ACTIVE DAMAGE CONTROL USING ARTIFICIAL INTELLIGENCE: INITIAL STUDIES INTO IDENTIFICATION AND MITIGATION

by

David H. Kiel

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Mechanical Engineering

APPROVED:

[illegible]

Dr. Harry H. Robertshaw, Chairman

[illegible]

Dr. Craig A. Rogers

[illegible]

Dr. Alfred L. Wicks

June, 1993
Blacksburg, Virginia

c.2

LV
5655
V855
1993
K554
c.2

ACTIVE DAMAGE CONTROL USING ARTIFICIAL INTELLIGENCE: INITIAL STUDIES INTO IDENTIFICATION AND MITIGATION

by

David Harlan Kiel

Committee Chairman: Dr. Harry H. Robertshaw

Mechanical Engineering

Abstract

This thesis presents an initial investigation into Active Damage Control (ADC) using Artificial Intelligence (AI). AI can alleviate the sometimes complicated task of modelling the system and also provides an adaptable solution process. The two research areas of ADC, damage identification and damage control, are studied in separate investigations.

An AI technique called "rule induction" is used for the damage identification study. Velocity data from three plates (one without damage, one with damage at the center, and one with damage at the edge) are acquired using a laser data acquisition system. A set of rules is then induced from these data which accurately identifies which plates have damage and where the damage is located. With regard to the damage control, a real-time, machine-learning technique called "BOXES" is used to locally control the vibration of various systems by identifying their vibrational patterns. Using this technique, it is shown that the computer successfully learns an effective control law for various simulations using its trials and failures as the only learning information. It is also seen that the learning algorithm is somewhat less effective when experimentally applying this method to a pin-pin, aluminum beam. A discussion of possible improvements are presented in the future work section.

Acknowledgements

I would like to express my sincere appreciation and gratitude to my advisor, Dr. Harry H. Robertshaw. His encouragement, motivation, and technical expertise deserves far more than this "traditional" acknowledgement. In addition, I am indebted to committee member Dr. Craig Rogers for providing me with the useful facilities at the Center for Intelligent Material Systems and Structure (CIMSS) with which this research was developed. The insight and helpful suggestions of committee member Dr. Alfred Wicks was also instrumental to these investigations. This research was funded by the Army Research Office grant DAAL03-92-G-0180 for research into Active Damage Control.

Several individuals at CIMSS and in the Smart Structures Lab also deserve mention for their much-appreciated assistance and friendship. Craig Dempsey, Judi Lynch, John Walker, Tony Ackerman, Steve Stein, Mahesh Subramaniam, Mark Lin, and Francesco Betti are only a few of the individuals who provided moral support and laughter which made these last 15 months so enjoyable. Many kudos goes to Yu Liu at the Modal Lab for her assistance in acquiring the data with the laser system. In addition, Dan Cole, Will Saunders, and Gary Fagan all provided priceless expertise regarding the transputer boards and the "BOXES" experiment in general. Special thanks go to my roommate, Doug, and my part-time roommate, Troy, for all their support. Without these individuals, this research could not have been accomplished.

I would like to express my eternal appreciation to my parents for their unyielding love and support. Always taking an interest in my research, my parents gave me the confidence and desire to achieve. Lastly and most importantly I would like to thank my best friend, Pamela Laine Hopkins. Although 3500 miles away for the greater portion of my graduate studies, she provided me with the inspiration and love I hope to have for the rest of my life.

In the immortal words of the late Roger Miller, "All you gotta do is set your mind to it, and knuckle down, and buckle down, and DO IT, DO IT, DO IT!"

Contents

Chapter 1: Introduction 1

1.2 Active Damage Control Literature Review 3

1.2.1 Damage Identification Literature Review 3

1.2.2 Damage Control Literature Review 6

1.2 Artificial Intelligence Literature Review 10

1.4 Scope and Objectives 17

1.4 Outline of the Thesis 19

Chapter 2: Study of Damage Identification Using Rule Induction 20

2.1 Experimental Setup and Data Acquisition 21

2.2 KnowledgeSEEKER 33

2.2.1 KnowledgeSEEKER Background 34

2.2.2 KnowledgeSEEKER Specifications 38

2.3 Resulting Rules and Evaluation 39

Chapter 3: Study of Damage Control Using Rule Induction 53

3.1 'BOXES' Technique 57

3.1.1 "BOXES" Background 57

3.1.2 "BOXES" Extensions 60

3.1.3 Discretization Threshholds 63

3.2 Machine Learning Algorithm 64

3.3 Matlab Simulations 65

3.3.1 Solution Structure 66

3.3.2 Simulation Results and Evaluation 71

3.4 Beam Experiment 80

3.4.1 Solution Structure 85

3.4.2 Experimental Results and Evaluation 88

Chapter 4: Conclusions and Recommendations for Future Work 91

4.1 Damage Identification 91

4.2 Damage Control 94

References 97

Appendix A: Damage Identification Rules Generated from Unmodified Decision Tree . 101

Appendix B: Damage Identification Rules Generated from Modified Decision Tree . . . 116

Appendix C: "BOXES" Codes Used to Simulate and Control Matlab Simulations 127

Appendix D: "BOXES" T800 and T222 Codes Used for Beam Experiment 150

VITA 158

List of Figures

Fig. No.	Title	Page
Figure 1.1	Faculties of Artificial Intelligence	12
Figure 2.1	Plate Definitions and Dimensions	22
Figure 2.2	Plate and Associated Data Point Numbering System	24
Figure 2.3	Frequency Response Function for Non-damaged Plate	26
Figure 2.4	Frequency Response Function for Center-damaged Plate	27
Figure 2.5	Frequency Response Function for Edge-damaged Plate	28
Figure 2.6	Laser-produced Mode Shape for Non-damaged Plate	29
Figure 2.7	Laser-produced Mode Shape for Center-damaged Plate	30
Figure 2.8	Laser-produced Mode Shape for Edge-damaged Plate	31
Figure 2.9	Flow Diagram of Data Acquisition System	32
Figure 2.10	Decision Tree Describing Plate Data (Unmodified)	40
Figure 2.11	Decision Tree Describing Plate Data -- Continued	41
Figure 2.12	Decision Tree Describing Plate Data -- Continued	42
Figure 2.13	Decision Tree Describing Plate Data -- Continued	43
Figure 2.14	Decision Tree Describing Plate Data -- Continued	44

Figure 2.15	Decision Tree Describing Plate Data -- Continued	45
Figure 2.16	Decision Tree Describing Plate Data (Modified to Identify "Sensitive" Areas)	48
Figure 2.17	Decision Tree Describing Plate Data -- Continued	49
Figure 2.18	Decision Tree Describing Plate Data -- Continued	50
Figure 2.19	"Sensitive" Areas on Plates as Described by Modified Decision Tree	51
Figure 3.1	Pole and Cart System used in Original "BOXES" Experiment	59
Figure 3.2	State Space Representation Taken from Chambers and Michie's "'BOXES' as a Model of Pattern Formation"	61
Figure 3.3	Spring-Mass Systems modeled for Theoretical Investigation: a) 1-Mass Simulation (damper on sinusoidal disturbance), b) 2-Mass Simulation (dampers on sinusoidal disturbance)	67
Figure 3.4	Visualization of "BOXES" Matrix used in Simulations	68
Figure 3.5	"BOXES" Controlled Response for 1-Mass System Simulation with Impulse Disturbance	72
Figure 3.6	"BOXES" Controlled Response for 1-Mass System Simulation with Step Disturbance	73

Figure 3.7 "BOXES" Controlled Response for 1-Mass System Simulation with Sinusoidal Disturbance 74

Figure 3.8 "BOXES" Controlled Response for 2-Mass System Simulation with Impulse Disturbance 76

Figure 3.9 "BOXES" Controlled Response for 2-Mass System Simulation with Step Disturbance 77

Figure 3.10 "BOXES" Controlled Response for 2-Mass System Simulation with Sinusoidal Disturbance 78

Figure 3.11 Flow Diagram for "BOXES" Learning System 82

Figure 3.12 Control System Architecture 83

Figure 3.13 "BOXES" Controlled Response for Aluminum Beam with Sinusoidal Disturbance: a) Uncontrolled Response, b) Test Run 1, c) Test Run 2, d) Test Run 3 89

Chapter 1

Introduction

Active Damage Control is a structural control process used for alleviating high strains in structures by active control techniques. This method of structural control has relied on the complicated, and sometimes undeterminable, modelling of the structure for successful damage control. Artificial intelligence techniques for active damage control could both alleviate the complicated task of modelling a system and also provide for a solution process which can adapt to the problem of interest.

The concept of active damage control can be logically split into two main categories: damage identification and damage control. Basic research issues in damage identification consist of not only identifying whether the structure has damage, but also locating this damage. With regard to damage control, the research question is to prevent more damage by controlling the vibration of the entire structure or simply the local area where the damage exists.

This theory of applying artificial intelligence to control tasks is not a new idea. For decades, researchers and scientists have postulated for decades that there are many vital advantages to having a computer "learn". Machine learning alleviates the sometimes impossible mathematical task of modelling a complex system; e.g., time-varying systems, combinatorial control problems, etc. (Pospelov and Pospelov, 1979). This method of machine learning needs not define the system but only to identify it, and does this by way of an empirical relationship between the inputs and outputs.

In this thesis, the artificial intelligence technique known as inductive learning is used as a tool for active damage control. Inductive learning, also known as learning from examples, uses a set of positive and negative samples to induce general concepts that describe all of the positive examples and none of the negative examples. More formally, a computer learning from examples uses sample data to generate an up-dated basis for improved classification of subsequent data from the same source (Michie, 1992).

The purpose of this research is to prove or disprove the viability of using this theory of inductive learning in active damage control. This will consist of not only identifying structural damage, but determining the location of the damage. In addition, this method of learning from examples is tested as a control mechanism for a vibrating structure. The author feels that the successful use of this artificial intelligence technique in solving these two separate but related problems will lead to future optimism in applying artificial intelligence to active damage control.

The following sections contain a brief background of what has been done in the fields of damage identification and damage control. Also, a brief overview of inductive learning and artificial intelligence in general is presented.

1.1 Active Damage Control Literature Review

Scientists have been developing methods of damage control for many years. The most notable of these methods are mechanical treatment of the structure, surface treatment of the structure, adaptive damage control, and active damage control (Li, 1992). The focus of this thesis is limited to the area of active damage control.

As mentioned before, active damage control is active control techniques used to alleviate high stresses in structures (Rogers et al., 1991). This technique inherently contains two distinct problems with two distinct backgrounds. The first is the problem of damage identification and the second is the problem of damage control. For organizational considerations, the literature reviews are discussed in separate sections.

1.1.1 Damage Identification Literature Review

Many techniques have been under investigation over the last few years to identify if and where damage exists in a variety of different structures. Natural frequencies, mode shapes, damping ratios, and correlation coefficients are only a few of the methods presently being used for identifying and quantifying damage in structures. The following literature reviews are by no means supposed to be, or even intended to be, a complete grouping of works on the subject. The select list of works described here is simply intended to show the logical progression of the technology.

Methods of non-destructive testing for damage identification first began in the late seventies. Research dealt with the location of defects in structures using the measurements of natural frequencies (Cawley and Adams, 1979). In this case, the authors made measurements at a single point on an aluminum plate to not only detect and locate damage but also to roughly quantify the damage. An unusual result from this research states that symmetrical structure shapes cause some problems in locating the damage. This characteristic not only identifies damage at the correct location but also symmetrically across the major axis. On the other hand, if the structure is asymmetrical, the damage site is uniquely defined at the correct location. It should be noted that this research was performed theoretically using finite element analysis.

Wolff and Richardson (1989) performed a similar test on a plate experimentally using an impact hammer. The purpose of this research was to identify the correlation between the physical changes and modal parameter changes of a damaged structure. In this case, frequency response functions were used to detect bolt tightness between a flat plate structure and a rib stiffener. It should be noted that the modal frequencies themselves were not enough to detect damage. The authors used curve fitters of the experimental FRF data and super-imposed the plotted mode shapes. Again, the frequency response function approach to modal testing sufficiently predicted structure damage and its location.

The use of modal analysis for crack identification was performed in both a theoretical and experimental capacity in 1990 (Gomes and Silva, 1990). The experimental procedure was performed on a free-free steel beam for several crack locations and depths. Although the method was successful, a very interesting result was identified. This result is that cracks with a small depth tended to have little or no influence on the natural frequencies relative to an uncracked beam behavior.

Similar to the testbed used for this thesis, Richardson and Mannan (1992) tested an aluminum plate with a small saw cut at one edge acting as the damage. Once again, a modal approach was taken to successfully identify and locate the damage in the structure. The method used here was to locate the damage by comparing a set of modal data from the undamaged structure and the frequencies of its modes after the damage was imposed. Again in this paper the author admits that small amounts of damage in the structure may not affect a structure's dynamics, and therefore may not be detectable.

In an unpublished paper, the author and Liu showed that certain types of crack damage to a plate can be detected using signal processing techniques (Kiel and Liu, 1992). A correlation coefficient analysis was performed on three separate composite plates. One contained no damage, one contained a crack in the center, and the third contained a crack at the edge. Velocity data gathered using a laser beam was cross-correlated in an attempt to distinguish which plates contained the damage. Results showed that there was a noticeable difference between a plate with no damage and a plate with damage located at the edge. On the other hand, less of a distinction was noticed when the crack was located at the center of the plate. Again the symmetry of the damage with respect to the plate played a large part in the contrasting results.

1.1.2 Damage Control Literature Review

The second, and just as important, problem to be addressed in the area of active damage control is exactly how to control the damage. That is to say, once the damage has occurred in a structure (whether it be a crack, delamination, etc.), how can the vibration of this area be suppressed in order to control the damage from spreading. In the crudest sense, there are basically only two ways to control damage in a structure: globally and locally. Methods of globally controlling structural vibration, i.e., the entire structure, have been around for many decades and are discussed in detail in any classical and modern control systems book. The more direct method, and the topic of this thesis, is to control the vibration to the structure at the damaged location only (local control).

One of the new and fast-developing fields is an exciting material concept known as "intelligent material systems." An intelligent material system is a hybrid material system with integrated sensors, control processors, and induced strain actuators used to provide computational/control capabilities. This method combines sensors, actuators, and computer technology to actively, and non-destructively, decrease the stress and strain field at the crucial, damaged location (Rogers et. al., 1991).

Many techniques of using induced strain actuators have been investigated including Shape Memory Alloys (SMAs) and piezoelectric, electrostrictive, and magnestrictive actuators (Rogers, Liang, and Li, 1991). In the aforementioned paper, these techniques were shown to successfully reduce stress and strain at a crack tip and to increase fatigue life span of engineering structures.

Along these lines, smart material actuators have also been shown to control the growth of delaminations in composite structures (Hanagud et. al., 1992). Delaminations are essentially interlaminar ply separations and are some of the most commonly observed damages. Piezoelectric sensors and actuators were used to reduce axial stresses in a beam, thereby reducing the damage growth rate.

Modal control algorithms based on a Positive Position Feedback (PPF) strategy have also been used for vibration control. This method was originally suggested by Caughey and Goh (Caughey and Goh, 1987) as an alternative to the standard direct velocity feedback. In short, this method feeds back generalized modal displacements to successfully accomplish vibration suppression. The conventional modal controllers use negative feedback from both the modal position and velocity. Second-order filters are used to feed back the position signals.

Baz, Poh and Fedor have since used the Positive Position Feedback strategy to dampen out multi-modes of vibration of a simple cantilevered beam (Baz et. al., 1989). Slight variations in the method (such as first order filters instead of second order) enable a simplification of the controller design and the prevention of any steady-state errors. Results from the flexible cantilevered beam show that uniform damping can be achieved for all the controlled modes. In addition, the structural vibration was effectively suppressed using the PPF method.

A similar experiment using the PPF strategy has also been performed on a flexible box-type structure analogous to large space structures (Poh and Baz, 1990). Again the position signals were fed back through a first order filter. Once again, the results indicated the effectiveness of using positive position feedback to suppress structural vibration.

A final investigation which uses the PPF strategy for vibrational control was performed by Fanson and Caughey (Fanson and Caughey, 1990). The experimental testbed was a thin flexible beam made of aluminum. The first three modes were controlled to suppress the vibration of the simply supported beam. Results show that the dynamic response of all the controlled modes was "significantly reduced." It should be noted that one PPF filter was needed for **each** modal suppression unless the modes are clustered.

Another method of local control which has been proven effective in vibration suppression is called Power Flow or Wave Junction control. This method controls the vibration at a damage location by using concepts of disturbance propagation and reflection. The response of a flexible structure to a locally applied force is viewed as waves of a travelling elastic disturbance. Conceptually, active controllers are designed as wave absorbers to "shunt" energy to non-critical locations, away from the damage sites. Various controllers or compensators have been tested to evaluate the energy absorbing and reflecting characteristics (for example, see Aubrun, 1980, von Flotow, 1986, Signorelli, 1988).

In one investigation by von Flotow (von Flotow and Schafer, 1986), wave-absorbing controllers are used to absorb wave energy at the end of a flexible stainless steel beam. The wave absorbers are applied to the free end of a hanging, clamped-free beam and local deflections are fed back to control force and moment. In the study, various sensor and actuator locations were studied to identify optimum locations. Both synthesized and ideal compensators were compared with regard to effectiveness. In addition to realizing dampened vibration at the targeted location, a connection was identified between power flow and "velocity feedback schemes with frequency dependent gains." Some compensators exhibited instabilities at low frequencies.

Various techniques for deriving compensators for wave absorbers have also been studied (Miller et. al., 1989). In these cases, the objective was to actively "alter the wave scattering properties of a junction" or, in our case, damage location. The various compensating techniques were evaluated using a so-called frequency domain cost function. The disturbance is measured and input to the actuators (located at the junction) to eliminate outgoing waves. The compensator methods used were a noncausal control method, a causal fixed-form parameter method, and a Wiener-Hopf solution method. All these solution methods were tested using the free end of a dispersive, undamped Bernoulli-Euler beam. All three methods were shown to either suppress or eliminate resonance behavior. The investigation also found that effectiveness of this method can change based on structural geometry.

1.2 Artificial Intelligence Literature Review

The ultimate goal of artificial intelligence (AI), and the common conception (or misconception), is that computers will one day be able to mimic humans in all aspects. These human capabilities include what are known as the external representations and the internal representation (see Fig. 1.1). External representations which are represented by input/output tasks include such AI research fields as Vision, Natural Language Processing, Robotics, and Speech. The internal representation research fields include Deduction and Search, Planning, Explanation, and the topic of this thesis, **Machine Learning** (Charniak and McDermott, 1985).

We begin our discussion with the area of learning, otherwise known as Machine Learning. The machine learning field of Artificial Intelligence is a very broad subject which ranges from learning by rote (the trivial) to learning by discovery (essentially, the impossible). As described in Michalski et. al. (1983), this broad topic of machine learning is normally said to contain five main methods: rote learning, learning from instruction, learning from examples, learning by analogy, and learning by discovery (all defined below).

rote learning: The simplest form of learning is by rote. This method simply directly implants new knowledge into the computer whether it be by programming or the memorization of given facts and data. No inferences or extensions are drawn from the incoming information.

learning from instruction: A second method of machine learning is

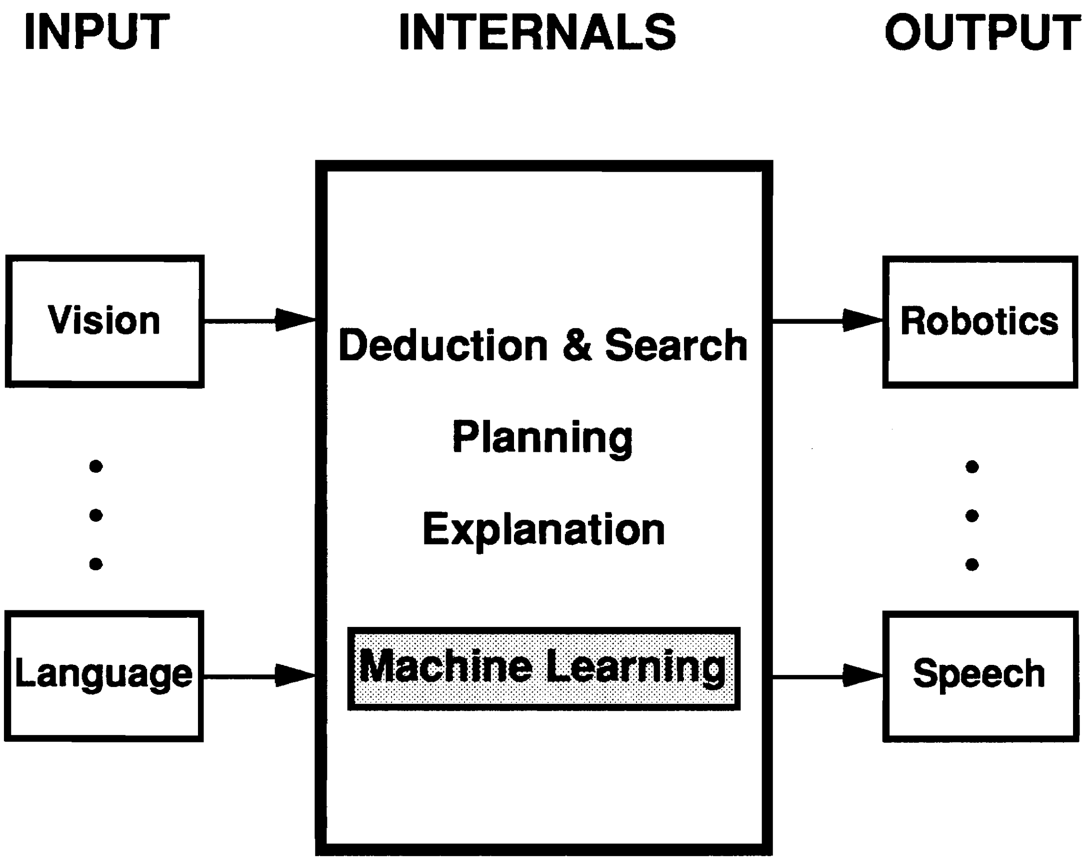


Figure 1.1: Faculties of Artificial Intelligence

learning by being told or from instruction. In this case, the learning device accepts information from a source and uses the information effectively. This background information is used to infer more information about the source.

learning from examples: A computer learning from examples, or learning inductively, consists of hypothesizing general rules from specific examples of information provided to it from the environment. The sources of the information can come from a teacher, the learning device itself, or the external environment. This type of machine learning is the method used in these investigations.

learning by analogy: This method of learning acquires new facts or skills by transforming existing knowledge based on new problems which "resemble" previously solved problems. The new facts or skills to be developed must be analogous to other previously stored information.

learning by discovery: This method is by far the toughest of all learning techniques. Learning by discovery involves observations or inferences which span several concepts acquired to date. This form of unsupervised learning requires the learner to infer various general concepts without the use of a dictionary of information or particular instances (Michalski et. al., 1983).

