THESIS REPORT

Master's Degree

Linear Control Theory as Applied to Smart Structures

by G.A. Kantor
Advisor: W.P. Dayawansa

M.S. 95 -12

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Abstract

Title of Thesis: Linear Control Theory
as Applied to Smart Structures

Name of degree candidate: George A. Kantor

Degree and year: Master of Science, 1995

Thesis directed by: Associate Professor W.P. Dayawansa
Department of Electrical Engineering and
Institute for Systems Research

This thesis investigates linear control theory as applied to smart structures. Specifically, the problem of active vibration damping in a flexible cantilever beam using piezo-electric ceramic crystals (PZT) as sensors and actuators is addressed.

The problem of controlling linear time invariant systems subject to hard input constraints is considered. Some existing methods are reviewed and some new methods are presented.

The subject of rapid prototyping and automatic system identification is addressed. Automatic system identification techniques are developed and combined with a comercially available rapid prototyping system to create a test bed which can be used to rapidly model and control existing plants.

Closed loop control laws designed to actively damp vibrations in a flexible cantilever beam were tested in various laboratory experiments. The experiments

are described, the design of the control laws is discussed, and the results of the experiments are presented.

Finally, impact between a flexible robotic arm and a fixed sphere is adressed. A model for the impact forces is developed and compared with experimentally determined impact data. An open loop control law using a neural network is implemented to control the magnitude of the impact force.

Linear Control Theory as Applied to Smart Structures

by

George A. Kantor

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1995

Advisory Committee:

Associate Professor W.P. Dayawansa, Chairman/Advisor
Professor P.S. Krishnaprasad (Co-Advisor)
Associate Professor J.S. Sirkis

Acknowledgements

I would like to thank my advisors, Dr. W.P. Dayawansa and Dr. P.S. Krishnaprasad for all of their guidance over the course of this project. I would also like to extend my thanks to Dr. J.S. Sirkis for taking the time to be the third committee member. Finally, I would like to thank all of the students in the Intelligent Servosystems Laboratory for their support and encouragement.

This research was supported in part by the Army Research Office University grant #DAAL03-92-G01201, the Engineering Research Program grant #CD 88030122, and National Science Foundation grant #ECE 9096121

Table of Contents

List of Figures vii

1 Introduction 1

2 Finding the Controllable Set 5

2.1 Preliminaries 6

2.2 Approximating the Controllable Set 9

2.3 Examples 11

2.3.1 Example 1: 2-D Case With Real Eigenvalues 11

2.3.2 Example 2: 2-D Case With Complex Eigenvalue 13

3 Saturating Control Laws 17

3.1 Classification Controllers 17

3.1.1 Nearest Neighbor 19

3.1.2 Look-up Table 20

3.1.3 Neural Network 20

3.1.4 An Example 21

3.2 Modified Linear Controllers 26

3.2.1 Preliminaries 26

3.2.2 Gutman-Hagander 27

3.2.3 Control Using the *msat* Function 28
3.2.4 Dual Mode Controllers 31
3.3 Conclusion 33

4 Rapid Prototyping Using the AC-100 **35**
4.1 The AC-100 36
4.1.1 MatrixX and SystemBuild 36
4.1.2 Interactive Animation Builder 38
4.1.3 Hardware Connection Editor 39
4.1.4 Compiling the Controller 39
4.2 Physical Description 40
4.3 Conclusion 40

5 System Identification **42**
5.1 Recursive Least Squares 43
5.1.1 The Model Structure 43
5.1.2 Calculating the Error 44
5.1.3 Updating the Model Parameters 45
5.1.4 Convergence of Model Parameters 46
5.1.5 Example: Identification of a DC Motor 47
5.2 Identification Using Wavelets 50
5.2.1 Using the Matching Pursuits Software 50
5.2.2 Example: Flexible Beam 51
5.2.3 Conclusion 53

6 Fixed Flexible Beam Experiment **55**
6.1 Experimental Set Up 56

6.1.1 The PZT Pairs 56
6.1.2 Electrical Description 59
6.2 Modeling the System 60
6.2.1 Finite Element Model 62
6.2.2 Experimental Model 63
6.3 Controller Design 64
6.3.1 Observer Design 64
6.3.2 Linear Quadratic Regulator 66
6.3.3 Gutman-Hagander Controller 67
6.3.4 Positive Position Feedback 69
6.4 Results 70
6.5 Conclusion 74

7 Rotating Flexible Beam Experiment **75**

7.1 Experimental Set Up 76
7.2 Modeling the System 77
7.2.1 Finite Element Model 77
7.2.2 Experimental Model 82
7.3 Controller Design 84
7.4 Results 85
7.4.1 Performance at Fixed Motor Velocities 86
7.4.2 Slowly Varying Motor Velocities 86
7.5 Conclusion 90

8 Impact of a Flexible Robotic Arm **92**

8.1 Experimental Set Up 93

8.2 Modeling the System 94
8.2.1 The Finite Element Model 95
8.2.2 Hertz Law of Impact 98
8.2.3 Computing the Impact Force Numerically 101
8.3 A Neural Network Impact Controller 104
8.3.1 The Input Signal 104
8.3.2 Collecting Data 104
8.3.3 Training the Neural Network 105
8.3.4 Implementation 106
8.4 Conclusion 107

9 Conclusion **110**

Bibliography **113**

List of Figures

2.1 Geometrical Interpretation of $\langle h, x \rangle$ 8

2.2 Approximation of Controllable Set for Example 1 12

2.3 Exact Controllable Set for Example 1 14

2.4 Approximate Controllable Set for Example 2 15

2.5 Trajectory of Reverse Time System (Example 2) with Constant Input 16

3.1 Example using nearest neighbor control. (a) switching curve. (b) trajectory of system under nearest neighbor control 23

3.2 Example with data set preprocessed into a 2 dimensional matrix. (a) approximate switching curve. (b) trajectory under control . . 24

3.3 Example using a neural network for feedback control. (a)actual and approximated switching curves. (b) trajectory under control. 25

3.4 Important sets for the *msat* controller 30

3.5 Important sets for the dual mode controller 32

4.1 AC-100 block diagram 37

4.2 The AC-100 cart 41

5.1 Block diagram of Recursive Least Squares estimation 47

5.2 Experimental set up of DC motor for RLS identification 48
5.3 Parameters estimates of DC motor identification 49
5.4 Step response of DC motor and estimated model 50
5.5 Flexible frequency response 52

6.1 Flexible beam diagram (not to scale) 57
6.2 PZT mounting scheme 59
6.3 Low pass filter schematic 60
6.4 Power amplifier schematic 61
6.5 Block diagram of experimental set up 61
6.6 3 element Galerkin model 63
6.7 Block diagram of PPF controller 71
6.8 Response of the beam under LQR control 72
6.9 Response of the beam under GH control 72
6.10 Response of the beam under PPF control 73
6.11 Comparison of beam response envelopes 73

7.1 3 element Galerkin model of continuously rotating flexible beam . 78
7.2 Frequency response of rotating beam, $\omega_m = 0$ 83
7.3 Frequency response of rotating beam, $\omega_m = 4$ 84
7.4 Block diagram of experimental set-up 86
7.5 Disturbance response of beam at $\omega_m = 0$ rad/sec 87
7.6 Disturbance response of beam at $\omega_m = 4$ rad/sec 87
7.7 Disturbance response of beam using no control 88
7.8 Disturbance response of beam using controller $H_0(s)$ 89
7.9 Disturbance response of beam using controller $H_4(s)$ 89

7.10 Disturbance response of beam using gain scheduling controller . . 90

8.1 Mounting of the FSR (force sensitive resistor) 93

8.2 Flexible arm set-up 94

8.3 Electrical set up of FSR (force sensitive resistor) 95

8.4 Galerkin model of flexible arm 98

8.5 Relative displacement between beam and sphere during impact . . 101

8.6 Model generated impact force as a function of time 104

8.7 Measured impact force as a function of time 104

8.8 Motor control signal parameterized by T 106

8.9 Training data for neural network 107

8.10 Performance of trained neural network 108

8.11 Block diagram of neural network controller 109

8.12 Performance of neural network controller 109

Chapter 1

Introduction

Recent years have seen a large amount of interest generated on the subject of smart structures. While there are no rigorous criteria which characterize them, smart structures are generally constructed of components which have sensors and/or actuators integrated within the natural structure of the material. This thesis investigates the aspects of automatic control associated with smart structures. Specifically, the problem of vibration damping in a flexible aluminum cantilever beam using piezo-electric ceramic crystals (PZT) mounted to the surface of the beam as both sensors and actuators is addressed.

This thesis can be broken into three distinct parts. The first part, Chapters 2 and 3, deals with the problem of controlling linear systems with input constraints. The actuators employed in smart structures are generally small in size so that they can be integrated seamlessly within the material. This often means that the control effort exerted by the actuator is also small. Hence constraints on the input need to be considered when designing a controller. The aluminum beam with PZT actuators is a good example of this. Safety and practical considerations limit the maximum voltage which can be applied to the PZT. Even at this

maximum voltage, the force applied by the PZT is small compared to the force required to bend the beam. Linear control theory contains a large number of tools for designing controllers for linear systems, but a surprisingly small amount of work has been done on designing controllers for linear systems with input constraints.

The first step in approaching the constrained input problem is to determine the controllable set of the system. This is the subject of Chapter 2. The controllable set is the set of all initial conditions for which there exists a control which drives the system from the initial condition to the origin in finite time while satisfying the input constraints. The controllable set can be approximated by applying the maximum principle to the reverse time system. This method not only provides an approximation of the controllable set, but it also provides a method of finding an open loop control which will drive any initial condition to the origin in the shortest time possible.

Chapter 3 discusses some control laws which are designed to stabilize an unstable linear system while obeying the input constraints. The controllers presented in Chapter 3 can be broken into two groups: classification controllers and modified linear controllers. As their name suggests, the first group uses classification techniques such as nearest neighbor, look-up tables, and neural networks to transform the open loop control law derived in Chapter 2 into an implementable closed loop control law. The modified linear controllers begin with a low gain linear feedback law which stabilizes the plant in a desired region of the state space while obeying the input constraint. This low gain controller is then modified to improve performance of the closed loop system. The Gutman-Hagander control law [13] is an example of a modified linear controller. The classification

controllers have the advantage that they stabilize every initial condition in the controllable set while the modified linear controllers are generally more easily implemented.

The second part, Chapters 4 and 5, describes the development of the tools necessary to perform the experimental work to follow. Chapter 4 describes the AC-100, a system that can be used to rapidly design and implement controllers. The AC-100 was used to help implement the controllers for all of the experimental work in Chapters 5-8.

Most modern controller design techniques, including those discussed in Chapter 3, require an accurate model of the system to be controlled. Models derived from physical principles often contain parameters which are difficult to measure accurately. Hence, an empirical method of determining a model for an arbitrary plant is sought. Chapter 5 presents two methods of experimentally determining the transfer function of any stable linear time invariant system. These methods are recursive least squares (RLS) and matching pursuits (MP). RLS is capable of estimating a real rational transfer function model online while MP is a wavelet based method capable of generating accurate low order models. Both methods are implemented using the AC-100.

The third part, Chapters 6, 7, and 8, discusses the development and implementation of controllers for the flexible cantilever beam in different configurations. The configuration used in Chapter 6 is a flexible aluminum beam with PZT sensors and actuators mounted to a fixed base. The control objective is to actively damp vibrations in the beam using closed loop control. Here, the constraint satisfying Gutman-Hagander is compared with two controllers which do not incorporate any input constraints, the linear quadratic regulator (LQR)

and the positive position feedback (PPF) controller.

In Chapter 7, the beam from Chapter 6 is mounted to the hub of a DC motor. The motivation for this configuration comes from the desire to study and control vibrations in spinning objects such as the blades of a fan or a helicopter rotor. The rotation creates an interesting variation of the fixed beam problem because the parameters of the beam change as the angular velocity of the hub increases. A model for this system is developed and linear controllers are designed to damp vibrations in the beam at fixed motor velocities. These controllers are then joined together to form a gain scheduling controller for the system with varying motor velocity.

The effect of impact forces on flexible robotic manipulators is the subject of a great deal of modern research. In chapter 8, a one link flexible robot arm consisting of an aluminum beam attached to the hub of a DC motor is considered. A model which can predict the impact forces generated in a collision between the arm and a fixed aluminum sphere is derived using the Hertz law of impact. The predictions of this model are then compared with experimental impact data. Finally, a neural network is trained from experimental data to generate an open loop motor control signal which will bring the arm into contact with the sphere to create a specified impact force.

In the final chapter, the main results of the thesis are briefly reviewed. As always, many interesting problems remain. Some of these are identified as subjects for future research.

Chapter 2

Finding the Controllable Set

The plant to be considered is a controllable linear time invariant system of dimension n, represented as follows:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.1}$$

where $x \in R^n$ and $u \in R^m$. In addition, the control inputs are constrained to be such that

$$|u_i| \leq k_i, \qquad i = 1, 2, \ldots, m. \tag{2.2}$$

The goal is to find a control $u : [0, \infty) \rightarrow R^m$ that drives the state from some $x_0 = x(0)$ to the origin. For controllable linear time invariant systems without input constraints it is well known that this goal can not only be satisfied, but that any $x_0 \in R^n$ can be driven to the origin in arbitrarily small time[29]. In the case with constrained inputs, however, this result does not hold. Given a system described by Equations 2.1 and 2.2 there may be initial conditions that cannot be driven to the origin, even when given an infinite amount of time. With this in mind, the objective of this chapter is to find the set of all initial conditions x_0 for which there exists a control which will drive the state of Equation 2.1 from x_0 to the origin.

2.1 Preliminaries

A formal concept of Controllable and Reachable Sets must first be defined:

Definition 2.1 *Let $\phi(t, x_0, u)$ denote the solution at time t to Equation 2.1 driven by control u with the initial condition $x(0) = x_0$. Given some $T > 0$, the* **Controllable Set at time T**, *denoted by $\mathcal{C}_T$, is defined to be the set of all $p \in R^n$ for which there exists a control $u : [0, T] \to R^m$ which satisfies Equation 2.2 such that $\phi(T, p, u) = 0$. The* **Controllable Set** *denoted by $\mathcal{C}$ is defined to be the limit as $T \to \infty$ of $\mathcal{C}_T$.*

Definition 2.2 *Given $T > 0$, the* **Reachable Set at time T**, *denoted by $\mathcal{R}_T$, is defined to be the set of all $p \in R^n$ for which there exists a control $u : [0, T] \to R^m$ which satisfies Equation 2.2 such that $\phi(T, 0, u) = p$. The* **Reachable Set**, *denoted by $\mathcal{R}$, is defined to be the limit as $T \to \infty$ of $\mathcal{R}_T$.*

Theorem 2.1 ([25]) *Consider the linear time invariant system in Equation 2.1 with initial condition $x(0) = 0$ subject to the input constraint given by Equation 2.2. Let $T \in [0, \infty)$. The reachable set $\mathcal{R}_T$ is convex and compact.*

Note that if the control $u : [0, \infty) \to R^m$ drives the system from x_0 to 0 in time T, then the control u^B defined by

$$u^B(t) = u(T - t), \quad t \in [0, T]$$

drives the system

$$\dot{x}^B(t) = -(Ax^B(t) + Bu^B(t)) \tag{2.3}$$

from 0 back to x_0 in time T. Equation 2.3 can be thought of as the reverse time system corresponding to Equation 2.1. This leads to the following fact:

Fact 2.1 *The controllable set for a given system is equivalent to the reachable set for the corresponding reverse time system.*

Fact 2.1 will be exploited to help compute the controllable set of.

Corollary 2.1 *Let $T \in [0, \infty)$. The controllable set $\mathcal{C}_T$ of the system given by Equation 2.1 subject to the input constraint given by Equation 2.2 is convex and compact.*

Corollary 2.1 follows from Theorem 2.1 using Fact 2.1.

Fact 2.2 *If the matrix A in Equation 2.1 is Hurwitz, then the controllable set is R^n.*

This fact is clear. When A is Hurwitz (i.e. all of the eigenvalues of A have negative real part), the system with $u \equiv 0$ is exponentially stable. Solutions starting from all initial conditions in R^n drift to the origin.

In the sequel it will be necessary to identify points on the boundary of $\mathcal{R}_T$ and $\mathcal{C}_T$. This will be done by making use of the following fact:

Fact 2.3 *Let Ω be a convex, compact subset of R^n which contains the origin. Let $\partial\Omega$ denote the boundary of Ω. Then $x \in \partial\Omega$ if and only if*

$$\langle h, x \rangle \geq \langle h, y \rangle$$

for all $y \in \Omega$, some $h \in R^n$.

The quantity $\langle h, x \rangle$ can be thought of as the length of the projection of x onto h (see Figure 2.1). This quantity is maximized only if $x \in \partial\Omega$. To see this, suppose x maximizes $\langle h, x \rangle$ such that $x \in \Omega$. If $x \in int(\Omega)$, then there exists

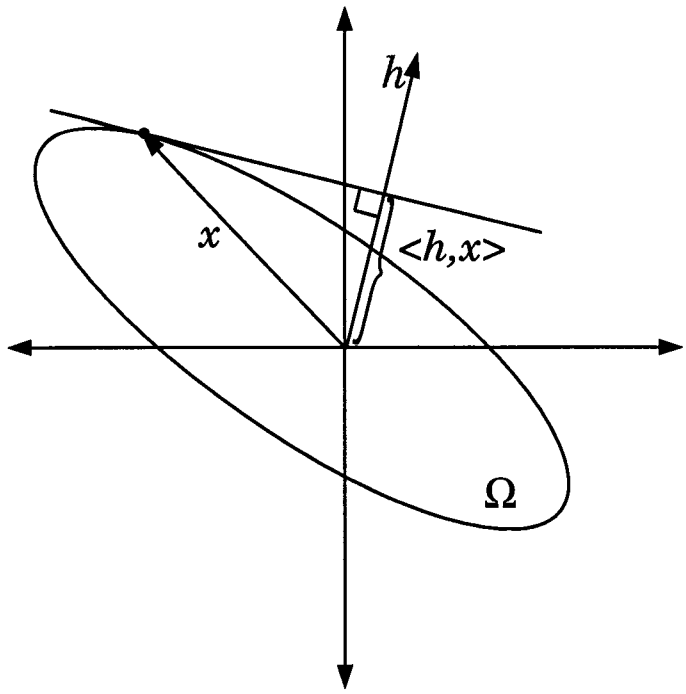

Figure 2.1: Geometrical Interpretation of $\langle h, x \rangle$

$\epsilon > 0$ such that $(x + \epsilon h) \in \Omega$. Clearly, $\langle (x + \epsilon h), h \rangle > \langle x, h \rangle$, so by contradiction x must be on the boundary of Ω. Conversely, if $x \in \partial\Omega$ then by convexity of Ω there exists $h \in R^n$ such that $\langle h, x \rangle \geq \langle h, y \rangle$ for all $y \in \Omega$.

Finally, the maximum principle will be needed to solve for the controllable set. For the purposes of this paper, it can be stated as follows:

Theorem 2.2 (Maximum Principle [31]) *Consider the linear time invariant given by Equation system 2.1 with initial condition $x(0) = 0$ subject to the input constraint given by Equation 2.2. Let $T \in [0, \infty)$. Let $p(t) \in R^n$ satisfy the adjoint equation*

$$\dot{p}(t) = -A^T(t)p(t), \qquad p(T) = h \tag{2.4}$$

Then $u : [0,T] \rightarrow R^m$ *maximizes the quantity* $\langle h, x(T) \rangle$ *if and only if*

$$p^T(t)Bu(t) = sup\{p^T(t)Bv \Big| v \in R^m, |v_i| \leq k_i, i = 1, 2, \ldots, m\} \tag{2.5}$$

for all $t \in [0,T]$.

Letting b_i be the ith column of B yields

$$p^T(t)Bv = \sum_{i=1}^{m} p^T(t)b_i v_i. \tag{2.6}$$

The value of $p^T(t)Bv$ attains its supremum if and only if every element of the summation in the right hand side of Equation 2.6 attains its supremum. Hence, given $p(t)$, the optimal $u(t)$ can be found by maximizing each element of the sum. This leads to the following formula for $u(t)$:

$$u_i(t) = \begin{cases} k_i & \text{if} \quad p^T(t)b_i > 0 \\ -k_i & \text{if} \quad p^T(t)b_i < 0 \\ \text{don't care} & \text{if} \quad p^T(t)b_i = 0 \end{cases} \tag{2.7}$$

2.2 Approximating the Controllable Set

Let $\mathcal{C}_T$ denote the controllable set at time T of the system given by Equation 2.1 subject to the input constraint given by Equation 2.2. One way of finding a point on the boundary of $\mathcal{C}_T$ is to find the control which drives the state from the initial condition $x(0) = x_0$ to the final condition $x(T) = 0$ for the initial condition x_0 which maximizes the quantity $\langle h, x_0 \rangle$ for some $h \in R^n$. Hence, one is faced with an optimal control problem on a system with varying initial conditions, a problem which is not easily solved in general.

The properties of the reverse time system provide for an easier solution to this problem. Let $\mathcal{R}_T$ denote the reachable set at time T of the reverse time

system with u^B subject to the constraint and with the initial condition $x(0) = 0$. Since the controllable set of any system is equivalent to the reachable set of the corresponding reverse time system, $\mathcal{R}_T = \mathcal{C}_T$.

To find a point on the the boundary of $\mathcal{R}_T$ we need to find $u^B : [0, T] \rightarrow R^m$ such that the quantity $\langle h, x(T) \rangle$ is maximized. This control is easily found using the maximum principle. The reverse time adjoint system is given by

$$\dot{p}^B(t) = A^T p^B(t); \qquad p^B(T) = h. \tag{2.8}$$

Let $\Phi_A(\cdot, \cdot)$ denote the transition matrix of A. Then the solution of Equation 2.8 can be written as

$$p^B(t) = \Phi^T_{-A}(T, t)h. \tag{2.9}$$

Substituting into Equation 2.7 shows that the control which maximizes $\langle h, x(T) \rangle$ for the reverse time system is given by

$$u^B_j(t, h) = k_j sgn(-h^T \Phi_{-A}(T, t) b_j), \qquad j = 1, 2, \ldots, m. \tag{2.10}$$

This is a slightly stonger version of the result given by Pecsvaradi and Narendra [25].

Clearly, for any $h \in R^n$, all points on the trajectory of the reverse time system under the control given by Equation 2.10 are in $\mathcal{R}_T$. Since $\mathcal{R}_T$ is convex, all points in the convex hull of such trajectories are also in $\mathcal{R}_T$. Hence, $\mathcal{R}_T$ can be approximated by taking the convex hull of the union of the trajectories of the reverse tim system under the control given by Equation 2.10 for a variety of directions h. An algorithm for approximating the controllable set can now be written as follows:

Step 1: *Choose a suitably large final time T.*

Step 2: *Choose k directions h_i, $i = 1, 2, \ldots, k$.*

Step 3: *Let $i = 1$.*

Step 4: *Find $p^B(t), t \in [0, T]$ by numerically integrating the reverse time adjoint equation (Equation 2.8) backwards from time T subject to the final condition $p^B(T) = h_i$.*

Step 5: *Find the control $u^B(t), t \in [0, T]$ using Equation 2.7.*

Step 6: *Find the i^{th} trajectory, $\left[x^B(t), \quad t \in [0, T]\right]_i$, by numerically integrating the reverse time system (Equation 2.3) from $t = 0$ to $t = T$ subject to the initial condition $x^B(0) = 0$.*

Step 7: *Increment i and go to step 4 until $i > k$.*

Step 8: *Let*

$$\widehat{\mathcal{R}_T} = \triangle\left(\left[x^B(t), \quad t \in [0, T]\right]_i, \quad i = 1, 2, \ldots, k\right),$$

where $\triangle(\cdot)$ represents the convex hull.

2.3 Examples

2.3.1 Example 1: 2-D Case With Real Eigenvalues

Consider the system

$$\dot{x}(t) = \begin{bmatrix} 4 & 5 \\ 3 & -6 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 2 \end{bmatrix} u(t) \tag{2.11}$$

subject to the constraint $|u| \leq 1$. The eigenvalues of the matrix A are $\lambda_1 = -2$ and $\lambda_2 = 3$.

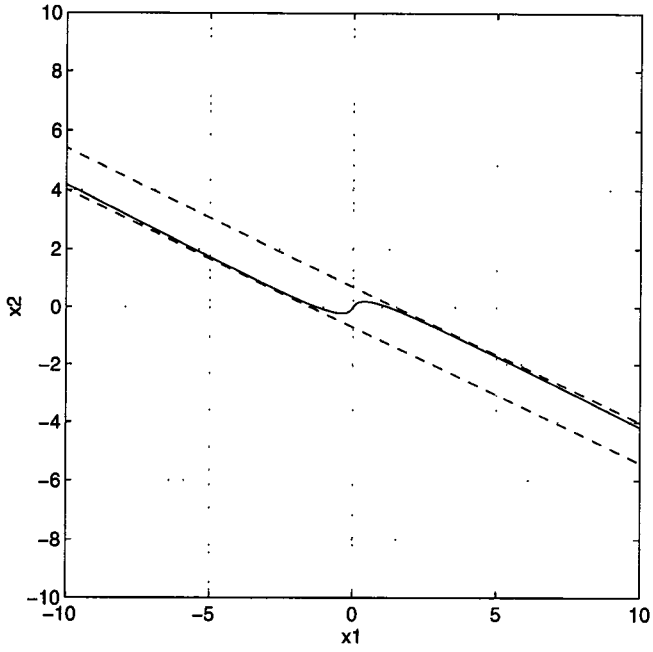

Figure 2.2: Approximation of Controllable Set for Example 1

An approximation of the controllable set of this system can be found by implementing the approximation algorithm using MATLAB. Figure 2.2 shows this approximation. The solid lines represent the trajectories of the system under optimal inputs while the dotted lines represent the boundary of the convex hull of these trajectories, or the boundary of $\widehat{\mathcal{R}_T}$.

