

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.



University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 1350765

Payload adaptive control of a flexible manipulator using neural networks

Askew, Craig Steven, M.S.

The University of Arizona, 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

**PAYLOAD ADAPTIVE CONTROL OF A FLEXIBLE
MANIPULATOR USING NEURAL NETWORKS**

by
Craig Steven Askew

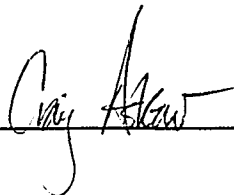
A Thesis Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING
In the Graduate College
THE UNIVERSITY OF ARIZONA
1992

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____



APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:



Malur K. Sundareshan
Professor of
Electrical and Computer Engineering

9/23/92

Date

TABLE OF CONTENTS

LIST OF FIGURES	5
LIST OF TABLES	7
ABSTRACT	8
1. INTRODUCTION	9
1.1. Rigid vs. Flexible Manipulators	9
1.2. Control Methods for Flexible Robots	11
1.2.1. Classical Control	12
1.2.2. Computed Torque Method	12
1.2.3. Optimal Control	13
1.2.4. Singular Perturbation Control	14
1.2.5. Adaptive Control	15
1.2.6. Nonlinear Control	17
1.3. Neural Networks	18
1.3.1. Artificial Neural Networks	19
1.3.2. Neural Networks in Control	26
1.4. Contributions of this Thesis	28
1.5. Organization of Thesis	30
2. MODELING AND SIMULATION	32
2.1. Introduction	32
2.2. Beam Theory and Assumptions	36
2.3. Development of Kinematic Equations	38
2.3.1. Hamiltonian	38
2.3.2. Nonlinear Euler Equations and Boundary Conditions	44
2.3.3. Non-dimensional Euler Equations	48
2.4. Mode Shapes and Modal Frequencies	51
2.4.1. Orthogonality of Mode Shapes	53
2.5. Modal Expansion of Nonlinear and Linear Models	55
2.6. Observability and Controllability of the Linear Model	59
2.7. Simulation using MATLAB	62

2.8. Results and Numerical Difficulties	64
2.9. Pulse Response	66
3. NEURAL NETWORK CLASSIFICATION APPROACH FOR PAYLOAD ADAPTIVE CONTROL	73
3.1. Introduction	73
3.2. Control Methodology	75
3.2.1. Control Objectives	76
3.3. Identification of Payload	83
3.4. Neural Network Architecture and Training	90
3.4.1. Multilayered Static Neural Networks	90
3.4.2. Backpropagation Learning Algorithm	93
3.4.3. Network Training for Payload Identification	98
3.5. Performance of Neural Network-Based Control	100
3.6. Implementation Issues	111
4. ADAPTIVE VARIABLE STRUCTURE CONTROL BY PAY- LOAD IDENTIFICATION	114
4.1. Introduction	114
4.2. Variable Structure Control	115
4.3. Some Previous Results using VSC for Flexible Manipulators	124
4.3.1. Tip Position VSC	124
4.3.2. Hub Rotation plus Pole Placement Approach	128
4.3.3. VSC State Regulator Method	135
4.4. Payload Identification Plus VSC	145
5. CONCLUSIONS	158
5.1. Introduction	158
5.2. Summary of Results Reported in this Thesis	158
5.3. Directions for Further Research	161
REFERENCES	164

LIST OF FIGURES

1.1. Sigmoidal Function Transfer Characteristic	21
2.1. Euler-Bernoulli and Timoshenko motion of the transverse section .	37
2.2. Single-Link Flexible Manipulator	39
2.3. Tip Deflection due to Pulse Input, No Payload	67
2.4. Hub Rotation due to Pulse Input, No Payload	68
2.5. Tip Deflection due to Pulse Input, Payload: $\mu = \kappa = 0.5$	68
2.6. Hub Rotation due to Pulse Input, Payload: $\mu = \kappa = 0.5$	69
2.7. Tip Deflection Comparison, 1 Flexible mode (—) vs. 4 Flexible Modes (-), Payload: $\mu = \kappa = 0.5$	70
2.8. Maximum Tip Deflection with respect to Payload	71
2.9. Maximum Tip Deflection with respect to Payload (Enlarged Scale)	72
3.1. Tip Deflection of Flexible Manipulator M_0	79
3.2. Hub Rotation of Flexible Manipulator M_0	80
3.3. Tip Deflection of Flexible Manipulator $M_{1.5}$	80
3.4. Hub Rotation of Flexible Manipulator $M_{1.5}$	81
3.5. Tip Response Under Test Control	85
3.6. Mean Value of Tip Deflection vs. Maximum Tip Deflection, $\kappa = \mu$	87
3.7. Mean Value of Tip Deflection vs. Maximum Tip Deflection, $\kappa = \frac{1}{2}\mu$	88
3.8. Static Multilayer Neural Network	91
3.9. Nodal Connections in a Static Network	92
3.10. Neural Network-Based Control System	100
3.11. Class 1 Range of Tip Deflection	101
3.12. Class 1 Range of Hub Response	101
3.13. Class 2 Range of Tip Deflection	102
3.14. Class 2 Range of Hub Response	103
3.15. Tip Deflection - No Payload	104
3.16. Hub Rotation - No Payload	104
3.17. Tip Deflection - Payload $\mu = \kappa = 0.4$ (Class 1)	109
3.18. Hub Rotation - Payload $\mu = \kappa = 0.4$ (Class 1)	109
3.19. Tip Deflection - Payload $\mu = \kappa = 1.2$ (Class 2)	110
3.20. Hub Rotation - Payload $\mu = \kappa = 1.2$ (Class 2)	110

3.21. Time Sequence for Practical Implementation	112
4.1. Tip Deflection, VSC Hub Regulator Control	122
4.2. Hub Rotation, VSC Hub Regulator Control	122
4.3. Sliding Line Trajectory, VSC Hub Regulator Control	123
4.4. Phase Plane Trajectory, VSC Hub Regulator Control	123
4.5. Tip Deflection, VSC Tip Position Approach	126
4.6. Hub Rotation, VSC Tip Position Approach	127
4.7. Phase Plane Trajectory, VSC Tip Position Approach	127
4.8. Tip Deflection, VSC + Pole Placement Approach	132
4.9. Hub Rotation, VSC + Pole Placement Approach	133
4.10. Phase Plane Trajectory, VSC + Pole Placement Approach	133
4.11. Tip Deflection under VSC control, No Payload	141
4.12. Hub Rotation under VSC control, No Payload	142
4.13. Sliding Line Trajectory, No Payload	142
4.14. Tip Deflection of Arm: Class 0 Payload, NNVSC	146
4.15. Hub Rotation of Arm: Class 0 Payload, NNVSC	146
4.16. Tip Deflection of Arm: Class 1 Payload($\mu = 0.4$), NNVSC	147
4.17. Hub Rotation of Arm: Class 1 Payload($\mu = 0.4$), NNVSC	148
4.18. Tip Deflection of Arm: Class 1 Payload($\mu = 0.4$), Class 0 NNVSC	148
4.19. Hub Rotation of Arm: Class 1 Payload($\mu = 0.4$), Class 0 NNVSC	149
4.20. Sliding Mode of Arm: Class 1 Payload($\mu = 0.4$), Class 0 NNVSC	149
4.21. Tip Deflection of Arm: Class 2 Payload($\mu = 1.2$), NNVSC	151
4.22. Hub Rotation of Arm: Class 2 Payload($\mu = 1.2$) NNVSC	152
4.23. Tip Deflection of Arm: Class 2 Payload($\mu = 1.2$) Class 0 NNVSC	152
4.24. Hub Rotation of Arm: Class 2 Payload($\mu = 1.2$) Class 0 NNVSC	153
4.25. Sliding Mode of Arm: Class 2 Payload($\mu = 1.2$) Class 0 NNVSC	153
4.26. Tip Deflection Trajectory	156
4.27. Hub Rotation Trajectory	157
4.28. Tip Position Trajectory	157

LIST OF TABLES

2.1. Parameters of the CIRSSE Single-Link Flexible Manipulator	65
2.2. Modal Frequency Comparison with CIRSSE Arm	65

ABSTRACT

Flexible manipulators provide significant advantages over the commonly-used rigid robots due to their lightweight properties, but an accurate control of these manipulators is more difficult to attain, and it is especially demanding in task executions involving changing payloads. This thesis addresses the problem of payload adaptive control of flexible manipulators.

The nonlinear model describing the manipulator dynamics is completely derived and is then used for an accurate computer simulation of the flexible manipulator motions. Payload identification is implemented by using a novel neural network approach to identify distinct payload classes from tip deflection patterns which result from different payloads. The identification procedure is then used to select a controller which best meets the control objectives specifying hub speed and maximum tip deflection. Two distinct controller synthesis procedures, one using a pole-placement design and one employing a variable structure technique, are developed. The merits of payload adaptive control are shown by several simulation experiments .

CHAPTER 1

INTRODUCTION

1.1 Rigid vs. Flexible Manipulators

Throughout modern times, industry has continually sought to increase automation as a means to improve product quality and productivity. This trend shows no signs of slowing, and in the past decade, manipulators and robots have been implemented on a large scale in some industries. Automation utilizing robotics has been shown to be successful in many industrial applications in improving the productivity and quality of industry, but there are still some major drawbacks to the types of manipulators being widely used today.

One of the major shortcomings of the robots in use today is their excessive weight. In order to ensure a high precision of movement and positioning accuracy, strict rigidity is a required characteristic of the links and joints, and the only way to satisfy this requirement is to make the robot very heavy. As a result of the large weight, larger actuators are required to move them, which in turn consume more power.

If the links were made of lightweight materials such as aluminum alloys or composite materials, the weight of the robot could be significantly reduced. Manipulators composed of these materials enjoy many advantages over rigid robots, including the following:

- Higher speed
- Smaller actuators
- Lower energy consumption
- Safer operation due to reduced inertia since the beam is lighter
- Lower overall cost
- Lower overall mass to be transported (especially valuable for space applications)
- Lower mounting strength requirements.

While the above advantages make lightweight or flexible manipulators attractive, the main drawback in real world applications is the flexibility introduced by the lightweight construction. The use of aluminum alloys or composite materials results in links which are strong in compression, but relatively weak in bending. Therefore, in order to achieve a high positioning accuracy, more complicated controls must be used, and the design of such controls is by no means a simple procedure.

In order to investigate the problems arising in flexible manipulator control and modeling, several mechanical models have been constructed. Some of the more well-known ones are:

1. MIT FLEXBOT - Department of Mechanical Engineering, MIT
2. ARMA - Department of Electrical Engineering, Ohio State University
3. RALF - Department of Electrical Engineering, Georgia Institute of Technology
4. Stanford Flexible Arm - Department of Aeronautics and Astronautics, Stanford University
5. CIRSE - NASA Center for Intelligent Robotic Systems for Space Exploration, Rensselaer Polytechnic Institute
6. Colorado State Univ. Flexible Manipulator - Department of Electrical Engineering, CSU.

These physical models have proven to be valuable testbeds for modeling, system identification and controller design and evaluation.

1.2 Control Methods for Flexible Robots

Efforts directed to the design of controls for flexible manipulators have increased considerably in the past few years. Both modern and classical control techniques

have been applied to the flexible manipulator, some with more success than others. While there exist many methods for control, only some of the more popular techniques are discussed here in the interests of brevity.

1.2.1 Classical Control

In their pioneering work in 1984, Cannon and Schmitz [14] addressed the problem of noncollocated sensors and actuators which is inherent with flexible manipulators. They showed that because the sensor and the actuator are located at opposite ends of the manipulator (and thus noncollocated), the simple classical PD and PID design approaches that work well for collocated systems cannot be applied without large vibration resulting, which consequently produces large positioning errors of the end-effector.

A similar conclusion was arrived at by Shung and Vidyasagar [60] for the conventional PI controller. Stabilization was not achieved with this method, and it was concluded that the noncollocated nature of the system was the reason.

1.2.2 Computed Torque Method

The computed torque method is a technique that is popularly used on rigid manipulators, and its extension to the case of flexible robots was conducted by De Luca et al. [18]. A dynamic inverse system was designed for use with both full order and reduced order models for open loop control, and closed loop control

using PD control in conjunction with the computed torque method was given. A simulation case study of a simplified one-link flexible arm showed the effect of the choice of output variables on the stability of the closed-loop system and on the overall tracking performance.

1.2.3 Optimal Control

A linear quadratic Gaussian regulator (LQG) was first used by Cannon and Schmitz [14] on flexible manipulators. This control minimized a performance index which was based on the tip position and the torque applied. For the design of the LQG regulator, it was assumed that the plant was perturbed by a white noise process input, and an optimal observer was designed as well to reconstruct the states for feedback control. A similar approach was used by Lee [37], except that unlike [14], a nonlinear model was considered and the effect of gravity was modeled, as the arm was operated in a vertical plane rather than a horizontal one.

Another form of optimal control is the so-called H_2 optimal control, which uses a more accurate norm (the H_2 norm) to model the disturbances of the system, rather than using the white noise assumption of the LQG compensator. Vinke and Vidyasagar [70] designed a controller of this type which minimized the mean squared tracking error, while being constrained to a bounded input. This controller was realized with a stable factorization approach using a Lagrangian multiplier in

conjunction with H_2 optimization. Promising results were found for two different types of models.

Another approach, this time using H_∞ control, was investigated by Lin [39]. This approach uses a two-stage control; first, the rigid body dynamics are controlled with a partial feedback linearization control towards a desired state, and then when in the vicinity of this state, a robust stabilization control is activated to suppress elastic vibrations. The damping of the vibrations is achieved by H_∞ control, which is similar to the H_2 control except that a different norm is used to model the disturbances.

1.2.4 Singular Perturbation Control

The singular perturbation technique is dependent on decoupling the system into a “fast” system, composed of high frequency dynamics, and a “slow” system of slower dynamics which dominate the overall system dynamics. This technique is especially applicable to flexible manipulators, which are distributed parameter systems containing an infinite number of high frequency modes.

Siciliano and Book [61] used this technique by decoupling the system into one composed of the rigid mode (slow system) and the flexible modes (fast system). The convenience of this approach is that well defined procedures used on rigid robots can be used to control the slow system. The flexible modes are damped using an optimal control with the slow state variables acting as parameters of the performance index.

A more recent application of singular perturbation control [59] used an interesting actuator in order to dampen the oscillations of the link. The slow system variables were controlled in the same manner as [61], but the fast system vibrations were damped using distributed control. This distributed control was realized using a piezoelectric polymer film actuator which induced a strain along the longitudinal axis of the link to dampen the vibrational modes when a voltage was applied to it.

1.2.5 Adaptive Control

Adaptive control can be divided into two areas [46]: direct adaptive control and indirect adaptive control. In direct adaptive control, the controller parameters are directly adapted in a manner which minimizes or improves a specified performance index for the system. Indirect adaptive control is realized by updating control parameters based on plant parameters which are estimated on-line. There have been applications of both of these types of adaptive control to flexible manipulators.

Direct adaptive control has seen the most application using model reference adaptive control (MRAC) techniques, in which it is desired to match a plant output with that of a reference model. Sasiadek and Srinivasan [58] used a MRAC scheme to make a one-link flexible robot follow a reference model which was a double-integrator controlled by a simple PID controller. A nonlinearity compensation and decoupling control had parameters which directly corresponded to plant parameters

adjusted by the adaptive scheme. PID control was used in conjunction with the adaptive control to further stabilize the system, but the PID gains were fixed.

Yuh [77] presented a discrete MRAC technique in which the flexibilities of the manipulator were modeled as disturbances on a rigid link system. Performance results were developed for both collocated and noncollocated systems, which once again showed the unsatisfactory performance of collocated sensor-actuator systems in damping tip vibrations. In both [77] and [58], no payload changes were simulated in the results.

Most adaptive schemes which account for payload changes utilize indirect adaptive techniques, usually with separate identification techniques coupled with self-tuning regulators. Rovner and Cannon [55] used a Recursive Least Squares (RLS) scheme to identify the parameters of the transfer function of the flexible arm and then once the convergence of the parameters was complete, regulator gains were calculated using LQG optimal control theory. Several different payloads were tested with success, but the payload was never changed while in a continuous motion. This problem was addressed by Yang et al. [75] who also compensated for transients induced when a payload was released. The simulations showed successful results, but the arm was moved very slowly during these. Other papers, notably [78], present faster frequency domain identification techniques, but these do not address the adaptive control of flexible robots.

1.2.6 Nonlinear Control

Since the flexible manipulator is characterized by highly nonlinear dynamic equations, nonlinear control techniques are desirable in order to achieve more accurate control than is possible with traditional linear techniques. Lee and Castelazo [36] presented a nonlinear scheme using a multiplication of state variables. An iterative design procedure was outlined to reduce the overshoot and the settling times resulting from corresponding linear methods by using this state variable coupling feedback. Simulation results showed that these performance quantities were indeed improved when compared to linear controls. A procedure for designing nonlinear pole placement regulator and observer was outlined by Nicosia et al. [48] in 1989, and simulations showed the effectiveness of the controller/observer system.

In the nonlinear method described in [40], the manipulator system dynamics was first transformed into error-driven equations by nonlinear feedforward and PID feedback control. The system was then stabilized using the second method of Lyapunov. The control was tested on a robot modeled with flexible links and on one with flexible joints.

Another form of nonlinear control is variable structure control (VSC) which is very popular due to the robustness properties resulting from the use of it. In this type of control, the nonlinearity is introduced by a controller which switches between two or more distinct structures to force the states of the system towards a hypersurface known as a sliding manifold. The hypersurface on which the states

are desired to reside is designed such that the system dynamics are asymptotically stable. Applications of this technique for flexible manipulators has been shown in [52, 47, 62]. A more detailed examination of these papers and design procedures of VSC will be presented in Chapter 4, as the research described in that chapter is concerned with the use of VSC as well.

1.3 Neural Networks

In the continuing quest for the improvement of robotic operation, it has always been a goal to approach human performance in varying degrees. The reason for this is simple. When a robot is controlled, it is done classically by deriving the differential equations which govern the motion of the arm and then using these equations to solve for the required torque to place the end-effector at the desired location. On the other hand, when a human wishes to place his or her hand at a location, no solution of the dynamical equations are needed. Although the action might be less precise, it is much more adaptive and successful.

The reason for the success of the brain over that of a serial computer is mainly due to the structural differences between the two. The brain contains a massively connected structure of neurons which can be thought of as processing elements. This parallelism has many advantages over the serial connection of most computers today. It allows humans to use the great amount of sensing information to respond quickly to different situations despite the fact that the delay time of the processing elements

of the brain has been shown to be much slower than that of semiconductor devices [1]. Another advantage is that humans perform control mainly through learning and adaptation. These advantages have sparked interest in models of the human processing elements, called neural networks. In order to model these biological neural networks, a closer look at the underlying physiology is required.

Studies of the anatomy of the brain have led to estimations that the human brain has over 10^{11} neurons. Each of these neurons receives information from other neurons via a matrix of connection weights called synapses. There exist more than 1000 of these at the input and the output of each neuron, showing the parallel nature of the processing in the brain. The output of each neuron is realized by an output fiber called an axon, which sends impulses called action potentials along to other neurons. The function of this network can be changed or adapted by varying the weights of the synapses which determine the criteria of the neuron to fire (or output an action potential) or not. This process is known as learning the synapse weights.

1.3.1 Artificial Neural Networks

Networks whose purpose is to mimic the function of the biological neural networks of the brain are known as artificial neural networks, or simply neural networks as they are referred to in this thesis. The structure of a neural network can be characterized by processing elements called nodes which are usually nonlinear in nature

and are organized in layers which are connected in a parallel fashion. The entire network may be made up of several layers, each containing many nodes connected to one another.

Processing nodes can be categorized as either static or dynamic. For static nodes, an algebraic equation such as

$$y_i = f_i\left(\sum_{j=1}^n w_{ij}x_j\right) \quad (1.1)$$

where y_i is the output of node i of the present layer, w_{ij} is the interconnection weight associating the output, x_j , of node j of the previous layer and node i of the present layer. The function f_i can take many different forms, but most of the time it is a bounded, non-decreasing function such as a sigmoid or a threshold function such as that shown in Figure 1.1. Thus (1.1) merely states that the input signals are weighted and summed and then nonlinearly processed to produce the output of the node, y_i .

A dynamic node is characterized by a differential (rather than an algebraic) equation of the form

$$\dot{u}_i = -a_i u_i + \xi_i(\mathbf{u}, W, \mathbf{b}) \quad (1.2)$$

where a_i represents a connection weight of self-feedback, W is the weight matrix of the nodal connections, \mathbf{b} is a vector of external inputs which could be from another layer, and ξ_i is a nonlinear function of the form discussed earlier in conjunction with (1.1). Thus, in this type of node, the nodal output is continually changing.

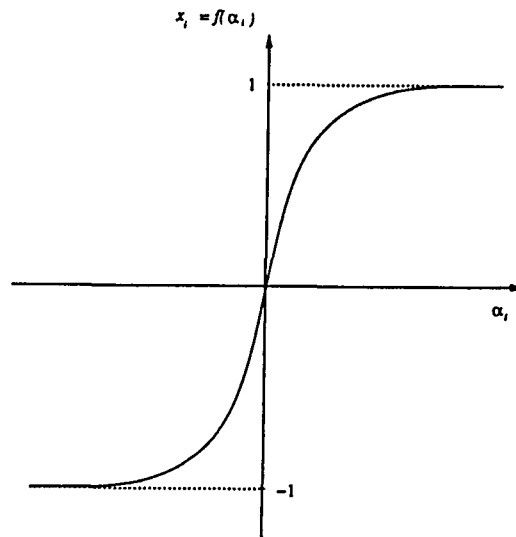


Figure 1.1: Sigmoidal Function Transfer Characteristic

While (1.2) represents a continuous time dynamical model of nodal processing, there are discrete models of this type as well.

The design of a neural network primarily consists of setting the weight values of the interconnection matrix. This can be done by two methods. One is known as synthesis and involves a systematic algorithm of computational steps which is run to arrive at a set of weights which will allow the network to perform a desired function. The other is known as learning. In this method, the network is started with arbitrary initial weight values and is allowed to adapt or learn the correct value of the weights which best satisfies the performance requirements specified for the network. This learning is performed by comparing network outputs with some desired performance criteria and changing the weights of the interconnections if the network is performing inadequately. The advantages of learning are easily seen due

to the adaptability of the network to different environments. As a result, there has been much research in this area during the past few years.

Training procedures to teach the network (or allow it to learn the correct weights) fall into two categories: supervised training and unsupervised training. Brief descriptions of types of neural networks which use each of these training schemes are given in the sections below. For more detailed descriptions, two excellent sources are [49, 41].

Multilayer Static Neural Networks

A multilayer static neural network consists of an input layer, an output layer and a number of hidden layers, which consist of static nodes such as those described by (1.1). The processing of signals is in a feedforward direction and no feedback or recurrent connections are used. Such a network has been shown to be successful in nonlinear function approximation and pattern classification tasks. The training algorithm commonly used with this type of network is known as backpropagation. This is a supervised training scheme which adjusts the weights of the network by comparing the output of the network with the desired outputs and backpropagating this error down through the hidden layers in order to adjust the weights to reduce the total output error. Networks of this type are used in the research reported in this thesis, and a more detailed examination of the network architecture and the backpropagation algorithm will be given in Chapter 3.

Hopfield Networks

Although originally known as an additive model whose aspects have been studied as far back as 1943 by McCulloch and Pitts [43], this model has received more attention since the detailed examination of its characteristics by Hopfield [25]. This type of network was first developed for binary vector inputs as an associative memory, which could generate a true pattern based on the input of the same pattern which had been corrupted with noise or the input of a partial pattern.

In this associative memory application, patterns are stored by setting the interconnection weights according to an outer product rule. The matrix W of weights is usually selected as symmetric with all diagonal elements zero. The nodes of this network are dynamic in nature, and when the weights are properly adjusted, the outputs will converge to a stable state corresponding to the input presented to the network. This stability is shown by Lyapunov analysis of an appropriate function representing the energy of the system. In this way the output of the network can be made to converge to a state which represents the correct pattern for an input which consists of a partial or distorted representation of the same pattern.

Neural networks based on the Hopfield model have found other applications besides that of associative memory. In particular, networks utilizing similar recurrent connections and convergence properties have been shown to have better learning rates [64, 65], and have also been used in the adaptive control of rigid robots [28, 27].

Hamming Networks

The Hamming network is usually constructed of two subnetworks, one of which is a special type known as MAXNET. This subnet computes the Hamming distance between the input pattern and a class exemplar. If the Hamming distance between a pattern x and the exemplar representing class C_j is smaller than the distance to all other class exemplars, then x is deemed to belong to C_j . The Hamming distance used in this algorithm is defined as

$$H(x, u_j) = N - \sum_i (u_{ij} x_i) \quad (1.3)$$

where N is the number of features in the pattern, u_{ij} are the weights of the lower subnet in which the feature values of the class exemplars are encoded (because these weights are determined beforehand, this is considered a supervised network), and x_i are the feature values of the pattern x .

Lippman [41] has shown that this type of network serves as a good classifier, in particular because of its property of boosting the value of the largest output while continually repressing lesser outputs, until finally the only output is that corresponding to the chosen class. Because of this binary type output, this network is used primarily for binary classification applications.

Carpenter-Grossberg Networks

The three previous networks have all used supervised training techniques. An example of a neural network which uses unsupervised training, i.e. no feature or desired output information is explicitly made available to the network, is a Carpenter-Grossberg network which is based on Adaptive Resonance Theory (ART) [15]. This neural network considers the first input it receives as an exemplar representing a class, and the next input that follows is compared to it. If the distance between the two patterns is less than a defined threshold, also called vigilance, the patterns are 'clustered' together in that class. If the distance exceeds the vigilance, that pattern becomes a new exemplar representing a different class. This clustering algorithm is known as the leader algorithm, because if the input "follows the leader" it is clustered with it, otherwise it becomes a "leader" itself.

This network performs well with perfect inputs, but the performance is degraded in the presence of noise. To counter this, Lippman [41] suggests that weights might have to be adapted more slowly and the vigilance or threshold may have to be self-adjusted during the training phase.

Kohonen's Self-Organizing Networks

This neural network model also uses unsupervised training. Once again, the inspiration for this model by Kohonen [32] came from biological nervous systems. In particular, this network is modeled after special characteristics of some nerves,

specifically those nerve cells and fibers in the auditory pathway. These cells and fibers are arranged anatomically such that the position of the nerve cell has a special significance in that it is sensitive to the frequency of sound that is most prevalent in that part of the pathway.

Kohonen's algorithm is for a neural network structure in which all input nodes are directly connected to the output nodes. In addition, output nodes have many interconnections among themselves as well. Continuous-valued input vectors are presented as inputs sequentially in time with no desired output specified. The weights of the network are adjusted such that cluster or vector centers will be specified after a sufficient number of inputs have been presented. The weights also are organized such that outputs are topologically close to physically similar inputs and a natural organization results. This can be a benefit by reducing the complexity of networks with many layers of processing. A particular application of this network is in speech recognition as a vector quantizer.

1.3.2 Neural Networks in Control

To date, there has been little, if any, research in controls using neural networks for flexible manipulators. However, several papers have been presented in recent years which provide applications for rigid robots which show promise in being generalized for use on flexible robotic arms.

Kuperstein and Wang [34] have put forth a design for a neural controller for a one-link rigid manipulator that is general enough to be applied to many different configurations. They have shown that an unsupervised-learning controller could accurately move an unforeseen payload to arbitrary targets with no end-point oscillation. The controller is general enough that it could be adapted to many novel working environments.

Rabelo and Avala [53] used neural networks in a hierarchical control scheme in which one neural system performed the higher-level trajectory planning task, while a second neural system performed the lower-level trajectory control. The trajectory planning was initiated by a network which determined if the desired position was in the workspace of attainable locations by the arm. A second set of networks mapped the possible coordinates of all of the different configurations of the arm which place the end-effector in the desired position. A third network completed the trajectory planning task by choosing the best solution based on the knowledge available from sensor inputs.

The two papers cited above present neural controllers which can theoretically be generalized to flexible manipulators, but a lot of work must be done before this can be implemented in practical situations. One recent paper which has used neural networks in the control scheme does show promise for application to flexible manipulators, though. Leahy, Johnson, and Rogers [35] presented a method which

exploits the efficient pattern classification capabilities of neural networks, in particular static multilayer networks. In their design, the neural network adapted a set of parameters reflecting the mass of the payload in a computed-torque control scheme for a PUMA industrial robot arm. This was performed by having the network identify classes based on the error patterns generated which were unique for different masses of payload. First a specific trajectory was planned and discretized into time intervals, and for each of these intervals one neural network was trained to classify the payload on the basis of the position error inputs for that time frame. This classification was then used to update the controller parameters to achieve better trajectory tracking capabilities. It is clear that such a use of neural networks is much simpler and more easily implementable than most control strategies of the type of [34] or [53]. These aspects will be discussed in greater detail in Chapter 3.

1.4 Contributions of this Thesis

The contributions of this thesis are threefold, and can be summarized as follows:

1. A nonlinear model is developed for the flexible manipulator using the Energy Assumed Modes method. This model is a great improvement over linear models, especially for conducting simulation studies, because nonlinearities present in the physical manipulator are present in the numerical model. A *further improvement in accurately modeling the physical system is obtained by using the exact modal frequencies of the linearized model as a basis for*

expansion of the nonlinear system. This expansion more precisely reflects the actual dynamics of the flexible manipulator than the simpler methods prevalent in the literature which assume that the modal frequencies of the manipulator are equivalent to those of a simpler unrelated problem such as a clamped beam or a hinged beam.

2. A payload identification scheme using a 3-layer neural network trained by backpropagation is developed. This technique classifies the manipulated payload into one of three classes based only on the information concerning the tip deflection. Simulation results are obtained which show the importance of payload information in the realizations of the control objectives of the tip deflection and convergence rates when a regulator control is used.
3. A variable structure control employing an eigenstructure assignment technique is designed for a flexible manipulator and compared to two other VSC techniques and its superiority is shown by simulation. Because VSC has excellent robustness qualities that linear control methods do not, it is used in place of the previously derived regulator control in conjunction with the payload identification scheme. In this way a payload adaptive robust control is synthesized for the flexible manipulator which is insensitive to external disturbances and unmodeled dynamics in addition to being able to adapt the control inputs to reflect the payload being carried at the tip of the manipulator.

1.5 Organization of Thesis

This thesis is primarily concerned with the demonstration of the use of payload identification for flexible manipulators and how control schemes can be adapted using information about the payload. In Chapter 2, the nonlinear dynamic equations of the flexible manipulator are derived. The derivation includes the solution of the exact modes of the linearized system and the modal expansion of the nonlinear integro-partial differential equations to obtain a set of nonlinear ordinary differential equations which are used for simulation. The state space form of the linear model is derived and the important characteristics of controllability and observability are examined. Numerical difficulties encountered when conducting simulations are discussed and solutions to these are presented. A simulation of the flexible manipulator when subjected to a pulse input is shown for both cases of no payload and payload present, and the effects on tip deflection and hub rotation dynamics are delineated.

Chapter 3 concentrates on the use of neural networks to identify classes of payloads being carried by the flexible arm. The tip deflection pattern groupings resulting from the excitation of the modes of the manipulator are shown and a multilayer backpropagation network is trained to recognize these patterns and classify them into one of three payload categories. Control objectives for the arm are presented and a linear pole-placement regulator is designed based on these objectives for each class. The gains which correspond to the class identified by the neural network are

implemented in order to drive the states of the arm to zero. Design considerations for the placement of the poles and how they correspond to the control objectives are discussed and examples are given. Finally, simulations are presented to show the advantages gained when the control corresponding to the correct payload class is used, supporting the need for payload identification for effective control.

In Chapter 4, variable structure control methods are explored in an attempt to provide a more robust control to be used with the payload identification employing neural networks. Three distinct methods are tried and compared, and a method which elegantly regulates all states simultaneously is shown to be a superior method. Finally, the use of the payload identification scheme in conjunction with VSC for all payload classes is presented and the results are discussed.

The thesis is concluded in Chapter 5 with a summary of the specific contributions of the techniques presented and a brief outline of possible directions for further research into this problem.