It was not until the late fifties that machine learning made a big impact in the artificial intelligence community. One of the first and most famous uses of machine learning was performed using the game of checkers. Samuel (1959) used machine rote-learning to prove that a computer can learn to play a better game of checkers than the average person in a "remarkably short period of time." General game playing techniques or heuristics were used by the computer to search through possible moves and associate a value to each board position. Each time the computer took a turn, it searched three ply from the present board state and saved the information. Anytime a board situation was already available in the dictionary the computer would use the additional time to extend the search deeper thereby improving with time. In this way, the computer used dictionaries to improve its playing performance; i.e., rote-learning.

The seventies was the period when learning from examples emerged as a legitimate method for learning. Among others, Winston's Learning Blocks World Concepts dealt with rule induction for simple toy blocks (Winston, 1970). In addition, this method developed maximally-specific conjunctive generalizations (MSC-Generalizations) of the input examples. The input examples are provided by a teacher or source and the method uses "near miss" negative examples to determine the generalized description of the concept (Michalski et. al., 1983).

Hayes-Roth's work on inductive learning was also instrumental to the emergence of learning from examples. Hayes-Roth's work at Carnegie-Melon University dealt with finding these MSC-Generalizations from only a group of positive examples (see, for example, Hayes-Roth, 1976 or Hayes-Roth and McDermott, 1977).

In 1977, Mitchell's Version Spaces method to rule learning was introduced and went a long way to effectively using knowledge-based systems (Mitchell, 1977). This method is a candidate elimination approach to rule learning. The algorithm starts with a listing of all plausible rule versions which applies to the list of instances (this is the version space) and methodically eliminates rules which conflict with the training instances. These training instances are classified in Boolean form to distinguish positive instances from negative instances with regard to a given rule. For each remaining example, the rule of interest is specialized if the example is negative or the rule is generalized if the rule is positive. To specialize the rule, the program matches the model to the sample description and identifies the most important difference. To generalize the rule, the program matches the model to the sample description and determines the type of each difference. In this way, the algorithm narrows down to the best rule which describes the data. Although this method of learning was shown to be quite effective, it is only available for training instances which can be classified in Boolean form.

Now we turn our discussion to the artificial intelligence work which has been performed in the control systems area. The "Boxes" technique of pattern formation, developed by R.A. Chambers and Donald Michie in 1968, is one of the original Artificial Intelligence involvements in control systems. This algorithm works by discretizing the states of a given system and categorizing them into a matrix (boxes). "States" in this sense are the minimum number of variables needed to describe the system completely using an assumed equation of motion. For each box, different possible inputs to the system are tested as a control and the outputs are stored along with a measure of the effectiveness. Using all this information, the machine can "learn" or evaluate what the best inputs are to the system for any combination of states.

This method of discretizing the states of a system into subspaces was originally performed on a cart and pole system by Widrow & Smith (1964). The system contains a rigid pole which is hinged to a cart with wheels. The pole is only allowed to swing in one dimension similar to the movement ability of the cart on the track. The controller can apply either a full force left or full force right input. A failure is obtained when either the pole falls or when the cart is moved a specified distance from the starting location.

Then in 1972, this system was adopted by Michie & Chambers (1972) to perform an experiment in adaptive control using "Boxes". At any given state-situation "box" the computer would begin by randomly applying a left or right (L or R) full force response and recording the time until failure. Each successive time the simulation would hit that state, the computer would determine the previous response which averaged the longest time until failure, apply that response, and then reaverage the results. At the end of the simulation, each box was associated with the input (L or R) that averaged the longest result. In this way, the program learned on its own as opposed to being "taught".

Neural Networks have also been extensively used in an effort to solve difficult learning control problems. A crude definition of an Artificial Neural System or Neural Network is a mathematical model which attempts to mimic the human brain. More specifically, Neural Networks consist of many simple nodes connected together by weights of different strengths. These neuron-like adaptive elements were used by Barto, Sutton, and Anderson to control the pole and cart system similar to the one described above (Barto et. al., 1983). The authors assumed there was no accurate model of the system and the only feedback evaluating performance was a failure signal which occurred when the pole fell over. In this case, the learning system consisted of an associative search element (ASE) and an adaptive critic element (ACE). The ASE constructed the input/output associations and the ACE constructed the more informed evaluation function. Many other Neural Network studies using adaptive element have been performed (see Rosenblatt, 1962 and Widrow and Smith, 1964) but they are not as instrumental for our development.

1.3 Scope and Objectives

The focus of this thesis is to show the viability of using artificial intelligence, specifically rule induction, for the purpose of active damage control. The intention of this method is to overcome some of the problems associated with other techniques as described in the literature reviews.

The purpose of this thesis is to separately investigate the two basic tasks associated with active damage control. The first is to use artificial intelligence to identify damage in a composite plate with a sawcut. The second focus of this thesis is to use artificial intelligence to locally control or suppress the movement of a vibrating system.

As stated above, the first task is to investigate the viability of damage identification using artificial intelligence. Specifically, rule induction is used to analyze laser data acquired from both damaged and non-damaged composite plates. The data from these plates are analyzed using a software package called KnowledgeSEEKER. The package, created by Firstmark Technologies, LTD, allows its users to extract decision-making information from a database in the form of decision or regression trees.

The viability of damage control using artificial intelligence is investigated in a different manner. The control of a point on a beam is performed experimentally using a rule-induction method developed by Michie and Chambers called "Boxes" (Michie and Chambers, 1968). Once again, vibrational information from the structure is used to develop the optimum local control for a specified point on the beam.

It is expected that this method of rule induction within the field of Artificial Intelligence will perform well enough to promote optimistic ideas for future research into applying Artificial Intelligence to recent technological research areas.

1.4 Outline of the Thesis

The first chapter of this thesis discusses the related works in both the field of artificial intelligence and active damage control. Chapter 2 contains the investigation of using rule induction for the purpose of damage identification. This includes a detailed description of the data gathering process from the plates, a complete theoretical description of the KnowledgeSEEKER software package, and the results of this part of the thesis. Chapter 3 contains the investigation of using rule induction, specifically "BOXES", for local vibration control. This includes a theoretical discussion of the "BOXES" method of rule induction, a series of theoretical simulations ensuring the effectiveness of this technique, a detailed description of the experimental testbed, and the results of this part of the thesis. Finally, chapter 4 discusses the results found in the previous two chapters and ties them together. This chapter also includes some recommendations for future work.

Chapter 2

Study of Damage Identification Using Rule Induction

"The distinct aim of machine intelligence within the general field of computer science is artfully to encroach on this [computer science] preserve, annexing to the machine's domain more and more of the human's elusive aptitudes...to construct general rules from particular instances and to define concepts via examples."

D. Michie (1967)
Introduction
Machine Intelligence 1

The following is an initial investigation into the use of artificial intelligence for damage identification. The method of learning from examples is performed on a set of pregathered plate velocities in an attempt to distinguish which plates have damage and which do not. The motivation behind the investigation is not only to use artificial intelligence in the determination of whether or not damage is present in structures, but also to use these techniques to evaluate where the damage has occurred.

In general terms, the investigation consists of gathering velocity data over three composite plates using a laser beam. The three carbon composite plates are 0.3 m (12 in.) by 0.3 m (12 in.) and consist of one plate with no damage, one plate with damage at the center, and one plate with damage at the edge (see Fig. 2.1). Damage, in this case, is a thin razor cut four inches long.

Using knowledgeSEEKER, a statistical rule generation package, the velocity data is transformed into a logical set of rule descriptions. The rules should then correctly identify subsequent recorded velocity from the same sources and correctly predict if it came from a plate with damage, and if so, where the damage occurred (i.e., at the center or at the edge). Of secondary interest, is the number of rules required to correctly describe the data. For practical applications it is important to minimize the loss of "transparency." Transparency is the ability to interpret this rule set information.

In the end, the success of this investigation relies upon what percent of the cases the rule set correctly describes and how large a rule set is required for this accuracy. In addition, this technique is used to analyze the information for "critical" or "sensitive" areas on the plate where distinct differences in the data result in accurate damage detection. It is the author's hope that these techniques, if successful, can be extended to other composite materials and structures for damage identification and location.

2.1 Experimental Setup and Data Acquisition

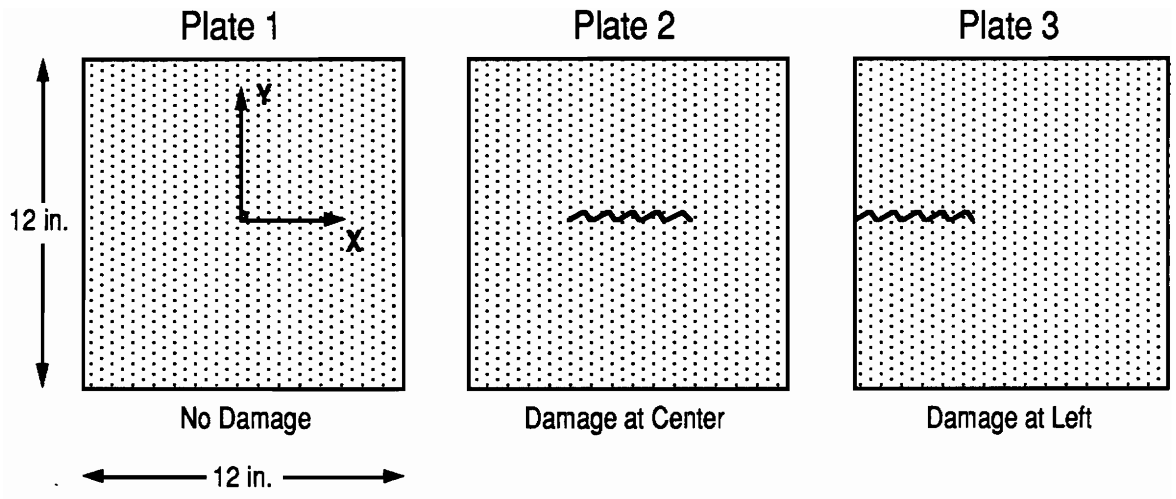


Figure 2.1: Plate Definitions and Dimensions

As mentioned, velocity data from three plates are analyzed in an attempt to distinguish which data sets are associated with the damaged plates and which data sets are associated with the undamaged plate. The three thin 0.3 m (12 in.) by 0.3 m (12 in.) plates are all made of the same composite material. The plates are hung from the ceiling using fishing wire to simulate free-free boundary conditions.

The dynamic response of the plates (i.e., the real and imaginary parts of the vibration velocities) are measured using a laser scanning sensor system. The plates are driven at a single frequency and the data is gathered using the Omitron 9000 series laser system. The scanning area of the plates is divided into a thirty-point by thirty-point matrix. This results in a total of 900 scanning points for each of the three plates to be analyzed by the software package. The order in which the points are scanned by the laser is shown in Fig. 2.2.

Each scanning point in the data file contains the following information:

1) Point location -- theta x
2) Point location -- theta y
3) Point velocity -- real
4) Point velocity -- imaginary

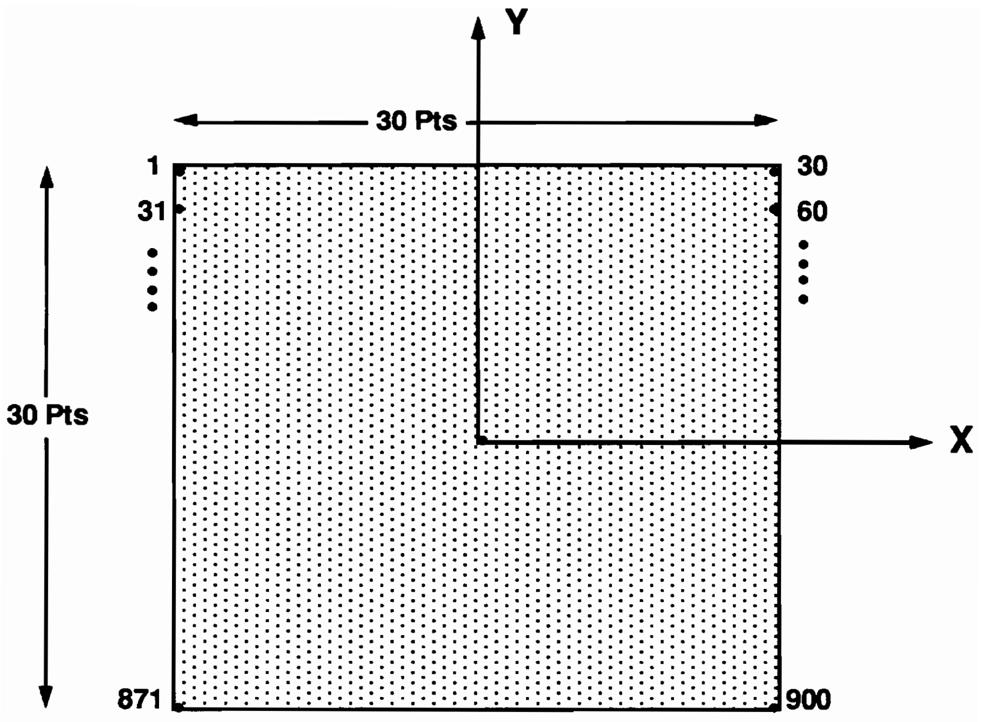


Figure 2.2: Plate and Associated Data Point Numbering System

The data file is then modified to include which points are associated with either no damage, center damage, or edge damage. This investigation uses the aforementioned attributes or predictor variables to accurately "break down" the database. The damage status (none, center, or edge) is to be defined as the dependent variable while the other five attributes (thetax, thetay, real velocity, and imaginary velocity) are to be defined as the independent variables.

Before deciding at which frequency to excite the plates, the frequency response functions were analyzed (Liu, 1993). The FRFs for the composite plates of interest are shown in Figs. 2.3, 2.4, and 2.5. Since the lower modes are more prominent, we will use one of the lower, more distinct modal frequencies to excite the plates. In short, the 208 Hz mode is chosen because it is associated with a low, clean natural frequency.

The resonances associated with this mode vary slightly from plate to plate. The author makes the assumption that these slight frequency differences (of about 5 Hz) are associated with material inconsistencies in the plates. Therefore, the comparison of the modes or resonances is used rather than comparing exact frequencies. Graphical displays of the mode shapes for plates one, two, and three as produced by the laser system are shown in Figs. 2.6, 2.7, and 2.8 respectively.

A flow diagram of the data acquisition process is shown in Fig. 2.9. The plates are