For this example, it is possible to solve for controllable set exactly. The **PVA** function in MATRIXx can be used to find the (2×2) matrix M such that the matrix MAM^{-1} is the Real Jordan form of A [16].

$$M = \begin{bmatrix} 0.8944 & -0.3162 \\ -0.4472 & 0.9487 \end{bmatrix}$$

Employing the coordinate transformation $z = M^{-1}x$ yields the new system

$$\dot{z}(t) = \begin{bmatrix} -2 & 0 \\ 0 & 3 \end{bmatrix} z(t) + \begin{bmatrix} 2.2361 \\ 3.1623 \end{bmatrix} u(t). \tag{2.12}$$

These are two uncoupled first order ODE's. The first ODE,

$$\dot{z}_1(t) = -2z_1(t) + 2.2361u(t) \tag{2.13}$$

is exponentially stable while the second ODE,

$$\dot{z}_2(t) = 3z_2(t) + 3.1623u(t) \tag{2.14}$$

is unstable. If z_2 can be driven to 0 in time T, then z_1 can be allowed to drift to the origin by using the control $u(t) = 0$ for all $t > T$. Hence, the total state is controllable if and only if z_2 can be driven to the origin. Clearly, this is true if and only if $|3z_2(0)| < 3.1623$, or $|z_2(0)| < 1.054$. Transferring back to x coordinates, this inequality becomes

$$|-0.4472x_1(0) + 0.9487x_2(0)| < 1.054$$

The plot of the exact controllable set is contained in Figure 2.3. As should be expected, the approximation $\widehat{\mathcal{C}}$ is smaller than the exact controllable set.

2.3.2 Example 2: 2-D Case With Complex Eigenvalue

Consider the system

$$\dot{x}(t) = \begin{bmatrix} 1 & -9 \\ 6 & 2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{2.15}$$

subject to the constraint $|u| \leq 1$. The matrix A has eigenvalues at $1.5 \pm j7.331$. The reachable set for this system was approximated using MATLAB and is plotted in Figure 2.4. The figure shows one trajectory which spirals outward from

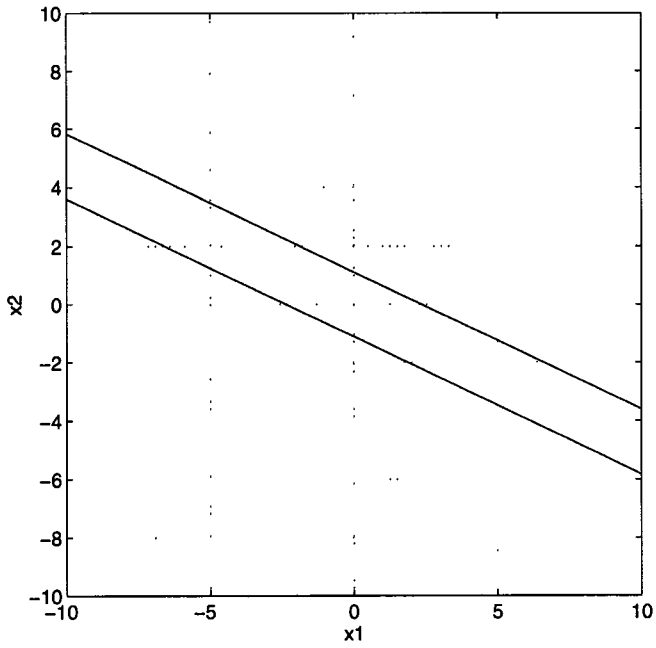

Figure 2.3: Exact Controllable Set for Example 1

the origin and converges to a periodic orbit which circles the origin. The trajectory converges to the same periodic orbit regardless of the choice of h. Hence, this periodic orbit is the boundary of the approximation of the controllable set.

The careful study of the reverse time system corresponding to Equation 2.15 yields an interesting observation. The control $u(t) = 1$ for all $t > 0$ does not drive the system to the boundary of its reachable set. Instead, the system converges exponentially to a new equilibrium point well inside the reachable set. Figure 2.5 shows this. The solid line in Figure 2.5 is the trajectory of the reverse time system starting at $x(0) = 0$ with the constant control $u(t) = 1$ for all t. The dashed line marks the boundary of the reachable set. In order to approach the boundary of the reachable set, $u(t)$ must switch back and forth between $+1$ and -1 at a frequency that is approximately equal to the natural frequency of the

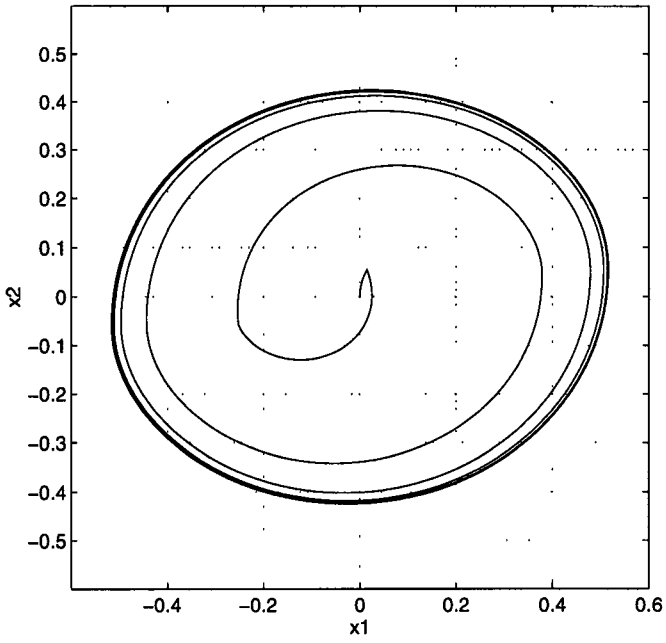

Figure 2.4: Approximate Controllable Set for Example 2

system. If the reverse time system was a physical system, the u which drives the state to the boundary could be thought of as the u which takes the most advantage of the transfer of energy in the system.

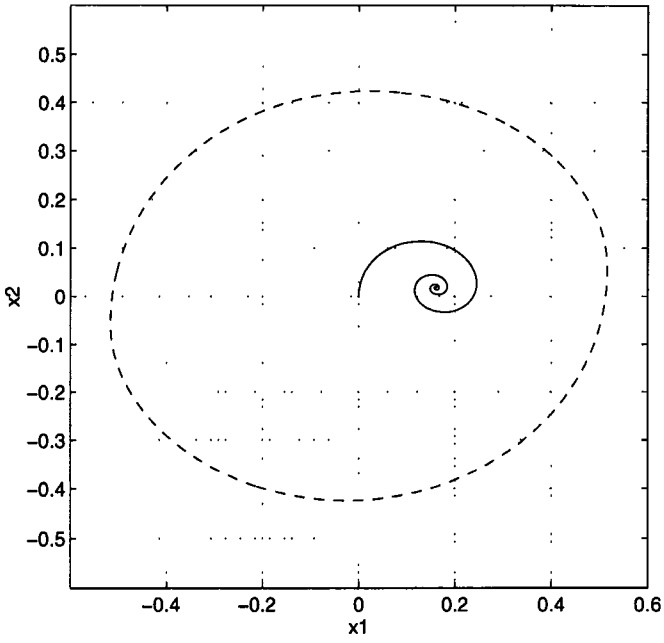

Figure 2.5: Trajectory of Reverse Time System (Example 2) with Constant Input

Chapter 3

Saturating Control Laws

This chapter presents some controllers for the constrained input problem. The controllers to be presented fall into two distinct groups. The first group, classification controllers, is discussed in Section 3.1. The classification controllers use a "bang-bang" control law to drive the state of the system to the origin from any initial condition contained in the controllable set. The second group of controllers, modified linear controllers, is discussed in Section 3.2. These controllers start with the assumption that there exists a linear feedback law which stabilizes the system while satisfying the input constraints for the set of states in which the plant is expected to operate. Then, as the name suggests, this linear feedback law is modified to improve performance of the closed loop system.

3.1 Classification Controllers

Recall from Section 2.2 that for time $T > 0, h \in R^n$, the control

$$u_j^B(t,h) = k_j sgn(-h^T \Phi_{-A}(T,t) b_j), \qquad j = 1, 2, \ldots, m. \tag{3.1}$$

drives trajectories of the reverse time system

$$\dot{x}^B(t) = -(Ax^B(t) + Bu^B(t)), \quad x^B(0) = 0 \tag{3.2}$$

to $x^B(T)$, which happens to be on the boundary of the controllable set at time T. The control $u : [0, T] \rightarrow R^m$ given by

$$u(t) = u^B(T - t), \quad t \in [0, T] \tag{3.3}$$

drives the forward time system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.4}$$

from the initial condition $x(0) = x^B(T)$ to the origin in time T. Hence, given any $x_0 \in \mathcal{C}$ some iterative method can be used to find values for h and T so that $x^B(T)$ will equal x_0, and the resulting control $u : [0, T] \rightarrow R^m$ given by Equations 3.1 and 3.3 will drive the forward time system from $x(0) = x_0$ to the origin in time T. This provides a method of finding an open loop control to drive any $x_0 \in \mathcal{C}$ to the origin.

A naive way to derive a discrete closed loop control law for a the system would be to compute the open loop control u at each step using the above method. This technique would work, but the extremely large number of computations necessary at each step would prohibit its use for most applications.

A more practical approach comes from formulating the problem as a classification problem. At any given time, the input given by Equations 3.1 and 3.3 will take one of a finite number of possible values. Specifically, for a system with m inputs, there are at most 2^m different values which the input vector can take. Hence each state can be assigned one of a finite number of classes according to the input vector which should be used to drive the state to the origin.

In general, complicated switching surfaces separate the states belonging the different classes. It is usually difficult to represent these surfaces in closed form, but their effect can be approximated. To do this, a large number of states in $\mathcal{C}$ are selected. The open loop control associated with each state is then computed and each state is classified according to the input vector which should be used at that state. This creates a data set consisting of states in $\mathcal{C}$ and their corresponding classes. The controller must then be able to classify arbitrary states in $\mathcal{C}$ based on the information contained in the data set. Three methods of classification are described below.

3.1.1 Nearest Neighbor

The concept of nearest neighbor classification is straightforward: the class of a given state is assigned to be the class of the nearest state contained in the data set. Implementation of nearest neighbor classification is not quite so easy. Since the data set is generally large, searching through it for the nearest neighbor can take a long time.

One way to alleviate this problem is to use an efficient storage and retrieval system such as the k-d tree [11]. The k-d tree is a generalization of the binary decision tree often used for scalar data searches. If the data set has N elements, the time necessary to search the list linearly is proportional to N. The time necessary to search the list using the k-d tree algorithm is proportional to $\log_2(N)$. For example, if the data set contains 10,000 elements, a k-d tree search runs about 750 times faster than a linear search.

Nearest neighbor classification provides the most accurate method of classification available. It's accuracy is limited only by the resolution of the data set

and the time allowed for the search.

3.1.2 Look-up Table

Preprocessing the data set to put it in the format of a look-up table can eliminate the searching time all together. First, a grid is created over $\mathcal{C}$. Then, each point on the grid is asigned a class using its nearest neighbor. The grid is then stored in an n-dimensional array or look-up table. Now, classifying an unknown state can be done by rounding the state to the nearest point on the grid and looking up its class in the array or look-up table.

The look-up table method of classification runs much faster than the nearest neighbor approach, but this gain comes at the cost of accuracy. Also, while this method works well for low dimensional systems, higher dimensional systems require prohibitively large amounts of memory.

3.1.3 Neural Network

Neural networks have been shown to work well as classification controllers. In fact, L.G. Kraft and D. Dietz have succesfully applied a CMAC neural network to solve the time optimal control problem [18], a problem which is actually a specific case of the constrained input problem. The CMAC neural networks are easy to train, but here backpropagation neural networks are used due to the wide availability of information and software on the subject [7], [15] .

A backpropagation neural network for use as a classifying controller for a system with n states and m inputs must have n inputs and m outputs. Choosing the number of hidden layers and the number of neurons per layer is a subject of current research [5]. Here these numbers are found by trial and error. The same

holds for the transfer function at each layer, however good results can usually be obtained by using a sigmoid function for elements of the hidden layers and a linear function for the output layer.

The network is then trained on the data set. The states in the data set are presented as inputs and the m-vector associated with the class is presented as the target. Once the network is satisfactorily trained, a second data set should be created to test the accuracy of the neteork for data different from the training data.

When the network is used as a controller, each input to the plant should always be on one of its constraints. Hence, if s_i is the ith output of the neural network, then u_i is assigned as follows:

$$u_i = \begin{cases} -k_i & \text{if} \quad s_i < 0 \\ k_i & \text{if} \quad s_i \geq 0 \end{cases} \tag{3.5}$$

The use of a neural network to solve the constrained input problem has several advantages over the nearest neighbor approach. First, the neural network is trained with the data set off line, so the controller does not need access to the data set while running. Hence the problem of memory space does not present itself. Also, since the neural network finds it's answer by making a relatively small number of simple calculations, it runs much faster than nearest neighbor, which has to search through the entire data set to obtain each answer.

3.1.4 An Example

Consider the system

$$\dot{x}(t) = \begin{bmatrix} 3 & 4 \\ -9 & 2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{3.6}$$

subject to the constraint $|u| \leq 1$. A data set was compiled as described above. The system was then simulated for each of the 3 described classification controllers.

Nearest Neighbor Controller

The system was simulated on a Sparc 20 using the method of nearest neighbor to determine the stabilizing input in a closed loop manner. The nearest neighbor algorithm was implemented with the help of the k-d tree. Code for the storage and retrieval of data from the k-d tree was graciously provided by J. Schneider [30]. The results are plotted in Figure 3.1. Note that the switching curve plotted in Figure 3.1(a) is the most accurate possible approximation of the actual switching curve from the given data set.

Look-up Table Controller

To implement a look-up table controller, a 100 by 100 grid with x_1 and x_2 evenly distributed between -0.3 and 0.3 was created. Each point on the grid was then classified using nearest neighbor implemented with a k-d tree. The resulting classes are then stored in a 2-dimensional array G such that

$$G\left(\frac{99(x_1-(-0.3))}{0.6}+1, \frac{99(x_2-(-0.3))}{0.6}+1\right) = \text{class}(x). \tag{3.7}$$

An x which is not on the grid can be classified by rounding it to the nearest point on the grid. Hence

$$\text{class}(x) = G\left(\text{round}\left(\frac{99(x_1-(-0.3))}{0.6}+1\right), \text{round}\left(\frac{99(x_2-(-0.3))}{0.6}+1\right)\right). \tag{3.8}$$

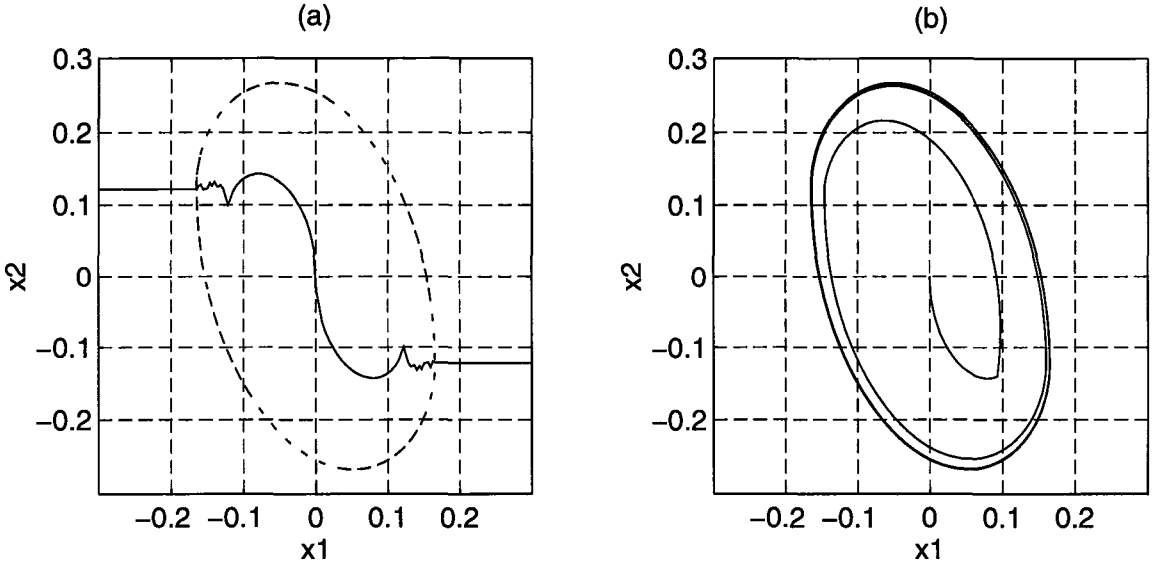

Figure 3.1: Example using nearest neighbor control. (a) switching curve. (b) trajectory of system under nearest neighbor control

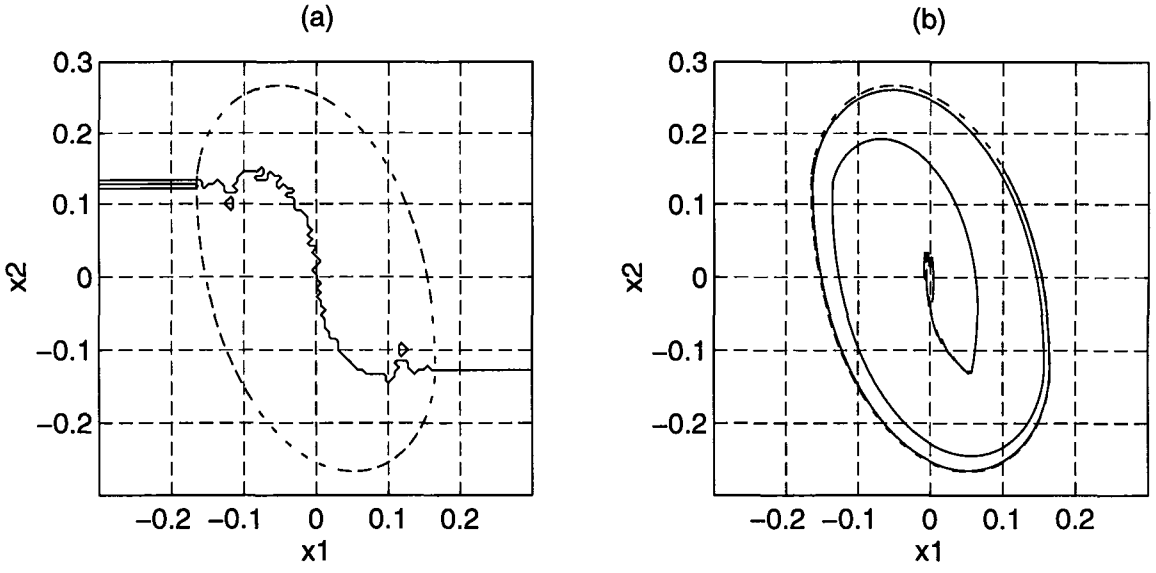

Figure 3.2: Example with data set preprocessed into a 2 dimensional matrix. (a) approximate switching curve. (b) trajectory under control

The results of the simulation are found in Figure 3.2. Note the comparison between the approximate switching curve plotted in Figure 3.2(a) and the switching curve found using nearest neighbor, Figure 3.1(a).

Neural Network Controller

A 2-input, single output backpropagation neural network was trained using the MATLAB Neural Network Toolbox [7]. The network was chosen to have 2 hidden layers with 10 neurons in each hidden layer. A logarithmic sigmoid function was used for the two hidden layers while a linear transfer function was used to the output layer. The system was simulated using the resulting neural network for feedback control. The results of the simulation are plotted in Figure 3.3.

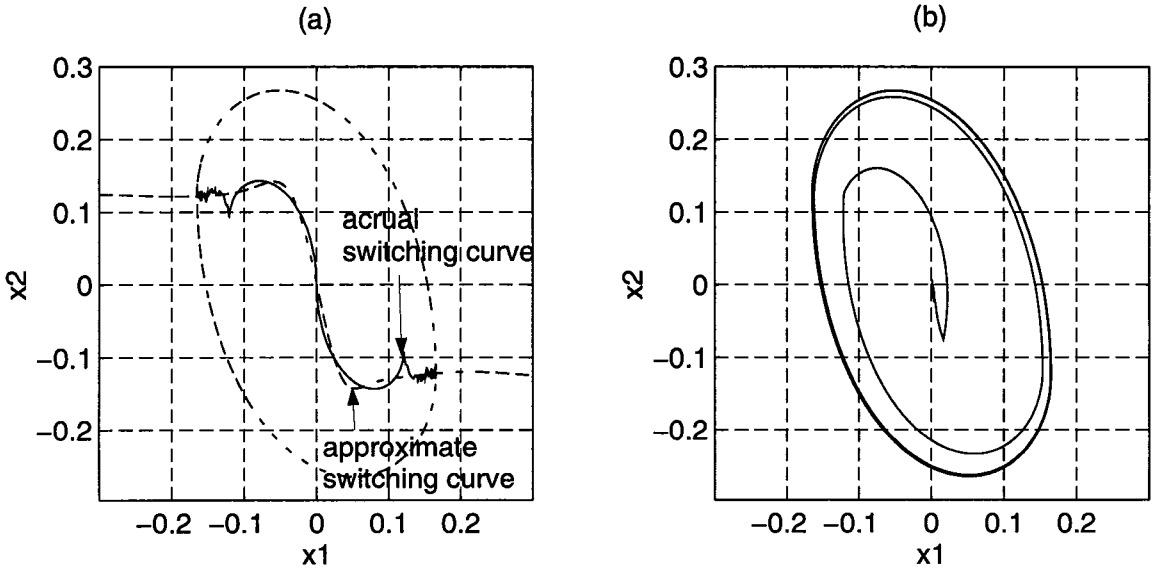

Figure 3.3: Example using a neural network for feedback control. (a)actual and approximated switching curves. (b) trajectory under control.

3.2 Modified Linear Controllers

As the name implies, modified linear controllers start with a linear state feedback which is then modified to improve its performance. These controllers will not, in general, stabilize the entire controllable set. It is assumed that the set of possible initial conditions taken on by the plant is some subset of the controllable set. Further, it is also assumed that there exists a linear state feedback law which stabilizes the plant while satisfying the input constraint for all possible initial conditions. If these conditions are not met then a modified linear controller cannot be used.

3.2.1 Preliminaries

First, some relevant sets are defined.

Definition 3.1 *Define D to be the set containing all possible initial conditions which can be taken by the plant.*

Definition 3.2 *Let L be an $m \times n$ matrix of constant real numbers. Define $E(L)$ to be the set of all states such that the m-vector $-Lx$ satisfies the input constraints given by Equation 2.2. In other words, if l_i represents the ith row of L,*

$$E(L) \triangleq \left\{ x \in R^n \middle| \, |l_i x| < k_i \right\}.$$

Finally, notions of two different saturation functions are formalized.

Definition 3.3 *Let v be an m-vector. Define the sat function to be*

$$sat(v) \triangleq \begin{bmatrix} sat(v_1) \\ sat(v_2) \\ \vdots \\ sat(v_m) \end{bmatrix} \tag{3.9}$$

where

$$sat(v_i) \triangleq \begin{cases} -k_i & if \quad v_i \leq -k_i \\ v_i & if \quad -k_i < v_i < k_i \\ k_i & if \quad v_i \geq k_i \end{cases} \tag{3.10}$$

Definition 3.4 *Let v be an m-vector. Define the msat function to be*

$$msat(v) \triangleq \alpha v \tag{3.11}$$

where

$$\alpha = min\left\{\frac{k_1}{|v_1|}, \frac{k_2}{|v_2|}, \ldots, \frac{k_m}{|v_m|}, 1\right\}. \tag{3.12}$$

The *sat* function simply clips the elements of v that are out of range. The *msat* function reduces all elements of v in proportion so that all of the constraints are met, hence preserving the direction of control [14].

3.2.2 Gutman-Hagander

P. Gutman and P. Hagander have proposed a modified linear controller which uses two linear feedback laws which are added together and then saturated using the *sat* function [13]. First, an $m \times n$ feedback matrix L is computed so that the feedback law $u = -BLx$ stabilizes the system while satisfying the input constraints for all initial conditions $x_0 \in D$. One way of doing this is to solve the Linear Quadratic Optimal control problem, increasing the penalty on the

control until the constraints are met. Next, an $n \times n$ symmetric positive definite matrix P is chosen such that

$$P(A - BL) + (A - BL)^T P < 0. \tag{3.13}$$

In order for P to be a valid choice, there must exist $c > 0$ such that

$$\sup_{x \in D} x^T P x \leq c \leq \min_{i=1,2,\ldots,m} \left\{ \frac{k_i^2}{l_i P^{-1} l_i^T} \right\}. \tag{3.14}$$

An $m \times m$ diagonal matrix M is chosen such that all of its elements are greater than or equal to 0. Finally, the feedback control law becomes

$$u = sat((-BL - MB^T P)x). \tag{3.15}$$

The resulting closed loop system is guaranteed to be stable for all initial conditions $x(0) \in D$. The performance of the closed loop system can be tuned by adjusting the elements of M during simulations.

3.2.3 Control Using the *msat* Function

The *msat* function proves to be very useful for the control of stable linear systems with constrained inputs [14]. The basis of this usefulness is stated in the following fact:

Proposition 1 *Consider an assymptotically stable linear time invariant system of the form given in Equation 2.1. Let L be a constant $m \times n$ feedback matrix. Assume that there exists a symmetric, positive definite $(n \times n)$ matrix P such that*

$$PA + A^T P < 0 \tag{3.16}$$

and

$$P(A - BL) + (A - BL)^T P < 0. \tag{3.17}$$

Then the closed loop system with msat saturated feedback

$$\dot{x} = Ax - B\ msat(Lx) \tag{3.18}$$

is asymptotically stable.