CHAPTER 2

MODELING AND SIMULATION

2.1 Introduction

The first step in any computer simulation of a dynamical system is obviously to find the mathematical description of the system dynamics. This description need not be a unique one; indeed there could exist many different mathematical models, all describing the same input-output characteristics of a particular physical process. The modeling of the one-link flexible manipulator is no different in this aspect.

There are two major categories of modeling techniques for flexible robots. One is the *Finite Element Method* and the other is the *Energy Assumed Modes Method*. The Finite Element Method (FEM) is so-named because each link is expressed as a number of elements, each with a certain degree of freedom. The Energy Assumed Modes (EAM) Method gets its title from the fact that it is based on an energy method such as *Hamilton's* or *Lagrange's Principle* which yields an integro-partial differential equation, and is simplified to an ordinary differential equation by expanding the equation with assumed modes.

The Finite Element Method calculates the inverse and forward dynamics of the arm to provide the solutions for torque inputs to generate desired outputs. While this is more easily generalized to multi-link flexible manipulators, it provides less physical insight and intuition in the design of the control to meet specified objectives. The FEM technique was first used in modeling lightweight robots by Book [11] in 1982. Early frequency domain techniques [7, 9] were too computationally intensive to be used in the synthesis of control laws. Bayo and Moulin [8] developed a more efficient real-time algorithm in 1989.

Underlying the assumed modes technique is another choice in the model derivation. There are generally two options: the assumed modes of the model are based on generic vibrational modes derived from a related simpler problem, usually a cantilever beam, or the assumed modes are based on the exact vibration modes of the linearized model of the single link arm.

The cantilever beam vibration modes are derived from one of two conditions. The beam is either clamped, which means that the hub of the real system is assumed to have infinite inertia, or the hub is pinned, which corresponds to a hub in the real system which is assumed to have no inertia. It is obvious that the hub of the manipulator has finite inertia, and the real system is somewhere in between the case of a clamped beam and a pinned beam. For this reason, the assumed modes chosen in this manner will never satisfy the actual boundary conditions or the governing differential equation of the manipulator. Models based on either of these assumed

modes are thus only approximations, and a comparative study by [3] concluded that the true behavior of the beam is in fact between these two approaches.

On the other hand, the exact vibration modes allow a simpler orthogonal relationship to be derived, which simplifies the final form of the dynamic model. Although it requires more work to derive these exact modes, this is done off-line. Once the model is implemented for either simulation or control, the computational burden is much less than using the assumed modes based on a cantilever beam.

Since the assumed modes method theoretically needs an infinite number of modes to give full accuracy, the issue of how many modes are needed for effective control becomes an important one. Krishnan and Vidyasagar [33] have shown that control can be effective using a low order finite set of modes.

A comparison of both modeling techniques (FEM and EAM) for a two-link flexible robot arm was conducted by Hohenbicler, Plöckinger and Lugner [24]. They concluded that while the accuracy of the FEM is better for more elements or links, the calculation time for this method was slower by a factor of more than 100 than that of the EAM technique. For this reason, the finite element method has been suggested for use as a method of testing the quality of other modeling techniques, and not as the preferred technique for simulation.

Many researchers have used the assumed modes technique for modeling flexible manipulators, including [14] and [23]. Cannon and Schmitz were the first to use this approach, and their work is regarded as the classical study of the single-link

flexible manipulator. Of particular interest are those which use the exact modes [10, 71, 73], and these provide a more accurate and simpler representation of the system dynamics.

In addition to deriving the exact solution of the linearized vibration modes and giving the orthogonality condition between the modes based on the exact linearized modal frequencies, Wang [73] has developed a non-dimensional model which describes the nonlinear dynamics of a one-link flexible manipulator. The non-dimensionality feature is attractive because it greatly simplifies the final dynamic equation describing the system and allows one to see the relationships between the beam parameters and those of the payload easily. By including the nonlinear terms, this model will give a more accurate representation of how the real flexible manipulator will respond under different controls. The aim of every model to be used for simulation purposes is to be as accurate as possible, and therefore the model derived in this thesis will follow this derivation closely.

This chapter shows in detail the development of the dynamical equations describing the motion of a single-link flexible manipulator. First the Hamiltonian is developed from the expressions for kinetic energy, potential energy and work. The governing Euler equations and boundary conditions are found from the Hamiltonian and the vibrational or modal frequencies are calculated from these. The nonlinear model is expanded along a finite truncated set of these modal frequencies, yielding

the ordinary differential equations which describe the state-space model of the manipulator. This model is then simulated using the software package MATLAB, and the pulse response of the system is explored.

2.2 Beam Theory and Assumptions

The mathematical model of the single-link flexible arm relies heavily on beam theory to describe its dynamics, as one would expect. There are two major beam theories in mechanics, known as the *Euler-Bernoulli Theory* and the *Timoshenko Theory*.

In Euler-Bernoulli theory, the *Normal Plane Assumption* is used. This assumption states that the complete transverse section of the beam, plane and normal to the longitudinal axis of the beam before deformation, remains plane and normal after deformation as well. Thus any twisting that might occur during deformation, known as shear deformation, is completely neglected using this assumption.

The Timoshenko theory doesn't use the Normal Plane Assumption; instead it uses the more accurate *Plane Assumption*. The Plane Assumption states that a transverse section that is normal and plane before deformation remains plane but not normal to the longitudinal axis after deformation [73]. Figure 2.1 shows the differences between the two theories. The figure shows the cross-section of the beam, one using Timoshenko theory to model its dynamics, the other using Euler-Bernoulli theory. Because Timoshenko theory uses the Plane Assumption, there is

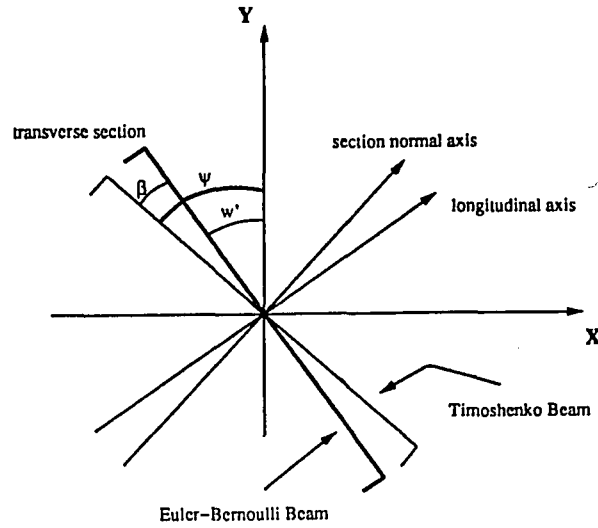


Figure 2.1: Euler-Bernoulli and Timoshenko motion of the transverse section

an additional rotation of the cross-section which accounts for the section normal axis to not coincide with the longitudinal axis of the beam. This additional rotation is indicated in Figure 2.1 by the angle β , and is known as shear deformation. The Normal Plane Assumption used by the Euler-Bernoulli model essentially means that β is zero, and the Euler-Bernoulli section normal axis coincides with the longitudinal axis.

The dynamics of the beam can be described by two parameters. One is the deflection of the beam from the axis corresponding to rigid rotation, $w(s)$, where s is the location on the beam where the deflection is being measured, as shown in Figure 2.2. The other is the rotation of the cross-section of the beam, which is ψ as shown in Figure 2.1. For the Timoshenko beam,

$$\psi = w'(s) + \beta, \quad (2.1)$$

while for the Euler-Bernoulli beam,

$$\psi = w'(s), \quad (2.2)$$

where the $(')$ operator denotes differentiation with respect to the X -axis.

Thus the Euler-Bernoulli theory describes deformation by only the flexible displacement, w , while the Timoshenko theory describes it by the two independent functions w and ψ . While there exist models based on the Timoshenko theory [73, 12], the mathematical tractability using the Euler-Bernoulli theory is simpler and thus it is the most prevalent in the literature. Based on this fact, the model used in this thesis is chosen to follow the Euler-Bernoulli theory.

2.3 Development of Kinematic Equations

2.3.1 Hamiltonian

Figure 2.2 shows a flexible manipulator carrying a load at its tip. The manipulator, which is simply a flexible beam fixed to a hub, rotates in the horizontal plane. The base coordinate system is (X_o, Y_o) and (X, Y) is a local coordinate system fixed on the hub. Initially it is assumed that the x-axes of all coordinate systems coincide with the base coordinate X_o axis. In the derivation given in this section, the operators $(\dot{})$ and $()'$ denote differentiation with respect to time and differentiation with respect to a spatial variable, respectively.

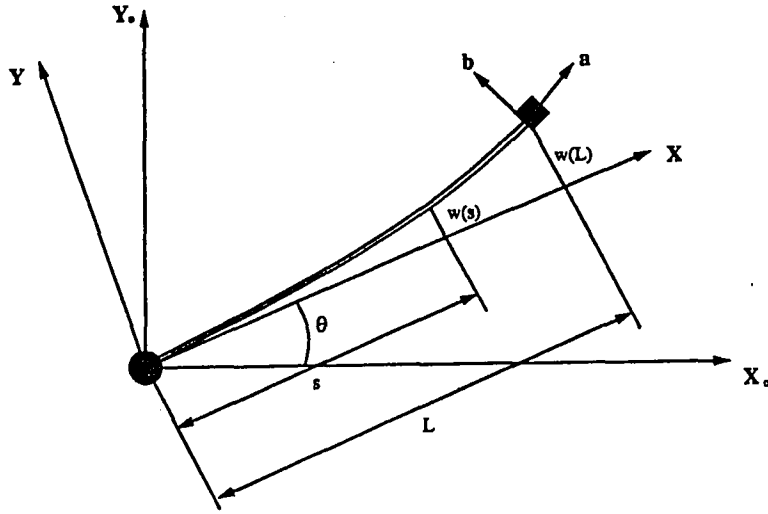


Figure 2.2: Single-Link Flexible Manipulator

From trigonometry, the base coordinates of a point (x_s, y_s) on the beam can be found as:

$$\begin{aligned} x_s &= s \cos \theta - w(s) \sin \theta \\ y_s &= s \sin \theta - w(s) \cos \theta \end{aligned} \quad (2.3)$$

where the variable θ denotes the rotation of the hub, and $w(s)$ denotes the tip deflection at a distance s from the hub, as shown in Figure 2.2. Notice that there are infinite values of $w(s)$ satisfying (2.3) for a specific (x_s, y_s) , and thus this is a distributed parameter system with infinite degrees of freedom. In order to show the effect of shear deformation, it will be included in the derivation at first, to be general. Later, the equation will be simplified to neglect this term.

Thus a point (x_δ, y_δ) resulting from the rotation of the plane transverse section during deformation is

$$\begin{aligned}x_\delta &= x_s - \delta \sin(\psi + \theta) \\y_\delta &= y_s - \delta \cos(\psi + \theta)\end{aligned}\tag{2.4}$$

where δ is the distance to the longitudinal axis of the beam. The base coordinates of a point (x_p, y_p) located on the payload can similarly be found as

$$\begin{aligned}x_p &= L \cos \theta - w(L) \sin \theta + a \cos(\psi + \theta) - b \sin(\psi + \theta) \\y_p &= L \sin \theta - w(L) \cos \theta + a \cos(\psi + \theta) - b \sin(\psi + \theta)\end{aligned}\tag{2.5}$$

where L is the length of the beam, as shown in Figure 2.2. The dynamic model derivation will utilize *Hamilton's Principle*, which states that

$$\delta \int_{t_0}^{t_1} (T + W - P) dt = 0\tag{2.6}$$

where δ in this case represents the variational operator, T is the kinetic energy and P is the potential energy of the manipulator-payload system, and W is the work done by the external forces on the system.

Kinetic Energy

The total kinetic energy is found as

$$T = T_b + T_h + T_p$$

where T_b, T_h , and T_p are the kinetic energies of the beam, hub, and payload, respectively. For the beam,

$$T_b = \frac{1}{2} \int_0^L T_{cross}(s) ds \quad (2.7)$$

where $T_{cross}(s)$ is the total kinetic energy of the cross-sectional slice at the distance s from the hub (see Figure 2.2), and is defined as

$$T_{cross}(s) = \iint_A \rho_b (\dot{x}_s^2 + \dot{y}_s^2) dA = \rho [\Delta_x^2 + \Delta_y^2 + S\dot{\psi}^2] \quad (2.8)$$

where

ρ_b = mass/volume

ρ = mass/length

I = moment of inertia of the beam cross-section

A = area of the cross-section

$S = I/A$ and is defined as rotary inertia

$$\Delta_x(s) = s\dot{\theta} + \dot{w}(s, t)$$

$$\Delta_y(s) = \dot{\theta}w(s, t).$$

Rotary inertia results from the fact that all points in the cross-section do not have the same motion, but in fact have different speeds associated with each point. Thus the final expression for T_b is

$$T_b = \frac{1}{2} \int_0^L \rho [\Delta_x^2 + \Delta_y^2 + S\dot{\psi}^2] ds. \quad (2.9)$$

The kinetic energy associated with the hub is simply one half the inertia of the hub, I_H , multiplied by the square of the rotational velocity of the hub, or

$$T_h = \frac{1}{2} I_H \dot{\theta}^2. \quad (2.10)$$

The kinetic energy associated with the tip load can similarly be described as

$$T_p = \frac{1}{2} \iint_{S_p} \rho_p (\dot{x}_p^2 + \dot{y}_p^2) dS_p$$

where ρ_p is the mass/volume of the payload, and S_p is the cross-sectional area of the payload. This expression can be simplified into

$$\begin{aligned} T_p = & \frac{M_p}{2} \{ \Delta_x^2(L) + \Delta_y^2(L) + [S\dot{\psi}^2(L) + \dot{\theta}](G_x \cos \psi(L) + G_y \sin \psi(L)) \} \\ & + \frac{J_p}{2} [\dot{\psi}(L) + \dot{\theta}]^2 \end{aligned} \quad (2.11)$$

in which

$$M_p = \iint_{S_p} \rho_p dS_p$$

and

$$J_p = \iint_{S_p} \rho_p (a^2 + b^2) dS_p$$

are the mass of the tip payload and the moment of inertia of the payload with respect to the local frame (a, b) , respectively, and

$$G_x = a_c \Delta_x + b_c \Delta_y$$

$$G_y = a_c \Delta_y - b_c \Delta_x$$

where (a_c, b_c) are the coordinates of the center of mass of the tip load with respect to (a, b) .

Potential Energy and Work

From beam theory [67], the potential energy of the beam can be expressed as

$$P = \frac{1}{2} \int_0^L \{D(\psi'(s))^2 + C[\psi(s) - (w'(s))^2]\} ds . \quad (2.12)$$

The prime denotes differentiation with respect to the X -axis. For a beam having uniform cross-section, $D = EI$ and $C = kGA$, where E is Young's modulus, G is the shear modulus and k is a shape factor depending on the specific shape of the beam. As before, I is the moment of inertia of the cross-section and A is the area of cross-section. The work done or applied to the manipulator is simply

$$W = \tau \theta \quad (2.13)$$

where τ is the torque applied at the hub of the manipulator.

Simplifications

As was alluded to earlier, some simplifying assumptions were made when modeling the arm. First, S is assumed to be zero. That is, all points on the same transverse cross-section have the same velocity as a point on the neutral axis of that cross-section. Secondly, G_x and G_y are assumed to be zero, implying that the payload is a point mass of moment J_p and mass M_p . Finally, the *Euler-Bernoulli Assumption* is invoked and thus shear deformation, β , is neglected. When these simplifications are implemented in (2.9, 2.10, 2.11, 2.12), the following result is

obtained:

$$T = \frac{1}{2}I_H\dot{\theta}^2 + \frac{1}{2}\int_0^L \rho[(s\dot{\theta} + \dot{w}(s))^2 + (\dot{\theta}w(s))^2]ds + \frac{M_p}{2}[(L\dot{\theta} + \dot{w}(L))^2 + (\dot{\theta}w(L))^2] + \frac{J_p}{2}[\dot{w}'(L) + \dot{\theta}]^2 \quad (2.14)$$

$$P = \frac{1}{2}\int_0^L [Dw''^2(s)]ds \quad (2.15)$$

$$W = \tau\theta. \quad (2.16)$$

2.3.2 Nonlinear Euler Equations and Boundary Conditions

Taking the variation of (2.14, 2.15, 2.16) results in

$$\delta P = \int_0^L Dw''(s)\delta w''(s)ds. \quad (2.17)$$

Expanding the integrand (and dropping the argument of w for convenience),

$$\begin{aligned} Dw''\delta w'' &= Dw''(\delta w')' \\ &= (Dw''\delta w')' - (Dw'')'\delta w' \\ &= (Dw''\delta w')' - [(Dw'')'\delta w]' + (Dw'')''\delta w. \end{aligned}$$

Using integration by parts,

$$\delta P = Dw''\delta w' \Big|_0^L - (Dw'')'\delta w \Big|_0^L + \int_0^L (Dw'')''\delta w ds. \quad (2.18)$$

Of course, the variation of the work applied is simply

$$\delta W = \tau\delta\theta. \quad (2.19)$$

For the kinetic energy, the variation is

$$\begin{aligned}\delta T = & I_H \dot{\theta} \delta \dot{\theta} + \int_0^L \{ \rho [s \dot{\theta} + \dot{w}] \delta (s \dot{\theta} + \dot{w}) + \delta ((\dot{\theta} w)^2) \} ds \\ & + \frac{M_p}{2} \delta [(L \dot{\theta} + \dot{w}^2(L))^2 + (\dot{\theta} w(L))^2] + \frac{J_p}{2} \delta [(\dot{w}'(L) + \dot{\theta})^2] \quad (2.20)\end{aligned}$$

or

$$\delta T = \delta T_h + \delta T_b + \delta T_p$$

where δT_h , δT_b , and δT_p , are the variations of the kinetic energy of the hub, beam and payload, respectively. The evaluation of the Hamiltonian is simplified if the above expressions are described by variations of undifferentiated functions, as opposed to time derivatives of functions. Therefore, some manipulation will be done to accomplish this for the expressions above. By using the product rule of differentiation, the variation of the kinetic energy associated with the hub is

$$\delta T_h = \overline{I_H \dot{\theta} \delta \theta} - I_H \ddot{\theta} \delta \theta \quad (2.21)$$

where the overline/dot notation $\overline{f(t)}$ denotes the time derivative of the entire function $f(t)$.

Similarly, for the beam ,

$$\begin{aligned}\delta T_b = & \rho \int_0^L \{ [s \dot{\theta} + \dot{w}] (s \delta \dot{\theta} + \delta \dot{w}) + \dot{\theta} w \delta (\dot{\theta} w) \} ds \\ = & \rho \int_0^L \{ [s \dot{\theta} + \dot{w}] (s \delta \dot{\theta} + \delta \dot{w}) + \dot{\theta} w (w \delta \dot{\theta} + \dot{\theta} \delta w) \} ds \\ = & \rho \int_0^L \{ [s (s \dot{\theta} + \dot{w}) + \dot{\theta} w^2] \delta \dot{\theta} + [s \dot{\theta} + \dot{w}] \delta \dot{w} + \dot{\theta}^2 w \delta w \} ds \\ = & \rho \int_0^L \{ \Delta_\theta \delta \dot{\theta} + \Delta_w \delta \dot{w} + \dot{\theta}^2 w \delta w \} ds\end{aligned}$$

$$\begin{aligned}
&= \rho \int_0^L \{ \overline{\Delta_{\theta} \dot{\delta \theta}} - \dot{\Delta_{\theta}} \delta \theta + \overline{\Delta_w \dot{\delta w}} - \dot{\Delta_w} \delta w + \dot{\theta}^2 w \delta w \} ds \\
\delta T_b &= \rho \int_0^L \{ \overline{\Delta_{\theta} \dot{\delta \theta}} + \overline{\Delta_w \dot{\delta w}} - \dot{\Delta_{\theta}} \delta \theta + (\dot{\theta}^2 w - \dot{\Delta_w}) \delta w \} ds
\end{aligned} \tag{2.22}$$

in which

$$\Delta_{\theta} = s(s\dot{\theta} + \dot{w}) + \dot{\theta}w^2 \tag{2.23}$$

$$\Delta_w = s\dot{\theta} + \dot{w}. \tag{2.24}$$

Finally, for the payload,

$$\begin{aligned}
\delta T_p &= M_p[L\dot{\theta} + \dot{w}(L)](\delta(L\dot{\theta}) + \delta\dot{w}(L)) + \dot{\theta}w(L)(w(L)\delta\dot{\theta} + \dot{\theta}\delta w(L)) \\
&\quad + J_p(\dot{w}'(L) + \dot{\theta})(\delta\dot{w}'(L) + \delta\dot{\theta}) \\
&= [M_p(L^2\dot{\theta} + L\dot{w}(L) + \dot{\theta}w^2(L) + J_p(\dot{w}'(L) + \dot{\theta}))]\delta\dot{\theta} \\
&\quad + M_p(L\dot{\theta} + \dot{w}(L))\delta\dot{w}(L) + M_p\dot{\theta}^2 w(L)\delta w(L) + J_p(\dot{w}'(L) + \dot{\theta})\delta\dot{w}'(L) \\
&= \Delta_{\theta_{pay}}\delta\dot{\theta} + \Delta_{w_1}\delta\dot{w}(L) + M_p\dot{\theta}^2 w(L) + \Delta_{w_2}\delta\dot{w}'(L) \\
\delta T_p &= \overline{\Delta_{\theta_{pay}} \dot{\delta \theta}} + \overline{\Delta_{w_1} \dot{\delta w}(L)} + \overline{\Delta_{w_2} \dot{\delta w}'(L)} \\
&\quad - \dot{\Delta_{\theta_{pay}}} \delta \theta + (M_p\dot{\theta}^2 w(L) - \dot{\Delta_{w_1}})\delta w(L) - \dot{\Delta_{w_2}} \delta w'(L)
\end{aligned} \tag{2.25}$$

where

$$\Delta_{\theta_{pay}} = M_p(L^2\dot{\theta} + L\dot{w}(L) + \dot{\theta}w^2(L)) + J_p(\dot{w}'(L) + \dot{\theta}) \tag{2.26}$$

$$\Delta_{w_1} = M_p(L\dot{\theta} + \dot{w}(L)) \tag{2.27}$$

$$\Delta_{w_2} = J_p(\dot{w}'(L) + \dot{\theta}). \tag{2.28}$$

Substituting (2.18, 2.19, 2.21, 2.22, 2.25) into the Hamiltonian,

$$\int_{t_0}^{t_1} \delta T dt - \int_{t_0}^{t_1} \delta P dt + \int_{t_0}^{t_1} \delta W dt = 0 \quad (2.29)$$

where

$$\begin{aligned} \int_{t_0}^{t_1} \delta T dt &= \int_{t_0}^{t_1} \{ I_H \dot{\theta} \delta \theta + \int_0^L \rho (\Delta_\theta \delta \theta + \Delta_w \delta w) ds + \Delta_{\theta_{pay}} \delta \theta \\ &\quad + \Delta_{w_1} \delta w(L) + \Delta_{w_2} \delta w'(L) \} \\ &\quad - I_H \ddot{\theta} \delta \theta - \int_0^L \rho [\dot{\Delta}_\theta \delta \theta + (\dot{\Delta}_w - \dot{\theta}^2 w) \delta w] ds \\ &\quad - \dot{\Delta}_{\theta_{pay}} \delta \theta + (\dot{\theta}^2 w(L) - \dot{\Delta}_{w_1}) \delta w(L) + \dot{\Delta}_{w_2} \delta w'(L) \} dt \end{aligned} \quad (2.30)$$

$$\int_{t_0}^{t_1} \delta P dt = \int_{t_0}^{t_1} \{ Dw'' \delta w' |_0^L - (Dw'')' \delta w |_0^L + \int_0^L (Dw'')'' \delta w ds \} dt \quad (2.31)$$

$$\int_{t_0}^{t_1} \delta W dt = \int_{t_0}^{t_1} \tau \delta \theta dt. \quad (2.32)$$

Noting that all variations at t_0 and t_1 are zero, and grouping similar terms,

$$\begin{aligned} &\int_{t_0}^{t_1} \{ (-I_H \ddot{\theta} - \int_0^L \rho \dot{\Delta}_\theta ds - \dot{\Delta}_{\theta_{pay}} + \tau) \delta \theta \} dt - \\ &\int_{t_0}^{t_1} \{ \int_0^L (\rho \dot{\Delta}_w - \rho \dot{\theta}^2 w + (Dw'')'') ds \} \delta w dt + \\ &\int_{t_0}^{t_1} \{ [\dot{\theta}^2 w(L) - \dot{\Delta}_{w_1} + (Dw'')'] \delta w(L) - (Dw''(L) + \dot{\Delta}_{w_2}) \delta w'(L) \} dt + \\ &\int_{t_0}^{t_1} \{ Dw''(0) \delta w'(0) - Dw'''(0) \delta w(0) \} dt = 0. \end{aligned} \quad (2.33)$$

Thus, the Euler equations describing the dynamics of the nonlinear system are:

$$(Dw'')'' + \rho(s\ddot{\theta} + \ddot{w} - \dot{\theta}^2 w) = 0 \quad (2.34)$$

$$I_H \ddot{\theta} + \frac{\partial}{\partial t} \int_0^L \rho [(s\dot{\theta} + \dot{w})s + \dot{\theta} \dot{w}^2] ds + \frac{\partial}{\partial t} \Delta_{\theta_{pay}} = \tau \quad (2.35)$$

with boundary conditions

$$w(0) = 0 \quad (2.36)$$

$$w'(0) = 0 \quad (2.37)$$

$$(Dw''(L))' + M_p(\dot{\theta}^2 w(L) - L\tilde{\theta} + \tilde{w}(L)) = 0 \quad (2.38)$$

$$Dw''(L) + J_p(\tilde{w}'(L) + \tilde{\theta}) = 0 . \quad (2.39)$$

2.3.3 Non-dimensional Euler Equations

The Euler equations given above are quite difficult to work with, as they are integro-partial differential equations. Some simplifications can be facilitated by an expansion of the equations. First, the equations will be linearized and the natural modal frequencies will be obtained from the characteristic equation of this set of equations. Then, these natural modes will be used as a basis for expansion of the general solution of the nonlinear equations.

The linearization is accomplished by truncating all higher order terms, resulting in the following :

$$(Dw'')'' + \rho(s\tilde{\theta} + \tilde{w}) = 0 \quad (2.40)$$

$$I_H \tilde{\theta} + \frac{\partial}{\partial t} \int_0^L \rho[(s\dot{\theta} + \dot{w})s]ds + \frac{\partial}{\partial t} \Delta_{\theta_{py}} = \tau . \quad (2.41)$$

In order to reduce the integral expression in (2.41) to an algebraic one, a relationship is derived using the boundary conditions. Equation (2.40) is multiplied by

s and integrated over the length of the beam to yield

$$\int_0^L [s(Dw'')'' + \rho s(s\ddot{\theta} + \ddot{w})] ds = 0. \quad (2.42)$$

Therefore,

$$\int_0^L \rho s(s\ddot{\theta} + \ddot{w}) ds = - \int_0^L s(Dw'')'' ds = - \int_0^L s d(Dw'')'. \quad (2.43)$$

Using integration by parts, and assuming D is a constant,

$$\begin{aligned} - \int_0^L s d(Dw'')' &= -s(Dw'')' \Big|_0^L + \int_0^L (Dw'')' ds \\ &= -LDw'''(L) + Dw'' \Big|_0^L \\ &= -LDw'''(L) + Dw''(L) - Dw''(0). \end{aligned}$$

By making use of the boundary conditions at the tip given by (2.38, 2.39) and noting that

$$\frac{\partial}{\partial t} \Delta_{\theta_{pay}} = J_p(\ddot{w}'(L) + \ddot{\theta}) + L[M_p(L\dot{\theta} + \dot{w}(L))], \quad (2.44)$$

the following is obtained:

$$- \int_0^L \rho s(s\ddot{\theta} + \ddot{w}) + \frac{\delta}{\delta t} \Delta_{\theta_{pay}} = -Dw''(0). \quad (2.45)$$

This result greatly simplifies the dynamic equations. To simplify the dynamic equations further, the quantity v , the total deflection of the arm, is defined as

$$v(x, t) = w(x, t) + x\theta(t). \quad (2.46)$$

Substituting (2.45) and (2.46) into (2.40) and (2.41), the resulting linear model is

$$Dv'''' + \rho \ddot{v} = 0 \quad (2.47)$$

$$I_H \ddot{\theta} - Dv''(0) = \tau \quad (2.48)$$

with corresponding boundary conditions

$$v(0, t) = 0 \quad (2.49)$$

$$v'(0, t) = \theta \quad (2.50)$$

$$Dv''(L) + J_p \ddot{v}(L) = 0 \quad (2.51)$$

$$Dv'''(L) = M_p \ddot{v} . \quad (2.52)$$

One should remember that a dot ($\dot{}$) denotes the derivative with respect to time whereas the prime ($'$) denotes the derivative with respect to the variable along the X -axis.

Before solving for the natural modal frequencies of the system, the model is made non-dimensional to simplify the nomenclature of the dynamic equations further. The following definitions are substituted into the Euler equations and boundary conditions above:

$$\begin{aligned} \xi = x/L, \quad z = v/L, \quad c^2 = \frac{\rho L^4}{D}, \\ \mu = \frac{M_p}{\rho L}, \quad \eta = \frac{I_H}{\rho L^3}, \quad \kappa = \frac{J_p}{\rho L^3} . \end{aligned} \quad (2.53)$$

By using these non-dimensional parameters and using (2.50) as the definition of θ , the non-dimensional model is obtained as

$$z''''(\xi) + c^2 \ddot{z}(\xi) = 0 \quad (2.54)$$

with boundary conditions

$$z(0) = 0 \quad (2.55)$$

$$\eta c^2 \ddot{z}'(0) - z''(0) = \frac{\tau L}{D} \quad (2.56)$$

$$z''(1) + c^2 \kappa \ddot{z}'(1) = 0 \quad (2.57)$$

$$z'''(1) - \mu c^2 \ddot{z}(1) = 0. \quad (2.58)$$

Note that a prime now indicates differentiation with respect to the non-dimensional coordinate ξ .

2.4 Mode Shapes and Modal Frequencies

To solve for the natural vibrational frequencies of the beam, consider the beam while resonating. In this state one can assume that z and $z'(0) = \theta$ are of the form

$$z(s, t) = z(s) \sin \omega t \quad (2.59)$$

$$z'(s, t) = z'(s) \sin \omega t \quad (2.60)$$

where ω is the physical vibrational frequency of the beam. Substituting (2.59, 2.60) in (2.54), and also assuming that since the beam is resonating, no external force is needed to excite the vibrational modes, i.e., $\tau = 0$, the linear model under vibration can be rewritten as

$$z'''' - m^2 z(\xi) = 0 \quad (2.61)$$

with boundary conditions

$$z(0) = 0 \quad (2.62)$$

$$\eta m^2 z'(0) + z''(0) = 0 \quad (2.63)$$

$$z''(1) - \kappa m^2 z'(1) = 0 \quad (2.64)$$

$$z'''(1) + \mu m^2 z(1) = 0. \quad (2.65)$$

$m = \alpha\omega$ is the dimensionless frequency.

