0000	-0.0000	-0.0000	0.0000	0.0000	0.1930

Abbreviated M matrix

2.7377	-0.8007	-0.1272	0.0000	0.0000	0.0000
-0.8007	5.0404	0.6528	0.0000	0.0000	0.0000
-0.1272	0.6528	1.1636	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.2000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.1790	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.1930

Un-abbreviated C matrix

1.2269	-4.1059	2.4947	-0.0033	-0.0080	0.0000
2.8269	4.7166	-7.6548	-0.0544	-0.0363	0.0005
-0.6873	3.8541	3.0327	-0.0240	-0.0300	0.0005
-0.0147	0.0043	-0.0009	0.4099	-0.0107	0.0003
-0.0687	-0.0185	-0.0064	0.0107	0.4270	0.0007
0.0000	0.0003	0.0003	0.0001	-0.0007	0.2160

Abbreviated C matrix

1.1940	-4.2722	2.3384	0.0000	0.0000	0.0000
2.8997	4.6993	-7.6613	0.0000	0.0000	0.0000
-0.6438	3.8307	3.0200	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.4116	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.4270	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.2160

Un-abbreviated K matrix

0.0000	692.0988	236.1637	0.4090	-0.2016	0.0000
0.0000	-299.9462	-216.8453	-0.4448	1.3559	0.0000
0.0000	-63.0536	-15.8041	-0.2746	0.4047	0.0000
0.0000	-0.4557	-0.3778	0.3183	0.3941	0.0000
0.0000	-0.0327	0.0167	0.3811	0.8506	0.0000
0.0000	-0.0068	-0.0068	-0.0065	-0.0118	0.0000

Abbreviated K matrix

0.0000	689.8047	234.4940	0.0000	0.0000	0.0000
0.0000	-299.8244	-216.2899	0.0000	0.0041	0.0000
0.0000	-63.0674	-15.7745	0.0000	0.0041	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0041	0.0041	0.0000	-0.0238	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table VIII: Eigenvalues for linear systems in Tables V-VII

Roots from Table V linear system

Un-abbreviated		Abbreviated	
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
1.9371	0.0000	1.9385	0.0000
-4.4707	0.0000	-4.4783	0.0000
-0.0990	0.0000	-0.0990	0.0000
0.0642	0.0000	0.0644	0.0000
-2.4413	0.0000	-2.4499	0.0000
-1.1554	0.0000	-1.1555	0.0000
-2.0327	0.0000	-2.0327	0.0000
-2.0413	0.0000	-2.0580	0.0000
-1.1189	0.0000	-1.1192	0.0000

Roots from Table VI linear system

Un-abbreviated		Abbreviated	
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
5.5144	0.0000	5.4988	0.0000
-1.6369	4.5895	-1.6402	4.5890
-1.6369	-4.5895	-1.6402	-4.5890
-4.8057	0.0000	-4.8128	0.0000
-2.6587	0.0000	-2.6476	0.0000
-1.1768	0.5761	-2.3815	0.0000
-1.1768	-0.5761	-0.0040	0.0000
-1.0309	0.6552	-2.0580	0.0000
-1.0309	-0.6552	0.0000	0.0000
-1.1188	0.0000	-1.1192	0.0000

Roots from Table VII linear system

Un-abbreviated		Abbreviated	
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
-2.5583	5.6843	-2.5599	5.6632
-2.5583	-5.6843	-2.5599	-5.6632
4.4242	0.0000	4.6033	0.0000
-6.1404	0.0000	-6.0603	0.0000
2.0829	0.0000	1.8839	0.0000
-1.1443	2.0590	-2.0580	0.0000
-1.1443	-2.0590	0.0000	0.0000
-0.2889	0.0000	0.0545	0.0000
-1.8023	0.0000	-2.4400	0.0000
-1.1190	0.0000	-1.1192	0.0000

Appendix A -- Nonlinear Equations

(Note: matrix elements not explicitly given are set equal to zero)

Mass matrix (M_{ij}=M_{ji}) - M(θ)

$$M_{11} = 3.64 + .8\cos(2\theta_2) - .01\sin(2\theta_3 + 2\theta_2) - .01\cos(\theta_3 + 2\theta_2) + .37\sin(\theta_3 + 2\theta_2) - .15\cos(2\theta_3 + 2\theta_2) + .37\sin(\theta_3) - .01\cos(\theta_3)$$

$$M_{12} = .69\sin(\theta_2) - .13\cos(\theta_3 + \theta_2) + .02\cos(\theta_2)$$

$$M_{13} = -.13\cos(\theta_3 + \theta_2)$$

$$M_{22} = 4.4 - .02\cos(\theta_3) + .74\sin(\theta_3)$$

$$M_{23} = .33 - .01\cos(\theta_3) + .37\sin(\theta_3)$$

$$M_{33} = 1.16$$

$$M_{44} = .20$$

$$M_{55} = .18$$

$$M_{66} = .19$$

Coriolis matrix - B(θ)

$$B_{11} = -1.6\sin(2\theta_2) - .01\cos(\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .30\sin(2\theta_3 + \theta_2) + .74\cos(\theta_3 + 2\theta_2) + .02\sin(\theta_3 + 2\theta_2)$$

$$B_{12} = -.02\cos(2\theta_3 + 2\theta_2) + .3\sin(2\theta_3 + 2\theta_2) + .37\cos(\theta_3 + 2\theta_2) + .01\sin(\theta_3 + 2\theta_2) + .37\cos(\theta_3) + .01\sin(\theta_3)$$

$$B_{16} = .27\sin(\theta_3 + \theta_2)$$

$$B_{26} = .02\sin(\theta_3) + .74\cos(\theta_3)$$

Centrifugal matrix - D(θ)

$$D_{12} = .69\cos(\theta_2) - .02\sin(\theta_2) + .13\sin(\theta_3 + \theta_2)$$

$$D_{13} = .13\sin(\theta_3 + \theta_2)$$

$$D_{21} = .8\sin(2\theta_2) + .01\cos(2\theta_3 + 2\theta_2) - .1492\sin(2\theta_3 + 2\theta_2) - .37\cos(\theta_3 + 2\theta_2) - .01\sin(\theta_3 + 2\theta_2)$$

$$D_{23} = .01\sin(\theta_3) + .37\cos(\theta_3)$$

$$D_{31} = .01\cos(2\theta_3 + 2\theta_2) - .15\sin(2\theta_3 + 2\theta_2) - .19\cos(\theta_3 + 2\theta_2) - .19\cos(\theta_3)$$

$$D_{32} = -.37\cos(\theta_3) - .01\sin(\theta_3)$$

Gravity vector - G(θ)

$$G_2 = -37.2\cos(\theta_2) + 1.02\sin(\theta_2) - 8.5\sin(\theta_3 + \theta_2) + .25\cos(\theta_3 + \theta_2)$$

$$G_3 = -8.5\sin(\theta_3 + \theta_2) + .25\cos(\theta_3 + \theta_2)$$

Friction vector - $V(\theta, \dot{\theta})$

$$V_1 = \begin{cases} 4.94\dot{\theta}_1 + 8.43, & \dot{\theta}_1 \geq 0 \\ 3.45\dot{\theta}_1 - 8.26, & \dot{\theta}_1 < 0 \end{cases}$$

$$V_2 = \begin{cases} 7.67\dot{\theta}_2 + 12.71, & \dot{\theta}_2 \geq 0 \\ 8.53\dot{\theta}_2 - 11.34, & \dot{\theta}_2 < 0 \end{cases}$$

$$V_3 = \begin{cases} 3.27\dot{\theta}_3 + 5.93, & \dot{\theta}_3 \geq 0 \\ 3.02\dot{\theta}_3 - 5.57, & \dot{\theta}_3 < 0 \end{cases}$$

$$V_4 = \begin{cases} .41\dot{\theta}_4 + .85, & \dot{\theta}_4 \geq 0 \\ .41\dot{\theta}_4 - 1.29, & \dot{\theta}_4 < 0 \end{cases}$$

$$V_5 = \begin{cases} .43\dot{\theta}_5 + .67, & \dot{\theta}_5 \geq 0 \\ .43\dot{\theta}_5 - 1.04, & \dot{\theta}_5 < 0 \end{cases}$$

$$V_6 = \begin{cases} .22\dot{\theta}_6 + .30, & \dot{\theta}_6 \geq 0 \\ .22\dot{\theta}_6 - .81, & \dot{\theta}_6 < 0 \end{cases}$$

Appendix B -- Linear Equations

(Note: joint variables denote the nominal trajectory point and matrix elements not explicitly given are set equal to zero)

Mass matrix (M_{ij}=M_{ji}) - $M(\theta)$
(Same form as in Appendix A)

Joint velocity sensitivity matrix - $C(\theta,\dot{\theta})$

$$C_{11} = [-1.60\sin(2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) + .74\cos(\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .02\sin(\theta_3 + 2\theta_2) - .01\cos(2\theta_2)]\dot{\theta}_2 + [.37\cos(\theta_3) + .37\cos(\theta_3 + 2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .01\sin(\theta_3) + .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_3$$

$$C_{12} = [-1.60\sin(2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) + .74\cos(\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .02\sin(\theta_3 + 2\theta_2) - .01\cos(2\theta_2)]\dot{\theta}_1 + [1.38\cos(\theta_2) + .27\sin(\theta_3 + \theta_2) - .05\sin(\theta_2)]\dot{\theta}_2 + .27\sin(\theta_3 + \theta_2)\dot{\theta}_3$$

$$C_{13} = [.37\cos(\theta_3) + .37\cos(\theta_3 + 2\theta_2) + .3\sin(2\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .01\sin(\theta_3) + .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_1 + .27\sin(\theta_3 + \theta_2)\dot{\theta}_2 + .27\sin(\theta_3 + \theta_2)\dot{\theta}_3$$

$$C_{21} = [1.60\sin(2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) - .74\cos(\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .02\sin(\theta_3 + 2\theta_2) + .01\cos(2\theta_2)]\dot{\theta}_1$$

$$C_{22} = [.74\cos(\theta_3) + .02\sin(\theta_3)]\dot{\theta}_3$$

$$C_{23} = [.74\cos(\theta_3) + .02\sin(\theta_3)]\dot{\theta}_3 + [.74\cos(\theta_3) + .02\sin(\theta_3)]\dot{\theta}_2$$

$$C_{31} = [-.37\cos(\theta_3) - .37\cos(\theta_3 + 2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .01\sin(\theta_3) - .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_1$$

$$C_{32} = [-.74\cos(\theta_3) - .02\sin(\theta_3)]\dot{\theta}_2$$

Joint position sensitivity matrix - $K(\theta,\dot{\theta},\ddot{\theta})$

$$K_{12} = [-1.60\sin(2\theta_2) + .74\cos(\theta_3 + 2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .02\sin(\theta_3 + 2\theta_2) - .01\cos(2\theta_2)]\ddot{\theta}_1 + [.69\cos(\theta_2) + .13\sin(\theta_3 + \theta_2) - .02\sin(\theta_2)]\ddot{\theta}_2 + .13\sin(\theta_3 + \theta_2)\ddot{\theta}_3 + [-3.20\cos(2\theta_2) - 1.5\sin(\theta_3 + 2\theta_2) + .6\cos(2\theta_3 + 2\theta_2) + .04\cos(\theta_3 + 2\theta_2) + .04\sin(2\theta_3 + 2\theta_2) + .03\sin(2\theta_2)]\dot{\theta}_1\dot{\theta}_2 + [-.74\sin(\theta_3 + 2\theta_2) + .60\cos(2\theta_3 + 2\theta_2) + .04\sin(2\theta_3 + 2\theta_2) + .02\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_3 + .27\cos(\theta_3 + \theta_2)\dot{\theta}_2\dot{\theta}_3 + [-.69\sin(\theta_2) + .13\cos(\theta_3 + \theta_2) - .02\cos(\theta_2)]\dot{\theta}_2^2 + .13\cos(\theta_3 + \theta_2)\dot{\theta}_3^2$$

$$K_{13} = [.37\cos(\theta_3) + .37\cos(\theta_3 + 2\theta_2) + .3\sin(2\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .01\sin(\theta_3 + 2\theta_2) + .01\sin(\theta_3)]\ddot{\theta}_1 + .13\sin(\theta_3 + \theta_2)(\ddot{\theta}_2 + \ddot{\theta}_3) + [-.74\sin(\theta_3 + 2\theta_2) + .6\cos(2\theta_3 + 2\theta_2) + .04\sin(2\theta_3 + 2\theta_2) + .02\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_2 + [.6\cos(2\theta_3 + 2\theta_2) - .37\sin(\theta_3) - .37\sin(\theta_3 + 2\theta_2) + .04\sin(2\theta_3 + 2\theta_2) + .01\cos(\theta_3) + .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_3 + .27\cos(\theta_3 + \theta_2)\dot{\theta}_2\dot{\theta}_3 + .13\cos(\theta_3 + \theta_2)(\dot{\theta}_2^2 + \dot{\theta}_3^2)$$

$$K_{22} = 37.23\sin(\theta_2) - 8.45\cos(\theta_3 + \theta_2) + 1.02\cos(\theta_2) - .25\sin(\theta_3 + \theta_2) - .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2) + [.69\cos(\theta_2) + .13\sin(\theta_3 + \theta_2) - .02\sin(\theta_2)]\ddot{\theta}_1 + [1.6\cos(2\theta_2) + .74\sin(\theta_3 + 2\theta_2) - .3\cos(2\theta_3 + 2\theta_2) - .02\sin(2\theta_3 + 2\theta_2) - .02\cos(\theta_3 + 2\theta_2) - .01\sin(2\theta_2)]\dot{\theta}_1^2$$

$$K_{23} = -8.54\cos(\theta_3 + \theta_2) - .25\sin(\theta_3 + \theta_2) - .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2) + .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [.74\cos(\theta_3) + .02\sin(\theta_3)]\ddot{\theta}_2 + [.37\cos(\theta_3) + .01\sin(\theta_3)]\ddot{\theta}_3 + [-.74\sin(\theta_3) + .02\cos(\theta_3)]\dot{\theta}_2\dot{\theta}_3 + [-.37\sin(\theta_3) + .01\cos(\theta_3)]\dot{\theta}_3^2 + [.37\sin(\theta_3 + 2\theta_2) - .30\cos(2\theta_3 + 2\theta_2) - .02\sin(2\theta_3 + 2\theta_2) - .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1^2$$

$$K_{25} = .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$K_{23} = -8.54\cos(\theta_3 + \theta_2) - .25\sin(\theta_3 + \theta_2) - .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2) + .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [.37\sin(\theta_3 + 2\theta_2) - .30\cos(2\theta_3 + 2\theta_2) - .02\sin(2\theta_3 + 2\theta_2) - .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1^2$$

$$K_{33} = -8.54\cos(\theta_3 + \theta_2) - .25\sin(\theta_3 + \theta_2) - .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2) + .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [.37\cos(\theta_3) + .01\sin(\theta_3)]\ddot{\theta}_2 + [.37\sin(\theta_3) - .01\cos(\theta_3)]\dot{\theta}_2^2 + [.19\sin(\theta_3 + 2\theta_2) + .19\sin(\theta_3) - .30\cos(2\theta_3 + 2\theta_2) - .02\sin(2\theta_3 + 2\theta_2)]\dot{\theta}_1^2$$

$$K_{35} = .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$K_{52} = .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$K_{53} = .01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$K_{55} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) - .01\cos(\theta_5 + \theta_3 + \theta_2)$$

Friction was added to the linear model as follows:

$$C_{11} = \begin{cases} C_{11} + 4.94, & \dot{\theta}_1 \geq 0 \\ C_{11} + 3.45, & \dot{\theta}_1 < 0 \end{cases}$$

$$C_{22} = \begin{cases} C_{22} + 7.67, & \dot{\theta}_2 \geq 0 \\ C_{22} + 8.53, & \dot{\theta}_2 < 0 \end{cases}$$

$$C_{33} = \begin{cases} C_{33} + 3.27, & \dot{\theta}_3 \geq 0 \\ C_{33} + 3.02, & \dot{\theta}_3 < 0 \end{cases}$$

$$C_{44} = C_{44} + .41$$

$$C_{55} = C_{55} + .43$$

$$C_{66} = C_{66} + .22$$

III. THE USE OF SYMBOLIC, COMPUTER GENERATED CONTROL LAWS IN HIGH SPEED TRAJECTORY FOLLOWING: ANALYSIS AND EXPERIMENTS

A paper to be submitted to The IEEE Transactions on Robotics and Automation

C.L. Clover
Iowa State University

Abstract

Prior research has demonstrated the robustness of robotic control algorithms based on trajectory linearization. Furthermore, the performance and stability of trajectory linearization schemes can be analyzed via eigenvalue analyses which are straightforward and intuitive. However, the use of these methods in robotics applications has been limited by the computational requirements of higher degree of freedom systems. This paper demonstrates the utility of computer-generated control laws based on explicit, time-varying trajectory linearization. The paper makes use of abbreviated symbolic nonlinear and linear models which allow for a very low computational burden while maintaining a high degree of robustness with respect to modeling errors and external disturbances. Symbolic processing software is used to automatically generate both the feedforward and feedback control loops in compilable code which is ready for implementation. Test results are given for the PUMA 560 which demonstrate these techniques.

I. INTRODUCTION

The last five years has seen a high growth rate in symbolic computing power which is readily available to most researchers. This is bringing about a fundamental change in control

system design for robotic systems. In the past, the immense complexity associated with higher degree of freedom robotics systems made analytical modeling with explicit, symbolic equations very difficult. On the other hand, these analytical expressions allow for intuitive stability and performance analyses. Modern symbolic processing software makes possible the analysis of the governing differential equations for systems as complex as six degree of freedom robots. Thus, the ability to create symbolic, computer generated control laws will provide new opportunities for robotics control system engineers to develop robust control laws which are intuitive and easy to implement.

This paper uses symbolic processing to develop a robust controller for a PUMA 560 which is to be used as a high speed human/machine interface. This controller is based on an explicit, symbolic state space linearization of the PUMA's nonlinear equations of motion about an arbitrary trajectory point. The computational complexity of this approach has limited its implementation in real-time systems. The paper demonstrates the utility of abbreviated symbolic linear equations which are used in developing a controller suitable for real-time implementation. The robustness of this controller is analyzed via straightforward eigenvalue analyses. The differences between trajectory linearization approaches and traditional computed torque approaches are also highlighted.

The next section of this paper surveys the literature. Other sections present control law derivations, robustness considerations, test/simulation results, and conclusions.

II. LITERATURE SURVEY

Some of the common trajectory control schemes found in the literature include decentralized (or independent) joint control [1], [2] and model referenced adaptive

control (MRAC) [2], [3], and [4]. (Note we are not considering force/impedance control algorithms in this paper.) A great deal of research has demonstrated that computed torque approaches (i.e. utilizing the nonlinear robot inverse dynamics equations evaluated at a nominal trajectory point to cancel system nonlinearities) can achieve good trajectory following (see, for example, [5]-[11]).

Many other researchers have developed variations of the computed torque idea. An excellent review of the various control schemes which are currently available in the literature can be found in [5].

Most "off the shelf" robot controllers utilize independent joint control with either constant PD or constant PID gains while neglecting manipulator dynamics. The next step up in terms of controller complexity is to utilize an inverse model in a computed torque approach. The conventional wisdom here is that an n degree of freedom nonlinear dynamic model, given by equation (1), can be used in a feedforward loop to cancel out the system nonlinearities while decoupling it according to equations (2)-(4).

$$\tau = M(q)\ddot{q} + V(q,\dot{q}) + F(q,\dot{q}) + G(q) \tag{1}$$

where τ is an n-vector of actuator inputs; q represents the n-vector of joint positions (usually denoted as θ for revolute joints or d for prismatic joints; M is the nxn inertial matrix; V is an nx1 Coriolis/centrifugal vector; F is an nxn friction matrix; and G is the nx1 gravity vector. Assuming we have exact knowledge of all terms in equation (1), the system can be linearized and decoupled as follows:

$$M(q)\ddot{q} + V(q,\dot{q}) + F(q,\dot{q}) + G(q) = M(q^*)(\ddot{q}^* + \Gamma_2\delta\dot{q} + \Gamma_1\delta q) + V(q^*,\dot{q}^*) + F(q^*,\dot{q}^*) + G(q^*) \quad (2)$$

which leads to closed-loop perturbation dynamics of

$$\delta\ddot{q} + \Gamma_2\delta\dot{q} + \Gamma_1\delta q = 0 \quad (3)$$

or

$$\frac{d}{dt}\begin{bmatrix}\delta q \\ \delta\dot{q}\end{bmatrix} = \begin{bmatrix}0 & I \\ -\Gamma_1 & -\Gamma_2\end{bmatrix}\begin{matrix}\delta q \\ \delta\dot{q}\end{matrix} \quad (4)$$

where * indicates the nominal trajectory; $\delta q = q^* - q$ is the perturbation or error from the nominal joint position; Γ_1 and Γ_2 are diagonal gain matrices; and I is an identity matrix.

This approach is also called "exact" linearization or feedback linearization [12]. Previous work in this area includes [13], [14], and [15]. A similar approach for Cartesian space control is presented in [16]. These approaches imply exact knowledge of the manipulator to be controlled. Therefore, robustness issues arise when the modeled robot is different from the actual robot. Also, calculating the inverse dynamic model of a higher degree of freedom robot can be computationally expensive.

Another interesting approach to the computed torque scheme assumes that the nonlinear equations don't exactly cancel out the system nonlinearities or decouple the equations, but rather give a feedforward torque that keeps the robot reasonably *close* to the nominal trajectory point at which the equations are evaluated. Therefore, Taylor series can be used to obtain reasonable linear approximations to the nonlinear differential equations of motion depending on the closeness of the system to the operating point about which the equations are linearized. This approach enables the use of state space linearization/perturbation methods

to determine the closed-loop feedback laws and relaxes the assumption that the nonlinear model exactly cancels the nonlinearities of the real system. However, the disadvantages are that the resulting linear/perturbation equations remain coupled and are of a form that is more analytically complex than the nonlinear equations. This complexity arises from chain rule differentiation of the nonlinear trigonometric terms common in dynamic robotic manipulator models.

The utility of trajectory based linearization in controlling robots is demonstrated in [17] and [18]. Furthermore, these authors were able to demonstrate in simulation the robustness of this method by maintaining accurate trajectory following in the presence of payload uncertainties. A Taylor series expansion can be used to linearize the nonlinear robot model near a nominal trajectory point while neglecting higher order terms.

$$f(q,\dot{q},\ddot{q}) = f(q^*,\dot{q}^*,\ddot{q}^*) + \left(\frac{\partial f}{\partial q}\Big|_*\right)\delta q + \left(\frac{\partial f}{\partial \dot{q}}\Big|_*\right)\delta\dot{q} + \left(\frac{\partial f}{\partial \ddot{q}}\Big|_*\right)\delta\ddot{q} \tag{5}$$

After canceling out the nominal components of the trajectory and retaining terms which are linear with respect to the perturbation state variables and inputs we have

$$\delta\tau = M(q^*)\delta\ddot{q} + C(q^*,\dot{q}^*)\delta\dot{q} + K(q^*,\dot{q}^*,\ddot{q}^*)\delta q \tag{6}$$

where $M(q^*)$, $C(q^*,\dot{q}^*)$, and $K(q^*,\dot{q}^*,\ddot{q}^*)$ are the nxn linearized trajectory sensitivity matrices in terms of the nominal trajectory. We can re-write equation (6) in state space form as

$$\frac{d}{dt}\begin{bmatrix}\delta q\\ \delta\dot{q}\end{bmatrix} = \begin{bmatrix}0 & I\\ -M^{-1}K & -M^{-1}C\end{bmatrix}\begin{bmatrix}\delta q\\ \delta\dot{q}\end{bmatrix} + \begin{bmatrix}0\\ M^{-1}\end{bmatrix}\delta\tau \tag{7}$$

which can be expressed in traditional form as

$$\delta\dot{x} = A(t)\delta x(t) + B(t)\delta u(t) \tag{8}$$

where $A(t) \in \Re^{2n,2n}$ and $B(t) \in \Re^{2n,n}$. The controller input is therefore the summation of the nominal torque, τ^*, calculated from equation (1) and the perturbation torque $\delta\tau$ which can be determined from any number of state space feedback design techniques.

A linear quadratic (LQ) control law is utilized in [17] and [18] to regulate the perturbations from the nominal trajectory toward zero. The basic control architecture consists of the Newton-Euler [19] iterations to calculate the nominal feedforward torques and a least squares identification algorithm to determine the linear equations. It is noted in [18] that this linear identification algorithm requires system parameters that are slowly time varying.

The utility of linear trajectory modeling which allows a designer to "shape" the transient tracking error response by placing the system eigenvalues is discussed in [20]. However, trajectory information is assumed to be known a priori for off-line controller gain calculations. Auto-regressive models are used in [21] and [22] which are identified on-line to develop a self-tuning control law based on pole placement. An approach is presented in [23] whereby the time varying gains of a linearized robot model are estimated directly rather than through on-line inertial parameter identification. A gain matrix is chosen which is diagonal and decoupled for simplicity. The phrase "linearized canceling control" is introduced in [23] to describe a pole placement scheme based on linearized models which uses a time-varying control term to cancel the time-varying terms in the linearized control system such that the overall closed-loop system is time-invariant. In other words, the system closed-loop eigenvalues remain fixed and trajectory independent.

The work in [17] was extended in [24] by introducing a dynamics operator to compute the feedforward nominal joint torques rather than using Newton-Euler recursions. On line recursive parameter estimation is again used to obtain the linear model, equation (8), while an LQ optimal control algorithm computes the gain matrices. This control architecture, based on trajectory linearization, is also shown to be robust against payload uncertainties. The numerical gain matrix calculation requires matrix a inversion and the nominal joint acceleration ($\ddot{q}^*$) is assumed to be zero.

Examples of other work in this area can be found in [25], [26], and [27]. Symbolic representations of the nonlinear and linear models of a simple two link robot are used in [25]. An LQ optimal control method is used to calculate gain matrices off-line for a set of pre-determined trajectory points. The issues associated with calculating the performance index weighting matrices are also discussed in [25]. Linear models to calculate controller gains based, again, on LQ optimal control methods are utilized in [26]. This method requires inversion of the inertia matrix and a modal matrix. Gain calculations are done off-line based on a priori trajectory knowledge. A trajectory linearization approach is presented in [27] which adds a "force-payload" sensitivity matrix to the equations.

Finally, a multivariable pole placement method for full MIMO (multi-input multi-output) gain determination is presented in [28]. Explicit expressions are developed for the gain matrices by utilizing similarity transformations developed in [29]. No matrix inversions are required in this method. Symbolic equations for a three link manipulator are used to demonstrate this technique.

This survey indicates that the main obstacle with perturbation control of robotic manipulators is that deriving the linearized equations of motion about an arbitrary nominal

trajectory is difficult both from an analytical viewpoint and computationally expensive from a numerical viewpoint. Either unsatisfactory approximations and assumptions have to be made or computationally expensive parameter estimation is employed. However, the advantages of being able to utilize traditional state space controller design methods make perturbation techniques very attractive from an intuitive standpoint.

This paper designs a control system based on traditional state space methods utilizing the explicit linear and nonlinear equations of motion of a six degree of freedom manipulator. Symbolic computing software enables the control engineer to quickly generate efficient control laws that maintain robustness with respect to both stability and performance. The next section presents a procedure for developing explicit, symbolic controller equations based on trajectory linearization techniques.

III. A COMPUTER GENERATED CONTROLLER

A. Background

This section presents the details of symbolically deriving a computer generated controller via linear state space methods. Figure 1 presents the overall control system architecture of a human/robot interface which we are developing to simulate dynamic force interactions between a user and a simulated object [30]. The present paper concerns itself only with the inner feedback loop used to control this system.

In this section, a set of gains are derived based on a pole placement approach. We choose this approach over others, such as LQ optimal control, since it allows more intuitive design and evaluation of the closed-loop system performance because the closed-loop eigenvalues can be directly specified. Furthermore, the design degree of freedom in LQ

methods usually involves the selection of the weighting matrices in the performance index. This selection process is usually heuristic and/or arbitrary and, in the end, the control system designer simply checks the closed-loop eigenvalues to see what effects the weighting matrices have on system response.