Proof

Equations 3.16 and 3.17 imply that

$$P(A - \alpha BL) + (A - \alpha BL)^T P < 0 \tag{3.19}$$

for all $\alpha \in [0, 1]$. Let $E(L)$ be the set of all $x \in R^n$ such that the m-vector $-Lx$ satisfies the input constraint. Let

$$V(x) = x^T P x \tag{3.20}$$

be a Lyapunov function candidate for the nonlinear system

$$\begin{aligned} \dot{x} &= Ax - B\ msat(Lx) \\ &= Ax - \alpha(x)BLx \end{aligned} \tag{3.21}$$

where $\alpha(x) \in [0, 1]$ for all $x \in R^n$. Then

$$\dot{V}(x) = \dot{x}^T P x + x^T P \dot{x} \tag{3.22}$$

$$= x^T (P(A - \alpha(x)BL) + (A - \alpha(x)BL)^T P)x \tag{3.23}$$

$$< 0 \quad \text{for all } x \in R^n. \tag{3.24}$$

□

Hence, L can be chosen such that the closed loop system meets design specifications by using any of the wide variety of available linear controller design tools (LQR, pole placement, etc.). Then, if there exists a P which satisfies

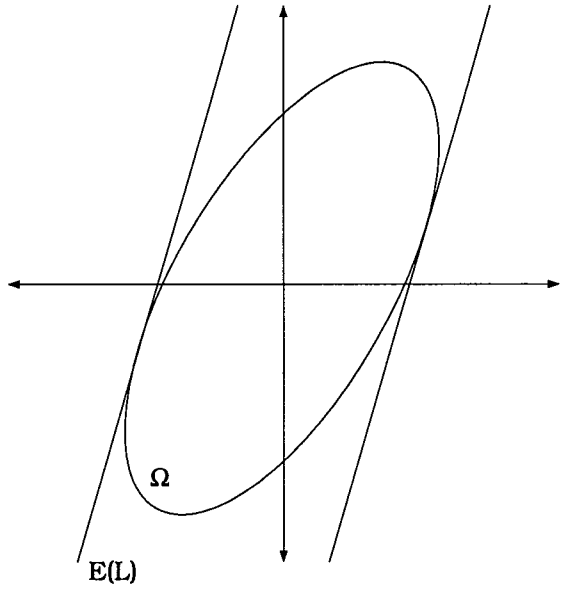

Figure 3.4: Important sets for the *msat* controller

Equations 3.16 and 3.17, the closed loop system resulting from Equation 3.18 will perform according to specifications close to the origin and is guaranteed to be asymtotically stable everywhere in R^n.

Figure 3.4 provides a graphical description of the important sets involved in the *msat* controller. $E(L)$ represents the set of states for which the m-vector $-Lx$ satisfies the input constraint given by Equation 2.2. Ω represents the largest subset of $E(M)$ which is positively invariant under the system

$$\dot{x} = (A - BL)x. \tag{3.25}$$

If the initial condition of the closed loop system is in Ω, then the system performs to the specifications selected when choosing M. If the initial condition is outside Ω then the closed loop system is asymtotically stable.

3.2.4 Dual Mode Controllers

Dual mode controllers use a"high gain" linear state feedback law close to the origin and use a "low gain" linear state feedback far away from the origin [13]. The closed loop system then performs according to desired performance criterion close to the origin, where the high gain controller meets the input constraints. The low gain feed back is then chosen to satisfy less stringent performance criterion, such as stability, for all initial conditions in D. The analysis of the *msat* function presented by P.J. Gyugyi and G. Franklin [14] provides motivation for the following sufficient condition for dual mode controller stability:

Proposition 2 *Consider the system given by Equation 2.1 subject to the input constraints given by Equation 2.2. Let L an M both be $m \times n$ state feedback matrices such that both systems*

$$\dot{x} = (A - BL)x \tag{3.26}$$

and

$$\dot{x} = (A - BM)x \tag{3.27}$$

are asymptotically stable. Let $E(M)$ represent the set of states for which the m-vector $-Mx$ satisfies the input constraint and let Ω_1 represent the largest subset of $E(M)$ which is positively invariant under the system give by Equation 3.27. Likewise, let $E(L)$ represent the set of states for which the m-vector $-Lx$ satisfies the input constraint and let Ω_2 represent the largest subset of $E(L)$ which is positively invariant under the system. WLOG, let $\Omega_1 \subset \Omega_2$ (See Figure 3.5). Further, assume that there exists a symetric, positive definite $(n \times n)$ matrix P such that

$$P(A - BL) + (A - BL)^T P < 0 \tag{3.28}$$

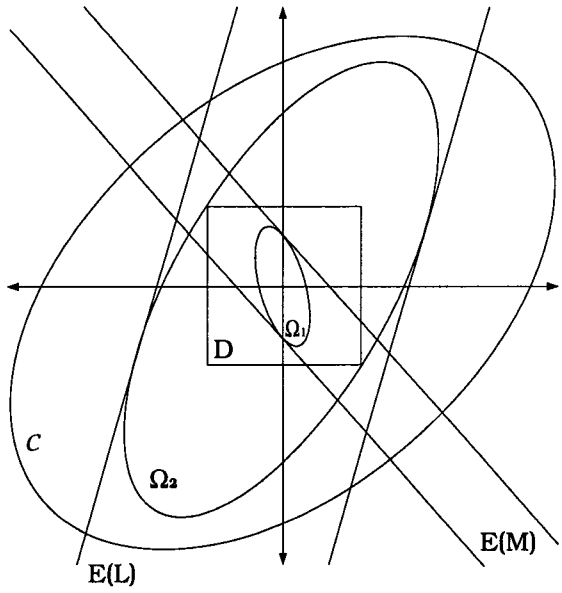

Figure 3.5: Important sets for the dual mode controller

$$\textit{and}$$

$$P(A - BM) + (A - BM)^T P \quad < \quad 0. \tag{3.29}$$

Then the control law

$$u = \begin{cases} -Mx & \textit{if } x \in \Omega_1 \\ -Lx & \textit{else} \end{cases} \tag{3.30}$$

will cause the resulting closed loop system to be asymtotically stable for all initial conditions $x_0 \in \Omega_2$.

The proof is very similar to the proof of Proposition 1. Conditions 3.28 and 3.29 provide the Lyapunov function $V(x) = x^T P x$ with $\dot{V}(x) < 0$ for all x in the invariant set Ω_2.

Proposition 2 provides a straightforward approach to dual mode controller design. First, choose L to so that the control law $u = -Lx$ stabilizes the

system while satisfying the input constraint for all initial conditions $x_0 \in D$. Then choose M so that the control law $u = -Mx$ causes the resulting closed loop system to perform according to the design specifications. Finally check to see if Conditions 3.28 and 3.29 are satisfied. If so, then the control law given by Equation 3.30 will cause the resulting closed loop system to meet design specifications for all initial condition $x_0 \in \Omega_1$ and stable for all $x_0 \in D$.

The design process could be made considerably easier if there were some conditions on $A, B, L,$ and M that would guarantee that Conditions 3.28 and 3.29 hold. This is left for future work.

3.3 Conclusion

Several controllers for use with LTI systems subject to input constraints are presented here. These controllers can be broken into two distinct groups: classifying controllers and modified linear controllers.

The classifying controllers transform the open loop stabilizing control found in Chapter 2 into a closed loop, "bang-bang" control law. The three classifying controllers presented were nearest neighbor, look-up table, and neural network. The nearest neighbor controller provides the most accurate method implementing the switching, however it is also the slowest of the three. The look-up table method is very fast, but it requires large amounts of memory, especially for high dimensional systems. The neural network controller is not as fast as a look-up table, but it is much faster than the nearest neighbor controller. Additionally, the neural network requires very little memory, making it easier to implement than a look-up table.

Currently, the Gutman-Hagander controller is the most useful of the modified linear controllers. The Gutman-Hagander design algorithm can be applied to any LTI system is and easily implemented.

Chapter 4

Rapid Prototyping Using the AC-100

To facilitate the implementation of the experiments that follow in the thesis, a rapid prototyping station centered around Integrated Systems Inc.'s AC-100 system was constructed. The development of this rapid prototyping station, or test bed, deserves some mention for a couple reasons. Firstly the author concentrated a large portion of his efforts towards building the test bed and learning how to use it. More importantly, the test bed was designed with no single control application in mind. As a result, the system allows the rapid development and implementation of control strategies for any given system. It will undoubtably prove to be a valuable asset to the research efforts of the Intelligent Servosystems Laboratory (ISL).

This chapter briefly describes the construction and use of the system. It may be used as a starting point for someone wishing to use the system for his own experiments.

4.1 The AC-100

Integrated Systems Inc.'s AC-100 is the heart of this rapid prototyping station. Software tools contained in AC-100 provide the user with the ability to design and simulate controllers, select input and output devices, generate C code to run the controller, download the controller to a digital signal processing (DSP) chip, and finally execute the controller in real time.

Most aspects of the AC-100 run on a SPARCstation IPX. As of this writing, the AC-100 software is only licensed to run on the SPARC known as **tulip** in the ISL. The controller is designed, I/O parameters are specified, and C code is generated on the SPARC. The resulting code is then downloaded via FTP to a Dell 486 PC (**rhea-pc**) which houses the DSP chip as well as the various I/O devices and their drivers. The controller is compiled and executed on the PC, but the user sends commands to the controller from the SPARC via the FTP link. A graphical representation of the AC-100 system is given in Figure 4.1.

The various aspects of the AC100 are described below.

4.1.1 MatrixX and SystemBuild

MatrixX is a powerful matrix manipulation program. It contains a wide variety of matrix functions designed specifically for use by control engineers. When used as part of the rapid prototyping station, MatrixX can be used to implement a desired control algorithm (e.g. LQR) and produce a controller for a given plant model.

SystemBuild, a component of MatrixX, is an interactive block diagram editor which can be used to graphically construct and simulate complex dynamical

AC100 Description

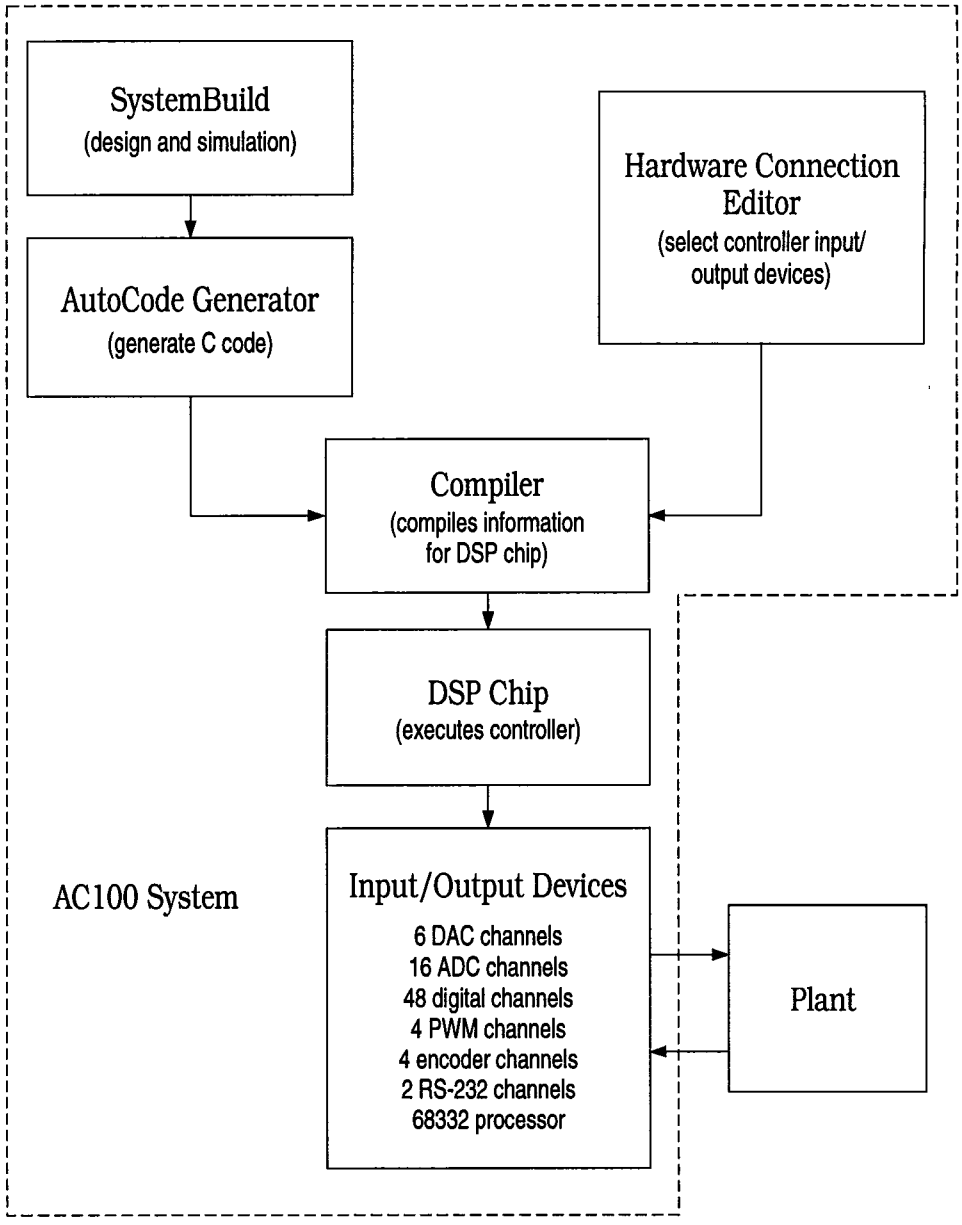

Figure 4.1: AC-100 block diagram

systems. It has the capability to numerically analyze continuous time dynamics, discrete time dynamics, and finite state machine descriptions as well as systems containing any combination of the three. For use with the AC-100, SystemBuild is used to construct a controller. The controller will, in general, have external inputs and outputs, all of which can be given mnemonic labels by the user. These inputs and outputs represent the plant's sensor signals and control inputs as well as controller parameters and display outputs. Once a controller which works satisfactorily in simulation is designed, SystemBuild is used to generate a real time file, or *.rtf* file which contains all of the information necessary to proceed with the prototyping procedure.

4.1.2 Interactive Animation Builder

The Interactive Animation Builder (IAB) allows the user to construct a graphical instrumentation panel which can be used to send commands to and monitor the outputs of the controller running on the PC. Numerical inputs can be sent to the controller via sliders and dials and logical inputs are sent using various types of switches. Controller information can be displayed using strip charts, dials, and various other graphical displays.

To use the IAB with the AC-100, the user constructs the desired instrument panel configuration, then uses the information contained in the *.rtf* file to connect the inputs and outputs of the elements on the instrument panel to the appropriate controller inputs and outputs, which still have the mnemonic labels given in SystemBuild.

4.1.3 Hardware Connection Editor

As its name implies, the Hardware Connection Editor (HCE) is used to connect the controller inputs and outputs to the appropriate I/O devices. Like the IAB, the HCE uses the information contained in the *.rtf* file so that the controller inputs and outputs still have the mnemonic labels given in SystemBuild.

The ISL's version of AC-100 has the following I/O capabilities:

- *6 channels digital to analog converter*
- *16 channels analog to digital converter*
- *72 channels of digital I/O (programmable)*
- *2 serial channels (RS-232-C or RS-422)*
- *4 channels quadrature position encoder*
- *4 channels pulse width modulation*
- *Motorola 68332 processor*

4.1.4 Compiling the Controller

The AutoCode program generates C code from the controller's *.rtf* file at the touch of a button. After that, AC-100 provides tools that automatically download the C code to the PC, link it with the appropriate device drivers, compile the information, and load it onto the DSP chip. Once this is done, the controller can be started, monitored, and manipulated from the SPARC using the instrumentation panel created with the IAB.

4.2 Physical Description

To facilitate use of the AC-100, a special cart has been designed and constructed. The PC sits on top of the cart. The various I/O devices on the PC have been connected to a number of easily accessible and appropriately chosen connectors mounted to a large panel on the front of the cart. Two power supplies and three general purpose amplifiers also reside on the cart. Hence, it is equipped to serve as a controller for a wide variety of plants with little or no additional equipment. The cart has been equipped with a long ethernet cable so it is easily transported to anywhere in the ISL. A photograph of the cart is shown in Figure 4.2.

4.3 Conclusion

The AC-100, facilitated by the cart on which it is mounted, allows for rapid design and implementation of control strategies for a wide variety of potential plants. This rapid prototyping capability will greatly reduce the amount of time and effort required to translate theory into practice, furthering the research capabilities of the ISL.

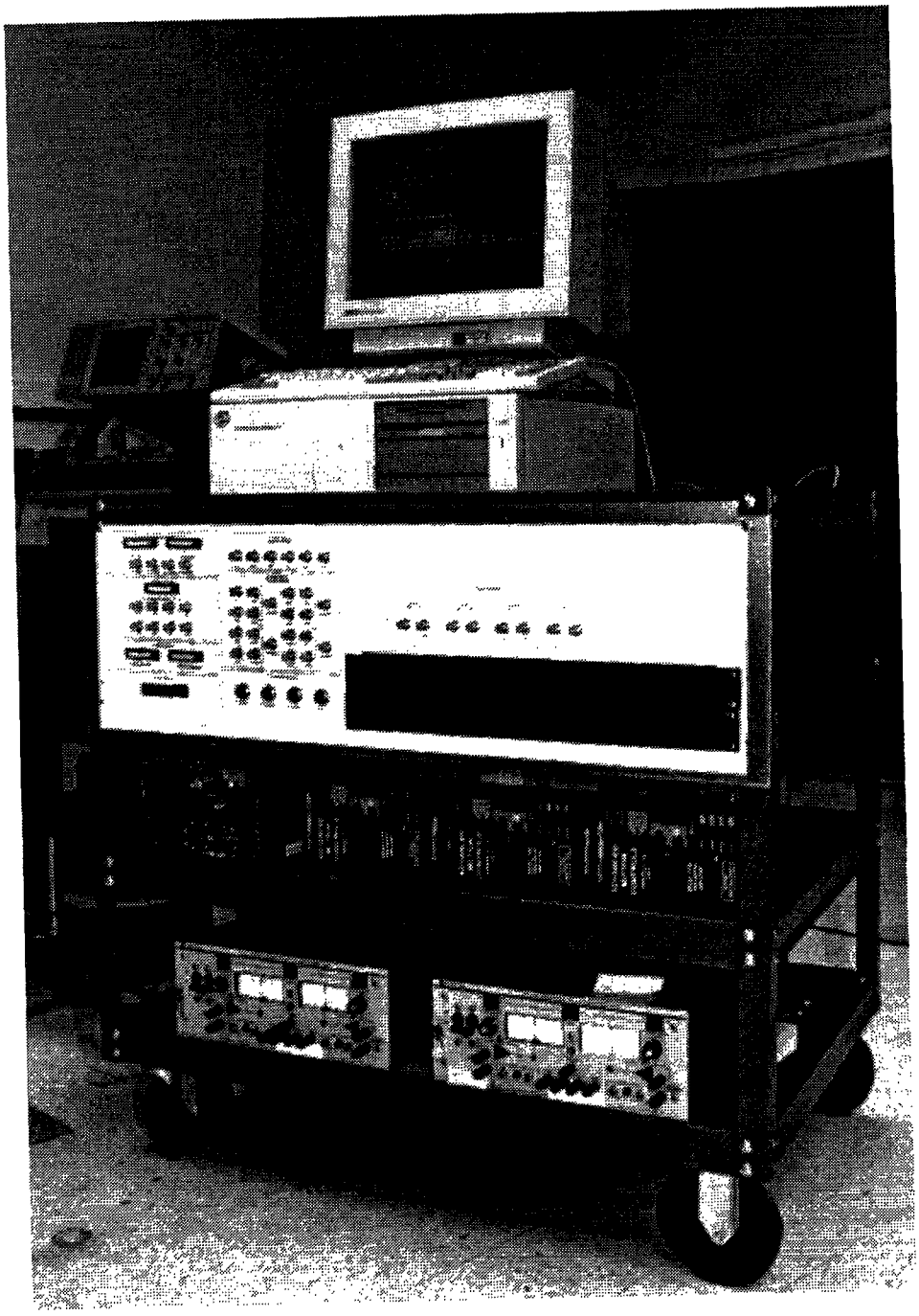

Figure 4.2: The AC-100 cart

Chapter 5

System Identification

Most modern controller design techniques, including those discussed in the previous chapters, require an accurate model of the system to be controlled. Models derived from physical principles can provide a great deal of valuable insight into a given system, but many times these models contain parameters which are difficult if not impossible to measure accurately. Hence a more experimental approach is required. We would like to obtain the model of an unknown plant given a known input and the corresponding measured output. This is the goal of system identification. This chapter presents two methods of experimentally determining the transfer function of a stable linear time invariant system. Recursive least squares (RLS), the first method discussed, is an on-line method of estimating a real rational transfer function. The second method discussed is matching pursuits (MP), a wavelet based real rational transfer function approximation algorithm. Both methods are implemented using the AC-100.

5.1 Recursive Least Squares

Recursive least squares estimation (RLS) is an on-line method that can be used to generate a mathematical model for any stable single input, single output linear time invariant plant. As the name suggests, the model error will be minimized in the sense of least squares. If $y(t)$ represents the plant output at time t and $\widehat{y}(t)$ represents the model output at time t, then the quantity

$$E(t) = \int_0^t (y(\sigma) - \widehat{y}(\sigma))^2 d\sigma \tag{5.1}$$

will be minimized for all $t > 0$.

To implement RLS, an initial guess of a model for the plant is made. The output of the model is then calculated based on a known input and compared to the actual plant output to determine the error. The parameters of the model are then adjusted according to error. The process is repeated until the model output suitably matches the plant output. A more detailed description of the above process follows.

5.1.1 The Model Structure

Since the RLS algorithm is usually implemented using a computer, it is natural to model the plant as a discrete time system using a z-transform. The z-transform of the output of the model can be written as

$$Y(z) = T(z)U(z) \tag{5.2}$$

where

$$T(z) = \frac{b_1 z^{n-1} + b_2 z^{n-2} + \ldots + b_{n-1} z + b_n}{z^n + a_1 z^{n-1} + \ldots + a_{n-1} z + a_n} \tag{5.3}$$

and $U(z)$ is the z-transform of the model input. The model parameters $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ are real and the integer n is the order of the model.

To choose an initial guess for the model, one must first determine the value of n. If the physical properties of the plant are well understood, there is usually an obvious choice for n. In some cases, however, the designer is forced to make a "reasonable" guess at n which is then adjusted by trial and error over several repetitions of the entire RLS process. Once n is chosen, then the initial guess for the model is obtained by setting the parameters $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ to zero.

5.1.2 Calculating the Error

The first step in calculating the model error is to calculate the model output. Recall that for the system described by Equations 5.2 and 5.3, the current value of the system output is given by

$$\begin{aligned} y(k) &= -(a_1 y(k-1) + a_2 y(k-2) + \ldots + a_n y(k-n)) && (5.4) \\ &\quad + b_1 u(k-1) + b_2 u(k-2) + \ldots + b_n u(k-n) && (5.5) \end{aligned}$$

where a_i and b_i for $i = 1, 2, \ldots, n$ are the unknown system parameters. Hence, based on the known plant input and the measured plant output, the model output is given by

$$\begin{aligned} \widehat{y}(k) &= -(\widehat{a}_1 y(k-1) + \widehat{a}_2 y(k-2) + \ldots + \widehat{a}_n y(k-n)) && (5.6) \\ &\quad + \widehat{b}_1 u(k-1) + \widehat{b}_2 u(k-2) + \ldots + \widehat{b}_n u(k-n) && (5.7) \end{aligned}$$

where $\widehat{a}_i$ and $\widehat{b}_i$ for $i = 1, 2, \ldots, n$ are the estimated model parameters. The model error is then

$$e(k) = y(k) - \widehat{y}(k). \qquad (5.8)$$

5.1.3 Updating the Model Parameters

There are many algorithms available to update the model parameters so that they will converge to the solution of the least squares problem [1] [20]. Here the projection algorithm presented by K.J. Astrom and B. Wittenmark [1] is used due to the simplicity of the required calculations.

First, some notation is necessary. Define

$$\widehat{\theta}(k) = \begin{bmatrix} \widehat{a}_1(k) \\ \vdots \\ \widehat{a}_n(k) \\ \widehat{b}_1(k) \\ \vdots \\ \widehat{b}_n(k) \end{bmatrix} \tag{5.9}$$

and define

$$\varphi(k) = \begin{bmatrix} -y(k-1) \\ \vdots \\ -y(k-n) \\ u(k-1) \\ \vdots \\ u(k-n) \end{bmatrix}. \tag{5.10}$$

Note that this notation can be used to write the model output as

$$\widehat{y}(k) = \varphi^T(k)\widehat{\theta}(k). \tag{5.11}$$

Now the parameter update law given by the projection algorithm is

$$\widehat{\theta}(k) = \widehat{\theta}(k-1) + \frac{\gamma\varphi(k)}{\alpha + \varphi^T(k)\varphi(k)} e(k-1) \tag{5.12}$$

where $\alpha \geq 0$ and $0 < \gamma < 2$.

5.1.4 Convergence of Model Parameters

A natural question which arises at this point is as follows: Under what conditions will the projection algorithm cause the model parameters to converge to the solution of the least squares problem? It turns out that the model parameters will converge provided that the input signal satisfies a condition known as persistant excitation. This notion is formally developed below.

Definition 5.1 *A square summable discrete time signal u is said to be* **persistantly exciting of order n** *if*

$$\lim_{t\to\infty} \frac{1}{t} \left(\sum_{k=1}^{t} (a_0 u(k-(n-1)) + a_1 u(k-(n-2)) + \ldots + a_{n-1} u(k)) \right)^2 > 0 \tag{5.13}$$

for all $a_0, \ldots, a_{n-1}$ such that $a_i \neq 0$ for at least one $i \in \{0, \ldots, n-1\}$.

In other words, a signal u which is persistantly exciting of order n cannot be filtered to 0 by any finite impulse response (FIR) filter of order $n-1$ or lower. This leads to the following theorem, which serves as a sufficient condition for model parameter convergence.

Theorem 5.1 ([3]) *Suppose RLS with the projection algorithm is used to identify an nth order plant using the model given in equation 5.3. The model parameters will converge to the solution of the least squares problem provided that u is persistantly exciting of order $2n$.*

This theorem is made more useful with the following facts.