The solution of the linear model under vibration can be written in the form

$$z(\xi) = C_1 \cos \lambda \xi + C_2 \sin \lambda \xi + C_3 \cosh \lambda \xi + C_4 \sinh \lambda \xi \quad (2.66)$$

where $\lambda = \sqrt{m}$. The coefficients of the solution can be solved from the boundary conditions. From (2.62), $C_1 = -C_3$, reducing the solution to

$$z(\xi) = C_1(\cos \lambda \xi - \cosh \lambda \xi) + C_2 \sin \lambda \xi + C_4 \sinh \lambda \xi. \quad (2.67)$$

Substituting the first and second derivative of (2.67) into (2.63), one more coefficient can be solved for as

$$C_1 = \frac{\eta \lambda^3}{2}(C_2 + C_4). \quad (2.68)$$

Therefore the solution can once again be rewritten as

$$z(\xi) = C_2 \left[\sin \lambda \xi + \frac{\eta \lambda^3}{2} (\cos \lambda \xi - \cosh \lambda \xi) \right] + C_4 \left[\sinh \lambda \xi + \frac{\eta \lambda^3}{2} (\cos \lambda \xi - \cosh \lambda \xi) \right] \quad (2.69)$$

or

$$z(\xi) = C_2 z_s(\xi) + C_4 z_h(\xi). \quad (2.70)$$

The final two boundary conditions at the tip are now utilized in order to solve for C_2 and C_4 .

$$C_2(z''_s(1) - \kappa m^2 z'_s(1)) + C_4(z''_h(1) - \kappa m^2 z'_h(1)) = 0$$

$$C_2(z'''_s(1) + \mu m^2 z_s(1)) + C_4(z'''_h(1) + \mu m^2 z_h(1)) = 0$$

or in matrix form,

$$\begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \begin{bmatrix} C_2 \\ C_4 \end{bmatrix} = 0 \quad (2.71)$$

where

$$\begin{aligned} z_{11} &= z_s''(1) - \kappa m^2 z_s'(1) \\ z_{12} &= z_h''(1) - \kappa m^2 z_h'(1) \\ z_{21} &= z_s'''(1) + \mu m^2 z_s(1) \\ z_{22} &= z_h'''(1) + \mu m^2 z_h(1) . \end{aligned}$$

Thus the modal shape function is

$$z(\xi, m) = z_s(\xi) - \frac{z_{11}(m)}{z_{12}(m)} z_h(\xi) \quad (2.72)$$

and the characteristic function to determine the natural frequencies of vibration is

$$\begin{vmatrix} z_{11}(m) & z_{12}(m) \\ z_{21}(m) & z_{22}(m) \end{vmatrix} = 0 . \quad (2.73)$$

Of course, the trivial solution $m = 0$ corresponds to the rigid mode. In this case, the mode shape is simply

$$z(\xi) = \xi . \quad (2.74)$$

2.4.1 Orthogonality of Mode Shapes

One can show (see [72] for details) that for two different modal frequencies m_i and m_j , the corresponding mode shapes z_i and z_j are orthogonal, which leads to the result

$$R(z_i, z_j) = \int_0^L z_i(\xi) z_j(\xi) d\xi + \eta z_i'(0) z_j'(0) + \mu z_i(1) z_j(1) + \kappa z_i'(1) z_j'(1) = 0 . \quad (2.75)$$

Use of this orthogonality property will be another simplifying operation in the development of the final state-space dynamic equations, by providing an orthogonal basis for the modal expansion of the nonlinear partial differential dynamic equation.

If two functions are orthogonal, the following relationship exists (the arguments of the mode shape functions are left out for convenience),

$$\int_0^1 z_i'' z_j'' d\xi = 0. \quad (2.76)$$

Evaluating this integral by integrating by parts (twice),

$$\int_0^1 z_i'' z_j'' d\xi = z_i'' z_j' \Big|_0^1 - z_j z_i''' \Big|_0^1 + \int_0^1 z_i'''' z_j d\xi = 0. \quad (2.77)$$

From equation (2.61), $z_i'''' = m_i^2 z_i$. Thus,

$$\int_0^1 z_i'' z_j'' d\xi = z_i'' z_j' \Big|_0^1 - z_j z_i''' \Big|_0^1 + \int_0^1 m_i^2 z_i z_j d\xi = 0. \quad (2.78)$$

Using the boundary conditions, one can now derive this result:

$$\begin{aligned} \int_0^1 z_i'' z_j'' d\xi &= m_i^2 \int_0^L z_i(\xi) z_j(\xi) d\xi + \eta z_i'(0) z_j'(0) + \mu z_i(1) z_j(1) + \kappa z_i'(1) z_j'(1) \\ &= m_i^2 R(z_i, z_j) = m_j^2 R(z_j, z_i) = 0. \end{aligned} \quad (2.79)$$

Thus,

$$\int_0^1 z_i'' z_j'' d\xi = (m_i^2 - m_j^2) R(z_i, z_j) = 0. \quad (2.80)$$

This equation can only equal zero if $R(z_i, z_j) = 0$, since the mode shapes m_i and m_j are not equal. Therefore, if two mode shape functions are orthogonal, $R(z_i, z_j)$ must be equal to zero.

Now let the normalized mode shapes for the flexible modes be defined as

$$z(\xi) = \frac{z(\xi, m_i)}{\sqrt{R(z_i, z_i)}}, \quad i = 1, 2, \dots, \infty \quad (2.81)$$

and the normalized rigid mode shape function be

$$z_0(\xi) = \frac{\xi}{\sqrt{Q(z_0, z_0)}} \quad (2.82)$$

where

$$Q(z_0, z_0) = \int_0^1 \xi^2 d\xi + \eta + \mu + \kappa. \quad (2.83)$$

With the mode shapes normalized, the orthogonality property between mode shapes gives the following results:

$$R(z_i, z_j) = \delta_{ij}, \quad \int_0^1 z_i'' z_j'' d\xi = m_i^2 \delta_{ij}, \quad \delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \quad (2.84)$$

2.5 Modal Expansion of Nonlinear and Linear Models

For facilitating the use of conventional control algorithms, the partial differential dynamic equation needs to be approximated by an ordinary differential equation. One way to do this is to use modal expansion, utilizing the mode shapes derived above. The response of the arm under a general torque can be defined as

$$z(\xi, t) = \sum_{i=0}^{\infty} z_i(\xi) q_i(t). \quad (2.85)$$

Similarly, for the *Euler-Bernoulli* model, the first derivative with respect to ξ can also be represented as

$$\alpha(\xi, t) = \frac{dz(\xi)}{d\xi} = \sum_{i=0}^{\infty} \alpha_i(\xi) q_i(t) \quad (2.86)$$

where

$$\alpha_i(\xi) = z'_i(\xi). \quad (2.87)$$

The infinite sum is then truncated to a finite one, and represented vectorially as,

$$z(\xi, t) = \sum_{i=0}^n z_i(\xi) q_i(t) = Z_N(\xi) q(t) \quad (2.88)$$

$$\alpha(\xi, t) = \Lambda_N(\xi) q(t) \quad (2.89)$$

where

$$q(t) = \begin{bmatrix} q_0(t) & \cdots & q_n(t) \end{bmatrix}^T, \quad Z_N(\xi) = \begin{bmatrix} z_0(\xi) & \cdots & z_n(\xi) \end{bmatrix} \quad (2.90)$$

and

$$\Lambda_N(\xi) = \begin{bmatrix} \alpha_0(\xi) & \cdots & \alpha_n(\xi) \end{bmatrix}. \quad (2.91)$$

$Z_N(\xi)$ and $\Lambda_N(\xi)$ are the vectors of the n normalized mode shapes. Substituting the modal expansions (2.88, 2.89) into the non-dimensional form of (2.14, 2.15, 2.16), one obtains the following expressions for the kinetic energy, potential energy, and work:

$$\begin{aligned} T = & \frac{c^2}{2} \left[\int_0^1 \{ \dot{q}^T Z_N^T Z_N q + (\dot{q}^T \Lambda_N(0) \Lambda_N(0) \dot{q}) (q^T W_N^T W_N q) \} d\xi \right. \\ & + \eta \dot{q}^T \Lambda_N^T(0) \Lambda_N(0) \dot{q} + \kappa \dot{q}^T \Lambda_N(1) \Lambda_N(1) \dot{q} + \mu \dot{q}^T Z_N^T(1) Z_N(1) \dot{q} \\ & \left. + \mu (\dot{q}^T \Lambda_N^T(0) \Lambda_N^T(0) \dot{q}) (q^T W_N^T(1) W_N(1) q) \right] \quad (2.92) \end{aligned}$$

$$P = \frac{1}{2} \int_0^1 q^T \Lambda_N'^T \Lambda_N' q d\xi \quad (2.93)$$

$$W = \frac{\tau L}{D} \Lambda_N(0) q \quad (2.94)$$

where

$$\theta = \Lambda_N(0)q = Z'_N(0)q, \quad W_N(\xi) = Z_N(\xi) - \xi\Lambda_N(0). \quad (2.95)$$

Combining all $\dot{q}^T(\cdot)\dot{q}$ terms in (2.92) one gets

$$T = \frac{c^2}{2}[\dot{q}^T R(Z_N, Z_N)\dot{q} + \dot{q}^T C_\alpha \dot{q} q^T C_\omega q]. \quad (2.96)$$

From (2.84),

$$R(z_i, z_j) = \begin{cases} 0 & i \neq j \\ 1 & i = j. \end{cases}$$

Therefore, $R(Z_N, Z_N) = I$, the identity matrix. The kinetic energy now is

$$T = \frac{c^2}{2}[\dot{q}^T \dot{q} + \dot{q}^T C_\alpha \dot{q} q^T C_\omega q] \quad (2.97)$$

where

$$C_\omega = \int_0^L W_N^T(\xi) W_N(\xi) d\xi + \mu W_N^T(1) W_N(1), \quad C_\alpha = \Lambda_N^T(0) \Lambda_N(0). \quad (2.98)$$

The orthogonality of the mode shapes also simplifies the expression for the potential energy. Since $\Lambda'_N(\xi) = Z''_N(\xi)$, one obtains from (2.84),

$$\int_0^1 \Lambda_N'^T(0) \Lambda'_N(0) d\xi = \begin{bmatrix} m_0^2 & 0 & \cdots & 0 \\ 0 & m_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n^2 \end{bmatrix}; \quad (2.99)$$

and, since $m_i = c\omega_i$, the potential energy can be expressed as

$$P = \frac{c^2}{2} q^T \Omega q \quad (2.100)$$

where

$$\Omega = \begin{bmatrix} \omega_0^2 & 0 & \cdots & 0 \\ 0 & \omega_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_n^2 \end{bmatrix}. \quad (2.101)$$

At this point, the variations of (2.94, 2.97, 2.100) can once again be calculated and substituted into the Hamiltonian to get the final dynamic equation, or the Euler-Lagrange equation may be used to achieve the same result. The final governing equation of the single link flexible manipulator is

$$(I + q^T C_\omega q C_\alpha) \ddot{q} + (\dot{q}^T C_\omega q I - C_\omega q \dot{q}^T) C_\alpha \dot{q} + \Omega q = \frac{\tau L}{c^2 D} \Lambda_N^T(0). \quad (2.102)$$

The linearized form of this equation can be found by simply eliminating all higher-order terms, or simply setting C_α and C_ω to zero, to yield

$$\ddot{q} + \Omega q = \frac{\tau L}{c^2 D} \Lambda_N^T(0). \quad (2.103)$$

The nonlinear equation given in (2.102) is used for simulation purposes. By including the nonlinearities of the dynamics in the simulation model, a more accurate prediction of the actual response of the physical beam can be obtained with the simulation. This can alert the user to difficulties or short-comings of control methods being tested, especially controls which are based on the linear model (2.103). If the operating point is not set right for these linear controls, the system could become unstable. It is far better and less costly to learn this during testing on the simulator than if the control was tested on the robot arm itself. For this reason, the nonlinear model, despite the extra effort in derivation, is a much better choice to represent the dynamics of the flexible manipulator.

2.6 Observability and Controllability of the Linear Model

A state-space representation of the linear dynamical equation (2.103) is

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = A \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + B\tau, \quad (2.104)$$

with outputs

$$\begin{bmatrix} \theta \\ \dot{\theta} \\ w \\ \dot{w} \\ S \\ \dot{S} \end{bmatrix} = C \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \quad (2.105)$$

where

$$A = \begin{bmatrix} 0 & I_{N \times N} \\ -\Omega & 0 \end{bmatrix}; B = \begin{bmatrix} 0 \\ \frac{L}{\mathcal{E}D} \Lambda_N(0) \end{bmatrix} \quad (2.106)$$

and

$$C = \begin{bmatrix} \Lambda_N(0) & 0 \\ 0 & \Lambda_N(0) \\ W_N(1) & 0 \\ 0 & W_N(1) \\ \Gamma(0.5) & 0 \\ 0 & \Gamma(0.5) \end{bmatrix}. \quad (2.107)$$

The outputs S and \dot{S} are the strain and strain rate, respectively. Γ is the strain shape, which is given by

$$\Gamma(\xi) = W''_N(\xi). \quad (2.108)$$

Before applying control or trying to observe any states of a given system, one should check whether the system is controllable or observable or both. If there is an uncontrollable or unobservable mode, it can be decoupled through a canonical decomposition and the rest of the system can then be controlled or observed.

A simple test [16] for determining the controllability and observability of a system is $\text{rank}(U_c) = n$ for a controllable system, and $\text{rank}(U_o) = n$ for an observable

system, where n is the dimension of the system, and

$$U_c = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \quad (2.109)$$

and

$$U_o = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}. \quad (2.110)$$

These conditions can be checked for simple linear systems, but when A is ill conditioned, and then multiplied with B or C , large errors will result, and the evaluation of the rank of the controllability or observability matrices will probably be incorrect. As it turns out in the present case, the condition number of A is very large, posing an ill-conditioned problem. For example, for $\mu = \kappa = 0$ (no payload), the condition number of A as calculated by MATLAB is infinite.

Therefore another way must be found to check the controllability and observability of the linear model. One way to avoid the multiplication of the A -matrix is to use the Hautus-Rosenbrock Test [16], which states that if

$$\text{rank} \begin{bmatrix} \lambda I - A & B \end{bmatrix} = n; \quad \text{rank} \begin{bmatrix} \lambda I - A \\ C \end{bmatrix} = n \quad (2.111)$$

where λ is the eigenvalue of the associated mode that is being tested, then that mode is controllable or observable. When this test is used, the linear model of the flexible arm is shown to be completely controllable and observable. But, if (2.109) and (2.110) are used, the respective ranks both come out to be 6, erroneously indicating that the system is neither controllable nor observable. Thus, the effect of numerically unstable methods is clearly seen.

What is important to note is the ill conditioned nature of A . Hence, care must be taken in all computations of controls to avoid the involvement of A in any multiplication.

Since the linear system is both controllable and observable, the poles of the system may be placed theoretically anywhere in the LHP. A simple way to do this is with state feedback, but in this system the states cannot be measured directly and must be estimated by an observer. The inputs to the observer are the outputs shown in (2.105), and these are all easily measurable and have been used in implementations by other researchers. There are several design procedures for linear observers that are outlined by Chen [16]. Perhaps the most useful in this case (since there are 10 states that must be estimated) is the reduced order observer. Chen shows that it is possible to reduce the complexity of the observer by the number of the outputs available. The equation describing the observer dynamics is

$$\dot{z} = Fz + Gy + Hu , \quad (2.112)$$

where F is a matrix chosen such that all of its eigenvalues have negative real parts and are disjoint from those of A , G is chosen such that $\{F, G\}$ is controllable, and $H = TB$, where T is solved from the equation

$$TA - FT = GC . \quad (2.113)$$

Once these quantities have been solved, the estimated states can be obtained from

$$\hat{x} = P^{-1} \begin{bmatrix} y \\ z \end{bmatrix} \quad (2.114)$$

where

$$P = \begin{bmatrix} C \\ T \end{bmatrix}. \quad (2.115)$$

Note that P must be nonsingular. If it turns out to be singular, a new F or G must be chosen and T must be recalculated. It must also be emphasized that a numerically stable method be used to solve for T , due to the multiplication of it with A in (2.113) (see [2, 5, 20] for details). It is clearly seen that the outputs y , reduce the number of states that need to be explicitly estimated.

Thus, for the case of 4 flexible modes (10 total) states, using the 6 outputs requires that only a 4th order observer be implemented to have estimates of all states. This reduction of order would greatly reduce the dimension of the observer needed and help speed the convergence of the error of the estimated states to zero. It is typically required that the dynamics of the observer be 2 - 5 times faster than the desired dynamics of the closed-loop system. While there are many interesting design considerations for observers, this thesis is concerned with control only. Therefore, it is assumed in all simulations that the states are available for feedback.

2.7 Simulation using MATLAB

The nonlinear equation given in (2.102) can be simulated entirely within a software package known as MATLAB, which is especially powerful with matrix manipulations and includes many built-in commands. These commands are convenient

because no language-level programming in FORTRAN or C is needed, and the simulation is self-contained in the software package. The process of simulation can be basically broken down into two parts: calculating the parameters, and solving the differential equation based on these parameters.

The first thing that needs to be done is to calculate the eigenmodes, or the vibrational frequencies. As presented earlier in (2.73), this can be accomplished by computing the determinant and finding those values of m which make the determinant 0. This is done by a combination of a binary search (which locates the sign changes of the function) and the `fsolve` function of MATLAB. This built-in function locates the zeros of a function, and the binary search gives the starting point of the search by `fsolve`, to get a more precise and reliable result.

After the eigenmodes are calculated, the mode shapes corresponding to each eigenmode are calculated, based on the boundary conditions at the tip of the beam. The normalization constant is then calculated according to (2.81). Thus, the mode shapes divided by the normalization constant give $Z_N(\xi)$, and likewise the parameter $\Lambda_N(\xi)$ is obtained.

Once these vectors are known, the C_α and C_ω matrices in (2.102) are computed, and then all parameters are known. These computations are only done once per simulation, but must be recalculated for every payload change. All numerical integration for the calculations of the parameters was done using standard Euler integration. The integration was done twice, once with 90 steps and once with 100

steps. The two results were then extrapolated to approximate a higher number of steps and thus get a more accurate result.

The differential equation is solved in MATLAB by using the ODE45 function, which is a fourth order Runge-Kutta integration algorithm. The algorithm has an adaptive step size to facilitate faster computation of the state time histories. When these time histories are computed, the corresponding outputs (θ , w , etc.) can be calculated and plotted in MATLAB as well.

2.8 Results and Numerical Difficulties

There were a couple of checks that were performed to make sure that the parameters were being calculated accurately. The eigenmodes were checked against the known vibrational frequencies of the CIRSSE flexible arm found in [71]. This arm is characterized by the parameters given in Table 2.1. The computed frequencies compared very well with the experimental results for the first 5 modes, as shown in Table 2.2. After the fifth mode, the accuracy of the calculation decreased. This was due to the fact that the exponential functions in \sinh and \cosh in the computation become very large in magnitude for higher frequencies, causing numerical sensitivity. For this reason, only four flexible modes were included in the model. Many researchers only include one or two modes in their models, and hence inclusion of four modes should be adequate to approximate the true dynamics of a single-link flexible manipulator.

Parameter		Value	
L	beam length	1.098	m
B	beam width	1.5875e-3	m
H	beam height	0.103	m
E	material Young's modulus	68950.0e-6	N/m^2
γ	beam material density	2.713e3	kg/m^3
I_H	hub inertia	0.007	$kg\ m^2$
ρ	beam mass per unit length	0.4436	kg/m
A	beam transverse area	1.6351e-4	m^2
k	beam shape factor	0.8497	-
E_{adj}	E adjustment factor	0.88	-
I_{Hadj}	I_H adjustment factor	2.5	-

Table 2.1: Parameters of the CIRSSE Single-Link Flexible Manipulator

mode number	calculated (Hz)	experimental (Hz)
0	0	0
1	2.9692	2.85
2	7.2608	7.20
3	17.9773	18.42
4	34.7523	35.65
5	57.2774	58.70
6	85.4823	88.00
7	119.348	126.3
8	158.869	166.6
9	204.048	214.4

Table 2.2: Modal Frequency Comparison with CIRSSE Arm

The mode shapes were also checked to insure that they were indeed orthogonal. Again, the orthogonality condition wasn't satisfied after the fifth mode for the same reasons as above, since the mode shapes depend on the modal frequencies. The orthogonality conditions of (2.84) were checked for the four flexible modes of the model and were found to hold.

The solution of the dynamic equation also had some numerical problems. When the input torque changed suddenly, such as when a step input is applied, singularities occurred, and the simulation stopped. After considerable research into the code of ODE45, it was found that the cause of this was the adaptive step size that the algorithm uses. In order to prevent the integration from "stepping" over a part of the simulation in which a rapid change occurs, and thus inadvertently concluding that the function has become discontinuous, the maximum step size must be limited. After this was done, the simulation performed well, as shown in the next section.

2.9 Pulse Response

The pulse response of the system was simulated to show the vibrational characteristics of the arm. The input was a 0.1 second pulse of torque with a magnitude of 1 N/m , applied at the hub. The tip deflection is shown in Figure 2.3 and the hub response is shown in Figure 2.4, with no payload attached to the tip of the beam. It can be seen that the flexible modes indeed have been excited, and will continue

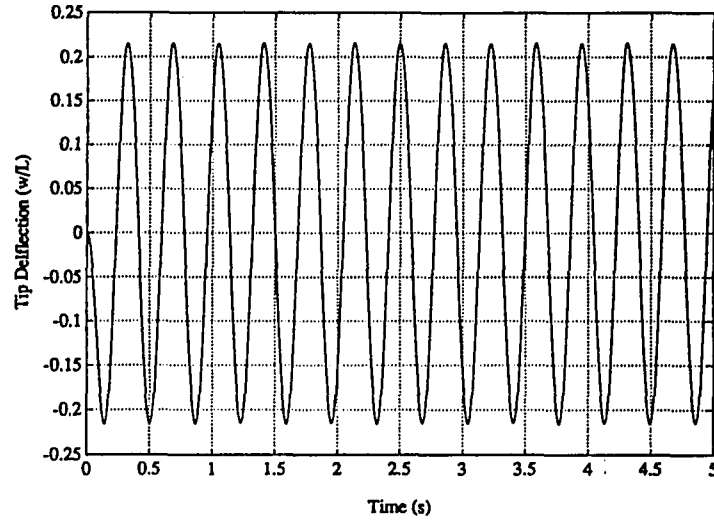


Figure 2.3: Tip Deflection due to Pulse Input, No Payload

to vibrate forever, because no friction or structural damping has been included in the model. The hub will also continue to rotate with constant velocity indefinitely as well, for the same reason. Note that the tip deflection is shown in terms of the dimensionless parameters; so a deflection of 0.1 means that the deflection is 10% of the length of the beam.

A second simulation is shown and this time the beam is carrying a payload, and hence its effects are seen on the dynamics in Figures 2.5 and 2.6. The payload is that corresponding to the non-dimensional payload parameters $\mu = 0.5$, and $\kappa = 0.5$. In other words, the mass at the end of the beam was half of the total mass of the beam, and the inertia associated with the payload was half the inertia of the hub. Note that in all simulations, the inertial and mass parameters of the payload are equal. This is an arbitrary choice and the parameters could be different if so needed.

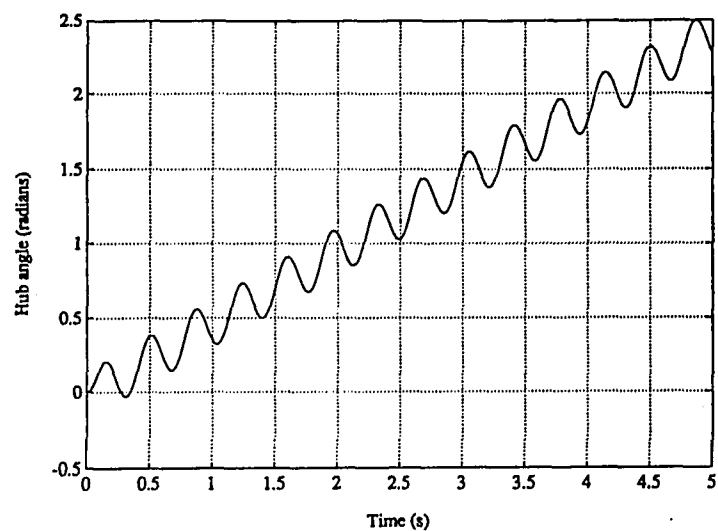


Figure 2.4: Hub Rotation due to Pulse Input, No Payload

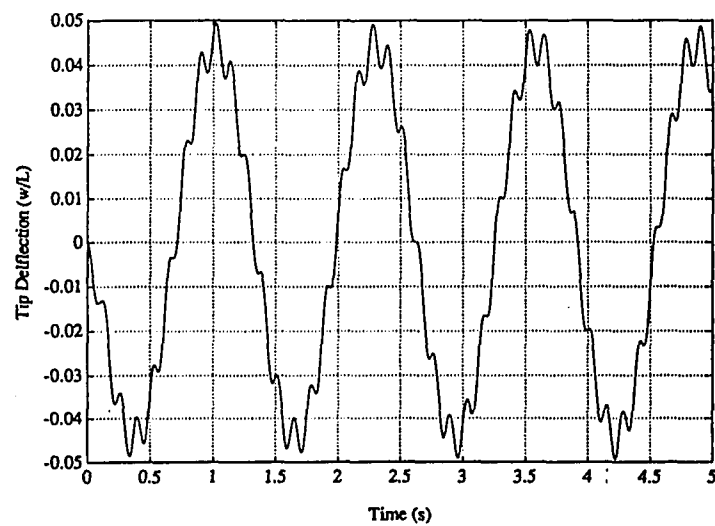


Figure 2.5: Tip Deflection due to Pulse Input, Payload: $\mu = \kappa = 0.5$

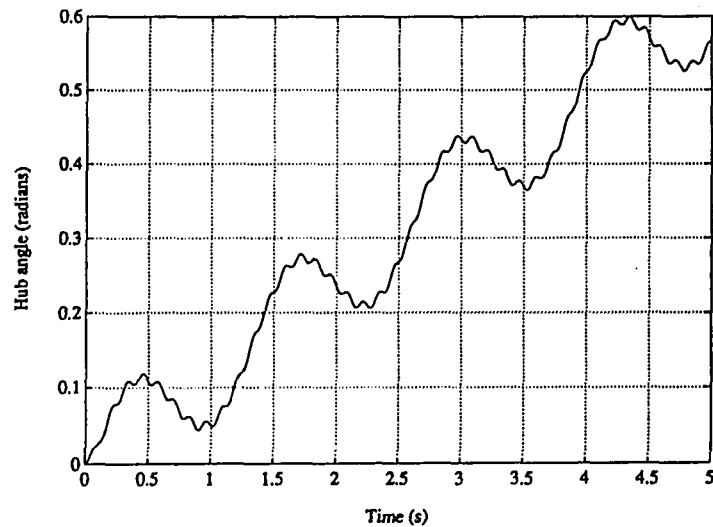


Figure 2.6: Hub Rotation due to Pulse Input, Payload: $\mu = \kappa = 0.5$

As can be seen, the response is different. The tip deflection has decreased, from 0.2157 to 0.0494, and the hub rotation is slower. This is expected, because as the payload increases, with the same magnitude of torque applied, the beam will become more difficult to move and the tip will vibrate less, especially the lower frequencies which have larger magnitudes and contribute more to the deflection of the beam. To illustrate the dominating effect of the lower frequencies, Figure 2.7 shows two responses. The solid line in the graph shows the response when the arm is modeled with 4 flexible modes, and the dashed line shows the response of the beam if only one flexible mode is included in the model. This clearly shows that the first flexible mode contributes the most to the tip deflection. However, the higher frequencies have an increased effect on the beam dynamics when compared to the no payload response, and this can be seen in the response. The no payload response

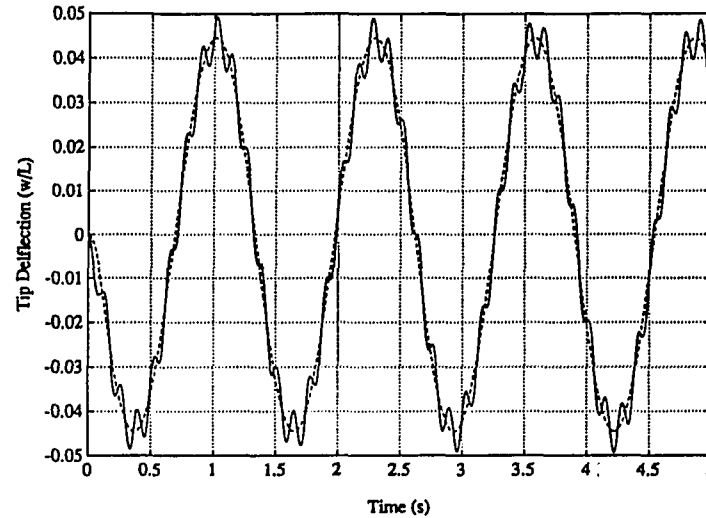


Figure 2.7: Tip Deflection Comparison, 1 Flexible mode (—) vs. 4 Flexible Modes (-), Payload: $\mu = \kappa = 0.5$

has contributions from the higher frequency modes as well, but these are so small in magnitude compared to the dynamics associated with the first flexible mode that they are undetectable in the response. For example, the peak value of the tip deflection that occurred with only one flexible mode modeled in the system with no payload was 0.2152, while the maximum tip deflection with four flexible modes was 0.2157. However, the importance of including more than one flexible mode in the model when a payload is attached to the arm is clearly seen in Figure 2.7.

Despite the noticeable effect resulting from the inclusion of the higher frequency modes, the maximum tip deflection will continue to decrease with larger payloads. One could think of the case of an infinite payload, in which the tip of the beam would be fixed, so the tip would not show any deflection and the hub would not be able to move freely anymore. This declining trend is seen in Figure 2.8, which shows

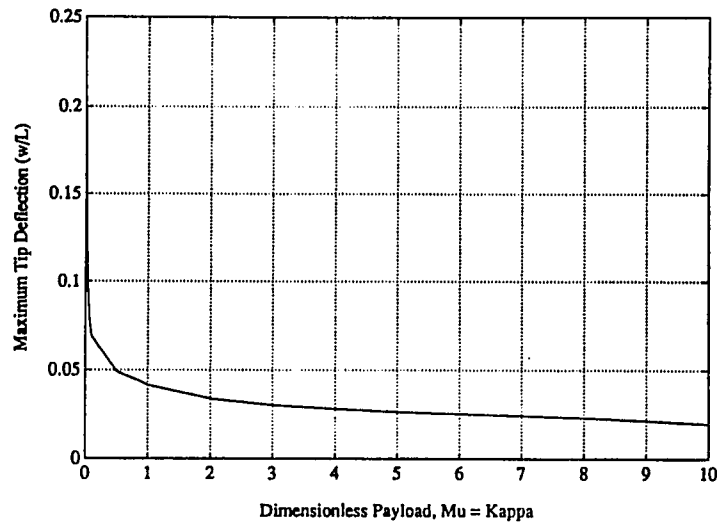


Figure 2.8: Maximum Tip Deflection with respect to Payload

the plot of maximum tip deflection from many simulations of different payloads..

Figure 2.9 shows a closer view of the trend in the range of small payloads. It is clearly seen that even small payloads cause a sharp decrease in the magnitude of the tip deflection for a step input. The dynamics of the beam are obviously considerably affected by the payload of the arm, and the control synthesis procedure that is used should either be robust enough to tolerate payload changes or adaptive with respect to the payload being carried to be effective in accomplishing the objectives it was designed for.

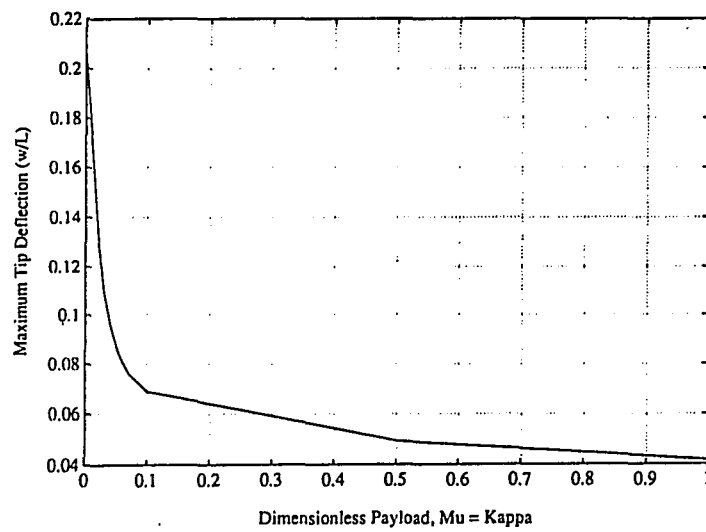


Figure 2.9: Maximum Tip Deflection with respect to Payload (Enlarged Scale)

CHAPTER 3

NEURAL NETWORK CLASSIFICATION APPROACH FOR PAYLOAD ADAPTIVE CONTROL

3.1 Introduction

For a rigid or flexible robot to be effective, it must be able to move a large range of payloads with accuracy. However, for flexible manipulators, the mass and inertia of the payload have a significant effect on the dynamics of the arm [50, 21]. This is evident from the two pulse responses shown in Chapter 2, which demonstrates that for the same input, the output could be quite different for different payloads at the tip of the robot. Thus, it is important to know or have a close approximation of the payload in advance, in order to apply the correct control input to produce the desired response of the arm. The payload, however, may not always be known, and even if it is known, it can be quite cumbersome and inefficient if the control has to be changed for each payload. One way to overcome this problem would be to design the control for the worst case, usually the heaviest payload, and use that control for all of the other lighter payloads. This has been done with rigid robots,

but of course the accuracy and efficiency is compromised for the lighter payloads. For flexible manipulators, designing for the worst case is not always desirable. The tip displacement can become larger for different payloads as also shown in the pulse response. A method for identifying the payload on-line quickly and accurately is a better solution to this problem.