Although the basic approach here is similar to previous work (for example, [17], [24] and [25]), its implementation is quite different. The distinction between this paper and others is that no adaptive parameter estimation algorithms are employed. Rather, parameters which have been directly measured are used. The justification for this decision was based on several factors. First of all, one of the primary disadvantages of many of the on-line parameter estimation algorithms available in the literature is that they are computationally expensive, especially for higher degree of freedom systems [12]. Also, most require a "sufficient excitation condition" [31] whereby a trajectory must be "rich enough" to excite all the manipulator dynamics in order for acceptable parameter convergence to occur [32]. In many important applications, trajectories are not known a priori and/or cannot be guaranteed to be "sufficiently exciting". Finally, inherent in any adaptive identification application is the idea that the parameters are unknown. But clearly there is a tradeoff between determining the system parameters through identification versus direct measurement. In many cases, disassembling a robot and measuring inertial and kinematic parameters directly may be no less difficult or time consuming than writing and testing on-line identification software. Parameter identification techniques are often justified based on the assumption that robot parameters are not constant. Although the overall inertial characteristics of a manipulator change as it moves about its workspace, individual link parameters such as mass, inertia, and kinematic dimensions do not change significantly over

the course of the life of a robot. Friction characteristics, however, can change as a robot ages but these parameters can be measured directly and are very repeatable [33]. The one parameter that does change significantly from application to application is end effector payload. However, this paper demonstrates that non-adaptive controllers can be developed which are robust against modeling errors and disturbances caused by uncertain payloads.

Computed torque techniques assume a priori knowledge of the kinematic, inertial, and friction parameters of a robot. For human/robot applications this is a more appropriate constraint than is the assumption of a priori trajectory knowledge.

B. Nonlinear Feedforward Torques

A central feature of many advanced manipulator control systems is the use of the inverse dynamic model, equation (1), to bring the system reasonably close to a desired operating point. Nominal torques are computed based on a desired trajectory to be followed. However, these calculations can be computationally expensive for higher degree of freedom systems. Previous efforts have sought to reduce the computational demands for computing the feedforward torques (see, for example, [34], [35], [36]). However, a priori parameter knowledge makes possible the use of abbreviated models [37]. Essentially, the models are computed in symbolic form and a significance criterion then eliminates calculations which have little impact on the results. [38] presents an abbreviated PUMA 560 model which requires 87 multiplications and 60 additions to calculate the nominal feedforward torques. This model was shown to be very close (within parameter measurement tolerances) to the un-abbreviated model but with a significant reduction in computational burden.

The explicit nonlinear torque equations used in obtaining this paper's results were

generated in the widely available software package, Maple [39], using a symbolic Newton-Euler [19] formulation. Maple offers the ability to write out equations in an optimized format (by eliminating redundant calculations) in either the C or Fortran programming languages. The entire process, from entering in PUMA 560 input data until the computer code is generated, takes about eight minutes. Clearly, employing symbolic processing software in this manner provides a substantial time savings for the control engineer.

C. Controller Gain Matrices

Given the equations for calculating the nominal feedforward torques, τ^*, we now turn our attention to deriving the controller compensating torque, $\delta\tau$. Figure 1 illustrates the feedback loop. We assume an error-based compensating torque such that trajectory perturbations away from the nominal state are driven toward zero. Therefore,

$$\delta\tau = -\Gamma_1\delta q - \Gamma_2\delta\dot{q} \tag{9}$$

such that,

$$\delta\dot{x} = [A(t) - B(t)\Gamma(t)]\delta x(t) \tag{10}$$

models the closed-loop system.

It is desirable to calculate the feedforward nominal torques and the controller gains on-line. In the past, however, on-line gain calculations for robotic control systems have usually been too computationally expensive for higher degree of freedom manipulators. One reason for this is that calculating the linearized robot model, equations (6), is more complex than calculating the nonlinear model [40],[41]. However, a priori knowledge of inertial and kinematic parameters allows us to develop abbreviated linear models. An explicit,

symbolic linear model for the PUMA 560 (in terms of a nominal trajectory) which requires 257 multiplications and 158 additions is given in [38]. Parameters were taken from [37]. It was shown that this model does a good job of matching the dominant open-loop poles of the system and that non-dominant eigenvalue mismatches between the abbreviated and un-abbreviated models are of little consequence in control system design applications.

We now derive the MIMO pole placement gain matrix which is used in Figure 1 through equation (7). There are several techniques available for MIMO pole placement (see, for example, [42] or [43]). This paper utilizes the method presented in [28]. The method is attractive because it lends itself to symbolic implementation, the gain matrix requires no matrix inversion, and no expensive linear algebra permutations such as singular value decomposition are necessary.

Results for a digital PID controller were presented in [28]. In this paper, a PD controller is utilized because integrator action is not desirable for human/robot interaction where there are no final steady-state positions to be reached. Therefore, a derivation for the PD gain matrix, based on the methodology in [28], will now be presented for both continuous and discrete applications.

Upon applying position and velocity error feedback to equation (7), the new closed-loop system is represented as

$$\delta\dot{x} = (A - B\Gamma)\delta x \tag{11}$$

where

$$u = \delta\tau = -B\Gamma \tag{12}$$

is the error based compensating torque and $\Gamma=[\Gamma_1|\Gamma_2] \in \Re^{n,2n}$ is the feedback gain matrix.

The closed-loop system matrix then becomes

$$A - B\Gamma = \begin{bmatrix} 0 & I \\ -M^{-1}(K + \Gamma_1) & -M^{-1}(C + \Gamma_2) \end{bmatrix} \tag{13}$$

The gain matrices, Γ_1 and $\Gamma_2 \in \Re^{n,n}$, can be easily determined since the closed-loop system characteristic equation

$$\det[SI - (A - B\Gamma)] = \det(SI - \Lambda) \tag{14}$$

where $\Lambda=[\lambda_1|\lambda_2] \in \Re^{2n,2n}$ is a diagonal matrix of desired closed-loop poles. Expanding equations (13) and (14) yields

$$|S^2I + M^{-1}(C + \Gamma_2)S + M^{-1}(K + \Gamma_1)| = |S^2I - (\lambda_1 + \lambda_2)S + \lambda_1\lambda_2| \tag{15}$$

We can match coefficients to solve for the gain submatrices, Γ_1 and Γ_2,

$$\begin{aligned} \Gamma_1 &= -K + M\lambda_1\lambda_2 \\ \Gamma_2 &= -C - M(\lambda_1 + \lambda_2) \end{aligned} \tag{16}$$

The diagonal entries in the desired pole submatrices, λ_1 and $\lambda_2 \in \Re^{n,n}$, can then be specified according to the desired closed-loop system response.

This gain matrix derivation is for a continuous system. We can determine a gain matrix for discrete applications by re-writing equation (11) as

$$\delta x_{i+1} = \Phi\delta x_i + \Psi\delta u_i \tag{17}$$

where $\Phi \in \Re^{2n,2n}$ and $\Psi \in \Re^{2n,n}$ are the discretized system matrices, A and B, respectively. We use a first order Euler finite difference approximation [44] to the state derivatives such that

$$\Phi = I + Ah$$
$$\Psi = Bh \tag{18}$$

where h is the digital sampling interval. Using equation (18), the discrete system open-loop matrices can now be written as

$$\Phi = \begin{bmatrix} I & Ih \\ -M^{-1}Kh & -M^{-1}Ch + I \end{bmatrix} \quad , \quad \Psi = \begin{bmatrix} 0 \\ M^{-1}h \end{bmatrix} \tag{19}$$

The closed-loop system then becomes

$$\Phi - \Psi\Gamma = \begin{bmatrix} I & Ih \\ -M^{-1}h(K + \Gamma_1) & -M^{-1}h(C + \Gamma_2) + I \end{bmatrix} \tag{20}$$

The gain matrices, Γ_1 and Γ_2, are again determined with

$$\det[ZI - (A - B\Gamma)] = \det(ZI - \Lambda) \tag{21}$$

Expanding equations (20) and (21) yields

$$\begin{aligned} |Z^2I + (-2I + M^{-1}Ch + M^{-1}\Gamma_2 h)Z + M^{-1}h(-C - \Gamma_2 + Kh + \Gamma_1 h) + I| \\ = |Z^2I - (\lambda_1 + \lambda_2)Z + \lambda_1\lambda_2| \end{aligned} \tag{22}$$

by which we can match coefficients to solve for the gain submatrices, Γ_1 and Γ_2,

$$\Gamma_1 = -K + \frac{M}{h^2}(\lambda_1\lambda_2 - \lambda_1 - \lambda_2 + I)$$
$$\Gamma_2 = -C + \frac{M}{h}(2I - \lambda_1 - \lambda_2) \tag{23}$$

At this point the time varying gain matrices, equation (16) or (23), are used to regulate the closed-loop system perturbations from the nominal operating point toward zero as the

robot moves along its desired trajectory. With proper choice of time varying gains, we can make the closed-loop system time invariant [23]. The linearized system matrices, $M(q^*)$, $C(q^*, \dot{q}^*)$, and $K(q^*, \dot{q}^*, \ddot{q}^*)$ could be evaluated at each sampling interval and used in equation (16) or (23). But using explicit symbolic linearized equations as in [38] allow this step to be eliminated. Rather, the explicit linearized equations can be utilized directly in equation (16) or (23) to obtain a complete symbolic representation for the system gain matrices in terms of the desired closed-loop system eigenvalues.

It is interesting to examine the relationship between traditional computed torque controllers given in equations (2)-(4) and the controller derived in this section. Upon expanding equation (2) into its nominal torque and perturbation torque components, we find that

$$\delta\tau = M(q^*)\Gamma_1\delta q + M(q^*)\Gamma_2\delta\dot{q} \tag{24}$$

The characteristic equation becomes

$$|S^2 I + \Gamma_2 S + \Gamma_1| = 0 \tag{25}$$

where

$$\begin{aligned} \Gamma_1 &= M(q^*)\lambda_1\lambda_2 \\ \Gamma_2 &= -M(q^*)(\lambda_1 + \lambda_2) \end{aligned} \tag{26}$$

Note the similarities between equations (16) and (26). Essentially, traditional computed torque schemes neglect the contributions of the linear system matrices, C and K, to the closed-loop eigenvalues. If the Coriolis/centrifugal terms are significant, these differences could become important at high speeds. Section 4 reveals that at high joint velocities and

accelerations, the closed-loop eigenvalues of traditional computed torque controllers must be placed further in the left half plane to maintain stability robustness than when using the controller presented in this paper. Placing the desired poles further in the left half plane leads to higher gains which may be undesirable.

The results of this section have two important implications. First of all, the entire controller development process, from symbolic feedforward torque computation to symbolic gain matrix derivation, is automated. The base robot parameters are entered into a set of Maple macros which derive the appropriate equations and then write out optimized computer code for immediate implementation. The second implication is that access to the symbolic gain matrix equations facilitates elimination of redundant calculations. This offers substantial savings in computational complexity over evaluating the linearized system matrices numerically, as in [41], followed by the additional step of using these matrices to solve equation (16) or (23). Computer generated control laws offer a substantial savings both in development time and calculation time, allowing the control engineer to quickly develop efficient and intuitive control laws.

IV. ROBUSTNESS ANALYSIS

An important concern when designing any robotic control system is the effect of mismatches between the theoretical model and the real robot. There are a number of sources for these mismatches including, measurement noise, unmodeled payloads and other dynamics, and linearizing about points which are far from the actual trajectory because of such things as actuator saturation, external disturbances, and inaccuracies from using a first order Euler difference to discretize the continuous system. Specifically, this section studies the impact

of these errors and external disturbances on closed-loop systems controlled by computed torque/pole placement techniques based on abbreviated trajectory linearization models. We proceed first by deriving some general stability results for controllers of the type developed in Section III followed by a detailed analysis of the PUMA 560.

A. Stability Analysis

The gain matrices in Section III can be re-written as

$$\begin{aligned} \Gamma_1 &= -\hat{K} + \hat{M}\lambda_1\lambda_2 \\ \Gamma_2 &= -\hat{C} - \hat{M}(\lambda_1 + \lambda_2) \end{aligned} \tag{27}$$

for the continuous time system and

$$\begin{aligned} \Gamma_1 &= -\hat{K} + \frac{\hat{M}}{h^2}(\lambda_1\lambda_2 - \lambda_1 - \lambda_2 + I) \\ \Gamma_2 &= -\hat{C} + \frac{\hat{M}}{h}(2I - \lambda_1 - \lambda_2) \end{aligned} \tag{28}$$

for the discrete time system, where $\hat{M}$, $\hat{C}$, and $\hat{K}$ represent the modeled linearized system matrices. Substituting equations (27) and (28) into equations (15) and (22) and simplifying yields the closed-loop characteristic equations in terms of the modeling errors. We assume a specific structure for the modeling errors, namely that

$$\hat{M} = \gamma M, \quad \hat{C} = \beta C, \quad \hat{K} = \alpha K \tag{29}$$

where $\gamma \in [0,\infty)$, $\beta \in (-\infty,\infty)$, $\alpha \in (-\infty,\infty)$ are scale factors which represent modeling errors and external disturbances which alter each linear system matrix. This structure is useful because it simplifies our analysis. We have also determined that it generally leads to stability

predictions which are conservative when compared with other modeling error structures. We think of γ, β, and α as the multiplicative factor derived from the largest percentage error between a modeled system matrix element and the actual system matrix element. Other matrix elements are multiplied by this factor even though in reality the other matrix element errors may not be as great. Note that traditional computed torque schemes imply $\alpha=\beta=0$. The closed-loop characteristic equation can be written in second order block matrix form as

$$\left|S^2 I + \left[(1 - \beta)M^{-1}C - \gamma(\lambda_1 + \lambda_2)\right]S + \left[(1 - \alpha)M^{-1}K + \gamma\lambda_1\lambda_2\right]\right| = 0 \qquad (30)$$

for a continuous system and

$$\begin{aligned}\left|Z^2 I + \left[-2I + \gamma(2I - \lambda_1 - \lambda_2) + M^{-1}Ch(1 - \beta)\right]Z\right. \\ \left. + \left[I(1 - \gamma) + \gamma\lambda_1\lambda_2 + M^{-1}Kh^2(1 - \alpha) - M^{-1}Ch(1 - \beta)\right]\right| = 0\end{aligned} \qquad (31)$$

for a discrete system.

The structure of equations (30) and (31) allow some qualitative statements about system stability to be made. First, note that the mass matrix is always positive definite in robotic systems. Therefore, γ is always greater than zero for a proper choice of the assumed inertial matrix, $\hat{M}$. Also, the desired pole submatrices, λ_1 and λ_2, are negative definite since a control system designer would not, in general, choose unstable poles for the closed-loop system.

Continuous system stability is assured by moving the desired pole locations further into the left half plane. Equation (27) shows that this is tantamount to increasing the controller gains. Also, note that errors in K must grow quadratically to influence stability as the desired closed-loop poles are increased. Also, choosing the modeled inertial matrix, $\hat{M}$, to be larger

than the actual system inertial matrix, M, increases stability. Again, equation (27) shows that increasing γ's lead to higher controller gains which makes intuitive sense. In the past, researchers ([7] and [45], for example) have noticed better results from their control algorithms when a payload is assumed even when none is present. An assumed payload when none is present is analogous to a $\gamma>1$. Therefore, the control system can be derived in terms of the maximum expected payload and continuous system stability and performance would not be negatively impacted when less payload is present. [12] gives more rigorous details using a Lyapunov stability analysis to show the stabilizing influence of a large assumed inertia matrix on computed torque controllers.

The above results are, in general, valid for the discrete time system as well. Obviously, however, the closed-loop gains for a discrete system can not be arbitrarily increased or instability will result as the closed-loop bandwidth approaches the digital sampling interval [44]. Also, high controller gains which increase the closed-loop system bandwidth can lead to problems with noise at higher frequencies. Equation (31) shows that, as the sampling interval is decreased, errors in the C and K linear system matrices become less important. As the sampling interval becomes very small, the discrete system will converge to the continuous system assuming negligible system noise and good numerical precision.

To help quantify some of the qualitative observations made here and by past authors, the next section uses a PUMA 560 for a numerical case study of the performance of both the continuous and discrete systems.

B. Numerical Analysis for the PUMA 560

The advantage of using a controller based on linearized manipulator equations of motion is that stability and performance analyses can be based on the closed-loop system eigenvalues.

Eigenvalue analysis offers a more intuitive view of the robot control system performance because it can be used to quantitatively evaluate the shift in closed-loop pole locations due to unknown parameter or modeling errors. While nonlinear techniques, such as Lyapunov analysis, are useful in determining robustness in terms of stability, their utility in determining performance robustness (ie, the changes in system response time and damping caused by parameter/modeling errors) is more limited. With explicit knowledge of the closed-loop eigenvalues, performance based on traditional second order system analysis is quite useful. This section presents several closed-loop root locus plots to illustrate the effects of modeling/parameter errors on controller performance as measured by pole location.

Consider the PUMA 560 at a particular trajectory point. From equations (30) and (31) it becomes apparent that the closed-loop system response is progressively effected by percentage errors in the elements of the linear system matrices C and K as the norms of these matrices become large. Although, these matrix norms are theoretically unbounded if the robot's joint rates and accelerations are unbounded, in practice joint rates and accelerations are bounded due to actuator limits. Table I presents a trajectory point and the linear system matrices evaluated based on a linear Taylor series expansion of the nonlinear equations of motion about this trajectory point (see [38] for further details). Note that the matrices in Table I were evaluated numerically with no abbreviation or simplification. Also, note that this trajectory point is unattainable by a standard PUMA with the maximum motor torques given in [37]. However, this particular trajectory point is useful because it serves as a worst case by which to evaluate robustness. It is well known that high speed trajectory following represents a much greater challenge for the robot control engineer than low speed trajectory following does. Also, motor dynamics become important when controlling the PUMA 560

[46]. These dynamics are accounted for here by adding the appropriate terms to the diagonal of the linear system matrices M (motor inertias) and C (back emf).

Figure 3a presents a root locus of the continuous system with the error parameter, α, varied from -3 to +3 with desired closed-loop poles all at -4.5±4.5j (natural frequency @ 1 Hz, damping @ 70% critical). Figure 3b presents a root locus of the continuous system with the error parameter, α, varied from -150 to +223 with desired closed-loop poles all at -45±45j (natural frequency @ 10 Hz, damping @ 70% critical). The range of stability with closed-loop poles at -4.5±4.5j is approximately $-0.4 < \alpha < 2.6$ and with closed-loop poles at -45±45j it is $-127.5 < \alpha < 160.0$. Note, as mentioned previously, that errors in the linear system matrix, K, become exponentially less important as the closed-loop poles are moved further out into the left half plane. However, even the 1 Hz system requires large errors before an eigenvalue moves into the right half plane.

Figure 4a presents a root locus of the continuous system with the error parameter, β, varied from -6 to +6 with desired closed-loop poles all at -4.5±4.5j. Figure 4b presents similar results with the error parameter β varied from -60 to +60 with desired closed-loop poles all at -45±45j. The range of stability with closed-loop poles at -4.5±4.5j is $-\infty < \beta < 4.8$ and with closed-loop poles at -45±45j it is $-\infty < \beta < 39.0$. The importance of errors diminishes in a linear fashion when the desired closed-loop eigenvalues are placed further in the left half plane.

Figure 5a presents a root locus of the continuous system with the error parameter, γ, varied from 0 to +7.25 with desired closed-loop poles all at -4.5±4.5j. Figure 5b presents similar results with the error parameter γ varied from 0 to +7.25 with desired closed-loop poles all at -45±45j. The range of stability with closed-loop poles at -4.5±4.5j is

$0 < \gamma < \infty$ and with closed-loop poles at -45±45j it is also $0 < \gamma < \infty$. With γ=0, all roots lie at the origin as expected. For the continuous system, increasing the norm of the modeled inertial matrix relative to the actual inertial matrix leads to roots which are further in the left half plane, however, γ cannot grow unbounded in the discrete system or instability will result for a given sampling interval as the system gains are increased. Figure 6 shows stability results for the discrete system. Notice that Figure 6 shows roots on a typical left/right plane rather than on the unit circle as is normally done for discrete systems. Mapping between the *s*- and *z*-planes is done according to

$$z = e^{sh} \tag{32}$$

As h becomes smaller, the *z*-plane roots converge to +1, which usually indicates marginal stability. However, what is really happening is that the discrete system is converging to behave like the continuous system. Although at sampling intervals on the order of machine numerical precision we can expect numerical problems to lead to instability, equation (32) indicates stability problems at much larger sampling intervals. Take as an example a 10 Hz system with continuous roots at -45±45j and a sampling frequency of 1000 Hz. Equation (32) leads to *z*-plane roots at 0.96±0.04j indicating that even small perturbations will lead to instability. This is clearly a poor prediction as numerical simulation of a simple second order control system will demonstrate. [47] calls for using δ transforms to unify stability analyses between the continuous and discrete domains. Figure 6, however, simply converts the discrete poles back into the continuous domain for ease of comparison with Figures 3a-5b. Note that the continuous system given in Table I is discretized exactly according to

$$\Phi = e^{Ah}$$
$$\Psi = \int_0^h e^{As} B\, ds \tag{33}$$

rather than with the first order Euler approximation of equation (18). This allows us to determine the adequacy of the Euler approximation used to derive the discrete controller gains in equation (23).

Figure 6 demonstrates that instability will eventually result as γ is increased above 4.55 when sampling at 200 Hz. Also, instability is predicted as γ gets very close to zero. Discrete results for varying α and β are substantially similar to those presented in Figures 3a-4b with the exception that the stability limits are slightly smaller depending on sample rate. Clearly, if the sampling rate is decreased too much, say to below the Shannon rate [44], instability will result. Likewise, as the sample rate is increased the results will converge to the continuous results.

Finally, Table II shows the differences in continuous system eigenvalue placement between the method presented in this paper and the traditional computed torque technique (α=β=0). Examination of equations (16) and (26) reveals that the gains derived from the pole placement technique presented in this paper and the gains derived from the traditional computed torque scheme converge as the desired closed-loop poles are moved very far into to the left half plane.

This section has shown that PUMA 560 controllers based on trajectory linearization are robust against linear system modeling errors. However, the PUMA robot is highly geared and motor dynamic properties such as inertia and friction are very pronounced. Other linearized robot models, especially those with direct drive transmission systems, should be

studied in a similar fashion. We expect high speed direct drive systems utilizing trajectory linearization models to show greater performance improvements over standard computed torque methods as the Coriolis and centrifugal terms grow in relative importance.

V. SIMULATION AND EXPERIMENTS WITH A PUMA 560

This section presents simulation and experimental results for the computed torque and pole placement method presented in Section III. These results demonstrate the applicability of this method in a real-time application. We begin with a brief overview of prior work, followed by details of our experimental setup some results.

A. Overview

Many control algorithms found in the literature have been demonstrated with simulation. These demonstrations are often performed with manipulators having two or three degrees of freedom. For example, simulation results for generic two degree of freedom manipulators can be found in [8], [25], [48], and [49] while results for two degree of freedom SCARA-type robots are given in [50] and [51]. Simulation results for joints two and three of a PUMA robot can be found in [3] and [26]. Simulation results for the first three joints of a PUMA robot are given in [1], [5], [18], [24], [27], and [45]. Other three degree of freedom robots which have been simulated under control are presented in [4], [7], and [28]. A rare example which contributes findings for all six degrees of freedom for a PUMA 600 is [23].

Simulation is a useful tool for analysis of various control system concepts but it often masks difficulties with real-time implementation. Therefore, it makes sense to rigorously test and validate these algorithms in an experimental setting. Many important applications require

six degrees of freedom. A majority of prior experimental data includes results from two or three degree of freedom systems. For example, experimental results for a two degree of freedom SCARA robot are found in [51], the elbow and wrist joints of a Toshiba-500V robot were tested in [52], and data for joints two and three of a PUMA 560 is presented in [53]. Non-PUMA three degree of freedom experiments have been performed as well [54],[55]. The most common experimental results published to date are for the first three joints of the PUMA robot (see for example, [6], [46], and [56]). Rare examples include [2] and [57] which present results for all six degrees of freedom for a PUMA 560.

There is a need for more test and simulation results for robotic systems utilizing more than three degrees of freedom. A contribution of this paper is to add to the literature further control system results for all six degrees of freedom for the PUMA 560 robot. Another contribution of this paper is to compare simulation results and experimental results in order to study the accuracy of our computer models. The issue of simulation validation is rarely addressed in the robotics literature. This is surprising given the large numbers of simulations currently being used to analyze and design these systems. A rare example where both simulation and experimental results are given is in [45]. If simulation is to be used as an effective evaluation tool, techniques must be developed to help gain confidence that numerical results accurately and reliably predict actual system performance. Such validation techniques are available in other areas such as the aerospace and automotive fields (see, for example, [58]).

B. Experimental setup

The LSI/11 VAL computer and servo cards in the PUMA have been replaced with a Trident Robotics TRC004 general purpose interface board which allows direct access to motor torques and encoders. A Trident TRC006 interface card serves as the I/O link between the TRC004 and a Pentium 90 Mhz personal computer. Watchdog timers are built into the system which shut down power to the robot in the case of hardware or software crashes. The Pentium computing capacity coupled with the Trident system enables very high (greater than 1000 Hz) servo loop update rates. Recently, authors such as [59], are realizing much greater real-time performance in terms of servo loop update rate by using digital signal processing chips (DSP's) or fast PC chips such as the Pentium. Indeed, [30] and [59] have reported Cartesian controllers that are capable of updating at speeds over 1000 Hz and include complicated forward dynamics, forward/inverse kinematics, Jacobian, and feedback loop calculations for the PUMA.

The main obstacle in attaining high sample rates has shifted from computing power to encoder resolution. In order to compute servo rate errors, backward finite difference is usually used. However, at high sample rates, poor resolution will lead to noisy finite difference quantities and to controller instabilities. Filtering the rate signals can help alleviate this problem but our applications are in human/robot interaction where trajectories can vary between high and low speeds. This means that no one set of filter characteristics is optimal over all trajectory speeds. Thus, adaptive filtering may improve performance by changing the filter response based on the trajectory. However, in this paper, the sampling rate was simply lowered to 500 Hz and a finite difference approximation is used for rate error.

When evaluating control strategies, the use of standard trajectories allow for greater

objectivity when analyzing experimental or simulation results [11]. A trajectory which exercises the first three joints of the PUMA 560 through a good portion of its dynamic range is given in [11]. This trajectory is a reasonable choice for algorithm comparison. Figures 7a-7d present this trajectory along with trajectories for the last three joints of the PUMA that were chosen.

C. Results

This subsection presents some experimental results which we have obtained using the methodology presented in Section III. Again, nominal feedforward torques are calculated using an abbreviated inverse dynamics model presented in [38]. The feedback gain matrices are based on an abbreviated linear PUMA model also presented in [38]. These feedback gains were automatically derived in terms of the PUMA 560 inertial and kinematic parameters. Nominal trajectory positions, velocities, and accelerations are left as variables that continuously change as the robot moves through its desired trajectory. Also, desired pole locations are left as variables and are user specified. The appendix presents explicit gain matrices which were automatically generated with symbolic processing software. Both continuous and discrete time equations are given. When the gain matrices are written out in optimized format with redundant calculations eliminated, the continuous gains require 336 multiplications and 186 additions while the discrete gains require 336 multiplications and 197 additions. (Note that, to make them easier to read, the appendix equations are not presented in optimized format.) Therefore, the feedforward and feedback loops can be updated with less than 700 total calculations indicating that this controller is quite efficient for robustly controlling all six degrees of freedom of the PUMA 560. Furthermore, since the controller gains are calculated on-line, this method offers much greater flexibility than algorithms which

require that gains be calculated off-line with a priori trajectory knowledge.

We compare three controllers (plus simulation results) here, the trajectory linearized controller presented in this paper (denoted as CT1) and two other traditional computed torque controllers. The subsequent two controllers are similar to each other in structure except that the first controller (denoted as CT2) uses the full inverse dynamics model while the second (denoted as CT3) uses only the diagonal terms of the inertia matrix and the gravity and frictions terms in the feedforward loop. The simulated results (denoted as SIM) are for controller CT1. (The un-abbreviated differential equations governing the motion of the PUMA were solved with numerical integration using a method presented in [60].)

It is important to note that all three controllers were computer generated and derived automatically. Compiler-ready code was created in about an hour with simulation used for final code check out before implementation on the PUMA. Since the symbolic processing software is generic, it accepts inertial and kinematic parameters for any robot.

Figures 8a-8f present measurements along with numerical results. The desired closed-loop eigenvalues were chosen to be -12±45j which gave reasonable tracking performance. Note that the continuous time gains were used as found in the appendix. (The high sample rate that the system uses leads to continuous and discrete time gains which are very close in magnitude.) The real portion of these roots was chosen below the threshold where finite differencing the encoders at 500 hz leads to instability due to resolution problems. A series of five tests were run for each controller with the results averaged for presentation here. Note that the magnitudes of the error profiles are of less importance than the relative difference between them. The figures show that all of the controllers do a good job of minimizing position errors. Figure 9 demonstrates that these joint space errors correspond to Cartesian

space position errors of less than 0.3 millimeters. As expected, the simulation results show smaller errors. The differences between the simulation results and the test results indicate there are unmodeled dynamics or incorrect parameters. However, these differences are so small that they don't give a clear indication of how to improve the simulation.

All of the position error profiles are fairly close to one another. However, as expected, the CT1 controller appears to do a slightly better job at points where Coriolis and centrifugal components are important. Table II indicates that the CT1 and CT2 controllers will behave similarly for the gains chosen here. The last three joint error profiles are nearly identical. This is to be expected as the dynamics of these joints are essentially uncoupled.