Fact 5.1 ([1]) *A signal u that is the sum of k sinusoids is persistently exciting of order $2k$.*

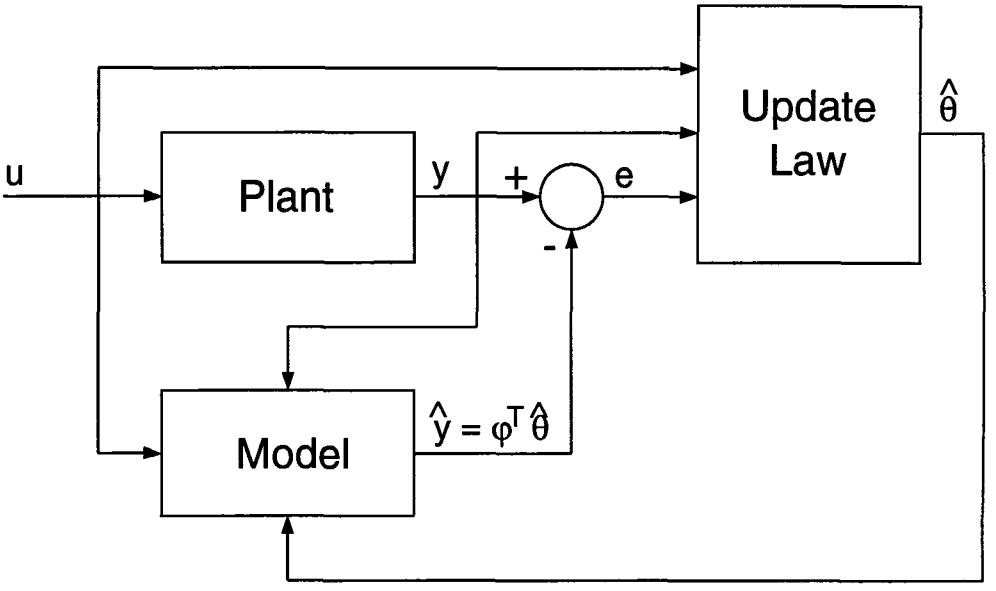

Figure 5.1: Block diagram of Recursive Least Squares estimation

Fact 5.2 ([1]) *A signal u that is an independant identically distributed (iid) process is persistantly exciting of all orders.*

In order to identify an nth order plant, an input signal which is persistantly exciting of order $2n$ is required. Either an input signal which consists of the sum of n sinusoids or an input signal consisting of an iid process will guarantee model parameter convergence.

The block diagram in Figure 5.1 provides a graphical description of the RLS process.

5.1.5 Example: Identification of a DC Motor

Here RLS with the projection algorithm is used to estimate the model of a DC motor. The set up used for this identification experiment is shown in Figure 5.2.

Using the power amp input voltage as the system input u and the tachometer

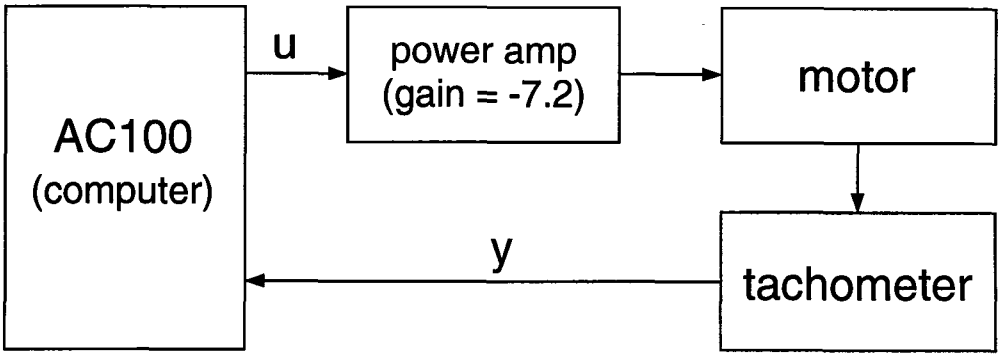

Figure 5.2: Experimental set up of DC motor for RLS identification

output voltage as the system output y, the system is well modeled by the transfer function

$$T(z) = \frac{b}{z+a} \tag{5.14}$$

for some real a and b [26]. Employing the notation of Subsection 5.1.3,

$$\widehat{\theta}(k) = \begin{bmatrix} a(k) \\ b(k) \end{bmatrix} \tag{5.15}$$

and

$$\varphi(k) = \begin{bmatrix} -y(k-1) \\ u(k-1) \end{bmatrix}. \tag{5.16}$$

Hence the update law given by Equation 5.12 can be written as

$$a(k) = a(k-1) + \frac{\gamma a(k-1)}{\alpha + a^2(k-1) + b^2(k-1)} e(k-1) \tag{5.17}$$

$$b(k) = b(k-1) + \frac{\gamma b(k-1)}{\alpha + a^2(k-1) + b^2(k-1)} e(k-1). \tag{5.18}$$

These equations are easily implemented on a computer using the AC-100. The motor system is modeled as a discrete time system sampled at 100 Hz using zero order hold. The input to the power amp $u(k)$ is a 2 Hz sine wave with an amplitude of 1 volt. It is also sampled at a rate of 100 Hz. Note that a sine

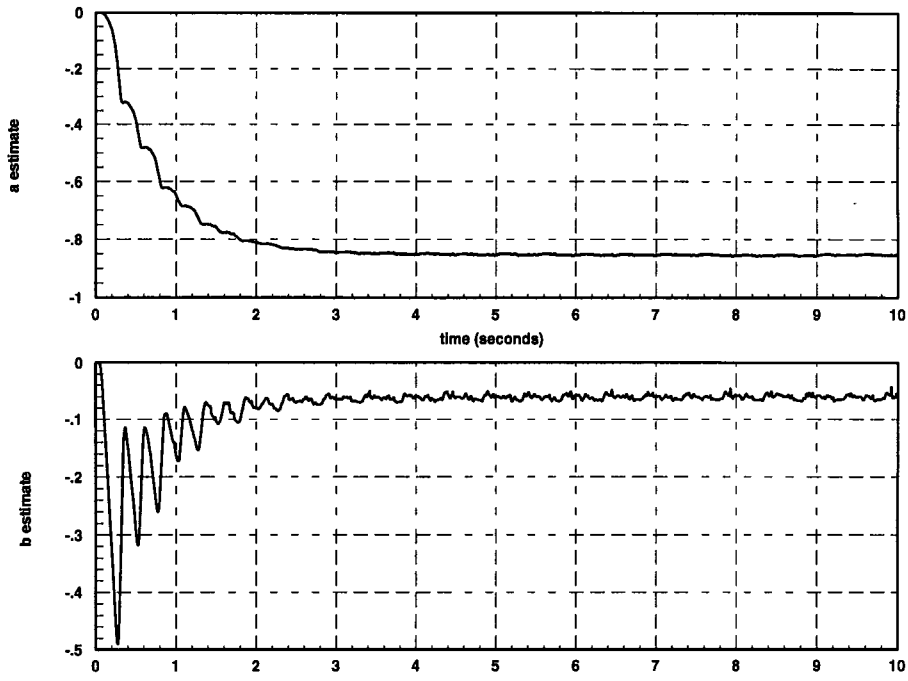

Figure 5.3: Parameters estimates of DC motor identification

wave is persistantly exciting up to order 2, hence it is sufficient to guarantee convergence of the first order model.

The results of the experiment with $\gamma = \alpha = 1$ are plotted in Figure 5.3 and Figure 5.4. Figure 5.3 shows how the estimated parameters a and b behave over time. As shown in the plot, a converges to -0.854 and b converges to -0.059. Hence, the model estimate is

$$T(z) = \frac{-0.059}{z - 0.854}. \tag{5.19}$$

Figure 5.4 shows the step response of this estimated model along with the measured step response of the motor. The two match very well.

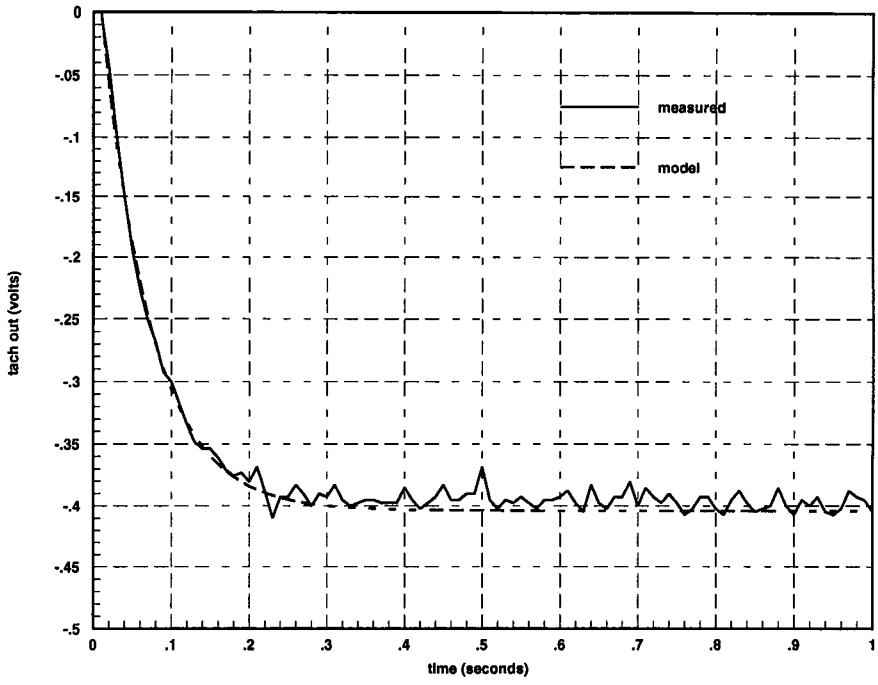

Figure 5.4: Step response of DC motor and estimated model

5.2 Identification Using Wavelets

Here matching pursuits (MP), a wavelet based real rational transfer function fitting algorithm, is used to perform system identification. The matching pursuits algorithm was developed by S. Mallat and Z. Zhang [21]. Additional work on the algorithm was done by Y.C. Pati, R. Rezaiifar, P.S. Krishnaprasad, and W.P. Dayawansa [24]. This work was then implemented in C code by Rezaiifar [28]. The use of the matching pursuits software has been integrated into a seamless system identification package using the AC-100 system. This is described below.

5.2.1 Using the Matching Pursuits Software

The MP algoritm produces a real rational transfer function model of a given system from the its experimentally determined complex frequency response. The

algorithm attempts to match the measured frequency response using a linear combination of a real rational analyzing wavelet and translations and dilations of the mother wavelet. The analyzing wavelet (or mother wavelet) [23] used by Rezaiifar's code is

$$\Psi(s) = \frac{1}{(s+\gamma)^2 + \xi^2}. \tag{5.20}$$

A wavelet which has been translated and dilated is then given by

$$\Psi_{m,n}(s) = a_0^{m/2}\Psi(a_0^m s - inb_0) \tag{5.21}$$

where a_0 is the dilation step size, m is the dilation level, b_0 is the translation step size, and n is the translation level [28]. The parameters a_0, b_0, γ, and ξ must be chosen wisely so that the algorithm matches the plant closely in the desired region of the frequency domain. Likewise, the user must choose the maximum number of translations (n_{max}) as well as the maximum and minimum dilation levels (m_{max} and m_{min}, respectively).

The procedure required to use the MP software is as follows:

Step 1: *Gather input/output data from plant to be identified.*

Step 2: *Calculate frequency response of plant from input/output data.*

Step 3: *Choose $a_0, b_0, \gamma, \xi, n_{max}, m_{max}$ and m_{min} based on the frequency response of the plant.*

Step 4: *Run matching pursuits software.*

5.2.2 Example: Flexible Beam

Here the system to be identified is the fixed flexible aluminum beam with a PZT sensor and PZT actuator as described in Section 6.1. PZT #1 is used as an

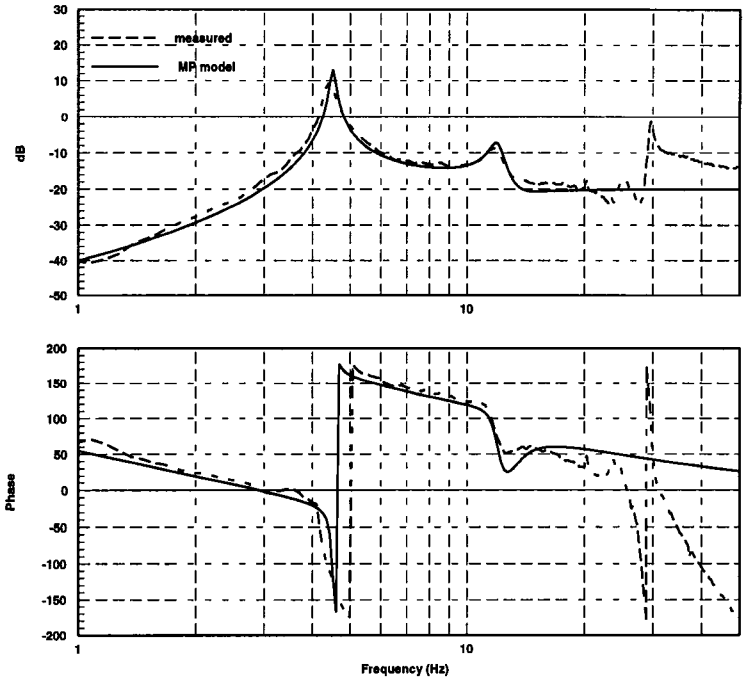

Figure 5.5: Flexible frequency response

actuator and PZT #2 is used as a sensor. The block diagram of the experimental set-up is shown in Figure 6.5. The input voltage to the power amplifier is defined to be the system input and the output voltage of the inverting amplifier is defined to be the system output.

The input to the beam is chosen to be the output of a uniformly distributed random number generator with maximum value 1 and minimum value -1. The input is updated every 0.01 seconds with a zero order hold between updates. The PZT voltage is sampled every 0.01 seconds with a zero order hold between samples. The values of the system input and output are stored over a period of 60 seconds. The data is then processed to generate the frequency response of the system ranging from 0.1 Hz to 50 Hz. This frequency response is plotted in Figure 5.5.

Now it is up to the designer to choose the MP parameters. The resonances of the flexible beam cause the frequency response to be well localized in the frequency domain. Since most of the vibrational energy of the beam is contained in the first few resonances, the rest of the frequency response is of little interest. In this example, the frequency range of interest lies from .1 Hz to 20 Hz. Hence, the MP parameters are chosen so that only wavelets localized to within this frequency range are used. This ensures a good match within the desired range while keeping the order of the generated model to a minimum. The frequency response of an 5th order model found using MP is shown along with the measured frequency response in Figure 5.5. As a point of comparison, RLS performed on the same system requires a 14th order model to achieve a similar match over the desired frequency range.

5.2.3 Conclusion

Both recursive least squares and matching pursuits provide methods of quickly determining an accurate model for SISO linear systems. RLS has the advantage that it is an on-line process and it is extremely easy to use. The user needs to know little more than the order of the system to be modeled in order to successfully employ RLS. These two facts make RLS useful for adaptive methods such as the self tuning regulator [1]. Matching pursuits, on the other hand, is more difficult to use. In order to successfully implement MP the user must first experimentally determine the system's frequency response. Additionally, MP has several parameters which must be intelligently chosen by the user in order to obtain a good model. There is however a reward for this extra effort. Once the parameters are correctly chosen MP provides an accurate model of much

lower order than RLS. In addition to system identification, MP can be used to reduce the order of existing models, making complex models more manageable. Hence, as is common in engineering, the designer must take these factors into consideration and choose which method of system identification is best to use for a given application.

Both methods have been integrated into seamless, easy to use sytems in the Intelligent Servosystems Laboratory using the AC-100. Any stable linear plant can now be quickly identified with a relatively small amount of effort.

Chapter 6

Fixed Flexible Beam Experiment

Much work has been done in recent years concerning vibration damping in a flexible beam using piezo-electric ceramic (PZT) actuators and sensors [8] [6] [12] [27]. However, very few have directly addressed the issue of control constraints. The voltage applied to a PZT actuator is limited by the PZT breakdown voltage as well as potential safety concerns. It is desirable to obtain the maximum possible control effort without exceeding this voltage limit. One way to do this is to use the constrained control feedback laws developed in Chapter 3. This chapter will discuss the use of the constrained controller presented by Gutman and Hagander [13] and discussed here in Section 3.2.2. The performance of the Gutman-Hagander (GH) controller is then compared to the performance of two popular controllers in frequent use which ignore the control constraints, the linear quadratic regulator (LQR) and the positive position feedback (PPF) controller [9].

6.1 Experimental Set Up

The beam consists of a rectangular piece of aluminum 30 inches long, 1 inch wide and 0.125 inches thick. One end of the beam is mounted into a solid aluminum vise, which is in turn mounted to a workbench. The beam is mounted in the vise so that both longest and shortest edges are parallel to the ground. The distance from the tip of the beam to the edge of the vise is 26 inches. This is shown in Figure 6.1. Five PZT pairs are mounted on the beam, each of them can be used either as a sensor or an actuator. More information about the PZT pairs is provided in Section 6.1.1. The PZT pairs are distributed along the beam as shown in Figure 6.1 and are numbered 1-5, with pair #1 being closest to the vise. An accelerometer (Sundstrand Data Control, Inc. model 2180) is mounted at the tip of the beam.

For this experiment, only PZT pair #1 and #2 will be used. Pair #1 will be used as the control actuator and pair #2 will be used as a sensor.

6.1.1 The PZT Pairs

Each PZT pair consists of two individual 1 inch by 2 inch by 0.010 inch PZT crystals (Matroc part number 6/648). The crystals are mounted on opposite sides of the beam and wired with opposite polarity so that they work together. If one of the elements expands, its mate (on the other side of the beam) contracts. Using the PZT's in this fashion doubles the maximum possible control authority and ensures that a bending moment is applied to the beam symmetrically.

Mounting the PZT's onto the beam with consistent results turns out to be somewhat of a difficult problem. The most straightforward method of bonding a

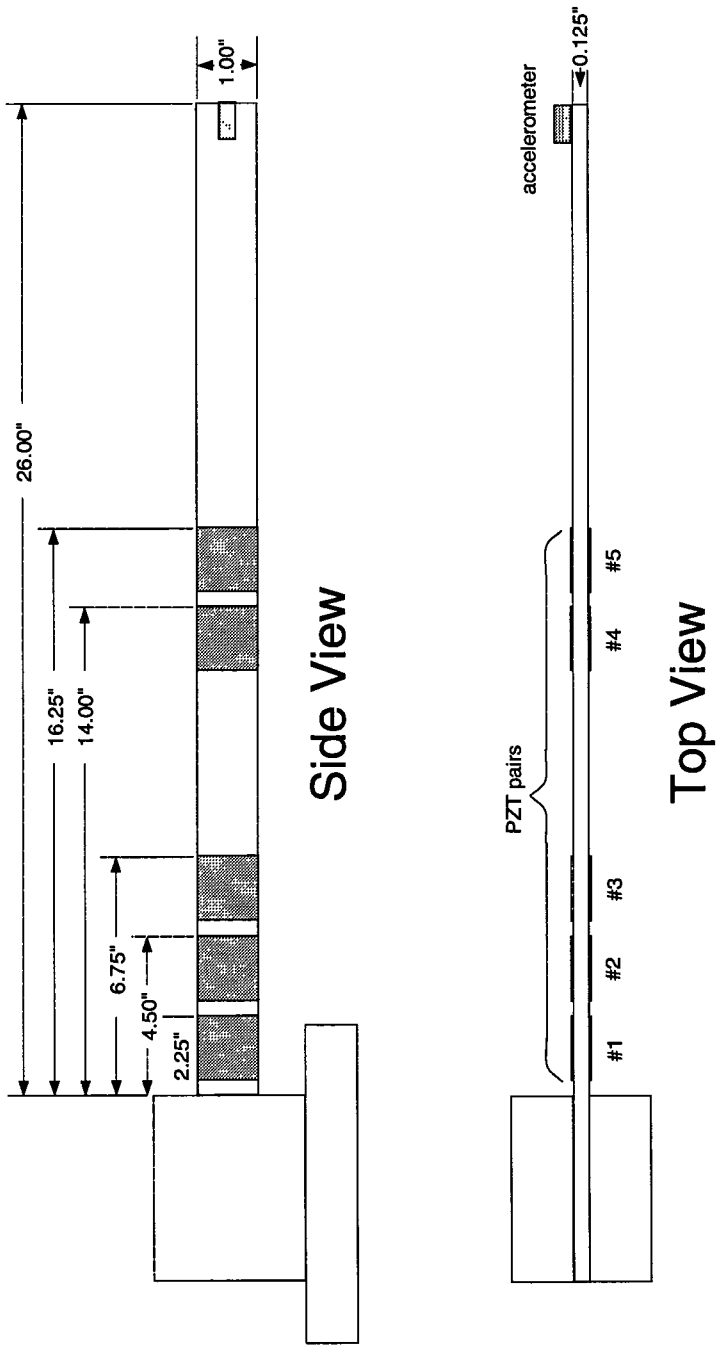

Figure 6.1: Flexible beam diagram (not to scale)

PZT to the beam is to simply glue it onto the beam flat using a strain gauge glue such as M-Line's M-200 epoxy. This method presents some complications. First, the mechanical coupling between the PZT and beam is inconsistent from trial to trial. Electrical coupling problems also arise. For most applications it is desirable to have one side of the PZT electrically connected to the beam. A conductive epoxy might seem to solve this problem, but commercially available conductive epoxies do not bond nearly as well as their non-conductive counterparts. Also, conductive epoxy can seep around one of the edges of the PZT and short the two sides together, making the PZT useless. Non-conductive epoxies bond well and present no potential of shorting the PZT, but the electrical connection between the PZT and the beam is hard to guarantee. In addition, some interesting PZT control techniques (such as Dosch, Inman and Garcia's self-sensing PZT actuators [8]) require that both sides of the PZT be electrically isolated from the beam. So it would be nice to be able to connect and disconnect the electrical connection at will. Finally, placing the PZT's accurately on the beam can be difficult. The epoxy cures almost immediately, so the PZT will be bonded to the beam wherever it first makes contact. Hence, the PZT must be placed accurately on the first try, a difficult task for even the most steady-handed engineer.

All of these problems are solved by the mounting scheme depicted in Figure 6.2. The glue used is M-Line's M-200 epoxy, which is non-conductive. Since only the ends of the PZT are close to the beam, only the ends will become bonded. This provides a consistent, easily repeatable mechanical coupling from beam to PZT. Another advantage of the small bonding area between the PZT and the beam is that it is easy to ensure that the epoxy layer will electrically insulate the PZT from the beam. Sliding a coiled piece of wire in the gap between PZT

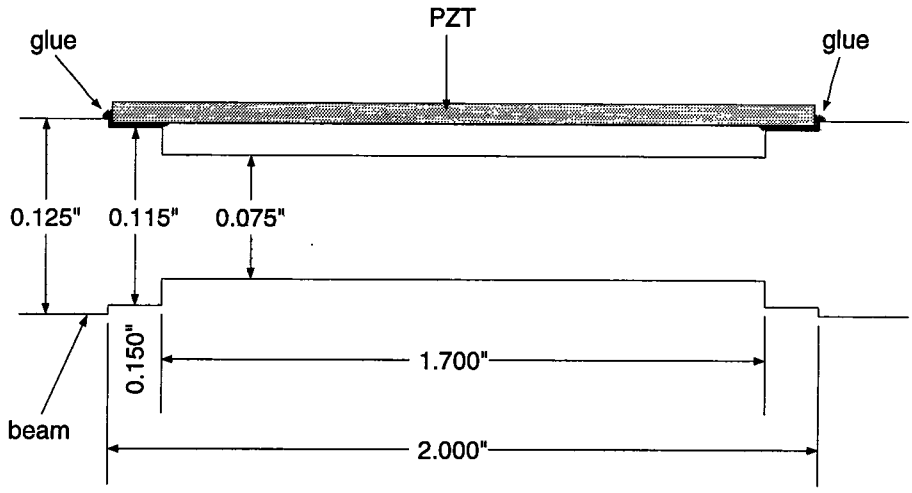

Figure 6.2: PZT mounting scheme

and beam will provide an electrical connection if one is desired. If no connection is desired, the coil can be removed. Finally, the top of the opening is machined to fit the PZT precisely, making it easy to place the PZT in the proper position accurately.

6.1.2 Electrical Description

The control signal is generated by AC-100 and is provided to the system as an analog voltage via one of the AC-100's DAC outputs. The DAC output is fed into the input of a low pass filter with a cut-off frequency of 200 Hz and a DC gain of 0.61 V/V. The schematic for this filter is shown in Figure 6.3. The purpose of the filter is to remove the sharp transitions (steps) in the DAC waveform while preserving the control intent contained in the low frequency components of the signal. This is needed since the PZT does not respond well to the sharp transitions of the DAC. The filter output is then fed into a power amplifier with a gain of 36 V/V. The schematic for the power amplifier is given in Figure 6.4.

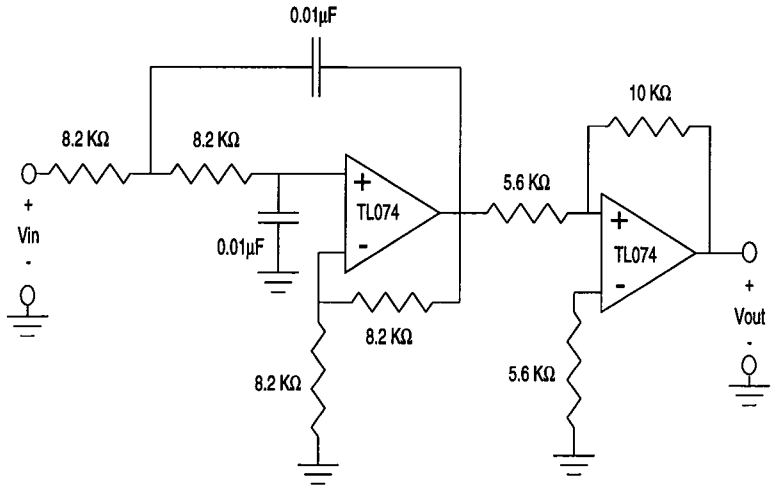

Figure 6.3: Low pass filter schematic

The output of the power amplifier is then fed into one side of PZT pair #1. The other side of PZT pair #1 is connected to the beam, which is grounded.

PZT pair #2 is used as a sensor. One end of the pair is connected to the beam (ground) and the other end is connected to a TL074 op amp configured as an inverting amplifier with a gain of 0.21. This attenuating amplifier is necessary so that the signal going into the computer does not exceed the +/- 5 volt range of the ADC. A block diagram of the entire set up is given in Figure 6.5.