Leahy et al. [35] have developed such an identification scheme and a corresponding adaptive control which uses the estimated payload for rigid robots. The PUMA 560 robot was tested with different payloads and controlled along a reference trajectory. The errors due to payload differences from the nominal were then measured and fed to a neural network, which classified the payload into one of four categories using the errors as inputs. The payload classification of the neural net was then used to update the computed torque control at that time to reflect the current payload of the arm. It is desired to extend this approach to flexible manipulators.

This chapter describes the use of neural nets to classify payload and provide an adaptive control for flexible or lightweight manipulators. First, the control methodology is given. A regulator is designed using a pole-placement technique and the effect of the payload on the response of the manipulator while under the regulator control is shown with a simple example. Next, the design of the test control to provide the dynamical information to the neural network is described. The patterns derived from this information are then used to train the network and a suitable architecture for accurate identification is found. Finally, the results of simulations

showing the effectiveness of the neural network-based adaptive control that was implemented are presented.

3.2 Control Methodology

For efficient use of a flexible arm, the vibration of the tip must be damped, while at the same time moving the tip to a desired location. If all states have zero values, the arm is at rest and is not vibrating. Therefore a regulator control lends itself well to flexible manipulators. A standard approach to classic regulator synthesis is pole placement design. In this case, the eigenvalues or poles located on the $j\omega$ -axis which are associated with the flexible modes must be shifted into the left-half plane to assure asymptotic stability of the state trajectories.

Since the linearized model that represents a single link flexible manipulator was found in Chapter 2 to be both controllable and observable, the poles can be placed by simply using state feedback of the observed states, with appropriate gains for the desired pole locations. Thus the control would be of the form

$$u = -Kx, \quad (3.1)$$

where K is a vector containing the feedback gains with which to place the poles at the desired locations. These gains can be computed by a simple straightforward procedure as given in [16]. The system is transformed into the equivalent controllable-canonical form, (A_c, B_c) , and then by comparison of coefficients, the feedback vector K_c is calculated and is then transformed back into the original

system coordinates. However, the transformation matrix is

$$P = \overline{U}_c U_c^{-1} \quad (3.2)$$

where U_c is the controllability matrix of the given system model and \overline{U}_c is the controllability matrix associated with the transformed system (A_c, B_c) . From the discussion given in the previous chapter, it is known that P can't be found accurately. Thus a different design procedure must be used to solve for K .

Kautsky, Nichols and Van Dooren [30] present four methods for computing robust solutions for the multi-input state-feedback pole placement problem (of which a special case is the single-input system, which is what the flexible manipulator is). The basic idea underlying these algorithms is to choose eigenvectors corresponding to the required eigenvalues such that the matrix of eigenvectors is as well-conditioned as possible. The multiplication of the A matrix is avoided by using QR decomposition to transform the system. MATLAB has a function which uses this technique and checks the accuracy of the calculated feedback vector with the desired pole locations. Therefore, it is assured that the feedback vector is calculated correctly.

3.2.1 Control Objectives

Now that the poles of the system can be placed accurately by feedback, the next issue is where to place those poles. The main advantage of a flexible manipulator over a rigid one is its lightweight properties which allow it to be moved faster with lower applied torque and this should be exploited by the control strategy to

move the manipulator quickly, while still minimizing the effects of the flexibilities introduced.

To gain an intuitive feel for how the pole locations affect the response of the arm with different payloads, consider the following example. Two manipulator systems will be used, one carrying no payload and the other carrying a payload of $\mu = \kappa = 1.5$, and will be named as below for clarity:

M_0 : CIRSSE Flexible Manipulator, carrying no payload

$M_{1.5}$: CIRSSE Flexible Manipulator, carrying a payload of $\mu = \kappa = 1.5$.

The parameter values of the CIRSSE flexible manipulator are shown in Table 2.1. Two state feedback controls will be calculated, one based on the parameters corresponding to M_0 and one based on the parameters corresponding to $M_{1.5}$. Thus, the payload condition of the arm is assumed known for the synthesis of the control for each manipulator system. It is desired to place the poles of both M_0 and $M_{1.5}$ at the same location, so that the responses of the manipulators with the same desired dynamics can be compared, and the effect of the payload can be seen. The pole locations are selected such that the real part of all the poles are located at -1 for both payload conditions. It should be emphasized that this choice of poles is an arbitrary one, and it was only desired to place the poles in the LHP to ensure an asymptotic convergence of the state to zero. These desired closed loop eigenvalues require the following feedback vectors in the two cases: first, for M_0 ,

$$K_0 = \begin{bmatrix} 0.5324 & 0.7709 & -4.2841 & -13.6899 & -36.0126 \end{bmatrix}$$

$$\begin{bmatrix} 0.8905 & 0.2619 & -1.0427 & -2.0798 & -3.4654 \end{bmatrix} \quad (3.3)$$

and for $M_{1.5}$,

$$K_{1.5} = \begin{bmatrix} 1.7715 & 3.4361 & -1.8588 & -7.1291 & -20.7510 \\ 3.2649 & 0.8696 & -0.4125 & -1.0675 & -2.1060 \end{bmatrix}. \quad (3.4)$$

The magnitudes of the gains are as one would expect. The rigid mode feedback gain in $K_{1.5}$ is much larger than the corresponding gain in K_0 , and this is required in order to be able to move the hub of $M_{1.5}$ at the same speed as the hub of M_0 . One interesting note is that higher frequencies require larger feedback gains in the no payload case than in the case of the system with payload.

Simulations were performed using each of these controls on M_0 and $M_{1.5}$. In this way, the consequences of not knowing the payload accurately when the feedback gain vector is designed can be seen. The results of the simulations are shown in Figures 3.1, 3.2, 3.3, and 3.4. Examination of these results gives much insight into the tradeoffs that must be considered when choosing the pole locations. First, comparing Figures 3.1 and 3.3, it is clearly seen that when the real parts of the pole locations are equal in both cases (i.e., when M_0 is controlled using K_0 and $M_{1.5}$ is controlled using $K_{1.5}$, the real values of all the poles of each system are located at -1), the system carrying a heavier payload, $M_{1.5}$, produces a larger transient maximum tip deflection. This is an intuitive result. Since the two systems must converge at the same rate, a larger torque must be applied to the manipulator carrying the payload. The inertia of the tip resists the applied torque, and hence,

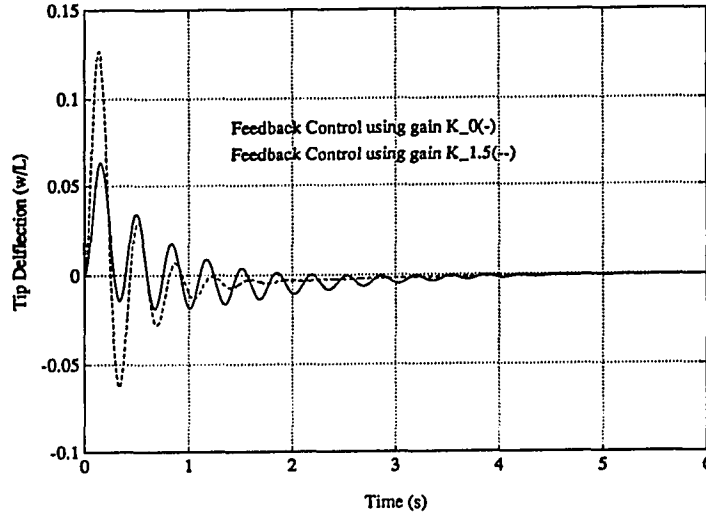


Figure 3.1: Tip Deflection of Flexible Manipulator M_0

the larger the inertia associated with the payload, the larger the tip deflection.

In Figure 3.1, it is seen that the tip deflection is much greater when the control $K_{1.5}$ is used. When this control is used, the closed-loop poles of the unloaded manipulator M_0 are

$$\Lambda_0 = \begin{bmatrix} -0.61 \pm j359.87 \\ -0.51 \pm j218.33 \\ -0.39 \pm j112.92 \\ -2.88 \pm j17.54 \\ -9.49 \\ -0.58 \end{bmatrix} \quad (3.5)$$

Thus the closed loop system has deeper LHP poles for the first flexible mode, which usually has the greatest magnitude of the flexible modes (as shown in Chapter 2), and for the rigid mode, one pole was moved deeper into the LHP, while the other migrated to the right. All other poles were also slower than the poles of the closed

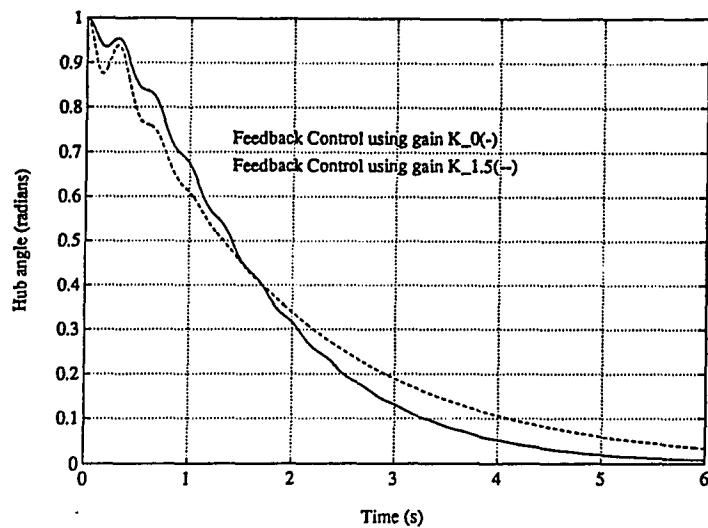


Figure 3.2: Hub Rotation of Flexible Manipulator M_0

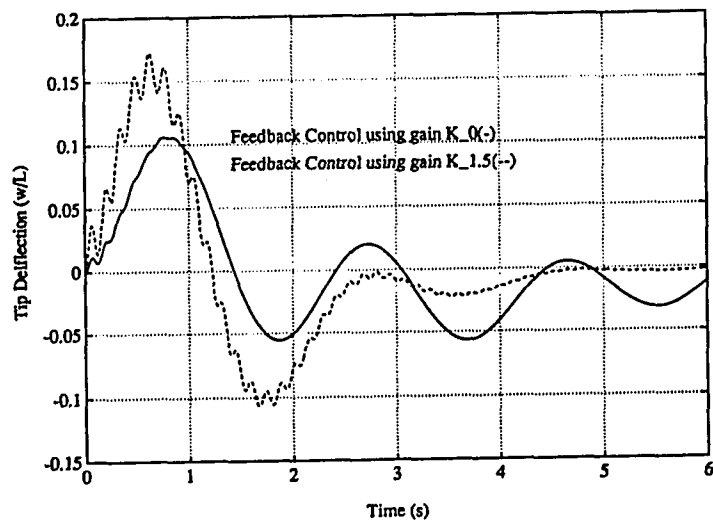


Figure 3.3: Tip Deflection of Flexible Manipulator $M_{1.5}$

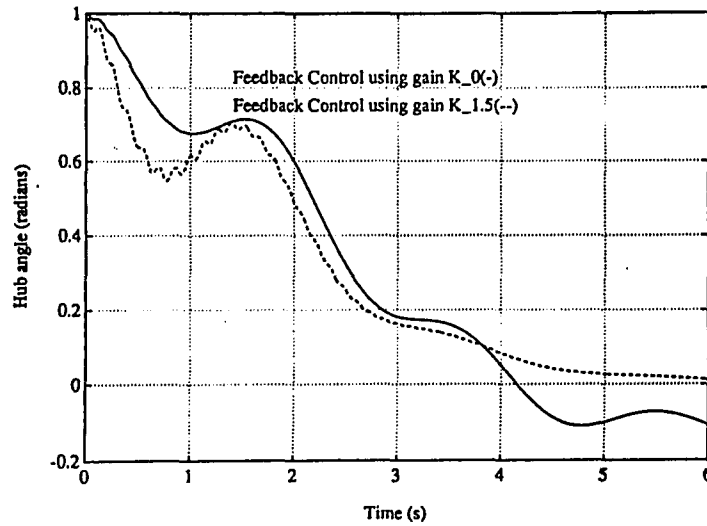


Figure 3.4: Hub Rotation of Flexible Manipulator $M_{1.5}$

loop system (all at -1) which used the control (with gain K_0) based on an assumption of no payload. Since the first flexible mode has stronger damping, the tip deflection goes to zero more quickly with this control. However, the hub still converges slower because although one rigid mode pole has much deeper placement, the effect of the other pole associated with it slowed the response.

As can be seen, there is definitely a different output for the two systems having the same poles. It is also seen that placing the poles deeper in the LHP, especially those associated with the rigid modes, is not a good idea. If this is done, the tip deflection increases, which is not desired. If the tip vibrations become excessive in magnitude, structural damage could result, the payload could be dropped, or the manipulator could strike something close to its path such as a wall. All of these consequences must be avoided and hence the speed of the hub convergence is limited

by how much tip deflection can be tolerated. This criterion varies with the specific application, as does the speed required of the manipulator. Hence, these factors should be considered before selecting the pole locations.

One way to insure that the tip deflection is kept below a certain bound would be to design a controller which guarantees the fastest convergence for the heaviest payload that the manipulator will carry in its tasks, while still remaining below the bound on the tip deflection. This method, however, is very inefficient because the speed of the arm would be limited by the heaviest payload, which cannot have poles as deep in the LHP. The robot would be able to accomplish a larger volume of work if the exact payload was known and the control was adjusted accordingly for each payload to achieve a nominal response. For each payload, the poles could be placed as deep in the LHP as the tip deflection criterion allows.

As stated earlier, the amount of tip deflection which can be tolerated for safe operation varies with each specific application. For all the simulations reported in this thesis, the tip deflection was always limited to a maximum of 0.1, in terms of the non-dimensional distance of the model. This means that the deflection was always less than or equal to 10% of the length of the beam. Thus the stated control objectives of all the simulations which will follow are:

Control Objectives for Flexible Manipulators

1. The tip deflection must never be allowed to exceed 10% of the length of the beam.

2. The hub is to be rotated as fast as possible consistent with meeting Objective 1.

3.3 Identification of Payload

The example of the previous section clearly shows that a payload identification scheme would greatly improve the operation and efficiency of the flexible manipulator. Identification has been primarily attempted by using adaptive methods by previous researchers, both in the frequency domain [68, 78], and in the time domain [68, 55, 66]. However, the time required to identify the payloads turns out to be quite long. The study done by Rovner and Cannon [55] required four seconds to identify the system parameters, and although the frequency domain techniques used by Tzes and Yurkovich [68] perform a more accurate estimation of the parameters, the computational burden is increased and the method still requires a lot of dynamic data of the beam. This is required for every payload change, and hence the robot must be taken off line to get the required data, or a persistently exciting control that performs no useful work must be applied to generate the input to the estimation scheme. In the example reported in [68], the beam was excited for 6 seconds before implementing the control.

Therefore, the efficiency gained by knowing the correct payload would be neutralized by the time required to perform the identification when using these methods.

However, the general adaptive algorithm outlined by Rovner and Cannon is useful in this thesis.

If, instead of using adaptive techniques, the response of the tip is considered as a distinct pattern for each payload, neural networks could be employed to quickly identify the payload and implement the correct control. Neural nets have been shown to be extremely quick in the solution of pattern recognition problems, and many advances have been made in recent years to improve the operation and practicality of these networks. For this reason, they would be a logical choice to speed up the identification of the payload.

In the previous section, it was seen that the tip displacement varied with payload when a feedback control was applied to the beam. It is therefore desirable to use this property in order to find some distinguishable classes of payload to be identified by a neural network. The range of possible payloads can vary with the needed application, and is chosen here for illustrative purposes to range from no payload to $\mu = \kappa = 1.5$. The hub is required to move a specified distance starting from an initial position to a final destination. Without loss of generality, this final destination could be chosen to be equal to zero. In order to provide as much information in the patterns as possible, the vibrational dynamics of the beam are to be excited in a manner similar to that used in the adaptive techniques [55, 68]. But instead of using a control which does no useful work in this time, the hub angle is still moved toward the desired position. Thus, a control referred to as the test control is designed to

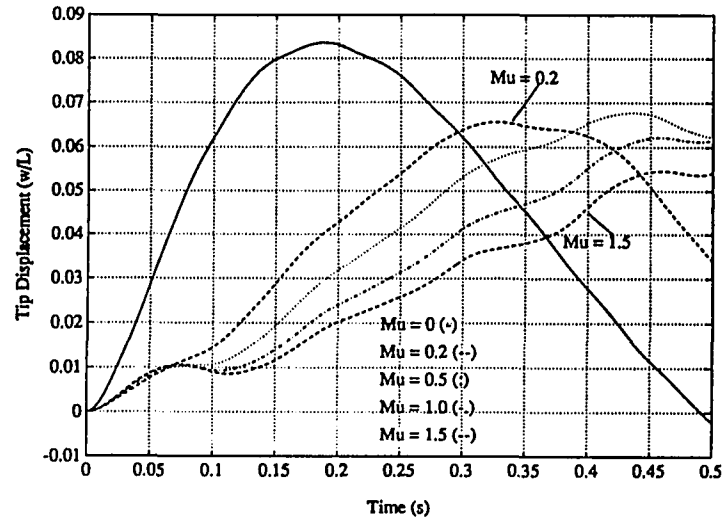


Figure 3.5: Tip Response Under Test Control

move the eigenvalue associated with the rigid mode into the left half-plane with all other flexible modes left on the $j\omega$ -axis. This can be accomplished by simply feeding back the hub angle θ , and the hub rotational speed, $\dot{\theta}$, since for a rigid manipulator, the rigid mode corresponds to this output. Therefore, pole placement techniques can be used to position the eigenvalues of this assumed rigid system which neglects all of the flexible modes. The magnitude of the pole associated with the rigid mode is kept small in order not to exceed the specified bound on the tip displacement.

The results of a 0.5 second simulation using this test control are shown in Figure 3.5 for several different payloads. It is clearly seen in Figure 3.5 that the no payload case behaves much differently than the cases with payload, and so it would seem that it could be considered a separate class. Within the payload cases, the ideal situation would be that in which all payloads are identified separately.

However, this is an impractical solution to the payload identification problem, as a reasonably sized neural network could never be trained to identify an infinite number of classes. Therefore, two classes are chosen for payloads, one for small payloads and one for large payloads, besides the case of no payload ($\mu = 0$). As it can be seen, the boundary between these two payload classes is not easily determined. Therefore, specific features of the responses were to be identified, in order to better quantify the differences between the three payload categories.

Many features were examined, including examining the frequency spectra of the responses with Fast Fourier Transforms, and calculating the average value of the tip displacement, standard deviation of the tip displacement, and the maximum value of the tip displacement. These features were plotted against one another to show possible groupings of payloads, if any. The most distinct class separation was found with the plot of the mean value of the tip deflection versus the maximum tip deflection. The three classes could be clearly separated by simple hyperplanes, as shown in Figure 3.6, so this feature map was used to determine the classes. The boundaries specifying the three classes were chosen to be:

Class 0 No payload

Class 1 Small Payload, Payload range : $0.2 \leq \mu = \kappa \leq 0.6$

Class 2 Large Payload, Payload range : $0.8 \leq \mu = \kappa \leq 1.5$.

Notice that the classes have some separation, and do not include all payloads (for instance, $\mu = \kappa = 0.7$). This is necessary, because if all payloads were represented,

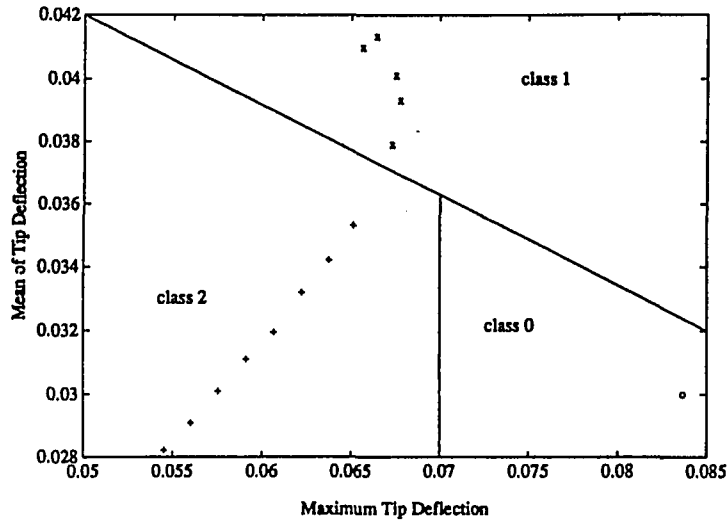


Figure 3.6: Mean Value of Tip Deflection vs. Maximum Tip Deflection, $\kappa = \mu$

it would become difficult to choose where the hyperplanes of Figure 3.6 would lie. In addition, once the hyperplanes were selected (arbitrarily), the neural network would have some difficulty classifying payloads when presented one which is close to the boundary between two classes. It is therefore necessary to have some knowledge of the types of payloads and ranges of mass and inertia associated with the payloads before training the neural network to classify them, in order to achieve accurate classification results.

In the simulations reported in this thesis, the inertia of the payload, κ , is always chosen to be equal to the mass of the payload, μ . This is an arbitrary choice for illustrative purposes only and in fact, in most situations, $\kappa \neq \mu$. Another set of simulations was done with $\kappa \neq \mu$ to show that distinct classes of payload may be found from the tip deflection data as well. Figure 3.7 shows the pattern clusters

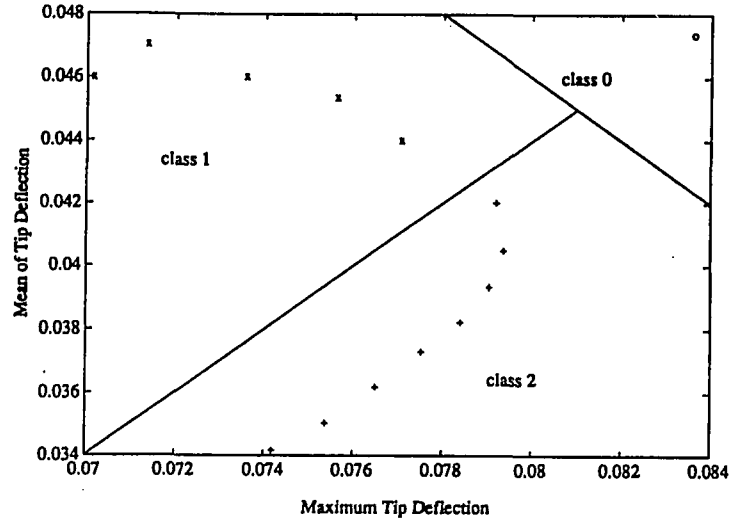


Figure 3.7: Mean Value of Tip Deflection vs. Maximum Tip Deflection, $\kappa = \frac{1}{2}\mu$

when $\kappa = \frac{1}{2}\mu$. Although the patterns comprising each group differ between the $\kappa = \mu$ and $\kappa = \frac{1}{2}\mu$ cases, it is clearly seen that the payloads can be grouped into classes for this case as well. Hence the inertia of the payload and the mass of the payload are not required to be equal in order to obtain distinct classes for identification.

Now that the classes have been determined, control feedback gain vectors can be found for each class to satisfy the two control objectives as given in Section 3.2.1. One vector is found for each payload class. The tip deflection criterion must not be violated when using these controls to operate a manipulator carrying any payload which is a member of the specific class that the control feedback vector was designed for.

In Section 3.2.1, it was seen that if the desired pole locations were selected to be the same for manipulators with different payloads, the one carrying the heavier payload had the larger transient tip deflection. Therefore, to find the feedback gain vector for a class, the parameters corresponding to the manipulator carrying the heaviest payload in that class should be used. The response of the arm carrying the heaviest payload will have the maximum tip deflection for that class, and if it satisfies the tip deflection criterion, all of the lighter payloads within that class will satisfy the criterion as well.

Thus, a worst case approach is used to synthesize the control for each class. As a result, the lighter payloads in each class will have less than maximum convergence rates, but this is more efficient than having only one class and using the gains based on the heaviest payload only.

It should be emphasized that consideration of three payload classes(classes 0, 1 and 2) in the work reported in this thesis is only for illustrative purposes and for proving the concept underlying the use of a neural network for the required classification. The number of classes can be increased with a corresponding sharper control performance resulting, at the expense of employing a larger neural network and the consequent increase in training complexity.

3.4 Neural Network Architecture and Training

At this point, the patterns could be classified based on the specific bounds of the standard deviation and maximum deflection features of each class. However, this first requires the computation of each feature, and then a sequential check on both features must be done to determine which class the present payload is a member of. This sequential computation is quite inefficient when compared to the parallel computational abilities afforded by neural networks.

The neural network is trained off-line with patterns of payloads that the manipulator is likely to carry. The interconnection weights are selected during training such that the vector of inputs representing the 0.5 second trajectory generates the desired output class which selects the proper feedback gains to be used in the control. This is done with a far less computational price than the sequential method of checking bounds for each class, and so a neural network pattern classification approach is clearly superior.

3.4.1 Multilayered Static Neural Networks

Many neural network models have been suggested in recent years, and the most commonly used networks for nonlinear mapping or pattern recognition tasks have been static multilayer networks. These networks consist of an input layer, an output layer, and a number of hidden layers. Each layer consists of several nodes, and each node is connected to all the nodes of the previous layer, as shown in Figures 3.8

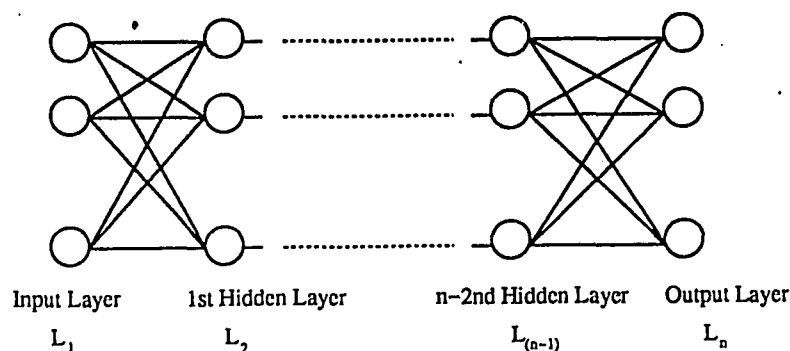


Figure 3.8: Static Multilayer Neural Network

and 3.9. There is some difference of opinion as to how a network architecture is defined. Some researchers disregard the input layer as a part of the architecture, since it only performs a linear computation [41]. Others include the input layer in counting the total number of layers describing the network [49], and the networks in this thesis will follow that definition. Thus, a network consisting of one input layer, one hidden layer, and one output layer will be referred to as a 3-layer network.

Each node in a layer computes an output as a function of all incoming signals. A linear node or neuron would simply output a weighted sum of the inputs, but the output of a nonlinear node is based on a nonlinear sigmoidal function. The use of nonlinear processing nodes has been shown to greatly improve the mapping capabilities of the static multilayer network. The sigmoidal function which typically

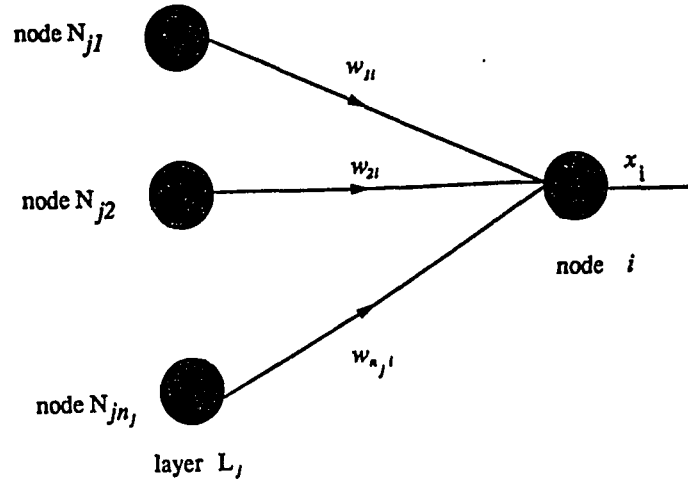


Figure 3.9: Nodal Connections in a Static Network

governs the output of a nonlinear node has been discussed in Chapter 1 and is shown in Figure 1.1. The function can be represented mathematically as

$$x_i = f(\alpha_i) = \frac{2}{1 + \exp^{-(\alpha_i + \theta_i)}} - 1 \quad (3.6)$$

with

$$\alpha_i = \sum_{j=1}^{n_j} w_{ji} x_j. \quad (3.7)$$

In the above equations, x_i is the output of node i , α_i is the sum of the inputs from the previous layer coming to node i , with w_{ji} being the weight associated with the interconnection of node j and node i , and n_j the total number of neurons in layer L_j . The parameter θ_i serves as a bias or threshold. If θ_i is positive, the sigmoidal function will shift to the left along the horizontal axis in Figure 1.1. For the nonlinear characteristic shown in Figure 1.1, the value of θ_i is zero.

3.4.2 Backpropagation Learning Algorithm

There are several training schemes for multilayer networks, but the one that is probably most popular is the backpropagation algorithm, introduced by Werbos in 1974 [74] and popularized by the work of Rumelhart, Hinton, and Williams [56]. It has been shown to be very successful with pattern recognition and deterministic problems [4, 22], and has been applied to a wide variety of tasks such as the exclusive OR problem [56], speech synthesis and recognition [51, 42], visual recognition [56, 44], and classification of radar and infrared images [38, 54].

The method employed in this approach is a supervised training procedure and feeds back the error at the output to the hidden layers, or “backpropagates” the error. During this process, the weights of the network are adjusted according to the rule

$$\Delta w_{ji} = -\mu \frac{\partial E_t}{\partial w_{ji}} = \mu \delta_j x_i \quad (3.8)$$

where Δw_{ji} is the change or adjustment in the weight between node j and node i , μ is a constant, x_i is again the output of the i^{th} node, and $\delta_j = x_{dj} - x_j$, with x_{dj} being the desired output of node j . It is shown that the backpropagation algorithm, also called the delta rule, is in fact a gradient descent learning algorithm for minimizing E_t , where

$$E_t = \sum_{k=1}^{n_o} \frac{1}{2} (y_{dk} - y_k)^2. \quad (3.9)$$

In this equation, n_o is the number of nodes in the output layer, and y_{dk} and y_k are the desired and the actual outputs of the k^{th} node of the output layer, respectively.