We are also interested in the acceleration errors since one of our goals is using the PUMA as a dynamic force interaction device. However, finite differencing the rate data (which is itself derived from finite difference data) leads to results which are too noisy. The algorithm presented in [61] is used for the rate data to obtain smooth acceleration results. Figures 10a-10f indicate that the differences among controllers are not clearly distinguishable. A much wider range of test trajectories may be necessary to determine acceleration control performance capabilities.

VI. DISCUSSION AND CONCLUSIONS

This paper has demonstrated the utility of symbolic processing software used to quickly generate complex robotic control software. In particular, a feedback controller has been derived based on time varying trajectory linearization about a nominal operating point. By using abbreviated symbolic models, fewer than 700 hundred calculations are required for both the feedforward and feedback loops, a computational burden easily handled in fast sampling,

real-time systems with modern computers. Straightforward eigenvalue analyses have been used to demonstrate stability and performance robustness in the presence of bounded modeling errors and disturbances.

This paper compares three computed torque controllers (all derived automatically with symbolic processing software) in an experimental test bed using a previously defined standardized test trajectory which exercises a PUMA 560 through much of its dynamic range. Experimental and simulation results are given for all six PUMA degrees of freedom. Eigenvalue analysis indicates that, for the PUMA, system performance when using traditional computed torque schemes versus those based on trajectory linearization will converge in the limit as gains are increased or as speed and acceleration are decreased. This result is born out experimentally as well.

A final analysis indicates that computed torque methods, as a whole, perform quite well. This well known result has been verified by others. However, a controller based on trajectory linearization as presented in this paper does not demonstrate significant performance improvements when used with a standard PUMA 560 configuration. For instance the PUMA in our lab is fuse limited such that its peak motor torque capabilities are halved. This, coupled with motor back emf, limits the maximum joint velocities attainable. Controllers based on trajectory linearization will show performance improvements as speed is increased. Direct drive systems should also see improvements as well because inertial dynamics generally decrease in relative magnitude compared with Coriolis and centrifugal dynamics. However, this remains to be seen as future work incorporates this type of controller in other robotic systems.

REFERENCES

[1] M. Jamshidi, H. Seraji, and Y.T. Kim, "Decentralized control of nonlinear robot models," *Robotics*, vol. 3, no. 3-4, pp. 361-370, 1987.

[2] Z.S. Tumeh, "Formulation and experimental validation of two decentralized discrete model-referenced adaptive manipulator control strategies," in *Proceedings of the 1992 American Control Conference* (Chicago, IL) vol. 4, June 1992, pp. 2936-2940.

[3] S. Zein-Sabatto and G.E. Cook, "Dynamic restructuring mechanisms for robot manipulator controls," in *Proceedings of IEEE SOUTHEASTCON '92* (Birmingham, AL), vol. 1, April 1992, pp. 418-421.

[4] S. Cetinkunt and R.L. Tsai, "Accurate, robust, adaptive control in contour tracking applications of Robotic Manipulators," *Robotics and Computer-Integrated Manufacturing*, vol. 8, no. 1, pp. 45-51, 1991.

[5] K. El Serafi and W. Khalil, "Energy based adaptive robot controller," in *Proceedings of the International Workshop on Nonlinear and Adaptive Control: Issues in Robotics* (Grenoble, France), November 1991, pp. 30-48.

[6] M.B. Leahy, D.E. Bossert, and P.V. Whalen, "Robust model-based control: an experimental case study," in *Proceedings of the 1990 International Conference on Robotics and Automation* (Cincinnati, OH), May 1990, pp. 1982-1987.

[7] V.D. Tourassis and C.P. Neuman, "Robust nonlinear feedback control for robotic manipulators", *IEE Proceedings: Part D, Control Theory and Applications*, vol. 132, pp. 134-143, July 1985.

[8] T. Ishihara, "Direct digital design of computed torque controllers," *Journal of Robotic Systems*, vol. 11, no. 3, pp. 197-209, 1994.

[9] Y.H. Chen, "Robust computed torque schemes for mechanical manipulators: nonadaptive versus adaptive," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 113, June 1991.

[10] W. Ham, "Adaptive control based on explicit model of robot manipulator," *IEEE Transactions on Automatic Control*, vol. 38, no. 4, pp. 654-658, April 1993.

[11] M.B. Leahy, "Experimental analysis of robot control: a performance standard for the PUMA-560," in *Proceedings of the IEEE Symposium on Intelligent Control* (Albany, NY), September 1989, pp. 257-264.

[12] F.L. Lewis, C.T. Abdallah, D.M. Dawson, *Control of Robot Manipulators*. New York: Macmillan Publishing Company, 1993.

[13] E. Freund, "Fast nonlinear control with arbitrary pole placement for industrial robots and manipulators," *The International Journal of Robotics Research*, vol. 1, no. 1, pp. 65-78, 1982.

[14] N. Coleman, N.K. Loh, and Y.L. Gu, "Optimal control with exact linearization of robotic manipulators," in *The Proceedings of the 1986 American Control Conference* (Seattle, WA), 1986, pp. 852-857.

[15] K. Kreuntz, " On manipulator control by exact linearization," *IEEE Transactions on Automatic Control*, vol. 34, no. 7, pp. 763-767, 1989.

[16] J.Y.S. Luh, M.W. Walker, and R.P. Paul, "Resolved-acceleration control of mechanical manipulators," *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 468-474, June 1990.

[17] C.S.G. Lee and M.J. Chung, "An adaptive control strategy for computer-based manipulators," in *Proceedings of the 21st IEEE Conference on Decision and Control* (Orlando, FL), pp. 95-100, December 1982.

[18] C.S.G. Lee and M.J. Chung, "Adaptive perturbation control with feedforward compensation for robot manipulators," *Simulation*, vol. 44, no. 3, pp. 127-136, March 1985.

[19] J.Y.S. Luh, M.W. Walker, and R.P. Paul, "On-line computational scheme for mechanical manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 102, no. 2, pp. 69-76.

[20] H. Seraji, "Linear multivariable control of robot manipulators," in *The Proceedings of the 1986 IEEE International Conference on Robotics and Automation* (San Francisco, CA), 1986, pp. 565-571.

[21] J.C.H. Chung and G.G. Leininger, "Task-level adaptive hybrid manipulator control," *The International Journal of Robotics Research*, vol. 9, no. 3, pp. 63-73, 1990.

[22] P.G. Backes, G.G. Leininger, and C.H. Chung, "Joint self-tuning with cartesian setpoints," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 108, June 1986.

[23] L. Guo and J. Angeles, "Controller estimation for the adaptive control of robotic manipulators," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 315-323, June 1989.

[24] A.A. Goldenberg, J.A. Apkarian, and H.W. Smith, "An approach to adaptive control of robot manipulators using the computed torque technique," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 111, no. 1, pp. 1-8, March 1989.

[25] S. Desa and B. Roth, "Synthesis of control systems for manipulators using multivariable robust servomechanism theory," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 19-34, 1985.

[26] A. Swarup and M. Gopal, "Robust trajectory control of a robot manipulator," *International Journal of Systems Science*, vol. 22, no. 11, pp. 2185-2194, 1991.

[27] P. Misra, R.V. Patel, and C.A. Balafoutis, "Robust control of robot manipulators using linearized dynamic models," in *The Proceedings of the International Symposium on Robot Manipulators: Modeling, Control and Education* (Albuquerque, NM), November 1986, pp. 117-123.

[28] R.J. Norcross, J.C. Wang, B.C. McInnis, and L.S. Shieh, "Pole placement methods for multivariable control of robotic manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 340-345, December 1986.

[29] L.S. Shieh and Y.T. Tsay, "Transformation of a class of multivariable control systems to block companion forms", *IEEE Transactions on Automatic Control*, vol. 27, no. 1, pp. 199-203, 1982.

[30] C.L. Clover, "A control system architecture for robots used to simulate dynamic force and moment interaction between humans and virtual objects," to be submitted to *IEEE Transactions on Systems, Man, and Cybernetics*.

[31] J.J. Craig, *Adaptive Control of Mechanical Manipulators*. Reading, MA: Addison-Wesley, 1988.

[32] J.J. Slotine and W. Li, "On the adaptive control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 3, pp. 49-59, 1987.

[33] B. Armstrong-Helouvry, *Control of Machines with Friction*. Boston:Kluwer, 1991.

[34] J.W. Burdick, "An algorithm for generation of efficient manipulator dynamic equations," in *The Proceedings of the 1986 International Conference on Robotics and Automation* (San Francisco, CA), 1986, pp. 212-218.

[35] C.P. Neuman and J.J. Murray, "Customized computational robot dynamics," *Journal of Robotic Systems*, vol. 4, no. 4, pp. 503-526, 1987.

[36] C. Li, "A new method of dynamics for robot manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, 1988.

[37] B. Armstrong, O. Khatib, J. Burdick, "The explicit dynamic model and inertial parameters of the PUMA 560 arm," in *The Proceedings of the 1986 International Conference on Robotics and Automation* (San Francisco, CA), 1986, pp. 212-218.

[38] C.L. Clover, "The utility of abbreviated linear and nonlinear robot models," to be submitted to *ASME Journal of Dynamic Systems, Measurement, and Control.*

[39] B.W. Char, K.O. Geddes, G.H. Gonnet, and S.M. Watt, *MAPLE User's Guide*, 4th edition, Waterloo, CA: WATCOM Publishing Ltd, 1988.

[40] C.P. Neuman, J.J. Murray, "Linearization and sensitivity functions of dynamics robot models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 14, no. 6, pp. 805-818, 1984.

[41] C.J. Li, "A new method for linearization od dynamic robot models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 1, pp. 2-17, 1990.

[42] B.C. Moore, "On the flexibility offered by state feedback in multivariable systems beyond closed loop eigenvalue assignment," *IEEE Transactions on Automatic Control*, vol. 21, no. 6, pp. 689-692, 1976.

[43] G.S. Miminis and C.C. Paige, "A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback," *Automatica*, Vol. 24, No. 3, pp. 343-356, 1988.

[44] K.J. Astrom and B. Wittenmark, *Computer Controlled Systems: Theory and Design.* Englewood Cliffs, NJ: Prentice Hall, 1990.

[45] S.N. Singh and A.A. Schy, "Robust Trajectory Following Control of Robotic Systems," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 107, pp. 308-315, December 1985.

[46] T.J. Tarn, A.K. Bejczy, X. Yun, and Z. Li, "Effect of motor dynamics on nonlinear feedback robot arm control," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 1, pp. 114-122, February 1991.

[47] R.H. Middleton and G.C. Goodwin, *Digital Control and Estimation: A Unified Approach.* Englewood Cliffs, NJ: Prentice-Hall, 1990.

[48] H.H. Lee and F.E. Culick, "Design of a robust adaptive control law for robotic manipulators," *Journal of Robotic Systems*, Vol. 11, No. 4, pp. 241-255, 1994.

[49] R. Kelly and R. Salgado, "PD control with computed feedforward of robotic manipulators: a design procedure," *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 4, pp. 566-571, August 1994.

[50] C.Y. Kuo and S.P.T. Wang, "Robust position control of robotic manipulator in Cartesian coordinates," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 5, pp. 653-659, October 1991.

[51] N. Sadegh, R. Horowitz, W.W. Kao, and M. Tomizuka, "A unified approach to the design of adaptive and repetitive controllers for robotic manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 112, pp. 618-629, December 1990.

[52] M. Tomizuka, R. Horowitz, G. Anwar, and Y.L. Jia, "Implementation of adaptive techniques for motion control of robotic manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 110, pp. 62-69, March 1988.

[53] Y. Zhou and C.W. de Silva, "Real-time control experiments using an industrial robot retrofitted with an open-structure controller," in *The Proceedings of the 1993 International Conference on Systems, Man, and Cybernetics* (Le Touquet, France, 1993), Vol. 4, pp. 553-559.

[54] P.K. Khosla and T. Kanade, "Real-time implementation and evaluation of computed-torque scheme," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 2, pp. 245-253, April 1989.

[55] T. Narikiyo and T. Izumi, "On model feedback control for robot manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 113, pp. 371-378, September, 1991.

[56] T.J. Tarn, S. Ganguly, A.K. Ramaorai, and G.T. Marth, "Experimental evaluation of the nonlinear feedback robot controller," in *The Proceedings of the 1991 International Conference on Robotics and Automation* (Sacramento, CA, April 1991), pp. 1638-1644.

[57] M.B. Leahy, K.P. Valavanis, and G.N. Saridis, "Evaluation of dynamic models for PUMA robot control," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 2, pp. 243-245, April 1989.

[58] J.E. Bernard and C.L. Clover, "Validation of computer simulations of vehicle dynamics," in *The Proceedings of the 1994 SAE International Conference and Exposition - Concepts in Vehicle Dynamics and Simulation* (Detroit, MI, March 1994), pp. 159-167.

[59] B.W. Drake and T.C.S. Hsia, "Implementation of a unified robot kinematics and inverse dynamics algorithm on a DSP chip," *IEEE Transactions on Industrial Electronics*, Vol. 40, No. 2, pp. 273-281, April 1993.

[60] M.W. Walker and D.E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 104, No. 3, pp. 205-211, 1982.

[61] H.J. Woltring, "A Fortran package for generalized, cross-validatory spline smoothing and differentiation," *Advances in Engineering Software*, Vol. 8, No. 2, pp. 104-107, 1986.

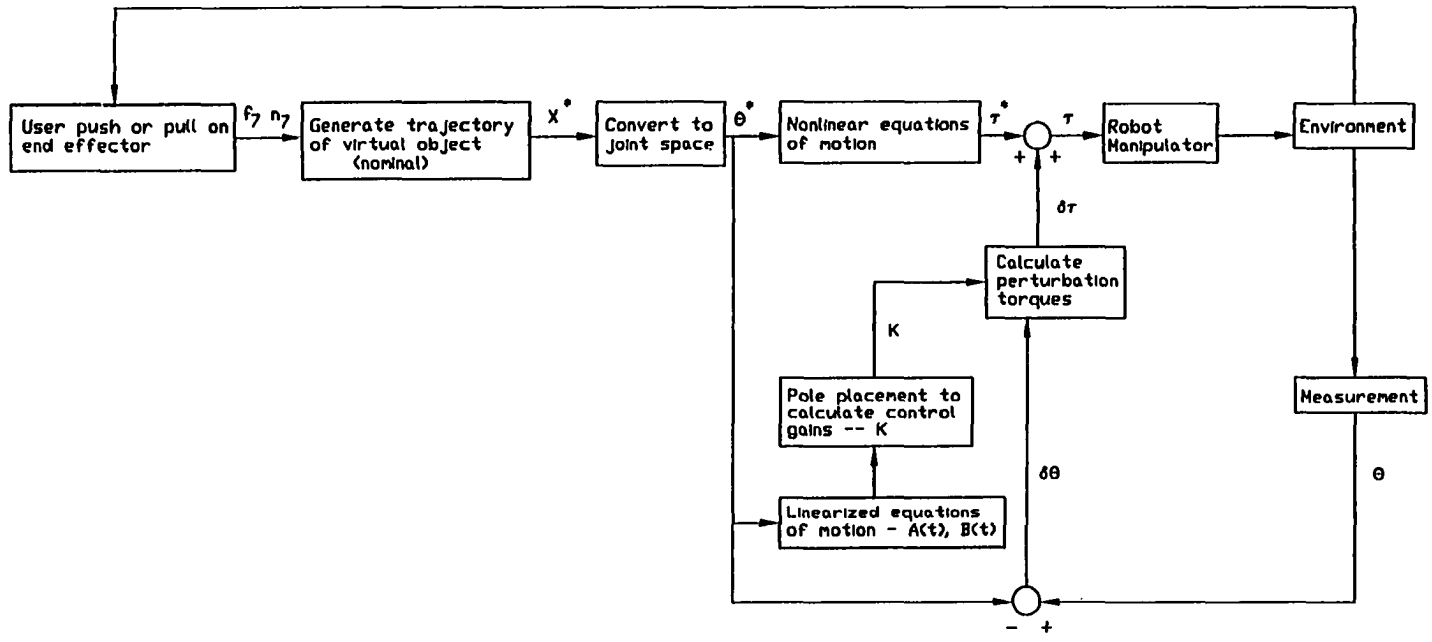

Figure 1: Controller logic for a force reflective robotic system

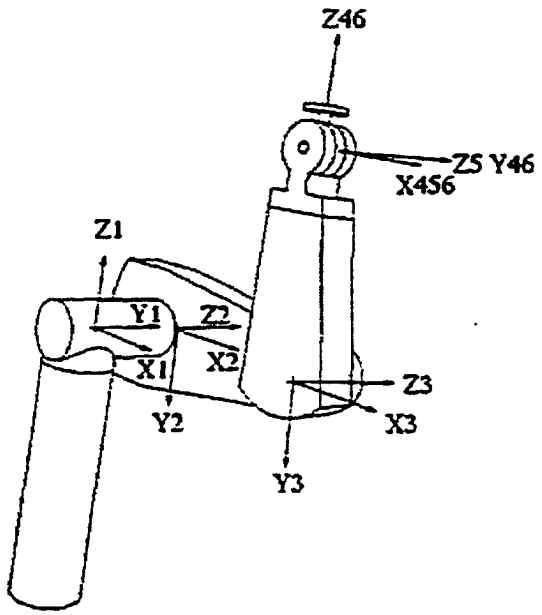

Figure 2: PUMA 560 coordinate axes

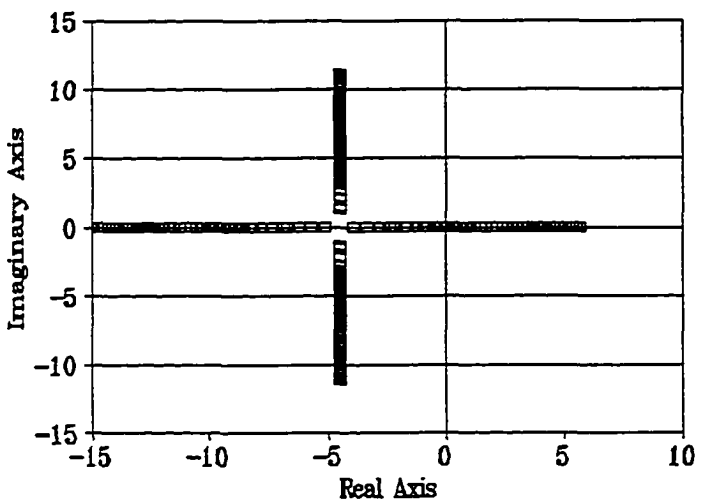

Figure 3a: Root Locus for varying α with desired poles at -4.5±4.5j (imaginary axis crossings at α=-0.4,2.6)

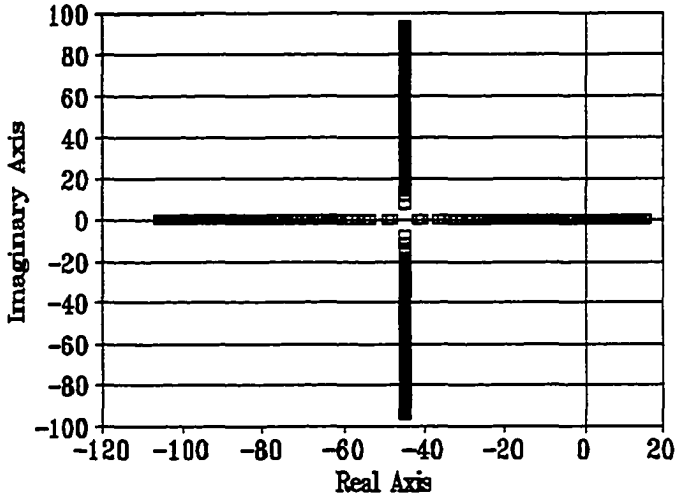

Figure 3b: Root Locus for varying α with desired poles at -45±45j
(imaginary axis crossings at α=-127.5,160.0)

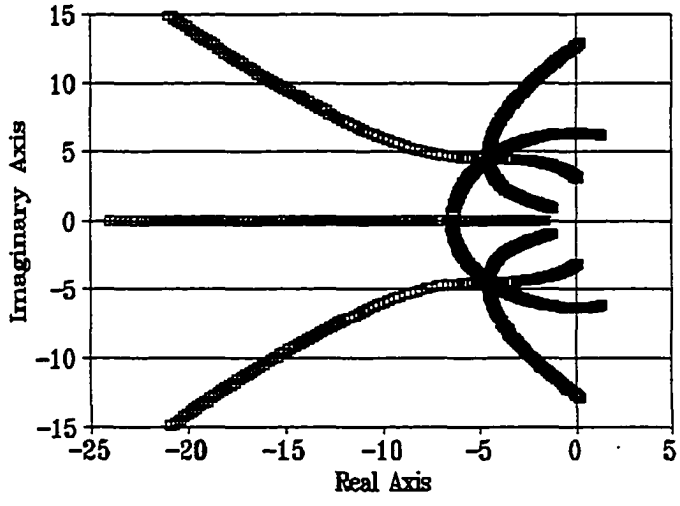

Figure 4a: Root Locus for varying β with desired poles at -4.5±4.5j
(imaginary axis crossing at β=4.8)

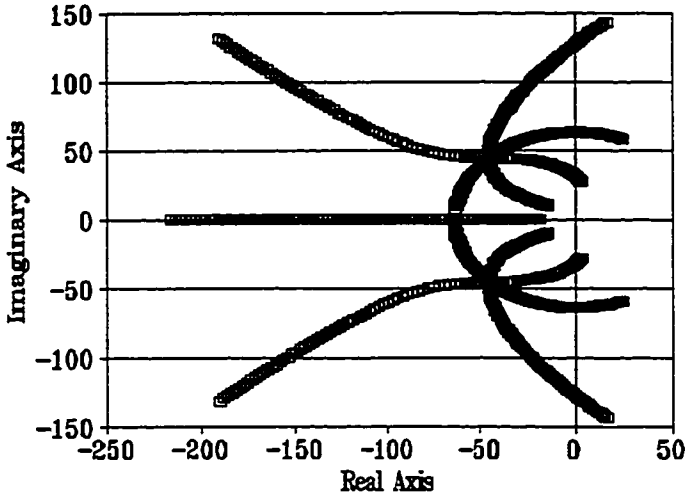

Figure 4b: Root Locus for varying β with desired poles at -45±45j
(imaginary axis crossing at β=39.0)

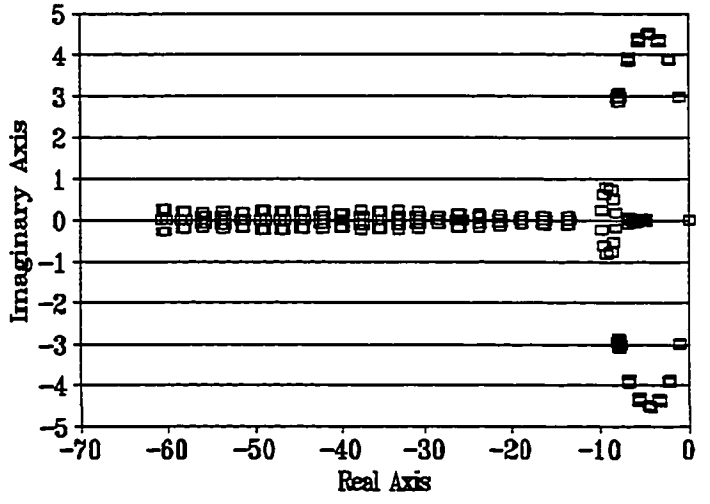

Figure 5a: Root Locus for varying γ with desired poles at -4.5±4.5j
(just touches imaginary axis at γ=0.0)

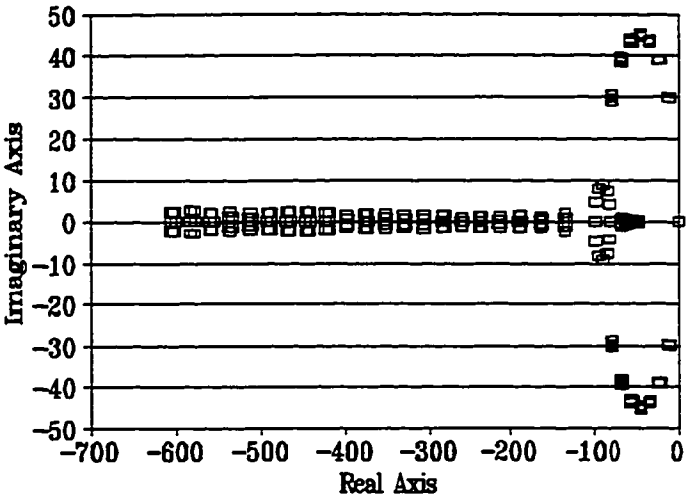

Figure 5b: Root Locus for varying γ with desired poles at -45±45j
(just touches imaginary axis at γ=0.0)

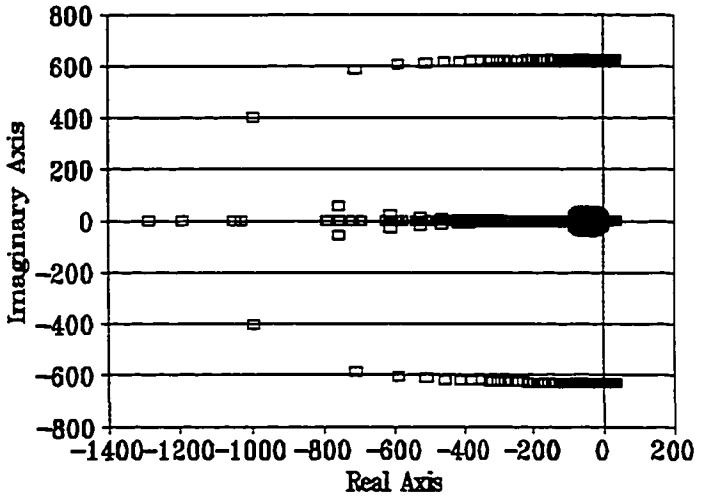

Figure 6: Root Locus for varying γ with desired poles at -45±45j assuming a 200 Hz sampling rate (imaginary axis crossings at γ=4.6 and γ<1)

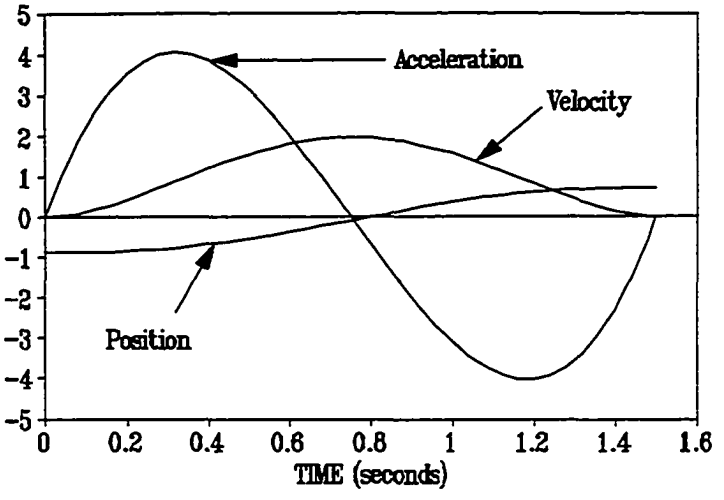

Figure 7a: Nominal trajectory for joints 1 and 4 (rad,rad/sec,rad/sec^2)

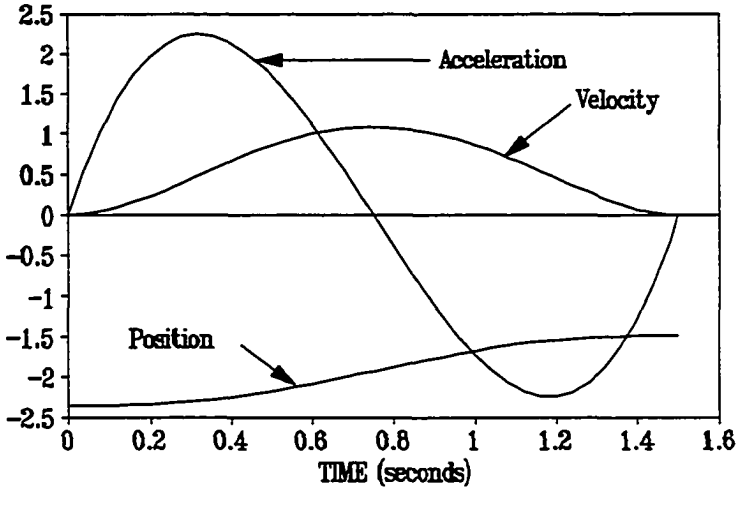

Figure 7b: Nominal trajectory for joint 2 (rad,rad/sec,rad/sec^2)