6.2 Modeling the System

An accurate model of the flexible beam system is required in order to implement the design algorithms mentioned in the beginning of this chapter. The model is also useful for simulations. Additionally, a physical model can provide insight which can lead to a thorough understanding of how the system operates. Here, a model of the flexible beam derived from the physical properties of the system is considered. The properties of this model are discussed and then compared with

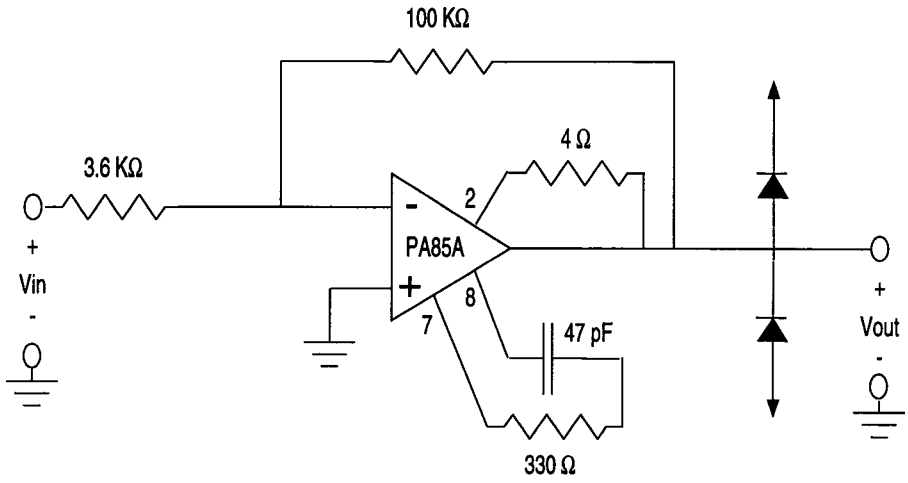

Figure 6.4: Power amplifier schematic

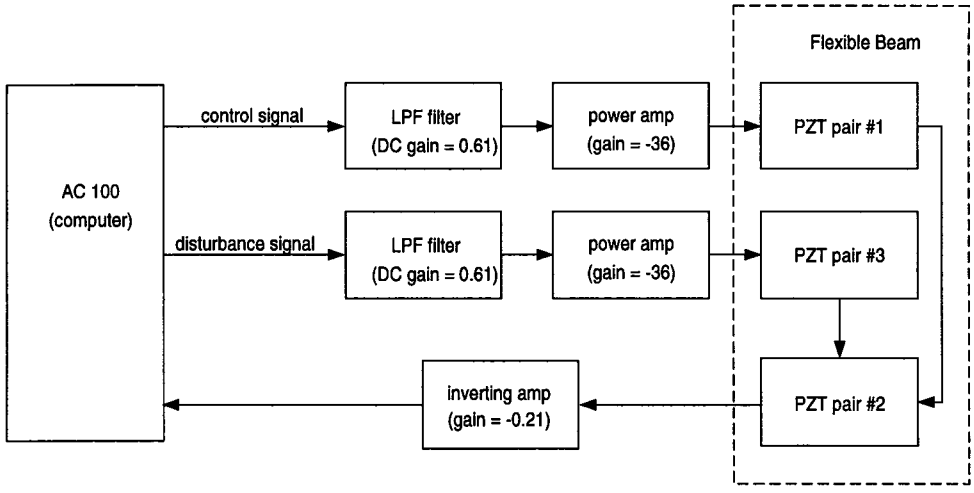

Figure 6.5: Block diagram of experimental set up

the experimental model found using wavelets in Section 5.2.2.

6.2.1 Finite Element Model

The dynamics of a flexible cantilever beam are inherently infinite dimensional. An infinite dimensional model is not useful for the purposes of this problem, hence it is necessary to approximate the beam with a finite dimensional system. One popular method of doing this is to approximate the beam as a finite number of rigid bodies connected by rotary joints with torsional springs as depicted in Figure 6.6. This is known as the Galerkin model [2]. The Euler-Lagrange method can be used to generate finite dimensional nonlinear equations of motion for the finite element model. Since the vibrations in the flexible beam consist of relatively small displacements the nonlinear model can be linearized about $\theta_k = \dot{\theta}_k = 0, k = 1, \ldots, N$ where θ_i represents the slope of the ith element and N represents the total number of elements in the model. In his M.S. thesis, R. Rezaiifar derives and linearizes the equations of motion [28]. PZT pair #1 (the actuator) can be used to generate a moment about the first joint of the model. It is assumed that this moment is proportional to the voltage applied to the PZT pair. Likewise, PZT pair #2 (the sensor) is used to measure the angle of the first element. It is assumed that the voltage generated by the PZT is directly proportional to the angle.

This model provides some good intuition about the behavior of the flexible beam. Stiffening the springs in the model causes the resonances to shift to the right in the frequency domain. Reduction of the mass or reduction of the length of the beam has the same effect. Distributing mass unevenly along the beam will produce non-symmetrical mode shapes since the motion of the heavy

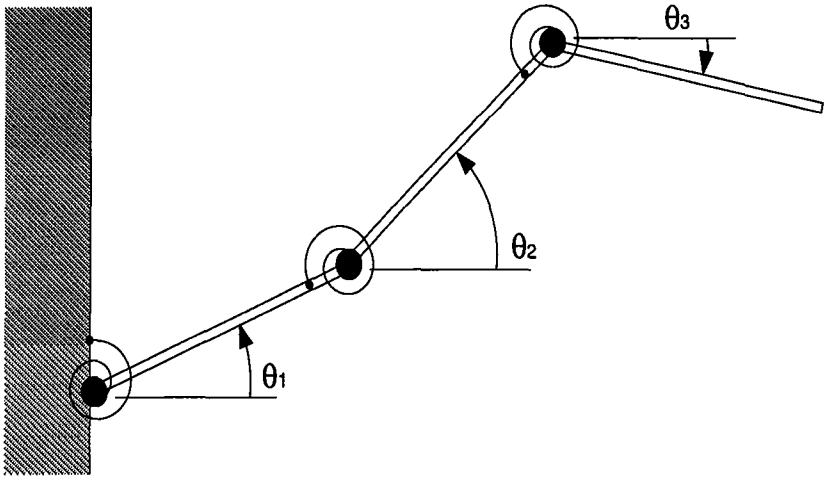

Figure 6.6: 3 element Galerkin model

elements will dominate the motion of the lighter elements. These observations are consistent with the behavior of the actual beam.

The model also provides some insight into ways to control the system. Proportional feedback will effectively increase the spring constant at the first joint, causing the beam to act stiffer. Derivative or velocity feedback will add damping to the first joint, causing vibrations in the overall beam to become more damped. These results match previous experimental findings for an actual beam[8].

6.2.2 Experimental Model

The Galerkin model is very useful for understanding the flexible beam, but it is not accurate enough to be used for controller design. This model relies on too many assumptions (such as the relationship between the actuator and joint moment) and too many unmeasureable parameters (such as the spring constants) to be reliable. Hence, an experimental model is needed.

The transfer function from the input to the filter connected to PZT pair #

the the output of PZT pair #2 was found experimentally using the wavelet based matching pursuits algorithm discussed in Section 5.2.2. The experimentally determined transfer function was found to be

$$T(s) = \frac{0.1(s-1.46)(s-7.68)(s-75.4)(s^2+17.6s+6.69\times10^3)}{(s+75.4)(s^2+0.777s+802)(s^2+6.13s+5.66\times10^3)}. \quad (6.1)$$

This 5th order model will be used to design the controllers used in the remainder of this chapter.

6.3 Controller Design

The three controllers to be considered in this chapter are the Gutman-Hagander controller (GH), the linear quadratic regulator (LQR), and the positive position feedback controller (PPF). The GH and LQR design algorithms are based on a state space model, so the real rational transfer function model given in the previous section must be translated into its equivalent state equation form. Such a translation (or realization) is a standard classical control technique and is discussed in any good book on linear control theory. Additionally, both GH and LQR require state feedback. Clearly, the states in the state space model are artificial and cannot be measured. Hence an observer will need to be designed. This section briefly describes the design of the observer as well as the GH, LQR, and PPF controllers.

6.3.1 Observer Design

Consider the observable LTI state space system given by

$$\dot{x}(t) = Ax(t) + Bu(t), x(0) = x_0$$

$$y(t) = Cx(t) \tag{6.2}$$

where $x \in R^n$, $u \in R^m$ and $y \in R^p$. The classical state space observer for this system is given by Rugh [29] as

$$\begin{aligned} \dot{\hat{x}}(t) &= A\hat{x}(t) + Bu(t) + H(y(t) - \hat{y}(t)), \hat{x}(0) = \hat{x}_0 \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{6.3}$$

Where H is known as the observer gain matrix. The error between actual and estimated states, $e(t) = x(t) - \hat{x}(t)$ then obeys the state equation

$$\dot{e}(t) = (A - HC)e(t), \quad e(0) = x_0 - \hat{x}_0. \tag{6.4}$$

Hence, the error of the estimated state can be made to go to zero by choosing H so that Equation 6.4 is exponentially stable. This is always possible since $[A, C]$ is observable.

There are many methods of choosing the observer gain matrix, H. Here H is chosen using standard LQR methods. Note that

$$\begin{aligned} \sigma(A - HC) &= \sigma((A - HC)^T) \\ &= \sigma(A^T - C^T H^T) \end{aligned} \tag{6.5}$$

where $\sigma(\cdot)$ denotes the spectrum. H^T can then be chosen so that the cost function

$$J(u) = \int_0^\infty \left(\|u(t)\|_2^2 + \alpha \|z(t)\|_2^2 \right) dt \tag{6.6}$$

is minimized for the system

$$\dot{z}(t) = A^T z(t) + C^T u(t), z(0) = z_0. \tag{6.7}$$

The input u which minimizes the cost function $J(u)$ for the above system is

$$u(t) = C\Pi_\infty x(t) \tag{6.8}$$

where Π_∞ is the unique symmetric positive definite solution to the algebraic Ricatti equation (ARE) [31]

$$A\Pi_\infty + \Pi_\infty A^T - \Pi_\infty C^T C \Pi_\infty = -\alpha I \tag{6.9}$$

where I is the $(n \times n)$ identity matrix. Hence,

$$H = (C\Pi_\infty)^T. \tag{6.10}$$

The parameter α represents the penalty on the observer error. This number should be chosen to be large so that the observer state will converge quickly to the actual state. Here, α was chosen to be 10^4.

6.3.2 Linear Quadratic Regulator

The linear quadratic regulator (LQR) is a well known algorithm to design a linear state feedback law for any linear time invariant system. The LQR method can be summarized as follows: Let Π_∞ be the unique symmetric positive definite solution the the algebraic Ricatti equation

$$A^T\Pi_\infty + \Pi_\infty A - \Pi_\infty B B^T \Pi_\infty = -L \tag{6.11}$$

where the $(n \times n)$ matrix L is symmetric and positive semi-definite. The linear state feedback law

$$u(t) = -B^T \Pi_\infty x(t) \tag{6.12}$$

will stabilize the system given in Equation 6.2 while minimizing the cost function

$$J(u) = \int_0^\infty \left(\langle u(t), u(t) \rangle + \langle x(t), Lx(t) \rangle \right) dt. \tag{6.13}$$

$L = L^T$ is the state penalty matrix. Since the control objective is to drive the system output $y(t) = Cx(t)$ to zero as quickly as possible, we set $L = \alpha C^T C$

so that $\langle x(t), Lx(t) \rangle = \alpha \langle y(t), y(t) \rangle$. Now the LQR controller can be determined by choosing one parameter, α. In simulation, a good value for α was found to be 0.5. The state feedback matrix found using the LQR algorithm with $\alpha = 0.5$ will hereafter be know as K_{LQR}. Hence, the control with reference input $v(t)$ becomes

$$u(t) = -K_{LQR}x(t) + v(t). \tag{6.14}$$

The system under LQR control implemented using the state observer found in Section 6.3.1 now becomes

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix} &= \begin{bmatrix} A & -BK_{LQR} \\ HC & A - HC - BK_{LQR} \end{bmatrix}\begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} v(t) \\ y(t) &= \begin{bmatrix} C & [0 \cdots 0] \end{bmatrix}\begin{bmatrix} x(t) \\ \hat{x}(t) \end{bmatrix}. \end{aligned} \tag{6.15}$$

6.3.3 Gutman-Hagander Controller

The constrained input controller design algorithm proposed by P. Gutman and P. Hagander was described briefly in Section 3.2.2. A more detailed description of the algorithm was presented by R. Rezaiifar in his M.S. thesis [28] and in the original paper by Gutman and Hagander [13]. The algorithm as applied to the flexible beam problem is described here. It is assumed that the input u is constrained so that $|u(t)| \leq 1$ for all $t \geq 0$.

Step 1: Finding the Initial Condition Set

The first step in the GH control algorithm is to find the set of possible initial conditions that the plant can take, denoted as D. To do this, the output of the beam (PZT pair #2) was connected to the observer designed in Section 6.3.1

and the estimated states were observed while the tip of the beam was manually deflected by one inch and released. This is considered to be the maximum possible disturbance input. The maximum absolute value attained by each state was recorded. These are denoted as $x_i^{max}, i = 1, \ldots, 5$. The set D is then chosen to be $\{x \in R^5 | \, |x_i| \leq x_i^{max}, i = 1, \ldots, 5\}$. It should be noted that this estimate for D is somewhat conservative since all of the states may not reach their maximum value at the same time. Still, it will suffice for this problem.

Step 2: Find Stabilizing Linear Feedback

The second step of the GH control algorithm is to find a linear state feedback matrix L which stabilizes the system while satisfying the input constraint for all $x \in D$. The flexible beam is already a stable system, so this step is not necessary. Furthermore, since L is not needed, the set $E = \{x \in R^5 | \, |u| = \left|L^T x\right| \leq 1\}$ is equal to all of R^5.

Step 3: Find P

Step 3 of the GH algorithm is to find $P = P^T > 0$ so that P solves the Lyapunov equation

$$PA + A^T P = -Q \tag{6.16}$$

for some positive definite Q. Q can be though of as a design parameter. Here, Q is chosen to be the 5×5 identity matrix. Equation 6.16 is solved easily using MatrixX.

Step 4: Check Validity of P

In order for P to be a valid choice, it must satisfy the inequality

$$\sup_{x \in D} x^T P x \leq \min_{x \in \partial E} x^T P x. \tag{6.17}$$

Since $E = R^5$ in this case, this inequality holds.

Step 5: Choose K

Now the control law

$$u = -sat(K B^T P x) \tag{6.18}$$

is guaranteed to stabilize the flexible beam system for all initial conditions $x_0 \in D$. K can be any $(m \times m)$ diagonal matrix with all of its elements greater than or equal to zero. Since the flexible beam is a single input system $(m = 1)$ K is a non-negative scalar.

To select K, MatrixX was used to plot the locations of the eigenvalues of the matrix $(A - BKB^T P)$ for various values of K. K was then chosen to be the value for which these eigenvalues were pushed farthest into the left half plane. This yielded $K = 0.002$.

6.3.4 Positive Position Feedback

Here the positive position feedback (PPF) method presented by J. Fanson and T. Caughey [9] is used to design a controller for the flexible beam. In the PPF controller, the position signal (i.e. the voltage of PZT pair #2) is passed through a bank of parallel second order low pass filters. One filter is required for each frequency that is to be controlled. For the flexible beam, it is only necessary to control disturbances of the frequencies which correspond to the resonances of

the beam since all other frequencies are naturally attenuated. Each filter has a transfer function of the form

$$T(s) = \frac{k\omega_c^2}{s^2 + 2\xi\omega_c s + \omega_c^2} \tag{6.19}$$

where ω_c is the filter's cut-off frequency in radians per second.

For the flexible beam, it is desirable to control the first two resonances of the beam. These are at 24.4 rad/s and 75.4 rad/s for the first and second modes, respectively. Hence, two filters are needed. The cut-off frequency for each filter is chosen to be 1.1 times the frequency of the mode to be controlled. The parameters k and ξ for the two filters are adjusted during a preliminary simulation step. The damping ratio, ξ, was chosen to be 0.2 for each filter. The DC gains were chosen to be 0.05 for the first filter and 0.01 for the second. See Figure 6.7 for a block diagram of the system under PPF control.

It should be noted that in addition to filtering the amplitude of the position signal, the PPF filters also shift the phase of the cut-off frequency by -90 degrees. In other words, PPF is equivalent to feeding back the velocity signal (derivative of position) of each mode to be controlled. According to the Galerkin model, this introduces a pure damping term at the first link and the system should be well damped. In fact, PPF feeds back pure damping even when the control signal is saturated. Hence, the PPF controller should work well under the input constraints.

6.4 Results

To test these three controllers, the tip of the beam was manually deflected by exactly one inch and released. The beam response was then measured. The

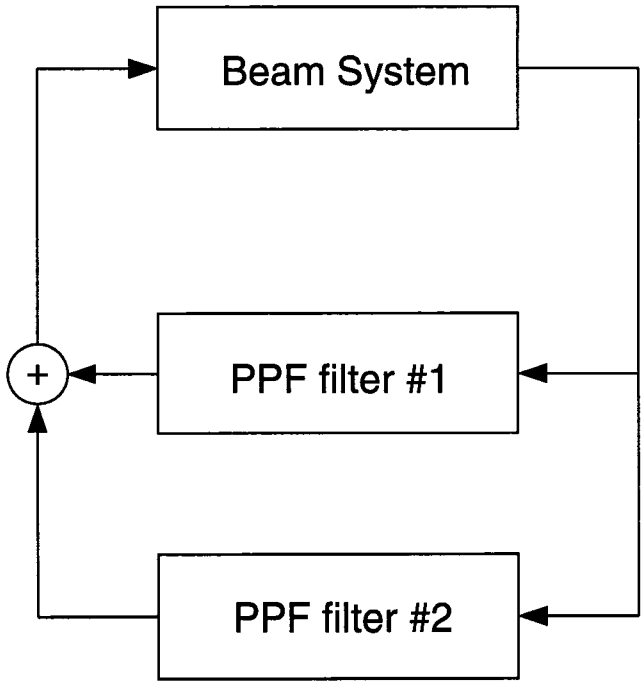

Figure 6.7: Block diagram of PPF controller

results of this experiment are plotted for each of the three controllers in Figures 6.8, 6.9, and 6.10. All three controllers improved the ability of the beam to reject the disturbance. The settling time (10%) of the beam with no controller was 6.42 seconds. The settling time with the LQR controller was 2.89 seconds while both the PPF and GH controllers had settling times of 2.45 seconds.

The settling time is defined to be the latest time at which the beam response exceeds ten percent of its maximum value. Because of the oscillatory nature of the beam response, comparing two responses using this measurement can be slightly misleading, especially when the two responses are very close. A more reliable method of comparing beam responses is to compare the plots of the responses' envelopes. As Figure 6.11 shows, the response of the GH controlled beam is slightly better than the response of the PPF controlled beam.

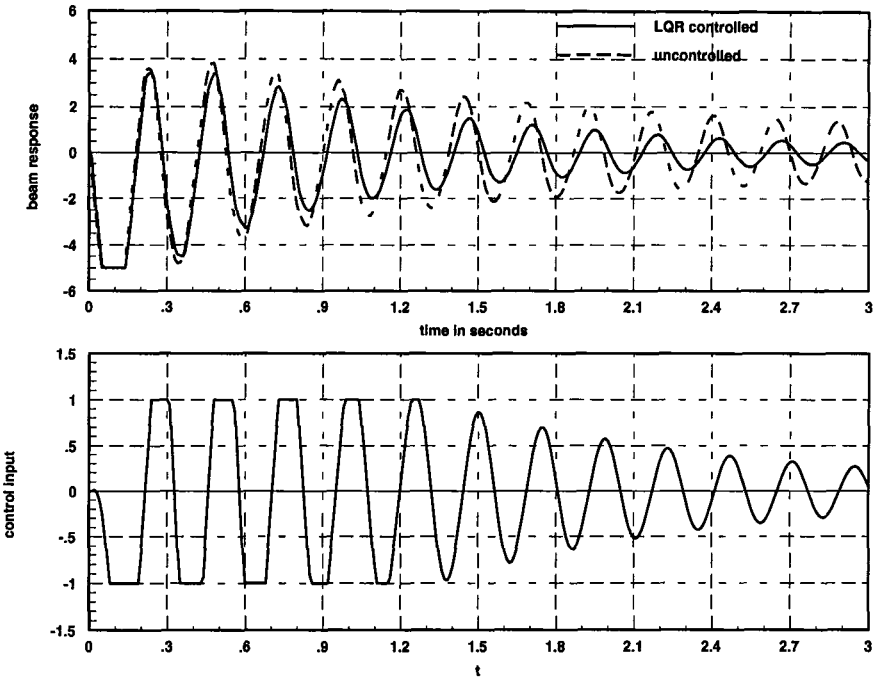

Figure 6.8: Response of the beam under LQR control

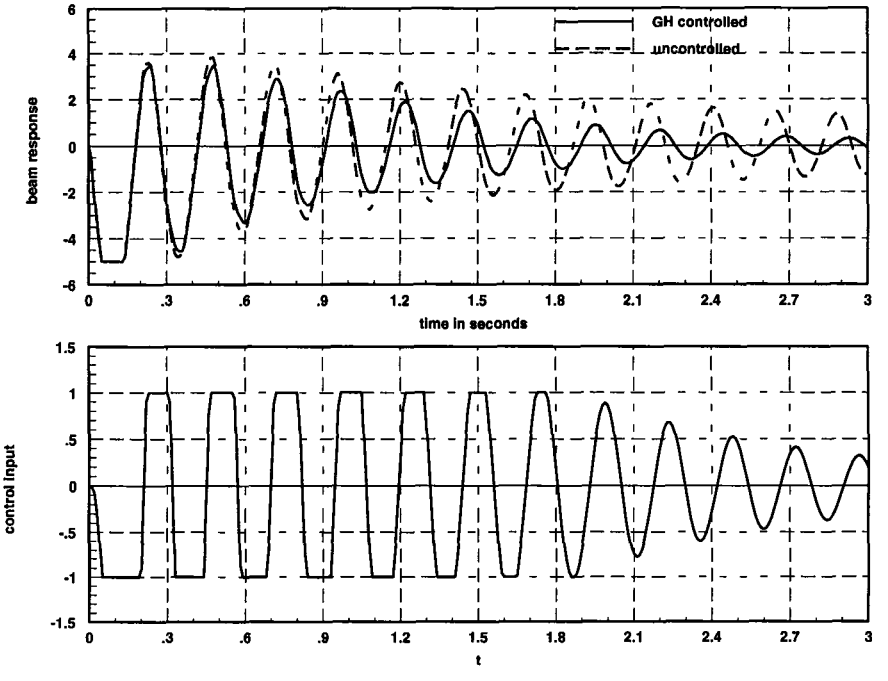

Figure 6.9: Response of the beam under GH control

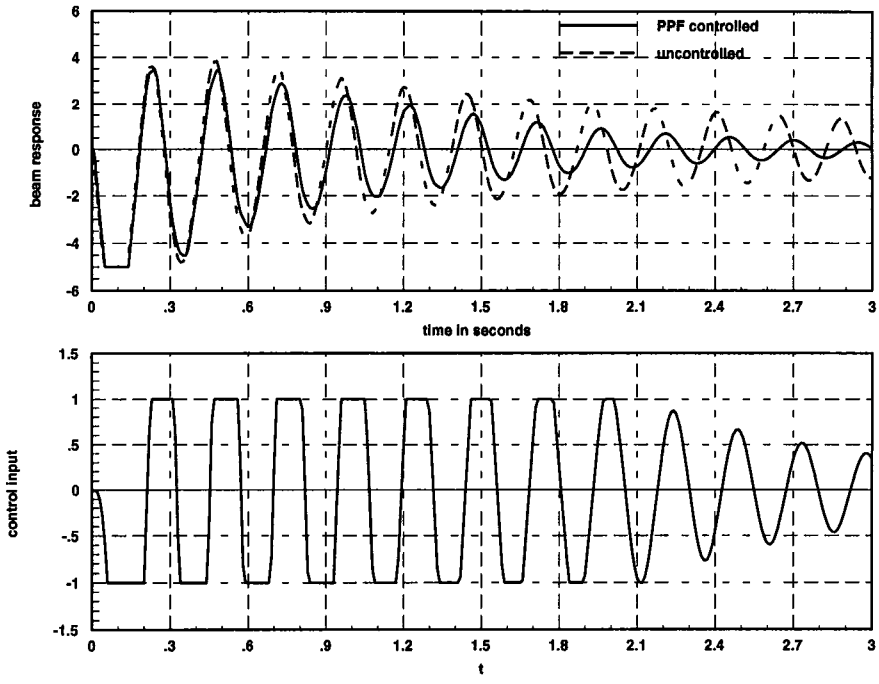

Figure 6.10: Response of the beam under PPF control

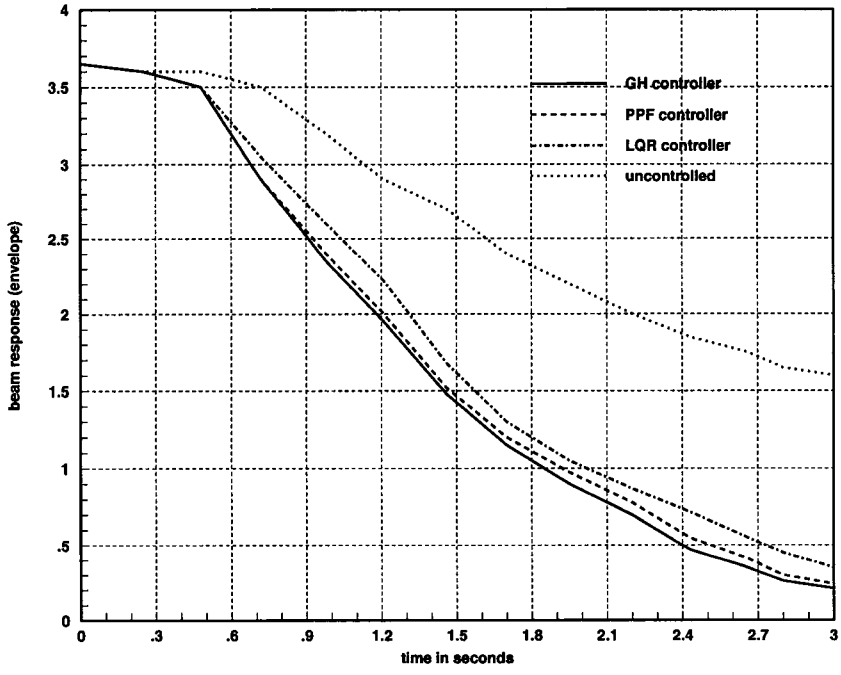

Figure 6.11: Comparison of beam response envelopes

Recall that the control input was constrained so that $|u| \leq 1$. If this constraint is removed, the performance of the controllers is significantly increased. The unconstrained settling times were 1.76 seconds, 1.44 seconds, and 1.37 seconds for the LQR, GH, and PPF respectively. Such an improvement is expected.