Since the desired outputs are known explicitly only at the output layer, the delta rule should be expressed as a function of the error at the network output, and not as in (3.8), since the desired output of a hidden node is not always known. The delta rule can be expressed to take this into account according to the following derivation. From (3.8), using the chain rule and the definition from (3.7), the partial derivatives can be evaluated as

$$\frac{\partial E_t}{\partial w_{ji}} = \frac{\partial E_t}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial w_{ji}} = \frac{\partial E_t}{\partial \alpha_j} \frac{\partial}{\partial w_{ji}} \sum_{i=1}^M w_{ji} x_i = -\delta_j x_i . \quad (3.10)$$

Therefore,

$$\delta_j = -\frac{\partial E_t}{\partial \alpha_j} . \quad (3.11)$$

This equation can also be expanded using the chain rule in the form

$$\delta_j = -\frac{\partial E_t}{\partial x_j} \frac{\partial x_j}{\partial \alpha_j} . \quad (3.12)$$

Since $x_j = f(\alpha_j)$ as in (3.6),

$$\frac{\partial x_j}{\partial \alpha_j} = \frac{df(\alpha_j)}{d\alpha_j} = f'(\alpha_j) . \quad (3.13)$$

Hence an expression for $\frac{\partial E_t}{\partial x_j}$ can be found for two cases: first when the node is an output node, and second when the node is a hidden node. When the node under consideration is an output node , then

$$\frac{\partial E_t}{\partial x_k} = -(y_{dk} - y_k) , \quad (3.14)$$

and therefore

$$\delta_k = (y_{dk} - y_k) f'(\alpha_k) \quad (3.15)$$

for any output layer node. However, if the node is in a hidden layer, $\frac{\partial E_t}{\partial x_j}$ cannot be evaluated directly. An equivalent expression made up of known terms and those that may be evaluated can be derived as follows:

$$\begin{aligned}
 -\frac{\partial E_t}{\partial x_j} &= -\frac{\partial E_t}{\partial(\sum_k \alpha_k)} \frac{(\partial \sum_k \alpha_k)}{\partial x_j} \\
 &= \sum_{k=1}^{N_k} \frac{\partial E_t}{\partial \alpha_k} \frac{\partial \alpha_k}{\partial x_j} \\
 &= \sum_{k=1}^{N_k} \frac{\partial E_t}{\partial \alpha_k} \frac{\partial}{\partial x_j} \left[\sum_{j=1}^{N_j} w_{kj} x_j \right] \\
 &= \sum_{k=1}^{N_k} \frac{\partial E_t}{\partial \alpha_k} w_{kj}, \quad j = 1, 2, \dots, N_j.
 \end{aligned} \tag{3.16}$$

Thus,

$$\frac{\partial E_t}{\partial x_j} = \sum_{k=1}^{N_k} \delta_k w_{kj} \tag{3.17}$$

and therefore

$$\delta_j = f'(\alpha_j) \sum_{k=1}^{N_k} \delta_k w_{kj} \tag{3.18}$$

where δ_k is the delta of the succeeding layer computed earlier. This is the result that is needed. Starting with the output layer, δ_k can be evaluated using (3.15) and then the errors can be propagated to the lower layers. The nonlinear sigmoidal function of (3.6) can be differentiated with respect to the input as follows to arrive at a simple expression for $f'(\alpha_k)$ as

$$\begin{aligned}
 f'(\alpha_k) &= \frac{2e^{-(\alpha_k + \theta_k)}}{(1 + e^{-(\alpha_k + \theta_k)})^2} \\
 &= \frac{2}{(1 + e^{-(\alpha_k + \theta_k)})} \left[1 - \frac{1}{(1 + e^{-(\alpha_k + \theta_k)})} \right] \\
 &= \frac{1}{2}(1 - x_k^2).
 \end{aligned} \tag{3.19}$$

The bias terms θ_j can also be learned in the same way [49]. They can be imagined as simply being another weight from a unit that always has an output of unity. The selection of the updating gain μ should be carefully done; its value should be kept small, or oscillation (or even divergence) in learning may result. Rumelhart, Hinton, and Williams [56] suggest adding a momentum term, γ , to smooth out these oscillations. With this addition, the delta rule is modified as

$$\Delta w_{ji}(n+1) = \mu \delta_j x_j + \gamma w_{ji}(n) . \quad (3.20)$$

The term associated with the constant γ , $w_{ji}(n)$, is the change undertaken by the weight in the previous iteration. Thus, the change at iteration $n+1$ should be somewhat similar to the change undertaken at step n , and the rate of change is limited to some degree. It has been shown that while γ tends to dampen oscillations, it also slows the learning rate [49].

Also note that since this is a gradient descent learning algorithm, there is the possibility of becoming trapped at a local minimum of the error space, causing oscillation and no further reduction in the error, E_t . This is one drawback of the backpropagation algorithm. Lippman [41] suggests allowing extra hidden units, lowering the gain term μ , and making many training runs with different sets of random weights as ways of avoiding getting stuck in local minima. Despite the occurrence of this problem, the backpropagation algorithm has been found to have generally good performance.

The various steps in the execution of the algorithm for processing the training set of patterns may now be summarized as follows [41]:

Step 1 Initialize weights: Set all weights and bias terms to random values. The net must not be allowed to start with a set of equal weights, or convergence is not possible [56].

Step 2 - Forward propagation: Present a vector of inputs and desired outputs to the network, and calculate the output y_o using (3.6).

Step 3 - Error Calculation: Compare the actual output y_o to the desired output y_d .

Step 4 - Weight Adjustment: Adapt weights by using (3.8), where δ_j is calculated using (3.15) if the node is in the output layer, or using (3.18) if the node is in a hidden layer. Proceed backwards beginning with the output layer.

Step 5: Go to Step 2 for presentation of next pattern.

The patterns can be a new vector presented every trial, or the same vector which is cycled through repeatedly until the weights have stabilized. When the weights have stabilized and the error is of acceptable value, the network is considered trained.

3.4.3 Network Training for Payload Identification

The architecture of the network needs to be found first. Unfortunately, there are no algorithms to date on choosing a suitable architecture, although in theory a 3-layer network (one input, one hidden, and one output layer) has been proven to be able to map arbitrary complex decision regions and can therefore separate populations of patterns even though they might be intermeshed spatially in the pattern space [41, 26, 17, 19]. There is also an uncertainty as to how many hidden nodes there should be in each layer. An excessive number of nodes can generate noise and increase the training complexity, but can provide fault tolerance [49]. Fewer numbers of nodes result in a poorer generalization performance, i.e., when the network is presented a pattern which is different from those in the training exemplars, the classification accuracy is worse than a similarly trained network which has more hidden layer nodes. As a practical matter, the selection of the architecture is usually accomplished by a method of trial and error, by progressively adding nodes until an architecture is found which satisfies the error requirement and is trained in a reasonable time period.

The neural network used in this thesis was simulated on the public domain software package PlaNet. Both networks with one and two hidden layers were experimented with, and the output layer was made up of nonlinear processing elements. The input was a two dimensional vector with elements the mean value of the tip

deflection and the maximum value of the tip deflection, obtained from the test control data. Note that while the features were computed from the raw data before being input to the network, the network can be generalized to identify the classes based on the raw data itself and not on the explicit features.

The output layer consisted of three nodes, one for each class. The desired outputs were defined to be an output of a magnitude of one from the node corresponding to the correct class, with the other nodes in the output layer giving a value of zero. Hidden layers were tested with numbers of nodes that ranged from 6 to 20, and the chosen architecture from these experiments was a network with one hidden layer consisting of 12 nodes. It was trained in 200,000 cycles through the training set, which consisted of 18 input-output training pairs of patterns.

The patterns corresponded to the responses of the manipulator with payloads at 0.1 increments of μ and κ within each class. The training exemplar set consisted of an equal representation for each class, that is, each class had the same number of patterns in the training set. This was to ensure that all classes were learned at an equal rate. This meant that some patterns were presented to the network for training more than once per cycle (such as the pattern for $\mu = \kappa = 0$).

The training was successful in that all patterns were learned and classified correctly. The payload class was identified by the node in the output layer with the maximum output value. This payload class was then used to update the control feedback gains.

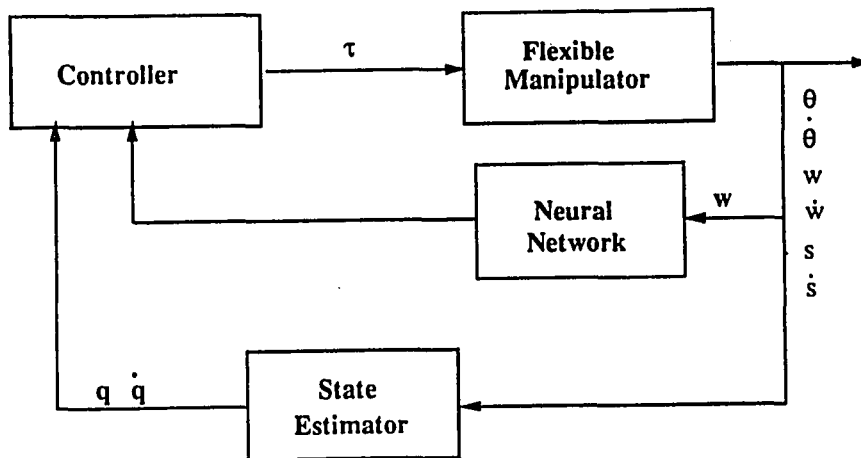


Figure 3.10: Neural Network-Based Control System

3.5 Performance of Neural Network-Based Control

Once the network is trained to identify the classes correctly, it could be used on-line to select the proper control gains corresponding to the load being carried by the flexible manipulator. The control gains are found by using the guidelines stated in Section 3.2.1 for each class and are stored. The neural network output determines which of the three vectors are selected and implemented. The closed-loop system is shown in Figure 3.10. In this architecture, the state estimator can be designed as described in [16], so the procedure is not covered here. It is shown in Figure 3.10 merely to ensure that all states are available for the controller.

The results of the simulation using the neural network-based control scheme is shown in Figures 3.11, 3.12, 3.13 and 3.14. These figures present simulations for the smallest payload, the largest payload, and an intermediate payload in each class.

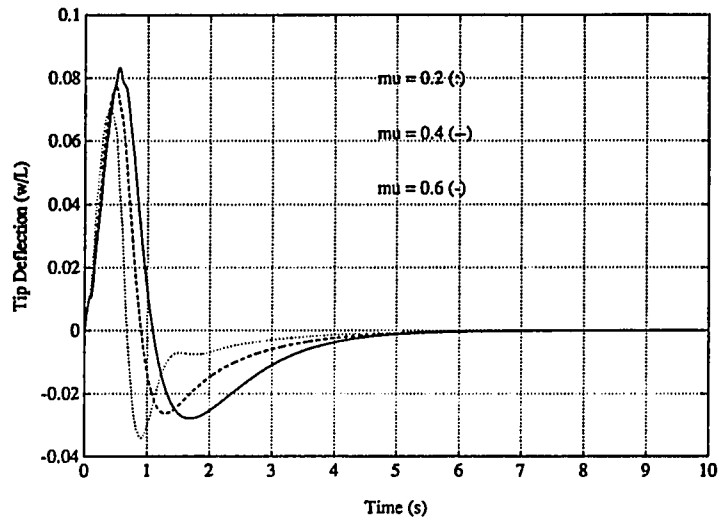


Figure 3.11: Class 1 Range of Tip Deflection

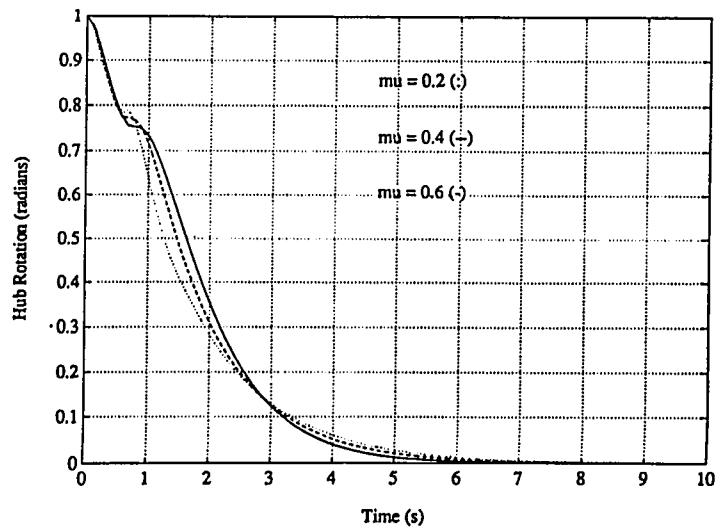


Figure 3.12: Class 1 Range of Hub Response

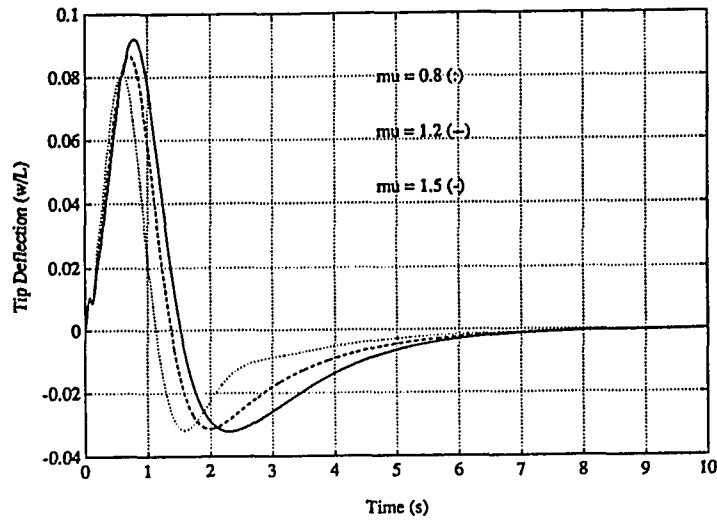


Figure 3.13: Class 2 Range of Tip Deflection

The robustness of the control needs to be particularly noticed. The tip deflection bound is preserved for the lightest payload in each class as well as the heaviest. This of course is because the control was designed with the heaviest payload in mind, which guarantees this result. Another consequence of this design approach is that the lighter payloads have hub rotations which are as slow or slower than in the case with the heaviest payload in each class, which will always happen if more than one payload is in the class.

Figures 3.15, 3.16, 3.17, 3.18, 3.19, and 3.20 display results with both the correct choice of control based on the output of the neural network and the consequences of a wrong choice of control. However, the neural network was accurate for all simulations of the payloads in correctly detecting the payload class, and the wrong control choices are forced only for illustrative purposes. The simulations used a

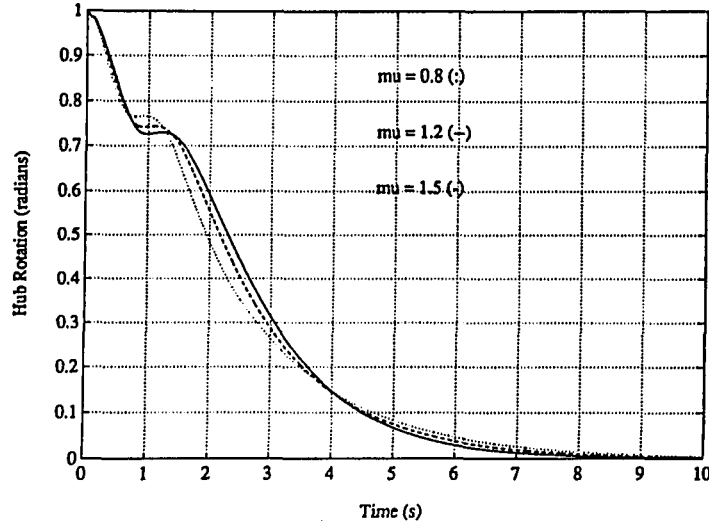


Figure 3.14: Class 2 Range of Hub Response

middle value in each class as the representative payload. Therefore, the value used for the class 1 payload was $\mu = \kappa = 0.4$, for class 2 it was $\mu = \kappa = 1.2$, and of course for class 0 it was $\mu = \kappa = 0$. There were numerous simulations done, but in the interest of brevity, only a few representative ones are shown.

In order to gain more insight from the responses of the arm under various controls, it is necessary to look at the pole locations for each case. The three gain vectors are

$$K_{Class\ 0} = \begin{bmatrix} 13.0973 & 98.3546 & -106.1394 & -114.4789 & -202.0499 \\ 7.1663 & 3.9911 & -1.9559 & -2.0357 & -3.4341 \end{bmatrix} \quad (3.21)$$

which places the closed-loop poles for the manipulator carrying the Class 0 payload ($\mu = \kappa = 0$) at

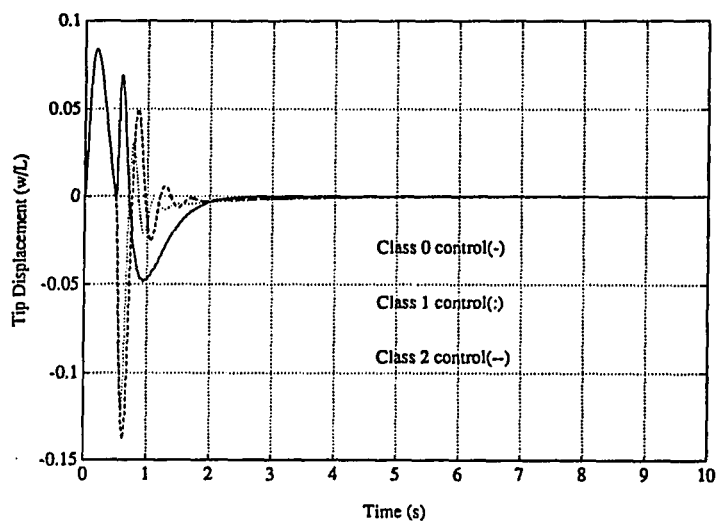


Figure 3.15: Tip Deflection - No Payload

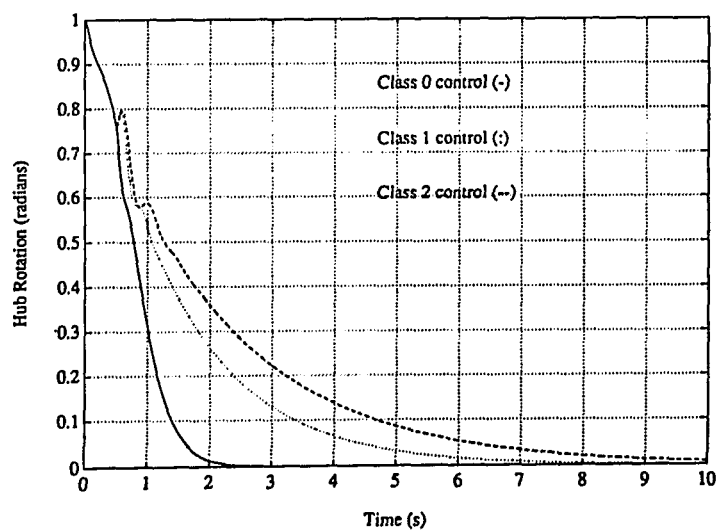


Figure 3.16: Hub Rotation - No Payload

$$\Lambda_{Class\ 0} = \begin{bmatrix} -1.0 \pm j359.87 \\ -1.0 \pm j218.33 \\ -2.0 \pm j112.92 \\ -21 \pm j18.23 \\ -3.7 \\ -4.5 \end{bmatrix}; \quad (3.22)$$

$$K_{Class\ 1} = \begin{bmatrix} 11.6013 & 71.5218 & -118.3814 & -104.5130 & -107.7610 \\ 17.1421 & 1.1057 & -3.8824 & -2.0613 & -2.0972 \end{bmatrix} \quad (3.23)$$

which places the closed-loop poles for the manipulator carrying the Class 1 payload ($\mu = \kappa = 0.4$) at

$$\Psi_{Class\ 1} = \begin{bmatrix} -0.99 \pm j225.57 \\ -1.94 \pm j119.48 \\ -9.42 \pm j50.25 \\ -4.71 \pm j3.16 \\ -17.85 \\ -0.89 \end{bmatrix}; \quad (3.24)$$

and

$$K_{Class\ 2} = \begin{bmatrix} 3.2928 & 17.1249 & -90.6244 & -251.2564 & -115.4626 \\ 7.1230 & 2.4458 & -6.0499 & -5.0582 & -2.0571 \end{bmatrix} \quad (3.25)$$

which places the closed-loop poles for the manipulator carrying the Class 2 payload ($\mu = \kappa = 1.2$) at

$$\Gamma_{Class\ 2} = \begin{bmatrix} -1.0 \pm j221.13 \\ -4.96 \pm j115.54 \\ -14.86 \pm j47.81 \\ -3.20 \pm j2.57 \\ -3.41 \\ -0.68 \end{bmatrix}. \quad (3.26)$$

Note that the odd pole locations (i.e. -3.41 and -0.68 of $\Gamma_{Class\ 2}$: why are the poles placed there instead of -3.4 and -0.7?) of $\Psi_{Class\ 1}$ and $\Gamma_{Class\ 2}$ occur because the feedback gain vectors $K_{Class\ 1}$ and $K_{Class\ 2}$ were designed for the heaviest payloads of those classes, or $\mu = \kappa = 0.6$ and $\mu = \kappa = 1.5$, respectively. When these gains are used with lighter payloads the poles “migrate” to seemingly odd values in the LHP, but as long as they remain in the LHP, this is perfectly acceptable. Keeping these observations in mind, and first looking at the simulation of the Class 0 case, (Figures 3.15 and 3.16), it is seen that the hub rotation and the tip deflection have both reached zero at approximately 3 seconds, under correct payload identification, or Class 0 control. The response of the arm when controls based on Class 1 and Class 2 payloads are used in this case is worse. For the Class 1 control, the bound on the tip deflection is exceeded, and the hub rotation is slower. This seems to contradict the intuitive feeling that the faster the tip moves, the larger the tip deflection. This, however is only true in a general sense. Let us look at the placement of the poles under Class 0 control and the corresponding locations for the Class 1 control:

$$\Lambda_{Class\ 0} = \begin{bmatrix} -1.0 \pm j359.87 \\ -1.0 \pm j218.33 \\ -2.0 \pm j112.92 \\ -21 \pm j18.23 \\ -3.7 \\ -4.5 \end{bmatrix}; \quad \Lambda_{Class\ 1} = \begin{bmatrix} -0.6 \pm j359.86 \\ -0.99 \pm j218.28 \\ -3.49 \pm j112.12 \\ -5.87 \pm j18.24 \\ -44.87 \\ -0.70 \end{bmatrix} \quad (3.27)$$

The higher frequencies are damped out approximately equally, but the pole locations for the lower frequencies are further to the left when Class 0 control is used. The main cause for the tip deflection being exceeded, however, is the large

magnitude of one of the rigid poles. The slower hub response is also mainly caused by the placement of the rigid poles. The low magnitude (-0.70) of one of the rigid poles causes the hub to move slower, despite the extremely large pole (-44.87) also associated with the rigid mode. From this it is seen that it is better to have the poles associated with the rigid mode (or in other words, the poles associated with the states q_0 and \dot{q}_0) be approximately the same in magnitude, so that both of the states q_0 and \dot{q}_0 go to zero quickly.

The response of the arm while under Class 2 control is similar to the Class 1 control response. Although the tip deflection is smaller, the price paid is a slower hub convergence. The pole locations for this control are

$$\Lambda_{\text{Class 2}} = \begin{bmatrix} -0.60 \pm j359.88 \\ -2.44 \pm j218.29 \\ -5.48 \pm j111.74 \\ -4.12 \pm j14.46 \\ -30.58 \\ -0.48 \end{bmatrix} \quad (3.28)$$

In this case, two of the flexible modes are damped more heavily than when using the Class 0 control, with the flexible mode associated with the highest frequency and the flexible mode associated with the lowest frequency being more lightly damped. However, instead of the rigid modes being to the right (or more lightly damped) than the Class 0 control scenario, they are much the same as in the case of Class 1 control, with one much larger pole and one much smaller pole. These pole magnitudes are not as large as those with Class 1 control, so it would seem that they should produce a smaller transient tip deflection than the Class 1 control. However, this is not the

case, as the transient tip deflection is larger. The relationship between these rigid poles is not clear, and this is one difficulty with the control design using state feedback.

The other two classes of responses (Figures 3.17, 3.18, 3.19, and 3.20) can be analyzed in the same manner. From the simulations it is seen that, in general, the Class 1 control produced the highest tip deflection when it was applied to payloads which were in Class 0 or Class 2. The Class 2 control produced the least tip deflection when it was applied to payloads which were members of Class 1 or Class 0. The general rule of thumb to follow in design is that the faster the hub is rotated, the larger the tip deflection becomes, when both poles associated with the rigid mode are selected to be approximately equal in magnitude.

The simulations clearly show that the adaptive control using payload identification is much more effective than using a fixed gain. The fixed gain that would be used for this system would be the Class 2 control feedback vector. Figures 3.15, 3.16, 3.17 and 3.18 clearly show the superiority of adapting the gains to the ones corresponding to the class of payload that the manipulator is carrying. In both of the lighter payload classes, the hub rotation and tip deflection converge much faster when using Class 0 or Class 1 controls. Since a robot is usually implemented in tasks which involve a lot of repetition, the time saved by adaptively controlling the robot based on the payload it is carrying can drastically improve its productivity and usefulness.

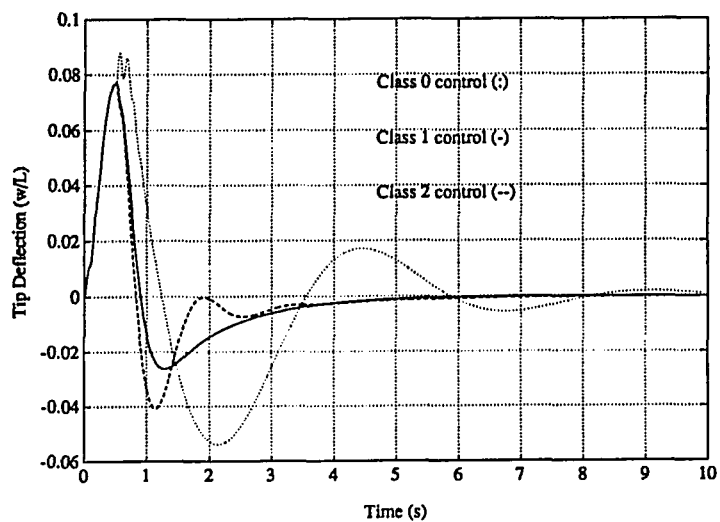


Figure 3.17: Tip Deflection - Payload $\mu = \kappa = 0.4$ (Class 1)

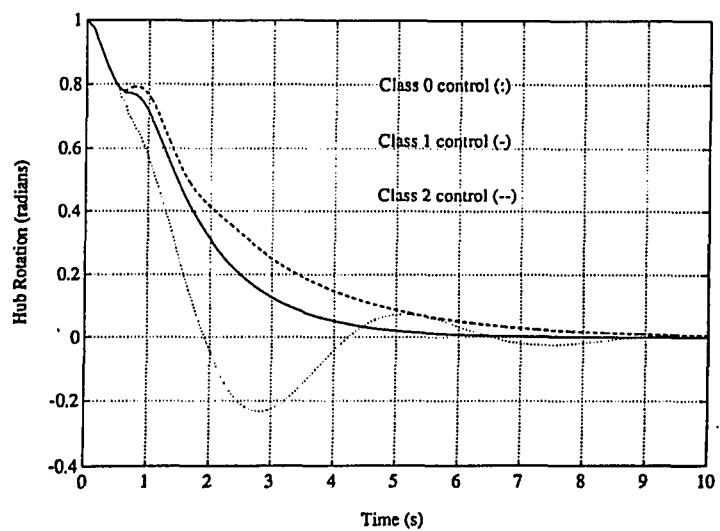


Figure 3.18: Hub Rotation - Payload $\mu = \kappa = 0.4$ (Class 1)

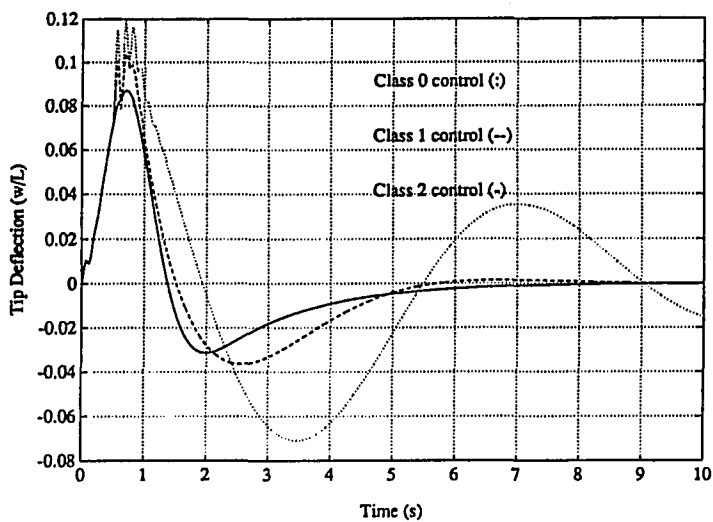


Figure 3.19: Tip Deflection - Payload $\mu = \kappa = 1.2$ (Class 2)

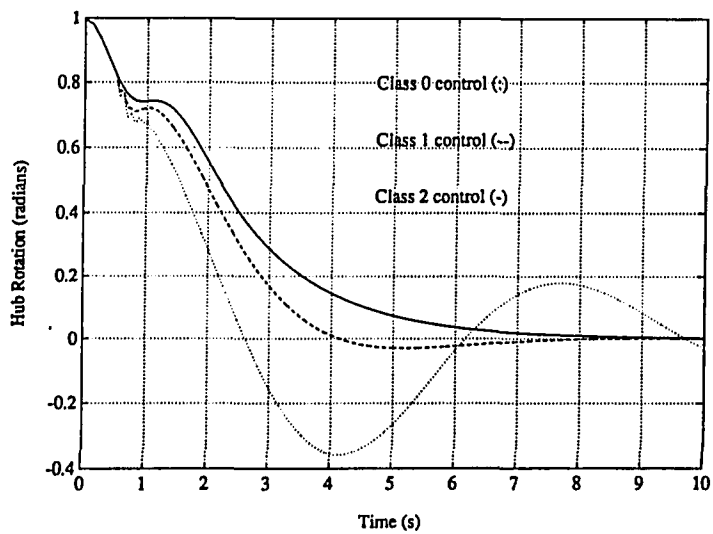


Figure 3.20: Hub Rotation - Payload $\mu = \kappa = 1.2$ (Class 2)

3.6 Implementation Issues

For practical implementation, there are several issues that should be considered. The first is the implementation of the observer once the payload class has been identified by the neural network. After the class has been identified, an observer must be implemented. This observer must be robust enough to accurately estimate the states for all payloads in the range of the class. This may not be practical using a standard observer such as those discussed in [16], and therefore more novel techniques using nonlinear methods may have to be investigated to design this robust observer, such as those discussed in [48, 80]. However, as stated before, this thesis is mainly concerned with the controller design and not observer design, and therefore this must be reserved for future research.

Once the observer is implemented after the payload class has been identified, the controller would not be implemented until the estimated states had sufficient time to converge to their true values. Therefore, the time sequence for implementation (as shown in Figure 3.21) would be as follows. At time t_0 , the test control is applied to the arm, exciting the beam dynamics. After a period of t_{ID} seconds, the raw data from that time period is sent to the neural network. The neural network then identifies the class of payload, and picks the proper observer parameters corresponding to that class. The observer is then activated and allowed t_{obs} seconds to converge to the actual state values. During this time of t_{obs} seconds, the test control is still being applied. After the convergence of the estimator, the controller

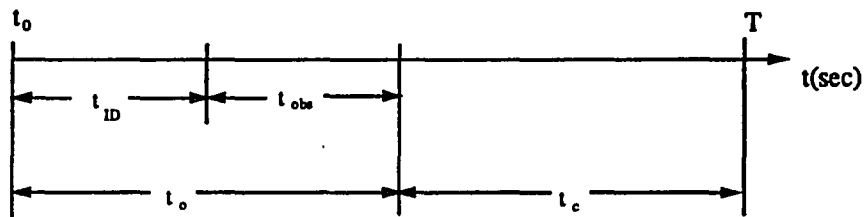


Figure 3.21: Time Sequence for Practical Implementation

corresponding to the payload class identified by the neural network is put on-line, replacing the test control and remains on-line for t_c seconds, or in other words until the payload is changed and identification must take place once again. Thus the total time that the test control is applied is t_o seconds, as shown in Figure 3.21. It is important to note that this figure is made under the assumption that three sets of observer parameters are available, one for each class, and the set of parameters corresponding to the correct payload class must be implemented to guarantee good estimation of the states.

Another issue is the effect of payload changes during operation. For most applications, the manipulator is rarely going to have its payload increased unexpectedly during operation, but there is always a possibility that the payload could be dropped. If this happens after the payload identification has been completed, there

must be a way of stopping motion and starting the procedure over. This could be implemented with what is known as a cross-fire sensor on the gripper of the flexible manipulator. This type of sensor simply fires a signal from one finger of the gripper to the other finger which has a sensor to detect it. Thus, if the payload is dropped, this sensor will alert the control to test the payload condition and then proceed with the adapted controller gains. In this way the control can adapt to 'on the fly' payload releases or drops. It might be argued that the sensor eliminates the need for re-identifying the payload, since if the payload is dropped there is obviously no payload on the arm and the no payload control could be implemented immediately. However, by performing the identification again, a safeguard is imposed for faulty sensor outputs. This is especially important for other control schemes, in which the dynamics might become unstable when the loaded manipulator is operated using control based on the no payload condition.