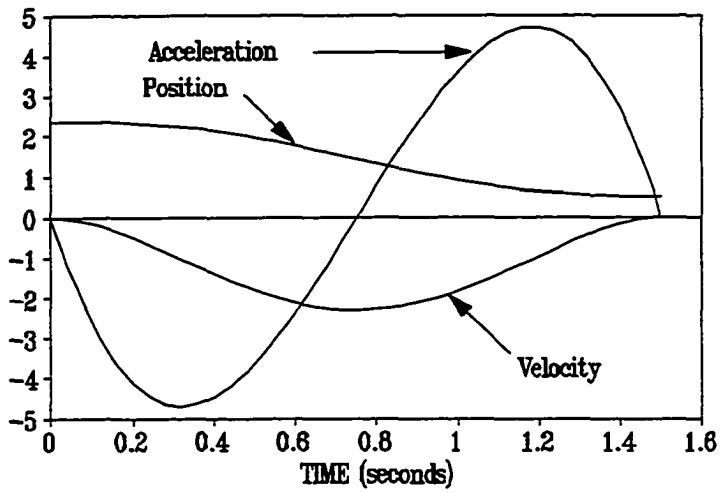

Figure 7c: Nominal trajectory for joints 3 and 6 (rad,rad/sec,rad/sec^2)

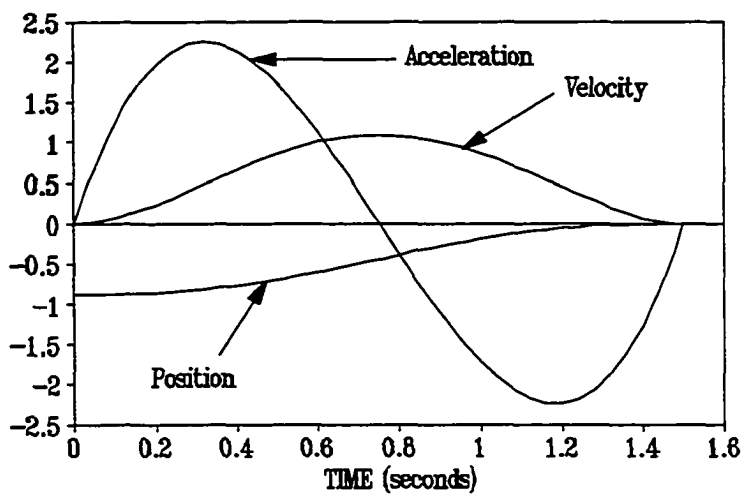

Figure 7d: Nominal trajectory for joint 5 (rad,rad/sec,rad/sec^2)

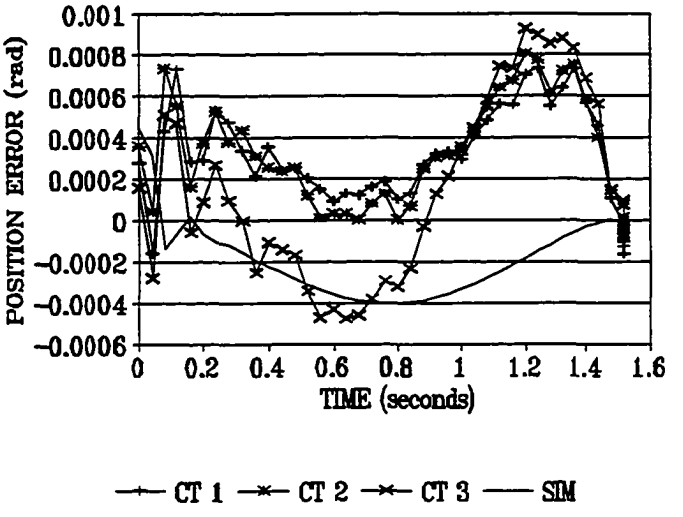

Figure 8a: Position error profiles for joint 1 (range of motion: 1.57 rad)

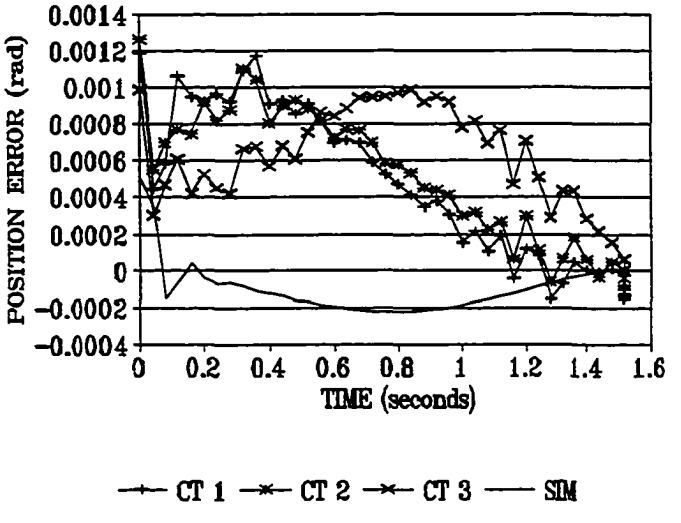

Figure 8b: Position error profiles for joint 2 (range of motion: 0.873 rad)

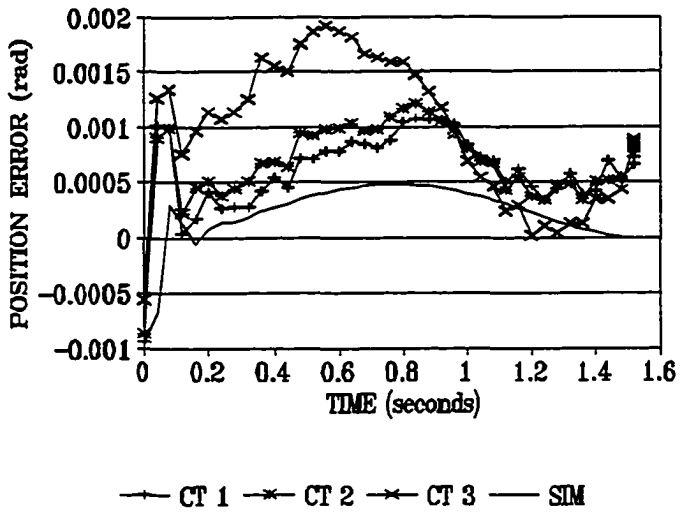

Figure 8c: Position error profiles for joint 3 (range of motion: 1.83 rad)

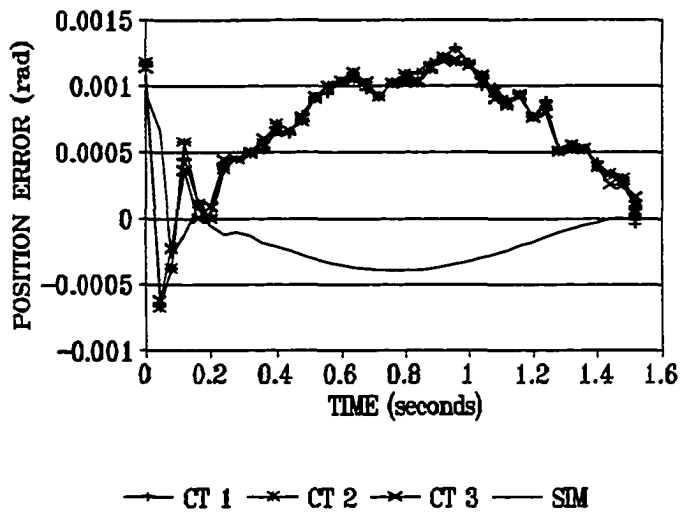

Figure 8d: Position error profiles for joint 4 (range of motion: 1.57 rad)

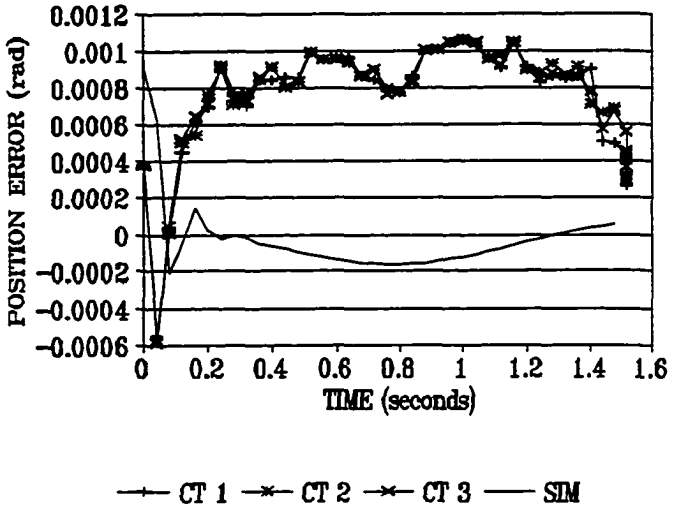

Figure 8e: Position error profiles for joint 5 (range of motion: 0.873 rad)

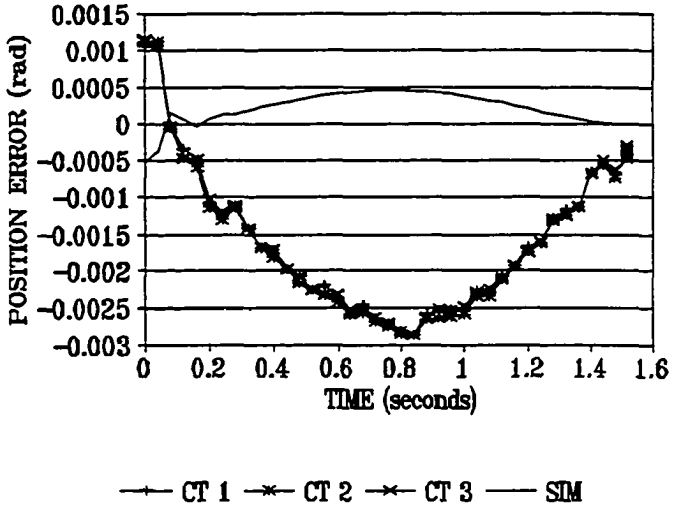

Figure 8f: Position error profiles for joint 6 (range of motion: 1.83 rad)

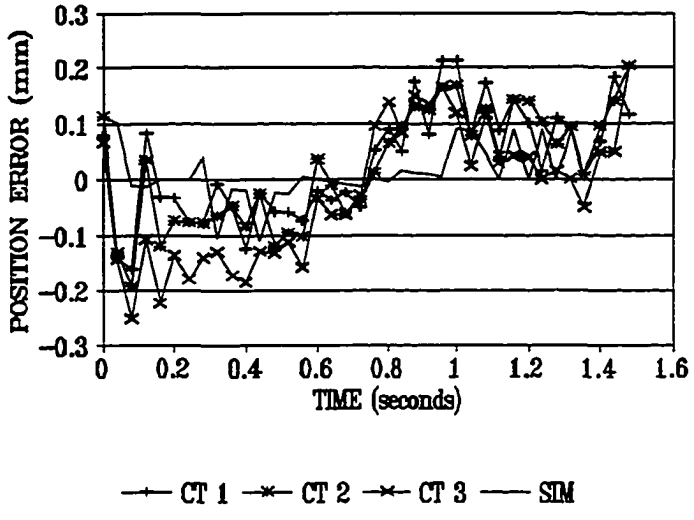

Figure 9: Cartesian position error profiles (range of motion: 0.52 meters)

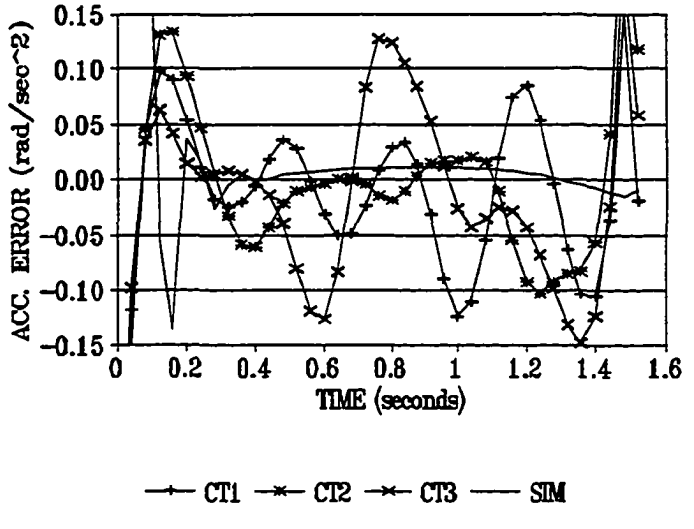

Figure 10a: Acceleration error profiles for joint 1 (peak acceleration: 4.03 rad/sec^2)

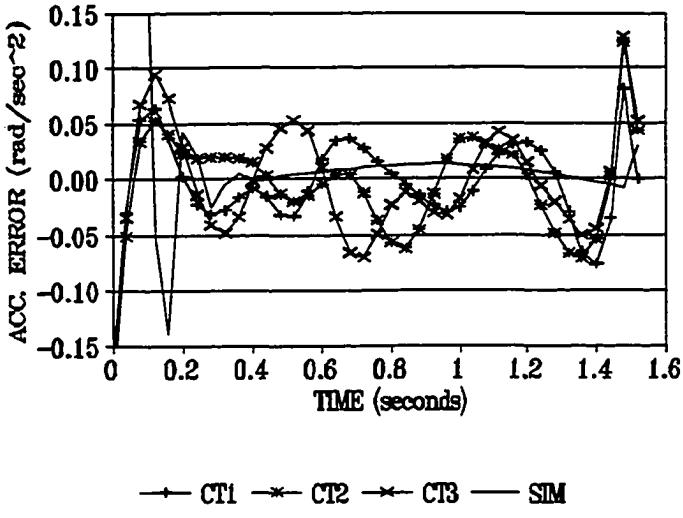

Figure 10b: Acceleration error profiles for joint 2 (peak acceleration: 2.24 rad/sec^2)

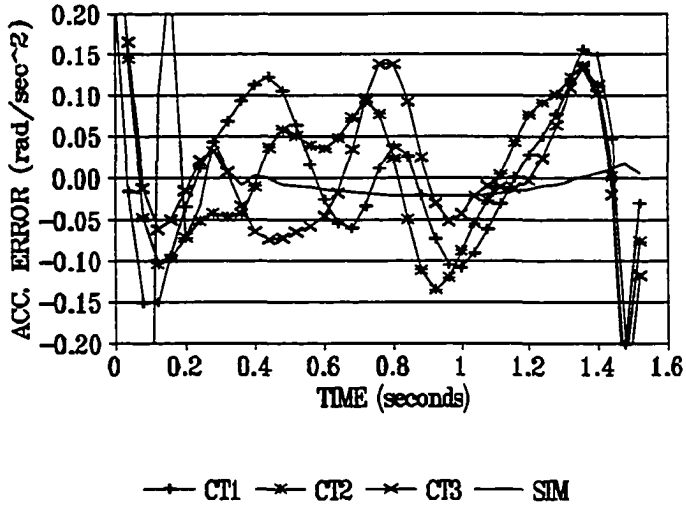

Figure 10c: Acceleration error profiles for joint 3 (peak acceleration: 4.70 rad/sec^2)

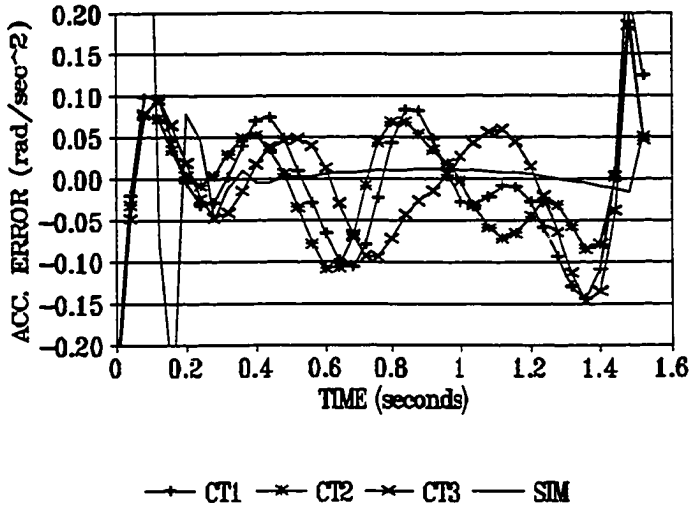

Figure 10d: Acceleration error profiles for joint 4 (peak acceleration: 4.03 rad/sec^2)

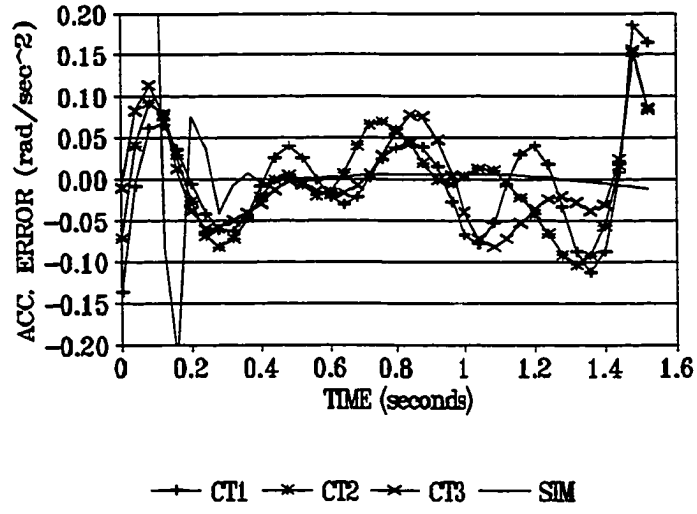

Figure 10e: Acceleration error profiles for joint 5 (peak acceleration: 2.24 rad/sec^2)

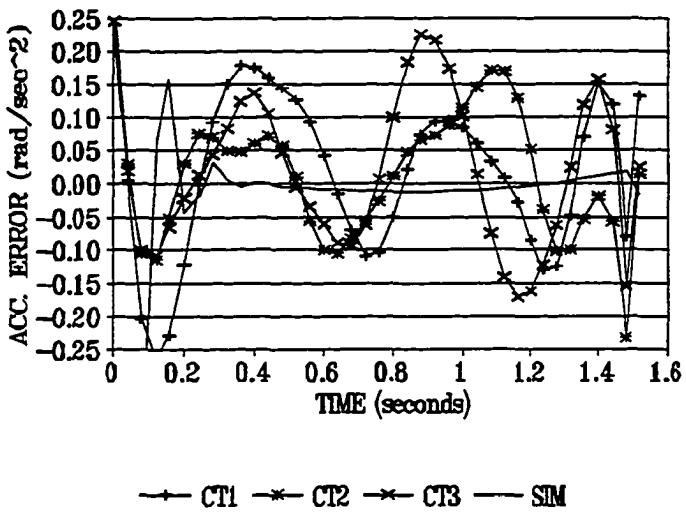

Figure 10f: Acceleration error profiles for joint 6 (peak acceleration: 4.70 rad/sec^2)

Table I: Linear system matrices for robustness analysis

Trajectory: joint positions=(2.44,4.90,1.06,1.60,5.80,4.49) rad,
joint velocities (-10,-10,-10,10,10,-10) rad/sec, and
joint accelerations (-100,-100,-100,-100,100,100) rad/sec^2

M matrix

2.7405	-0.7987	-0.1257	0.0014	-0.0003	0.0000
-0.7987	5.0443	0.6560	0.0012	-0.0003	-0.0000
-0.1257	0.6560	1.1658	0.0007	-0.0000	-0.0000
0.0014	0.0012	0.0007	0.2017	0.0000	0.0000
-0.0003	-0.0003	-0.0000	0.0000	0.1796	0.0000
0.0000	-0.0000	-0.0000	0.0000	0.0000	0.1930

C matrix

1.2269	-4.1059	2.4947	-0.0033	-0.0080	0.0000
2.8269	4.7166	-7.6548	-0.0544	-0.0363	0.0005
-0.6873	3.8541	3.0327	-0.0240	-0.0300	0.0005
-0.0147	0.0043	-0.0009	0.4099	-0.0107	0.0003
-0.0687	-0.0185	-0.0064	0.0107	0.4270	0.0007
0.0000	0.0003	0.0003	0.0001	-0.0007	0.2160

K matrix

0.0000	692.0988	236.1637	0.4090	-0.2016	0.0000
0.0000	-299.9462	-216.8453	-0.4448	1.3559	0.0000
0.0000	-63.0536	-15.8041	-0.2746	0.4047	0.0000
0.0000	-0.4557	-0.3778	0.3183	0.3941	0.0000
0.0000	-0.0327	0.0167	0.3811	0.8506	0.0000
0.0000	-0.0068	-0.0068	-0.0065	-0.0118	0.0000

Table II: Comparison of the closed loop eigenvalues for the pole placement method in this paper and traditional computed torque

Pole Placement	Computed Torque
Desired poles at -45±45j:	
-45.00+45.00j	-47.25+45.11j
-45.00-45.00j	-47.25-45.11j
-45.00+45.00j	-44.68+43.03j
-45.00-45.00j	-44.68-43.03j
-45.00+45.00j	-46.19+43.83j
-45.00-45.00j	-46.19-43.83j
-45.00+45.00j	-46.02+43.97j
-45.00-45.00j	-46.02-43.97j
-45.00+45.00j	-45.42+44.47j
-45.00-45.00j	-45.42-44.47j
-45.00+45.00j	-45.56+44.43j
-45.00-45.00j	-45.56-44.43j
Desired poles at -15±15j:	
-15.00+15.00j	-17.70+15.48j
-15.00-15.00j	-17.70-15.48j
-15.00+15.00j	-14.60+12.98j
-15.00-15.00j	-14.60-12.98j
-15.00+15.00j	-15.05+14.10j
-15.00-15.00j	-15.05-14.10j
-15.00+15.00j	-16.16+13.86j
-15.00-15.00j	-16.16-13.86j
-15.00+15.00j	-16.05+13.97j
-15.00-15.00j	-16.05-13.97j
-15.00+15.00j	-15.56+14.42j
-15.00-15.00j	-15.56-14.42j
Desired poles at -4.5±4.5j:	
-4.50+4.50j	-7.70+6.63j
-4.50-4.50j	-7.70-6.63j
-4.50+4.50j	-8.80+0.00j
-4.50-4.50j	-1.00-0.00j
-4.50+4.50j	-3.25+4.22j
-4.50-4.50j	-3.25-4.22j
-4.50+4.50j	-5.63+3.75j
-4.50-4.50j	-5.63-3.75j
-4.50+4.50j	-5.58+3.19j
-4.50-4.50j	-5.58-3.19j
-4.50+4.50j	-5.06+3.86j
-4.50-4.50j	-5.06-3.86j

APPENDIX

(Note: joint variables denote the nominal trajectory point and matrix elements not explicitly given are set equal to zero)

λ_{ri} is the real part of a desired pole, λ_{ci} is the complex part of a desired pole

Continuous time gain matrix - Γ_c

$$\Gamma_{11} = [3.64 + .8\cos(2\theta_2) - .01\sin(2\theta_3 + 2\theta_2) - .01\cos(\theta_3 + 2\theta_2) + .37\sin(\theta_3 + 2\theta_2) - .15\cos(2\theta_3 + 2\theta_2) + .37\sin(\theta_3) - .01\cos(\theta_3)][\lambda_{r1}^2 + \lambda_{c1}^2]$$

$$\Gamma_{12} = [.69\sin(\theta_2) - .13\cos(\theta_3 + \theta_2) + .02\cos(\theta_2)][\lambda_{r2}^2 + \lambda_{c2}^2] + [1.60\sin(2\theta_2) - .74\cos(\theta_3 + 2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .02\sin(\theta_3 + 2\theta_2) + .01\cos(2\theta_2)]\ddot{\theta}_1 + [-.69\cos(\theta_2) - .13\sin(\theta_3 + \theta_2) + .02\sin(\theta_2)]\ddot{\theta}_2 - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_3 + [3.20\cos(2\theta_2) + 1.5\sin(\theta_3 + 2\theta_2) - .6\cos(2\theta_3 + 2\theta_2) - .04\cos(\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .03\sin(2\theta_2)]\dot{\theta}_1\dot{\theta}_2 + [.74\sin(\theta_3 + 2\theta_2) - .60\cos(2\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .02\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_3 - .27\cos(\theta_3 + \theta_2)\dot{\theta}_2\dot{\theta}_3 + [.69\sin(\theta_2) - .13\cos(\theta_3 + \theta_2) + .02\cos(\theta_2)]\dot{\theta}_2^2 - .13\cos(\theta_3 + \theta_2)\dot{\theta}_3^2$$

$$\Gamma_{13} = [-.37\cos(\theta_3) - .37\cos(\theta_3 + 2\theta_2) - .3\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .01\sin(\theta_3 + 2\theta_2) - .01\sin(\theta_3)]\ddot{\theta}_1 - .13\sin(\theta_3 + \theta_2)(\ddot{\theta}_2 + \ddot{\theta}_3) + [.74\sin(\theta_3 + 2\theta_2) - .6\cos(2\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .02\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_2 + [-.6\cos(2\theta_3 + 2\theta_2) + .37\sin(\theta_3) + .37\sin(\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .01\cos(\theta_3) - .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_3 - .27\cos(\theta_3 + \theta_2)\dot{\theta}_2\dot{\theta}_3 - .13\cos(\theta_3 + \theta_2)(\dot{\theta}_2^2 + \dot{\theta}_3^2) - [.13\cos(\theta_3 + \theta_2)][\lambda_{r3}^2 + \lambda_{c3}^2]$$

$$\Gamma_{17} = [1.60\sin(2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) - .74\cos(\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .02\sin(\theta_3 + 2\theta_2) + .01\cos(2\theta_2)]\dot{\theta}_2 + [-.37\cos(\theta_3) - .37\cos(\theta_3 + 2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .01\sin(\theta_3) - .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_3 + 2[-3.64 - .8\cos(2\theta_2) - .37\sin(\theta_3 + 2\theta_2) - .37\sin(\theta_3) + .15\cos(2\theta_3 + 2\theta_2) + .01\sin(2\theta_3 + 2\theta_2) + .01\cos(\theta_3 + 2\theta_2) + .01\cos(\theta_3)]\lambda_{r1}$$

$$\Gamma_{18} = [1.60\sin(2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) - .74\cos(\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .02\sin(\theta_3 + 2\theta_2) + .01\cos(2\theta_2)]\dot{\theta}_1 + [-1.38\cos(\theta_2) - .27\sin(\theta_3 + \theta_2) + .05\sin(\theta_2)]\dot{\theta}_2 - .27\sin(\theta_3 + \theta_2)\dot{\theta}_3 + 2[-.69\sin(\theta_2) + .13\cos(\theta_3 + \theta_2) - .02\cos(\theta_2)]\lambda_{r2}$$

$$\Gamma_{19} = [-.37\cos(\theta_3) - .37\cos(\theta_3 + 2\theta_2) - .3\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .01\sin(\theta_3) - .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_1 - .27\sin(\theta_3 + \theta_2)\dot{\theta}_2 - .27\sin(\theta_3 + \theta_2)\dot{\theta}_3 + .27\cos(\theta_3 + \theta_2)\lambda_{r3}$$

$$\Gamma_{21} = [.69\sin(\theta_2) - .13\cos(\theta_3 + \theta_2) + .02\cos(\theta_2)][\lambda_{r1}^2 + \lambda_{c1}^2]$$

$$\Gamma_{22} = -37.23\sin(\theta_2) + 8.45\cos(\theta_3 + \theta_2) - 1.02\cos(\theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) + [-.69\cos(\theta_2) - .13\sin(\theta_3 + \theta_2) + .02\sin(\theta_2)]\ddot{\theta}_1 + [-1.6\cos(2\theta_2) - .74\sin(\theta_3 + 2\theta_2) + .3\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2) + .02\cos(\theta_3 + 2\theta_2) + .01\sin(2\theta_2)]\dot{\theta}_1^2 + [4.4 - .02\cos(\theta_3) + .74\sin(\theta_3)][\lambda_{r2}^2 + \lambda_{c2}^2]$$

$$\Gamma_{23} = 8.54\cos(\theta_3 + \theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [-.74\cos(\theta_3) - .02\sin(\theta_3)]\ddot{\theta}_2 + [-.37\cos(\theta_3) - .01\sin(\theta_3)]\ddot{\theta}_3 + [.74\sin(\theta_3) - .02\cos(\theta_3)]\dot{\theta}_2\dot{\theta}_3 + [.37\sin(\theta_3) - .01\cos(\theta_3)]\dot{\theta}_3^2 + [-.37\sin(\theta_3 + 2\theta_2) + .30\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2) + .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1^2 + [.33 - .01\cos(\theta_3) + .3721\sin(\theta_3)][\lambda_{r3}^2 + \lambda_{c3}^2]$$

$$\Gamma_{25} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{27} = [-1.60\sin(2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) + .74\cos(\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .02\sin(\theta_3 + 2\theta_2) - .01\cos(2\theta_2)]\dot{\theta}_1 + 2[-.69\sin(\theta_2) + .13\cos(\theta_3 + \theta_2) - .02\cos(\theta_2)]\lambda_{r1}$$