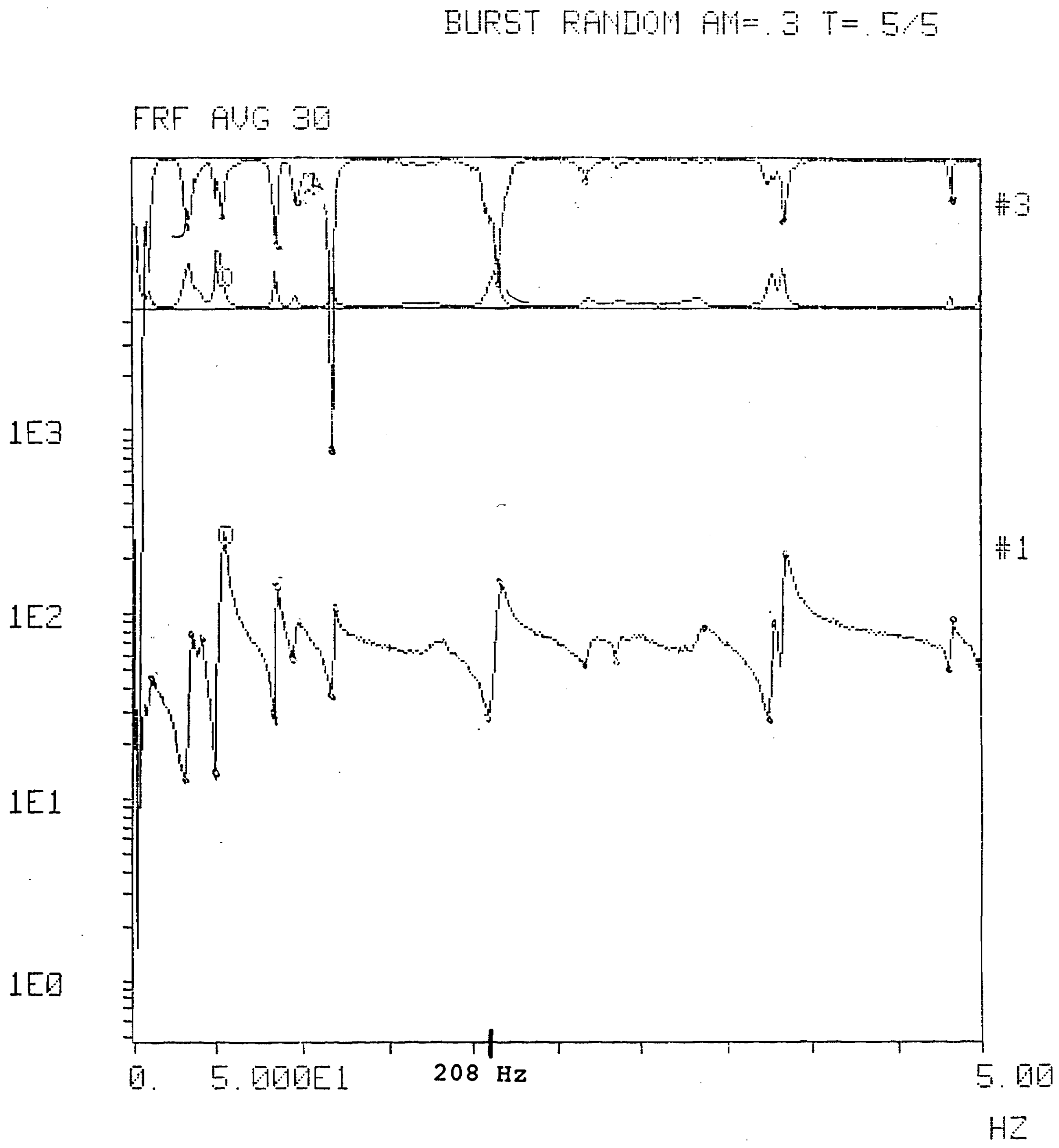


Figure 2.3: Frequency Response Function for Non-damaged Plate

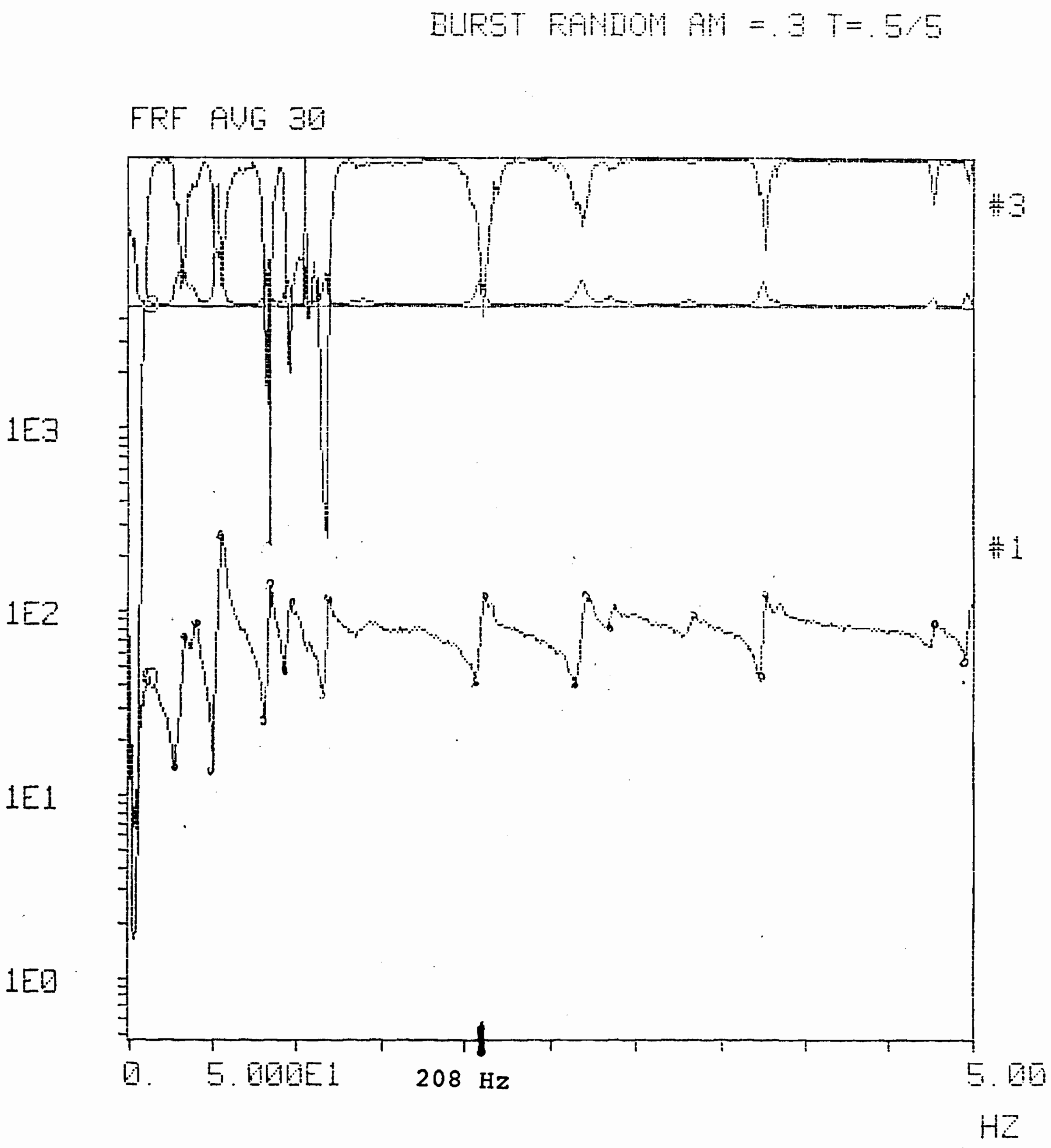


Figure 2.4: Frequency Response Function for Center-damaged Plate

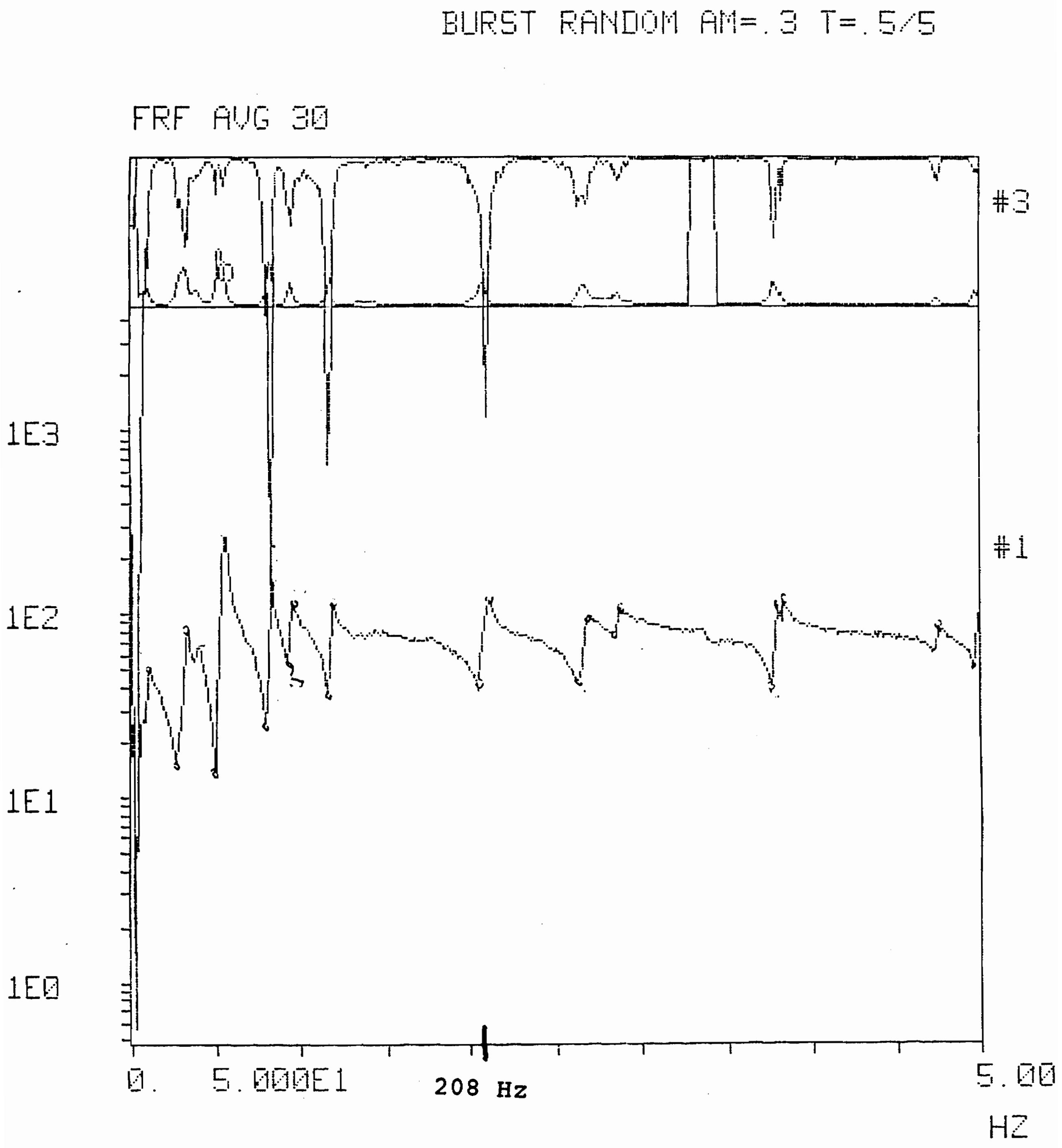


Figure 2.5: Frequency Response Function for Edge-damaged Plate

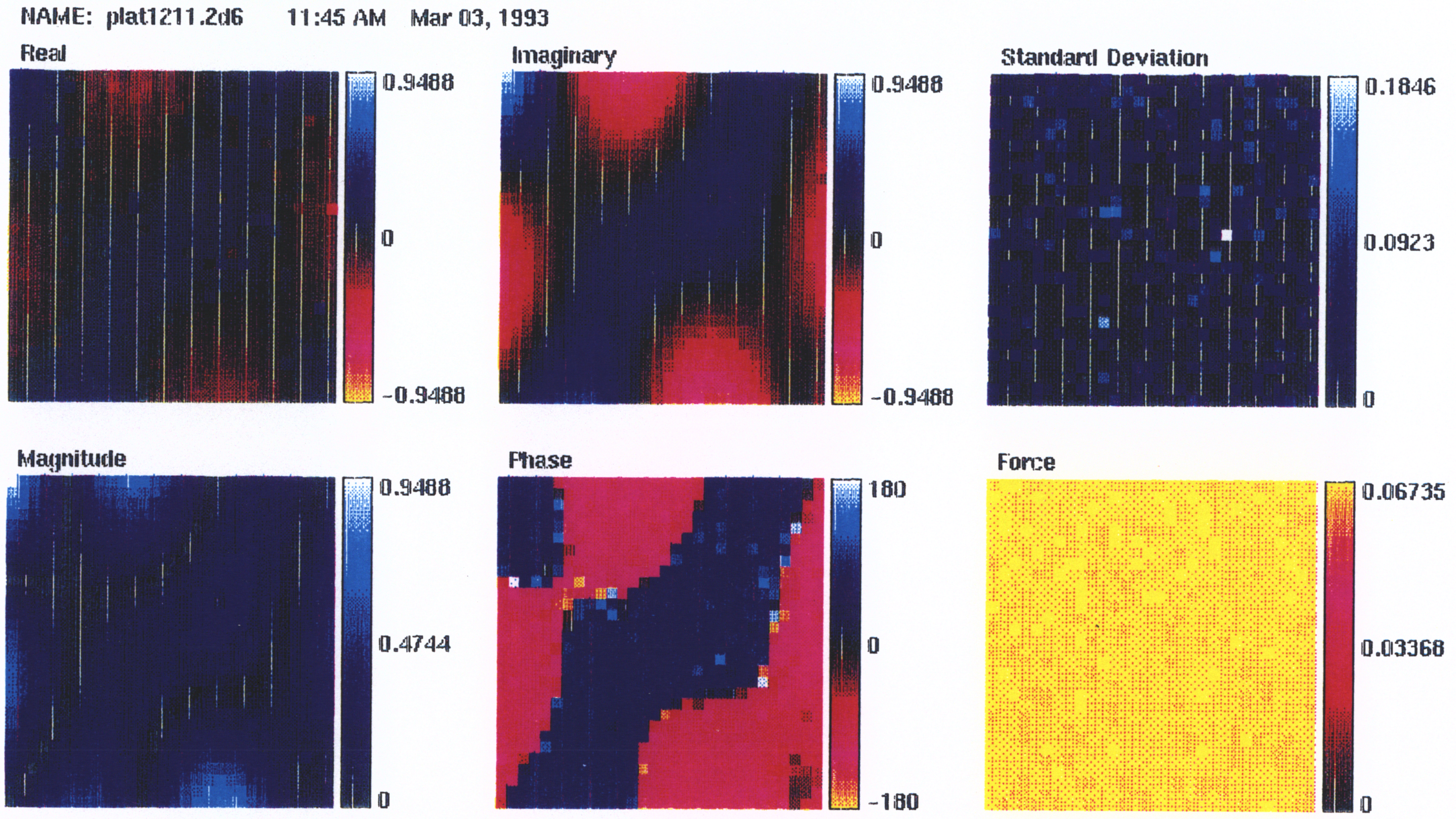


Figure 2.6: Laser-produced Mode Shape for Non-damaged Plate

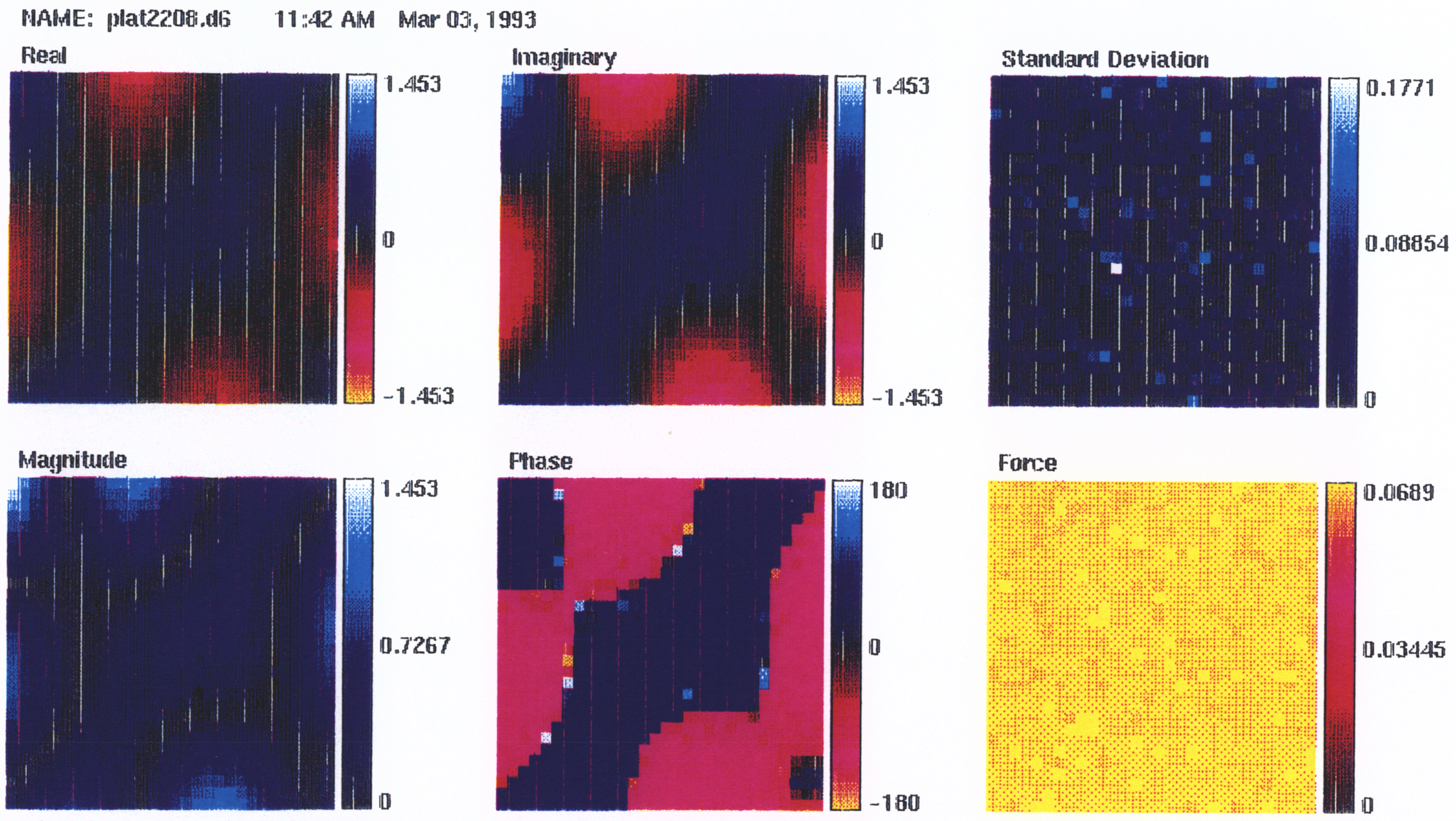


Figure 2.7: Laser-produced Mode Shape for Center-damaged Plate

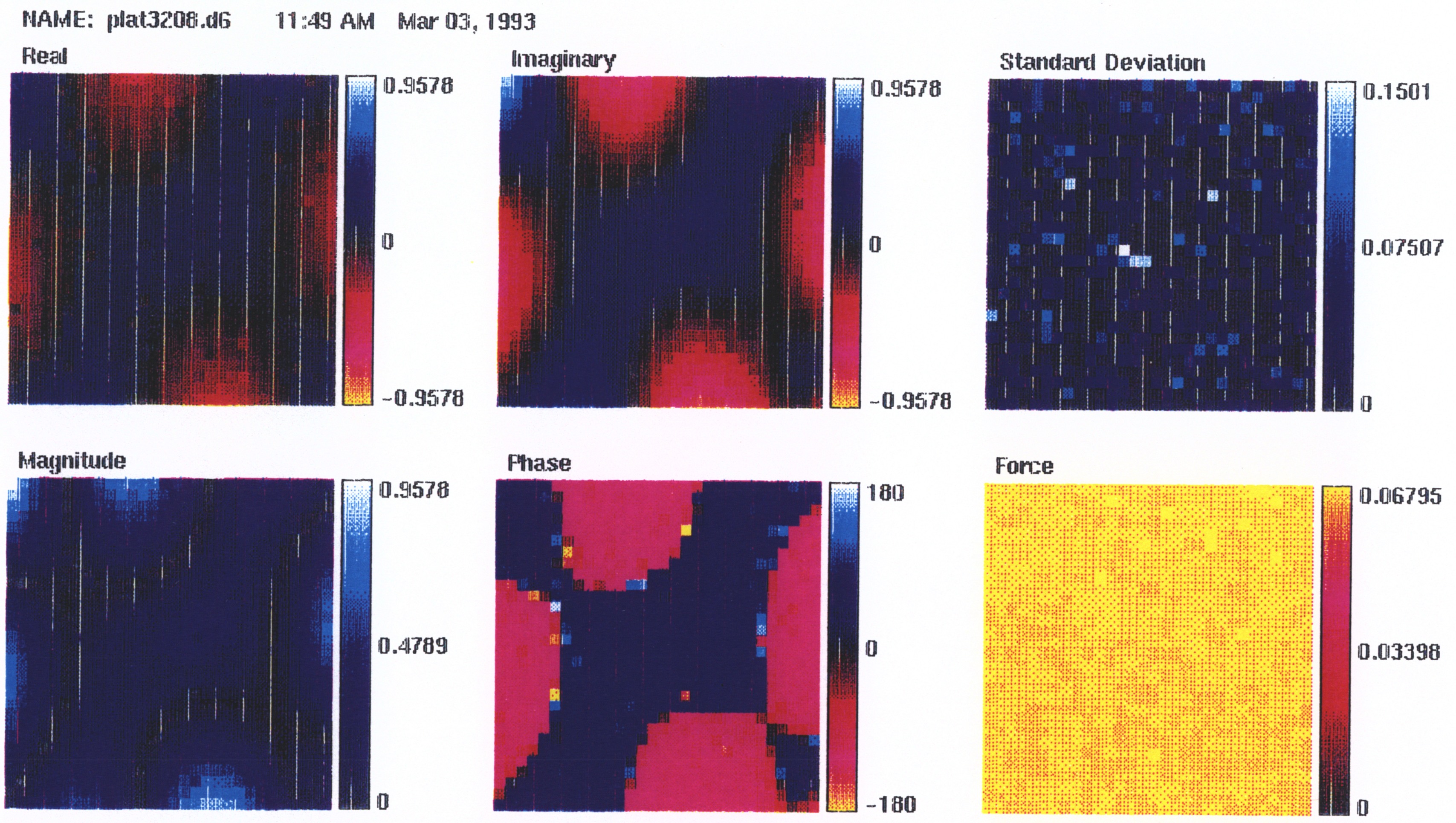


Figure 2.8: Laser-produced Mode Shape for Edge-damaged Plate

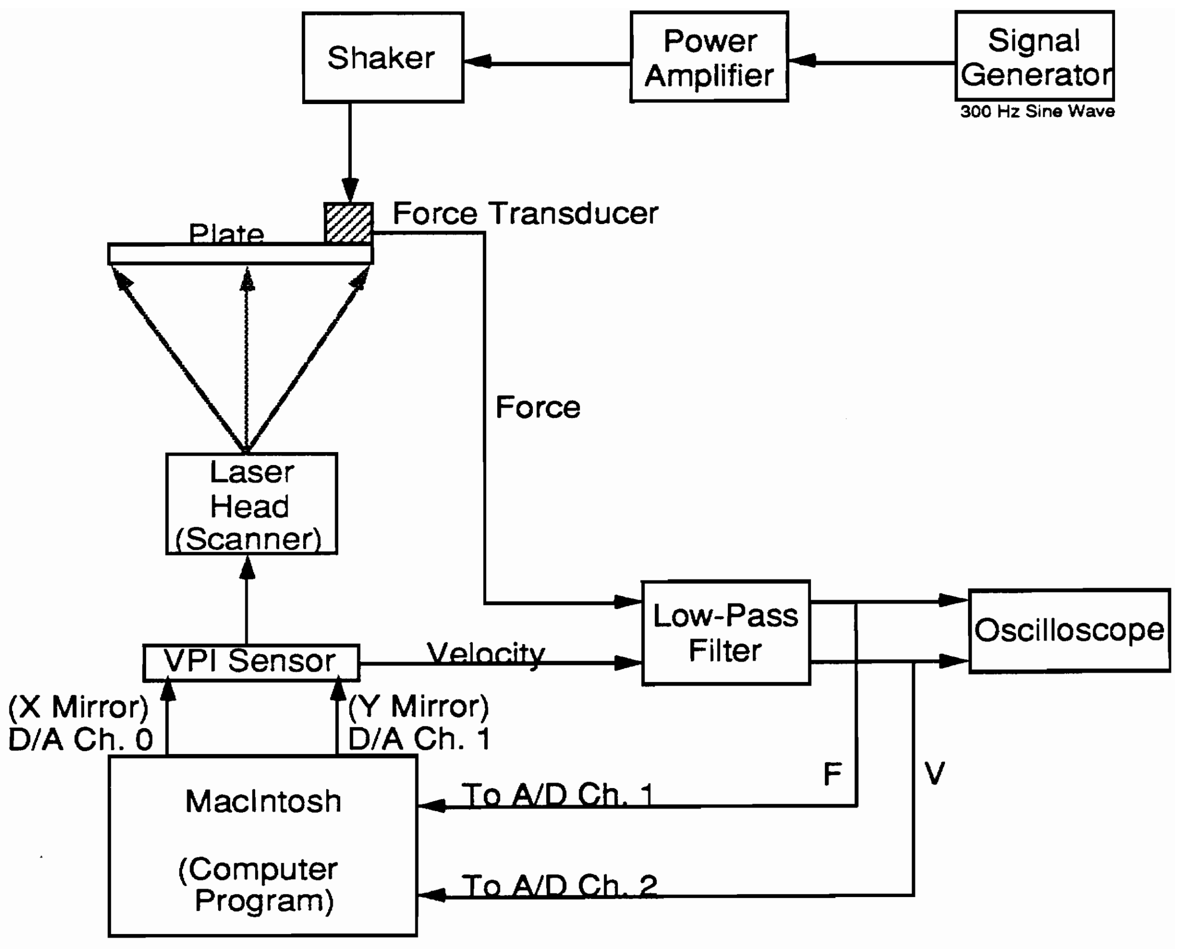


Figure 2.9: Flow Diagram of Data Acquisition System

excited with a force equivalent to 600 millivolts using a shaker located at the bottom, right-hand corner of each plate. The excitation force is monitored using a piezoelectric force transducer connected to the shaker.

The next important part of the investigation is to analyze the data using KnowledgeSEEKER; but, before this is discussed, it is important to understand the theoretical aspects of KnowledgeSEEKER.

2.2 KnowledgeSEEKER

The pregathered velocity data is then analyzed using the software package KnowledgeSEEKER. KnowledgeSEEKER is a software program created by Firstmark Technologies, LTD that allows its users to extract decision-making information from a database in the form of decision or regression trees. A set of primitive attributes of the damage are used by the program's statistical problem-solving algorithms to break down the data into rules. Theoretically, KnowledgeSEEKER is a computer model based on Kass' recursive partitioning algorithm, CHAID (Kass, 1980).

Classification and regression tree techniques have been used successfully to extract decision-making structures from large multivariate data sets. The main technique used is the recursive partitioning of a data set into mutually exclusive, exhaustive subsets that best describe the dependent variable. The information set developed from the plate velocity information in this study is most suitable for the application of the recursive partitioning technique.

2.2.1 KnowledgeSEEKER Background

The first technique of this structure, Automatic Interaction Detection (AID), was developed by Morgan and Sonquest (1963). AID uses an interval-scaled dependent variable to maximize the between-group sum-of-squares (F-statistic) at each bisection or split. Kass refined the process by using a nominal scaled dependent variable, and maximized the significance of a chi-squared statistic at each partition. Using this significance testing and an exhaustive search for the highest chi-squared, the partitioning locates the multi-way splits.

Kass classified the predictor variables into three types: Monotonic, Free, and Floating. Monotonic variables have a purely ordinal scale (i.e, they have known or unknown numeric values associated with their categories), thus only contiguous categories may be grouped together. Free predictors have purely nominal categories (i.e., differing in kind rather than degree) and any grouping of categories is permissible. For Floating predictors, all categories are on an ordinal scale but one. This either does not belong with the other categories or contains an unknown position. This type of variable allows the data set to contain missing values. The floating category may be grouped with any other category or group of categories, or by itself. The remaining categories may only be grouped contiguously.

A search method is employed to determine the optimum classes of interest. The first step in the process is to determine the best partition for each attribute, or predictor. Each attribute is compared and the most significant one is selected. The data is then partitioned based on this attribute. This process is repeated until no more significant partitioning is possible. Kass' algorithm is as follows:

Step 1. For each predictor in turn, cross-tabulate the categories of the predictor with the categories of the dependent variable and do steps 2 and 3.

Step 2. Find the pair of categories of the predictor (only considering allowable pairs as determined by the type of predictor).

Step 3. For each compound category consisting of three or more of the original categories, find the most significant binary split (constrained by the type of the predictor) into which the merger may be resolved. If the significance is beyond a critical value, implement the split and return to Step 2.

Step 4. Calculate the significance of each optimally merged predictor and isolate the most significant one. If this significance is greater than a preselected criterion value, subdivide the data according to the (merged) categories of the chosen predictor.

Step 5. For each partition of the data that has not yet been analyzed, return to Step 1.

This form of recursive partitioning, as described by Kass, is used in KnowledgeSEEKER. KnowlegeSEEKER also uses the same three classifications of categories (monotonic, free, and floating) that Kass describes in the CHAID model. The one difference between Kass' CHAID and KnowledgeSEEKER is the statistical measuring tool. KnowledgeSEEKER's k-nearest neighbor measuring tool avoids the strict assumption of normal probability density functions (Friedman, 1977).