6.5 Conclusion

All three controllers improved the beams ability to reject the disturbance. The LQR controller had the worst performance of the three, but this is to be expected since the LQR algorithm is not specifically designed for use as a saturating controller. The GH and PPF controllers performed almost equally well. Overall, PPF is probably a better choice of controller because it is much simpler to implement than the LQR and GH, both of which require an observer.

This experiment provides an important lesson. Automated design algorithms such as LQR and GH are nice to have, but they are no replacement for a thorough understanding of the physics of the system to be controlled. The PPF controller, which was designed based on intuition provided by the Galerkin beam model, performed almost as well as the fancier, automated methods and was much simpler to implement.

Chapter 7

Rotating Flexible Beam Experiment

This chapter addresses the problem of active damping of vibrations in a continuously rotating flexible beam. Like the fixed beam experiment of Chapter 6, an aluminum cantilever beam with PZT sensors and actuators is used. However, for this experiment the beam is mounted to a hub driven by a DC motor. This creates one very interesting twist to the fixed beam experiment: the parameters of the beam vary with the motor rotation rate. In particular, the resonant frequencies of the beam shift to the right in the frequency domain as the motor velocity increases. Hence, this experiment provides an interesting application for the design algorithms of robust and adaptive control.

In this chapter, the experimental set-up is described in detail. A model for the beam is developed from physical principles and compared to models found using the system identification techniques of Chapter 5. Linear controllers are designed for the system at two separate, fixed motor velocities. Finally, these two controllers are united to create a gain scheduling controller capable of providing active vibration damping to the beam over a range of slowly changing motor velocities.

7.1 Experimental Set Up

The beam is identical to the beam described in Section 6.1 with one major difference: single PZT crystals are mounted along the beam on one side only as opposed to the PZT pairs mounted along both sides of the fixed beam. The reasoning behind this is discussed at the end of this section. The PZT's are mounted to the beam using the mounting scheme described in Section 6.1.

The flexible arm apparatus designed and constructed by G.H. Frank [10] was modified to suit the needs of this experiment. Since the beam needs to rotate continuously, it was necessary to add a slipring to Frank's set-up so that electrical contact between the PZT's and the controller could be maintained during operation. To facilitate this, a new hub was constructed. The new hub extends approximately 8 inches upward from the motor housing, providing room to mount an 8 channel slipring. Mounted to the top of the hub is a removable circular platform with a 7 inch diameter. The beam is clamped to the platform with the PZT side facing down such that PZT #1 closest to the hub.

The electrical set-up is identical to the set-up described for the fixed beam (Section 6.1.2) with the exception that the connections to the PZT's are made via the slipring. For the purposes of this discussion, the slipring will be treated as a perfect short circuit at all times.

Clearly, at equilibrium the beam will bend downward due to the force of gravity. PZT's mounted on the top side of the beam would be extended, providing them with a negative mechanical bias. This is not good for the crystals and causes them to crack easily. For this reason, all of the PZT's are mounted on the bottom side of the beam only.

7.2 Modeling the System

As discussed in Section 6.2, an accurate model of the flexible beam system is useful for many reasons. Here, a model of the flexible beam derived from the physical properties of the system is considered. The properties of this model are discussed and then compared with models found using experimental methods.

7.2.1 Finite Element Model

The Galerkin model discussed in Section 6.2.1 can be easily applied to this problem with a couple of modifications. The two major differences between this problem and that of the fixed beam come from the gravitational and centrifugal forces. Each link in the model will be subject to a constant downward force of $m_i g$ where m_i is the mass of the ith link and g is the acceleration due to gravity. This accounts for the effect of gravity. Additionally, each link is subject to a force of $\omega m_i x_i$ in a direction radially outward from the hub. Here ω is the angular velocity of the motor and x_i is the x position of the center of gravity of the ith link. This accounts for the centrifugal force due to rotation. Since both of these forces are applied in a constant direction with respect to the beam, the rotating beam can be modeled as a fixed beam with these new forces in place. A diagram of a 3-link version of this model is shown in Figure 7.1.

The equations of motion for this system can be found using the Euler-Lagrange method. For and n-link beam

$$\tau_k - \tau_{k+1} = \frac{d}{dt}\left(\frac{\partial E}{\partial \dot{\theta}_k}\right) - \frac{\partial E}{\partial \theta_k} + \frac{\partial U}{\partial \theta_k}, \tag{7.1}$$

where

θ_i = the slope of the ith link,

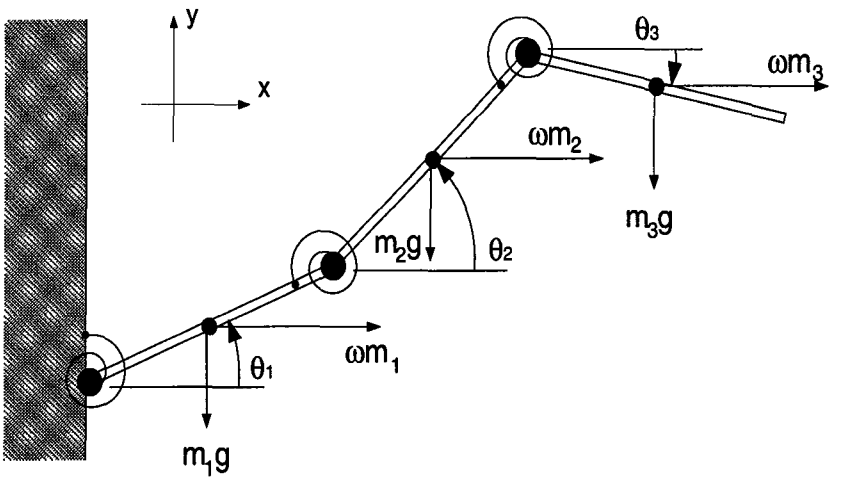

Figure 7.1: 3 element Galerkin model of continuously rotating flexible beam

E = the kinetic energy of the system,

U = the potential energy, and

τ_i = the external moment generated at the ith joint.

The kinetic energy can be expressed as a function of the translational and rotational velocities of each link. Specifically,

$$E = \sum_{i=1}^{n} \frac{1}{2} m_i (\dot{x}_i^2 + \dot{y}_i^2) + \sum_{i=1}^{n} \frac{1}{2} I_i \dot{\theta}_i^2 \tag{7.2}$$

where

m_i = the mass of the ith link,

I_i = the moment of inertia of the ith link about its center of mass, and

(x_i, y_i) = the cartesian position of the center of mass of the ith link.

Likewise, the potential energy can be expressed as the sum of the energy stored in the springs, the displacement of the links against gravity, and the displacement of the links against the centrifugal force. Specifically,

$$U = \sum_{i=1}^{n} \frac{1}{2} k_i (\theta_i - \theta_{i-1})^2 + \sum_{i=1}^{n} m_i g y_i - \sum_{i=1}^{n} m_i \omega x_i^2, \tag{7.3}$$

where

ω = the angular velocity of the motor and

g = the acceleration due to gravity.

This is identical to Rezaiifar's formulation of the fixed beam model [28] with the addition of the gravity and centrifugal force terms in the potential energy equation.

Tedious application of the chain rule yields

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_k}\right) &= \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\sum_{l=1}^{i}\frac{\partial^2 x_i}{\partial\theta_j\partial\theta_l}\dot{\theta}_j\dot{\theta}_l+\sum_{j=1}^{i}\frac{\partial x_i}{\partial\theta_j}\ddot{\theta}_j\right)\frac{\partial x_i}{\partial\theta_k}\right] \\
&+ \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\sum_{l=1}^{i}\frac{\partial^2 y_i}{\partial\theta_j\partial\theta_l}\dot{\theta}_j\dot{\theta}_l+\sum_{j=1}^{i}\frac{\partial y_i}{\partial\theta_j}\ddot{\theta}_j\right)\frac{\partial y_i}{\partial\theta_k}\right] \\
&+ \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\frac{\partial x_i}{\partial\theta_j}\dot{\theta}_j\right)\sum_{l=1}^{i}\frac{\partial^2 x_i}{\partial\theta_k\partial\theta_l}\dot{\theta}_l\right] \\
&+ \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\frac{\partial y_i}{\partial\theta_j}\dot{\theta}_j\right)\sum_{l=1}^{i}\frac{\partial^2 y_i}{\partial\theta_k\partial\theta_l}\dot{\theta}_l\right]+I_k\ddot{\theta}_k, \qquad (7.4)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial U}{\partial\theta_k} &= k_k(\theta_k-\theta_{k-1})-k_{k+1}(\theta_{k+1}-\theta_k) \\
&+ \sum_{i=1}^{n}m_i g\frac{\partial y_i}{\partial\theta_k}-\sum_{i=1}^{n}2m_i\omega x_i\frac{\partial x_i}{\partial\theta_k}, \qquad (7.5)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial E}{\partial\theta_k} &= \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\frac{\partial x_i}{\partial\theta_j}\dot{\theta}_j\right)\left(\sum_{l=1}^{i}\frac{\partial^2 x_i}{\partial\theta_k\partial\theta_l}\dot{\theta}_l\right)\right] \\
&+ \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\frac{\partial y_i}{\partial\theta_j}\dot{\theta}_j\right)\left(\sum_{l=1}^{i}\frac{\partial^2 y_i}{\partial\theta_k\partial\theta_l}\dot{\theta}_l\right)\right]. \qquad (7.6)
\end{aligned}$$

Substituting Equations 7.4, 7.5, and 7.6 into equation 7.1 yields

$$
\begin{aligned}
\tau_k - \tau_{k+1} &= \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\sum_{l=1}^{i}\frac{\partial^2 x_i}{\partial\theta_j\partial\theta_l}\dot{\theta}_j\dot{\theta}_l + \sum_{j=1}^{i}\frac{\partial x_i}{\partial\theta_j}\ddot{\theta}_j\right)\frac{\partial x_i}{\partial\theta_k}\right] \\
&+ \sum_{i=1}^{n}\left[m_i\left(\sum_{j=1}^{i}\sum_{l=1}^{i}\frac{\partial^2 y_i}{\partial\theta_j\partial\theta_l}\dot{\theta}_j\dot{\theta}_l + \sum_{j=1}^{i}\frac{\partial y_i}{\partial\theta_j}\ddot{\theta}_j\right)\frac{\partial y_i}{\partial\theta_k}\right] \\
&+ I_k\ddot{\theta}_k + k_k(\theta_k - \theta_{k-1}) - k_{k+1}(\theta_{k+1} - \theta_k) \\
&+ \sum_{i=1}^{n} m_i g\frac{\partial y_i}{\partial\theta_k} - \sum_{i=1}^{n} 2m_i\omega x_i\frac{\partial x_i}{\partial\theta_k}. \qquad (7.7)
\end{aligned}
$$

Assuming the mass of each link to be evenly distributed and following a series of substitutions similar to Rezaiifar [28] yields the following expression for the equations of motion of the beam. Note that this expression does not contain any partial derivatives.

$$
\begin{aligned}
\tau_k - \tau_{k+1} &= \sum_{i=k+1}^{n} m_i\left[\sum_{j=1}^{i-1} l_k l_j\left(\sin(\theta_k - \theta_j)\dot{\theta}_j^2 + \cos(\theta_k - \theta_j)\ddot{\theta}_j\right)\right] \\
&+ \frac{m_k}{2}\left[\sum_{j=1}^{k-1} l_k l_j\left(\sin(\theta_k - \theta_j)\dot{\theta}_j^2 + \cos(\theta_k - \theta_j)\ddot{\theta}_j\right)\right] \\
&+ \sum_{i=k+1}^{n}\frac{m_i}{2}\left[l_k l_i\left(\sin(\theta_k - \theta_i)\dot{\theta}_i^2 + \cos(\theta_k - \theta_i)\ddot{\theta}_i\right)\right] \\
&+ \frac{m_k l_k^2}{4}\ddot{\theta}_k + I_k\ddot{\theta}_k \\
&+ k_k(\theta_k - \theta_{k-1}) - k_{k+1}(\theta_{k+1} - \theta_k) + l_k g\cos(\theta_k)\left[\sum_{i=k+1}^{n} m_i + \frac{m_k}{2}\right] \\
&- l_k\omega\sin(\theta_k)\sum_{i=k+1}^{n} 2m_i\left(\sum_{j=1}^{i-1} l_j\cos(\theta_j) + \frac{l_i}{2}\cos(\theta_i)\right) \\
&- l_k\omega\sin(\theta_k)m_k\left(\sum_{j=1}^{i-1} l_j\cos(\theta_j) + \frac{l_k}{2}\cos(\theta_k)\right) \qquad (7.8)
\end{aligned}
$$

for $k = 1, 2, \ldots, n$.

Now for the purpose of understanding the characteristics of the model, several simplifying assumptions are made. They are

- *the number of links is two* ($n = 2$),
- *the beam is in a zero gravity environment* ($g = 0$),
- *the lengths of the two links are equal* ($l_1 = l_2 = l$),
- *the masses of the two links are equal* ($m_1 = m_2 = m$), *and*
- *the two spring constants are equal* ($k_1 = k_2 = k$).

The equations of motion of the simplified two link beam can now be expressed as

$$\begin{aligned} \tau_1 - \tau_2 &= ml^2\left[\frac{4}{3}\ddot{\theta}_1 + \frac{1}{2}\cos(\theta_2 - \theta_1)\ddot{\theta}_2 - \frac{1}{2}\sin(\theta_2 - \theta_1)\dot{\theta}_2^2\right] \\ &+ 2ml^2\omega\sin(\theta_1)\left[\frac{5\cos(\theta_1)}{4} + \frac{\cos(\theta_2)}{2}\right] + k\left(2\theta_1 - \theta_2\right) \end{aligned} \tag{7.9}$$

and

$$\begin{aligned} \tau_2 &= ml^2\left[\frac{1}{2}\cos(\theta_2 - \theta_1)\ddot{\theta}_1 + \frac{1}{3}\ddot{\theta}_2 + \frac{1}{2}\sin(\theta_2 - \theta_1)\dot{\theta}_1^2\right] \\ &+ ml^2\omega\sin(\theta_2)\left[\cos(\theta_1) + \frac{\cos(\theta_2)}{2}\right] + k\left(\theta_2 - \theta_1\right) \end{aligned} \tag{7.10}$$

Finally, these equations can be linearized and put into state-space form to get

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\left(6\omega + \frac{3k}{ml^2}\right) & \left(\frac{54\omega}{7} + \frac{30k}{7ml^2}\right) & 0 & 0 \\ \left(9\omega + \frac{12k}{ml^2}\right) & -\left(\frac{144\omega}{7} + \frac{66k}{7ml^2}\right) & 0 & 0 \end{bmatrix}\begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \\ &+ \frac{3}{7ml^2}\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 4 & -10 \\ -6 & 11 \end{bmatrix}\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}. \end{aligned} \tag{7.11}$$

Now the eigenvalues of this system can be calculated. They are

$$\begin{aligned} \lambda_{1,2} &= \pm j\sqrt{6\omega + \frac{6k}{ml^2}} \\ \lambda_{3,4} &= \pm +- j\sqrt{\frac{144\omega}{7} + \frac{66k}{7ml^2}}. \end{aligned} \tag{7.12}$$

Clearly, as ω increases, the magnitudes of all four eigenvalues also increase. Since the eigenvalue magnitudes correspond to the resonant frequencies of the system, this means that the resonances of the system also increase with ω. Hence, the rotating flexible beam can be though of as an uncertain linear system with its uncertainties parameterized by ω.

Note that the eigenvalues also increase as k increases and decrease as either m or l decrease. These facts concur with the intuition gained for the fixed beam in Chapter 6.

7.2.2 Experimental Model

The model derived in the previous section contains many parameters that are difficult if not impossible to measure. For this reason, it is necessary to experimentally determine a model for the system. Clearly, if the motor velocity ω_m is allowed to vary, the system cannot be modeled as being linear time invariant (LTI). It would be nice to model the system as LTI since there are a wide variety of control tools and algorithms which address LTI systems. One method of solving this problem would be to obtain several models of the system at different fixed motor velocities. Once the motor velocity is fixed, the system is well modeled as LTI and a controller for each motor velocity can be designed using existing techniques. Then, when the motor velocity is allowed to vary slowly, some sort of gain scheduling algorithm can be employed to choose the correct

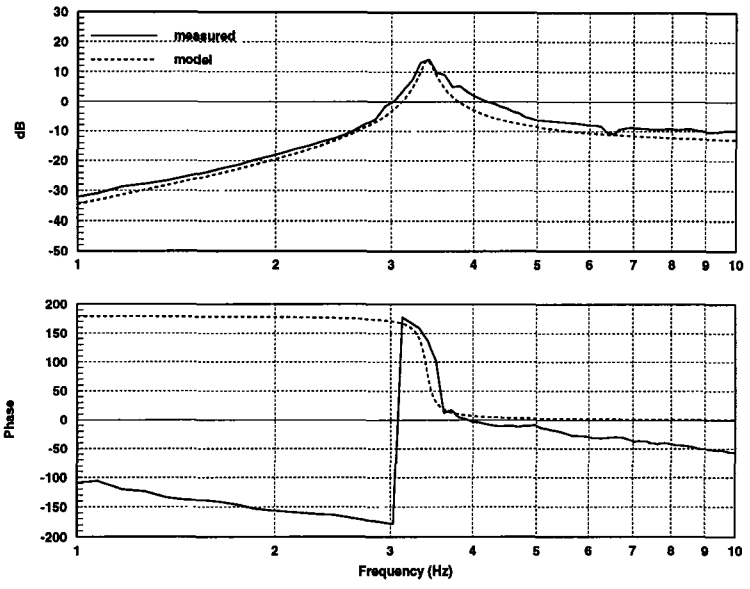

Figure 7.2: Frequency response of rotating beam, $\omega_m = 0$

controller for the system at any point in time. This is the strategy attempted here. Hence, instead of trying to find one model to describe the action of the system over all motor velocities, different models will be found for different fixed motor velocities.

The frequency responses of the system with $\omega_m = 0$ and $\omega_m = 4$ rad/sec are plotted in Figures 7.2 and 7.3. For simplicity, only the first mode of vibration will be addressed here. From the figures, it is clear that the first resonant frequency of the system increases as the motor velocity increases, as predicted by the finite element model. The beam's natural damping also increases with motor velocity. The finite element model does not account for any damping terms so this behavior is not predicted.

At this point, the wavelet based matching pursuits (Section 5.2) or recursive least squares (Section 5.1) could be used to obtain a model for these systems. However, since only one resonance is being considered here models are easily

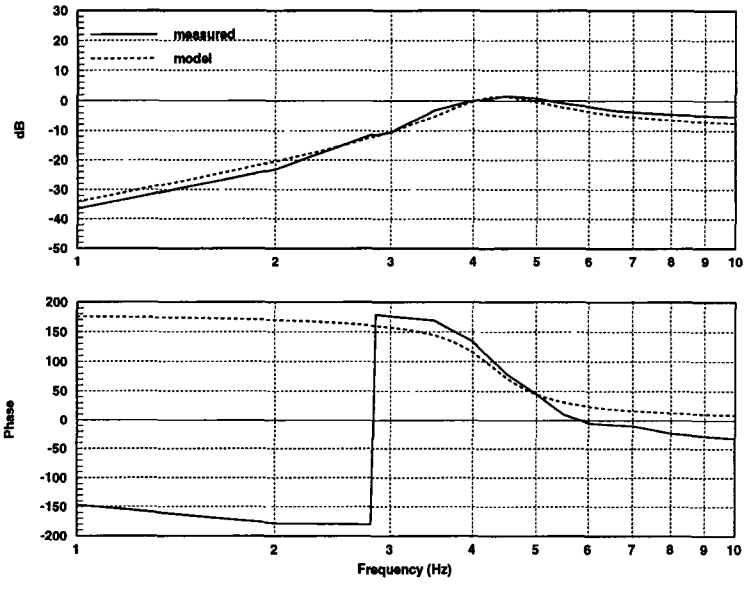

Figure 7.3: Frequency response of rotating beam, $\omega_m = 4$

obtained by hand. The model for $\omega_m = 0$ is

$$T_0(s) = \frac{0.2s^2}{s^2 + 2 \cdot 0.02 \cdot 2\pi \cdot 3.4s + (2\pi \cdot 3.4)^2} \tag{7.13}$$

and the model for $\omega_m = 4$ is

$$T_4(s) = \frac{0.35s^2}{s^2 + 2 \cdot 0.15 \cdot 2\pi \cdot 4.3s + (2\pi \cdot 4.3)^2}. \tag{7.14}$$

The frequency response of each model is plotted with its corresponding experimetnally determined frequency response in Figures 7.2 and 7.3.

7.3 Controller Design

Here the positive position feedback (PPF) controller presented by J. Fanson and T. Caughey [9] is used to design a controller for each motor velocity. Since only one the first mode is being considered, the PPF controller for each motor velocity

will consist of one second order low pass filter. The PPF controller for $w_m = 0$ is

$$H_0(s) = \frac{(3.7 \cdot 2\pi)^2}{s^2 + 2 \cdot 0.2 \cdot 2\pi \cdot 3.7s + (2\pi \cdot 3.7)^2} \tag{7.15}$$

and the PPF controller for $w_m = 4$ is

$$H_4(s) = \frac{(4.9 \cdot 2\pi)^2}{s^2 + 2 \cdot 0.2 \cdot 2\pi \cdot 4.9s + (2\pi \cdot 4.9)^2}. \tag{7.16}$$

The switching point for the gain scheduling controller will be found by trial and error on the actual system.

7.4 Results

To test these controllers, PZT #3 will be used to generate a disturbance, PZT #2 will be used as a sensor, and PZT #1 will be used as the control actuator. A block diagram of the experimental set-up is shown in Figure 7.4. Before the controllers can be tested, however, some measure of system performance must be defined. This performance measure should express the ability of the system to attenuate vibrations of various frequencies at various motor velocities. Hence, the 3 dimensional plot of the absolute value of the output signal versus the disturbance signal frequency and motor velocity will provide a good visual display of controller performance. Further, integrating this surface will provide a scalar measure of performance for the controller. Specifically, disturbance signals will be sine waves ranging in frequency from 1 to 10 Hz in increments of 1 Hz. The motor velocities will range from 0 to 4 rad/sec in increments of 1 rad/sec.

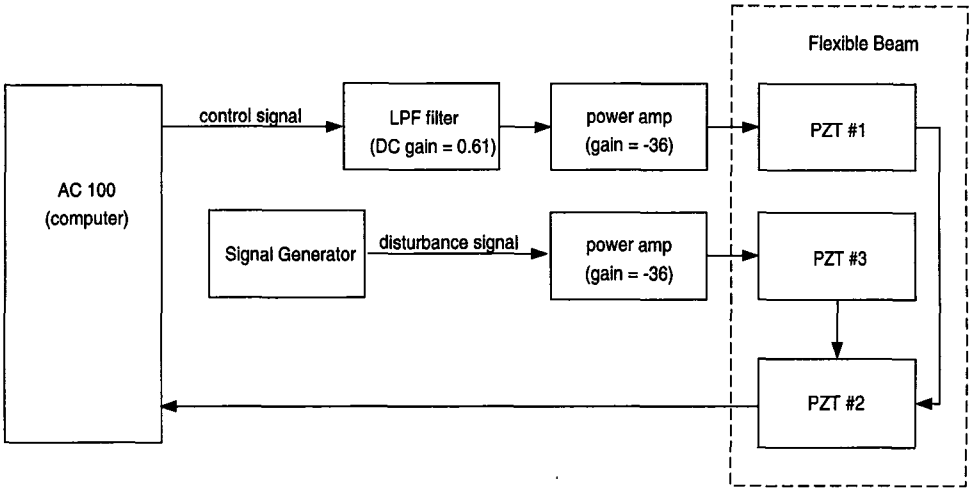

Figure 7.4: Block diagram of experimental set-up

7.4.1 Performance at Fixed Motor Velocities

First, each controller is tested at the fixed motor velocity for which is was designed. These results are plotted in Figures 7.5 and 7.6. The horizontal axis of each plot represents the frequency in Hertz of the 5 volt sine wave sent to PZT #3. The vertical axis represents the magnitude of the resulting signal at the sensor PZT #2 in decibels. Both controllers reject the disturbance signal significantly around the resonance of the system, but do little away from the resonance. This makes sense since PZT actuator is relatively weak and the beam naturally rejects frequencies away from its resonances.