$$\Gamma_{28} = [-.74\cos(\theta_3) - .02\sin(\theta_3)]\dot{\theta}_3 + 2[-4.4 - .74\sin(\theta_3) + .02\cos(\theta_3)]\lambda_{r2}$$

$$\Gamma_{29} = [-.74\cos(\theta_3) - .02\sin(\theta_3)]\dot{\theta}_3 + [-.74\cos(\theta_3) - .02\sin(\theta_3)]\dot{\theta}_2 + 2[-.33 - .37\sin(\theta_3) + .01\cos(\theta_3)]\lambda_{r3}$$

$$\Gamma_{31} = [-.13\cos(\theta_3 + \theta_2)][\lambda_{r1}^2 + \lambda_{c1}^2]$$

$$\Gamma_{32} = 8.54\cos(\theta_3 + \theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [-.37\sin(\theta_3 + 2\theta_2) + .30\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2) + .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1^2 + [.33 + .37\sin(\theta_3) - .01\cos(\theta_3)][\lambda_{r2}^2 + \lambda_{c2}^2]$$

$$\Gamma_{33} = 8.54\cos(\theta_3 + \theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [-.37\cos(\theta_3) - .01\sin(\theta_3)]\ddot{\theta}_2 + [-.37\sin(\theta_3) + .01\cos(\theta_3)]\dot{\theta}_2^2 + [-.19\sin(\theta_3 + 2\theta_2) - .19\sin(\theta_3) + .30\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2)]\dot{\theta}_1^2 + 1.16[\lambda_{r3}^2 + \lambda_{c3}^2]$$

$$\Gamma_{35} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{37} = [.37\cos(\theta_3) + .37\cos(\theta_3 + 2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .01\sin(\theta_3) + .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_1 + .27\cos(\theta_3 + \theta_2)\lambda_{r1}$$

$$\Gamma_{38} = [.74\cos(\theta_3) + .02\sin(\theta_3)]\dot{\theta}_2 + 2[-.33 + .01\cos(\theta_3) - .37\sin(\theta_3)]\lambda_{r3}$$

$$\Gamma_{39} = -2.33\lambda_{r3}$$

$$\Gamma_{44} = .20[\lambda_{r4}^2 + \lambda_{c4}^2]$$

$$\Gamma_{410} = -.40\lambda_{r4}$$

$$\Gamma_{52} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{53} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{55} = .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) + .18[\lambda_{r5}^2 + \lambda_{c5}^2]$$

$$\Gamma_{511} = -.36\lambda_{r5}$$

$$\Gamma_{66} = .19[\lambda_{r6}^2 + \lambda_{c6}^2]$$

$$\Gamma_{612} = -.39\lambda_{r6}$$

Discrete time gain matrix - Γ_d

$$\Gamma_{11} = [3.64 + .8\cos(2\theta_2) - .01\sin(2\theta_3 + 2\theta_2) - .01\cos(\theta_3 + 2\theta_2) + .37\sin(\theta_3 + 2\theta_2) - .15\cos(2\theta_3 + 2\theta_2) + .37\sin(\theta_3) - .01\cos(\theta_3)]\frac{[\lambda_{r1}^2 + \lambda_{c1}^2 - 2\lambda_{r1} + 1]}{h^2}$$

$$\Gamma_{12} = [.69\sin(\theta_2) - .13\cos(\theta_3 + \theta_2) + .02\cos(\theta_2)]\frac{[\lambda_{r2}^2 + \lambda_{c2}^2 - 2\lambda_{r2} + 1]}{h^2} + [1.60\sin(2\theta_2) - .74\cos(\theta_3 + 2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .02\sin(\theta_3 + 2\theta_2) + .01\cos(2\theta_2)]\ddot{\theta}_1 + [-.69\cos(\theta_2) - .13\sin(\theta_3 + \theta_2) + .02\sin(\theta_2)]\ddot{\theta}_2 - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_3 + [3.20\cos(2\theta_2) + 1.5\sin(\theta_3 + 2\theta_2) - .6\cos(2\theta_3 + 2\theta_2) - .04\cos(\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .03\sin(2\theta_2)]\dot{\theta}_1\dot{\theta}_2 + [.74\sin(\theta_3 + 2\theta_2) - .60\cos(2\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .02\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_3 - .27\cos(\theta_3 + \theta_2)\dot{\theta}_2\dot{\theta}_3 + [.69\sin(\theta_2) - .13\cos(\theta_3 + \theta_2) + .02\cos(\theta_2)]\dot{\theta}_2^2 - .13\cos(\theta_3 + \theta_2)\dot{\theta}_3^2$$

$$\Gamma_{13} = [-.37\cos(\theta_3) - .37\cos(\theta_3 + 2\theta_2) - .3\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .01\sin(\theta_3 + 2\theta_2) - .01\sin(\theta_3)]\ddot{\theta}_1 - .13\sin(\theta_3 + \theta_2)(\ddot{\theta}_2 + \ddot{\theta}_3) + [.74\sin(\theta_3 + 2\theta_2) - .6\cos(2\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .02\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_2 + [-.6\cos(2\theta_3 + 2\theta_2) + .37\sin(\theta_3) + .37\sin(\theta_3 + 2\theta_2) - .04\sin(2\theta_3 + 2\theta_2) - .01\cos(\theta_3) - .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1\dot{\theta}_3 - .27\cos(\theta_3 + \theta_2)\dot{\theta}_2\dot{\theta}_3 - .13\cos(\theta_3 + \theta_2)(\dot{\theta}_2^2 + \dot{\theta}_3^2) - [.13\cos(\theta_3 + \theta_2)]\frac{[\lambda_{r3}^2 + \lambda_{c3}^2 - 2\lambda_{r3} + 1]}{h^2}$$

$$\Gamma_{17} = [1.60\sin(2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) - .74\cos(\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .02\sin(\theta_3 + 2\theta_2) + .01\cos(2\theta_2)]\dot{\theta}_2 + [-.37\cos(\theta_3) - .37\cos(\theta_3 + 2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .01\sin(\theta_3) - .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_3 + 2[-3.64 - .8\cos(2\theta_2) - .37\sin(\theta_3 + 2\theta_2) - .37\sin(\theta_3) + .15\cos(2\theta_3 + 2\theta_2) + .01\sin(2\theta_3 + 2\theta_2) + .01\cos(\theta_3 + 2\theta_2) + .01\cos(\theta_3)]\frac{[\lambda_{r1} - 1]}{h}$$

$$\Gamma_{18} = [1.60\sin(2\theta_2) - .30\sin(2\theta_3 + 2\theta_2) - .74\cos(\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .02\sin(\theta_3 + 2\theta_2) + .01\cos(2\theta_2)]\dot{\theta}_1 + [-1.38\cos(\theta_2) - .27\sin(\theta_3 + \theta_2) + .05\sin(\theta_2)]\dot{\theta}_2 - .27\sin(\theta_3 + \theta_2)\dot{\theta}_3 + 2[-.69\sin(\theta_2) + .13\cos(\theta_3 + \theta_2) - .02\cos(\theta_2)]\frac{[\lambda_{r2} - 1]}{h}$$

$$\Gamma_{19} = [-.37\cos(\theta_3) - .37\cos(\theta_3 + 2\theta_2) - .3\sin(2\theta_3 + 2\theta_2) + .02\cos(2\theta_3 + 2\theta_2) - .01\sin(\theta_3) - .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_1 - .27\sin(\theta_3 + \theta_2)\dot{\theta}_2 - .27\sin(\theta_3 + \theta_2)\dot{\theta}_3 + [.27\cos(\theta_3 + \theta_2)]\frac{[\lambda_{r3} - 1]}{h}$$

$$\Gamma_{21} = [.69\sin(\theta_2) - .13\cos(\theta_3 + \theta_2) + .02\cos(\theta_2)]\frac{[\lambda_{r1}^2 + \lambda_{c1}^2 - 2\lambda_{r1} + 1]}{h^2}$$

$$\Gamma_{22} = -37.23\sin(\theta_2) + 8.45\cos(\theta_3 + \theta_2) - 1.02\cos(\theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) + [-.69\cos(\theta_2) - .13\sin(\theta_3 + \theta_2) + .02\sin(\theta_2)]\ddot{\theta}_1 + [-1.6\cos(2\theta_2) - .74\sin(\theta_3 + 2\theta_2) + .3\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2) + .02\cos(\theta_3 + 2\theta_2) + .01\sin(2\theta_2)]\dot{\theta}_1^2 + [4.4 - .02\cos(\theta_3) + .74\sin(\theta_3)]\frac{[\lambda_{r2}^2 + \lambda_{c2}^2 - 2\lambda_{r2} + 1]}{h^2}$$

$$\Gamma_{23} = 8.54\cos(\theta_3 + \theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [-.74\cos(\theta_3) - .02\sin(\theta_3)]\ddot{\theta}_2 + [-.37\cos(\theta_3) - .01\sin(\theta_3)]\ddot{\theta}_3 + [.74\sin(\theta_3) - .02\cos(\theta_3)]\dot{\theta}_2\dot{\theta}_3 + [.37\sin(\theta_3) - .01\cos(\theta_3)]\dot{\theta}_3^2 + [-.37\sin(\theta_3 + 2\theta_2) + .30\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2) + .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1^2 + [.33 - .01\cos(\theta_3) + .3721\sin(\theta_3)]\frac{[\lambda_{r3}^2 + \lambda_{c3}^2 - 2\lambda_{r3} + 1]}{h^2}$$

$$\Gamma_{25} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{27} = [-1.60\sin(2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) + .74\cos(\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .02\sin(\theta_3 + 2\theta_2) - .01\cos(2\theta_2)]\dot{\theta}_1 + 2[-.69\sin(\theta_2) + .13\cos(\theta_3 + \theta_2) - .02\cos(\theta_2)]\frac{[\lambda_{r1} - 1]}{h^2}$$

$$\Gamma_{28} = [-.74\cos(\theta_3) - .02\sin(\theta_3)]\dot{\theta}_3 + 2[-4.4 - .74\sin(\theta_3) + .02\cos(\theta_3)]\frac{[\lambda_{r2} - 1]}{h}$$

$$\Gamma_{29} = [-.74\cos(\theta_3) - .02\sin(\theta_3)]\dot{\theta}_3 + [-.74\cos(\theta_3) - .02\sin(\theta_3)]\dot{\theta}_2 + 2[-.33 - .37\sin(\theta_3) + .01\cos(\theta_3)]\frac{[\lambda_{r3} - 1]}{h}$$

$$\Gamma_{31} = [-.13\cos(\theta_3 + \theta_2)]\frac{[\lambda_{r1}^2 + \lambda_{c1}^2 - 2\lambda_{r1} + 1]}{h^2}$$

$$\Gamma_{32} = 8.54\cos(\theta_3 + \theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [-.37\sin(\theta_3 + 2\theta_2) + .30\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2) + .01\cos(\theta_3 + 2\theta_2)]\dot{\theta}_1^2 + [.33 + .37\sin(\theta_3) - .01\cos(\theta_3)]\frac{[\lambda_{r2}^2 + \lambda_{c2}^2 - 2\lambda_{r2} + 1]}{h^2}$$

$$\Gamma_{33} = 8.54\cos(\theta_3 + \theta_2) + .25\sin(\theta_3 + \theta_2) + .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) - .13\sin(\theta_3 + \theta_2)\ddot{\theta}_1 + [-.37\cos(\theta_3) - .01\sin(\theta_3)]\ddot{\theta}_2 + [-.37\sin(\theta_3) + .01\cos(\theta_3)]\dot{\theta}_2^2 + [-.19\sin(\theta_3 + 2\theta_2) - .19\sin(\theta_3) + .30\cos(2\theta_3 + 2\theta_2) + .02\sin(2\theta_3 + 2\theta_2)]\dot{\theta}_1^2 + 1.16\frac{[\lambda_{r3}^2 + \lambda_{c3}^2 - 2\lambda_{r3} + 1]}{h^2}$$

$$\Gamma_{35} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{37} = [.37\cos(\theta_3) + .37\cos(\theta_3 + 2\theta_2) + .30\sin(2\theta_3 + 2\theta_2) - .02\cos(2\theta_3 + 2\theta_2) + .01\sin(\theta_3) + .01\sin(\theta_3 + 2\theta_2)]\dot{\theta}_1 + \frac{[.27\cos(\theta_3 + \theta_2)][\lambda_{r1} - 1]}{h}$$

$$\Gamma_{38} = [.74\cos(\theta_3) + .02\sin(\theta_3)]\dot{\theta}_2 + 2[-.33 + .01\cos(\theta_3) - .37\sin(\theta_3)]\frac{[\lambda_{r3} - 1]}{h}$$

$$\Gamma_{39} = \frac{-2.33[\lambda_{r3} - 1]}{h}$$

$$\Gamma_{44} = \frac{.20[\lambda_{r4}^2 + \lambda_{c4}^2 - 2\lambda_{r4} + 1]}{h^2}$$

$$\Gamma_{410} = \frac{-.40[\lambda_{r4} - 1]}{h}$$

$$\Gamma_{52} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{53} = -.01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2)$$

$$\Gamma_{55} = .01\cos(-\theta_5 + \theta_3 + \theta_2) + .01\cos(\theta_5 + \theta_3 + \theta_2) + \frac{.18[\lambda_{r5}^2 + \lambda_{c5}^2 - 2\lambda_{r5} + 1]}{h^2}$$

$$\Gamma_{511} = \frac{-.36[\lambda_{r5} - 1]}{h}$$

$$\Gamma_{66} = \frac{.19[\lambda_{r6}^2 + \lambda_{c6}^2 - 2\lambda_{r6} + 1]}{h^2}$$

$$\Gamma_{612} = \frac{-.39[\lambda_{r6} - 1]}{h}$$

Friction was added to both the continuous and discrete gain matrices as follows:

$$\Gamma_{17} = \begin{cases} \Gamma_{17} - 4.94, & \dot{\theta}_1 \geq 0 \\ \Gamma_{17} - 3.45, & \dot{\theta}_1 < 0 \end{cases}$$

$$\Gamma_{28} = \begin{cases} \Gamma_{28} - 7.67, & \dot{\theta}_2 \geq 0 \\ \Gamma_{28} - 8.53, & \dot{\theta}_2 < 0 \end{cases}$$

$$\Gamma_{39} = \begin{cases} \Gamma_{39} - 3.27, & \dot{\theta}_3 \geq 0 \\ \Gamma_{39} - 3.02, & \dot{\theta}_3 < 0 \end{cases}$$

$$\Gamma_{410} = \Gamma_{410} - .41$$

$$\Gamma_{511} = \Gamma_{511} - .43$$

$$\Gamma_{612} = \Gamma_{612} - .22$$

IV. A CONTROL SYSTEM ARCHITECTURE FOR ROBOTS USED TO SIMULATE DYNAMIC FORCE AND MOMENT INTERACTION BETWEEN HUMANS AND VIRTUAL OBJECTS

A paper to be submitted to The IEEE Transactions on Systems, Man, and Cybernetics

C.L. Clover
Iowa State University

Abstract

Haptic or kinesthetic feedback is essential in many important virtual reality and telepresence applications. Previous research focusses on simulating static forces such as those encountered when interacting with a stiff object such as a wall. Also, past studies usually employ custom-made devices which are not readily available to other researchers. Consequently, many of the results found in the haptic feedback literature cannot be independently replicated.

This paper demonstrates that "off the shelf" general purpose robotics equipment can be incorporated into an effective haptic/kinesthetic feedback system. This system can accommodate a wide variety of virtual reality applications including training and telerobotics. The paper shows that the mechanical deficiencies (e.g. friction, inertia, and backlash) often associated with general purpose manipulators can be overcome with a suitable control system architecture.

An admittance control scheme is utilized which enables the simulation of dynamic force and moment interaction as well as contact with stiff objects. Test data for a PUMA 560 validates the techniques presented in this paper.

I. INTRODUCTION

A. Overview

This paper seeks to advance the state of the art in human/robot interface technology by demonstrating techniques which augment visual systems with the sense of touch or "feel". (The term "robot" is used loosely here to refer to any mechanical device which allows work interaction with a human operator via force and/or motion inputs.) Common phrases used in this area are "tactile force feedback", dealing with the sense of the *type* of contact; "kinesthetic/haptic force feedback", dealing with the sense of motion of the hand/arm system and the forces imparted to it; or generically as "force reflective feedback". The literature indicates that, for the most part, researchers have only recently (within the last fifteen years or so) begun to examine this branch of robotics in depth. Much of the work in this area has studied the use of human/robot interfaces in telepresence or telerobotics applications. Some of the recent force feedback/telerobotics applications include "synthetic fixturing" [1],[2] and "teleprogramming" in the presence of communications delays [3],[4]. Others have investigated the use of custom designed mechanical devices as kinesthetic/haptic interfaces in virtual reality (VR) and scientific visualization applications ranging from astronaut training to molecular docking [5],[6].

Previous work in human/robot kinesthetic interface technology usually assumes point contacts (therefore eliminating moments/torques) with objects that are neither accelerating nor decelerating, in order to impart *static* forces to a human operator. Furthermore, previous applications often utilize generic, non model-based forces. Only the direction of the force is used and its magnitude is neglected. This is appropriate for applications which do not benefit from more realistic force generation. This paper attempts to lay the groundwork for

a wide variety of applications where there is a need for accurate dynamic force simulation and generation.

NASA, a primary contributor and consumer of VR research, has concluded that dynamic force reflective feedback in VR systems offers an effective low-cost alternative in training astronauts for space-related tasks [5]. Current training procedures include the use of full-scale underwater mock-ups which are very costly to operate and maintain. Because of the dramatic decrease in hardware costs associated with virtual reality technology, computer generated training environments, which make use of force reflection technology, can offer effective training scenarios at a greatly reduced cost. The recent Hubble Space Telescope repair mission offers a vivid example. Astronauts were attached to the space shuttle's robotic arm and employed as human end effectors. They removed old components and inserted new ones into the telescope. Maneuvering these components was often difficult, especially when the objects were translating and rotating relative to one another. Although weightless in space, many payloads have substantial inertia. The dynamics of handling objects (large or small) in a weightless environment is a new experience to most astronauts and requires considerable training. Furthermore, underwater mock-ups of such scenarios often do not recreate these dynamics accurately. Accurate dynamic force and moment simulation achieved using a suitable haptic interface device has the potential for providing realistic and cost effective ground-based training.

There are a host of other applications besides astronaut training which could be enhanced with more realistic force reflection. Therefore, this paper seeks to provide an in depth analysis of a particular haptic interface system which we have developed to provide realistic dynamic force and moment interaction for manipulating generic objects in virtual

environments. The next sections of the paper survey the relevant literature and provide further motivations for this research. Subsequent sections provide details of the control system that is used, test results, and conclusions.

B. Literature Survey

Design and analysis of haptic/kinesthetic feedback mechanisms is a relatively new field of study, [6]. In [7] we have what is generally recognized as the first attempt to incorporate force feedback and robotics. An early effort at combining scientific visualization with force feedback is described in [8]. A general control system overview as well as forward and inverse kinematics of an early force reflective telerobotics system is given in [9]. However, these early systems suffered from a lack of computer power capable of driving the graphics and control systems at levels required for good visual and force bandwidths.

It is clear that force feedback adds a vital dimension to a wide range of scientific visualization and telepresence applications [6],[10],[11]. Nevertheless, it is also clear that research in this area is lagging behind the technologies associated with current graphics and computational capabilities [12].

Thus far, a number of researchers have reported on the basic characteristics required in designing a haptic/kinesthetic feedback device. See [13] for a good overview of these characteristics. The consensus in the literature seems to be that the "ideal" interface device is one which has low friction, inertia and backlash, is highly backdriveable, has a large force range and bandwidth, and has a suitable working volume. These can be conflicting goals. For example, larger working volumes and larger force capabilities require devices which are physically larger and which tend to have more inertia and friction coupled with a lower

bandwidth. Therefore, any decision to use a given kinesthetic feedback device depends on the intended application. For example, if the goal is to simulate gross dynamic forces and motion when manipulating a substantial mass, then force range and working volume are key considerations and force bandwidth becomes a secondary consideration. On the other hand, force bandwidth becomes a dominant criteria if fine motion of small masses coupled with a desire to "feel" high frequency vibrations is needed. It is interesting to note that the recent trend in haptic/kinesthetic interface design seems to be moving toward the later, with light duty devices that have low friction and inertia and a large force bandwidth. The next few paragraphs describe some recently developed high bandwidth - low force output devices.

A four degree of freedom (DOF) "manipulandum" which emulates a hand tool that can be used to explore computer generated virtual objects is presented in [14]. The authors note the importance of position sensor resolution on system stability. Since the manipulandum is designed for high bandwidth tasks, its working volume is restricted and static loads are limited to 45 N and 135 N-cm. A nine DOF grounded exoskeleton device which applies reaction forces to the fingers and palm has been developed by [15]. The working volume is a sphere with an approximate 30 cm diameter and maximum payload is 2.3 Kg. The SAFiRE (sensing and force reflecting exoskeleton) is described in [16]. This is an arm grounded device providing a large working volume. It applies forces to the thumb, forefinger, and wrist. Since the SAFiRE's design is proprietary, very little performance information is currently available.

In contrast to the SAFiRE, the PHANToM (personal haptic interface mechanism) device [17] is grounded and thus has a smaller working volume. However, the PHANToM working volume is suitable for the types of tasks it was designed for (e.g., those requiring

fine, finger tip motions/forces). Three open-loop controlled finger forces are applied with 10N peak and 1.5N continuous upper limits. SPIDAR (space interface device for artificial reality) [18] uses remote motors attached to the thumb and forefingers via "tensed strings". As a result, it has a somewhat larger working volume but forces are limited to 4 N. A force feedback system which simulates forces from manipulating pen-like instruments such as a surgical scalpel is presented in [19]. Static and dynamic modeling is presented.

Some of the haptic interface devices being developed are magnetically actuated rather than mechanically actuated. Preliminary results for a device which imparts forces at a finger tip via a magnetic coil attached to the finger are presented in [20]. The advantage of this system over mechanically-based systems is that there are no rigid attachment points, thus obviating the challenges of compensating for inertia and friction. An analysis of simulating stiff walls and static friction with a magnetically levitated device described in [21] is presented in [22]. Again the problems of friction, inertia, and backdriveability are eliminated since there are no physical contacts. The authors noted that adding a "brake pulse" can be an effective way to increase perceived stiffness of a virtual wall.

We now consider recent research aimed at developing systems capable of simulating forces involving whole arm motion rather than finger tip and hand motion. In addition to the SAFiRE, the EAM (Exoskeletal Arm Master), which is a ground-based exoskeleton that applies forces directly at various points along the arm, has also been developed [16]. Again, the EAM design is proprietary and little performance information is currently available. SPICE [23], [24] is a more traditional six DOF robotic device developed for simulating stiff virtual objects such as walls, free-form surfaces, and push buttons. A three DOF "orthosis" used to apply virtual forces has been developed [25]. Potential

application areas include physical rehabilitation and space crew training. The orthosis enables a user to manipulate a virtual object as seen through a head mounted graphics display. This device is pneumatically actuated and has a high power to weight ratio. Finally, GROPE-III [26] is a well known force feedback system used in molecular docking applications. This is a pioneering device in which scientific visualization and kinesthetic feedback have been combined into a tool capable of augmenting research studies over a wide range of topics. The system has a fairly large range of motion and a peak force output of 36 N.

Note that open-loop force control is used in nearly all of haptic interface systems described in this section. The extent to which open-loop approaches are valid depends on how important the "disturbances" of friction, backlash, and inertia are in the mechanism design. Presumably the best way to determine this would be via force transducer measurements. However, very little force validation data is presented in the literature. Furthermore, without force measurements, simulation of dynamic force interaction is impossible.

In telerobotics applications, force measurement is recognized as a key to system performance and stability [27], [28]. Devices which incorporate force measurement include those developed by [13] and [29]. A three DOF planar haptic interface with a 1200 Hz sampling bandwidth device is presented in [13]. Guidelines for measuring such performance characteristics as minimum controllable force and force bandwidth are given. Stability and performance analyses for a two DOF device are given in [29]. The authors note that human arm dynamics do not affect system performance but play a major role in stability. An analytical overview of a generic dynamic force simulator based on measured finger tip forces is provided in [30]. An alternative view to kinesthetic interaction at the human-

machine interface is that the haptic device should regulate the relationship between the operator's intentional motion and that of the device rather than regulate a force relationship [31].

Finally, a simplified model of piano key action has been developed [32] in order to illustrate an approach which utilizes the actual ordinary differential equations (ODE's) of motion of a dynamic system as part of the control loop to develop the necessary driving point impedance in the force/motion controller. Utilizing generic ODE's in this manner allows a haptic interface system the flexibility needed to simulate a wide range of dynamic motions and forces from the intergalactic level down to the molecular level.

C. Motivations

The haptic interface literature lacks a sufficient number of experimental examples wherein dynamic force and moment interaction has been considered. Indeed, published test results which validate performance characteristics for most of the haptic interface devices described in the literature, are rare.

The emphasis in haptic interface research has been on developing new designs rather than on utilizing existing technology. While these efforts in interface design continue, valuable results, which can be readily duplicated and expanded upon in other laboratories, can be achieved with "off the shelf" equipment. Several advantages that general purpose robots have over custom interface devices such as exoskeletons are listed in [33]. These advantages include less complexity, bulk, weight, and increased safety. The disadvantage of most robotic systems include relatively high friction, inertia, and backlash. Therefore, the key is to develop feedback control systems which can adequately compensate for these mechanical deficiencies at some acceptable level.

II. CONTROL SYSTEM ARCHITECTURE

A. Overview

This section discusses a basic hardware and software architecture that has been developed for controlling robot/human interactions resulting from the simulation of virtual object manipulation. As noted previously, one goal is to determine if off the shelf robotics equipment can be effectively used for this purpose. Furthermore, since technology that is readily available has been used, others will be able to expand and improve upon these results without the need for customized equipment which may be difficult to obtain.

We start by again drawing from the telerobotics force feedback literature. An overall design framework is given in [34]. Such systems are "bilateral" in that information flows in two directions between operator and robot. A two-port model is used to characterize performance and stability based upon a Taylor series linearization about a given operating point. Others have since characterized the stability of haptic interface devices in contact with a compliant environment such as the human hand. In [35], it is argued that a robot should have interaction port behavior of a passive system to be stable while in [36] Nyquist techniques are utilized to establish stability limits with both low- and high-frequency maneuvers.

Most of the current work in this area has its foundation in impedance theory [37] and its application to robotics. Impedance may be thought of in terms of a physical system accepting a motion input and yielding a force output while its counterpart, admittance, implies a system which accepts force inputs and yields motion outputs. This distinction is crucial in the present paper. Controlling the interactions between a manipulator and its environment implies that the robot must act as an impedance while the environment acts as an admittance

[37]. But what of the case where two manipulators are acting in series as with a human arm/robot system? One approach is to model the human/machine interface as two impedances acting in series [38]. This seems to violate the traditional argument that along any degree of freedom of two dynamically interacting systems, if one system is an impedance than the other must necessarily be an admittance [37]. Furthermore, others [29] have noted that, in free space at least, humans seem to manipulate objects via position control rather than force control. Therefore, it would seem natural that the dynamic interaction between a human/robot system could be reasonably simulated by assuming the robot as an admittance.

Admittance control enforces the natural dynamic behavior of the human/robot system by controlling the robot's dynamic response to interaction forces rather than directly controlling force [39]. The literature provides various admittance control strategies including model-based and adaptive algorithms, but many suffer from complexity and computational burdens when implemented in higher degree of freedom systems, especially when no a priori trajectory knowledge is available [40].

This paper's overall approach is rather simplistic, but it allows straightforward implementation in an experimental test bed. We assume the human/robot interaction forces are available and measured with a force transducer. Based on the inertial characteristics of the virtual object with whose interaction we wish to simulate, a desired trajectory (including position, velocity, and acceleration) can be calculated in a straightforward manner via Newton's laws. Tightly controlled end effector acceleration leads to accurate dynamic forces at the human/robot interaction port. However, note that measured forces and moments are not controlled directly, but rather serve as the inputs to trajectory calculations. (An example

of a system where force outputs are not directly used for force control is offered in [41].) Errors in trajectory following will manifest themselves to the human user as a deviation from the expected mass of the object. The manner in which force feedback changes apparent end effector inertia has been previously noted in [42].

B. Software

Figure 1 presents the software architecture. The user (who ideally is fitted with a head mounted display) views the virtual object and manipulates it (and therefore the robot end effector) accordingly. Interaction forces are measured and input to rigid body mechanics software which calculates the position, velocity, and acceleration of the virtual object (and therefore the robot end effector) subject to pre-specified inertial characteristics.