CHAPTER 4

ADAPTIVE VARIABLE STRUCTURE CONTROL BY PAYLOAD IDENTIFICATION

4.1 Introduction

In Chapter 3 it was shown that a regulator control using state feedback whose design is based on a linear model of the flexible manipulator performed well when applied to the original nonlinear model. However, it is known that a regulator such as this is not a highly robust control when disturbances are introduced into the system. These disturbances could be caused by a number of sources external to the manipulator/payload system, such as wind gusts if the manipulator is in such an environment. Disturbances could also be introduced if there are dynamics of the beam, payload, or actuator which are either unmodeled or modeled poorly by the simulator.

One class of control that has been proven to be quite robust in the presence of such disturbances is the so-called variable structure control (VSC). This type of control has been used on rigid robots for a number of years for this reason, and it has been proven to be a satisfactory control mechanism for nonlinear systems in

which the model parameters could not be precisely known or which are subject to disturbances due to unmodeled dynamics or external sources. Therefore, instead of using the less robust linear regulator, the use of variable structure control would be an improvement in the neural network control scheme of Chapter 3.

This chapter describes the implementation of a payload adaptive VSC for flexible manipulators. The basics of variable structure theory are presented and a demonstration of robustness in the presence of disturbances is given. Three examples of implementations of VSC specifically for flexible arms are explained, and compared, showing simulations of each. The best scheme among these is selected and used in conjunction with the neural network payload identification scheme derived in Chapter 3. Finally, the results of this variable structure scheme when the arm is operated while carrying each of the three payload classes are examined and compared with the results of the linear regulator control of Chapter 3.

4.2 Variable Structure Control

The main characteristic or feature of variable structure control is the sliding motion [80]. Sliding motion occurs near a specified manifold or hypersurface when the control is constantly switched such that the system state is always directed or forced onto this surface in the state space. The hypersurface must be chosen such that the dynamics of the system are asymptotically stable while sliding on it. This is the first phase of the design.

The second and final design stage is the selection of a control law such that the system state variables are attracted toward the sliding surface at all times. To show the basic concepts of VSC, consider the simple second order system

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -a_1x_1 - a_2x_2 + bu\end{aligned}\tag{4.1}$$

where x_1 and x_2 are the state variables to be controlled and u is the scalar control applied. The parameters a_i and b can be either constant or time-varying, and their exact values may be unknown. A discontinuous control is selected such that

$$u = \begin{cases} u^+, & s > 0 \\ u^-, & s < 0 \end{cases}\tag{4.2}$$

where

$$s = Gx_1 + x_2, \quad G > 0\tag{4.3}$$

and is defined as the switching function. The line at which the control is discontinuous is $s = 0$. This is the sliding manifold. When the system is sliding, or in other words $s = 0$, the following differential equation describes the dynamics of the system:

$$s = Gx_1 + \dot{x}_1 = 0.\tag{4.4}$$

The solution of this system is clearly

$$x_1(t) = x_1(t_0)e^{-G(t-t_0)}.\tag{4.5}$$

Thus, the order of the dynamics has been reduced by one, and the system behaves like an asymptotically stable first-order system with a pole at $-G$ during the sliding motion. The dynamics are also independent of b over a specified magnitude range [80]. This shows that a properly designed sliding line will give stable motion when the system is sliding.

The control that drives the system state variables to the sliding mode is governed by the so-called *reaching condition*. This is the condition that is necessary for the motion of the system states to be drawn toward the switching line. When s is positive, the control must assure that \dot{s} is negative and vice versa. Thus, the following are needed

$$\lim_{s \rightarrow 0^+} \dot{s} < 0 ; \quad \lim_{s \rightarrow 0^-} \dot{s} > 0 \quad (4.6)$$

on each side of the switching line. Both conditions may be combined to give the reaching condition [69]

$$s\dot{s} < 0 . \quad (4.7)$$

Thus, as suggested in [45], a control which insures that

$$\dot{s} = -k \operatorname{sgn}(s) , \quad k > 0 \quad (4.8)$$

where

$$\operatorname{sgn}(s) = \begin{cases} 1 & s > 0 \\ -1 & s < 0 \end{cases} \quad (4.9)$$

will satisfy the reaching condition in (4.7). This is evident, for \dot{s} will always have the opposite sign of s , and will always force the system states to the sliding line

$s = 0$. Therefore the control can be evaluated as follows: First the expression for \dot{s} is evaluated and set equal to $-k \operatorname{sgn}(s)$,

$$\begin{aligned}\dot{s} &= G\dot{x}_1 + \dot{x}_2 \\ &= Gx_2 - a_1x_1 - a_2x_2 + bu \\ &= -k \operatorname{sgn}(s)\end{aligned}\tag{4.10}$$

and then u is solved from the above expression to be

$$u = \frac{1}{b}(-k \operatorname{sgn}(s) + a_1x_1 + a_2x_2 - Gx_2).\tag{4.11}$$

This control is discontinuous and must oscillate to keep the state variables on the sliding line $s = 0$. For an ideal sliding mode, the frequency of this oscillation must be infinite, which because of switching delays, hysteresis, and other physical realities, cannot be achieved. As a result, the state will merely stay in a neighborhood of $s = 0$, continually passing across the sliding line as u switches between two or more distinct values. This process is what is known as *chattering*, and is undesirable in any physical system in which rapid switching could damage the control actuators. One way to eliminate chattering [13, 63] is to approximate the discontinuous control with one that is continuous. This can be done by changing the $\operatorname{sgn}(s)$ function to a saturation function, i.e.

$$\operatorname{sat}(s) = \frac{s}{(|s| + d)}\tag{4.12}$$

where d is a small positive number. Note that

$$\operatorname{sgn}(s) = \frac{s}{|s|}.\tag{4.13}$$

Another form of the saturation function that has been used in VSC [47] is

$$\text{sat}(s) = \begin{cases} 1 & s > \epsilon \\ s/\epsilon & |s| \leq \epsilon \\ -1 & s < -\epsilon \end{cases} \quad (4.14)$$

The most attractive feature of using VSC is its robustness to parameter variations. This can be shown [27] in the following manner. Consider the perturbed system dynamics

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -a_1 x_1 - a_2 x_2 + bu + \nu \end{aligned} \quad (4.15)$$

where ν is a term modeling all parameter variations and disturbances. The control function given by (4.11) is substituted into the system equations (4.15), resulting in

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -k \text{sgn}(s) - Gx_2 + \nu. \end{aligned} \quad (4.16)$$

If conditions can be found such that the reaching condition (4.7) is still satisfied in the presence of the disturbance, the disturbance will be rejected during the sliding motion. This can be shown by evaluating

$$\begin{aligned} s\dot{s} &= s(G\dot{x}_1 + \dot{x}_2) \\ &= s(-k \text{sgn}(s) - Gx_2 + \nu + Gx_2) \end{aligned}$$

$$\begin{aligned}
&= s(-k \operatorname{sgn}(s) + \nu) \\
&= s\nu - sk \operatorname{sgn}(s) .
\end{aligned} \tag{4.17}$$

The reaching condition is satisfied if

$$k > \frac{\nu}{\operatorname{sgn}(s)} . \tag{4.18}$$

It is seen in (4.18) that only a lower bound is specified for disturbance rejection. Thus all disturbances can be rejected if k is made sufficiently large. However, this large control gain selection can cause excessive chattering around the sliding manifold, and is not desirable. In the specific case of flexible manipulators, large control gains can cause excessive tip deflections as well, as will be shown later in this chapter.

To illustrate the application of variable structure control to flexible manipulator control, it will be applied to drive the hub angle to zero. In order to regulate the hub angle, the output θ of the manipulator is the error variable used in the sliding line equation that the VSC wishes to force to zero. For convenience, the linearized equation of the manipulator dynamics from Chapter 2 is once again stated,

$$\ddot{q} + \Omega q = \frac{\tau L}{c^2 D} \Lambda_N^T(0) . \tag{4.19}$$

The sliding line equation is selected as

$$s = \dot{\theta} + G\theta , \quad G > 0 \tag{4.20}$$

where θ and $\dot{\theta}$ are described in terms of the generalized coordinate q as

$$\theta = \Lambda_N(0)q ; \quad \dot{\theta} = \Lambda_N(0)\dot{q} \tag{4.21}$$

and $G = 6$ is chosen in this example for purposes of illustration.

Using (4.8), to find the control necessary to satisfy the reaching condition, an expression for \dot{s} is derived to be

$$\begin{aligned}
 \dot{s} &= \ddot{\theta} + G\dot{\theta} \\
 &= \Lambda_N(0)\ddot{q} + G\dot{\theta} \\
 &= \Lambda_N(0)\left(\frac{\tau L}{c^2 D}\Lambda_N^T(0) - \Omega q\right) + G\dot{\theta} \\
 &= \frac{\tau L}{c^2 D}\Lambda_N(0)\Lambda_N^T(0) - \Lambda_N(0)\Omega q + G\dot{\theta}. \tag{4.22}
 \end{aligned}$$

Thus, the control is of the form of (4.11), and is

$$\tau = \frac{-k \operatorname{sgn}(s) + \Lambda_N(0)\Omega q - G\dot{\theta}}{\frac{L}{c^2 D}\Lambda_N(0)\Lambda_N^T(0)} \tag{4.23}$$

The control gain k was chosen to be 3.3 for the simulation (again it is an arbitrary choice for illustrative purposes), and the discontinuous sign function was approximated by a saturation function as shown in (4.14), with ϵ chosen to be 0.3. The results of the simulation are shown in Figures 4.1, 4.2, 4.3 and 4.4.

The control performed well in regulating the hub rotation. The hub angle was quickly driven to zero, as shown in Figure 4.2, and the phase trajectory (Figure 4.4) of θ and $\dot{\theta}$ clearly shows the reaching and sliding phases of the control. The chattering phenomenon is minimal, as a result of the saturation function employed to approximate the discontinuous control. However, it can be seen in Figure 4.1 that the flexible modes of the beam have been excited by the control applied to the hub, causing an undamped tip deflection, which is an expected result. Hence, a

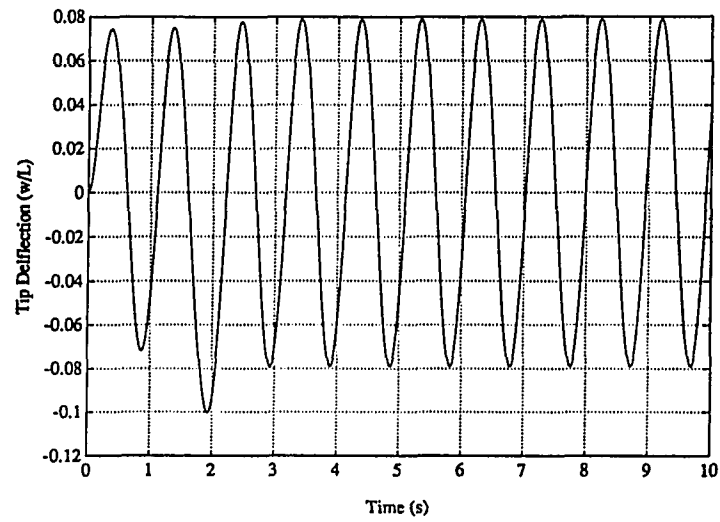


Figure 4.1: Tip Deflection, VSC Hub Regulator Control

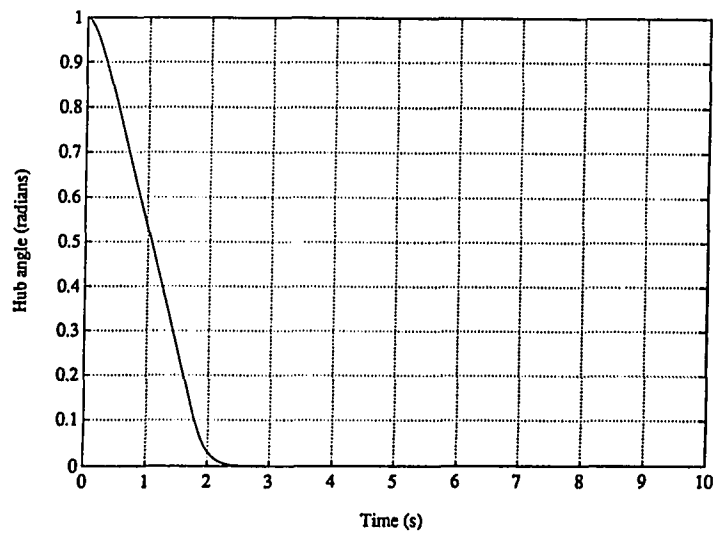


Figure 4.2: Hub Rotation, VSC Hub Regulator Control

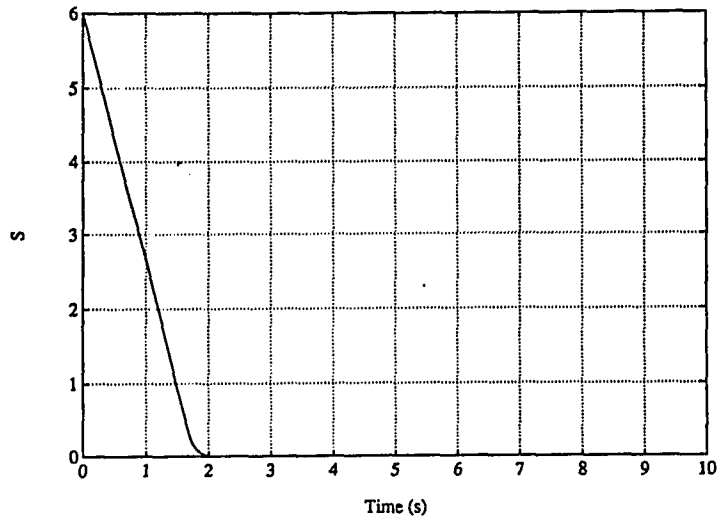


Figure 4.3: Sliding Line Trajectory, VSC Hub Regulator Control

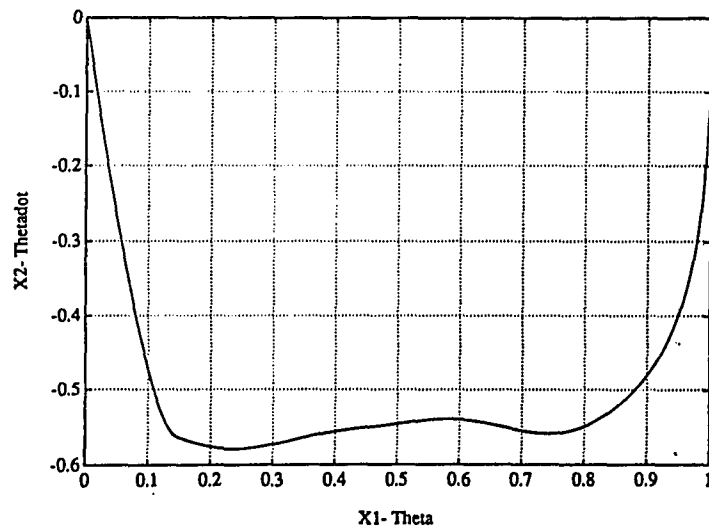


Figure 4.4: Phase Plane Trajectory, VSC Hub Regulator Control

different VSC must be designed in order to dampen these flexible vibrations. The next section presents several methods which address this issue.

4.3 Some Previous Results using VSC for Flexible Manipulators

In recent years, many control methods first devised for use on rigid robots have been extended to flexible ones. Variable structure techniques have not been an exception to this trend. The disturbance rejection properties of VSC make it especially appealing for application to flexible arms, where the model must be an approximation of the actual nonlinear distributed parameter system of the arm. In particular, three recent approaches will be discussed, simulated and compared.

4.3.1 Tip Position VSC

A logical solution to the problem of dampening the excitation of the flexible modes induced by the hub regulating VSC of the previous section would be to instead regulate the tip position. This approach was presented by Qian and Ma [52]. The design procedure is very similar to that of the hub regulation scheme except of course the sliding line is now changed to

$$s = x_2 + Gx_1 \quad (4.24)$$

where x_1 is the tip position, y_{tip} , and x_2 is the time derivative of the tip position, \dot{y}_{tip} . These quantities are functions of q expressed as

$$y_{tip} = Z_N(1)q ; \quad \dot{y}_{tip} = Z_N(1)\dot{q} . \quad (4.25)$$

Recall from Chapter 2 that $Z_N(1)$ is the vector of mode shapes of the manipulator evaluated at the tip.

The control that insures that $\dot{s} = -k \operatorname{sgn}(s)$ is

$$\tau = \frac{-k \operatorname{sgn}(s) + Z_N(1)\Omega q - Gx_2}{\frac{L}{\epsilon^2 D} Z_N(1)\Lambda_N^T(0)} . \quad (4.26)$$

Thus, a very simple control is derived to drive the tip position to zero. However, the tip position is a function of both the hub angle and the deflection caused by the flexibilities of the arm. Insuring that the tip goes to zero does not insure that the hub angle goes to zero as well. The position of the tip is defined in non-dimensional coordinates as

$$y_{tip} = Z_N(1)q = w + \theta \quad (4.27)$$

Therefore, the tip can be driven to zero by simply making the tip deflection, w , be equal and opposite in sign to the hub rotation. This approach of course is not a desirable method of control, and undesirable results are precisely what occur when this method is used, as shown in Figures 4.5 and 4.6. Instead of a reduction in the magnitude of the hub angle, it is actually increased very rapidly, causing the tip deflection to become negative, which drives y_{tip} towards zero. This requires an ever-increasing magnitude of tip deflection, which of course will cause damage to

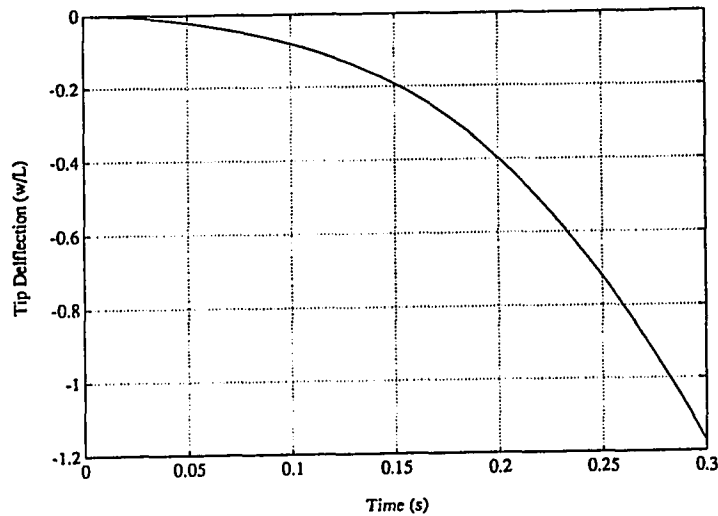


Figure 4.5: Tip Deflection, VSC Tip Position Approach

the arm and also require a constantly increasing torque. The phase plane trajectory (Figure 4.7) shows that the states are unable to stay in the neighborhood of the switching line, because the beam will eventually deflect back in the other direction. The control is unable to prevent this from happening, and the system becomes unstable.

It should be noted that in Qian and Ma's simulation, the model that they used contained some structural damping in the formulation of the linear model used for their simulations and design of the control which might have been enough for the control to be successful. However, the only simulation results shown in the paper were of the tip position, and the actual value of the structural damping term was not given. The modeling of the structural damping is not well developed, and it

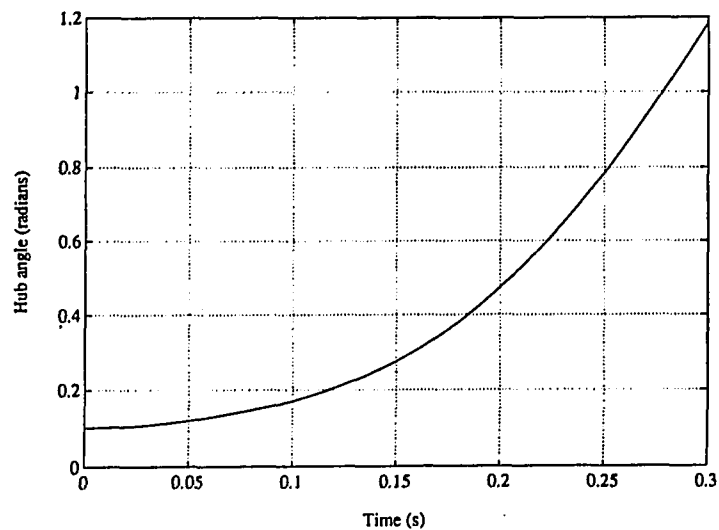


Figure 4.6: Hub Rotation, VSC Tip Position Approach

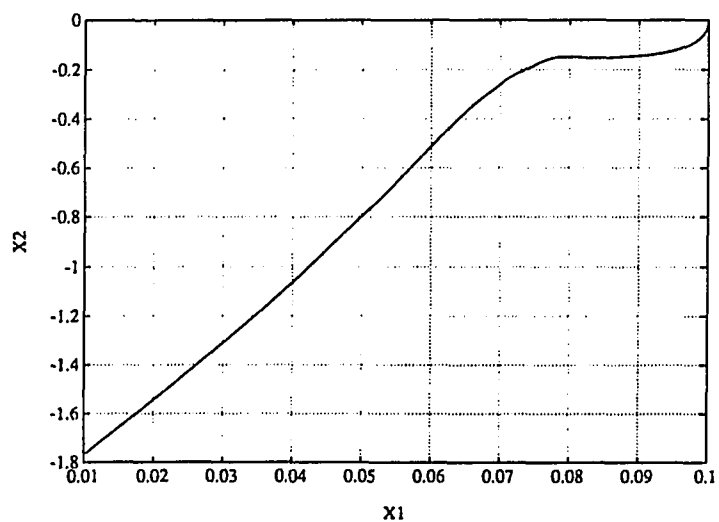


Figure 4.7: Phase Plane Trajectory, VSC Tip Position Approach

is difficult to accurately quantify, and thus a control system should not depend on these natural damping terms for successful results.

4.3.2 Hub Rotation plus Pole Placement Approach

The drawback in the hub regulation example described in Section 4.2 is that the flexible modes were excited. Nathan and Singh [47] proposed to drive the hub angle to zero in the same manner, but when the hub has entered a neighborhood close to zero, a switching logic adds linear state feedback to the VSC to dampen the oscillations of the tip due to the flexible modes.

The control is constructed in two parts. First, the VSC is designed for the hub. The expressions for s , \dot{s} and the control are the same as (4.20), (4.22) and (4.23), respectively, and are re-stated for convenience. The nonlinear saturation function (4.14) is substituted into (4.23) as well for a practical implementation to be possible.

$$s = \dot{\theta} + G\theta, \quad G > 0 \quad (4.28)$$

$$\dot{s} = \frac{\tau L}{c^2 D} \Lambda_N(0) \Lambda_N^T(0) - \Lambda_N(0) \Omega q + G\dot{\theta} \quad (4.29)$$

$$\tau_{vsc} = \frac{-k \text{sat}(s) + \Lambda_N(0) \Omega q - G\dot{\theta}}{\frac{L}{c^2 D} \Lambda_N(0) \Lambda_N^T(0)}. \quad (4.30)$$

The second phase of the design is the construction of the pole placement stabilizer, which is desired to be of the form

$$\tau_s = \frac{w}{\frac{L}{c^2 D} \Lambda_N(0) \Lambda_N^T(0)} \quad (4.31)$$

where w is the stabilizer input and is to be determined later. The total input to the system is

$$\tau = \tau_{usc} + \tau_s . \quad (4.32)$$

Substituting (4.31) and (4.32) into (4.22) gives

$$\dot{s} = -k \text{sat}(s) + w . \quad (4.33)$$

For $w = 0$, it is seen that $s(t) \rightarrow 0$ as $t \rightarrow \infty$.

The stabilizer is designed in a new state space, with state vector z defined as

$$z = \begin{bmatrix} \theta \\ s \\ p \\ \dot{p} \end{bmatrix} \quad (4.34)$$

where p and \dot{p} are vectors consisting of the $n - 1$ flexible modes of the system and the time derivatives, respectively, and are given as

$$p = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \quad \dot{p} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} . \quad (4.35)$$

When the stabilizer is operating, the dynamics of the state vector z are given by the following differential equation

$$\dot{z} = Fz + Ew \quad (4.36)$$

where F and E are found as follows by evaluating the time derivatives of the state vector components. Using (4.28), $\dot{\theta}$ is found to be

$$\dot{\theta} = s - G\theta . \quad (4.37)$$

Since the stabilizer control is based on a linear pole placement design, the state space must be linear, and therefore (4.33) must be stated in its linearized form as

$$\dot{s} = \frac{-k}{\epsilon} s + w. \quad (4.38)$$

The final expression that must be solved for is \ddot{p} . By eliminating the rigid mode from (4.19), one gets

$$\ddot{p} = \frac{\tau L}{c^2 D} \Lambda_F^T(0) - \Omega_F p \quad (4.39)$$

where

$$\Lambda_F(0) = \begin{bmatrix} \alpha_1(0) & \alpha_2(0) & \cdots & \alpha_n(0) \end{bmatrix} \quad (4.40)$$

and

$$\Omega_F = \begin{bmatrix} \omega_1^2 & 0 & \cdots & 0 \\ 0 & \omega_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_n^2 \end{bmatrix}. \quad (4.41)$$

Substituting (4.32) into (4.39) gives

$$\ddot{p} = \left[\frac{\frac{-k}{\epsilon} s + \Lambda_N(0) \Omega q - G \dot{\theta} + w}{\Delta} \right] \Lambda_F(0) - \Omega_F p \quad (4.42)$$

where

$$\Delta = \Lambda_N(0) \Lambda_N^T(0). \quad (4.43)$$

Noting that

$$\Lambda_N(0) \Omega q = \Lambda_F(0) \Omega_F p, \quad (4.44)$$

and substituting (4.37) into (4.42), \ddot{p} can be expressed solely in terms of the state vector z as

$$\ddot{p} = \frac{\frac{-k}{\epsilon} - G}{\Delta} \Lambda_F^T(0)s + \frac{G^2}{\Delta} \Lambda_F^T(0)\theta + \left[\frac{\Lambda_F^T(0)\Lambda_F(0)\Omega_F}{\Delta} - \Omega_F \right] p + \frac{\Lambda_F^T(0)}{\Delta} w. \quad (4.45)$$

Combining (4.37), (4.38), and (4.45) into state space form, gives

$$\dot{z} = \begin{bmatrix} -G & 1 & O_{1 \times 4} & O_{1 \times 4} \\ 0 & \frac{-k}{\epsilon} & O_{1 \times 4} & O_{1 \times 4} \\ O_{4 \times 1} & O_{4 \times 1} & O_{4 \times 4} & I_{4 \times 4} \\ \frac{G^2}{\Delta} \Lambda_F^T(0) & \frac{\frac{-k}{\epsilon} - G}{\Delta} \Lambda_F^T(0) & \left[\frac{\Lambda_F^T(0)\Lambda_F(0)\Omega_F}{\Delta} - \Omega_F \right] & O_{4 \times 1} \end{bmatrix} z + \begin{bmatrix} 0 \\ 1 \\ O_{4 \times 1} \\ \frac{\Lambda_F^T(0)}{\Delta} \end{bmatrix} w \quad (4.46)$$

where I and O are the identity and null matrices, respectively.

The closed loop poles of this system may be placed with the standard technique given in Chapter 3 with a state feedback control

$$w = -L_f z. \quad (4.47)$$

The results of the simulation using this method are shown in Figures 4.8, 4.9, and 4.10. The pole locations for the closed loop system were chosen to be

$$\Lambda = \begin{bmatrix} -1.50 \pm j359.05 \\ -1.50 \pm j216.94 \\ -1.50 \pm j109.97 \\ -1.50 \pm j6.51 \\ -6.0 \\ -11.17 \end{bmatrix}. \quad (4.48)$$

The locations of the poles associated with the flexible modes were simply chosen to be in the LHP such that the oscillations of the tip were damped. The real poles corresponding to the rigid mode were chosen to be at the same locations as those in the VSC example for the hub control given in Section 4.2 to show how this method

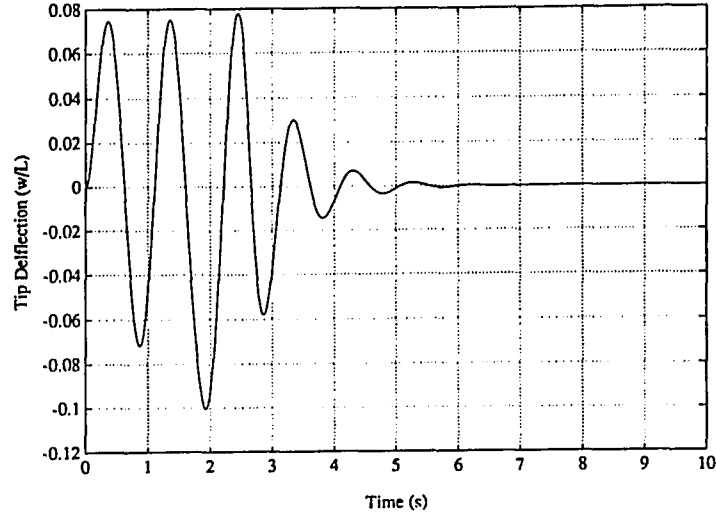


Figure 4.8: Tip Deflection, VSC + Pole Placement Approach

is an extension of that example. These poles are both determined by G and k and can be calculated explicitly by simply calculating the eigenvalues of F . The corresponding feedback vector L_f which places the poles at these locations is

$$L_f = \begin{bmatrix} -30.5 & 5.7 & 243.6 & -1284.5 & -3094.2 \\ 6394.9 & -13.9 & -59.6 & -130.6 & -219.6 \end{bmatrix}. \quad (4.49)$$

The parameters associated with the VSC were selected as $G = 6$, and $k = 3.3$, as they were in the hub VSC example given earlier. It is apparent that the switching logic added the stabilizer to the control loop at approximately 2.5 seconds. It is clearly seen from the phase plane trajectory plot (Figure 4.10), that the states are moved off of the sliding line at this time. The control effort also moved the hub as shown in Figure 4.9. The total time for convergence to zero of both the

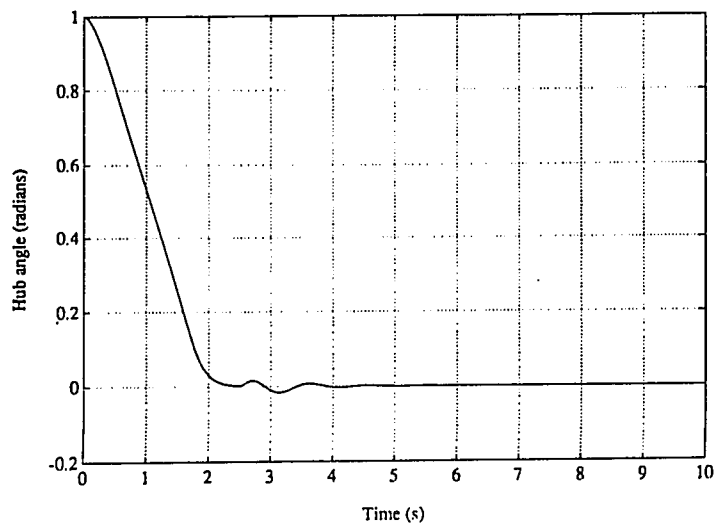


Figure 4.9: Hub Rotation, VSC + Pole Placement Approach

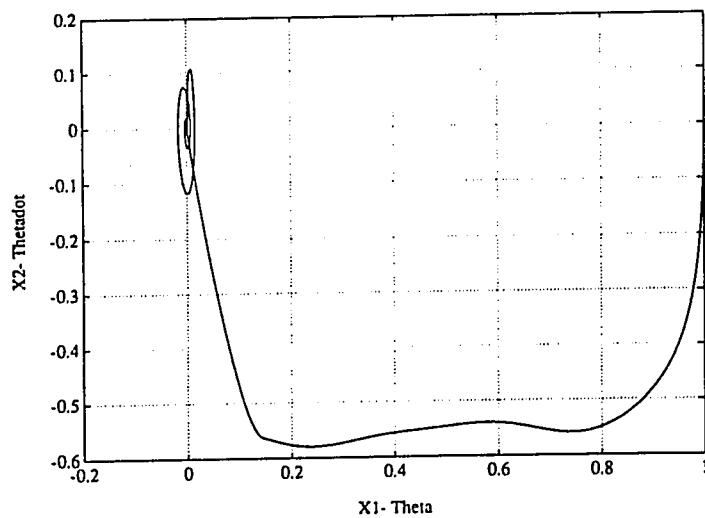


Figure 4.10: Phase Plane Trajectory, VSC + Pole Placement Approach

tip deflection and the hub rotation was approximately 6 seconds. This is twice as slow as the linear regulator control designed in Chapter 3. However, the control is more robust in this case. The reason for this is that because the pole placement isn't activated until the hub angle is relatively small, the stabilizer control, which is simply a linear pole placement controller is actually being applied to a linear system. This is because most of the nonlinear effects of the model are due to the hub dynamics, which can be neglected when the hub is only required to move a small angle. In contrast, the linear regulator designed in Chapter 3 is always being applied to a system which is highly nonlinear and is therefore only guaranteed to stabilize the system within a region around an operating point.