7.4.2 Slowly Varying Motor Velocities

The objective here is to build one controller which can control the system well at all velocities. The first step in doing this is to investigate the performance of the individual controllers over a variety of motor velocities. Figure 7.7 shows the response of the uncontrolled beam over a variety of motor velocities and

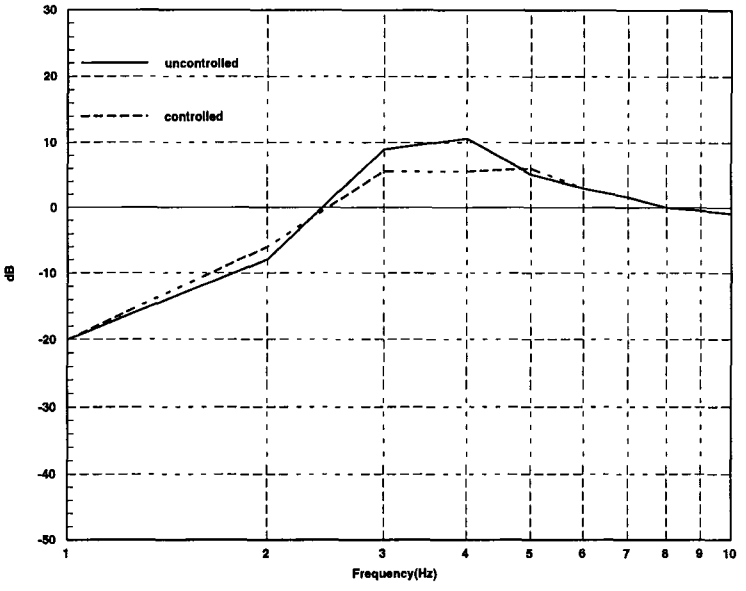

Figure 7.5: Disturbance response of beam at $\omega_m = 0$ rad/sec

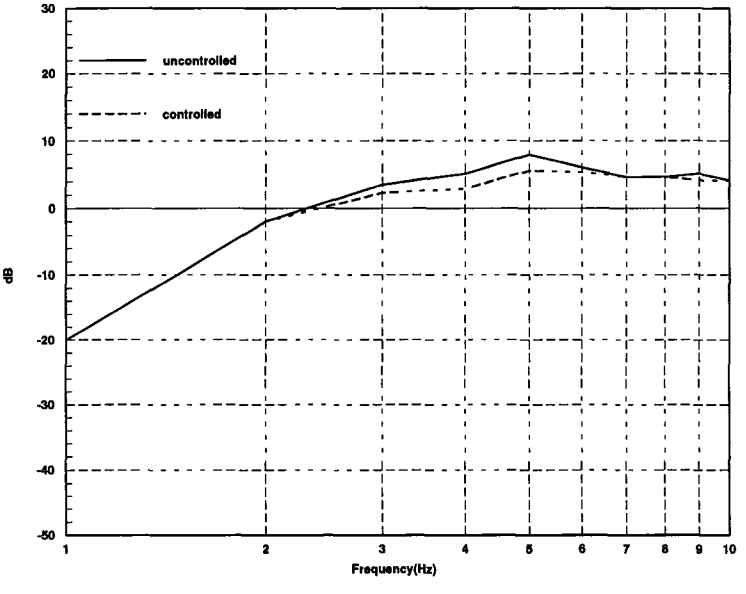

Figure 7.6: Disturbance response of beam at $\omega_m = 4$ rad/sec

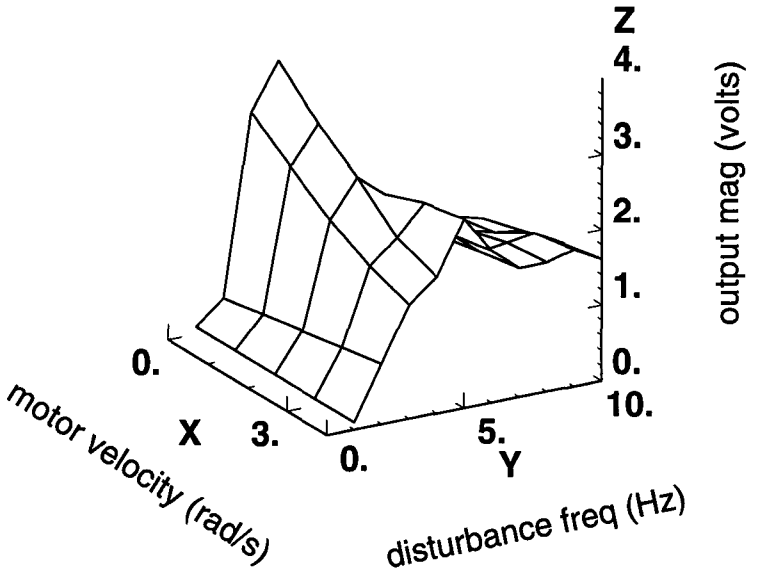

Figure 7.7: Disturbance response of beam using no control

disturbance frequencies. Figures 7.8 and 7.9 depict the performance of the two controllers over the same variety of motor velocities and disturbance frequencies. The X axis represents the motor velocity, ω_m in rad/sec, the Y axis represents the disturbance frequency in Hertz, and the Z axis represents the corresponding magnitude of the sensor voltage in volts. As expected, each controller performs well when operating near its intended motor velocity, but does not perform well away from this velocity. The controller designed for $\omega_m = 4$ is unstable at $\omega_m = 0$ and does a poor job of rejecting disturbance frequencies near 3 Hz at low motor velocities. Likewise, the controller designed for $\omega_m = 0$ does not reject disturbance frequencies near 5 Hz at high motor velocities. As these experiments show, the uncontrolled beam actually responds better over a range of motor velocities than either of the two controllers.

Given these results, the most straightforward way to build one controller to control the system at all motor velocities is to switch between the two individual

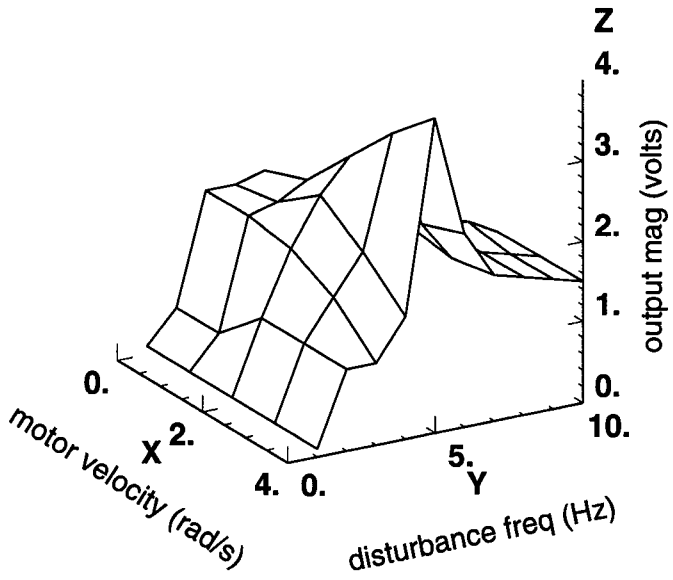

Figure 7.8: Disturbance response of beam using controller $H_0(s)$

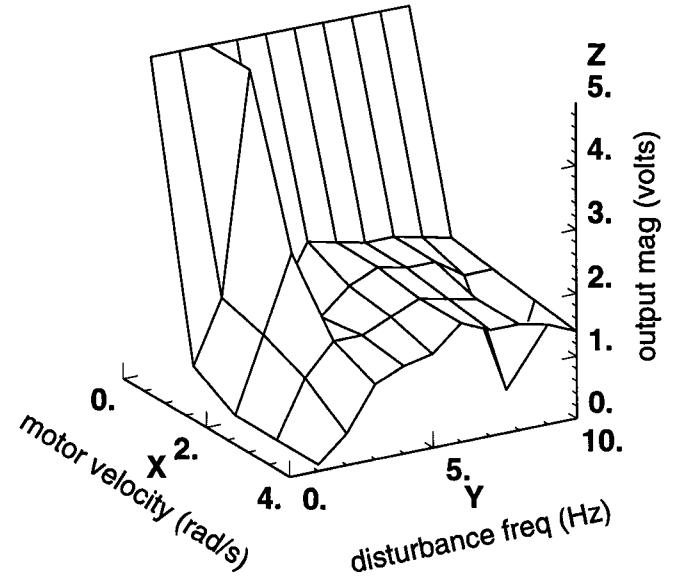

Figure 7.9: Disturbance response of beam using controller $H_4(s)$

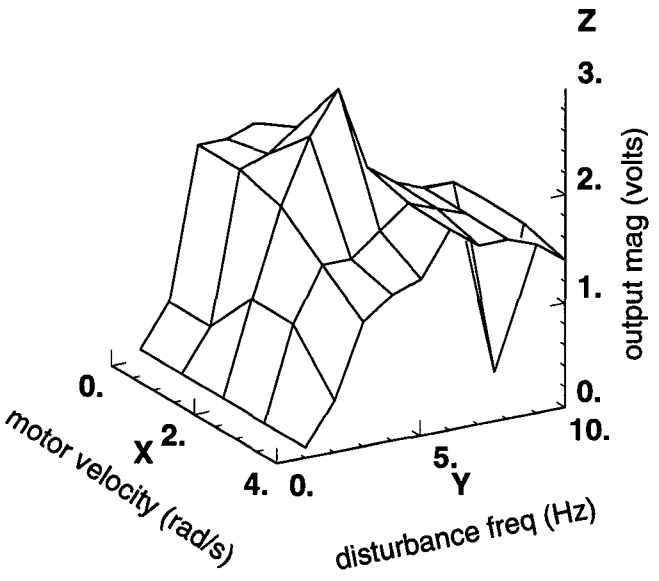

Figure 7.10: Disturbance response of beam using gain scheduling controller

controllers depending on the motor velocity. That is use $H_0(s)$ when $\omega_m \leq 2$ and use $H_4(s)$ when $\omega_m > 2$. This approach is a simple form of gain scheduling. The performance of this gain scheduling controller is depicted in Figure 7.10.

To obtain a scalar performance measure, the surfaces depicted in Figures 7.8, 7.9, and 7.10 can be integrated. These values turn out to be

$$\begin{aligned} J(mboxuncontrolled) &= 74.4 \\ J(H_0(s)) &= 78.4 \\ J(H_4(s)) &= \infty(\text{unstable at } \omega_m = 0) \\ J(gain\ sched) &= 66.3. \end{aligned} \tag{7.17}$$

7.5 Conclusion

The continuously rotating flexible beam changes characteristics as the motor velocity increases. Specifically, the resonant frequencies of the beam increase

with the motor velocity. This presents an interesting control problem. Here standard linear control techniques have been applied to control the system at fixed motor velocities. Then a simple gain scheduling controller was devised to switch between the linear controllers. For slowly varying motor velocity, the gain scheduling controller worked better over the range of possible motor velocities than any one linear controller, hence can be called a success.

The gain scheduling is the simplest method of adressing this problem. Many of the tools that have been developed recently in adaptive and robust control can be applied directly to this problem. This is left for future work.

Chapter 8

Impact of a Flexible Robotic Arm

Many applications of modern robotic technology involve mechanical interaction between a robotic manipulator and some object in the manipulator's workspace. Naturally, this interaction always involves physical contact between the object and the manipulator. In some cases, particularly those involving a manipulator composed of flexible links, the collision between the object and the manipulator can excite an undesirable response, sometimes leading to catastrophic failure. For this reason, a great deal of effort has been concentrated on the problem of modeling the impact forces involved in collisions between flexible objects [33], [34], [4], [19]. Such models can be used to predict the behavior of the system due to impact.

In this chapter, the characteristics of a collision between a single link flexible robotic arm and a fixed object are investigated. The experimental set up is described and a model for the impact is developed using the Hertz law of impact. Finally, an open loop controller composed of a neural network is employed to control the peak magnitude of the impact force between the arm and the fixed object.

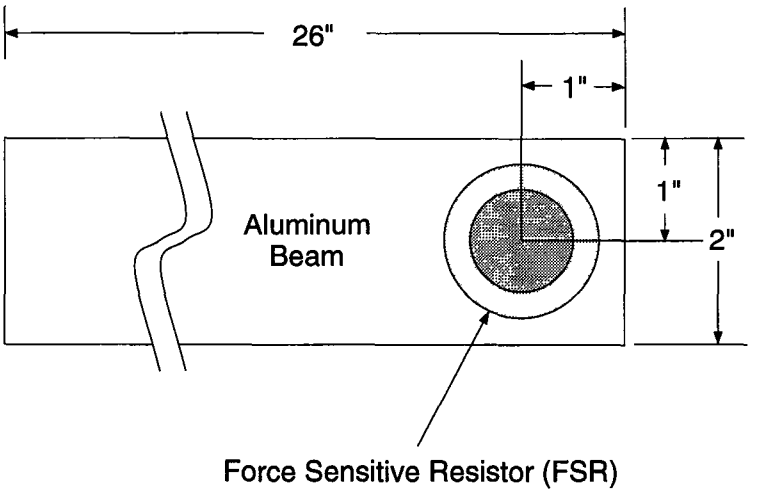

Figure 8.1: Mounting of the FSR (force sensitive resistor)

8.1 Experimental Set Up

As in Chapter 7, a modified version of the flexible arm apparatus designed and constructed by G.H. Frank [10] is used for this experiment. The new hub described in Section 7.1 is not actually necessary to this experiment, however it is used since it is already in place. An aluminum beam measuring 30" $\times$ 2" $\times$ 0.125" is mounted in a vise centered on the vertical axis of the hub. The beam is mounted so that the flat side of the beam (the 30" $\times$ 2" side) is parallel to the jaws of the vise and the long skinny side (the 2" $\times$ 0.125" side) is parallel to the floor and 26" stick out of the vise on one side. A force sensitive resistor (FSR) is mounted on the flat side of the beam with as shown in Figure 8.1. A photograph of the entire set up is shown in Figure 8.2.

The object which the beam collides with consists of a 1" radius aluminum sphere mounted to an adjustable stand. The stand is placed so that the sphere contacts the center of the FSR when the beam collides with the sphere.

The electrical set up is straight forward. The FSR is used as part of a voltage

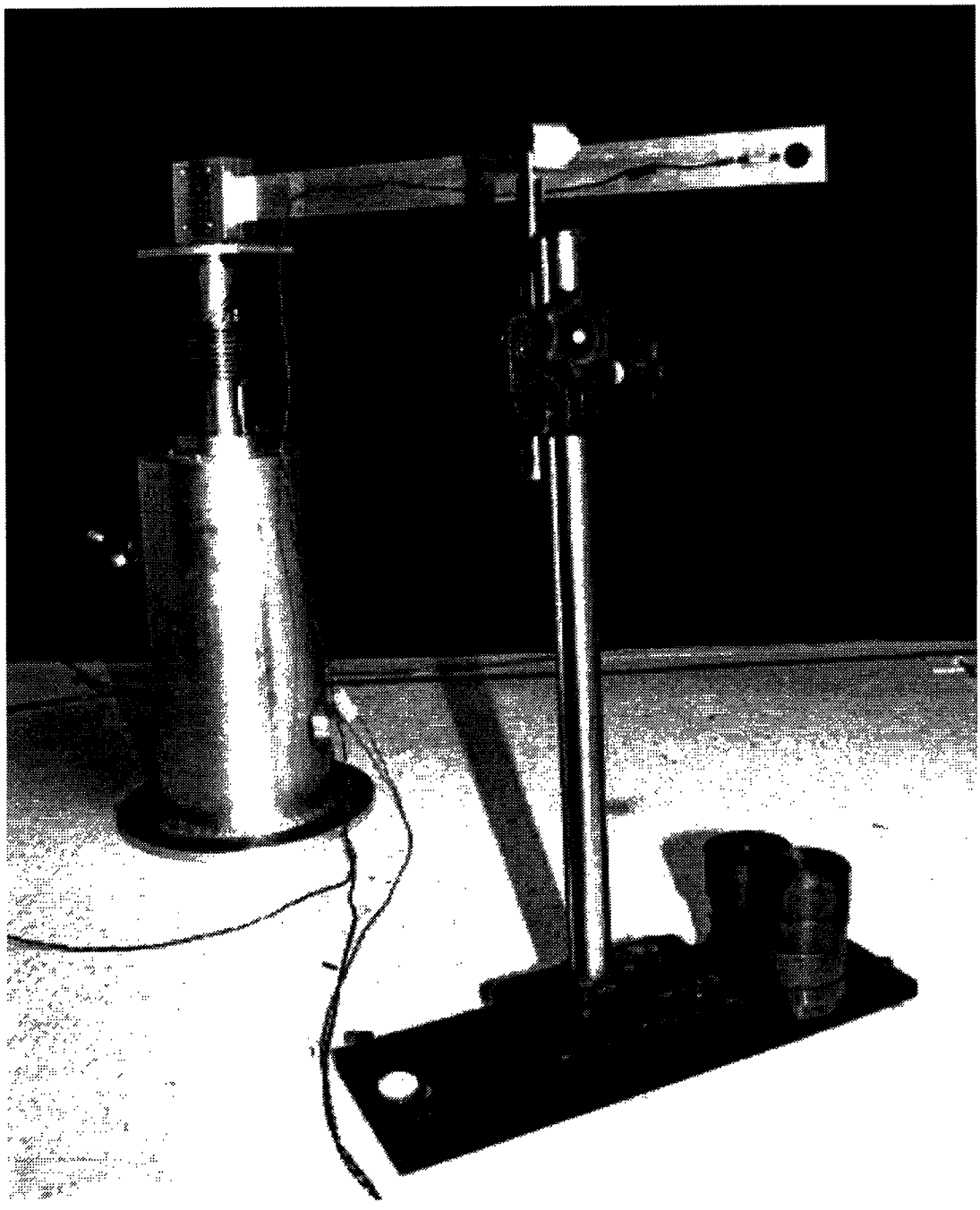

Figure 8.2: Flexible arm set-up

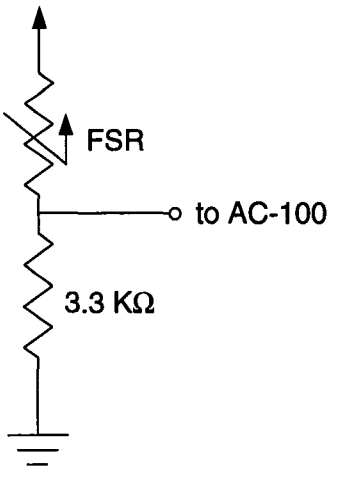

Figure 8.3: Electrical set up of FSR (force sensitive resistor)

divider, the output of which is connected to one of the analog inputs of the AC-100 as shown in Figure 8.3. The translation from the divider voltage to its corresponding force is accomplished using software. The motor control signal is generated by the AC-100 and passed through a power amplifier to the motor.

8.2 Modeling the System

Here a model of the response of the flexible arm apparatus due to impact is developed. First, the finite element model of the fixed beam system is briefly reviewed. This model is then modified to accommodate the rotating hub and an impact force at the tip of the beam. Then, this model is used in combination with Hertz law of impact to determine the impact force between the moving beam and a the fixed aluminum sphere. Finally, the impact force predicted by the model is compared to experimental impact data.

8.2.1 The Finite Element Model

As discussed in Chapters 6 and 7, one popular method of modeling the flexible beam is to approximate the beam as a finite number of rigid bodies connected by rotary joints with torsional springs as depicted in Figure 6.6. This is known as the Galerkin model [2]. The equations of motion for Galerkin model of the cantilever beam derived using the Euler-Lagrange method are

$$\begin{aligned}
\tau_k - \tau_{k+1} &= \sum_{i=k+1}^{n} m_i \left[\sum_{j=1}^{i-1} l_k l_j \left(\sin(\theta_k - \theta_j)\dot{\theta}_j^2 + \cos(\theta_k - \theta_j)\ddot{\theta}_j \right) \right] \\
&+ \frac{m_k}{2} \left[\sum_{j=1}^{k-1} l_k l_j \left(\sin(\theta_k - \theta_j)\dot{\theta}_j^2 + \cos(\theta_k - \theta_j)\ddot{\theta}_j \right) \right] \\
&+ \sum_{i=k+1}^{n} \frac{m_i}{2} \left[l_k l_i \left(\sin(\theta_k - \theta_i)\dot{\theta}_i^2 + \cos(\theta_k - \theta_i)\ddot{\theta}_i \right) \right] \\
&+ \frac{m_k l_k^2}{4}\ddot{\theta}_k + I_k\ddot{\theta}_k + k_k(\theta_k - \theta_{k-1}) - k_{k+1}(\theta_{k+1} - \theta_k) \qquad (8.1)
\end{aligned}$$

for $k = 1, 2, \ldots, n$ where

- n = *the number of links,*
- θ_i = *the slope of the* i*th link,*
- τ_i = *the external moment generated at the* i*th joint,*
- m_i = *the mass of the* i*th link,*
- I_i = *the moment of inertia of the* i*th link about its center of mass,*
- l_i = *the length of the* i*th link, and*
- k_i = *the spring constant at the* i*th joint.*

These equations were derived by Rezaiifar [28]. It should also be noted that these equations are identical to the equations of motion derived in Chapter 7 for

the continuously rotating beam (Equation 7.8) with both the motor velocity, ω, and the acceleration due to gravity, g, set equal to zero.

The model given by Equation 8.1 can be slightly modified to provide an accurate representation of the flexible arm aparatus with a force applied to the tip. In the flexible arm, the first link of the beam is attached via a spring to the hub of the motor. The hub is free to rotate and is subject to an input torque τ_0 from the motor. This adds another degree of freedom to the system, hence the following equation of motion must be added to the model:

$$\tau_0 - \tau_1 = I_0\ddot{\theta}_0 - k_1(\theta_1 - \theta_0) \tag{8.2}$$

where I_0 is the moment of inertia of the hub and θ_0 is the angle of the hub. $\theta_0 = 0$ when the undeflected beam is in line with the positive x axis.

A force applied to the tip of the beam generates a moment about the nth joint equal to $Fl_n\cos(\theta_n)$. Since the system has already been linearized, it has already been assumed that θ_n is small so the effective moment applied to the nth joint by the force F can be approximated as Fl_n. Since there are no PZTs on this beam, the moments applied to all other joints are zero.

Finally, the y position of the tip of the beam will be necessary for the Hertz analysis. Simple trigonometry yields

$$y_{\text{tip}} = \sum_{i=1}^{n} l_i \sin(\theta_0 + \theta_i) \tag{8.3}$$

As stated before, it has already been assumed that $\theta_i, i = 1, 2, \ldots, n$ are small. Furthermore, it is assumed that the impact will occur on the x axis, hence it can be assumed that θ_0 is small for the duration of the impact. Hence, the y position of the tip can be approximated as

$$y_{\text{tip}} = \sum_{i=1}^{n} l_i(\theta_0 + \theta_i). \tag{8.4}$$

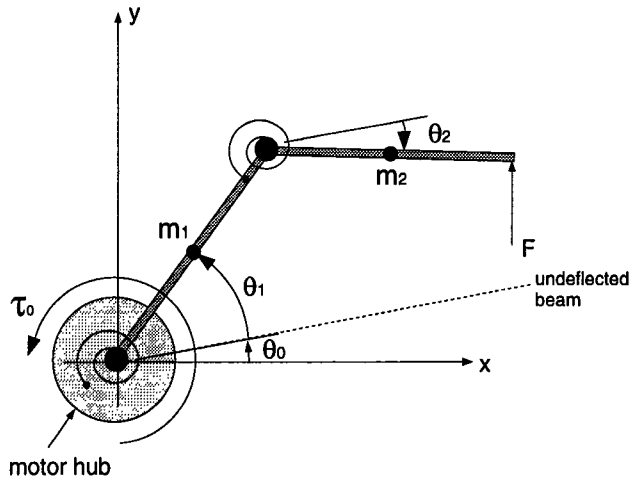

Figure 8.4: Galerkin model of flexible arm

This completes the modifications necessary to the fixed flexible beam model. A diagram of a 2-link version of this model is shown in Figure 8.4. To summarize, the equations of motion for the flexible arm subject to motor torque τ_0 and a force F at the tip are:

$$
\begin{aligned}
\tau_0 &= I_0\ddot{\theta}_0 - k_1(\theta_1 - \theta_0) \\
\tau_k - \tau_{k+1} &= \sum_{i=k+1}^{n} m_i \left[\sum_{j=1}^{i-1} l_k l_j \left(\sin(\theta_k - \theta_j)\dot{\theta}_j^2 + \cos(\theta_k - \theta_j)\ddot{\theta}_j \right) \right] \\
&+ \frac{m_k}{2} \left[\sum_{j=1}^{k-1} l_k l_j \left(\sin(\theta_k - \theta_j)\dot{\theta}_j^2 + \cos(\theta_k - \theta_j)\ddot{\theta}_j \right) \right] \\
&+ \sum_{i=k+1}^{n} \frac{m_i}{2} \left[l_k l_i \left(\sin(\theta_k - \theta_i)\dot{\theta}_i^2 + \cos(\theta_k - \theta_i)\ddot{\theta}_i \right) \right] \\
&+ \frac{m_k l_k^2}{4}\ddot{\theta}_k + I_k\ddot{\theta}_k \\
&+ k_k(\theta_k - \theta_{k-1}) - k_{k+1}(\theta_{k+1} - \theta_k) \\
y_{\text{tip}} &= \sum_{i=1}^{n} l_i(\theta_0 + \theta_i).
\end{aligned}
\tag{8.5}
$$

where $\tau_i = 0$ for $i = 1, 2, \ldots, n-1$ and $\tau_n = Fl_n$.

In his masters thesis, Rezaiifar [28] developed a Mathematica script file called *EqOfMotion.m* which is capable of linearizing the system of equations listed in Equation 8.1 and placing them in state space form. This script file is easily modified to translate the equations of motion for the flexible arm with tip force. The state space equation for the system can then be written as

$$\dot{x} = Ax + Bu, \qquad y_{\text{tip}} = Cx \tag{8.6}$$

where

$$x = \begin{bmatrix} \theta_0 \\ \dot{\theta}_0 \\ \theta_1 \\ \dot{\theta}_1 \\ \vdots \\ \theta_n \\ \dot{\theta}_n \end{bmatrix} \tag{8.7}$$

and

$$u = \begin{bmatrix} \tau_0 \\ F \end{bmatrix}. \tag{8.8}$$

8.2.2 Hertz Law of Impact

Hertz provided the first satisfactory analysis of the stresses at the contact of two elastic bodies by posing the problem as a similar problem in elastostatics [33]. Hertz law of impact is

$$\alpha(t) = K\left[F(t)\right]^{2/3} \tag{8.9}$$

where $\alpha(t)$ is the relative displacement between the bodies at time t, $F(t)$ is the impact force between the bodies, and K is the Hertz constant [32]

$$K = \left(\frac{4}{3}\left[\frac{q}{Q_1 + Q_2\sqrt{A+B}}\right]\right)^{-2/3}. \tag{8.10}$$

The constants q, A, B, Q_1, and Q_2 depend on the local geometry of the region of impact. In this case, a plane (the beam) is colliding with a sphere and the constants become

$$\begin{aligned} q &= \pi^{2/3} \\ A &= \frac{1}{2R_1} \\ B &= \frac{1}{2R_1} \\ Q_1 &= (1-\mu_1^2)/E_1\pi \\ Q_2 &= (1-\mu_2^2)/E_2\pi \end{aligned} \tag{8.11}$$

where R_1 is the radius of the sphere, μ_1, μ_2 is the Poisson ratio of the sphere and the beam, respectively, and E_1, E_2 is the Young's modulus of the sphere and the beam, respectively [17].