The desired trajectory is specified in Cartesian space. We diverge from [42] here in that the robot is controlled in joint space rather than directly in Cartesian space. The method in [42] avoids the inverse kinematic calculations relating Cartesian positions to joint positions but requires the Jacobian and its time derivative which enables the calculation of the inverse kinematic joint velocities and accelerations. The technique presented in [42] also requires computation of the inverse of the robot's inertia matrix and its mobility tensor. Clearly there is a trade-off between the complexity of the inverse kinematic solutions versus the inversion of these matrices. Section IV reveals that, for robots with spherical wrist geometry and six or fewer degrees of freedom, the inverse kinematic calculations can be performed very efficiently.

One motivation for controlling the robot in joint space is that high performance and robust computed torque schemes can be used rather easily. Here, a variant of the computed torque method is employed that utilizes a pole placement approach which allows for

specification of the closed loop system eigenvalues [43]. The advantages of this method are that it provides for very good trajectory following performance and is an easy and intuitive way to investigate system stability via eigenvalue analyses. Furthermore, the feedforward plus feedback loops of this controller make use of abbreviated linear and nonlinear models [44] such that good six degree of freedom trajectory control is achieved with fewer than 700 calculations [43]. This is easily computed in real-time with modern hardware and frees up computer time for trajectory generation, inverse kinematics, and collision detection algorithms.

C. Hardware

Most haptic/kinesthetic feedback systems are custom made such that they are either one of a kind or very expensive to purchase. This makes replicating results nearly impossible. Thus far, much has been made of the design requirements for these devices such as low inertia, low friction, and no backlash. This would seem to preclude the use of common industrial manipulators as force displays in computer generated virtual environments. One of the goals of this paper is to determine the extent to which this is or isn't true. We therefore take as our test case the PUMA 560. This six degree of freedom manipulator has become a standard research tool for studying many aspects of robotics and is widely available.

The PUMA 560 is everything conventional wisdom says kinesthetic force display devices should not be. It is highly geared, which leads to backlash, and it has a great deal of inertia and friction. Its advantages are that it has a relatively large workspace and high force output capabilities. However, the potential for high force output can also pose safety concerns. Every effort has been made to ensure system safety by incorporating many of the

procedures and precautions given in [45] and [46]. Note, however, that safety is always a concern with force feedback devices and that this issue has largely been ignored in the literature.

In light of the PUMA's mechanical shortcomings, it would appear that this robot offers something of a worst case test scenario. In other words, we presume that modern robots will exhibit better performance and that results presented here can only be improved upon. However, the PUMA's limitations should not be considered in and of themselves. Rather, what is important is the extent to which its control system can mitigate such effects as inertia and friction. The two extremes are building a mechanically perfect device versus building a controller which compensates perfectly for mechanical limitations. All haptic/kinesthetic feedback systems will lie somewhere between these two extremes.

The data acquisition components of our system are also readily available and quite inexpensive. The LSI/11 VAL computer and servo cards in the PUMA have been replaced with a Trident Robotics TRC004 general purpose interface board which allows direct access to motor torques and encoders. A Trident TRC006 interface card serves as the I/O link between the TRC004 and a Pentium 90 Mhz personal computer. Watchdog timers are built into the system which shut down power to the robot in the case of hardware or software crashes. The Pentium computing capacity coupled with the Trident system enables us to achieve very high servo loop update rates as required. Table I presents a list of control system execution times in a manner similar to [47]. Note that the software was compiled with a 16-bit compiler and does not take full advantage of the 32-bit Pentium architecture. We expect these times will decrease significantly with the use of a more sophisticated 32-bit compiler.

Force data is measured with an ATI force transducer type Gamma with a parallel port interface. Force sensing ranges are 133 N (30 lbs) in the x and y directions and 266 N (60 lbs) in the z direction. Torque sensing range is 11.3 N-M (100 in-lb) about all three axes. Force resolution is 0.11 N (0.4 oz) in the x and y directions and 0.22 N (0.8 oz) in the z direction. Torque resolution is 0.006 N-M (0.8 in-oz). Clearly the force transducer range and resolution will place upper and lower limits on the range of dynamic interactions we are able to simulate.

Graphics images are displayed on a Silicon Graphics Indy. Future work will include the incorporation of head mounted displays. The following sections present important details on the calculations necessary to step through the control loop given in Figure 1.

III. TRAJECTORY GENERATION

The starting point for developing the control system for our human/robot interface begins with trajectory generation. Here, trajectory generation refers to specifying the time histories of acceleration, velocity, and position for some virtual object whose output motion we wish to simulate and follow with the PUMA end effector. This trajectory is initially evaluated in Cartesian space and converted to joint space with equations presented in the next section.

Rather than explicitly controlling the interaction forces between robot and human, we seek to develop a time-varying trajectory generation system which mimics the acceleration, velocity, and position of a generic virtual object subjected to time-varying input forces. The simulated acceleration of the virtual object will lead to the proper interaction forces imparted

to the user provided the control system is able to keep the robot close to the desired trajectory.

A. Virtual Object Kinematics

This sub-section presents the kinematic relationships which enable the determination of the acceleration, velocity, position, and orientation of the virtual object. These relationships derive from straightforward rigid body mechanics and are presented here for completeness.

We assume that the virtual object is a six degree of freedom rigid body. Traditionally, the kinematic quantities are derived by determining the motion of the translating and rotating body-fixed axes relative to some fixed inertial coordinate system. The following vector quantities are defined.

$$\begin{aligned} F &= (F_x,F_y,F_z) \\ \Gamma &= (L,M,N) \\ A_c &= (A_x,A_y,A_z) \\ V_c &= (U,V,W) \\ \omega &= (P,Q,R) \end{aligned} \tag{1}$$

where,

F and Γ are the external forces and moments acting on the object;

A_c is the acceleration vector of the object's center of gravity;

V_c is the body axis linear velocity vector of the mass center;

ω is the body axis angular velocity vector of the object.

The acceleration of the virtual object with respect to its body-fixed coordinate system is given by

$$A_c = \frac{dV_c}{dt} = \frac{\delta V_c}{\delta t} + \omega \otimes V_c \tag{2}$$

where δ denotes the acceleration as seen from the body axis system, and $\otimes$ denotes a vector cross product. Equation (2) is usually re-written as

$$\begin{aligned} A_x &= \dot{U} + WQ - VR \\ A_y &= \dot{V} + UR - WP \\ A_z &= \dot{W} + VP - UQ \end{aligned} \tag{3}$$

We now define the orientation of the virtual object with respect to an inertial coordinate system. There are a number of choices for specifying the angular coordinates of a rigid body. We choose the quaternion set known as Euler parameters. Even though Euler parameters are slightly less efficient computationally (requiring four degrees of freedom to describe an arbitrary orientation rather than three), they avoid the singularity problems of other angular coordinates, such as Euler angles, when angular velocity calculations are needed.

The total rotational transformation matrix between two coordinate systems in terms of Euler parameters is derived in [48] as

$$R_e = 2\begin{bmatrix} e_0^2+e_1^2-\frac{1}{2} & e_1e_2-e_0e_3 & e_1e_3+e_0e_2 \\ e_1e_2+e_0e_3 & e_0^2+e_2^2-\frac{1}{2} & e_2e_3-e_0e_1 \\ e_1e_3-e_0e_2 & e_2e_3+e_0e_1 & e_0^2+e_3^2-\frac{1}{2} \end{bmatrix} \tag{4}$$

where e_0-e_3 are the Euler parameters.

We also require the relationship between the angular velocity vector, ω, and the time derivatives of the Euler parameters. It can be shown that the angular velocity vector taken

with respect to the body fixed coordinate system is related to the Euler parameter derivatives as follows [48].

$$\dot{e} = \frac{1}{2} L^{T} \omega \tag{5}$$

where,

$$L = \begin{bmatrix} -e_1 & e_0 & e_3 & -e_2 \\ -e_2 & -e_3 & e_0 & e_1 \\ -e_3 & e_2 & -e_1 & e_0 \end{bmatrix} \tag{6}$$

This equation can then be integrated in order to determine orientation based on the Euler parameters.

Given the orientation of the virtual object, its position with respect to the inertial coordinate frame can then be determined. Global position derivatives are found by writing the local velocity vector (U,V,W) with respect to the inertial coordinate system by pre-multiplying this vector by the rotational transformation matrix of equation (4).

$$\begin{bmatrix} \frac{dX}{dt} \\ \frac{dY}{dt} \\ \frac{dZ}{dt} \end{bmatrix} = \left[R_e \right] \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{7}$$

where (X,Y,Z) is the global position vector found by integrating equation (7).

B. Virtual Object Dynamics

Calculating the position and orientation of our virtual object requires both linear and angular velocities. Also, for simulation of the inertial interaction forces and moments, the linear and angular accelerations must also be computed. Thus, a dynamic analysis is necessary. In general, the relationship between forces imparted to an object in gravity and its resulting acceleration can be given by Euler force equations of the following form.

$$\begin{aligned} mA_x &= m(\dot{U} + WQ - VR) = -mgr_{e_{31}} + F_x \\ mA_y &= m(\dot{V} + UR - WP) = -mgr_{e_{32}} + F_y \\ mA_z &= m(\dot{W} + VP - UQ) = -mgr_{e_{33}} + F_z \end{aligned} \tag{8}$$

where m is the virtual object's mass;

g is the gravitational constant;

r_{3i} is the bottom row of the rotational transformation matrix;

$F_{X\text{-}Y\text{-}Z}$ are the forces along the respective coordinate directions that a user applies to the virtual object and simultaneously to the robot's end effector.

The velocity derivatives in equation (8) are solved for and integrated in order to obtain the local velocity vector (U,V,W). Similarly, the relationship between the moments imparted to an object and the resulting angular acceleration are given by the following Euler moment equations.

$$I_c\dot{\omega} - \omega \otimes I_c\omega = \Gamma \tag{9}$$

where Γ is the vector of moments (L,M,N) about the body-fixed axes that a user applies to the virtual object and simultaneously to the robot's end effector; and I_c is the virtual object's inertia tensor with respect to the body fixed axes and referenced to the center of mass.

$$I_c = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{10}$$

The angular velocity derivatives in equation (9) are solved for and integrated in order to compute the angular velocity vector in body-fixed coordinates (P,Q,R).

Note that, in general, we do not apply forces directly at the center of mass of an object. Therefore, the moments imparted at the contact point must be transformed to their equivalent representations at the object's center of mass with the following.

$$\begin{aligned} F &= F_{cp} \\ \Gamma &= \Gamma_{cp} + d \otimes F_{cp} \end{aligned} \tag{11}$$

where, F_{cp} and Γ_{cp} are the contact point forces and moments, respectively, and d is the position vector from the center of mass to the contact point.

We also must transform the calculated virtual object center of mass motion to the contact point with the following.

$$A_{cp} = A_c + \dot{\omega} \otimes d + \omega \otimes \omega \otimes d \tag{12}$$

$$V_{cp} = V_c + \omega \otimes d \tag{13}$$

$$\begin{bmatrix} X_{cp} \\ Y_{cp} \\ Z_{cp} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + [R_e]d \tag{14}$$

The basic logic flow for the trajectory calculations given in this section is as follows. The inputs for this trajectory generation software are the initial conditions for all state

variables and a 6-vector containing force and moment data at the contact point. The force/moment data is generated when a user grasps a fixture attached to the robot's end effector and the subsequent forces and moments are measured by a force transducer (also mounted on the end effector) and fed to the data acquisition system.

The software contains desired parameters for the mass, inertia matrix, and the center of mass location relative to the contact point of the virtual object to be simulated. With inertial parameters and measured data from the force transducer, linear and angular accelerations are calculated from equations (8), (9), (11), and (12). Equations (8) and (9) are then integrated numerically using a 4th order Runge-Kutta algorithm [49] to obtain the local velocity vector (U,V,W) and the local angular velocity vector (P,Q,R). The linear velocities are transformed to the contact point with equation (13). Initial conditions are taken from the previous control loop cycle. Equation (7) is then integrated and transformed via equation (14) to calculate the position of the virtual object (and therefore the robot's end effector) and equation (5) is integrated to calculate the object's (and therefore the robot's end effector) orientation.

The output from the trajectory generation software specifies the complete trajectory (position, velocity, and acceleration) of the virtual object and robot end effector in Cartesian space. Velocity, position, and orientation data are stored and used as initial conditions during the next cycle of the control loop. However, since the system inputs from the force transducer are sampled only once per integration step instead of four times as required by the Runge-Kutta algorithm, numerical accuracy may be reduced [32].

Because a joint space servo loop is used to control the PUMA trajectory, the Cartesian trajectory given above must be converted into an equivalent joint space trajectory. The

following section details these calculations.

IV. INVERSE KINEMATICS

This section develops the equations necessary for converting the Cartesian trajectory of a virtual object (as specified in Section III) into the joint space of a PUMA 560. The robotics literature has concentrated primarily on studying the relationships between Cartesian position and orientation and the corresponding manipulator joint positions. This is understandable because the equations that relate the Cartesian positions to joint space are nonlinear and often have no analytical solution. Furthermore, these equations usually involve the existence of multiple solutions and singularity points. However, in many applications it is desirable to also convert Cartesian velocities and accelerations into their joint space equivalents. Although this problem is linear, it is generally solved numerically with computationally inefficient matrix inversions. This becomes a challenge for real-time control as the number of desired degrees of freedom grow.

In this paper, we derive *analytical* inverse kinematic relationships which relate the Cartesian position, velocity, and acceleration of a virtual object to an equivalent representation in robot joint space yielding the desired reference joint coordinates, q^*, $\dot{q}^*$, and $\ddot{q}^*$. This section demonstrates that this is possible due to the spherical nature of the wrist of the PUMA robot. A spherical wrist allows the inverse problem to be de-coupled from a complicated six degree of freedom problem, yielding two simple three degree of freedom problems. The techniques used in this section have been developed previously by [50] and [52] among others. The objective here is to apply these techniques to the PUMA 560 and in the process give the complete inverse kinematic solution steps from position

through acceleration.

A. Position

To solve for the desired joint positions, we must examine the forward kinematic equations for the PUMA 560 manipulator. Here, the particular methodology for representing the kinematics for serial chain robots makes use of Denavit-Hartenberg (D-H) notation [50]. The rotation matrix, R, relating the orientation of joint i+1 to joint i and the location of joint i+1 with respect to joint i (denoted as P_{i+1} below) can be written in terms of D-H parameters. Figure 2 and equations (15)-(17) present this formulation. (Note that c=cos() and s=sin().)

$$R = \begin{bmatrix} c\theta_{i+1} & -s\theta_i & 0 \\ s\theta_{i+1}c\alpha_i & c\theta_{i+1}c\alpha_i & -s\alpha_i \\ s\theta_{i+1}s\alpha_i & c\theta_{i+1}s\alpha_i & c\alpha_i \end{bmatrix} \tag{15}$$

$$P_{i+1} = \begin{bmatrix} a_i \\ -s\alpha_i d_{i+1} \\ c\alpha_i d_{i+1} \end{bmatrix} \tag{16}$$

or as an equivalent 4x4 homogeneous transformation matrix,

$$T_{i+1} = \begin{bmatrix} c\theta_{i+1} & -s\theta_i & 0 & a_i \\ s\theta_{i+1}c\alpha_i & c\theta_{i+1}c\alpha_i & -s\alpha_i & -s\alpha_i d_{i+1} \\ s\theta_{i+1}s\alpha_i & c\theta_{i+1}s\alpha_i & c\alpha_i & c\alpha_i d_{i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

where,

a_i is the link length and is defined as the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$;

α_i is the link twist and is defined as the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$;

d_i is the link offset and is defined as the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$;

θ_i is the joint angle and is defined as the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$.

The D-H parameters for the PUMA 560 can be found in [51] and are shown in Figure 2.

We assume that the origin of the coordinate system used in describing the position and orientation of the virtual object is at the base of the PUMA robot. Therefore, we can relate the position of the virtual object contact point to the PUMA tip position with

$$\begin{bmatrix} . & . & . & X \\ . & R_e & . & Y \\ . & . & . & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = T_0 = T_1 \, x \, T_2 \, x \, T_3 \, x \, T_4 \, x \, T_5 \, x \, T_6 \qquad (18)$$

Equation (18) indicates that we have a total of twelve equations from which to obtain a solution for the six joint positions, q^*. Nine of these equations come from the rotation matrix portion of equation (18) and three come from the position equation portion of equation (18). However, of the nine equations given by the rotation matrix, only three are independent. Thus, we have a total of six independent equations with which to solve for the desired joint positions.

We follow [50] by deriving the position of the PUMA end effector with respect to the joint 1 coordinate frame, which can be written symbolically as

$$P_6^1 = \begin{bmatrix} a_3\cos(\theta_2+\theta_3) + d_4\sin(\theta_2+\theta_3) + a_2\cos\theta_2 \\ d_2 + d_3 \\ d_4\cos(\theta_2+\theta_3) - a_3\sin(\theta_2+\theta_3) - a_2\sin\theta_2 \end{bmatrix} \tag{19}$$

This can also be written in terms of the joint 1 angle and the desired end effector position with respect to the base frame as

$$P_6^1 = \begin{bmatrix} X\cos\theta_1 + Y\sin\theta_1 \\ -X\sin\theta_1 + Y\cos\theta_1 \\ Z \end{bmatrix} \tag{20}$$

Equating the second entry of the two yields

$$Y\cos\theta_1 - X\sin\theta_1 = d_2 + d_3 \tag{21}$$

which has the form

$$A\cos\theta + B\sin\theta = C \tag{22}$$

and can be solved for θ with

$$\theta = 2\tan^{-1}\left(\frac{B\pm\sqrt{B^2+A^2-C^2}}{A+C}\right) \tag{23}$$

Since there are always two solutions to this equation, we must choose which angle to use. In this paper, the "near" (ie. $\min[\theta_i - \theta_{i-1}]$) solution is always chosen so that no sudden robot motions are caused due to large servo errors. Note that if A+C=0, we set θ=180°. If the robot encounters a workspace boundary or singularity, we simply set $\theta_{i+1}=\theta_i$.

Next, we evaluate θ_2 and θ_3. To do this, the first and third equations resulting from equating (19) and (20) are solved by squaring and adding the two equations. This eliminates

θ_2 and leaves

$$2a_3a_2\cos\theta_3 + 2d_4a_2\sin\theta_3 = (X\cos\theta_1 + Y\sin\theta_1)^2 + Z^2 - a_3^2 + d_4^2 + a_2^2 \qquad (24)$$

which, knowing θ_1 from the first step, is of the form of equation (22) and can be solved using equation (23). Expanding and collecting the terms of the elements of equations (19) and (20) gives

$$(-a_3\sin\theta_3 + d_4\cos\theta_3)\cos\theta_2 + (-d_4\sin\theta_3 - a_3\cos\theta_3 - a_2)\sin\theta_2 = Z \qquad (25)$$

which, knowing θ_3 from the second step, is also of the form of equation (22) and can be solved using equation (23).

Given the first three joint angles and the desired position of the reference trajectory, we can solve for the wrist angles. The relative rotation between the wrist (joint 6) and the arm (joint 3) is

$$R_w = R_6^3 = (R_1 \; x \; R_2 \; x \; R_3)^T \; x \; R_e \qquad (26)$$

where R_1-R_3 are known since we have already computed θ_1-θ_3 and R_e derives from the trajectory generation software. Applying [52]'s methodology, we first note that

$$R_4 \; x \; R_5 = \begin{bmatrix} \cos\theta_4\cos\theta_5 & -\cos\theta_4\sin\theta_5 & -\sin\theta_4 \\ \sin\theta_5 & \cos\theta_5 & 0 \\ \sin\theta_4\cos\theta_5 & -\sin\theta_4\sin\theta_5 & \cos\theta_4 \end{bmatrix} \qquad (27)$$

which is equal to

$$R_w \; x \; R_6^T = \begin{bmatrix} R_w(1,1)\cos\theta_6 - R_w(1,2)\sin\theta_6 & -R_w(1,3) & R_w(1,1)\sin\theta_6 + R_w(1,2)\cos\theta_6 \\ R_w(2,1)\cos\theta_6 - R_w(2,2)\sin\theta_6 & -R_w(2,3) & R_w(2,1)\sin\theta_6 + R_w(2,2)\cos\theta_6 \\ R_w(3,1)\cos\theta_6 - R_w(3,2)\sin\theta_6 & -R_w(3,3) & R_w(3,1)\sin\theta_6 + R_w(3,2)\cos\theta_6 \end{bmatrix} \quad (28)$$

such that

$$\theta_4 = \tan_2^{-1}\left[R_w(3,3) \; , \; R_w(1,3)\right] \quad (29)$$

Also,

$$R_5 \; x \; R_6 = \begin{bmatrix} \cos\theta_5\cos\theta_6 & -\cos\theta_5\sin\theta_6 & \sin\theta_5 \\ \sin\theta_6 & \cos\theta_6 & 0 \\ -\sin\theta_5\cos\theta_6 & \sin\theta_5\sin\theta_6 & \cos\theta_5 \end{bmatrix} \quad (30)$$

which is equal to

$$R_4^T \; x \; R_w = \begin{bmatrix} R_w(1,1)\cos\theta_4 + R_w(3,1)\sin\theta_4 & R_w(1,2)\cos\theta_4 + R_w(3,2)\sin\theta_4 & R_w(1,3)\cos\theta_4 + R_w(3,3)\sin\theta_4 \\ -R_w(1,1)\sin\theta_4 + R_w(3,1)\cos\theta_4 & -R_w(1,2)\sin\theta_4 + R_w(3,2)\cos\theta_4 & -R_w(1,3)\sin\theta_4 + R_w(3,3)\cos\theta_4 \\ -R_w(2,1) & -R_w(2,2) & -R_w(2,3) \end{bmatrix} \quad (31)$$

such that

$$\theta_5 = \tan_2^{-1}\left(\left[R_w(1,3)\cos\theta_4 + R_w(3,3)\sin\theta_4\right] \; , \; -R_w(2,3) \right) \quad (32)$$

and

$$\theta_6 = \tan_2^{-1}\left(\left[-R_w(1,1)\sin\theta_4 + R_w(3,1)\cos\theta_4\right] \; , \; \left[-R_w(1,2)\sin\theta_4 + R_w(3,2)\cos\theta_4\right] \right) \quad (33)$$

B. Velocity

We now solve for reference joint velocities given the joint angles. In general, the joint velocities are determined from

$$\begin{bmatrix} U \\ V \\ W \\ P \\ Q \\ R \end{bmatrix} = J^6 \dot{q}^* \tag{34}$$

where J^6 is the 6x6 Jacobian written in the PUMA end effector reference frame. Although numerical inversion of the Jacobian is straightforward, it is computationally expensive and may not be practical for real-time implementation in many cases. However, joint velocities and accelerations for six degree of freedom robots with spherical wrists (ie., the last three joint axes intersect) can be solved for analytically by partitioning the wrist kinematics from the arm kinematics [52].

We apply the methodology in [52] to a PUMA 560 by first noting that we are implicitly assuming that the local body fixed coordinate system of the virtual object is coincident with the joint 6 reference coordinate frame. The trajectory planner in Section III gives the linear velocity (U_{cp},V_{cp},W_{cp}) and angular velocity (P,Q,R) of the virtual object, and therefore the PUMA end effector, with respect to this reference frame. Because of the spherical geometry of the PUMA's end effector, if we write this velocity in the reference frame of joint 3, the wrist linear velocity becomes a function of the first three joint velocities only.

$$V_3 = \begin{bmatrix} d_4(\dot{\theta}_2+\dot{\theta}_3)-(d_2+d_3)\cos(\theta_2+\theta_3)\dot{\theta}_1+a_2\sin\theta_3\dot{\theta}_2 \\ a_3(\dot{\theta}_2+\dot{\theta}_3)+(d_2+d_3)\sin(\theta_2+\theta_3)\dot{\theta}_1+a_2\cos\theta_3\dot{\theta}_2 \\ (a_3\cos(\theta_2+\theta_3)+d_4\sin(\theta_2+\theta_3)+a_2\cos\theta_2)\dot{\theta}_1 \end{bmatrix} \tag{35}$$

or

$$V_3 = \begin{bmatrix} -(d_2+d_3)\cos(\theta_2+\theta_3) & d_4+a_2\sin\theta_3 & d_4 \\ (d_2+d_3)\sin(\theta_2+\theta_3) & a_3+a_2\cos\theta_3 & a_3 \\ a_3\cos(\theta_2+\theta_3)+d_4\sin(\theta_2+\theta_3)+a_2\cos\theta_2) & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{36}$$

also,

$$V_3 = [R_4][R_5][R_6] \begin{bmatrix} U_{cp} \\ V_{cp} \\ W_{cp} \end{bmatrix} \tag{37}$$

By equating the above expressions, the reference velocities of joints 1-3 can be found analytically either through symbolic inversion of the matrix in equation (36) or by Gaussian elimination.

In order to determine the last three reference joint velocities, we write the end effector relative angular velocity with respect to the PUMA arm in the joint 4 reference coordinate frame as

$$\omega_{rel} = [R_5][R_6]\omega_6 - [R_4]^T\omega_3 = [R_5][R_6] \begin{bmatrix} P \\ Q \\ R \end{bmatrix} - [R_4]^T \begin{bmatrix} -\sin(\theta_2 + \theta_3)\dot{\theta}_1 \\ -\cos(\theta_2 + \theta_3)\dot{\theta}_1 \\ \dot{\theta}_2 + \dot{\theta}_3 \end{bmatrix} \tag{38}$$

which, for the PUMA 560, can also be evaluated as

$$\omega_{rel} = \begin{bmatrix} \sin\theta_5\dot{\theta}_6 \\ \dot{\theta}_5 \\ \cos\theta_5\dot{\theta}_6 + \dot{\theta}_4 \end{bmatrix} \tag{39}$$

or,

$$\begin{bmatrix} \dot{\theta}_4^* \\ \dot{\theta}_5^* \\ \dot{\theta}_6^* \end{bmatrix} = \begin{bmatrix} -\dfrac{\cos\theta_5^*\omega_{rel_x} - \sin\theta_5^*\omega_{rel_z}}{\sin\theta_5^*} \\ \omega_{rel_y} \\ \dfrac{\omega_{rel_x}}{\sin\theta_5^*} \end{bmatrix} \tag{40}$$

Substituting equation (38) into equation (40) leads to a solution for the joint 4 - joint 6 reference velocities in terms of the desired virtual object angular velocity.

C. Acceleration

The final step in the process of specifying the trajectory that we wish the force reflecting robot to follow, is solving for the reference joint accelerations. These quantities are perhaps most critical because simulating the dynamic force/moment interaction between a human and a virtual object requires accurate control of the Cartesian accelerations.

By differentiating equation (34), the general relationship between Cartesian and joint space accelerations is

$$\begin{bmatrix} A_x \\ A_y \\ A_z \\ \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = J^6 \ddot{q}^\bullet + \dot{J}^6 \dot{q}^\bullet \tag{41}$$

Solving this equation numerically in real-time is challenging due to the computational burden of inverting the 6x6 Jacobian and evaluating its time derivative.

Again, we apply the methodology in [52] to the PUMA 560. The trajectory planner in Section III gives the linear acceleration (A_x, A_y, A_z) and angular acceleration ($\dot{P}$, $\dot{Q}$, $\dot{R}$) of the contact point with the virtual object, and therefore the PUMA end effector, with respect to the joint 6 reference frame. Because of the spherical geometry of the PUMA's end effector, if we write acceleration in the reference frame of joint 3, the wrist linear acceleration is, again, a function of the first three joint velocities and accelerations only. For the PUMA 560, it can be shown that,

$$\begin{aligned} \dot{J}^3\dot{\theta} = -\Big(d_4\sin(2\theta_2+2\theta_3) + a_2\cos(2\theta_2+\theta_3) + a_2\cos\theta_3 + a_3\cos(2\theta_2+2\theta_3) + a_3\Big)\frac{\dot{\theta}_1^2}{2} \\ + a_2\cos\theta_3\dot{\theta}_2^2 + a_3\dot{\theta}_2^2 + a_3\dot{\theta}_3^2 + 2a_3\dot{\theta}_2\dot{\theta}_3 \end{aligned} \tag{42}$$

$$\begin{aligned} \Big(-d_4\cos(2\theta_2+2\theta_3) + a_2\sin(2\theta_2+\theta_3) + a_2\sin\theta_3 + a_3\sin(2\theta_2+2\theta_3) + d_4\Big)\frac{\dot{\theta}_1^2}{2} \\ + a_2\sin\theta_3\dot{\theta}_2^2 + d_4\dot{\theta}_2^2 + d_4\dot{\theta}_3^2 + 2d_4\dot{\theta}_2\dot{\theta}_3 \end{aligned}$$

$$-(d_2+d_3)\dot{\theta}_1^2 + (2d_4\cos(\theta_2+\theta_3) - 2a_3\sin(\theta_2+\theta_3) - 2a_2\sin\theta_2)\dot{\theta}_1\dot{\theta}_2 + (2d_4\cos(\theta_2+\theta_3) - 2a_3\sin(\theta_2+\theta_3))\dot{\theta}_1\dot{\theta}_3$$

also note,

$$A_3 = [R_4][R_5][R_6]\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} \tag{43}$$

such that,

$$\ddot{\theta} = (J^3)^{-1}\left([R_4][R_5][R_6]\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} - \dot{J}_3\dot{\theta}\right) \tag{44}$$

can be solved for the first three joint accelerations. Note that we already have a symbolic expression for $(J^3)^{-1}$ from calculating the first three joint velocities, so a repeat of this operation is unnecessary.