The algorithm used in software program to search for the best k-way split is described by Biggs, DeVille, and Suen (1991) and shown below.

Step 1. Select the pair of categories of the predictor variable that is most similar using an F or chi-square significance test considering only pairs that can be merged given this type of predictor variable. Set the resulting grouping of categories as possible split number i+1 (1 for the first iteration). Repeat this step until only two groups are left, say with possible split number i=n*.

Step 2. Calculate the significance (probability p) of each of the n* sets of groupings of categories using the F or chi-squared test. The grouping with the lowest probability p is taken to be the 'best' split of the node for that predictor variable.

Step 3. Determine if this 'best' split is significant.

The test of significance is based on the grouping of categories with the highest level of significance using chi-squared. Because of this, a control needs to be used to reduce the probability of type 1 error, otherwise known as finding a split significant when no relationship exists between response and predictor variables. The method employed by KnowledgeSEEKER and proposed by Kass is the Bonferroni adjustment factor. Biggs et. al. (1991) show that the Bonferroni inequality allows for the setting of the significance level to keep the type 1 error rate below a preset level; i.e., minimize the discovery of chance relationships.

2.2.2 KnowledgeSEEKER Specifications

Having discussed the theoretical background, the specifications chosen for the rule-generation process are now presented. The Bonferroni adjustment, as described above, is used to minimize the discovery of chance relationships. Increasing the size of the adjustment increases the level of conservatism. For this reason, the adjustment is set to a level of 1.0.

One specification of interest is the growth method to be used by KnowledgeSEEKER. The choices are limited to either cluster or exhaustive. For this purpose, the exhaustive method, although more inefficient, is better since it maximizes the statistical significance and displays the strongest relationships.

The filtering method is the final specification to be defined. The options are 1) decision support; 2) prediction; or 3) exploration. The prediction support option is chosen and is associated with a 95% certainty rating. This setting confirms that the displayed relationships are valid with only a 0.05 filter setting. Using the prediction setting, it is highly unlikely that one will produce any misleading or chance results.

Before proceeding, it should be noted that many of the aforementioned specifications have been chosen during the data analysis progression but the present settings are identified as producing the "best" results as defined in the following section.

2.3 Resulting Rules and Evaluation

The results of this initial damage identification study using KnowledgeSEEKER are shown in the six-page connected figure (Figs. 2.10, 2.11, 2.12, 2.13, 2.14, and 2.15). The connected figure presents 71 separate rules in the form of a decision tree and represent the most significant decision splits as found by KnowledgeSEEKER. The rules, typed out in generic form, are shown in appendix A as a reference.

Each box contains four pieces of information showing the damage breakdown of that rule. The number in the lowest part of the box indicates the total number of cases which are described by the present breakdown. The three percentages in the top part of each box represent the breakdown of cases related to the type of damage. The first percentage corresponds to center damage, the second to edge damage, and the third to no damage. The ideal case occurs when a final box in a branch contains a set of cases which all belong to the same damage classification; i.e., one group is 100% while the other two correspond to 0%.

For example, rule number one states that if, for any given piece of velocity data, the real part of the velocity is between -3.779 and -1.761 meters per second, and the imaginary

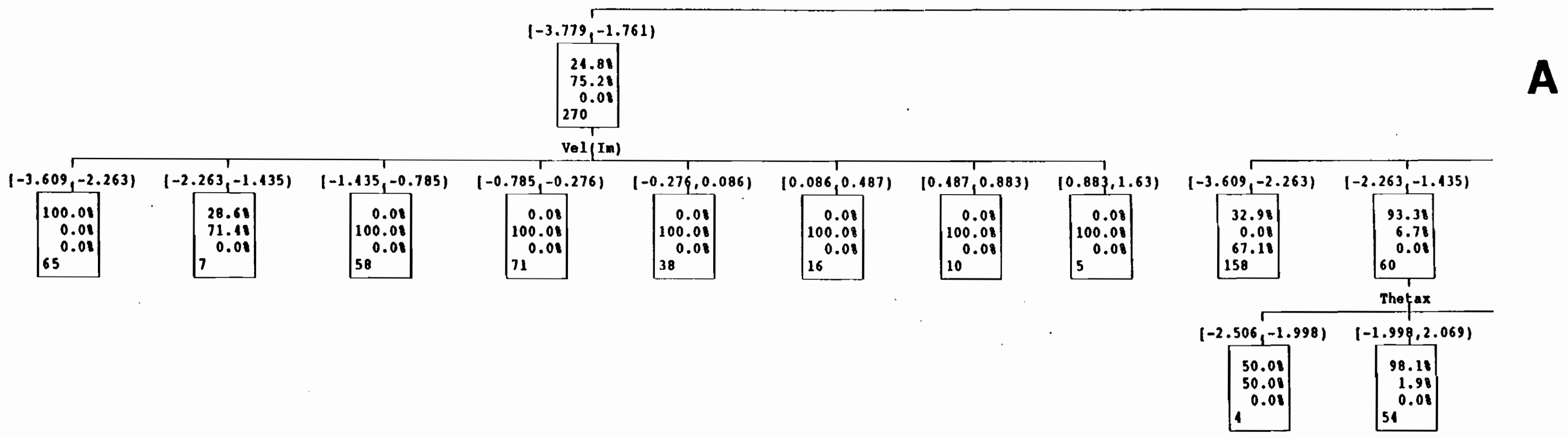


Figure 2.10: Decision Tree Describing Plate Data (Unmodified)

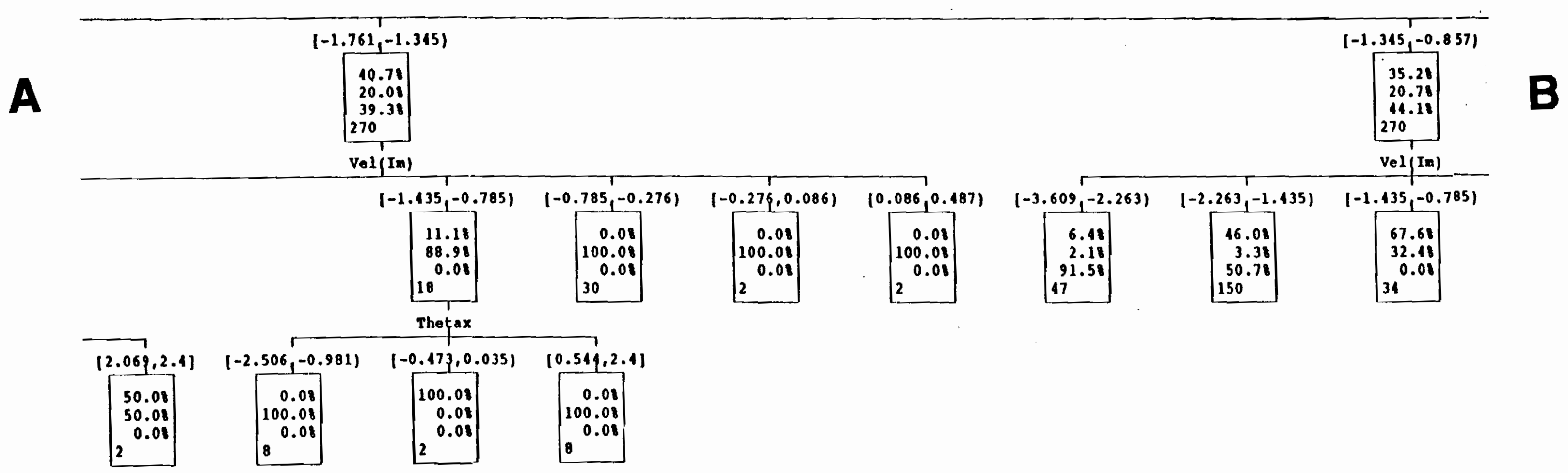


Figure 2.11: Decision Tree Describing Plate Data -- Continued

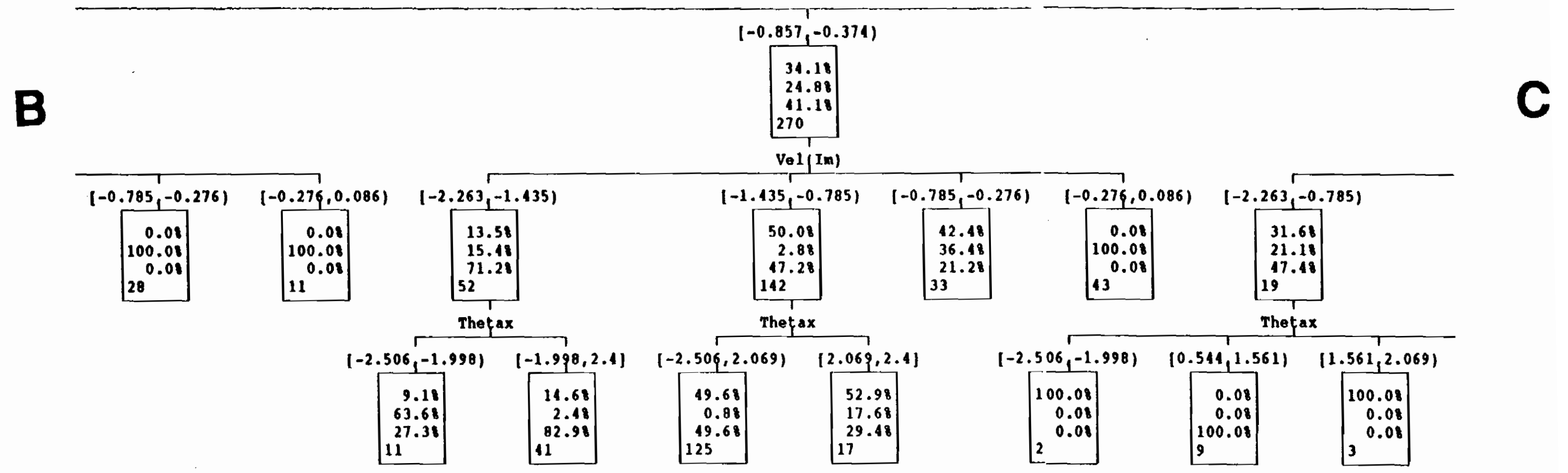


Figure 2.12: Decision Tree Describing Plate Data -- Continued

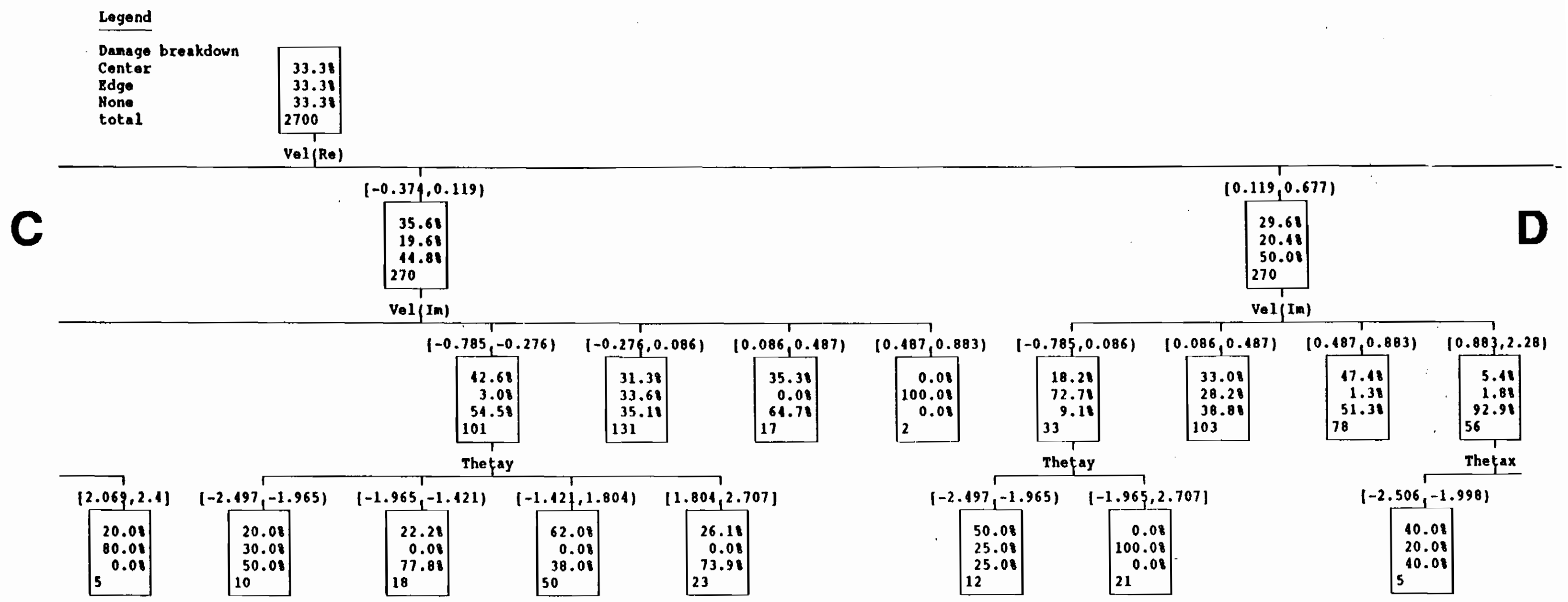


Figure 2.13: Decision Tree Describing Plate Data -- Continued

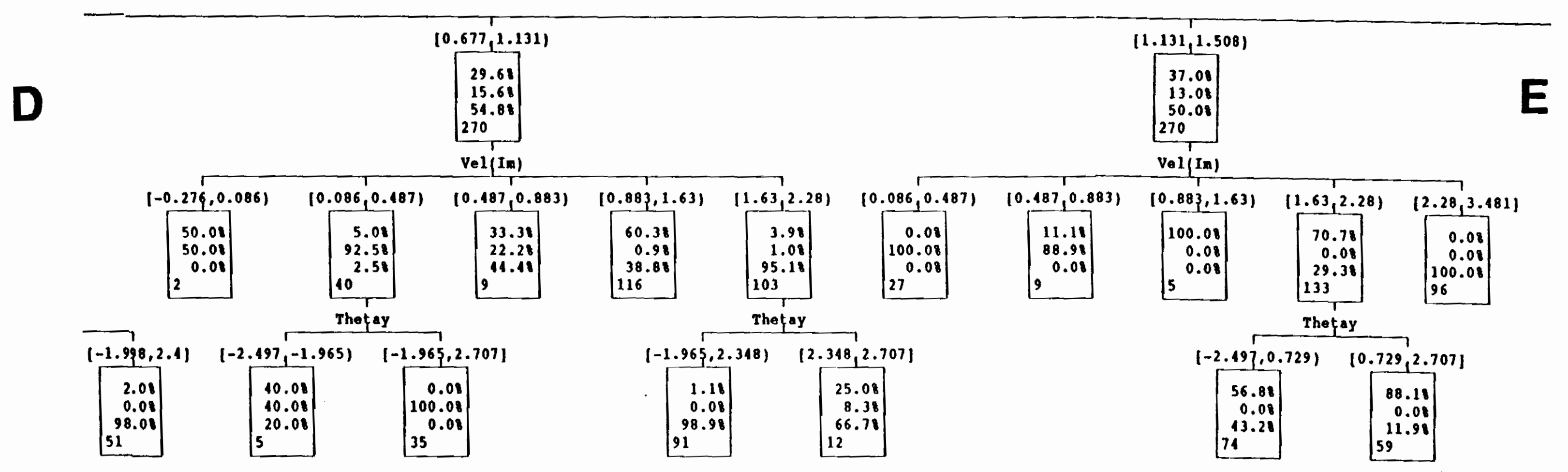


Figure 2.14: Decision Tree Describing Plate Data -- Continued

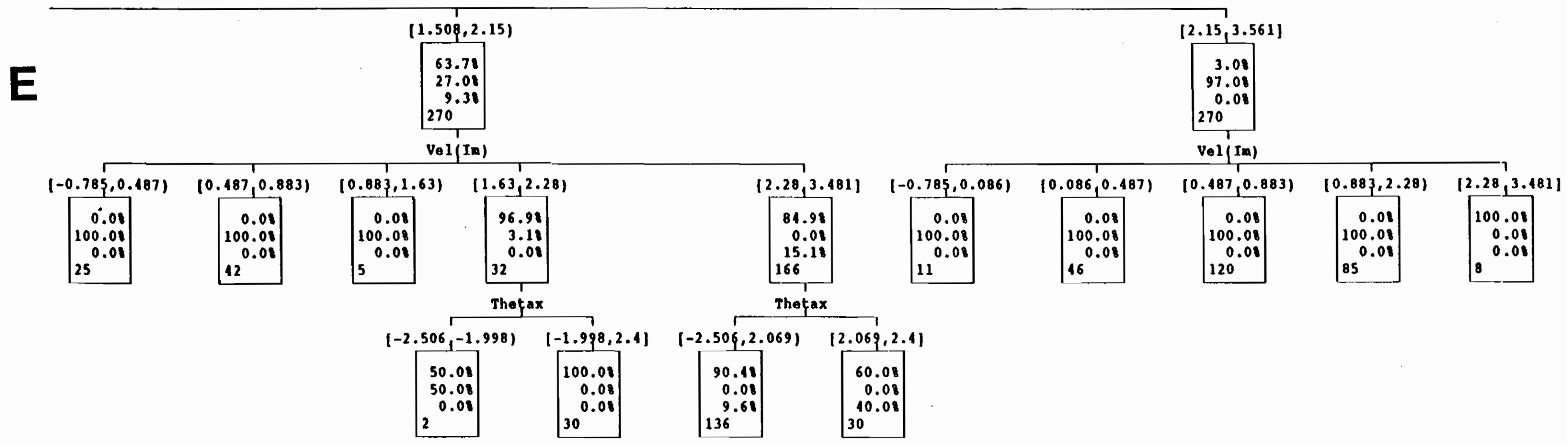


Figure 2.15: Decision Tree Describing Plate Data -- Continued

part of the velocity is between -3.609 and -2.263 meters per second, then there is a 100% chance that the damage occurs in the center of the plate for the 65 cases that apply. Similarly, rule two states that if the real part of the velocity is between -3.779 and -1.761 meters per second, and the imaginary part of the velocity is between -2.263 and -1.435 meters per second, then there is a 28.6% chance that the damage occurs in the center; there is a 71.4% chance the damage occurs at the edge; and there is a 0% chance that there is no damage for the 7 cases that apply.

The most statistically significant split for the data set is the real part of the velocity. The data inherently splits into ten subgroups using this attribute. For each of these subgroups, the most accurate rule splits all turn out to be based on the imaginary part of the velocity. From there, the subgroups (or leaves) contain more definite significant splits using the location of the data point; i.e., theta x or theta y.

As mentioned earlier, the result is 71 independent rules needed to accurately classify the data gathered by the laser system. This results in a 77.4% accurate classification of the data. The accuracy is measured by accounting for the total percentage of cases which are not described by the majority in any given box. For example, in rule two 71.4% or 5 of the 7 cases have a damage classification of edge damage. This is the majority. Thus, the other 2 cases are inaccurate and defined as such. In this way one can generalize the accuracy of the entire rule set.

There are some other interesting points to note when evaluating the data. Among all the rules which contain some type of dispute, the center damage versus no damage dispute occurs 68% of the time. This clearly shows that it is much harder to distinguish damage in the center of the plate than damage at the edge of the plate when analyzing velocity information. This is consistent with the symmetry issue as discussed in Section 1.1.1, pages 4 and 6.

Another decision tree derived from the data is seen in the other three-page connected figure (Figs. 2.16, 2.17, and 2.18). In this case, the velocity data is forcibly split, first at theta x and then at theta y. KnowledgeSEEKER then continued branching the data with respect to the imaginary and real velocities.

When the data is split in this manner, we can see where the critical or sensitive areas are on the plate. That is to say, this connected figure shows what areas of the plate have distinct velocity information when compared to damaged and undamaged plates. In addition, this rule set is much smaller and hence more easily understood.