The control is successful in accomplishing its task, but it has several disadvantages. First, it must have a switching logic that is based on both the hub position and the hub velocity to tell when the stabilizer should be included in the control loop. Furthermore, the control is inefficient, because the tip vibrations are not suppressed until the hub has come close to the desired angle. In order to suppress these vibrational modes, the hub may have to be moved away from the neighborhood in which the stabilizer is operating, thus shutting it off. The control will eventually stabilize of course, but the time it takes to accomplish this will not be as fast as something such as the linear regulator of Chapter 3. Also, the additional stabilizer control is a disturbance to the VSC. This means that the faster the flexible poles are wished to be damped, the larger the control gain must be made to insure that the

reaching condition holds. This larger control gain results in larger tip deflections, so the speed is limited as it was using the linear regulator control.

4.3.3 VSC State Regulator Method

Both of the previous methods have disadvantages that limit the performance of the control. A variable structure control is needed which regulates all states simultaneously, rather than merely regulating a specific output as both methods so far have tried to do. This characteristic, however, requires a more rigorous hyperplane design, in order to make sure that the sliding motion is asymptotically stable.

Such a state regulator VSC method has been explored very briefly by [62], using a hyperplane design method suggested in [80]. A much more complete examination of this approach to flexible manipulators is given in this section. This approach uses a sliding line of the form

$$s = Cx \quad (4.50)$$

and is based on a specific canonical transformation of the linear state equations

$$\dot{x}(t) = Ax(t) + B\tau(t) \quad (4.51)$$

in which, for the flexible manipulator,

$$x(t) = \begin{bmatrix} q(t) \\ \dot{q}(t) \end{bmatrix}. \quad (4.52)$$

The canonical form which is needed is obtained through an $n \times n$ linear transformation matrix, T , such that

$$TB = \begin{bmatrix} 0 \\ B_2 \end{bmatrix} \quad (4.53)$$

where B_2 is $m \times m$ and non-singular. The transformation T can be obtained through QU (also known as QR) factorization of B , where B is decomposed into

$$B = Q \begin{bmatrix} U \\ 0 \end{bmatrix} \quad (4.54)$$

where Q is $n \times n$ and orthogonal and U is $m \times m$. Therefore, T can be obtained by “flipping” the rows of Q^T , i.e. the last row becomes the first, etc. Hence T is an orthogonal matrix as well, and there is no inversion problem numerically, since $T^{-1} = T^T$.

The new (transformed) state vector is now defined as

$$y = T \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \quad (4.55)$$

and the transformed state equations are

$$\dot{y}(t) = TAT^T y(t) + TB\tau(t). \quad (4.56)$$

Consider the partitioning of the state vector y in the form

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \quad y_1 \in R^{n-m}, \quad y_2 \in R^m. \quad (4.57)$$

If the other matrices appearing in (4.56) and (4.50) are partitioned similarly as

$$TAT^T = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}; \quad CT^T = \begin{bmatrix} C_1 & C_2 \end{bmatrix}, \quad (4.58)$$

then the state equations may be expressed in the form

$$\dot{y}_1(t) = A_{11}y_1(t) + A_{12}y_2(t) \quad (4.59)$$

$$\dot{y}_2(t) = A_{21}y_1(t) + A_{22}y_2(t) + B_2\tau(t), \quad (4.60)$$

and the sliding condition is

$$C_1y_1(t) + C_2y_2(t) = 0. \quad (4.61)$$

The aim of the first part of this design is to design an asymptotically stable sliding mode. While sliding, $y_1(t)$ and $y_2(t)$ are related by (4.61)

$$y_2(t) = -C_2^{-1}C_1y_1(t). \quad (4.62)$$

If (4.62) is substituted into (4.59), the state dynamics during the sliding motion are described as

$$\dot{y}_1 = (A_{11} - A_{12}F)y_1 \quad (4.63)$$

where

$$F = -C_2^{-1}C_1. \quad (4.64)$$

Thus y_2 takes on the role of a state feedback control during sliding, and F can be appropriately designed to place the eigenvalues in the LHP during this time. This technique will insure that all states will be asymptotically stable, which is what was initially sought. After F is solved, one way to get a C vector is to fix C_2 to be $I_{m \times m}$ and therefore

$$C = \begin{bmatrix} F & I_{m \times m} \end{bmatrix} T. \quad (4.65)$$

In other situations, different forms of C than the one derived by the above method (such as a diagonal form) may simplify the control design [80]. However, for the problem of flexible manipulator control, this form for C is satisfactory.

The second ⁰ phase is once again the design of the control to bring the states to the sliding line. The control structure for this method is based on a procedure outlined by Ryan and Corless [57] and has a form consisting of two parts, one linear feedback part and one nonlinear part,

$$u(x) = u_L(x) + u_N(x) = Lx + \frac{\sigma}{|Mx|}Nx. \quad (4.66)$$

To simplify the design and aid in the understanding of the quantities σ , N , and M as well as the selection of L , the transformed state y is once again transformed such that

$$z = T_2 y \quad (4.67)$$

where

$$T_2 = \begin{bmatrix} I_{(n-m) \times (n-m)} & O_{m \times m} \\ F & I_{m \times m} \end{bmatrix}. \quad (4.68)$$

T_2 is non-singular and has an inverse given by

$$T_2^{-1} = \begin{bmatrix} I_{(n-m) \times (n-m)} & O_{m \times m} \\ -F & I_{m \times m} \end{bmatrix}. \quad (4.69)$$

Thus partitioning z^T into $[z_1 \ z_2]^T$, one obtains

$$z_1 = y_1; \quad z_2 = Fy_1 + y_2 \quad (4.70)$$

and the condition for sliding is $z_2 = 0$. The transformed state equations in z -space are

$$\dot{z}_1 = Jz_1 + A_{12}z_2 \quad (4.71)$$

$$\dot{z}_2 = Hz_1 + \Phi z_2 + B_2u \quad (4.72)$$

where

$$J = A_{11} - A_{12}F \quad (4.73)$$

$$H = FJ - A_{22}F + A_{21} \quad (4.74)$$

$$\Phi = FA_{12} + A_{22} . \quad (4.75)$$

Hence, in order to attain the sliding mode and remain there, it is necessary that both z_2 and \dot{z}_2 become identically zero. To accomplish this, the linear part of the control is used. From (4.72), the control which forces \dot{z}_2 to be zero is

$$u_L(z) = -B_2^{-1}[Hz_1 + (\Phi - \Phi^*)z_2] \quad (4.76)$$

where Φ^* is any matrix with poles in the LHP (in order to force z_2 to zero). Transforming L back into the original state space gives

$$L = -B_2^{-1}[H(\Phi - \Phi^*)]T_2T . \quad (4.77)$$

This linear control only serves to drive z_2 to zero asymptotically. In order to attain sliding in finite time, the nonlinear discontinuous control is implemented. Letting P_2 denote the positive definite solution of the Lyapunov equation

$$P_2\Phi^* + (\Phi^*)^TP_2 + I_{m \times m} = 0 \quad (4.78)$$

then $P_2 z_2 = 0$ only if $z_2 = 0$. Thus, a control may be selected which is discontinuous at $z_2 = 0$, and continuous elsewhere, as

$$u_N(z) = \frac{-\sigma}{|P_2 z_2|} B_2^{-1} P_2 z_2; \quad z_2 \neq 0 \quad (4.79)$$

where $\sigma > 0$ and is free to be selected by the designer, and when $z_2 = 0$, u_N may be arbitrarily selected such that $|u_N| \leq \sigma$. Transforming back into x -space, the nonlinear control is

$$u_N(x) = \frac{-\sigma}{|Mx|} B_2^{-1} Mx \quad (4.80)$$

where

$$M = \begin{bmatrix} 0 & P_2 \end{bmatrix} T_2 T. \quad (4.81)$$

Note that for the flexible manipulator, B_2 is a scalar, and therefore, the control can be expressed as

$$u_N(x) = -\sigma B_2^{-1} \text{sgn}(Mx). \quad (4.82)$$

Once again, for practical purposes, the saturation function (4.14) can be used instead of the discontinuous sgn function in implementing (4.82).

Example for Flexible manipulators

In order to see how the parameters affect the response of the arm, a simulation will be done when it is known that the manipulator has no payload on it. The sliding hyperplane was designed such that the poles of the system while sliding were all 1 unit to the left of the $j\omega$ -axis, for illustrative purposes. The parameters associated

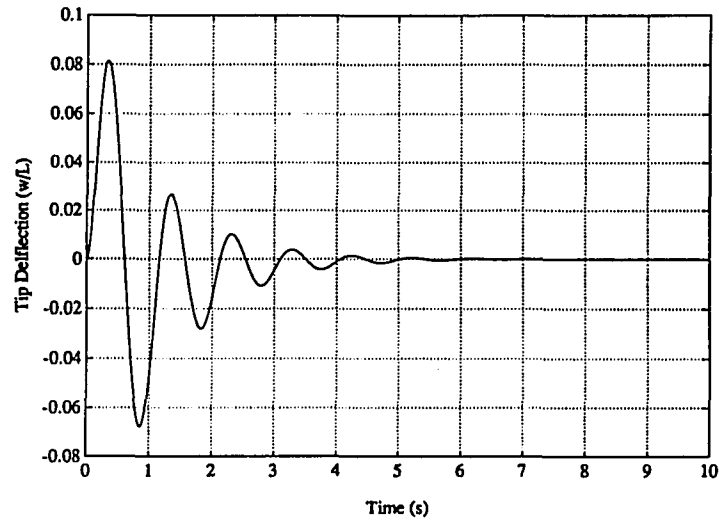


Figure 4.11: Tip Deflection under VSC control, No Payload

with the control to satisfy the reaching condition, Φ^* and σ were chosen arbitrarily to be -3 and 3, respectively. The simulation results are shown in Figures 4.11, 4.12, and 4.13.

The results show successful state regulation by this method. The control could obviously be increased (as the tip deflection is still well below 0.1) in order to speed up the response, and it still converges as well with these less than nominal parameters as the VSC control of Nathan and Singh which had a maximum tip deflection magnitude of 0.1. This shows the superiority of this approach when compared to their method.

The choices of the pole locations and other parameters and how these choices affect the response of the arm aren't very well defined. There are no specific algorithms which allow explicit values of tip deflection and hub speed to be incorporated

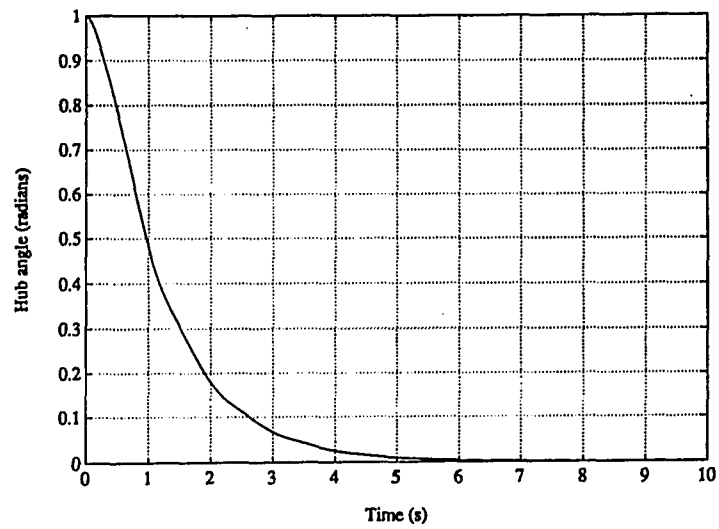


Figure 4.12: Hub Rotation under VSC control, No Payload

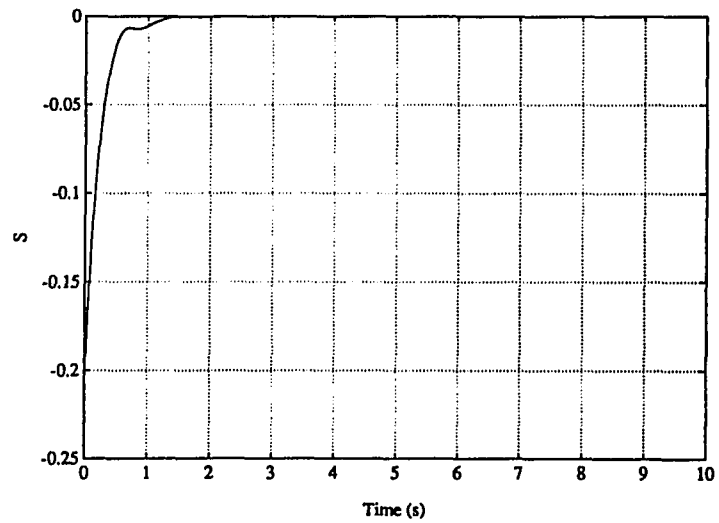


Figure 4.13: Sliding Line Trajectory, No Payload

as design parameters to date. But there are some general rules which can aid in the determination of specific parameters.

While sliding, the structure of the hyperplane determines the dynamics of the arm. The choice of the vector of parameters, C , depends on the eigenvalues of the desired dynamics. The poles can be looked at in three groups, the rigid pole, the poles associated with the first flexible mode, and the rest of the eigenvalues associated with the higher order flexible modes. The first two groups have the most prominent effect on the arm dynamics, as one would expect. The rigid mode eigenvalue affects the speed of the hub rotation the most, and as a result also affects the tip deflection the most as well. As it is moved deeper into the LHP, both the hub speed and tip deflection increase. The same is true of the first flexible mode, although on a smaller scale for the hub speed, and the tip deflection, despite having an initial larger transient magnitude, will go to zero faster. The poles associated with the higher modes can be moved deep into the LHP, but no significant effect is seen in either the tip deflection or the hub speed. The deeper the poles, the larger the feedback gains required to place them there, however, which could result in larger costs for the amplifiers used to create the gains.

The control to drive the states to the sliding hyperplane also affects the response. The faster the states are driven towards the hyperplane, the larger the tip deflection and the faster the hub is moved. In terms of the z -space in the design procedure, the time required to reach the sliding hyperplane from an arbitrary initial state z^0

is given by [80] as

$$t_{reach} \leq \frac{1}{\sigma} \sqrt{\frac{\langle z_2^0, P_2 z_2^0 \rangle}{\varsigma_{min}(P_2)}} \quad (4.83)$$

where ς_{min} denotes the minimum eigenvalue of P_2 , and $\langle \cdot, \cdot \rangle$ is the usual Euclidean inner product on R^m . This relationship shows that a large value of σ would produce a large tip deflection. For the flexible manipulator, P_2 is a scalar, thus the solution of the Lyapunov equation (4.78) yields

$$P_2 = \frac{-1}{2\Phi^*} \quad (4.84)$$

Thus a choice of a more negative Φ^* will also reduce the reaching time, increasing the tip deflection, although it won't be as strong as the effect of σ . In light of this relationship, a simple way of determining these two design parameters is to fix Φ^* at a location in the LHP, and adjust σ to reduce the reaching time while still making sure that the bound on the maximum tip deflection is not exceeded. This is an iterative procedure, so by fixing one parameter and adjusting the other, the design process is simpler and faster.

The reaching time should be minimized because during this time the system is not robust to parameter changes and disturbances. Although several researchers have stated that the use of VSC is sufficient to eliminate the need for payload identification altogether [47, 76], this is not true if the tip deflection is desired to be limited during the reaching time period, and if the dynamics are much different than expected (i.e. the payload is much different than the control was designed for), the control applied to force the system to the sliding hyperplane may fail completely,

causing the system to become unstable. For this reason, it is still critical to identify the payload before using VSC.

4.4 Payload Identification Plus VSC

The neural network payload identification scheme from Chapter 3 can be employed in conjunction with variable structure control to get a truly robust payload adaptive control scheme for flexible manipulators. This control scheme will be referred to as NNVSC. The neural network that was trained for the 3 classes previously was used once again so the classes remained the same as before. A VSC was calculated for each payload class and its parameters were stored in the same manner as with the linear regulator scheme, and the neural net was used once again to pick the control associated with the correct payload. These three controls will be referred to as Class 0 control, Class 1 control, and Class 2 control in the following discussion. The simulation results for the Class 0 payload are shown in Figures 4.14 and 4.15. Once again the figures show the responses under both correct and incorrect payload identifications, but the incorrect identifications were forced for the sake of comparison and were not actual mistakes of the neural network identification scheme.

The results for the Class 0 payload show the value of identifying the payload before applying the VSC. The hub speed is much greater for the Class 0 control, converging in about 3 seconds. The tip deflection was also reduced to zero by that

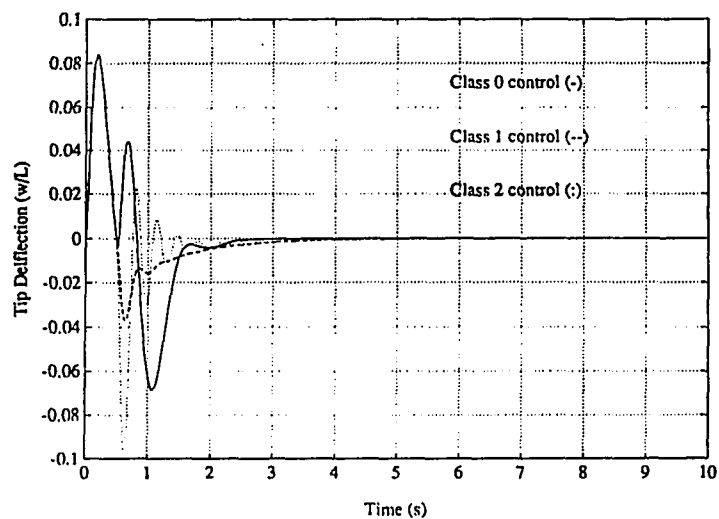


Figure 4.14: Tip Deflection of Arm: Class 0 Payload, NNVSC

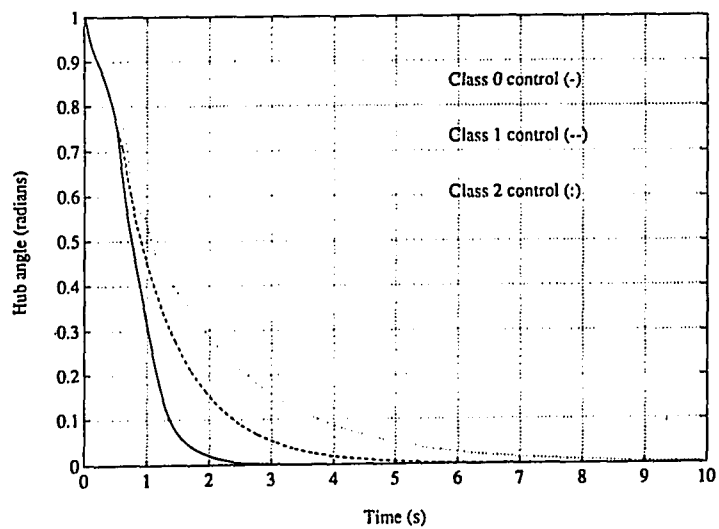


Figure 4.15: Hub Rotation of Arm: Class 0 Payload, NNVSC

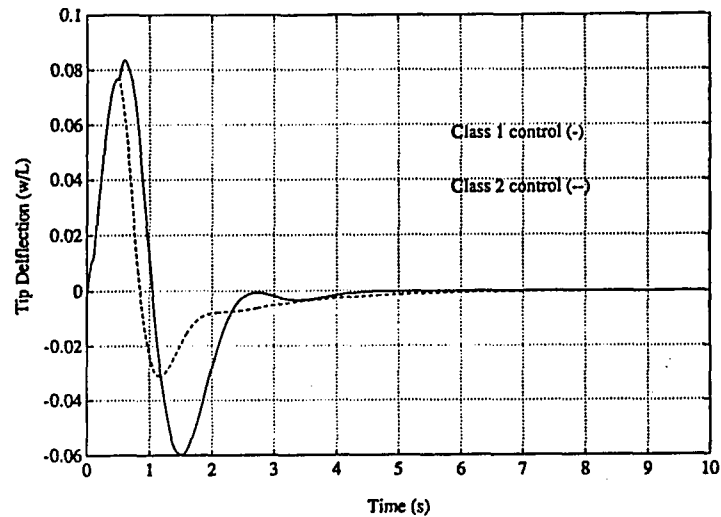


Figure 4.16: Tip Deflection of Arm: Class 1 Payload($\mu = 0.4$), NNVSC

time. The Class 1 and Class 2 controls each operate the beam slower than necessary. The inefficiency of the Class 2 control in increasing the tip deflection while also resulting in a very slow regulation of the hub angle needs to be particularly noted.

The results of the simulation of a Class 1 payload ($\mu = 0.4$) are shown in Figures 4.16 and 4.17. These two figures present the effects of the application of Class 1 and Class 2 controls to the arm. The Class 1 control is superior and is clearly seen in the much faster hub convergence, as expected. However, when Class 0 control is used, some interesting results occur, as shown in Figures 4.18, 4.19, and 4.20.

When Class 0 control is applied, the control obviously is not successful. The reason for this result is because the sliding condition is never attained. The reason why this behavior happens can be seen by examining the control parameters that

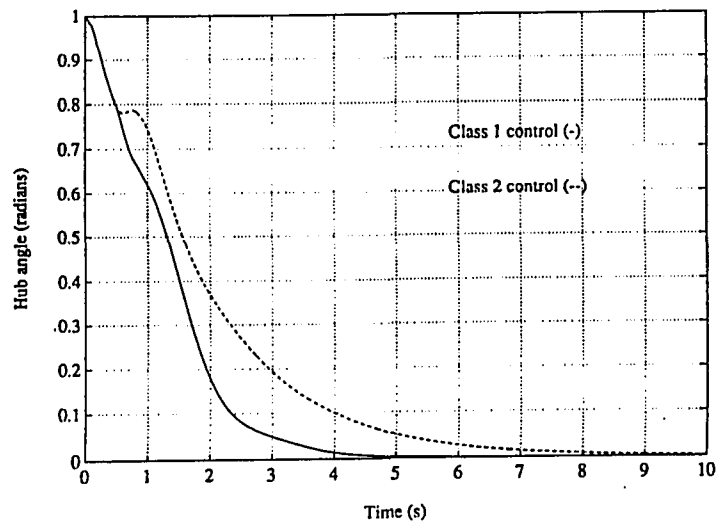


Figure 4.17: Hub Rotation of Arm: Class 1 Payload($\mu = 0.4$), NNVSC

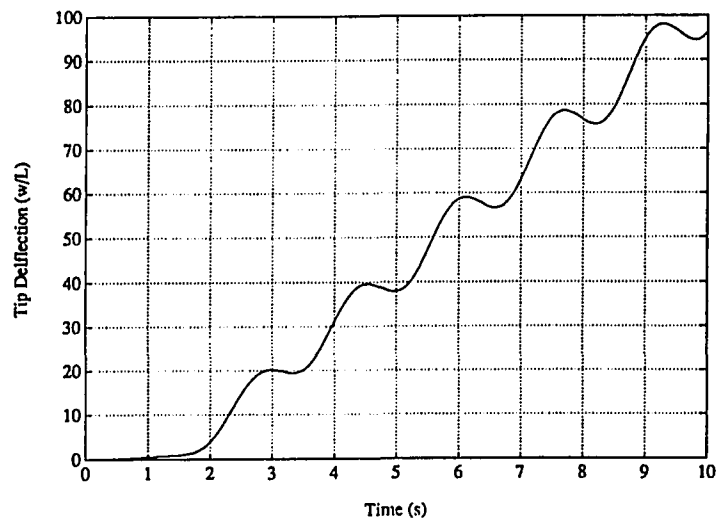


Figure 4.18: Tip Deflection of Arm: Class 1 Payload($\mu = 0.4$), Class 0 NNVSC

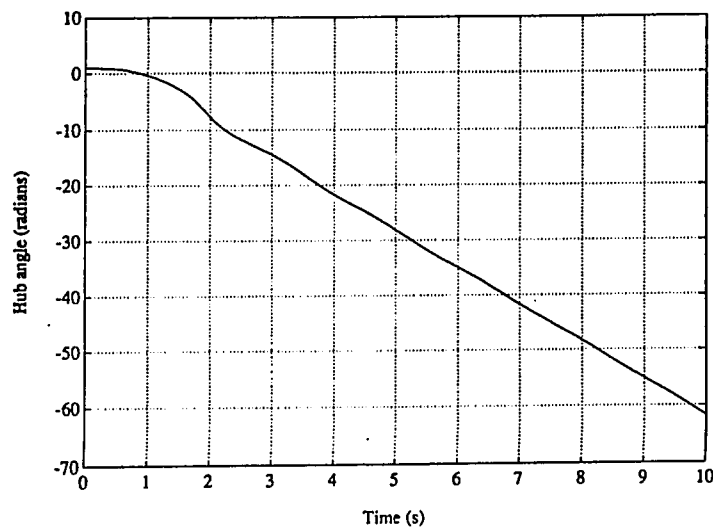


Figure 4.19: Hub Rotation of Arm: Class 1 Payload($\mu = 0.4$), Class 0 NNVSC

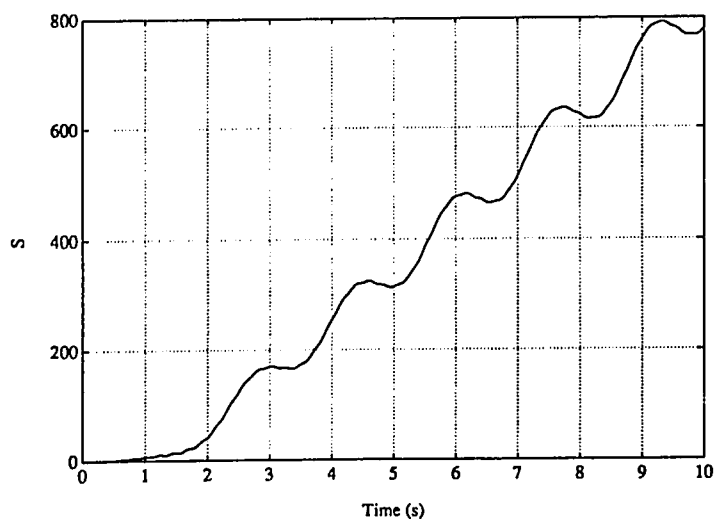


Figure 4.20: Sliding Mode of Arm: Class 1 Payload($\mu = 0.4$), Class 0 NNVSC

are used for this control. The attainment of the sliding mode depends primarily on the linear portion of the control (4.76). Therefore, by examining the vector

$$\Upsilon = \begin{bmatrix} H & (\Phi - \Phi^*) \end{bmatrix} \quad (4.85)$$

that results when the feedback gain vector L corresponding to each class is used, some insight can be gained as to why a Class 2 control works, but a Class 0 control doesn't. The vector Υ contains the information pertaining to the dynamics of z_2 , which must be forced to zero in order to attain the sliding mode. When a Class 1 control is used,

$$\Upsilon = \begin{bmatrix} 31.4 & 30.7 & 50.2 & -46.2 & 23.7 \\ -9404.5 & -5196.2 & -2380.8 & 141.0 & 38.6 \end{bmatrix}. \quad (4.86)$$

When a Class 2 control is used,

$$\Upsilon = \begin{bmatrix} 50.2 & 40.2 & 50.6 & -44.2 & 22.60 \\ -8786.4 & -5008.6 & -2358.6 & 145.7 & 38.9 \end{bmatrix}. \quad (4.87)$$

It is apparent that the two controls are close enough that the dynamics of z_2 will be driven to zero regardless of the payload difference. However, when a Class 0 control is used,

$$\Upsilon = \begin{bmatrix} 24.4 & 19.1 & 21.0 & -36.4 & 17.6 \\ -6651.4 & -4089.5 & -2152.5 & 212.5 & 23.7 \end{bmatrix}. \quad (4.88)$$

This totally wrong assumption of the parameters associated with the dynamics of z_2 (which are particularly different for the last five elements of Υ) does not guarantee

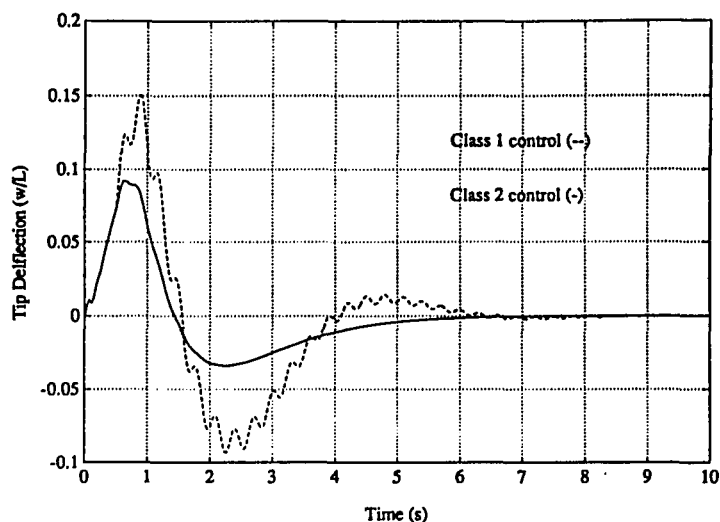


Figure 4.21: Tip Deflection of Arm: Class 2 Payload($\mu = 1.2$), NNVSC

that the control will force z_2 to zero. Thus, the sliding mode is never attained, and the control drives the system to an unstable state. The results are similar for a Class 2 payload, as shown in Figures 4.21, 4.22, and 4.23.

Although the convergence rate for the hub rotation of the arm carrying a Class 2 payload is slower when using a Class 2 control than the hub rotation using a Class 1 control, the tradeoff is once again tip deflection. The Class 1 control is too fast, and exceeds the bound on tip deflection, as seen in Figure 4.21. Another interesting consequence of using Class 1 control is that the high frequency poles are damped much slower than when Class 2 control is used. Once again the Class 0 control causes the system to become unstable, which is not unexpected since the same control didn't work for the lighter Class 1 payload.