In order to calculate the last three reference joint accelerations, we need the end effector relative angular acceleration with respect to the PUMA arm in the joint 4 reference coordinate frame. This can be found from differentiating the relative angular velocity as

$$\dot{\omega}_{rel} = \dot{\omega}_6^4 - \dot{\omega}_3^4 - \omega_3^4 \otimes \omega_{rel}^4 \tag{45}$$

or,

$$\dot{\omega}_{rel} = [R_5][R_6]\dot{\omega}_6 - [R_4]^T\dot{\omega}_3 - [R_4]^T\omega_3 \otimes \omega_{rel}$$

$$= [R_5][R_6]\begin{bmatrix}\dot{P}\\ \dot{Q}\\ \dot{R}\end{bmatrix} - [R_4]^T\begin{bmatrix}-\sin(\theta_2+\theta_3)\ddot{\theta}_1 - \cos(\theta_2+\theta_3)\dot{\theta}_1\dot{\theta}_2 - \cos(\theta_2+\theta_3)\dot{\theta}_1\dot{\theta}_3\\ -\cos(\theta_2+\theta_3)\ddot{\theta}_1 + \sin(\theta_2+\theta_3)\dot{\theta}_1\dot{\theta}_2 + \sin(\theta_2+\theta_3)\dot{\theta}_1\dot{\theta}_3\\ \ddot{\theta}_2 + \ddot{\theta}_3\end{bmatrix} \quad (46)$$

$$- [R_4]^T\begin{bmatrix}-\sin(\theta_2+\theta_3)\dot{\theta}_1\\ \cos(\theta_2+\theta_3)\dot{\theta}_1\\ \dot{\theta}_2 + \dot{\theta}_3\end{bmatrix} \otimes \begin{bmatrix}\sin\theta_5\dot{\theta}_6\\ \dot{\theta}_5\\ \dot{\theta}_4 + \cos\theta_5\dot{\theta}_6\end{bmatrix}$$

which, for the PUMA 560, can also be evaluated as

$$\dot{\omega}_{rel} = \begin{bmatrix}\sin\theta_5\ddot{\theta}_6 + \cos(\theta_5)\dot{\theta}_5\dot{\theta}_6 - \dot{\theta}_4\dot{\theta}_5\\ \ddot{\theta}_5 + \sin\theta_5\dot{\theta}_4\dot{\theta}_6\\ \cos\theta_5\ddot{\theta}_6 + \ddot{\theta}_4 - \sin(\theta_5)\dot{\theta}_5\dot{\theta}_6\end{bmatrix} \quad (47)$$

or,

$$\ddot{\theta}_6^* = \frac{\dot{\omega}_{rel_1} + \dot{\theta}_4\dot{\theta}_5 - \cos\theta_5\dot{\theta}_5\dot{\theta}_6}{\sin\theta_5}$$
$$\ddot{\theta}_5^* = \dot{\omega}_{rel_2} - \sin\theta_5\dot{\theta}_4\dot{\theta}_6 \quad (48)$$
$$\ddot{\theta}_4^* = \dot{\omega}_{rel_3} + \sin\theta_5\dot{\theta}_5\dot{\theta}_6 - \cos\theta_5\ddot{\theta}_6$$

We have now completely specified the desired PUMA trajectory in joint space given the contact dynamics between a human operator and a virtual object. Because the inverse kinematic equations are symbolically implemented, we can optimize them by grouping like terms that only need be calculated once during each sampling instant. This greatly reduces the total number of calculations over those required in strictly numerical approaches. Table I provides computation details which illustrate these efficiencies.

V. TEST RESULTS

This section presents some preliminary experimental results. In order to make the data more straightforward to analyze, we present four test cases involving force and moment interaction with objects having a wide range of inertial characteristics and motion. We further divide the experiments into purely translational and purely rotational cases. (The system readily handles combined translational and rotational motion). Finally, we give an example of a collision between the virtual object and a stiff wall to demonstrate the system's capability of simulating non-inertial force interaction as well as purely inertial interactions.

A. Translational Motion

This subsection analyzes the PUMA's ability to adequately simulate inertial force-only interaction. The virtual object is cube shaped with all sides measuring 12.5 cm. An interface fixture mounted to the force transducer on the end effector is used for manipulating the object as shown in Figure 4. Compensation is required to balance the gravity and inertia of the fixture. To review, the user applies forces to this interface fixture while trying to move the virtual object and therefore the robot end effector. These forces are measured, digitized, and used as real-time input to the differential equations of motion of the object. These equations yield a desired trajectory which the robot is constrained to follow.

Figures 5-8 demonstrate position following along with velocity, acceleration, and force profiles for the body-fixed y direction of a 5 kg mass. The user is forcing the object/robot to follow a circular trajectory with a diameter of approximately 20 cm. The measured trajectory, indicated by the boxes in the figures, shows that the motion is correct. Figures 9 and 10 show total acceleration and force vector magnitudes when simulating a 5 kg and 150 kg mass. Special consideration needs to be given to the acceleration and velocity results.

In general, the PUMA robot comes only with optical encoders for joint position. Therefore, some form of numerical differentiation is required to find joint velocities and accelerations. Encoder resolution also becomes an issue here at high sampling rates and/or at low speeds. Reasonable joint acceleration data cannot be obtained from finite differenced velocity data. Finite difference calculations were used to find the joint velocities. The velocities were then spline fit using the algorithm in [53] and this spline was differentiated to determine joint accelerations. Given the joint positions, velocities, and accelerations, the data was further post-processed with forward kinematics software which calculated the Cartesian trajectory. An alternative method would be to mount three axis linear and angular accelerometers at the end effector. However, given that the PUMA kinematic parameters are fixed, post processing the joint data is the more practical solution. Still, future validation work should include the use of accelerometers.

As a final check of our results, we divide the force magnitude by the acceleration magnitude. Under ideal circumstances this plot would be constant at the desired mass value. Figures 11 and 12 present these results. Table II presents some statistics. Note that the data from the force transducer is not noise free. Since this data is input directly into the trajectory generation software, the reference cartesian trajectory is not noise free. The smoothing process which was outlined above is not perfect and leads to mismatches between the splined acceleration data and actual data. In short, we believe measured acceleration data from accelerometers would show an improvement in results over those presented in Table II. Nevertheless, the fact that the mean of this data is very close to the nominal value and that the frequency of the noise is quite high leads us to believe that the PUMA can satisfactorily simulate dynamic force interactions over a very wide range of impedances. Furthermore, we

have subjective evidence from users of this system which indicates that the forces "feel" right.

B. Rotational Motion

We now look at experimental results involving only rotational motion due to moment-only inputs. The same cube-shaped object is used in these experiments as well. Figures 13-15 show profiles of body-fixed, z-axis motion and moment input for an object with an inertia tensor equal to diag[0.83,0.83,0.83] kg-m^2. The figures again demonstrate good trajectory following through angular acceleration. Figures 16 and 17 show total angular acceleration and moment vector magnitudes when simulating a diag[0.83,0.83,0.83] kg-m^2 and diag[12.5,12.5,12.5] kg-m^2 object respectively.

Again, as a final check of our results, we divide the moment magnitude by the angular acceleration magnitude. Figures 18 and 19 present these results. Statistics are given in Table II. It is interesting to note that the standard deviations presented in Table II are all about 10 percent of the mean for all but the diag[12.5,12.5,12.5] kg-m^2 case. Encoder resolution becomes a factor here (as does force transducer noise) because the high inertia kept motion at low levels even with reasonably large moment inputs.

One solution to these challenges is to filter both the force transducer data (some filtering is already done in hardware) and the finite differenced joint velocity data. We attempted to implement simple second order filtering, but with limited success. Even small phase lags associated with these types of filters were noticeable to users and the system stability was adversely affected as the lags became greater. Further analysis is needed in this area and more sophisticated filtering techniques may become necessary if the results are to be noticeably improved.

C. Collision Simulation

This subsection presents an experiment involving force simulation of contact with a stiff object. Previous researchers [54] have noted the importance of high sample rates and inherent system damping when trying to increase a virtual wall's stiffness. This work has its foundation in direct force control, rather than with admittance control as presented here. We have found that the approach presented here gives good stiff wall "feel" without the need for extremely high sample rates. Figures 20-21 illustrates this point. For simplicity, we show a case where only one dimensional translational motion is allowed. A 10 kg virtual object (and therefore, the robot end effector) is given a push in the direction of the wall. The wall is modeled as a simple spring/damper system. The control system update rate is set at 500 Hz while the wall natural frequency is set at 25 Hz with critical damping. Although this sample rate is significantly lower than that called for in [54], Figures 20-21 reveal that the system's initial response to the collision is quite good but has some oscillation after initial impact. Note that we are attempting to simulate a nearly perfectly plastic collision such that the object's velocity should be nearly zero immediately after impact. Note, again, that the acceleration results were not directly measured. More rigorous experiments will include the use of accelerometers. However, subjective experience with other haptic devices indicates that the this system works quite well and avoids the contact instability problems found with other devices. In short, users indicate that the PUMA does remarkably well at providing the stiff "feel" of a wall.

VI. DISCUSSION AND CONCLUSIONS

This paper demonstrates the feasibility of using readily available robotics equipment for simulating dynamic force and moment interactions between humans and virtual objects. A PUMA 560 and off the shelf visualization hardware provided a test bed. Feedforward and feedback control loops were used to compensate for the PUMA's inertia and friction. Equations have been derived which efficiently compute the dynamic trajectory of the simulated six degree of freedom object and convert this Cartesian space trajectory into joint space inputs for the feedforward and feedback loops.

The paper points out the effects of force transducer noise and low encoder resolution on system performance. Simple second order filtering was found to be inadequate as human subjects were able to sense even small phase lags and system stability suffered.

In addition to dynamic force interaction, we found that the PUMA robot was also able to simulate non-inertial force interaction with objects such as stiff walls. The admittance control approach used in this paper allowed for much lower sampling rates to achieve stable interaction with stiff objects than with traditional direct force control approaches.

The test results presented here support the conclusion that conventional robots with high friction, backlash, and inertia can be made to be effective kinesthetic/haptic force displays provided an adequate control system is used that can compensate for these effects. Future work will include the use of accelerometers in validation studies and further analysis of force and velocity filtering issues.

REFERENCES

[1] C. Sayers and R. Paul, "Synthetic fixturing", in *The Proceedings of the 1993 ASME Winter Annual Meeting: Advances in Robotics, Mechatronics, and Haptic Interfaces* (New Orleans, LA), November 1993, pp. 37-46.

[2] L.B. Rosenberg, "Virtual haptic overlays enhance performance in telepresence tasks," in *The Proceedings of the International Society for Optical Engineering (SPIE)* (Boston, MA), vol. 2351, 1995, pp. 99-108.

[3] J. Funda and R. Paul, "A symbolic teleoperator interface for time-delayed underwater robot manipulation," in *The Proceedings of Oceans '91* (Honolulu, HI), 1991, pp. 1526-1533.

[4] C. Fagerer, D. Dickmanns, and E.D. Dickmanns, "Visual grasping with long delay time of a free floating object in orbit," *Autonomous Robots*, vol. 1, no. 1, pp. 53-68, 1994.

[5] B.R. Loftin, "Virtual environments for aerospace training," in *Proceedings of the 1994 Wescon Conference* (Anaheim, CA), 1994, pp. 384-387.

[6] F.P. Brooks, Jr., M. Ouh-Young, J.J. Batter, and P.J. Kilpatrick, "Project GROPE - haptic displays for scientific visualization," *Computer Graphics*, vol. 24, no. 4, pp. 177-185, 1990.

[7] R.C. Goertz and W.M. Thompson, "Electronically controlled manipulator," *Nucleonics*, pp. 46-47, 1954.

[8] J.J. Batter and F.P. Brooks, Jr., "GROPE-I: a computer display to the sense of feel," in *Information Processing: Proceedings of the IFIP Congress '71*, 1971, pp. 759-763.

[9] A.K. Bejczy and J.K. Salisbury, "Kinesthetic coupling for remote manipulators," *Computers in Mechanical Engineering*, vol 2, no. 1, pp. 48-62, 1983.

[10] B. Hannaford, L. Wood, D.A. McAffee, and H. Zak, "Performance evaluation of a six-axis generalized force-reflecting teleoperator," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 620-633, 1991.

[11] S.A. Stansfield, "A robotic perceptual system utilizing passive vision and active touch," *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 138-161, 1988.

[12] W. McNeely, G.C. Burdea, B. Hannaford, M. Hirose, S. Jacobsen, K. Salisbury, and S. Tachi, "Wither force feedback," in *The Proceedings of the 1995 Virtual Reality Annual International Symposium* (Triangle Park, NC), 1995, pp. 226-227.

[13] R.E. Ellis, O.M. Ismaeli, and M.G. Lipsett, "Design and evaluation of a high-performance prototype planar haptic interface," in *The Proceedings of the 1993 ASME Winter Annual Meeting: Advances in Robotics, Mechatronics, and Haptic Interfaces* (New Orleans, LA), 1993, pp. 55-64.

[14] P.A. Millman, M. Stanley, and J.E. Colgate, "Design of a high performance haptic interface to virtual environments," in *The Proceedings of the 1993 IEEE Virtual Reality Annual International Symposium* (Seattle, WA), 1993, pp. 216-222.

[15] H. Iwata, "Artificial reality with force-feedback: development of desktop virtual space with compact master manipulator," *Computer Graphics*, vol. 24, no. 4, pp. 165-170, August 1990.

[16] B.A. Marcus, B. An, and B. Eberman, "EXOS research on master controllers for robotic devices," in *Fifth Annual Workshop on Space Operations, Applications, and Research (SOAR '91)*, 1991, pp. 238-245.

[17] T.H. Massie and J.K. Salisbury, "The PHANToM haptic interface: a device for probing virtual objects," in *The Proceedings of the 1994 ASME Winter Annual Meeting: Dynamic Systems and Control* (Chicago, IL), vol. 55-1, 1994, pp. 295-301.

[18] M. Ishii and M. Sato, "Force sensations in pick-and-place tasks," in *The Proceedings of the 1994 ASME Winter Annual Meeting: Dynamic Systems and Control* (Chicago, IL), vol. 55-1, 1994, pp. 339-344.

[19] P. Buttolo and B. Hannaford, "Pen-based force display for precision manipulation in virtual environments," in *The Proceedings of the 1995 Virtual Reality Annual International Symposium* (Research Triangle, NC), 1995, pp. 217-224.

[20] G.R. Luecke and J. Winkler, "A magnetic interface for robot-applied virtual forces," in *The Proceedings of the 1994 ASME Winter Annual Meeting: Dynamic Systems and Control* (Chicago, IL), 1994, pp. 271-275.

[21] R.L. Hollis, S.E. Salcudean, and A.P. Allan, "A six degree-of-freedom magnetically levitated variable compliance fine motion wrist: design, modeling, and control," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 320-332, 1991.

[22] S.E. Salcudean and T.D. Vlaar, "On the simulation of stiff walls and static friction with a magnetically levitated input/output device," in *The Proceedings of the 1994 ASME Winter Annual Meeting: Dynamic Systems and Control* (Chicago, IL), 1994, pp. 303-309.

[23] Y. Adachi, T. Kumano, and K. Ogino, "Sensory evaluation of virtual haptic push-buttons," in *The Proceedings of the 1994 ASME Winter Annual Meeting* (Chicago, IL), 1994, pp. 361-368.

[24] Y. Adachi, T. Kumano, and K. Ogino, "Intermediate representation for stiff virtual objects," in *The Proceedings of the 1995 Virtual Reality Annual International Symposium* (Research Triangle, NC), 1995, pp. 203-210.

[25] T. Takeda and Y. Tsutsul, "Development of a virtual training environment," in *The Proceedings of the 1993 ASME Winter Annual Meeting: Advances in Robotics, Mechatronics, and Haptic Interfaces* (New Orleans, LA), 1993, pp. 1-10.

[26] M. Ouh-young, et al., "Using a manipulator for force display in molecular docking," in *The Proceedings of the 1988 IEEE International Conference on Robotics and Automation* (Philadelphia, PA) , 1988, pp. 1824-1829.

[27] S. Leake, "A cartesian force reflecting teleoperation system," *Computers in Electrical Engineering*, Vol. 17, No. 3, pp. 133-146, 1991.

[28] S.C. Jacobsen, F.M. Smith, E.K. Iverson, and D.K. Backman, "High performance, high dexterity, force reflective teleoperator," in *The Proceedings of the 38th Conference on Remote Systems Technology* (Washington, DC), 1990, pp. 180-185.

[29] H. Kazerooni and M.G. Her, "The dynamics and control of a haptic interface device," *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 4, pp. 453-464, August 1994.

[30] H. Hashimoto, Y. Kunii, M. Buss, and F. Harashima, "Dynamic force simulator for force feedback human-machine interaction," in *The Proceedings of the 1993 Virtual Reality Annual International Symposium* (Seattle, WA), 1993, pp. 209-215.

[31] K. Kosuge, Y. Fujisawa, and T. Fukuda, "Control of a man-machine system interacting with the environment," *Advanced Robotics*, Vol 8., No. 4, pp. 429-441, 1994.

[32] B. Gillespie and M. Cutkosky, "Interactive dynamics with haptic display," in *The Proceedings of the 1993 ASME Winter Annual Meeting: Advances in Robotics, Mechatronics, and Haptic Interfaces* (New Orleans, LA), 1993, pp. 65-72.

[33] W.A. McNeely, "Robot graphics: a new approach to force feedback for virtual reality," in *The Proceedings of the 1993 Virtual Reality Annual International Symposium* (Seattle, WA), 1986, pp. 336-341.

[34] B. Hannaford, "A design framework for teleoperators with kinesthetic feedback," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 4, pp. 426-434, August 1989.

[35] J.E. Colgate, "Coupled stability of multiport systems - theory and experiments," *The Journal of Dynamic Systems, Measurement, and Control*, Vol. 116, pp. 419-428, September 1994.

[36] H. Kazerooni and T.J. Snyder, "Case study on haptic devices: human-induced instability in powered hand controllers," *The Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, January-February 1995.

[37] N. Hogan, "Impedance control: an approach to manipulation (part 1 - theory)," *The Journal of Dynamic Systems, Measurement, and Control*, Vol. 107, pp. 1-7, March 1985.

[38] G. Burdea and J. Zhuang, "Dexterous telerobotics with force feedback - an overview (part 1 - human factors)," *Robotica*, Vol. 9, pp. 171-178, 1991.

[39] B.B. Mathewson and W.S. Newman, "Integration of force strategies and natural admittance control," in *The Proceedings of the 1994 AMSE Winter Annual Meeting: Dynamic Systems and Control* (Chicago, IL), 1994, pp. 237-242.

[40] R. Colbaugh, H. Seraji, and K. Glass, "Adaptive compliant motion control for dexterous manipulators," *The International Journal of Robotics Research*, Vol. 14, No. 3, pp. 270-280, June 1995.

[41] W.S. Kim, "Development of a new force regulating control scheme and an application to a teleoperation training simulator," in *The Proceedings of the 1992 International Conference on Robotics and Automation* (Nice, France), 1992, pp. 1412-1419.

[42] N. Hogan, "Stable execution of contact tasks using impedance control," in *The Proceedings of the 1987 International Conference on Robotics and Automation* (Raleigh, NC), 1987, pp. 1047-1054.

[43] C.L. Clover, "The use of symbolic, computer generated control laws in high speed trajectory following: analysis and experiments," to be submitted to *The IEEE Transactions on Robotics and Automation.*

[44] C.L. Clover, "The utility of abbreviated linear and nonlinear robot models," to be submitted to *The Journal of Dynamic Systems, Measurement, and Control.*

[45] M.S. Sanders, P.D. Brodt, and R.N. Wentworth, "Safety in the industrial robot environment," in *University Programs in Computer-Aided Engineering, Design, and Manufacturing: Proceedings of the Seventh Annual Conference* (Laramie, WY), 1989, pp. 191-197.

[46] N. Hogan, "Controlling impedance at the man/machine interface," in *The Proceedings of the 1989 IEEE International Conference on Robotics and Automation* (Scottsdale, AZ), 1989, pp. 1626-1631.

[47] B.W. Drake and T.C.S. Hsia, "Implementation of a unified robot kinematics and inverse dynamics algorithm on a DSP chip," *IEEE Transactions on Industrial Electronics*, Vol. 40, No. 2, pp. 273-281, April 1993.

[48] P.E. Nikravesh, *Computer-Aided Analysis of Mechanical Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[49] R.K. Miller, *Introduction to Differential Equations.* Englewood Cliffs, NJ: Prentice Hall, 1991.

[50] J.J. Craig, *Introduction to Robotics: Mechanics and Control.* Reading, MA:Addison-Wesley, 1989.

[51] B. Armstrong, O. Khatib, J. Burdick, "The explicit dynamic model and inertial parameters of the PUMA 560 arm," in *Proceedings of the 1986 International Conference on Robotics and Automation* (San Francisco, CA), 1986, pp. 510-518.

[52] J.M Hollerbach and G. Sahar, "Wrist-partitioned inverse kinematic accelerations and manipulator dynamics," *The International Journal of Robotics Research*, vol. 2, no. 4, pp. 61-76, 1983.

[53] H.J. Woltring, "A Fortran package for generalized, cross-validatory spline smoothing and differentiation," *Advances in Engineering Software*, Vol. 8, No. 2, pp. 1-4-107, 1986.

[54] J.E. Colgate and J.M. Brown, "Factors affecting the z-width of a haptic display," in *The Proceedings of the 1994 Conference on Robotics and Automation* (San Diego, CA), 1994, pp. 3205-3210.

[55] M.B. Leahy, Jr. et al., "Efficient PUMA manipulator Jacobian calculation and inversion," *Journal of Robotic Systems*, Vol. 4, No. 2, pp. 185-197, 1987.

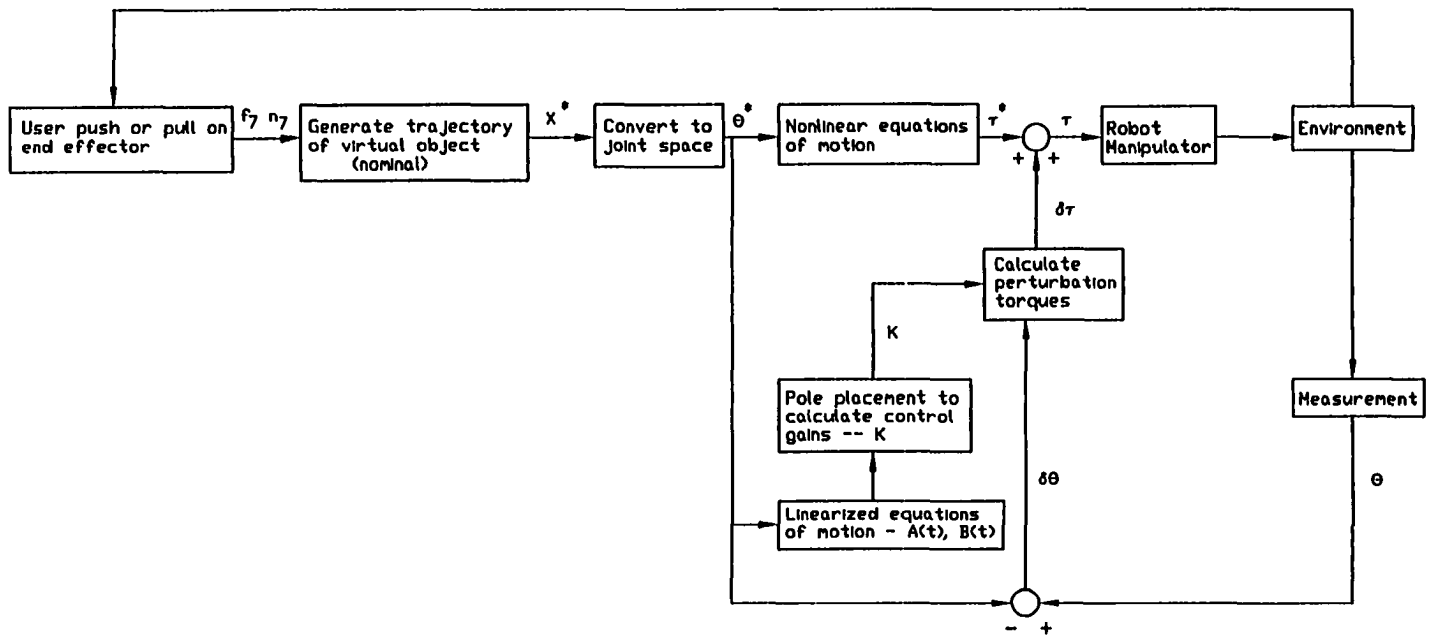

Figure 1: Controller logic for a force reflective robotic system

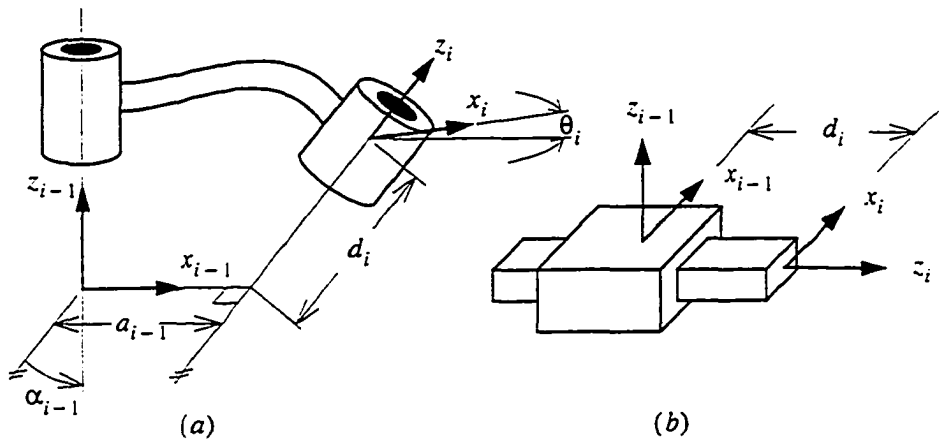

Figure 2: D-H parameter descriptions between link i and link i-1 for (a) a revolute joint and (b) a prismatic joint

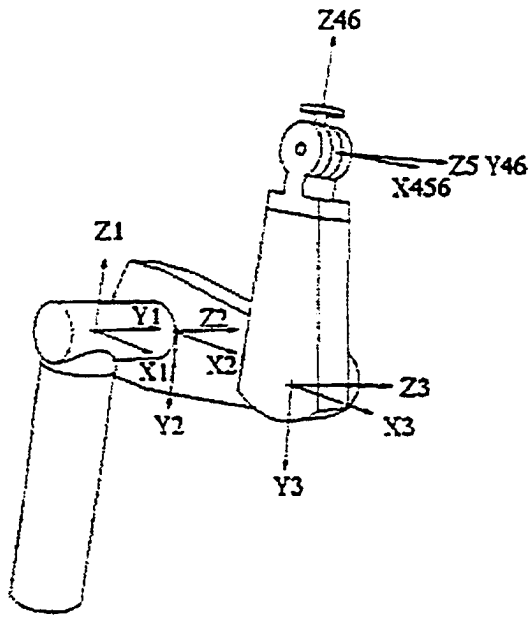

Figure 3: PUMA 560 coordinate axes

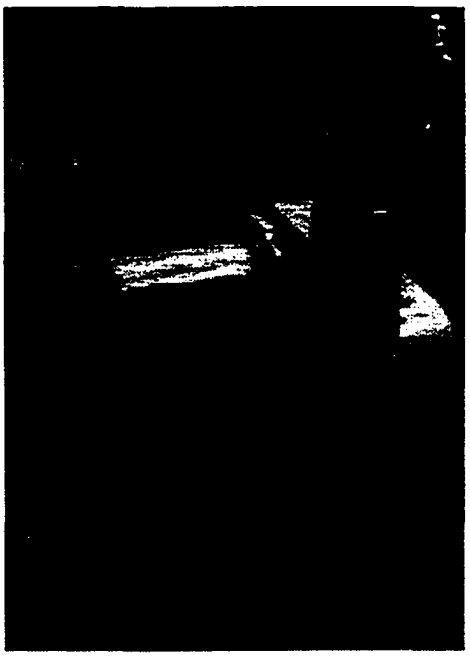

Figure 4: The PUMA 560 mounted with force/moment transducer and gripping fixture

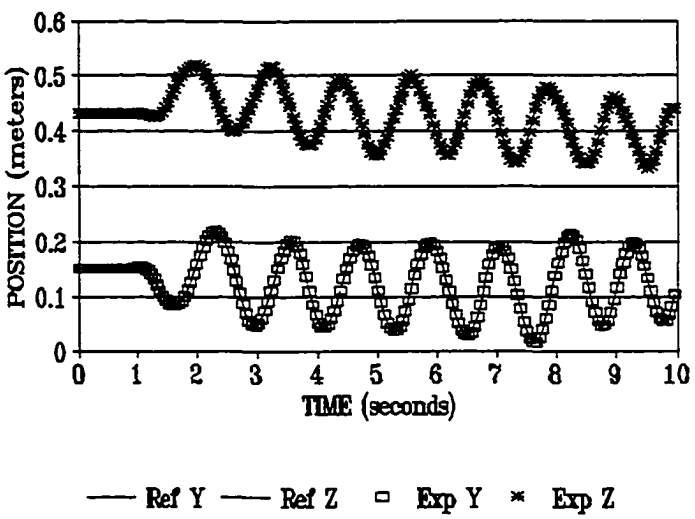

Figure 5: Reference and experimental Cartesian X and Y positions with respect to the global origin