The six most sensitive areas (or rules) which most accurately identify the three plates are indicated on the connected figure with arrows. To more easily identify where these "sensitive" areas occur, a graphic depiction of the plate and critical areas is shown in Fig. 2.19. In each of the six locations, the plate with no damage has the smallest velocities,

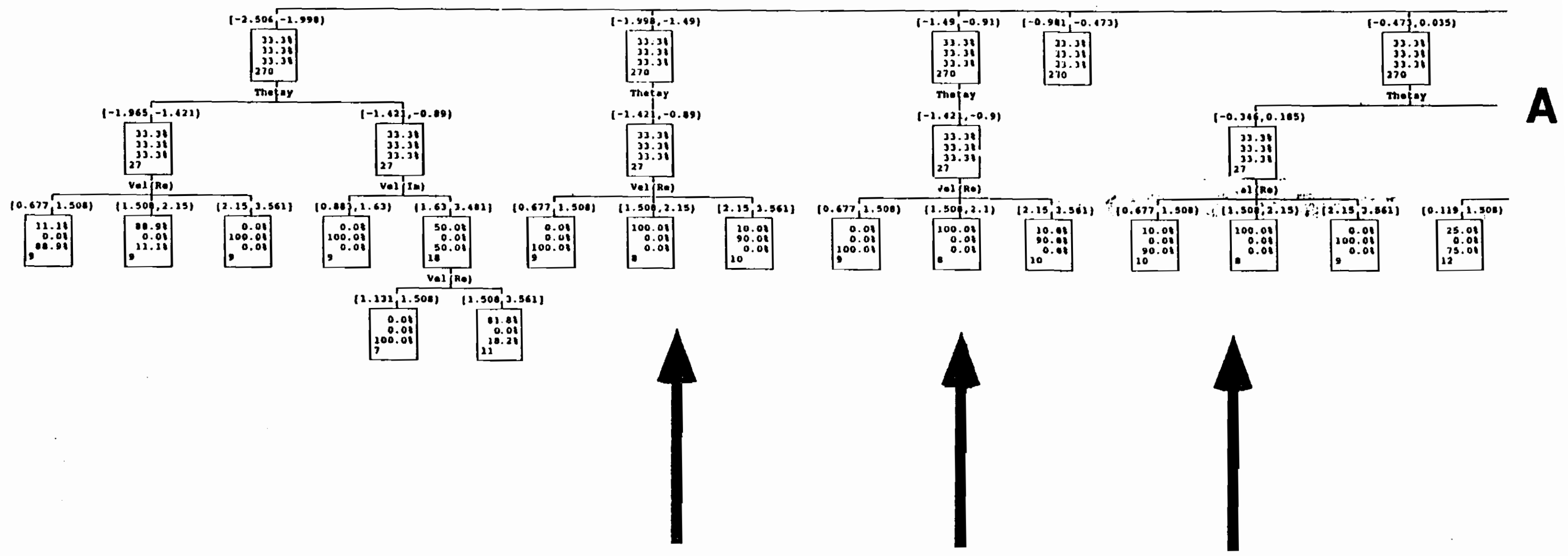


Figure 2.16: Decision Tree Describing Plate Data (Modified to Identify "Sensitive" Areas)

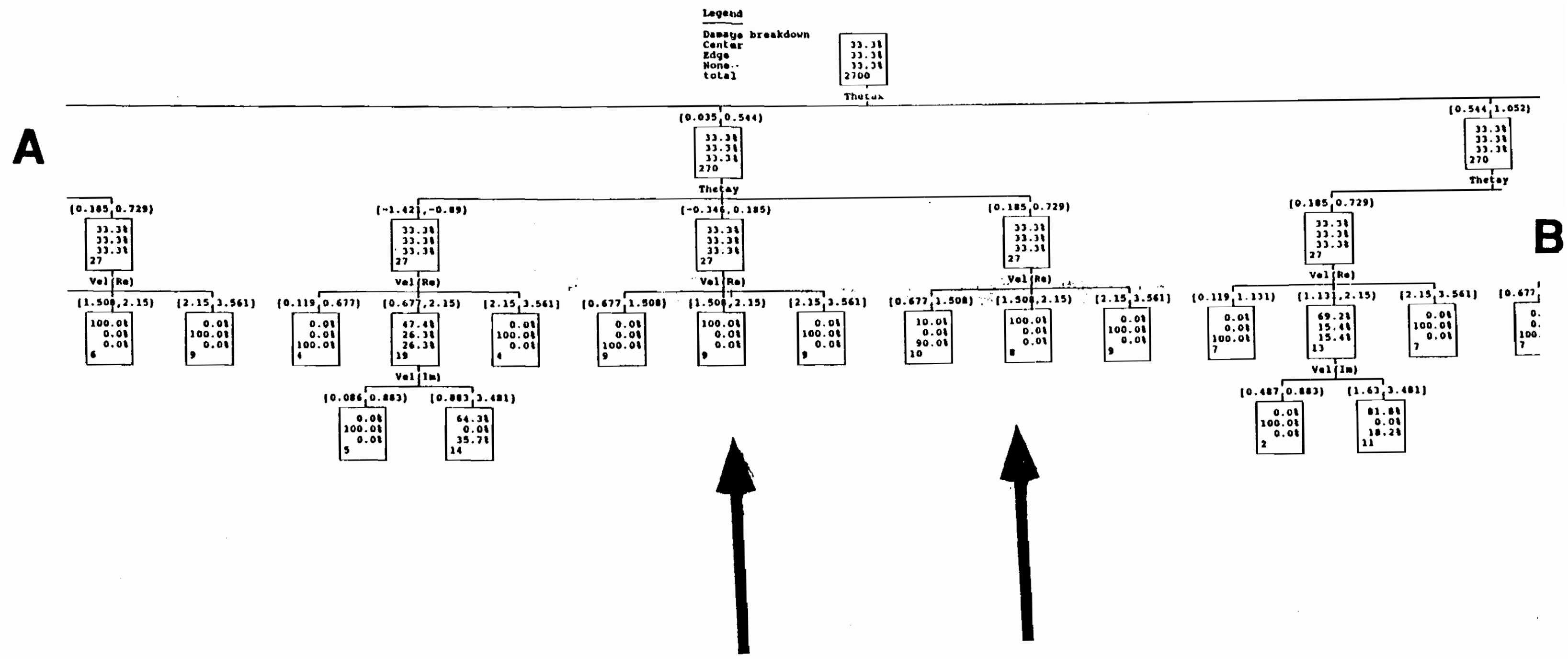


Figure 2.17: Decision Tree Describing Plate Data -- Continued

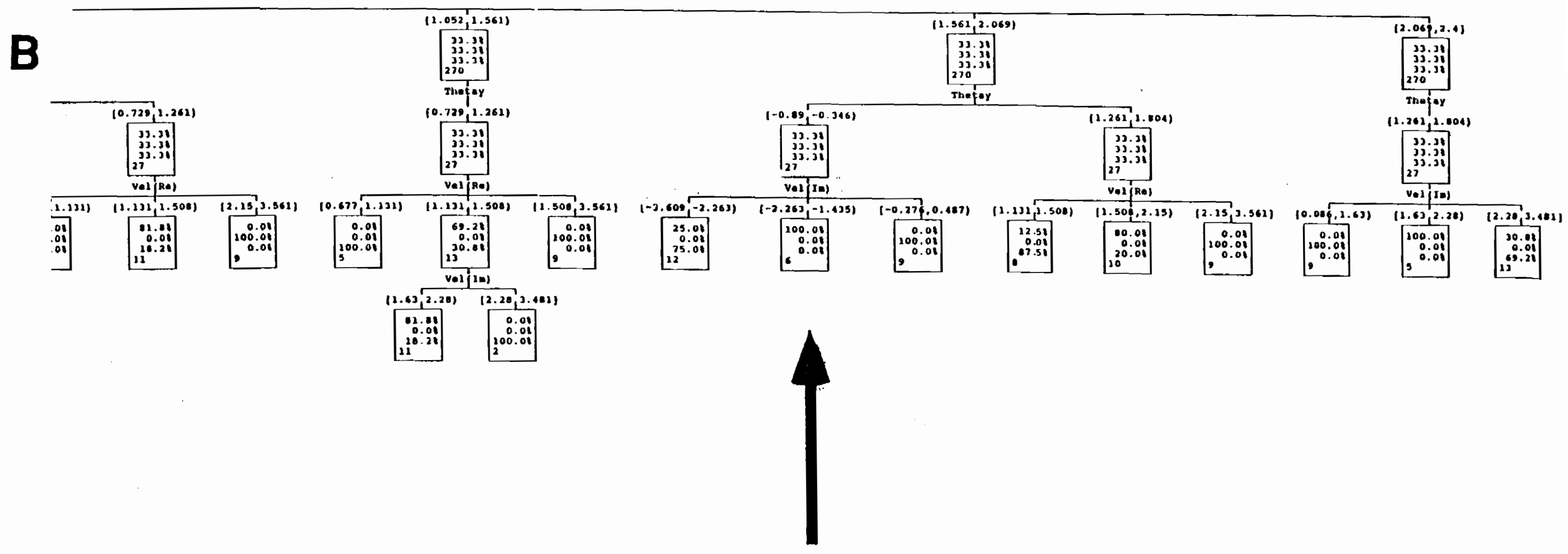


Figure 2.18: Decision Tree Describing Plate Data -- Continued

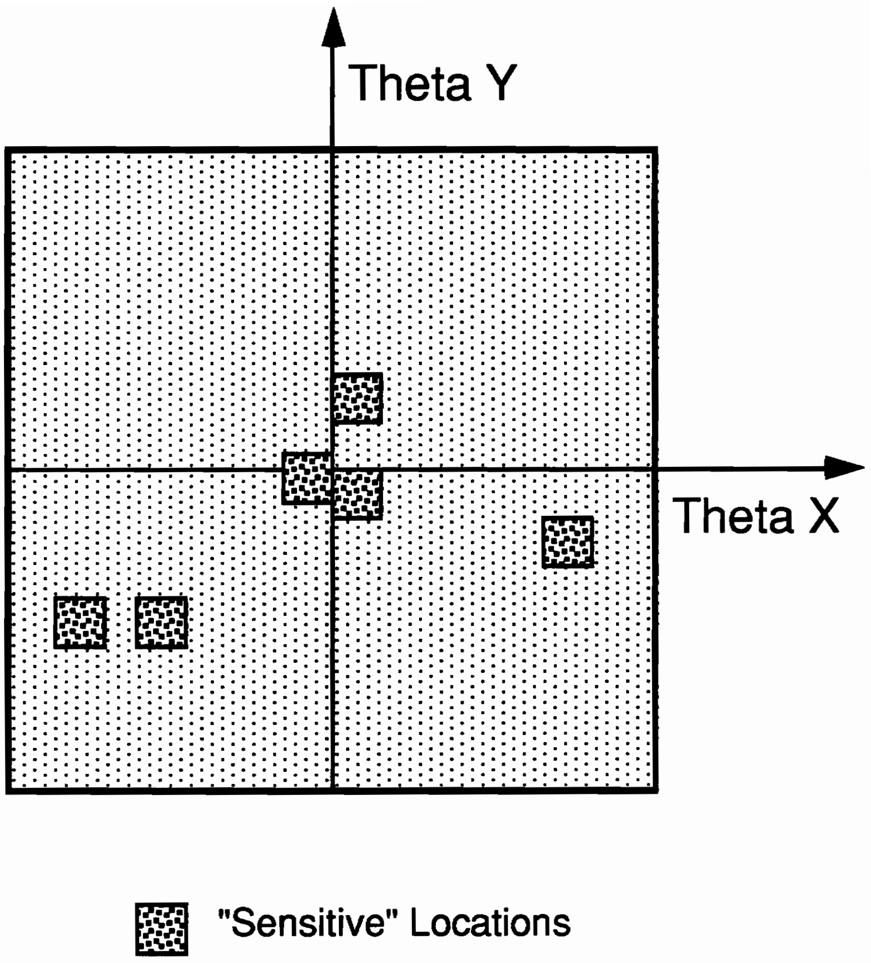


Figure 2.19: "Sensitive" Areas on Plates as Described by Modified Decision Tree

the plate with center damage has the medium velocities, and the plate with edge damage has the largest velocities. A priori, this seems to make sense the plates with damage have more flexibility and movement than the plate without damage. The rules typed out in generic form are shown in appendix B.

It has been shown that rule induction within the realm of Artificial Intelligence can be a very powerful method for damage identification. It is not only important to note how this machine learning technique can identify obscure patterns in the plate, but also that this technique can be applied to any type of vibrating structure for identifying damage. This non-destructive damage identification method could prove very useful in the many applicable and practical situations.

Chapter 3

Study of Damage Control Using Rule Induction

"'Trial and Error' was an archetype of what the knowledge engineering industry sees today as a design platitude: top-down decomposition into subproblems, with a rule-structured solution for each individual subproblem. This is the platitude, or in modern jargon the paradigm, of 'rule-based programming'."

D. Michie (1986)
Introduction, Section 1
On Machine Intelligence

The second, and equally important, focus of this thesis is the use of artificial intelligence to locally control the vibration of a beam. The control of a point on a beam is performed using a rule-induction method called "Boxes" developed by Michie and Chambers (Michie and Chambers, 1968). Once again, vibrational information from the structure is used to identify the optimum local control **in real time** for a specified point on the beam.

In general terms, the investigation first consists of applying the "BOXES" method to a few simple mass-spring systems by way of a simulation. One-mass and two-mass systems are investigated to theoretically evaluate the technique. The second part of the investigation consists of identifying the optimum input to a predetermined location on an aluminum beam. The beam is a 0.94 m (37 in.) by 76 mm (3 in.) by 3.0 mm (1/8 in.) aluminum beam which is simply supported on an interchangeable holding assembly. The excitation is performed using a piezoelectric actuator and the resulting measurements are recorded using strain gage sensors.

Using "Boxes," a real-time machine learning technique, several possible input voltages are evaluated to see which optimize the control at any given time. As developed by Michie and Chambers, "Boxes" originally worked using the discretized states of the system and categorizing them into a matrix (boxes). For each box, different possible inputs to the system are tested as a control and the outputs are used to calculate the effectiveness. Using all this information, the machine can "learn" or evaluate what the best inputs are to the system for any combination of states. Because measuring all the states is very difficult in practice, this investigation utilizes the displacement state at three time steps as the axes for the "BOXES" matrix. The following development shows that these measurements are, in fact, states of the system.

For example, one can begin by describing any second order, multi-degree-of-freedom system as

$$M\ddot{r} + Kr = BU \qquad (3.1)$$
$$y = \phi r$$

where r is a generalized coordinate, M is the mass matrix, K is the stiffness matrix, B is the control matrix, phi is a constant non-singular matrix, and y is the strain output to controlled. These equations can then be put into state space form resulting in

$$\begin{cases} \dot{X} = AX + BU \\ Y = CX \end{cases} \qquad (3.2)$$

where the matrices

$$X = \begin{pmatrix} r \\ \dot{r} \end{pmatrix}, \quad A = \begin{pmatrix} 0 & I \\ -M^{-1}K & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ M^{-1}B \end{pmatrix}, \quad C = \begin{bmatrix} \phi & 0 \end{bmatrix} \qquad (3.3)$$

Assuming a zero order hold on the input(s), this can be transformed into a discrete-time system and obtain

$$\begin{cases} \dot{X}_{k+1} = \Phi X_k + \Gamma U \\ y_k = CX_k \end{cases} \qquad (3.4)$$

Now, using a z-transform, the system looks like

$$\begin{cases} X(z) = (zI - \Phi)^{-1}\Gamma u(z) \\ Y(z) = CX(z) \\ \qquad \downarrow\downarrow \\ Y(z) = C(zI - \Phi)^{-1}\Gamma u(z) \end{cases} \qquad (3.5)$$

which for a single-input, single-output is

$$\frac{Y}{U} = \frac{B(z)}{A(z)} \tag{3.6}$$

Performing a change of basis on equation 3.3 to observable canonical form, the set of equations look like

$$\begin{cases} z_{k+1} = \begin{pmatrix} -a_0 & 1 & 0 & \cdots & 0 \\ -a_1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ -a_{n-1} & 0 & \cdots & 0 & 1 \\ -a_n & 0 & \cdots & 0 & 0 \end{pmatrix} Z_k + \begin{pmatrix} -b_1 \\ \vdots \\ \vdots \\ \vdots \\ -b_n \end{pmatrix} U_k \\ Y_k = \begin{bmatrix} 1 & 0 \cdots & \cdots & 0 \end{bmatrix} Z_n \end{cases} \tag{3.7}$$

Where the a_i are coefficients of the A(z) polynomial and b_i are coefficients of the B(z) polynomial. Thus, one notices that the transformed state vector Z(k) contains delayed values of Y(k) as shown below.

$$\begin{pmatrix} Y_k \\ Y_{k-1} \\ Y_{k-2} \end{pmatrix} = \begin{pmatrix} z_k \\ z_{k-1} \\ z_{k-2} \end{pmatrix} \tag{3.8}$$

Therefore, it has been shown that these measurements are states of the system.

The motivation behind this initial study of vibration mitigation is to show the viability of this method of control. Once this method is successful, this theory of using "Boxes" as a learning technique can be extended to higher order systems. For example, a successful control of a two-dimensional beam gives some experimental cornerstone for applying the method to a three-dimensional plate. These types of higher-order systems are extremely difficult to model and , hence, very difficult to control using classical control techniques.

In the end, the success of this investigation relies upon not only the accurate identification of control inputs to suppress the vibration of the structure but also the ability to perform this damage control in a practical setting.

3.1 'BOXES' Technique

As mentioned above, the 'BOXES' method of optimization is used to control the vibration for a point on the beam, thus controlling the damage. This techniques works by discretizing the time displacement at times (t), (t-1), and (t-2) and categorizing them into a matrix or boxes. For each box, different possible discretized inputs to the system are tested as a control in a trial and failure format and a resulting measure of effectiveness is stored. Using this information, the computer learns over time what the best inputs are to the system for any combination of displacements.

3.1.1 "BOXES" Background

This method of discretizing the system into subproblems was originally performed on a cart and pole system by Widrow and Smith (1964). This system shown in Fig. 3.1 was then adopted by Michie and Chambers (1968) to perform an experiment in artificial intelligence control. This experiment consisted of having a machine learn from a teacher. More specifically, a human practiced in the art of balancing the pole on the cart (simulation) would perform this task and the computer would record the states (cart displacement, cart velocity, pole angle, and pole angular velocity) with an indication of 'O.K.' or 'Fail.' If the input resulted in a failure (i.e., the pole fell past a certain angle or the cart moves a specified distance away from the starting point), it would be labeled with a 'Fail' otherwise it was labeled 'O.K.'. From this technique of quantizing the inputs, the computer could induce thresholds for the boxes and associate the boxes with "L" if the appropriate next move is left or "R" if the appropriate next move is right. In this way, the computer learned the appropriate response to any given situation or box.

Then in 1972, the same pole and cart system was adopted by Michie and Chambers (1972) to perform an experiment with "BOXES" using a slightly different idea. At any given state-situation "box" the computer would begin by randomly applying a left or right (L or R) full force response and recording the time until failure. Each successive time the simulation would hit that state combination, the computer would determine the previous response which averaged the longest time until failure, apply that response, and then reaverage the results. At the end of the simulation, each box was associated with

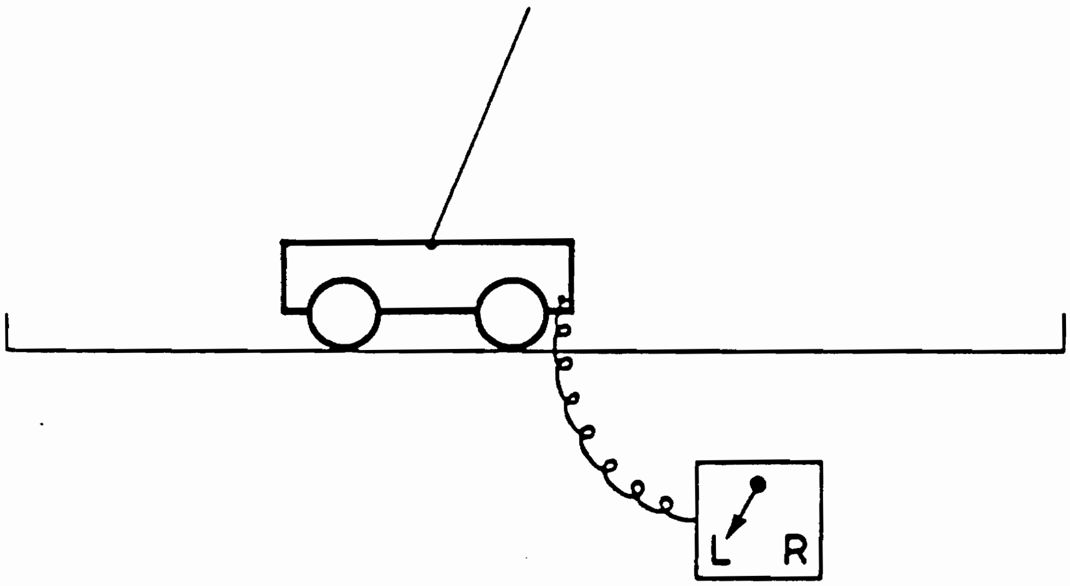


Figure 3.1: Pole and Cart System Used in Original "BOXES" Experiment (Michie and Chambers, 1972)

the input (L or R) that averaged the longest result. In this way, the program learned on its own in real time. Figure 3.2, taken from Chambers and Michie's "'Boxes' as a Model of Pattern Formation," shows that state space representation of the box and the "humanoid slave" who determines which option is the better response. The latter method of having the computer learn on its own is used in this initial study of damage control.

3.1.2 "BOXES" Extensions

There are a few extensions of the original "BOXES" which allow this technique to be applied to the task of vibration mitigation. First, the original experiment performed on the pole and cart by Michie and Chambers only compared two possible inputs: full force left or full force right. In this thesis, the input voltage to the piezoelectric actuator is discretized into nine possible states rather then only two in order to find the optimum input.

Secondly, the vibration systems to be controlled are discretized using only the displacement state at the previous time steps as opposed to present state values. It is rather difficult to directly measure all the states experimentally (as noted by the extensive research being performed on this task alone). Therefore, the author has chosen to use the last three displacement values at any given time t (i.e., X(k), X(k-1), and X(k-2)) to discretize the system. As shown in the development of equation 3.8, the displacement

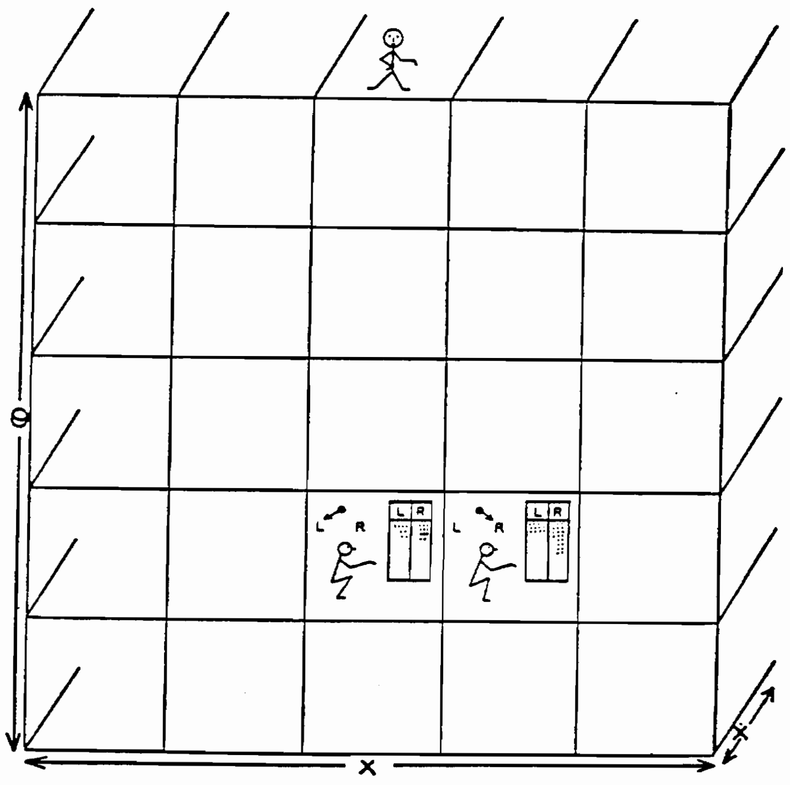


Figure 3.2: The State Space (for Clarity Omitting the $\dot{\theta}$ dimension) divided into "BOXES" with an independent automaton in each box and a 'leader' supervising (Michie and Chamber, 1972)

values at different time steps can be thought of as "states". In addition, the displacement changes or velocity will inherently be taken into account by the learning mechanism.

Thirdly, the performance index used in this experiment is a bit more extensive. In the original "BOXES" experiment performed on the pole and cart system, any given control input had an effectiveness based on the time until failure. In this thesis, the performance is judged on the minimum average displacement for the next three time steps. In other words, the performance index (P.I.) can be described as in the following equation:

$$P.I. = Ave\left[abs(X(k+1)) + abs(X(k+2)) + abs(X(k+3))\right] \tag{3.9}$$

This method not only identifies the optimum input for the present state, but it helps to prevent any "quick fixes" which may hinder the controlling process at a later time step.