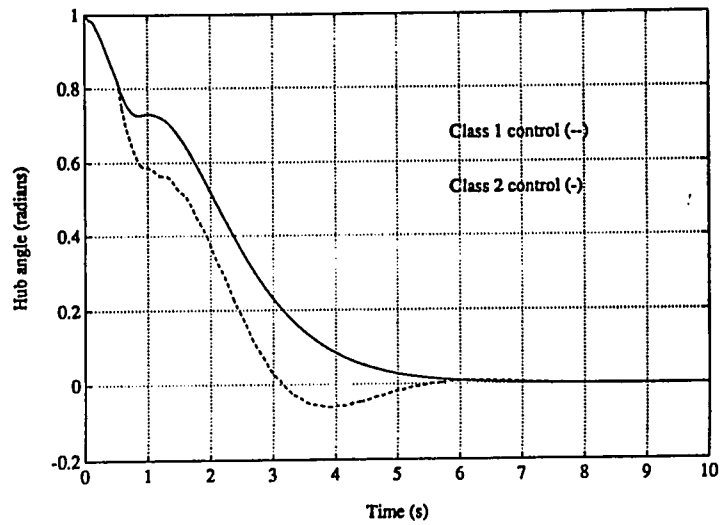


Figure 4.22: Hub Rotation of Arm: Class 2 Payload($\mu = 1.2$) NNVSC

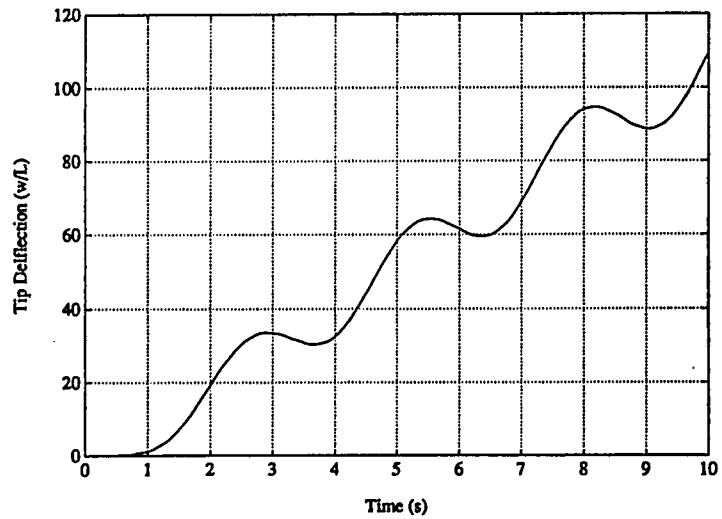


Figure 4.23: Tip Deflection of Arm: Class 2 Payload($\mu = 1.2$) Class 0 NNVSC

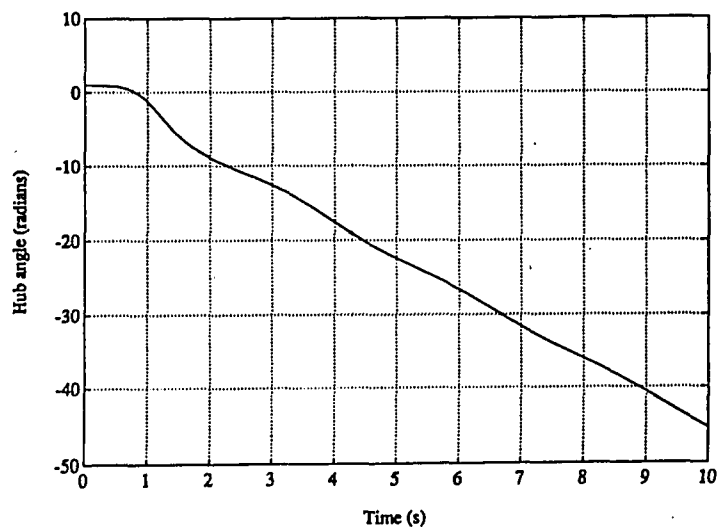


Figure 4.24: Hub Rotation of Arm: Class 2 Payload($\mu = 1.2$) Class 0 NNVSC

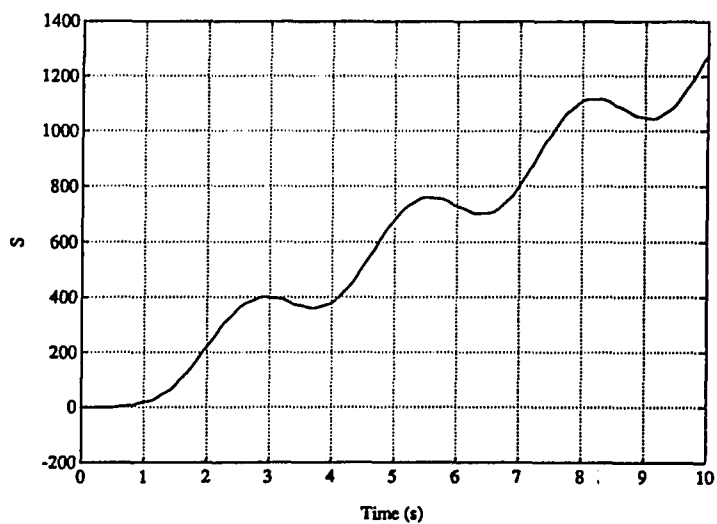


Figure 4.25: Sliding Mode of Arm: Class 2 Payload($\mu = 1.2$) Class 0 NNVSC

The VSC approach is attractive because of its robustness while sliding. But does this method perform as well as the approach of Chapter 3? The two techniques are compared to find out. When the payload is classified correctly (which was 100 percent of the time), the control using the variable structure approach converged a little faster for all payload classes. However, for VSC, if the payload identification scheme does make a mistake for whatever reason and classifies the payload as Class 0 when it is not, there could be serious consequences, as shown in Figures 4.16, 4.21, etc. A confirmation to the classification of no payload by the neural network could be obtained by a crossfire-type sensor such as the one discussed in Chapter 3, which could provide a secondary indication to check the class output of the network. If the two differ, the payload identification could be run a second time. In this context, the linear feedback method might be regarded as a better choice, although it can't be guaranteed that it is a stable control until it is tried out on an actual flexible robot first, in order to account for any unmodeled dynamics or disturbances that might occur.

Therefore, it can be concluded that while the VSC method is a more robust control, this property of robustness only occurs while in the sliding phase of the trajectory. In order to reap the benefits of the characteristics of the sliding motion, the control should ensure that the state of the system will be driven to the sliding hypersurface. The simulations shown in this chapter clearly indicate that the choice of control is greatly affected by the payload, and a good estimate should be available

to design the control such that the reaching condition is satisfied. Hence, it is critical that the payload be identified to ensure that safe motion occurs while the control attempts to drive the state towards the sliding manifold. This underscores the use of a fast identification scheme such as the neural network based approach used in this thesis.

To show a real world application of the neural network-based scheme, an example of a simple pick and place task performed by the manipulator will be shown. In this example, the flexible manipulator was simply desired to pick up a payload of $\mu = \kappa = 0.5$, move the hub and tip to one radian, drop the payload and return to the original starting point. This is a common task performed by robots, so it provides a practical example of how the control must adapt to payload changes. To show the superior performance of the NNVSC scheme over that of a fixed gain VSC, one simulation was performed with each control. The fixed gain was computed based on the heaviest payload in the range of payloads that the flexible manipulator is expected to encounter, or in this case $\mu = \kappa = 1.5$. This is the typical “worse case” scenario that has been discussed earlier, and if a non-adaptive control is used, this must be done to ensure that the bound on maximum tip deflection is not exceeded. The results of the simulations are shown in Figures 4.26, 4.27, and 4.28. These figures show how much more efficient the NNVSC scheme is than traditional VSC. The execution of the task took half the time of the fixed gain VSC for this example, showing the productivity gains that are possible with this approach. This once

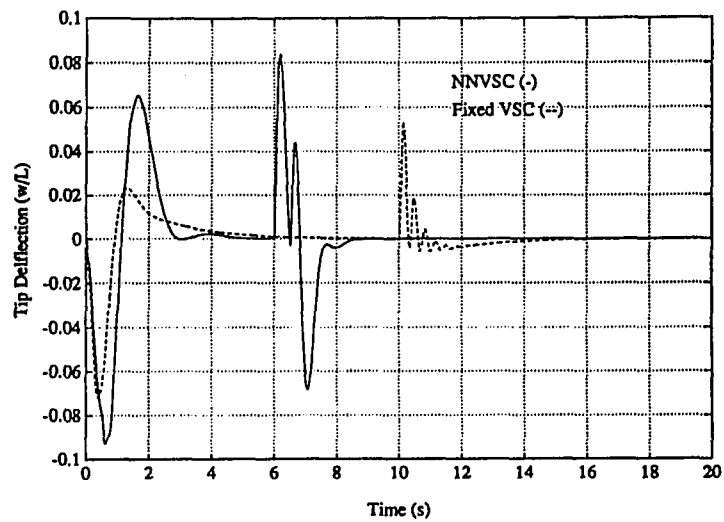


Figure 4.26: Tip Deflection Trajectory

again shows the benefits of payload identification when used in conjunction with VSC.

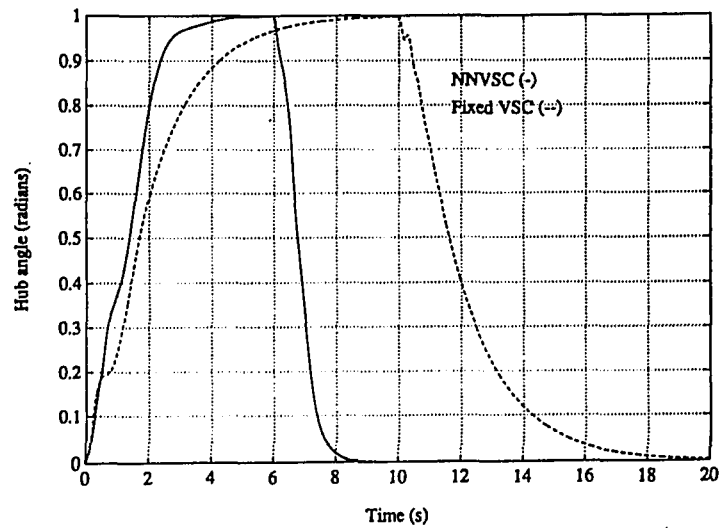


Figure 4.27: Hub Rotation Trajectory

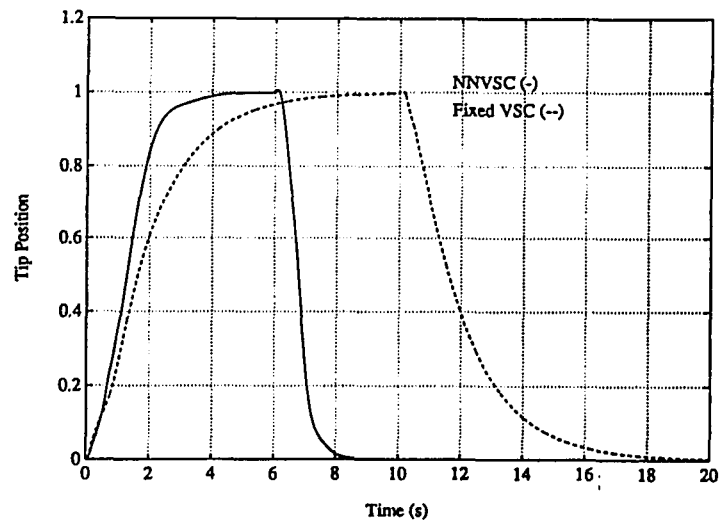


Figure 4.28: Tip Position Trajectory

CHAPTER 5

CONCLUSIONS

5.1 Introduction

In the decade since the first concentrated interest began in flexible robotics, the topics of research in this area have progressed in a logical fashion. The central issue that was focused on first was effective modeling of the complex dynamics which characterize the flexible manipulator. Gradually the research focus shifted to the synthesis of controls using these mathematical models. The next step in this natural progression is improving the robustness and adaptability of these controls to changing parameter values and disturbances to make the control schemes more effective and useful in many different environments. The focus of this thesis is in this area.

5.2 Summary of Results Reported in this Thesis

In Chapter 1, a comparison of the attributes of rigid and flexible manipulators was undertaken. Rigid robots are more precise and easier to control, but their

excessive weight is undesirable. Flexible manipulators are much lighter and use less energy but control for precise end point positioning becomes much more complex. The popular control methods for flexible manipulators were briefly described, as were the different types of neural networks and their use in control problems.

Chapter 2 set the stage for the analysis of flexible robots by showing the complete derivation of the nonlinear model to be used to simulate an actual manipulator. This chapter explicitly showed the solution of the exact linearized modal frequencies and proved the orthogonality of mode shape functions of these frequencies. The expansion of the nonlinear integro-partial differential equations using these mode shapes allowed the dynamics to be described by an ordinary differential equation, thus allowing familiar control schemes to be used on the manipulator. The controllability and observability of the linear model were checked and numerically stable methods of determining these properties were suggested. Finally, the pulse response behavior of the system was simulated for different payload masses attached to the tip of the manipulator.

Perhaps the biggest influence on a flexible manipulator is the payload which it is carrying. This effect on the system dynamics can greatly reduce the effectiveness of non-adaptive control schemes. By obtaining information on the type of payload at the tip of the manipulator and using this information to tailor the design of the controller to better meet the performance objectives specified, both productivity and accuracy can be greatly improved. In Chapter 3, the main contribution of

this thesis was introduced and developed, which specifically is a method to identify the payload. A novel scheme was designed using neural networks to quickly and accurately identify the class that a payload belonged to. Using only tip deflection data, the neural network classified the payload into one of three categories. The output of the network was used to choose the appropriate control corresponding to the class that the payload belonged to. In Chapter 3, the design of these controls was accomplished using linear pole placement techniques. While this control is quite effective, its linear nature and application to a highly nonlinear system makes it susceptible to other disturbances such as any unmodeled dynamics which may exist in an actual flexible manipulator. For this reason, a variable structure control, which is nonlinear in nature, was implemented in Chapter 4 to replace the pole placement controller.

Although VSC has been lauded for its robustness properties and some researchers have even suggested that payload identification is not necessary when using this control, it should be remembered that the robustness to parameter variations can occur only while sliding, and if variations occur which cause the system to stray from the sliding hypersurface, the variations must not be so large that the control cannot force the state of the system back to the sliding hypersurface. If the payload is unknown, the gains used to drive the states to the sliding manifold must be kept low to avoid excessive tip deflection. This reduction of gain also reduces the bound on allowable maximum parameter variations which can be rejected as disturbances

by VSC. Therefore, it is critical that the payload be identified even with robust VSC. This importance was shown in two simulations in Chapter 4. If the control is designed on the basis of a Class 0 payload and is used when the arm is carrying a Class 1 or 2 payload, the sliding mode is never attained, and the system becomes unstable. In this case, the parameters on the basis of which the control was designed are too different from the actual ones to force the system to the sliding hyperplane. These simulations emphatically underscore the need for payload identification when using VSC, and the simple neural network scheme developed in this thesis provides a fast computationally efficient method to accomplish this.

5.3 Directions for Further Research

Because this thesis covers such a wide range of topics, from neural networks to pole placement algorithms to variable structure control, a number of directions for further research could be pursued.

The issue of observer design must be addressed more rigorously and implemented with this control technique. Since the state variables are required for the control algorithms, an effective design method for an observer which can be used for each payload class needs to be developed. Likely candidates for this are variable structure observers [80] or some other nonlinear techniques [48].

Although the static multilayer neural networks trained by the backpropagation algorithm performed very well in this application, other types of networks and

training techniques are available which show improvements over the type used in this study. In an effort to improve training times, accuracy and other performance requirements, one may investigate the use of other types of neural networks, particularly recurrent networks with dynamic nodes which have shown much faster learning rates and improved performance [65, 64] over the static neural networks of this thesis. Another avenue of research which is becoming a popular topic is the use of fuzzy logic in pattern recognition tasks, and this also could be applied to this problem.

The time spent on the sliding line, and thus enjoying the benefits of excellent disturbance rejection properties, should be maximized when using VSC. This objective has been shown to be improved by adapting the sliding line using a dynamic recurrent neural network [27, 29] on rigid robots. This technique could be extended to this problem. Fuzzy logic has been explored in this area as well [31, 79], and would be a useful and interesting extension of this research.

In the design of the pole placement controller, it was difficult to obtain a method in which the transient tip deflection could be predicted by the placement of the closed-loop poles of the system, to insure that the maximum tip deflection bound was not exceeded. A more concrete relationship between these quantities would be useful in controller synthesis.

Finally, a more thorough search of possible features which are identifiable for any control by a neural network or other pattern recognition scheme would be useful.

If some payload dependent features which are always present in the outputs of the manipulator could be identified, payload identification could be run on-line all of the time and provide more frequent updates of the status of the payload. This would increase the adaptability of this method to sudden load changes. Intuitively it would seem that some use could be made of the torque applied previously and the resulting tip deflection for an update period, since heavier payloads require a larger torque to move them. This would also be a very practical direction for research.

REFERENCES

- [1] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model", *Psychological Review*, Volume 84, 1977, pp. 413-451.
- [2] E. S. Armstrong, *ORACLS: A Design System for Linear Multivariable Control*, (Dekker, New York, 1980).
- [3] E. Baribieri and Ü. Özgüner, "Unconstrained and Constrained Mode Expansion for a Flexible Slewing Link", *ASME Trans. Journ. of Dyn. Syst., Meas., and Cont.*, Volume 110, December, 1988.
- [4] G. Barna, R. Chrisley, and T. Kohonen, "Statistical Pattern Recognition with Neural Networks", *Neural Networks* Volume 1, No. 1, 1988, p. 7.
- [5] R. H. Bartels and G. H. Stewart, "Algorithm 432, a Solution of the Matrix Equation $AX+XB=C$ ", *Commun. Ass. Comput. Mach.* Volume 15, pp. 820-826, 1972.
- [6] E. Bayo, "Computed Torque for the Position Control of Open-Chain Flexible Robots", *IEEE 1988 International Conference on Robotics and Automation*, Volume 1, 1988.
- [7] E. Bayo, "A Finite-Element Approach to Control the End-point Motion of a Single-Link Flexible Robot", *Journal of Robotic Systems*, Volume 4, No. 1, 1987, pp. 63-75.
- [8] E. Bayo and H. Moulin, "An Efficient Computation of the Inverse Dynamics of Flexible Manipulators in the Time Domain", *IEEE 1989 International Conference on Robotics and Automation*, Volume 2, 1989, pp.710-715.
- [9] E. Bayo, R. Movaghar, and M. Medus, "Inverse Dynamics of a Single-Link Flexible Robot", *Int. Journal of Robotics and Automation*, Volume 3, No. 3, Fall 1988.
- [10] F. Bellezze, L. Lanari and G. Ulivi, "Exact Modeling of the Flexible Slewing Link", *IEEE 1990 International Conference on Robotics and Automation*, Volume 2, 1990, pp.734-739.
- [11] W. J. Book, "Recursive Lagrangian Dynamics of Flexible Manipulator Arms Via Transmission Matrices", *Proc. Second IFAC Symposium on CAD of Multivariable Technological Systems*, West Lafayette, IN., Sept. 1982, pp.5-17.

- [12] Z. -E. Boutaghou and A. G. Erdman, "A Unified Approach for the Dynamics of Beams Undergoing Arbitrary Spatial Motion", , *Journal of Vibration and Acoustics-Transactions of the ASME*, Volume 113, No. 4 October, 1991, pp.494-507.
- [13] J. A. Burton and A. S. I. Zinober, "Continuous Approximation of Variable Structure Control", *International Journal of Systems Science*, Volume 17, 1986, pp. 876-885.
- [14] R. H. Cannon, Jr. and Schmitz, E., "Precise Control of Flexible Manipulators", *Robotics Research: The First International Symposium*, (MIT Press, Cambridge, MA 1984) pp. 841-861.
- [15] G. A. Carpenter and S. Grossberg, "Neural Dynamics of Category Learning and Recognition: Attention, Memory Consolidation, and Amnesia", *Brain Structure, Learning, and Memory*, J. Davis, R. Newburgh, and E. Wegman, Eds. (AAAS Symposium Series, 1986).
- [16] Chi-Tsong Chen, *Linear System Theory and Design*, Second Edition. (Holt, Rhinehart and Winston, Inc., New York, 1984).
- [17] G. Cybenko, "Continuous Value Neural Networks with Two Hidden Layers are Sufficient", *Mathematics of Controls, Signals, and Systems*, Volume 2, 1989, pp. 303-314.
- [18] A. De Luca, P. Lucibello, and G. Ulivi, "Inversion Techniques for Trajectory Control of Flexible Robot Arms", *Journal of Robotic Systems* Vol. 6, No. 4, August 1989, pp. 325-344.
- [19] K. Funahashi, "On the Approximate Realization of Continuous Mappings by Neural Networks", *Neural Networks*, Volume 2, 1989, pp. 183-192.
- [20] G. H. Golub, S. Nash, and C. Van Loan, "A Hessenberg-Scheer Method for the problem $AX + XB = C$ ", *IEEE Transactions on Automatic Control*, Volume AC-24, 1979, pp. 909-913.
- [21] G. Guoguang, "Modeling and Control of a One-Link Flexible Manipulator", *Master's Thesis*, Department of Systems and Industrial Engineering, The University of Arizona, 1991.
- [22] L. Gupta and M. R. Sayeh, "Neural Networks for Planar Shape Classification", *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1988, pp. 936-939.
- [23] Gordon G. Hastings and Wayne J. Book, "Verification of a Linear Dynamic Model for Flexible Robotic Manipulators", *IEEE 1986 International Conference on Robotics And Automation*, Volume 2, 1986.

- [24] G. Hohenbichler, P. Plöckinger, and P. Lugner, "Comparison of a Modal-Expansion- and a Finite-Element-Model for a Two-Beam Flexible Robot Arm", *IFAC Proceedings Series n. 10*, 1989, pp. 35-39.
- [25] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Sciences*, Volume 74, 1982, pp. 2554-2558.
- [26] K. Hornik, M. Strinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, Volume 2, 1989, pp. 359-366.
- [27] A. Karakaşoğlu, "Neural Network-Based Approaches to Controller Design for Robot Manipulators", *Phd. Dissertation*, Department of Electrical and Computer Engineering, The University of Arizona, 1991.
- [28] A. Karakaşoğlu, S. I. Sudharsanan and M. K. Sundareshan, "Identification and Decentralized Adaptive Control of Robotic Manipulators using Dynamical Neural Networks", *Proceedings of the 1991 Int. Joint Conference on Neural Networks(IJCNN-91)*, Seattle, WA, July 1991.
- [29] A. Karakaşoğlu and M. K. Sundareshan, "A Recurrent Neural Network-Based Adaptive Variable Structure Model Following Control of Multijointed Robotic Manipulators", *Proceedings of the 31st IEEE Conference on Decision and Control*, Tucson, AZ, Dec. 1992.
- [30] J. Kautsky, N. K. Nichols, and P. Van Dooren "Robust Pole Assignment in Linear State Feedback", *International Journal of Control* Volume 41, No. 5, 1985, pp. 1129-1155.
- [31] A. Knafel, R. Swiniarski, and M. B. Zaremba, "Fuzzy Logic Control for Variable Structure Systems", *IEEE Conference on Systems Engineering*, 1989, pp. 419-422.
- [32] T. Kohonen, *Self-Organization and Associative Memory*, (Springer-Verlag, Berlin, 1984).
- [33] H. Krishnan and M. Vidyasagar, "Control of a Single-Link Flexible Beam Using a Hankel-Norm-Based Reduced Order Model", *IEEE 1988 International Conference on Robotics and Automation*, Volume 1, 1988.
- [34] M. Kuperstein and J. Wang, "Neural Controller for Adaptive Movements with Unforeseen Payloads", *IEEE Transactions on Neural Networks*, Volume 1, No. 1, March 1990, pp. 137-142.
- [35] M. Leahy, M. Johnson, and S. Rogers, "Neural Network Payload Estimation for Adaptive Robot Control", *IEEE Transactions on Neural Networks*, Volume 2, No. 1, January 1991, pp. 93-100.

- [36] H. Lee and I. A. Castelazo, "Nonlinear Feedback Control of a Flexible Robot Arm", *ASME, Dynamic Systems and Control Division (DSC)*, Volume 6, 1987, pp. 307-314.
- [37] J. D. Lee, "Application of Optimal Control Theory to Flexible Robotic Manipulators", *Robotics and Computer-Integrated Manufacturing*, Volume 7, No. 3/4, 1990, pp. 327-344.
- [38] S. Lehar, "Application of Back Propagation to Long Wave Infra-Red Signature Analysis", *Neural Networks* Volume 1, No. 1, 1988, p. 454.
- [39] L. -C. Lin, "State Feedback H_∞ Control of Manipulators with Flexible Joints and Links", *IEEE International Conference on Robotics and Automation*, Volume 1, 1991, pp. 218-223.
- [40] S.-H. Lin, S. Tosunoglu, and D. Tesar, "A Controller Design for Compliant Manipulators Modeled with Elastic Links and Joints", *Proceedings of the 29th IEEE Conference on Decision and Control*, Part 3, 1990, pp. 1936-1942.
- [41] R. Lippman, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, pp. 4-22, April 1987.
- [42] L. Liu, L. Lee, H. Wang and Y. Chang, "Layered Neural Nets Applied in the Recognition of Voiceless Unaspirated Stops", *IEE Proceedings, Part I: Communications, Speech and Vision*, Volume 138, No. 2, April 1991, pp. 69-75.
- [43] W.S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, Volume 5, 1943 pp. 115-133.
- [44] H. Midorikawa, "Face Pattern Identification by Back-Propagation Learning Procedure", *Neural Networks*, Volume 1, No. 1, 1988 p. 515.
- [45] R. G. Morgan and Ü. Özgüner, "A decentralized Variable Structure Control Algorithm for Robotic Manipulators", *IEEE Journal of Robotics and Automation*, Volume RA-1, No. 1, March 1985, pp. 57-65.
- [46] S. Narendra and A. N. Annaswamy, *Stable Adaptive Systems*, (Prentice-Hall, Englewood Cliffs, NJ, 1989).
- [47] P. Nathan and S. Singh, "Variable Structure Control of a Robotic Arm with Flexible Links", *1989 IEEE International Conference on Robotics and Automation*, Volume 2, 1989, pp. 882-887.
- [48] S. Nicosia, P. Tomei, and A. Tornambe, "Non-Linear Control and Observation Algorithms for a Single-Link Flexible Robot Arm", *International Journal of Control*, Volume 49, No. 3, March 1989, pp. 827-840.
- [49] Yoh-Han Pao, *Adaptive Pattern Recognition and Neural Networks*, (Addison-Wesley, Reading, MA 1989).

- [50] T. R. Parks and H. A. Pak, "Effect of Payload on the Dynamics of a Flexible Manipulator - Modeling for Control", *Journal of Dynamic Systems, Measurement, and Control*, Volume 113, September 1991, pp. 409-418.
- [51] S. M. Peeling, R. K. Moore, and M. J. Tomlinson, "The Multi-layer Perceptron as a Tool for Speech Pattern Processing Research", *Proceedings of the IoA Autumn Conference on Speech and Hearing*, 1986.
- [52] W. T. Qian and C. C. H. Ma, "A New Controller Design for a Flexible One-Link Manipulator", *IEEE Transactions on Automatic Control*, Volume 37, No. 1, January 1992, pp. 132-137.
- [53] L. C. Rabelo and X. J. Avala, "Hierarchical Neurocontroller Architecture for Intelligent Robotic Manipulation", *IEEE International Conference on Robotics and Automation*, Volume 3, 1991 pp. 2656-2661.
- [54] I. P. Roberts, "Neural Network for Radar Terrain Image Recognition", *Neural Networks*, Volume 1, No. 1, 1988, p. 463.
- [55] D. M. Rovner and R. H. Cannon, "Experiments Toward On-Line Identification and Control of a Very Flexible One-Link Manipulator", *The International Journal of Robotics Research*, Volume 6, no. 4, Winter 1987, pp. 3-19.
- [56] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing*, D. Rumelhart and J. McClelland (Eds.), Volume 1, (MIT Press, Cambridge, MA, 1986).
- [57] E. P. Ryan and M. Corless, "Ultimate Boundedness and Asymptotic Stability of a Class of Uncertain Dynamical Systems via Continuous and Discontinuous Feedback Control", *IMA J. Math. Control Information*, Volume 1, pp. 223-242.
- [58] J. Z. Sasiadek and R. Srinivasan, "Dynamic Modeling and Adaptive Control of a Single-Link Flexible Manipulator", *Journal of Guidance, Control and Dynamics*, Volume 12, No. 6, Nov.-Dec. 1988.
- [59] D. A. Schoenwald and Ü. Özgüner, "On Combining Slewing and Vibration Control in Flexible Manipulators via Singular Perturbations", *Proceedings of the 29th Conference on Decision and Control*, 1990, pp. 533-538.
- [60] I. Y. Shung and M. Vidyasagar, "Control of a Flexible Robot Arm with Bounded Input: Optimum Step Responses", *IEEE Conference on Robotics and Automation*, Volume 2, 1987, pp. 916-922.
- [61] B. Siciliano and W. J. Book, "A Singular Perturbation Approach to Control of Lightweight Flexible Manipulators", *The International Journal of Robotics Research*, Volume 7, No. 4, August 1988, pp. 79-90.
- [62] T. Singh, M. F. Golnaraghi, and R. N. Dubey, "Variable Structure Control of a Single-link Flexible Arm Robot", *Proceedings of the American Control Conference*, 1990, pp. 702-703.

- [63] J. -J. E. Slotine, "The Robust Control of Robot Manipulators", *International Journal of Robotic Research*, Volume 4, 1985, pp. 49-64.
- [64] S. I. Sudharsanan, "Equilibrium Characterization for a Class of Dynamical Neural Networks with Applications to Learning and Synthesis", *Ph.D. Dissertation*, Dept. of Electrical and Computer Engineering, The University of Arizona, 1991.
- [65] S. I. Sudharsanan and M. K. Sundareshan, "Training of a Three-Layer Recurrent Neural Network for Nonlinear Input-Output Mapping", *Proceedings of the 1991 Int. Joint Conference on Neural Networks (IJCNN-91)*, Seattle, WA, July 1991.
- [66] K. Sung, P. Kudva, and J. C. S. Yang, "Parameter Identification of a Flexible Manipulator", *Proceedings of the 1987 International Conference on Systems, Man, and Cybernetics*, pp. 578-582.
- [67] S. Timoshenko and G. H. MacCullough, *Elements of Strength of Materials*, Third Edition. (D. Van Nostrand Company, Inc., New York, 1949).
- [68] A. P. Tzes and S. Yurkovich, "Application and Comparison of On-Line Identification Methods for Flexible Manipulator Control", *The International Journal of Robotics Research*, Volume 10, No. 5, October, 1991, pp. 515-527.
- [69] V. I. Utkin, *Sliding Modes and their Application to Variable Structure Systems*, (MIR Publishers, Moscow, USSR, 1978).
- [70] D. Vinke and M. Vidyasagar, "New Techniques for H_2 Optimal Control of a Flexible Beam", *Proceedings of the 1991 IEEE Conference on Robotics and Automation*, Volume 3, pp. 2592-2597.
- [71] Fei-Yue Wang and John T. Wen, "Nonlinear Dynamical Model and Control for a Flexible Beam", Rensselaer Polytechnic Inst. Electrical, Computer, and System Engineering, CIRSSE Report # 75, Nov. 1990.
- [72] Fei-Yue Wang, "Frequency Sensitivity and Mode Orthogonality of One-Link Flexible Robot Arms: A Variational Approach", *SIE Working Paper*, Dept. of Systems and Industrial Engineering, The University of Arizona, 1992.
- [73] Fei-Yue Wang, "Modeling, Analysis, and Simulation of Lightweight Robot Arms with Consideration of Rotary Inertia and Shear Deformation", *SIE working paper*, Dept. of Systems and Industrial Engineering, The University of Arizona, 1991.
- [74] P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", *Ph.D. Dissertation*, Harvard University, Cambridge, MA 1974.

- [75] T. C. Yang, J. C. S. Yang, and P. Kudva, "Adaptive Control of a Single-Link Flexible Manipulator with Unknown Load", *IEE Proceedings, Part D: Control Theory and Applications*, Volume 138, No. 2, March 1991, pp. 153-159.
- [76] K. S. Yeung and Y. P. Chen, "Regulation of a One-link Flexible Robot Arm using Sliding-mode Technique", *International Journal of Control*, Volume 49, No. 6, June 1989, pp. 1965-1978.
- [77] J. Yuh, "Application of Discrete-Time Model Reference Adaptive Control of A Flexible Single-Link Robot", *Journal of Robotic Systems*, Volume 4, No.5, Oct. 1987.
- [78] S. Yurkovich, F. E. Pacheco, and A. P. Tzes, "On-Line Frequency Domain Information for Control of a Flexible-Link Robot with varying Payload", *IEEE International Conference on Robotics and Automation - 1989*, Volume 2, pp. 876-881.
- [79] M. B. Zaremba, "Design of Robust VSS Controllers", *Advances in Instrumentation, Proceedings* Volume 44, Part 4, 1989, pp. 1647-1652.
- [80] A. S. I. Zinober, ed. *Deterministic Control of Uncertain Systems*, IEE Control Engineering Series, Volume 40. (Peter Peregrinus, Ltd., London, U. K., 1990).