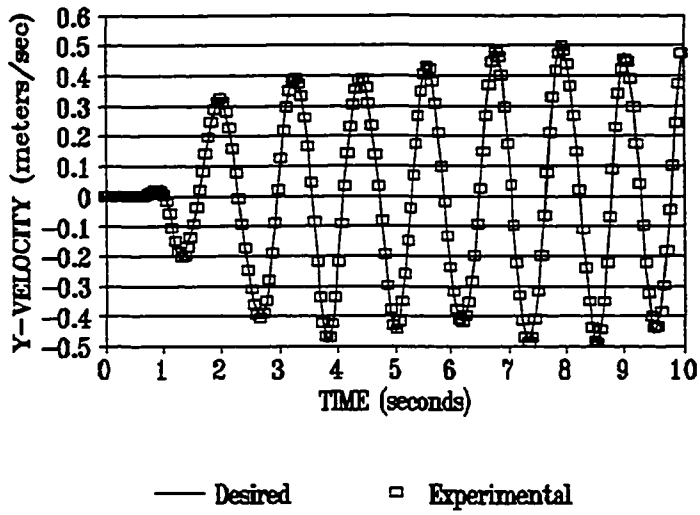

Figure 6: Reference and experimental results for the velocity component in the body fixed y-axis direction

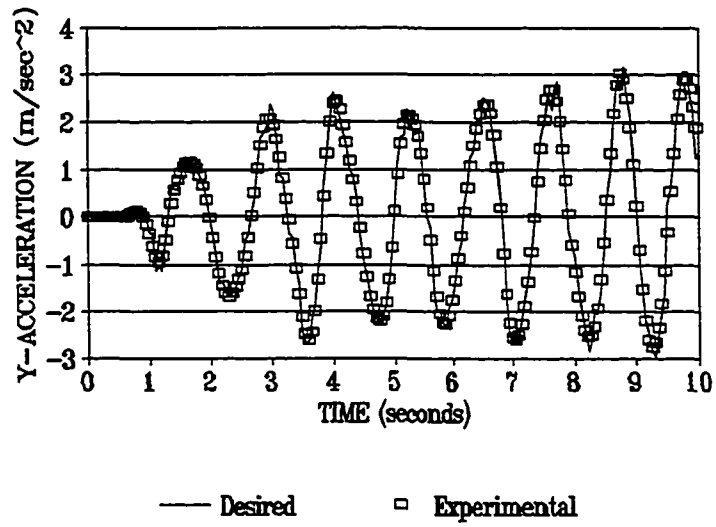

Figure 7: Reference and experimental results for the acceleration component in the body fixed y-axis direction

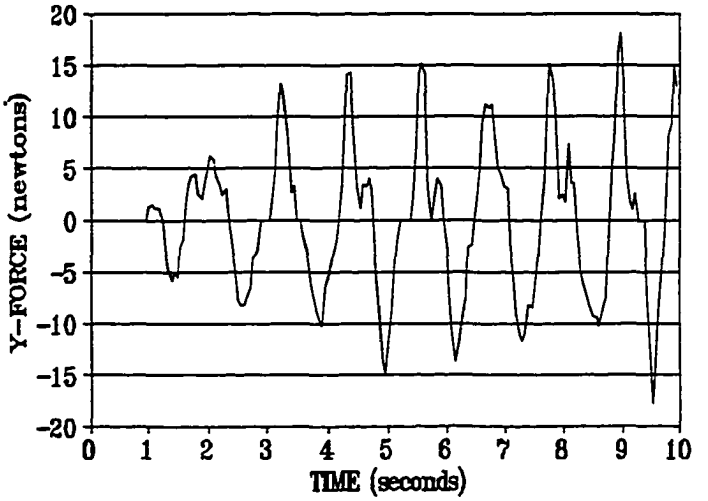

Figure 8: Force component in the body fixed y-axis direction

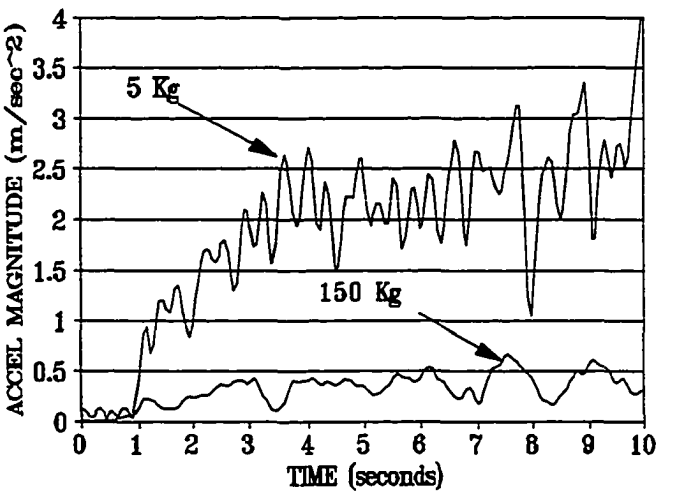

Figure 9: Total acceleration magnitudes

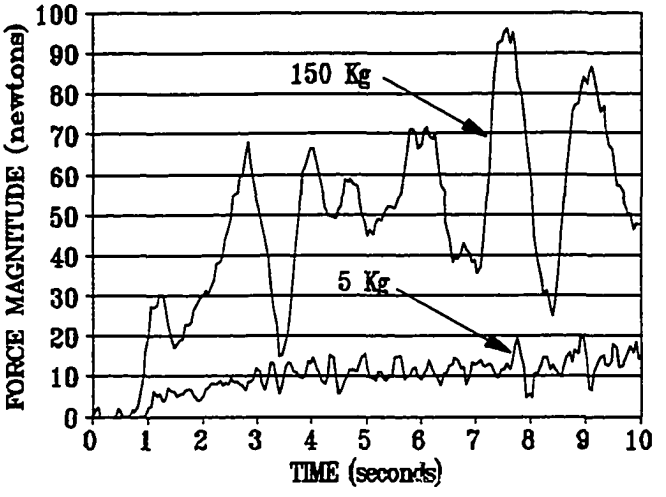

Figure 10: Total force magnitude

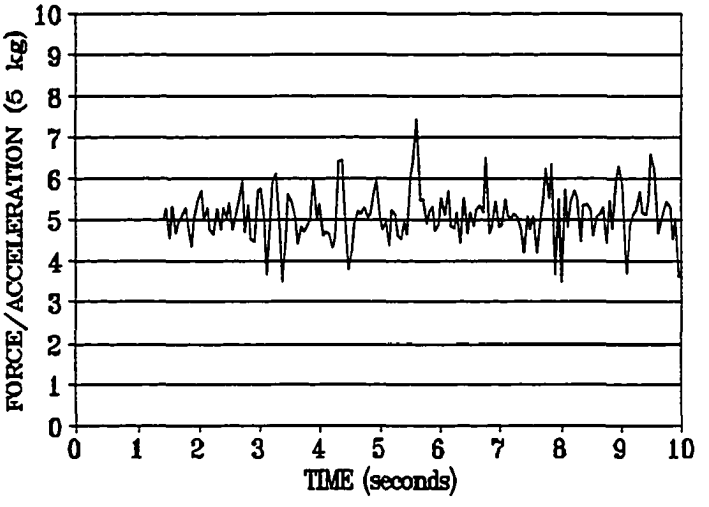

Figure 11: The magnitude of the force vector divided by the magnitude of the acceleration vector for the 5 kg case

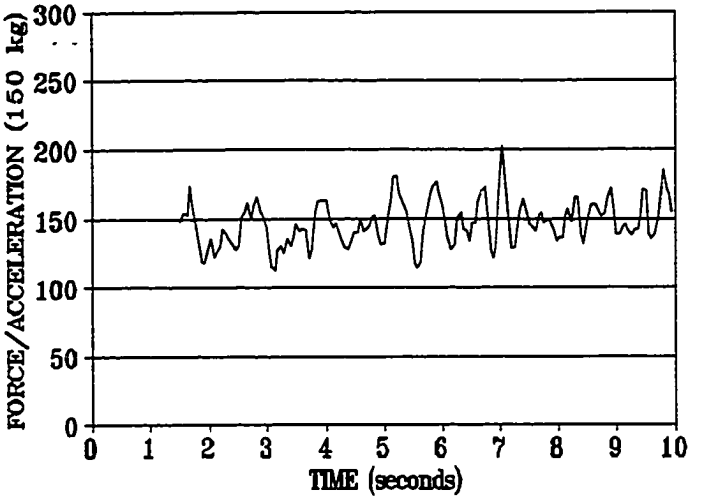

Figure 12: The magnitude of the force vector divided by the magnitude of the acceleration vector for the 150 kg case

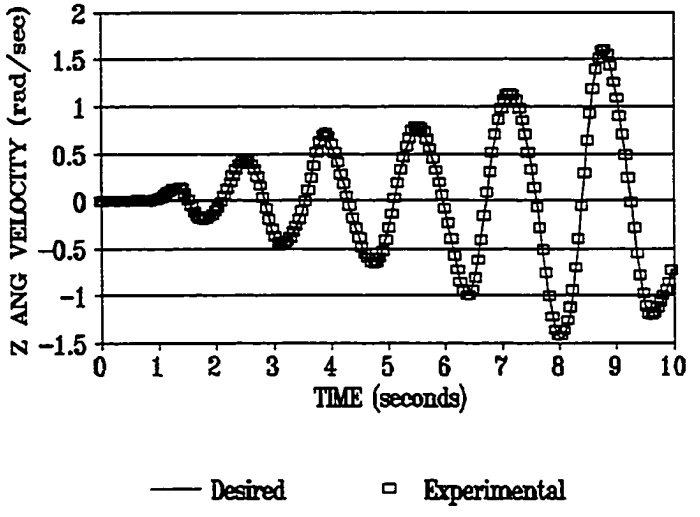

Figure 13: Reference and experimental results for the angular velocity component in the body fixed z-axis direction

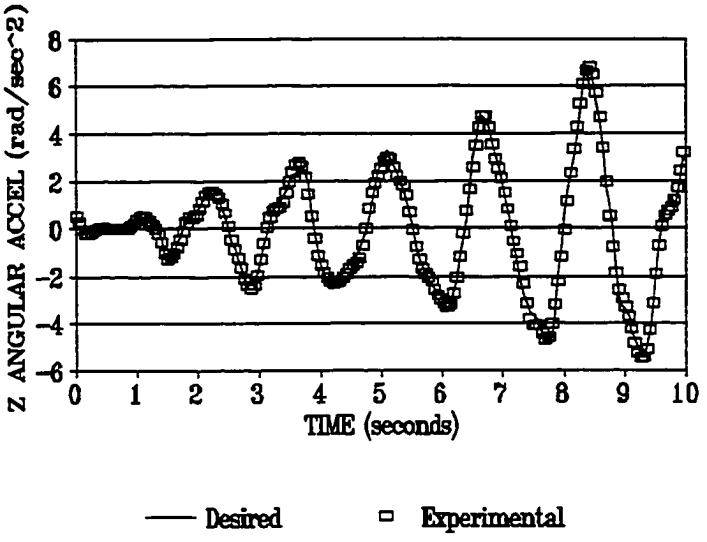

Figure 14: Reference and experimental results for the angular acceleration component in the body fixed z-axis direction

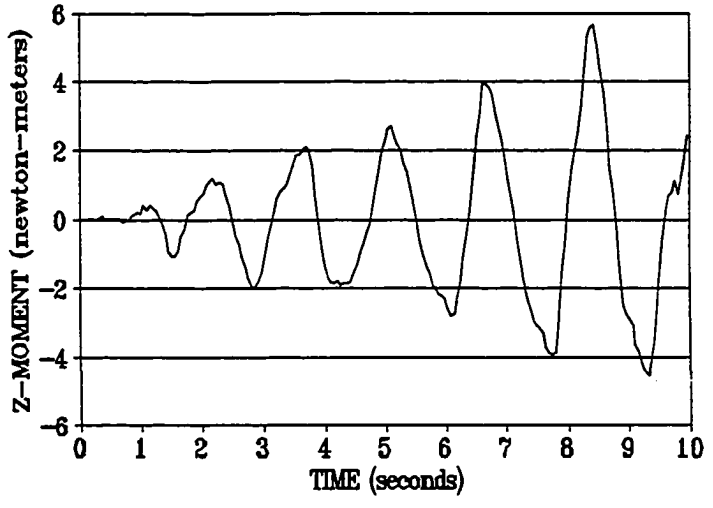

Figure 15: Moment component in the body fixed Z-axis direction

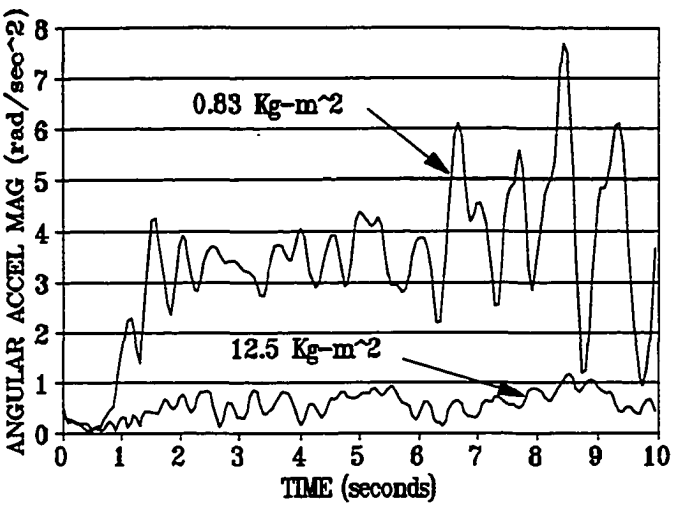

Figure 16: Total angular acceleration magnitude

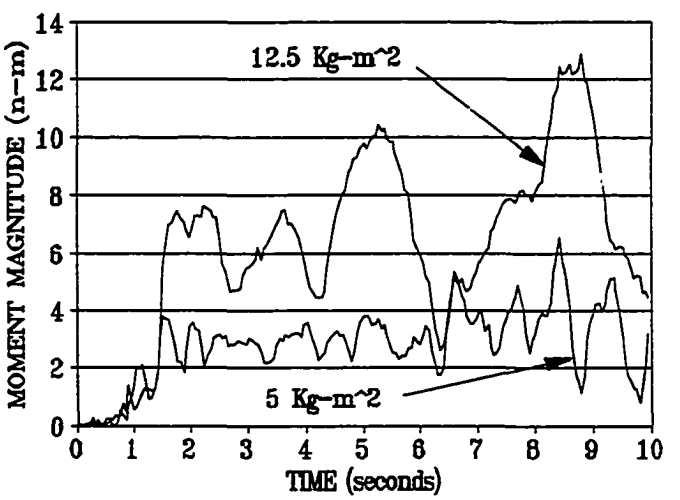

Figure 17: Total moment magnitude

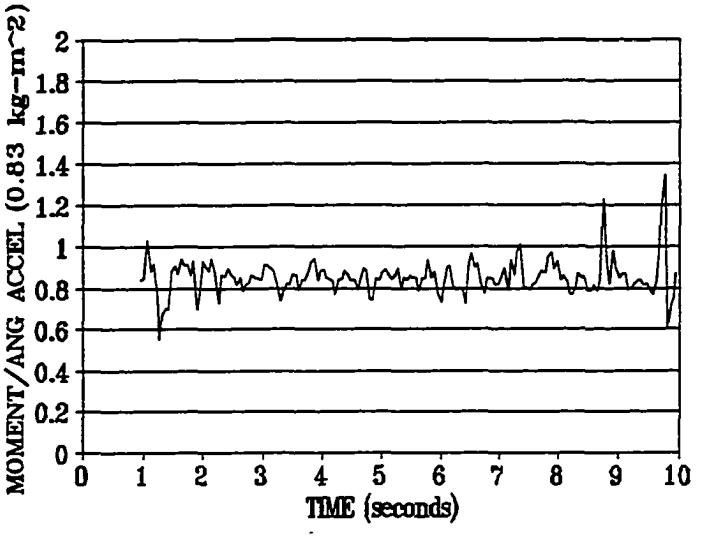

Figure 18: The magnitude of the moment vector divided by the magnitude of the angular acceleration vector for the 0.83 kg-m^2 case

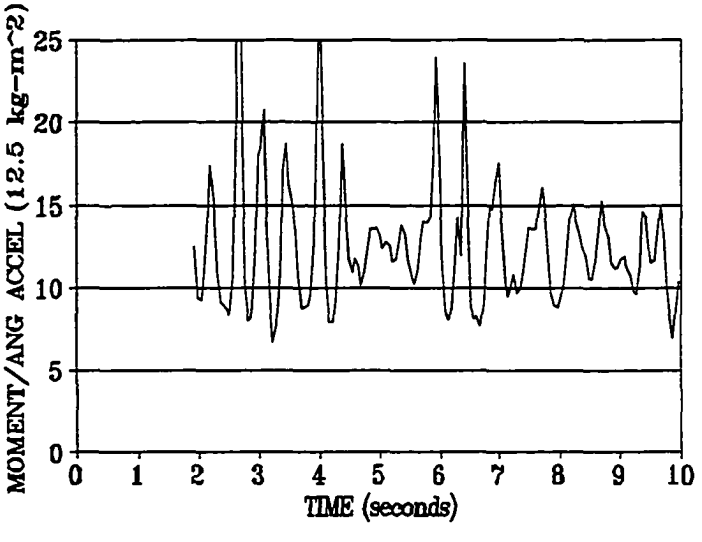

Figure 19: The magnitude of the moment vector divided by the magnitude of the angular acceleration vector for the 12.5 kg-m^2 case

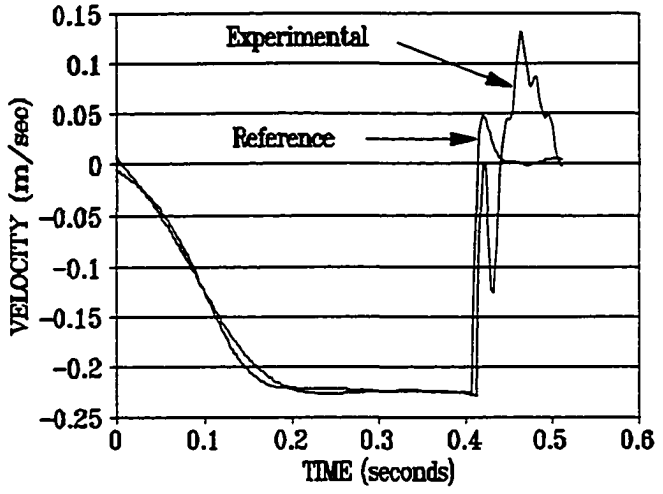

Figure 20: Velocity profile of a 10 kg object's collision with a stiff wall

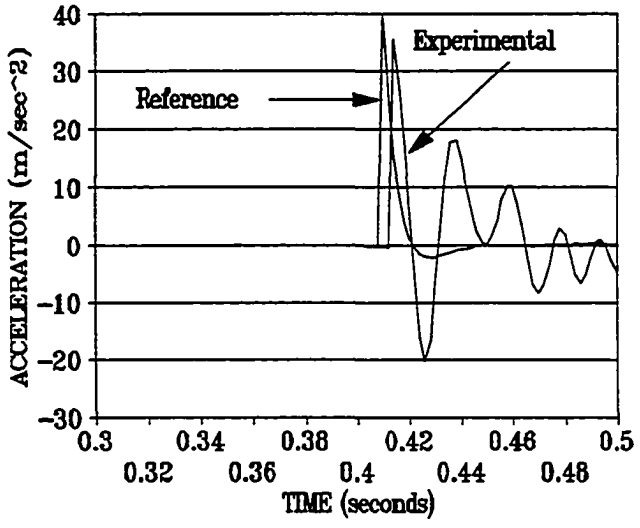

Figure 21: Acceleration profile of a 10 kg object's collision with a stiff wall

Table I: Control system execution times

Function		Time (ms)
Trajectory generation		0.188
Inverse kinematics	(pos,vel,acc)	0.154
Inverse dynamics	(w/o Jacobian)	0.244
Jacobian		0.088
Feedback gains		0.166
Collision detection (two cubes)		0.418
Trident D/A plus encoder reads		0.120
Miscellaneous		0.076

Table II: Statistics for mass and inertia results in Figures 11-12 and Figures 18-19

Case	Mean	Standard Deviation
5 kg	5.099	0.615
150 kg	146.9	15.89
0.83 kg-m^2	0.851	0.085
12.5 kg-m^2	12.65	4.541

APPENDIX

We give, for reference purposes, our Jacobian and transformation matrices. Note that the Jacobian is written with respect to the force transducer coordinate system which is coincident with the joint 6 frame. Although the actual equations used in our software have been optimized to eliminate redundant calculations, un-optimized equations are given here for easier reading (see [55] for further details on efficient Jacobian calculations). Also note that matrix elements not explicitly given are equal to zero.

Jacobian matrix (J^6):

$$J^6_{11} = (d_2+d_3)(c_6c_5c_4s_3s_2-c_6c_5c_4c_3c_2 +c_6s_5s_3c_2+c_6s_5c_3s_2+s_6s_4c_3c_2-s_6s_4s_3s_2) + a_2(s_6c_4c_2+c_6c_5s_4c_2) + a_3(c_6c_5s_4s_3s_2 +s_6c_4s_3s_2-s_6c_4c_3c_2-c_6c_5s_4c_3c_2) + d_4(c_6c_5s_4c_3s_2 +c_6c_5s_4s_3c_2+s_6c_4s_3c_2+s_6c_4c_3s_2+c_6c_5s_4c_2)$$

$$J^6_{21} = (d_2+d_3)(c_6s_4c_3c_2+s_6c_5c_4c_3c_2-s_6s_5s_3c_2 -s_6s_5c_3s_2-s_6c_5c_4s_3s_2-c_6s_4s_3s_2) + a_2(c_6c_4c_2 -s_6c_5s_4c_2) + a_3(s_6c_5s_4c_3c_2+c_6c_4s_3s_2 -s_6c_5s_4s_3s_2-c_6c_4c_3c_2) + d_4(-s_6c_5s_4c_3s_2 -s_6c_5s_4s_3c_2+c_6c_4s_3c_2+c_6c_4c_3s_2)$$

$$J^6_{31} = (d_2+d_3)(s_5c_4s_3s_2-s_5c_4c_3c_2-c_5c_3s_2 -c_5s_3c_2) + a_2s_5s_4c_2 + a_3(s_5s_4s_3s_2-s_5s_4c_3c_2) + d_4(s_5s_4c_3s_2+s_5s_4s_3c_2)$$

$$J^6_{41} = c_6s_5s_3s_2+s_6s_4s_3c_2+s_6s_4c_3s_2 -c_6c_5c_4c_3s_2-c_6c_5c_4s_3c_2-c_6s_5c_3c_2$$

$$J^6_{51} = s_6c_5c_4c_3s_2+s_6c_5c_4s_3c_2+s_6s_5c_3c_2 +c_6s_4s_3c_2+c_6s_4c_3s_2-s_6s_5s_3s_2$$

$$J^6_{61} = c_5c_3c_2-c_5s_3s_2-s_5c_4c_3s_2-s_5c_4s_3c_2$$

$$J^6_{12} = a_2(c_6c_5c_4s_3+c_6s_5c_3-s_6s_4s_3) - a_3c_6s_5 + d_4(c_6c_5c_4-s_6s_4)$$

$$J^6_{22} = -a_2(s_6c_5c_4s_3+c_6s_4s_3+s_6s_5c_3) + a_3s_6s_5 - d_4(s_6c_5c_4+c_6s_4)$$

$$J^6_{32} = a_2(s_5c_4s_3-c_5c_3)+a_3c_5+d_4s_5c_4$$

$$J^6_{42} = c_6c_5s_4+s_6c_4$$

$$J^6_{52} = c_6c_4-s_6c_5s_4$$

$$J^6_{62} = s_5s_4$$

$$J^6_{13} = -a_3c_6s_5+d_4(c_6c_5c_4-s_6s_4)$$

$$J^6_{23} = a_3s_6s_5-d_4(s_6c_5c_4+c_6s_4)$$

$$J^6_{33} = a_3c_5+d_4s_5c_4$$

$$J^6_{43} = c_6c_5s_4+s_6c_4$$

$$J^6_{53} = c_6c_4-s_6c_5s_4$$

$$J^6_{63} = s_5s_4$$

$$J^6_{44} = -c_6s_5$$

$$J^6_{54} = s_6s_5$$

$$J^6_{64} = c_5$$

$$J^6_{45} = s_6$$

$$J^6_{55} = c_6$$

$$J^6_{66} = 1.0$$

The transformation matrices of each link with respect to the previous link are given by:

$$T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ 0 & 0 & -1 & -d_4 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & -c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where,
$s_i = \sin \theta_i$
$c_i = \cos \theta_i$
$d_2 = 0.2435$ meters
$a_2 = 0.4318$ meters
$d_3 = -0.0934$ meters
$a_3 = -0.0203$ meters
$d_4 = 0.4331$ meters

V. GENERAL CONCLUSIONS

This dissertation presents research which advances the state of the art in haptic interface control system development. Chapter II of this document examined the use of abbreviated, symbolic modeling of the PUMA 560. It showed that abbreviated models yield inverse dynamic calculations that maintain a high level of agreement with the un-abbreviated model. The utility of abbreviated linear models was also studied. This turns out to be a more challenging task. Engineering judgment is required to determine the degree to which the abbreviated and un-abbreviated models should agree in areas such as trajectory sensitivity analysis or optimal path planning. These judgments are application dependent (ie, high speed vs. low speed). This dissertation was mainly concerned with control system design and Chapter II found that the abbreviated linearized PUMA model presented here is satisfactory. The computational savings in using abbreviated models, both linear and non-linear, is substantial and important in real-time applications.

The results of this dissertation are parameter, and therefore manipulator dependent. However, intuition tells us that these conclusions probably are valid for many industrial robots. Future work should involve deriving abbreviated linear and non-linear models for other robots to verify whether or not this is the case.

Chapter III demonstrates the utility of symbolic processing software used to quickly generate complex robotic control software. In particular, a feedback controller has been derived based on time varying trajectory linearization about a nominal operating point. By using abbreviated symbolic models, fewer than 700 hundred calculations are required for both the feedforward and feedback loops, a computational burden easily handled in fast sampling, real-time systems with modern computers. Straightforward eigenvalue analyses have been

used to demonstrate stability and performance robustness in the presence of bounded modeling errors and disturbances.

This chapter also compares three computed torque controllers (all derived automatically with symbolic processing software) in an experimental test bed using a previously defined standardized test trajectory which exercises a PUMA 560 through much of its dynamic range. Simulation and test results are given for all six PUMA degrees of freedom. Eigenvalue analysis indicates that, for the PUMA, system performance when using traditional computed torque schemes versus those based on trajectory linearization will converge in the limit as gains are increased or as speed and acceleration are decreased. This result is born out experimentally as well.

A final analysis indicates that computed torque methods, as a whole, perform quite well. This well known result has been verified in the past by others. However, a controller based on trajectory linearization as presented in this paper does not demonstrate significant performance improvements when used with a standard PUMA 560 configuration. For instance, the PUMA in our lab is fuse limited such that its peak motor torque capabilities are halved. This, coupled with motor back emf, limits the maximum joint velocities. Controllers based on trajectory linearization will show performance improvements as speed is increased. Direct drive systems will also see improvements as well because inertial dynamics generally decrease in relative magnitude compared with Coriolis and centrifugal dynamics. However, this remains to be seen as future work incorporates this type of controller in other robotic systems.

Chapter IV of this dissertation demonstrates the feasibility of using off the shelf robotics equipment for simulating dynamic force and moment interactions between humans and virtual

objects. A PUMA 560 was used in an experimental test bed along with visualization hardware and software to test the control system architecture presented here. Feedforward and feedback control loops were used to compensate for the PUMA's inertia and friction. Equations have been derived which efficiently compute the dynamic trajectory of the simulated six degree of freedom object and which convert this Cartesian space trajectory into joint space inputs for the feedforward and feedback loops.