A fourth extension exists in the intelligence of the program. Different ideas are introduced to ensure the machine learning program has the best opportunity to learn the optimum input to mitigate the vibrations. To be sure that all the input possibilities are attempted at least once, the performance index is initialized at an extremely low value of 0.001 meters for each input within each box. Another idea which is implemented is a technique used to avoid "control lock-in." Every so often, the averages used for the performance indices are re-initialized thereby giving each subsequent P.I. index more of an impact on the learning process.

Fifth, the original experiment was only performed on a pole and cart simulation not on an experimental testbed. To the author's knowledge, this is the first time the "BOXES" technique is also applied in an experimental fashion.

3.1.3 Discretization Thresholds

As described above, the three displacements used to identify the movement of the beam are X(k), X(k-1), and X(k-2). The input is a measure of voltage sent to the piezoelectric patch. A crucial task in this investigation is the process of discretizing or quantizing the displacements and identifying appropriate thresholds. To do this, an uncontrolled time response for the problem of interest is obtained from the system in hopes of inferring the most logical places to split the subspaces. The time response indicates where the maximum discretizations should occur. The splits used are not necessarily linear. In point of fact, the discretizations closest to the vibration goal (i.e., zero vibration) are more concentrated to maximize the vibration control.

It should be noted that these thresholds are the only aspect of this technique which require human expertise and prevent the machine from truly discovering on its own, albeit learning nonetheless.

Section 3.2 describes the machine learning algorithm used for both the theoretical and experimental investigations.

3.2 Machine Learning Algorithm

The machine learning code and the simulation used in the theoretical investigation are written in Matlab. Matlab is a high-performance interactive software package for scientific and engineering numeric computation created by The MathWorks, Inc. The machine learning code which is used in the experimental investigation is written in C. C is a powerful programming and compiler language available from Borland International, Inc.

Although these experiments use different programming languages, the algorithm is rather similar. The algorithm used for the learning exercises is an intricate loop which continually updates and optimizes the "BOXES". The basic loop consists of performing the following at each time step:

1) Identify the "Box" associated with the present time step by

identifying the appropriate discretized segments for X(k), X(k-1), and X(k-2).

2) Identify which of the discretized control inputs is presently the optimum for this "Box".

3) Apply the control input.

4) Update the "Box" by averaging in the subsequent performance index.

This cycle or updating process continues until the optimum values have been locked in and stay unchanged.

Although the actual program of the separate experiments vary based on the order of the system being controlled or if the technique is being applied theoretically or experimentally, the aforementioned steps are consistent. Some of the variables which change are the threshold values, the number of boxes required for effective control, and the actual control values.

3.3 Matlab Simulations

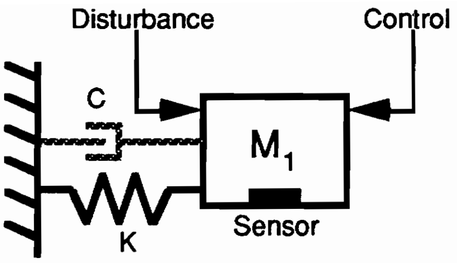
This technique of "BOXES" rule induction is first performed theoretically to examine how well the method works on ideal systems. The machine learning technique is applied theoretically to one- and two-mass, undamped systems with a impulse, step, and sinusoidal disturbances. This corresponds to six different simulations which are evaluated using Matlab software package. Visuals of the two different systems under investigation are shown in Fig. 3.3.

The following section describes the techniques used to simulate the systems and learning algorithms in Matlab.

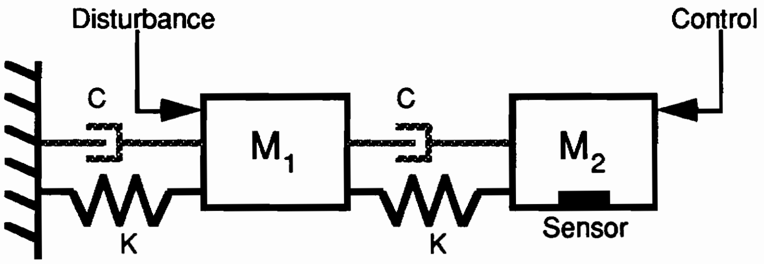
3.3.1 Solution Structure

For all the simulations used to evaluate the learning mechanism, the systems are defined with a unit mass and unit spring constants. The simulation is performed in Matlab using a continuous-to-discrete time conversion. This enables the computer to update the learning mechanism after each time step. For convenience the time step, T, is set equal to one second.

Each of the six simulations use the same "BOXES" matrix. A visualization of this is shown in Fig. 3.4. To avoid a combinatorial explosion, we will limit the number of discretizations to nine mass displacements at time k, nine mass displacements at time (k-



(a)



(b)

Figure 3.3: Spring-Mass-Damper Systems Modelled for Theoretical Investigation: a) 1-Mass Simulation (Damper on Sinusoidal Disturbance), b) 2-Mass Simulation (Dampers on Sinusoidal Disturbance)

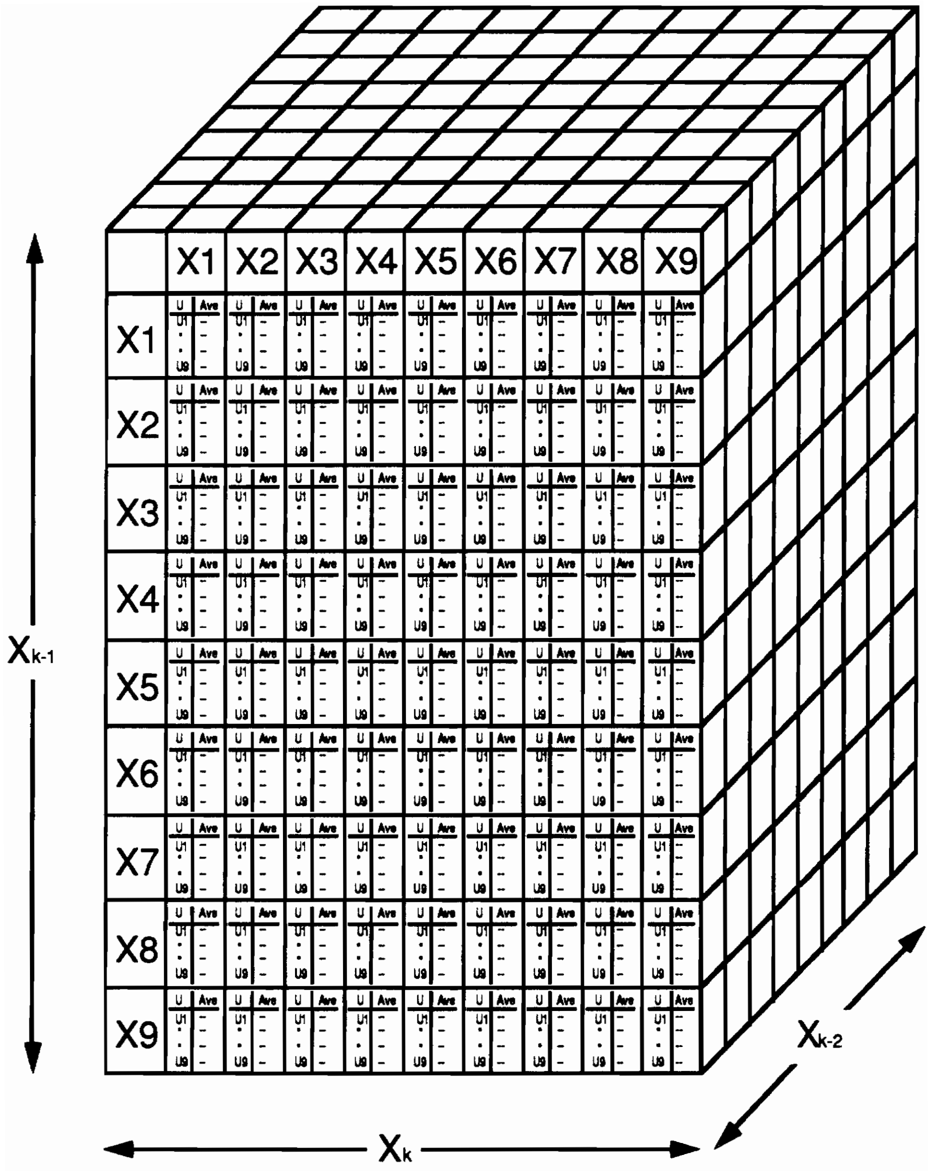


Figure 3.4: Visualization of "BOXES" Matrix Used in Simulations

1), and nine mass displacements at time (k-2). Then, as seen in Fig. 3.4, each of these boxes keep track of nine possible inputs and their average performance index. The Matlab limitation of 8188 elements per matrix prevents us from increasing the definition of the "BOXES" system. The input which results in the lowest average displacement over the next three time steps is considered to be the optimum for that box.

The next step in obtaining a solution is to establish the nine thresholds for the displacements. An uncontrolled response is obtained from the Matlab simulation in hopes of inferring the most logical places to split the subspaces. After examining the response, one notices that to effectively control the vibration, there should be more emphasis on the values occurring around the zero value. The decision on splitting the subspaces in this manner is quite important for effective and accurate rule-induction. If too wide a box is used, states which typically would use a certain optimum input may be lumped with another group.

Having said this, an example of the thresholds to be used for the single degree-of-freedom learning process is shown below:

$$\begin{aligned} & X1 < -10.0; \quad \text{cm} \\ -10.0 < \; & X2 < -5.0; \\ -5.0 < \; & X3 < -3.0; \\ -3.0 < \; & X4 < -1.0; \\ -1.0 < \; & X5 < 1.0; \end{aligned}$$

$1.0 < X6 < 3.0;$
$3.0 < X7 < 5.0;$
$5.0 < X8 < 10.0;$
$10.0 < X9;$

The nine possible control inputs are integers centered about zero. These values are chosen for simplicity and to avoid high energy inputs needed to the system. More specifically: U1 = -4 N, U2 = -3 N, U3 = -2 N, U4 = -1 N, U5 = 0 N, U6 = 1 N, U7 = 2 N, U8 = 3 N, and U9 = 4 N. In all, there are nine displacement groupings for the three past time steps or 729 "BOXES", each keeping track of nine possible inputs.

As mentioned earlier, the three different disturbances (impulse, step, and sinusoidal) are applied to the one and two degree-of-freedom systems. For the impulse and step disturbances, there is a unit spring constant and there is no damping. The reason the damping term is left out is to ensure that the total vibration control is due to the learning technique and not simply due to damping in the structure. On the other hand, for the sinusoidal input, the damping is included and both the spring constant and damping constant are 1.0. A unit mass is used for all the simulations.

The first three simulations are performed on a one mass, single degree-of-freedom system. As seen in Fig. 3.3, the disturbance, control input, and sensor are all collocated at the mass. The "BOXES" rule induction technique is first applied to an impulse disturbance of 50 Newton-Seconds, then to a step disturbance of 10 Newtons, and finally to a sinusoidal disturbance of amplitude 10 and period of 10 seconds. With regard to the step disturbance, it is more realistic to control the vibration to the steady state value of the system. For this reason, the learning mechanism is modified to guide the vibrational pattern to its steady-state value (in this case, a 50 mm displacement).

The other three simulations are performed on a two mass, two degree-of-freedom system. Here the disturbance force is located on mass 1 while the control input and sensor are collocated on mass 2. Once again, the learning mechanism is first applied to an impulse disturbance of 50 Newton-seconds, then to a step disturbance of 10 Newtons, and finally to a sinusoidal disturbance of amplitude 10 and period 10 seconds. Similarly to the one mass system, the mass is driven to the state displacement during the step disturbance excitation.

As a reference, appendix C contains the six programs which are used to simulate, control, and graph the response of each test.

The next section contains the results of the aforementioned simulations and a brief discussion.

3.3.2 Simulation Results and Evaluation

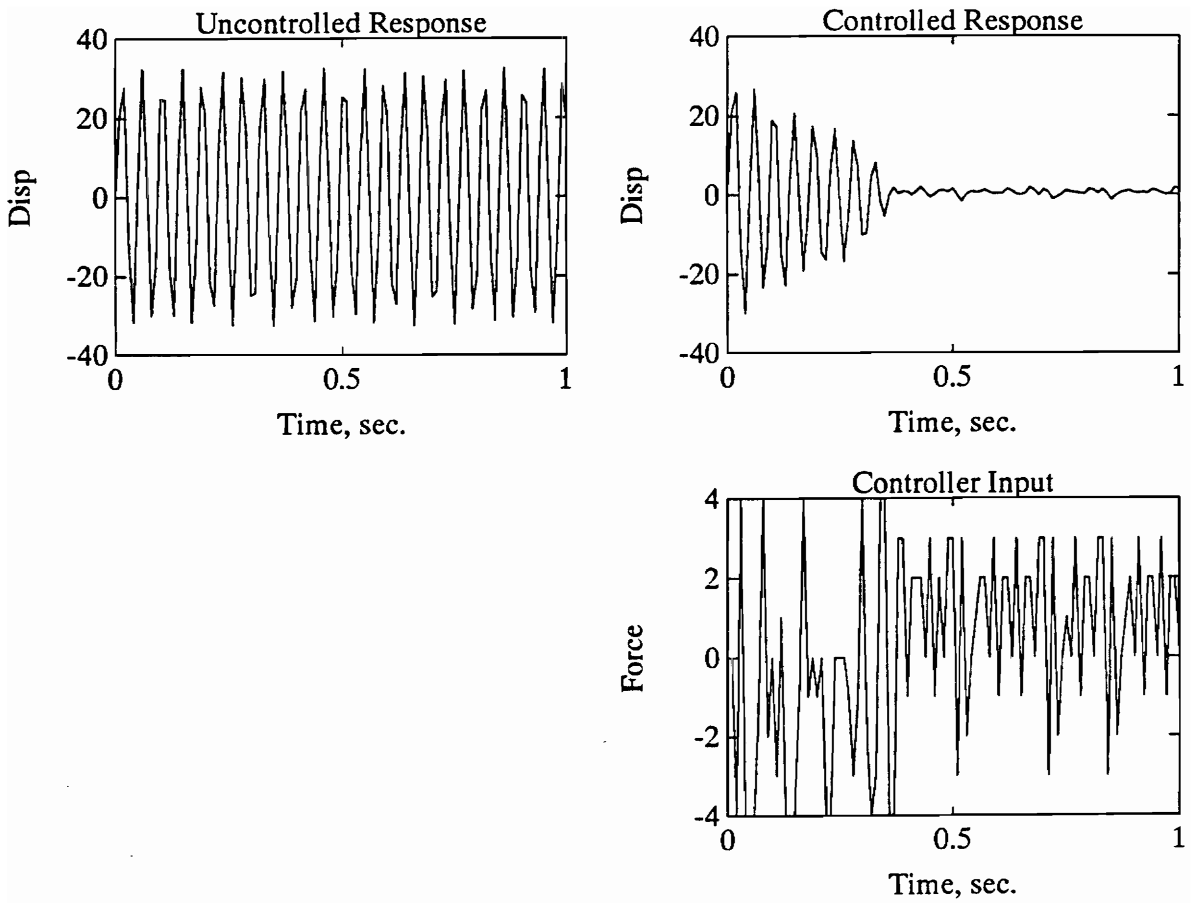


Figure 3.5: "BOXES" Controlled Response for 1-Mass System Simulation with Impulse Disturbance

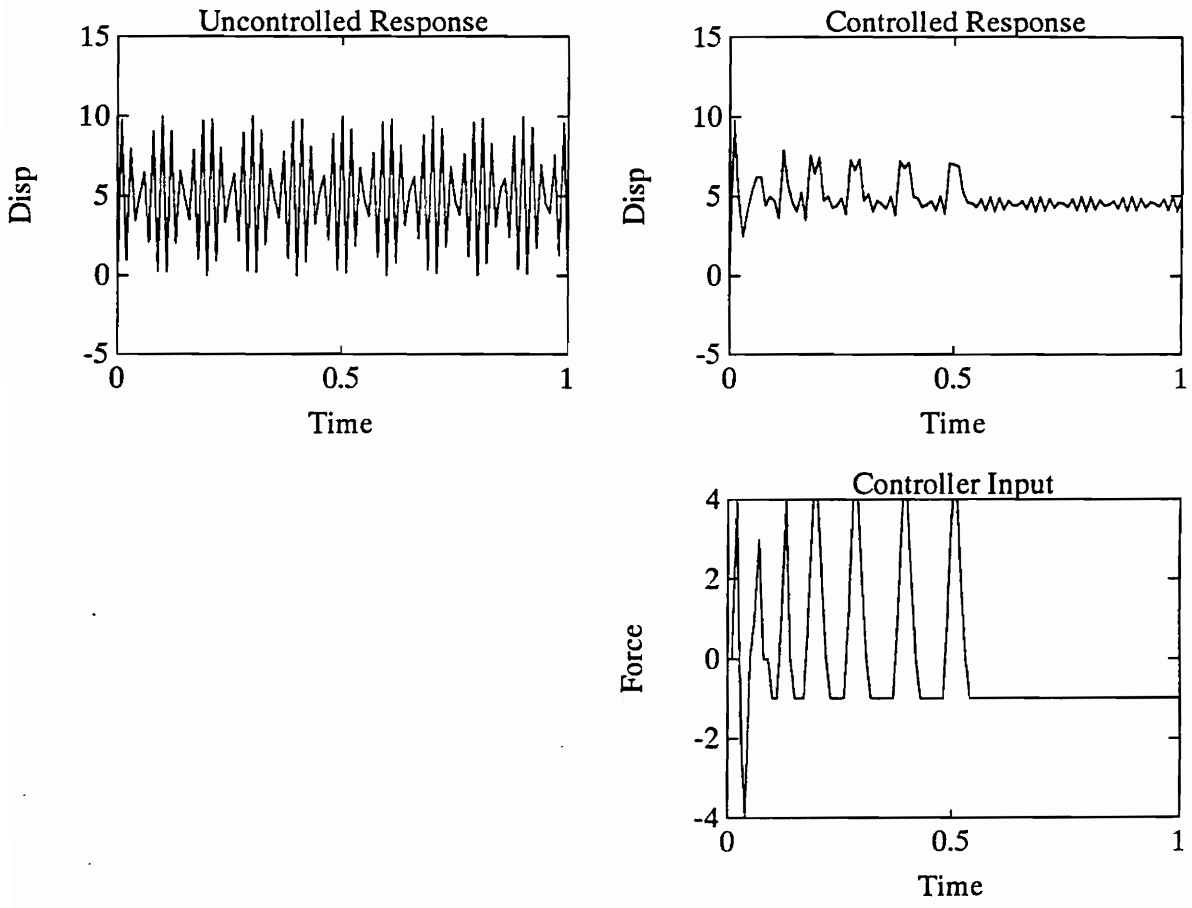


Figure 3.6: "BOXES" Controlled Response for 1-Mass System Simulation with Step Disturbance

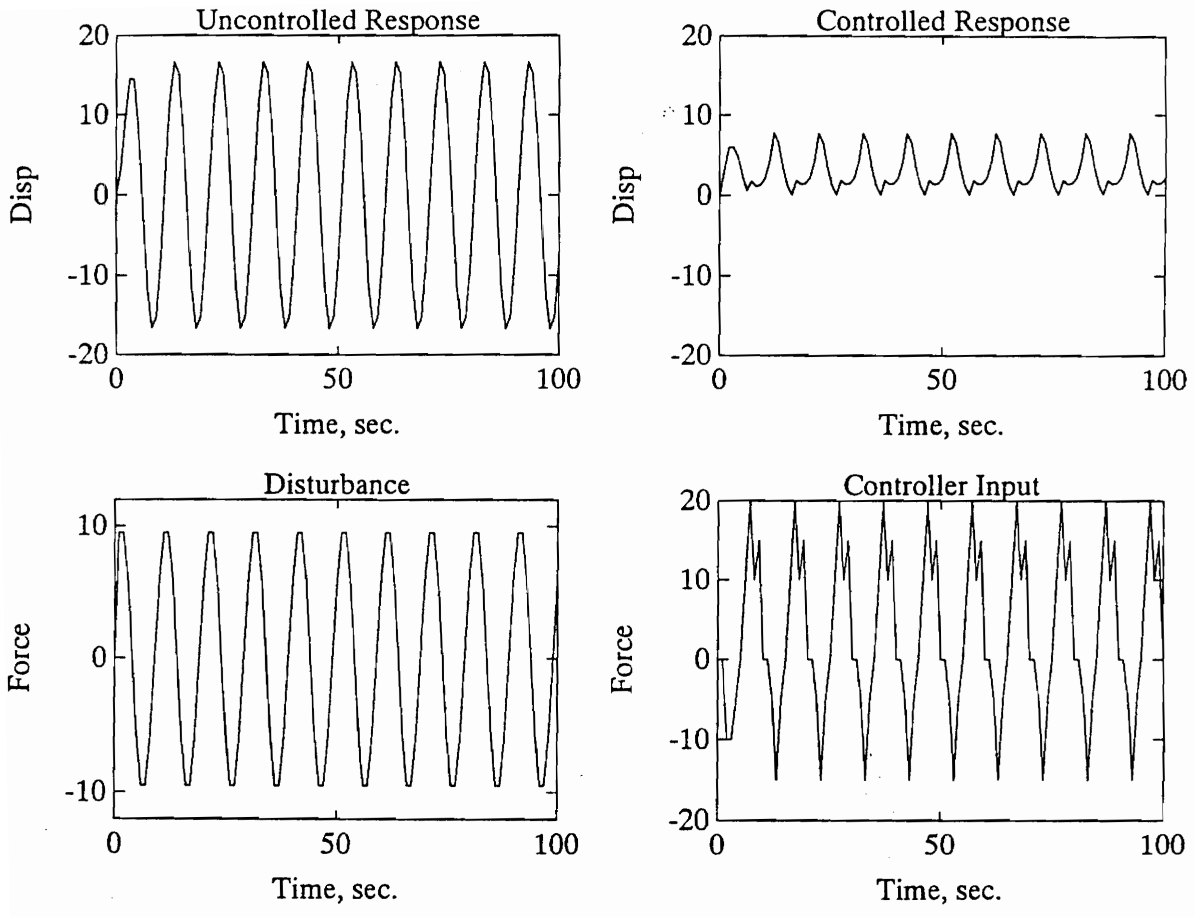


Figure 3.7: "BOXES" Controlled Response for 1-Mass System Simulation with Sinusoidal Disturbance

Figures 3.5, 3.6, and 3.7 show the results of applying the "BOXES" learning technique to the single degree-of-freedom system. Figure 3.5 is associated with the impulse disturbance, Fig. 3.6 is associated with the step disturbance, and Fig. 3.7 show the responses from the sinusoidal disturbance. In each figure, the first graph coincides with the uncontrolled response of the single mass system to the disturbance. The second graph shows the response of the controlled system using the "BOXES" algorithm. The final graph is time response of the controller inputs. The sinusoidal disturbances are shown in a separate graph for those simulations.