Figure 8.5 provides a sketch of the relative displacement between the beam the the sphere during impact. Since the sphere is fixed, the relative displacement $\alpha(t) = -y_{\text{tip}}(t)$. Naturally, y_{tip} is dependent on the unknown impact force, F. Using Hertz law of impact yields the relationship $-y(t) = K\left[F(t)\right]^{2/3}$. Substituting the solution of Equation 8.6 results in the integral equation

$$-C\left[e^{At}x(0) + \int_0^t e^{A(t-\sigma)}B\begin{bmatrix}\tau_0(\sigma)\\ F(\sigma)\end{bmatrix}d\sigma\right] = K[F(t)]^{2/3}. \tag{8.12}$$

Hence, if the state of the beam at the moment just before impact and the motor input τ_0 are known, then the model derived in the previous section can

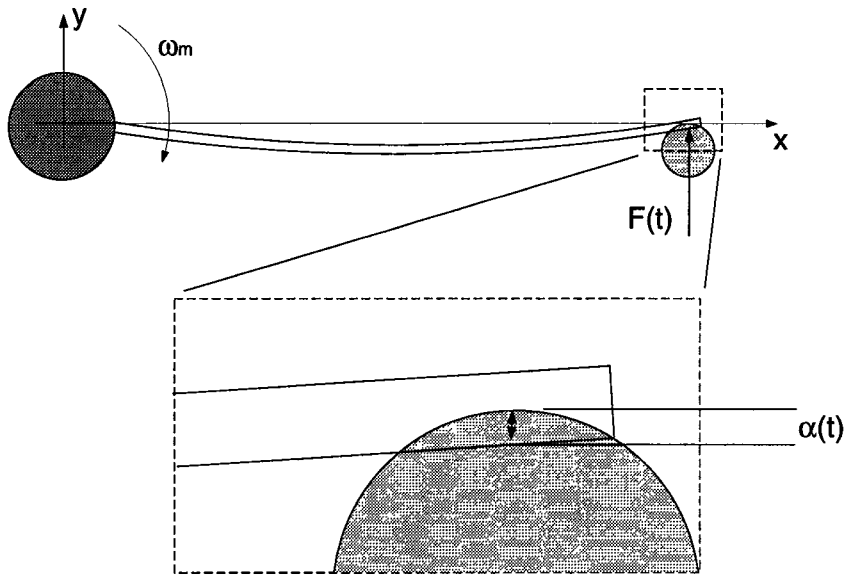

Figure 8.5: Relative displacement between beam and sphere during impact

be used in an iterative approach to numerically solve for the impact force F as a function of time. The algorithm to do this is as follows:

Step 1: *Choose a suitably small time increment T and a suitably large final time t_f.*

Step 2: *Let $F_0(t) = 0$ for all time steps from 0 to t_f.*

Step 3: *Let $i = 1$.*

Step 4: *Solve Equation 8.6 numerically using $F(t) = F_{(i-1)}(t)$ to obtain $y_i(t)$ for all time steps from 0 to t_f.*

Step 5: *Let $F_i(t) = (-y_i(t)/K)^{3/2}$.*

Step 6: *Increment i and repeat step 4 until $F_i(t)$ is sufficiently close to $F_{(i-1)}(t)$.*

This method is known as the method of Picard iteration. If it converges, it will converge to the solution of Equation 8.12, which is the impact force between the

beam and the sphere.

8.2.3 Computing the Impact Force Numerically

The above algorithm can be implemented using MATLAB. First it is necessary to define the model parameters. These are [22]

- $l = 0.6604$ *m (length of beam)*
- $\rho = 0.438$ *kg/m (beam mass per unit length)*
- $EI = 9.62$ *Nm (Youngs modulus* $\times$ *moment of cross section)*
- $I_0 = 1.89 \times 10^{-3}$ *kg/m*^{3} *(moment of inertia of hub)*
- $E_1 = E_2 = 79 \times 10^9$ *Pa (Young's modulus of beam and sphere)*
- $\mu_1 = \mu_2 = 0.33$ *(Poisson ratio of beam and sphere)*
- $R_1 = 0.0254$ *m (radius of sphere)*

The procedure for performing the Picard iterations is straight forward. First the modified *EqOfMotion.m* is used to obtain a state space model for the rotating beam. At the moment of impact, the beam is relaxed, the motor hub is at zero, and the motor hub is moving in a negative direction. This corresponds to the initial condition

$$\begin{aligned} \theta_i(0) &= 0, \quad i = 0, 1, \ldots, n, \\ \dot{\theta}_i(0) &= 0, \quad i = 1, 2, \ldots, n, \\ \dot{\theta}_0(0) &= \omega_m \end{aligned} \tag{8.13}$$

where ω_m is the angular velocity of the hub at the moment of impact. Using this information, the above algorithm can be implemented using MATLAB. Specifically, the script file *picard.m* was created for this purpose. For this case, the algorithm converges. The solution after five iterations is sufficiently accurate for the purposes of this discussion. A plot showing the numerically solved impact forces for various hub velocities is shown in Figure 8.6. A plot showing the experimentally measured impact forces for the same hub velocities is shown in Figure 8.7.

As the figures show, the impact force predicted from the model has a much greater peak magnitude and a much shorter duration than the experimentally measured impact forces. Many factors could cause this error. First, the model assumes the impact occurs between an aluminum beam and an aluminum sphere. In the experiment, an FSR is actually placed between the colliding bodies. The FSR is made up of material which is much softer than aluminum, so this is probably a large source of error in the model. Another possible source of error lies in the assumption that the aluminum sphere is fixed. In fact, the aluminum sphere was mounted to a stand. The stand was as solid as such a device can be expected to be, but still the forces during impact are capable of exciting vibrations in the stand. This phenomenon is not considered in the model. Finally, since the duration of the impact is so short, the time response of the electrical circuit probably distorts the impact signal.

In spite of these problems, the model still predicts the general behavior of the impact reasonably well. The basic shape of the impact curve is the same. Additionally, a smaller motor velocity results in an impact force of smaller peak magnitude and longer duration. This is exactly what the model predicts.

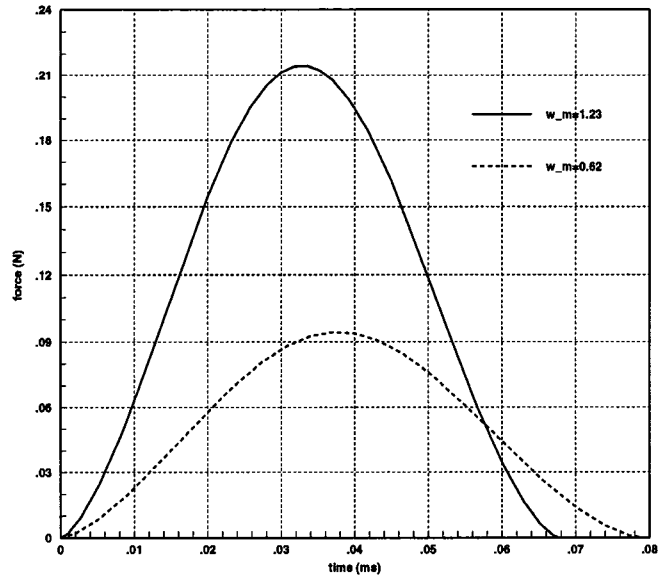

Figure 8.6: Model generated impact force as a function of time

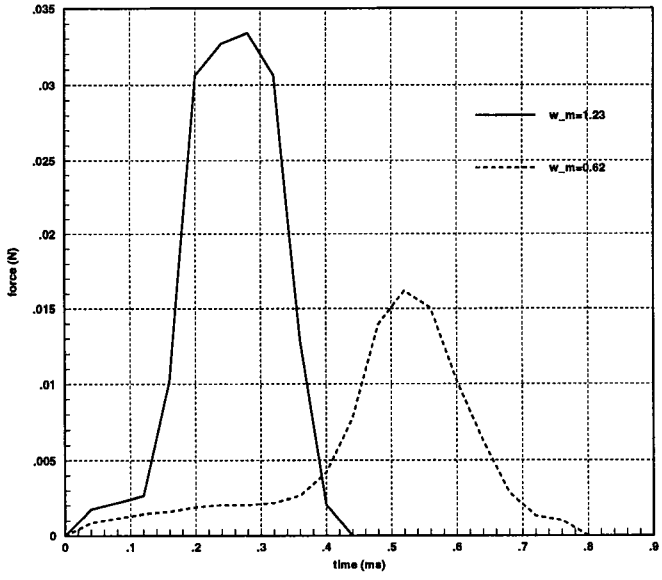

Figure 8.7: Measured impact force as a function of time

8.3 A Neural Network Impact Controller

For some robotic applications, it is desirable to control the peak magnitude of the impact force between the manipulator and some object. Using a robot to hammer a nail into a board, hit a ball, or stamp some design onto a piece of metal are examples of such tasks. Here, a neural network is trained from experimental data to generate an open loop motor control signal which will bring the arm into contact with the sphere to create a specified impact force.

8.3.1 The Input Signal

In order to simplify the problem, a one dimensional input signal will be used. The input signal to the motor will be defined as

$$u_T(t) = \begin{cases} 1 & \text{if } t < T \\ 0 & \text{if } t \geq T \end{cases} \tag{8.14}$$

where $T > 0$. Hence, the input signal $u_T(t)$ is parameterized by a single parameter, the switching time T. Figure 8.8 shows a graph of $u_T(t)$.

Now a neural network can be trained to generate the switching time T from the desired impact force magnitude.

8.3.2 Collecting Data

The aluminum sphere is placed so that the undeflected beam will collide with the sphere when $\theta_0 = 0$. The starting position of the motor hub is $\theta_0 = -180°$. The signal $u_T(t)$ is generated by the AC-100, amplified by a power amp, and fed to the motor. The impact is measured with the FSR mounted to the beam as discussed in Section 8.1.

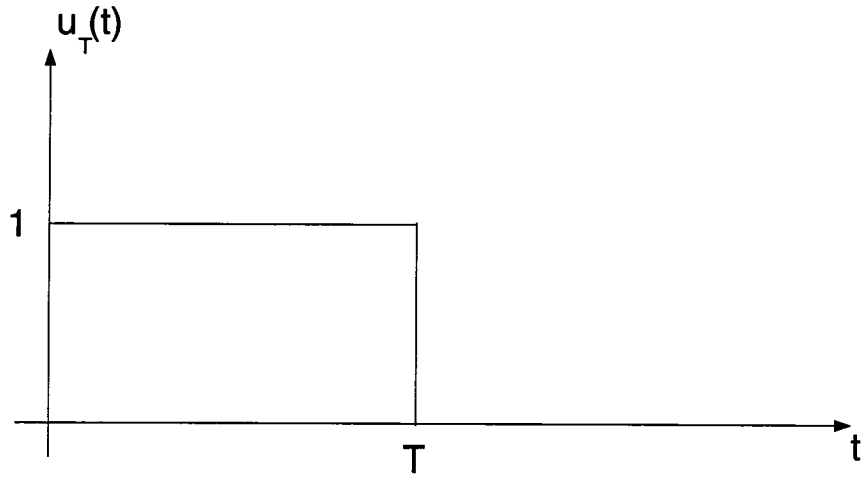

Figure 8.8: Motor control signal parameterized by T

To generate the data necessary to train the neural network, the magnitude of the impact force is measured and recorded for a series of different switching times. The first switching time was arbitrarily chosen to be $T = 0.10$ seconds. The switching time was then incremented by 0.02 seconds until the impact force became large enough to saturate the FSR, about 800 g. This happened at $T = 0.7$ seconds. A plot of impact magnitude versus switching time is shown in Figure 8.9. As shown in the plot, the first non-zero impact magnitude occurred at $T = 0.39$ seconds.

8.3.3 Training the Neural Network

The data obtained in the previous section is used to train a single input, single output backpropagation neural network which will ultimately serve as the controller. The network is chosen to be two layers with ten nodes in each layer. The first layer uses sigmoid transfer functions and the second layer uses linear transfer functions. This structure is capable of approximating any practical function

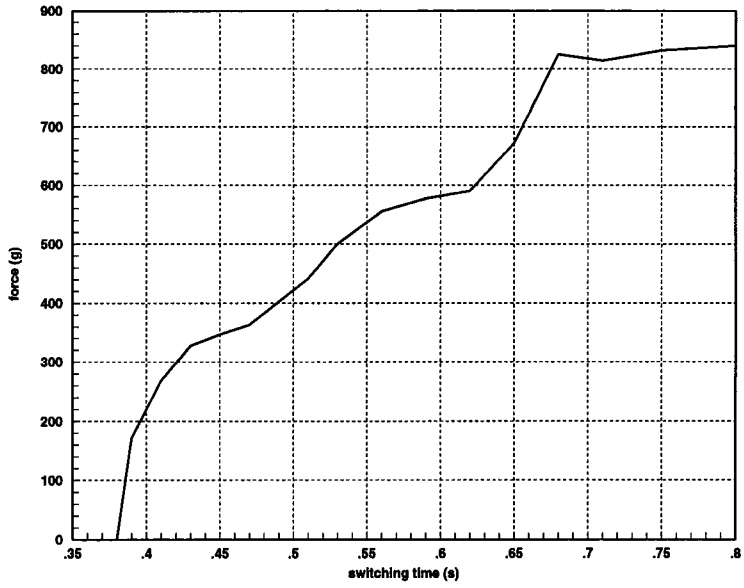

Figure 8.9: Training data for neural network

arbitrarily well [7].

Since the controller will need to specify the switching time as a function of desired impact, the impact forces measured in the previous section are presented to the neural network as inputs and the corresponding switching times are presented as target outputs. The network is trained using the tools available in the MATLAB Neural Network Toolbox. A plot of the input of the trained neural network to its output is shown in Figure 8.10. This compares quite well with the actual data shown in Figure 8.9.

8.3.4 Implementation

The neural network controller is implemented using the AC-100 as shown in the block diagram of Figure 8.11. The system takes a desired impact magnitude as input, then uses the neural network to generate an open loop control signal which moves the arm to impact with the sphere at the desired magnitude. Figure

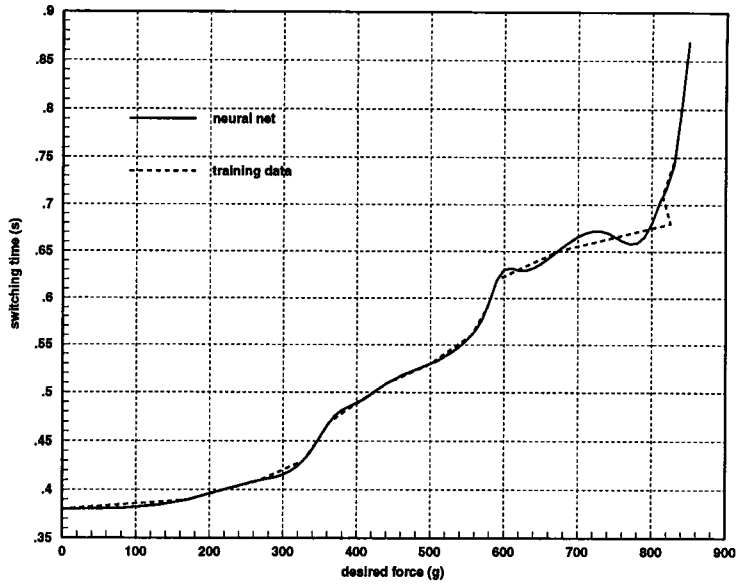

Figure 8.10: Performance of trained neural network

8.12 shows a plot of actual impact versus desired input. If the controller worked perfectly, this would be a straight 45° line. As the graph shows, this controller is not perfect, but it performs reasonably well, particulaly for impact forces above 350 g.

8.4 Conclusion

Models which can predict the general behaviour of the impact forces involved in the collision between flexible bodies have been developed. The next step is to use these models to generate good schemes for controlling the impact dynamics of flexible systems. This topic is the subject of much recent work and will undoubtably be the subject of much future work as well. Here experimental data has been used to create a controller which is capable of controlling the magnitude of impact between a flexible arm and a fixed sphere. This controller

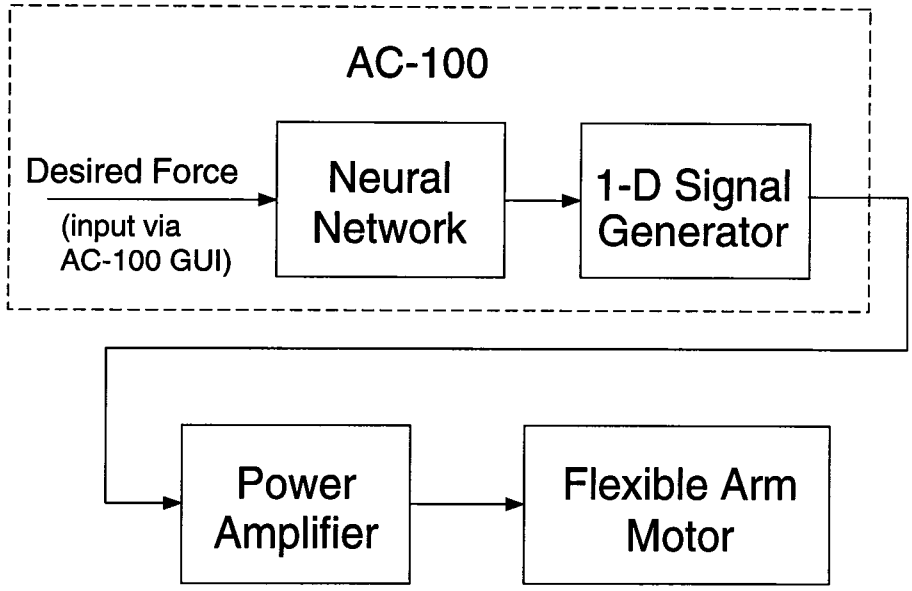

Figure 8.11: Block diagram of neural network controller

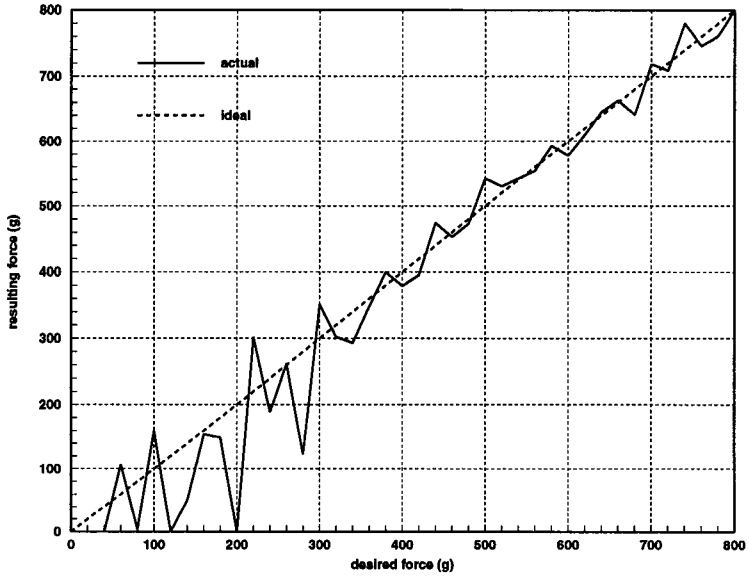

Figure 8.12: Performance of neural network controller

is open loop and its performance is easily diminished by disturbances to the system. Closing the loop is left as a subject for future work.

Chapter 9

Conclusion

This thesis investigates some of the issues involved in controlling systems composed of smart materials. Some theoretical concepts are discussed and some are put into practice. As always, there is much room for future work.

A method of finding a closed loop control law capable of stabilizing an unstable linear plant for every initial condition in its controllable set is developed in Chapters 2 and 3. The availability of such a control law is important, but its implementation is clumsy. An elegant solution to this problem would be a useful tool in the design of controllers for smart structures.

The work presented in Chapters 4 and 5 results in a general purpose test bed for controllers. The AC-100 along with the automatic system identification software provides a means by which new control concepts can be rapidly verified in a real world application. This test bed may not be the subject of future work, but it will almost certainly be used as a research tool for future projects.

The experimental work done in Chapters 6 and 7 validates the use of PZTs as actuators in smart materials as well as providing a demonstration of the test bed developed in the previous chapters. The controllers developed here successfully

damped the vibrations in the flexible beam using one sensor and one actuator. Distributed control using multiple sensors and actuators is the logical next step.

Chapter 8 presents the problem of controlling a flexible robotic manipulator subject to impact forces. A model to predict the impact forces caused in a collision was generated. The general behavior of its predictions compared favorably with experimental results. An open loop controller was developed to control the magnitude of impact. The subject of impact leaves a many unanswered questions. Closing the loop on the impact magnitude controller developed here would be a nice result. Another interesting problem which follows directly from this work is to use PZTs to control vibrations generated in the arm due to impact.

Bibliography

[1] K.J. Astrom and B. Wittenmark. *Adaptive Control.* Addison Wesley, 1989.

[2] J.D. Bartusek. Design and digital signal processor implementation of a controller for flexible structures. Master's thesis, University of Maryland, College Park, College Park, MD 20742, 1990.

[3] G. Blankenship. Class notes. ENEE 769C: Adaptive Control, 1993.

[4] B. Chapnik, G. Heppler, and J. Aplevich. Modeling impact on a one-link flexible robotic arm. *IEEE Transactions on Robotics and Automation*, 7:666–672, 1991.

[5] K.J. Cios and N. Liu. A machine learning method for generation of a neural network architecture: A continuous ID3 algorithm. *IEEE Transactions on Neural Networks*, 3(2):280–291, 1992.

[6] E.F. Crawley and J. de Luis. Use of piezoelectric actuators as elements of intelligent structures. *AIAA Journal*, 25(10):1373–1385, 1987.

[7] H. Demuth and M. Beale. *Neural Network Toolbox Users Guide.* The Math Works, Inc., Natick, Massachusetts, 1992.

[8] J.J. Dosch, D.J. Inman, and E. Garcia. A self-sensing piezoelectric actuator for collocated control. *Journal of Intelligent Materials, Systems, and Structures*, 3:166–185, January 1992.

[9] J.L. Fanson and T.K. Caughey. Positive position feedback control for large space structures. In *Proceedings of the 28th AIAA/ASME/ASC/AHS Structures Structural Dynamics and Materials Conference*, volume 2, pages 588–598, 1987.

[10] G.H. Frank. Design and real time control of a flexible arm. Master's thesis, University of Maryland, College Park, 1986.

[11] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[12] G.J. Gibbs and C.R. Fuller. Experiments on active control of vibrational power flow using piezoceramic actuators/sensors. *AIAA Journal*, 30(2):457–463, February 1992.

[13] P. Gutman and P.Hagander. A new design of constrained controllers for linear systems. *IEEE Transactions on Automatic Control*, AC-30(1):22–33, January 1985.

[14] P.J. Gyugyi and G. Franklin. Multivariable integral control with input constraints. In *Proceedings of the 32nd Conference on Decision and Control*, volume 3, pages 2505–2510, 1993.

[15] D. Hammerstrom. Neural networks at work. *IEEE Spectrum*, pages 26–32, June 1993.

[16] Integrated Systems, Inc., 3260 Jay St, Santa Clara, CA 95054. *Numerical Algorithm Reference*, February 1992.

[17] K. Johnson. *Contact Mechanics.* Cambridge University Press, 1985.

[18] L.G. Kraft and D. Dietz. Time optimal control using CMAC neural networks. In *Proceedings of the American Control Conference*, pages 2943–2944, June 1994.

[19] E. H. Lee. The impact of a mass striking a beam. *Journal of Applied Mechanics*, 7:A129–A138, 1940.

[20] L. Ljung. *System Identification-Theory for the User.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[21] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *Preprint*, 1992.

[22] C.L. Mantell. *Engineering Materials Handbook*, pages 6–6,33–9. McGraw-Hill Book Company, Inc., 1958.

[23] Y.C. Pati. *Wavelets and Time-Frequency Methods in Linear Systems and Neural Networks.* PhD thesis, University of Maryland, College Park, 1992.

[24] Y.C. Pati, R. Rezaiifar, P.S. Krishnaprasad, and W.P. Dayawansa. A fast recursive algorithm for system identification and model reduction using rational wavelets. In *Proc. of the 27th Annual Asilomar Conference on Signals Systems and Computers*, pages 35–39, November 1993.

[25] T. Pecsvaradi and K. S. Narendra. Reachable sets for linear dynamical systems. *Information and Control*, 19:319–344, 1971.

[26] C.L. Phillips and H.T. Nagle. *Digital Control System Analysis and Design.* Prentice Hall, Englewood Cliffs, NJ 07632, 1990.

[27] F. Pourki. Distributed controllers for flexible structures using piezo-electric actuator/sensors. In *Proceedings of the 32nd Conference on Decision and Control*, pages 1367–1369, December 1993.

[28] R. Rezaiifar. Smart structures and wavelet based system identification. Master's thesis, University of Maryland, College Park, 1993.

[29] W. J. Rugh. *Linear System Theory.* Prentice Hall, 1993.

[30] J. Schneider. High dimension action spaces in robot skill learning. In *Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1272–1278, 1994.

[31] A. Tits. Class notes. ENEE 664: Optimal Control, 1993.

[32] R. Venkataraman. A hybrid actuator. Master's thesis, University of Maryland, College Park, 1995.

[33] Q. Wei. *Modeling and Control of Dynamical Effects due to Impact on Flexible Structures.* PhD thesis, University of Maryland, College Park, 1994.

[34] A. Yigit, A. Ulsoy, and R. Scott. Dynamics of a radially rotating beam with impact. *ASME Journal of Vibration and Acoustics*, 112:1391–1413, 1990.