Similarly, Figs. 3.8, 3.9 and 3.10 show the results of applying this rule induction technique to the two degree of freedom system. Once again, Fig. 3.8 displays an uncontrolled response, a controlled response, and a controller input time response of the two mass system with an impulse disturbance. Figure 3.9 has the same pattern but relates to a step disturbance. Finally, figure 3.10 adds the sinusoidal disturbance as a special graph along with the other three graphs.

It can be seen from Figs. 3.5, 3.6, and 3.7 that the "BOXES" learning technique successfully controls a single degree-of-freedom system. For the impulse disturbance, the uncontrolled response oscillates from -0.3 to 0.3 m whereas the controlled response oscillates from -0.01 to 0.01 m. This corresponds to a vibration reduction of almost 97%. Likewise, the simulation with the step disturbance produces an uncontrolled oscillation

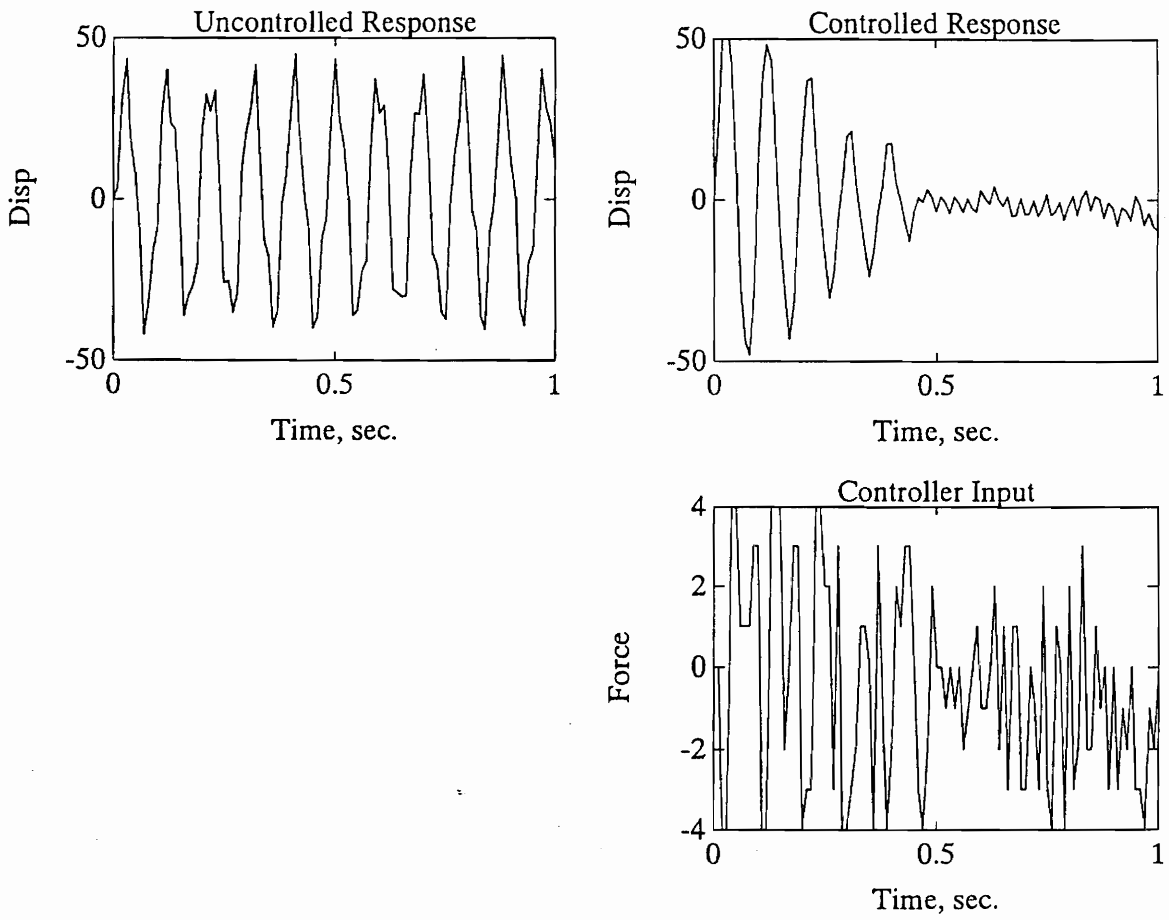


Figure 3.8: "BOXES" Controlled Response for 2-Mass System Simulation with Impulse Disturbance

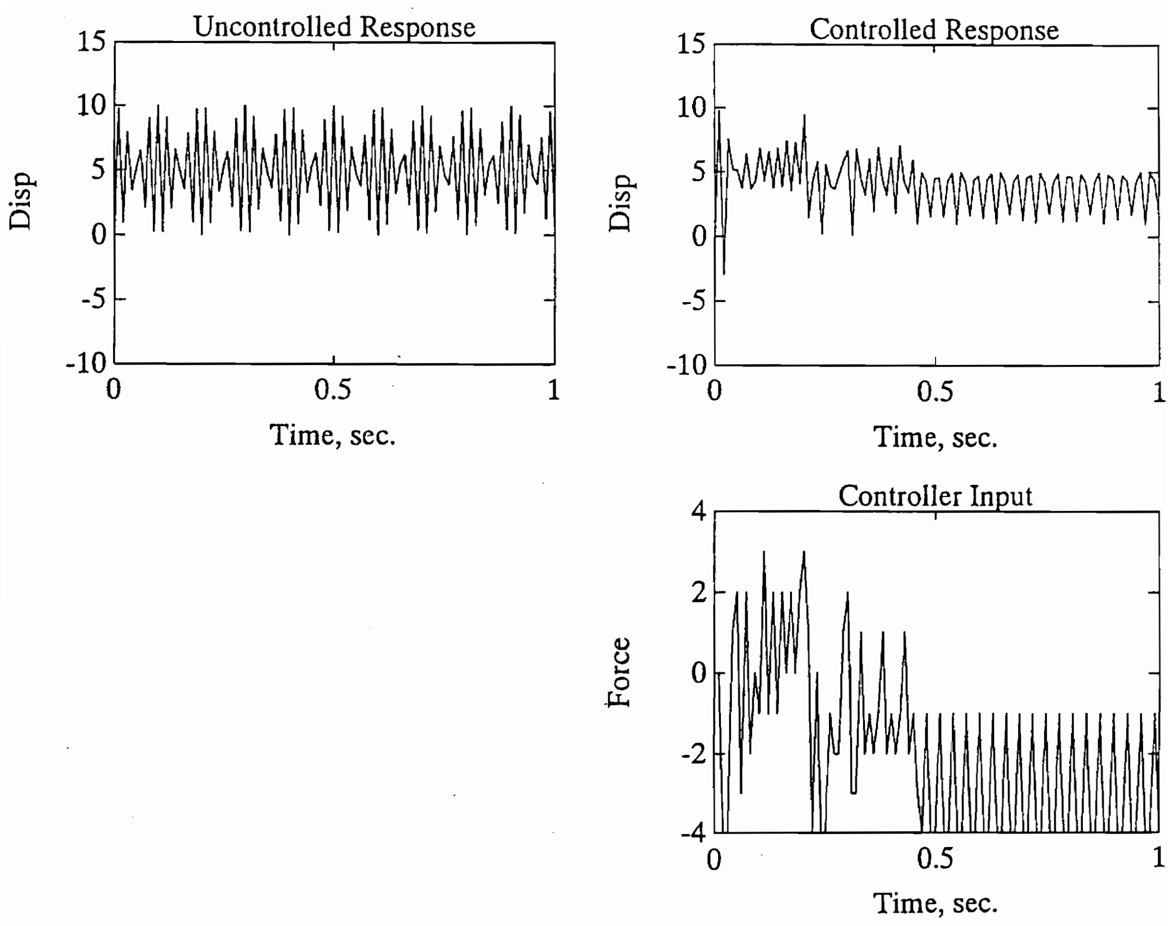


Figure 3.9: "BOXES" Controlled Response for 2-Mass System Simulation with Step Disturbance

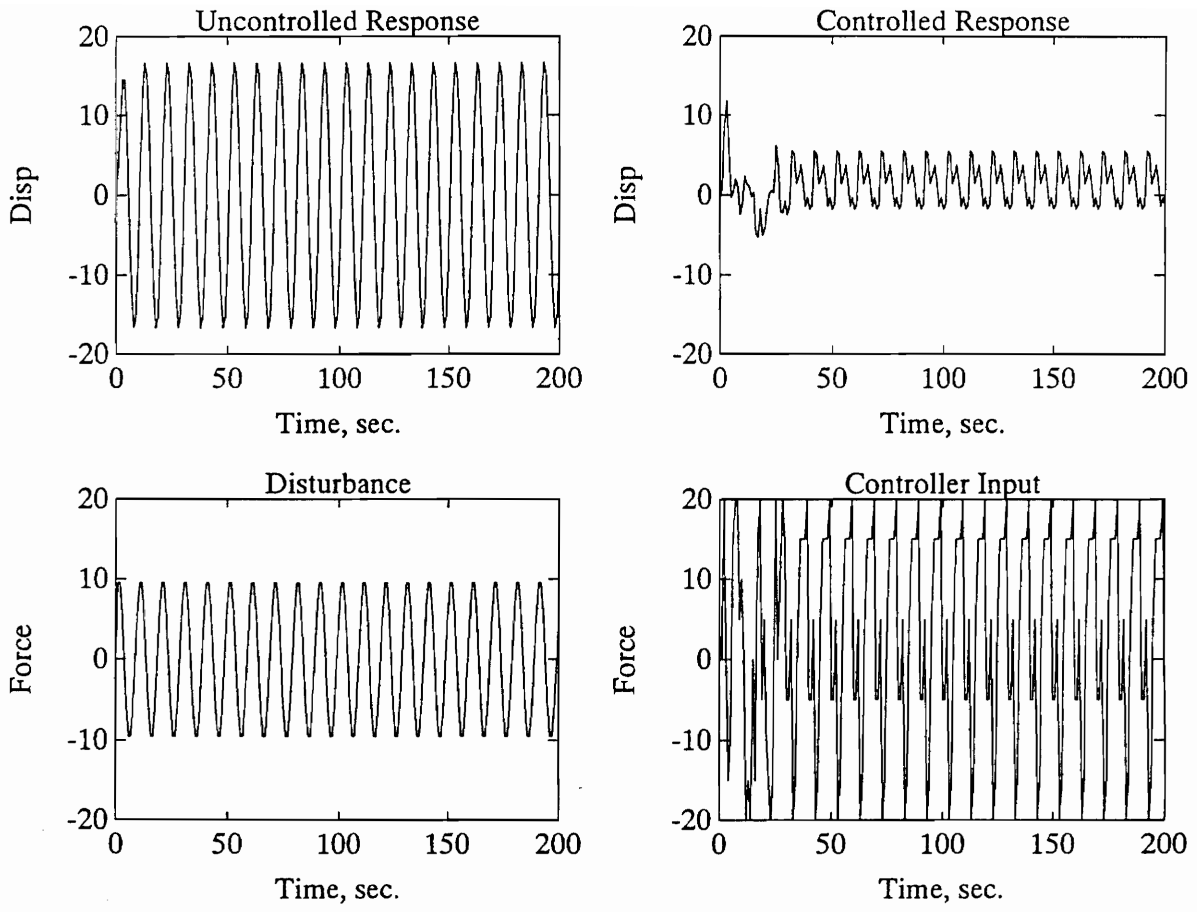


Figure 3.10: "BOXES" Controlled Response for 2-Mass System Simulation with Sinusoidal Disturbance

ranging from about 0 m to 0.10 m (about the steady state value of 0.05 m) and the controlled system only oscillates from 0.045 m to 0.055 m ($\pm$ 5 mm from the steady state value of 0.05 m). This results in a vibration reduction of 90%.

The learning technique does not work quite as well when the system is excited by a sinusoidal disturbance. In the case of the one mass system, the vibration reduces from $\pm$ 0.17 m to a 0.0 m to 70 mm oscillation. This results in a vibration reduction of almost 80%. Although this is a very distinct reduction, one would expect the learning system exactly cancel the sinusoidal input with an equivalent input force 180 degrees out of phase. The reason this did not occur is twofold. First, the lack of input discretizations does not allow the controller to exactly "fit" the disturbance to cancel it. Secondly, It is assumed that the optimum values are affected by subsequent trials. That is to say, although one value may be optimum at time t, if the control value at (k+1) is an extremely poor choice then the "correct" value which was applied at t will have a poor performance index associated with it.

On applying the "BOXES" technique to the two degree-of-freedom system, a substantial vibration reduction again occurs (see Figs. 3.8, 3.9, and 3.10). The impulse response reduces from a vibration oscillation of $\pm$ 0.4 m to $\pm$ 0.02 m (a vibration reduction of 95%). This method is not as effective when applied to the step disturbance. Here the vibration is reduced by about 60%. Again, the sinusoidal disturbance is not controlled as well as expected. "BOXES" again seems to have a problem "homing-in" on the exact, optimum control value even though the vibration is noticeably suppressed. The vibration reduction is about 80%.

It is evident from this investigation that the learning mechanism, as applied here, is less effective for more complex disturbances. The vibrational patterns in for higher order systems and more complex disturbances are harder to identify, and hence tougher for the "BOXES" rule induction technique to control.

Some modifications could be added to the learning algorithm to increase the effectiveness of the learning process. First, more discretizations for each time displacement could be added, thereby increasing the definition of the matrix and allowing the learning mechanism to identify more complex vibrational patterns. Also, increasing the number of possible control values allows the machine to identify the most accurate input for the present situation. Other options are to increase the "dimension" of the boxes system and take into account another time displacement at (t-4) or to use weights when defining the performance index.

3.4 Beam Experiment

Having shown that the "BOXES" technique of rule induction can mitigate vibration relatively well for ideal systems, the next logical step is to apply the method to an experimental system. Using this same "BOXES" technique, the optimum input to control the vibration of an aluminum beam is learned. The motivation in this part of the thesis is to show that this machine learning technique can implicitly identify the vibrational patterns of a practical system thereby controlling the vibration of a beam in real time.

In general terms, the experiment consists of exciting an aluminum beam using a piezoelectric actuator located three-fourths of the way along a beam. A strain gage and a piezoelectric control patch are collocated at the possible "damage" site which in this case is taken to be one-third of the way down the beam. These locations are chosen to ensure the observability of the first five plate vibration modes by avoiding a node. The beam is a 0.94 m (37 in.) by 76.0 mm (3 in.) by 3 mm (1/8 in.) aluminum beam which is situated in a simply-supported manner on an interchangeable holding assembly.

A schematic of the experimental setup is shown in Fig. 3.11. The output voltages from the strain gages are routed through a strain-gage amplifier and a low-pass filter to amplify the response signal. This signal is then accepted by a 386/387 computer using an interface box. This computer contains the most vital part of the data acquisition process - the transputer-based data acquisition board (TransDAC) (Ellis, 1990). The control system architecture of the TransDAC is shown in Fig. 3.12. The T222 transputer performs the data acquisition and control output while the T800 transputer performs the

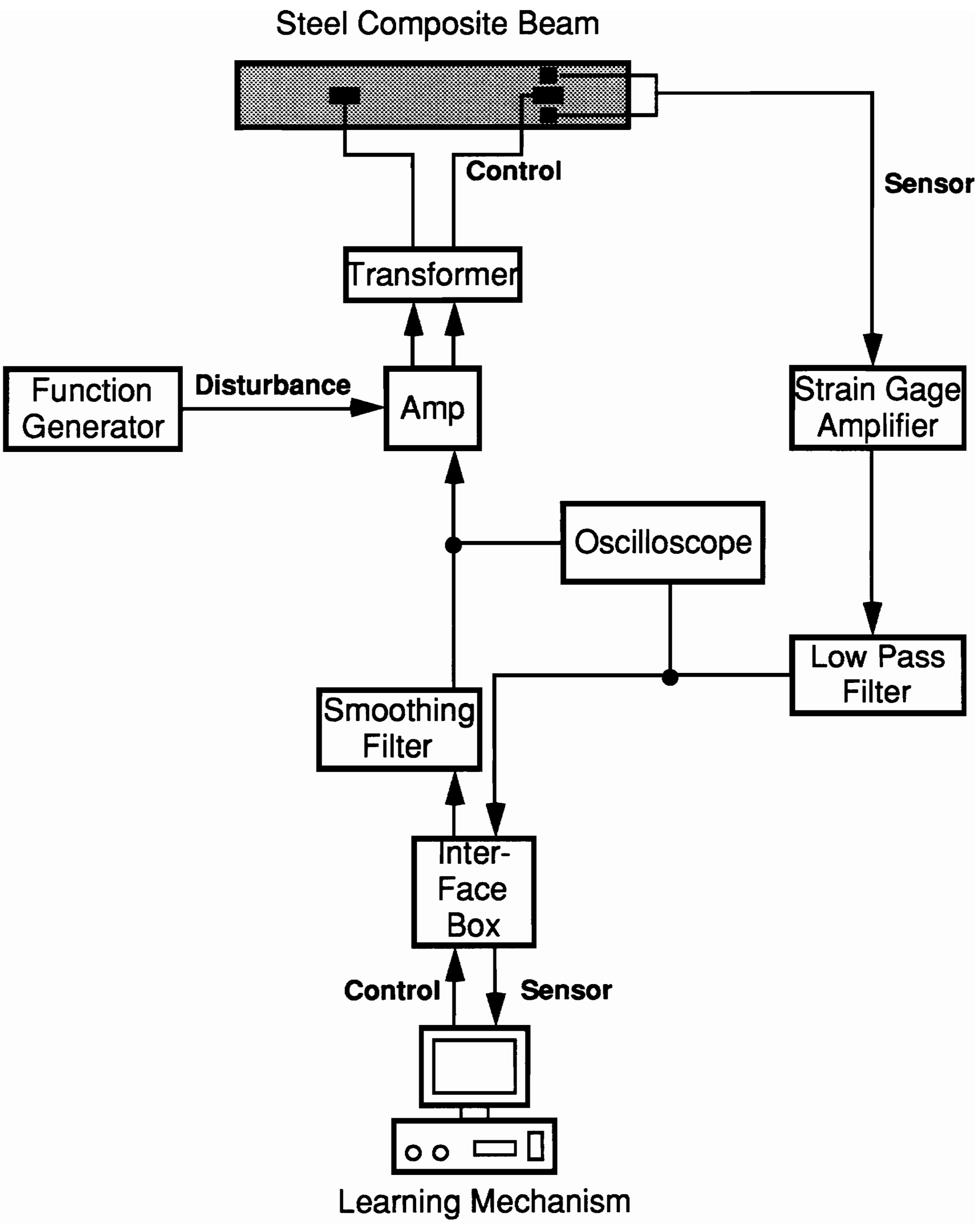


Figure 3.11: Flow Diagram for "BOXES" Learning System

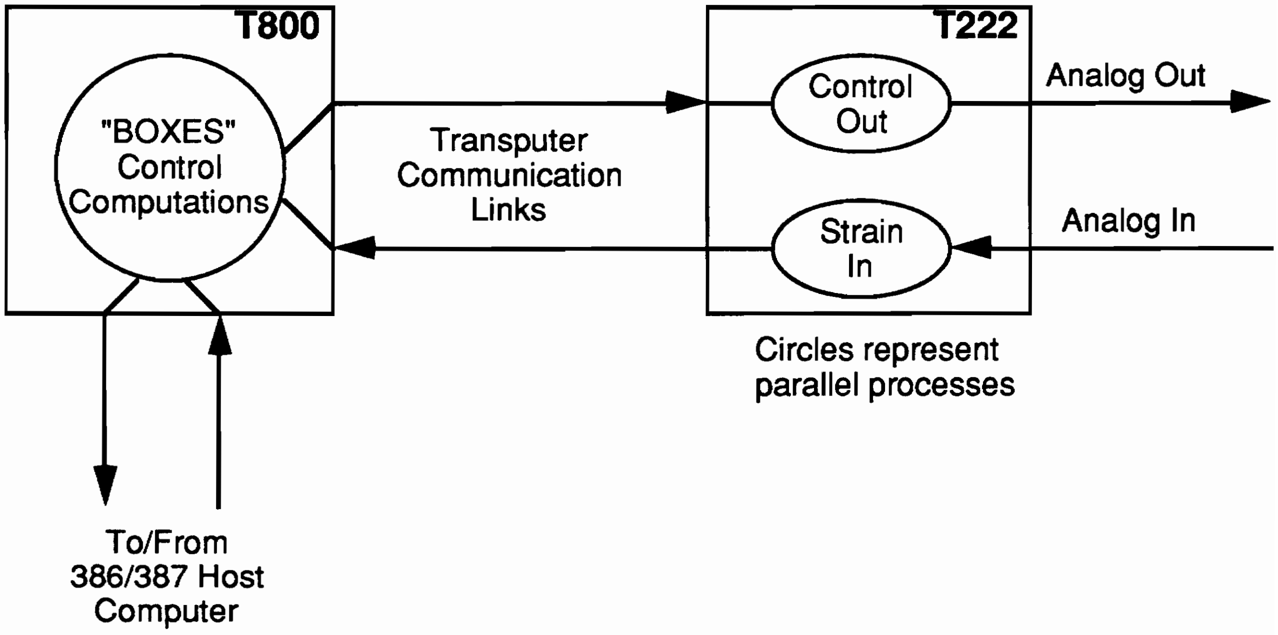

T800
"BOXES" Control Computations
Transputer Communication Links
T222
Control Out
Strain In
Analog Out
Analog In
Circles represent parallel processes
To/From 386/387 Host Computer

Figure 3.12: Control System Architecture

control law calculations. These two processes are performed in parallel and only communicate when the control law (located in the T800) needs an input or when it is ready to send the control voltage. In addition, the T222 transputer controls the analog-to-digital (A/D) conversions for the input voltages from the strain gage and digital-to-analog (D/A) conversions for the output voltages to the control patch (Rubenstein, 1991). The "BOXES" control law contained in the T800 transputer code is discussed in the next section.

Once the output signal from the control law is "DACed" out via the T222 transputer, it is again run through the interface box. From there the signal is sent through a smoothing filter to avoid damaging the piezoelectric patch from large voltage jumps. As shown in Fig. 3.11, the control voltage is amplified using a standard, constant-gain amplifier and a transformer. The signal is then sent to the control patch.

The final part of the experimental setup is the disturbance input. The disturbance is a sinusoidal wave of amplitude 650 millivolts and frequency 34.7 Hz. This frequency corresponds to the first visible mode of the beam. This signal is also run through the amplifier and transformer before being sent to the disturbance patch.

The next section explains the solution structure of the experimental tests.

3.4.1 Solution Structure

Although the same "BOXES" technique performed on the simulation is used in this experimental investigation, there are quite a few details which need to be addressed before the method can be applied. In addition, the solution structure of the experimental investigation is different from the theoretical investigation because the control law is implemented in a different programming language. The control technique for this investigation is performed in Turbo C. C is a powerful programming and compiler language available from Borland International, Inc.

One difference between this study and the simulation is the value to be discretized. In the case of the beam, the strain gage sensors produce a voltage which is proportional to the displacement. Instead of converting this value to a displacement and back again for the control we use the voltage itself by which to define "BOXES" matrix. The theory being that once the voltage from the sensor is controlled, the displacement is also controlled. The number of discretizations for the voltage are still limited to nine (as seen in Fig. 3.4). This time each of the boxes keep track of 15 possible inputs and their average performance indexes. The number of inputs is increased here to allow for more accurate controls while the learning algorithm is searching for the optimum.

Another important aspect of the solution structure is the definition of the performance index. This investigation uses the same performance index as the simulations (the displacement over three subsequent time steps). That is to say, the performance index

$$P.I. = Ave\left[abs(X(k)) + abs(X(k+1)) + abs(X(k+2)) \right] \tag{3.10}$$

This performance index allows the evaluation of a possible control over a period time.

As in the simulations, the nine sensory thresholds are inferred by examining an uncontrolled response of the beam from the aforementioned 650 millivolt, 34.7 Hz disturbance. To more effectively control the beam, more quantizations occur about the zero voltage value. Also, to ensure a high concentration of quantizations near the zero-value strain voltage, the maximum thresholds value will be kept to ± 1.0 volts.

The actual discretizations used in the experiment are shown below:

$$\begin{aligned}
&V1 < -1.0; \quad \text{Volts}\\
-1.0 < \ &V2 < -0.5;\\
-0.5 < \ &V3 < -0.3;\\
-0.3 < \ &V4 < -0.1;\\
-0.1 < \ &V5 < 0.1;\\
0.1 < \ &V6 < 0.3;\\
0.3 < \ &V7 < 0.5;\\
0.5 < \ &V8 < 1.0;\\
1.0 < \ &V9;
\end{aligned}$$

The 15 possible control inputs are also centered about zero. These values range in equal intervals from about -1.5 volts to +1.5 volts and, after being stepped up by the amplifier and transformer, become a control input of $\pm$ 90 volts to the beam.

In order for the algorithm to successfully learn the vibration patterns, each control calculation must take the same amount of time each time. Otherwise, different control inputs might have different performance indexes. This problem might occur when the voltages are discretized with the If-Then rules (see program contained in Appendix D). To avoid this problem, a dummy-loop is created to "stall" the program so that each loop takes exactly the same amount of time.

Finally, the C program contains two output vectors which are stored as Matlab data files. The first is the strain vector and the second is the control vector. The program proceeds through 5,000 loops to locally control the vibration and every fifth point voltage input and voltage control is saved to the output file. This allowed for a graphic display of how well the program is learning over time. Special care is taken to ensure there is no aliasing during the data acquisition.

As a reference, Appendix D contains the T800 code written in C (which contains the "BOXES" control program) and the T222 code written in C (which contains the input and output commands from the computer to the system).

The next section presents the results of this study along with some comments.

3.4.2 Experimental Results and Evaluation

Figures 3.13 shows the results of applying the "BOXES" learning technique to the beam excited at 34.7 Hz with a magnitude of 650 millivolts. The system runs at about 1900 cycles per second and takes about 2.6 seconds to complete the learning process. The x-axis on the figure corresponds to the time at which the 1000 strain samples were taken. The y-axis corresponds to the strain in volts.

As seen in Fig. 3.13, three separate test runs were performed for the **same** specifications. Multiple runs were tested to ensure a true pattern of learning and not simply a "fluke" occurence. The first graph shows the uncontrolled response while the other three correspond to three separate learning tests.

The strain from the uncontrolled response oscillates between +2.1 and -2.3 volts. On the other hand, the controlled responses only oscillate to $\pm$ 1.5 volts. This corresponds to a 32% reduction in vibration. Although the program successfully learned to reduce the vibration, it is not effective enough for a legitimate control law. Several modifications were performed on the learning program in hopes of finding a "simple" modification which would make a considerable improvement in the results.

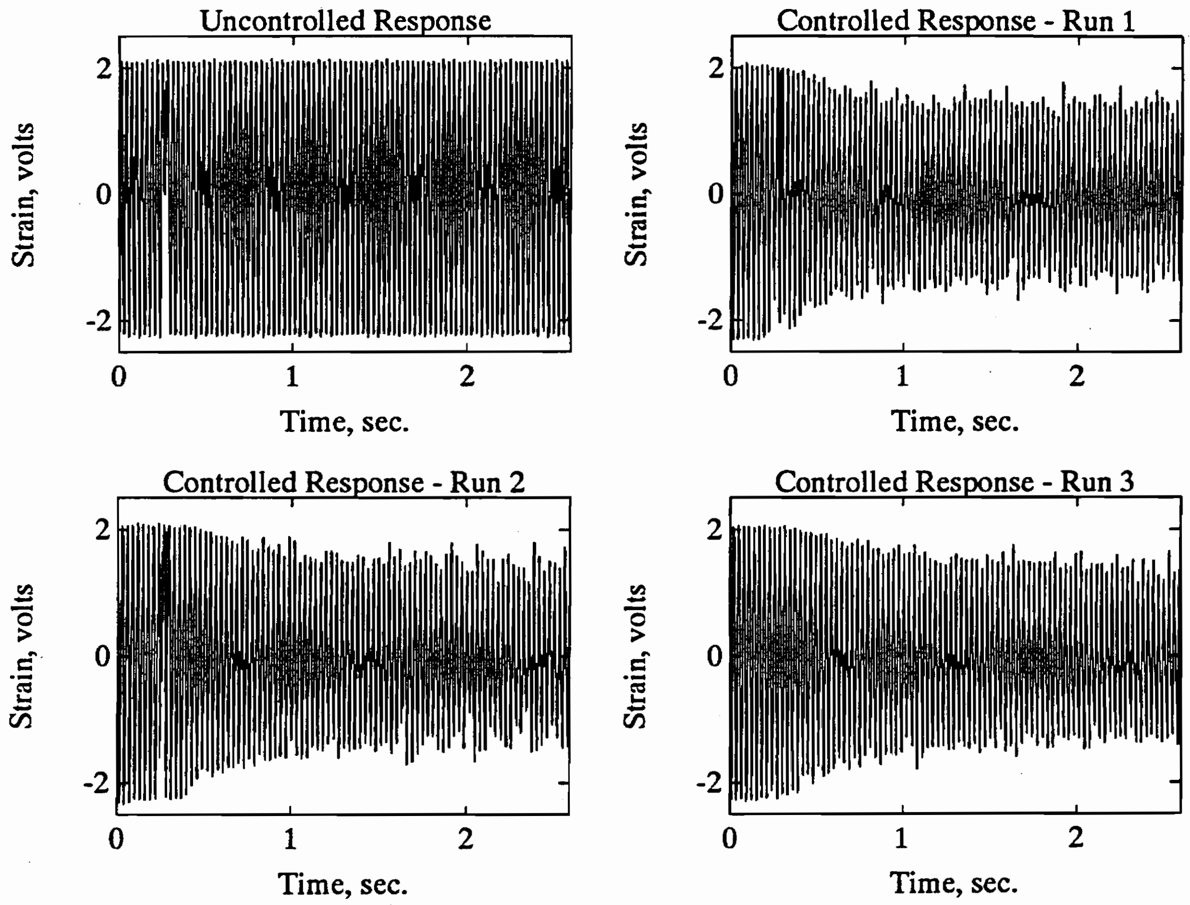


Figure 3.13: "BOXES" Controlled Response for Aluminum Beam with Sinusoidal Disturbance: a) Uncontrolled Response, b) Test Run 1, c) Test Run 2, d) Test Run 3

One possible improvement may be the elimination of time variations in the testbed. As it stands, the beam is situated with pinned-pinned boundary conditions where the nature of the clamps allows shifts during testing.

With regard to improving the actual learning process, each of the variables associated with the "BOXES" algorithm could be individually optimized for this particular experiment. This would include the number of quantizations, the range of possible control inputs, and the performance index. All of these factors have unlimited possibilities and optimizing each would further decrease the beam vibration.

In addition, the dependence of the control inputs could be eliminated. For example, if one voltage value is used as a control, the next three voltages over time (which are used to calculate the performance index) have an impact on the original control. Therefore, a control input at one time step may be adversely affected by a control input at the next time step.

The conclusions and future work sections contain some other possible theories on what sorts of work would be required to better control the vibration of this system in real time.

Chapter 4

Conclusions and Recommendations for Future Work

The stated objective of this thesis is to investigate the viability of using Artificial Intelligence or, more specifically, machine learning for active damage control. On a lower level, the purpose is to investigate the use of Artificial Intelligence in successfully accomplishing the two major tasks associated with this broad field of active damage control, i.e., damage identification and damage (vibration) control.

The following sections separately evaluate the success of these initial studies and reveal some ideas for further investigation within each preserve. Finally, some conclusions are made on whole about using this theory of Artificial Intelligence for active damage control.

4.1 Damage Identification

The first part in this two-part initial study was to investigate the viability of identifying damage using an Artificial Intelligence method called rule induction. A set of velocity data was acquired from three plates: one with no damage, one with damage at the center, and one with damage at the edge. A set of rules were then induced from these data by recursively partitioning the data into mutually exclusive, exhaustive subsets that best describe the dependent variable (damage).

This first study successfully developed a set of rules that describes the data. In addition, it was showed that rules can be induced which identify "sensitive" areas on the plates which show obvious distinctions based on damage location.

From a more general perspective, we have developed a cornerstone for believing that this rule induction method, given information about an undamaged structure and a damaged structure, can be used to identify "critical" areas on the structure. These areas can be kept as a reference for test locations during future non-destructive damage testing thereby identifying existence and/or location of damage.

There is one drawback, however, to the idea of implementing this technique of damage identification in practical situations. The fact is one can only identify damage in as many locations as there are reference data. That is to say, a set of applicable reference data is required for every damage location for which prediction is desired. It is possible that there are locations on structures which are consistently affected independent of the damage location. In this case, the technique would be invaluable for standard maintenance inspections.

There are some areas that require further investigation to ensure this method is accurately predicting structural differences which solely result from damage. This investigation applied the rule induction technique to one data set. Although this technique successfully developed a rule set describing the data set, the actual rules themselves have little statistical significance. Future experiments should be performed in a similar fashion using several data sets. This will avoid two rather critical problems. First, from a mathematical standpoint, it will increase the statistical significance of the rules. The more data sets used the more significant the rules will be. Second, using several data sets will ensure that the velocity changes solely result from the damage and not from variations inherent in different set-ups (e.g., temperature changes, slight boundary condition changes, external acoustical effects, etc.). It should be noted that the changes in dynamic response due to set-up variations are on the same order of magnitude as the changes due to damage. The following paragraph outlines an experiment which might remedy these possible problems.

For example, during the data acquisition process, the grid on the three plates could be reduced from a 30 x 30 point data grid to a 10 x 10 matrix. Then, nine data sets could be used in the rule induction process without increasing the memory requirement. In other words, the laser data acquisition process would be performed nine times for each plate. After the rules are induced from the data, another subsequent set of data could be recorded to test the accuracy of the rules. These should be the first steps in future research endeavors.

Secondly, an in-depth study can be performed on how different frequencies affect the "sensitive" areas as defined by this initial study. Is it of more importance to compare the same frequency of two different structures or the same mode? Also, experiments can be performed at different shaker forces to identify it the relationship between force and velocity is linear.

Thirdly, this method of damage identification should also be applied to a single plate: first without damage and then to the same plate after inflicting damage. This would ensure vibrational changes in the structure are caused solely by the damage and not by material inconsistencies inherent in different structures.

4.2 Damage Control

The second part of this initial study of active damage control investigated the viability of using rule induction for damage control. A machine learning technique called "BOXES" was performed on various theoretical and experimental systems. In these investigations, "BOXES" is used as a rule induction technique aimed at controlling structures by identifying their vibrational patterns.

At first, the technique was applied to ideal systems by way of a simulation. One and two degree-of-freedom systems were successfully controlled after having been excited by impulse and step disturbances. Admittedly these are rather simple systems but the idea here is that the algorithm learned an effective control law using its trials and failures as the only information from which to organize this stimulus-response pattern.

In addition, this machine learning method satisfactorily suppressed the steady-state vibration of both the one and two degree-of-freedom systems when excited by a sinusoidal disturbance.

On the other hand, using "BOXES" in an experimental setting was a bit less effective. This method was applied to a vibrating beam which was excited at its first visible mode. It is easy to hypothesize why this method did not work as effectively when applied in an experimental fashion, but there is probably no one explanation. In fact, an entire study could be performed specifically addressing the idea of optimizing the technique. As discussed in section 3.4.2, the possibilities include: noise to the system, the time-varying system, "control dependency", or a sub-optimum performance index definition.

In addition, it has not been shown that the learning mechanism can find the optimum solution **exactly**. Theoretically, this method learns sufficient control laws to suppress vibration in excited structures (as seen by the simulations) but is unable to consistently "nail" the optimum control. In other words, the technique is capable of finding the "local minimum" but not necessarily the optimal solution. The most logical reasoning for this is this theory of "control dependency." That is, the performance index of the optimum control for any given state situation can be adversely affected by the next trial control. Hence, what should be the optimal control at a particular time has become a "poor" choice.

As would be expected, there are innumerable possibilities for future research in applying "BOXES" to suppressing vibration. For one, a detailed study should be performed on the performance index and its implications to the learning process. This performance index is the "brains" in the machine learning technique and defining the best equation to use for different tasks will go a long way toward successfully controlling damage.

A second idea for future investigation would be to apply "BOXES" to a simulation of a vibrating beam. Understanding how well this technique can be applied to an ideal system for different disturbances may shed some light on how the time variations of the experimental investigation affect the learning process.

In short, rule induction and artificial intelligence in general are very powerful methods for problem solving and their uses will no doubt continue to grow in the future.

References

Barto, A.G., Sutton, R.S., and Anderson, C.W. (1983). "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems". *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, Number 5. September/October 1983, 835-846.

Baz, A., Poh, S., Fedor, J. (1989). "Independent Modal Space Control with Positive Position Feedback". *Proceedings of the 7th Conference on the Dynamics and Control of Large Structure*, May 1989, VPI&SU, Blacksburg, VA, 553-567.

Cawley, P. and Adams, R.D. (1979). "The Location of Defects in Structures from Measurements of Natural Frequencies". *Journal of Strain Analysis*, Vol. 14, Number 2, 49-57.

Charniak, E. and McDermott, D. (1985). <u>Introduction to Artificial Intelligence</u>. Addison-Wesley Publishing Company, Reading, Massachussetts, 6-7.

Ellis, G.K. (1990). "Design and programming of a medium performance data acquisition system based on transputers". Submitted for publication as a Center for Intelligent Material Systems and Structures internal report, VPI&SU, Blacksburg, VA.

Fanson, J.L. and Caughey, T.K. (1990). "Positive Position Feedback Control for Large Space Structures". *Proceedings of the 28th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Monterey, CA, 717-724.

Goh, C.J. and Caughey, T.K. (1985). "On the Stability Problem Caused by Finite Actuator Dynamics in the Collocated Control of Large Space Structures". *International Journal of Control*, Vol. 41, Number 3, 787-802.

Gomes, A. and Silva, M. (1990). "On the Use of Modal Analysis for Crack Identification". *Proceedings of the 8th International Modal Analysis Conference*, Kissimmee, FL., 1108-1115.

Hanagud, S. et. al. (1992). "Active Control of Delaminations in Composite Structures". *Proceedings of the 33rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Dallas, TX, 1819-1829.

Kiel, D.H. and Liu, Y. (1992). "Signal Processing as a Method of Damage Identification". Unpublished paper for the ME Signal Processing course (ME 5714) at Virginia Polytechnic Institute and State University. Copies may be obtained from David Kiel, Mechanical Engineering Department, VPI&SU, Blacksburg, Va, Tel: (703) 231-2906.

Li, S. (1992). "Modelling of Active Crack Damage Control and the Active Fatigue Damage Control of Adhesive Joint". Master's Thesis, Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (1983). Machine Learning, An Artificial Intelligence Approach. Tioga Publishing Corporation, Palo Alto, California, 3-11.

Michie, D. (1992). Artificial Intelligence notes for CS Artificial Intelligence course (CS/Stat 5984) at Virginia Polytechnic Institute and State University. Copies may be obtained from David Kiel, Mechanical Engineering Dept., VPI&SU, Tel: (703) 231-2906.

Michie, D. and Chambers, R.A. (1968). "BOXES: an experiment in adaptive control". *Machine Intelligence 2.* Oliver and Boyd, Edinburgh, 137-152.

Michie, D. and Chambers, R.A. (1972). "'Boxes' as a model of pattern formation." *Towards a Theoretical Biology, Vol. 1.* Oliver and Boyd, Edinburgh, 206-215.

Miller, D.W., Hall, S.R. and von Flotow, A.H. (1989). "Optimal Control of Power Flow at Structural Junctions". *Proceedings of the American Control Conference*, Pittsburgh, PA.

Mitchell, T.M. (1977). "Version Spaces: A Candidate Elimination Approach to Rule Learning". *International Joint Conference on Artificial Intelligence, Vol. 1,* MIT, Cambridge, Massachussetts. August 22-25, 1977; 305-310.

PC - MATLAB User's Guide, 1989, The MathWorks, Inc., South Natick, MA.

Poh, S. and Baz, A. (1990). "Active Control of a Flexible Structure Using a Modal Positive Position Feedback Controller". *NASA Technical Report Number N90-17371*, 31 pages.

Pospelov, G.S. and Pospelov, D.A. (1979). "Influence of Artificial Intelligence Methods on the Solution of Traditional Control Problems". In *Machine Intelligence 11.* Oxford University Press, Oxford, 435-454.

Richardson, M.H. and Mannan, M.A. (1992). "Remote Detection and Location of Structural Faults using Modal Parameters". *Proceedings of the 10th International Modal Analysis Conference*, San Diego, Ca, 502-507.

Rogers, C.A. et. al. (1991). "Smart Materials, Structures and Mathematical Issues for Active Damage Control". Research Proposal, CIMSS, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Rogers, C.A., Liang, C., and Li, S. (1991). "Active Damage Control of Hybrid Material Systems using Induced Strain Actuator". *Proceedings of the 32nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Part 2, Washington, DC, 1190-1203.

Rosenblatt, F. (1962). Principles of Neurodynamics. Spartan Publishers, New York, New York.

Rubenstein, S.P. (1991). "An Experiment in State-Space Vibration Control of Steady

Disturbances on a Simply-Supported Plate." Master's Thesis, Mechanical Engineering Department, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Samuel, A.L. (1959). "Some Studies in Machine Learning Using the Game of Checkers". In *IBM Journal, Research and Development*, 3, 211-229.

Turbo C User's Guide, 1992, Borland International, Inc., Scotts Valley, CA.

von Flotow, A.H. and Schafer, B. (1986). "Wave-Absorbing Controllers for a Flexible Beam". *Journal of Guidance, Control, and Dynamics*, Vol. 9, Number 6, November-December, 1986, 673-680.

Widrow, B. and Smith, F.W. (1964). "Pattern recognising control systems". *Computer and Information Sciences*, Clever Hume Press, 288-317.

Winston, P.H. (1970). "Learning Structural Descriptions from Examples". *Technical Report AL-TR-231*, MIT, Cambridge, Massachusetts, September, 1970.

Wolff, T. and Richardson, M. (1989). "Fault Detection in Structures from Changes in Their Modal Parameters". *Proceedings of the 7th International Modal Analysis Conference*, Las Vegas, Nevada, 87-94.

APPENDIX A

```
% This set of generic rules corresponds to the decision tree in Figures
% 11-13. There are 71 rules listed here which accurately describe 77.4%
% of the data. The "interesting" rules are the ones which yield correct-
% ly predicted data 100% of the time.


RULE_1 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [-3.609,-2.263)
THEN
        Damage = Center  100.0%


RULE_2 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [-2.263,-1.435)
THEN
        Damage = Center   28.6%
        Damage = Edge     71.4%


RULE_3 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [-1.435,-0.785)
THEN
        Damage = Edge    100.0%


RULE_4 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [-0.785,-0.276)
THEN
        Damage = Edge    100.0%
```

```
RULE_5 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [-0.276,0.086)
THEN
        Damage = Edge     100.0%


RULE_6 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [0.086,0.487)
THEN
        Damage = Edge     100.0%


RULE_7 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [0.487,0.883)
THEN
        Damage = Edge     100.0%


RULE_8 IF
        Vel(Re) = [-3.779,-1.761)
        Vel(Im) = [0.883,1.63)
THEN
        Damage = Edge     100.0%


RULE_9 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-3.609,-2.263)
THEN
        Damage = Center   32.9%
        Damage = None     67.1%


RULE_10 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-2.263,-1.435)
        Thetax = [-2.506,-1.998)
```

```
THEN
        Damage = Center   50.0%
        Damage = Edge     50.0%


RULE_11 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-2.263,-1.435)
        Thetax = [-1.998,2.069)
THEN
        Damage = Center   98.1%
        Damage = Edge      1.9%


RULE_12 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-2.263,-1.435)
        Thetax = [2.069,2.4]
THEN
        Damage = Center   50.0%
        Damage = Edge     50.0%


RULE_13 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-1.435,-0.785)
        Thetax = [-2.506,-0.981)
THEN
        Damage = Edge    100.0%


RULE_14 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-1.435,-0.785)
        Thetax = [-0.473,0.035)
THEN
        Damage = Center  100.0%


RULE_15 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-1.435,-0.785)
        Thetax = [0.544,2.4]
THEN
        Damage = Edge    100.0%


RULE_16 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-0.785,-0.276)
THEN
        Damage = Edge    100.0%


RULE_17 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [-0.276,0.086)
THEN
        Damage = Edge    100.0%


RULE_18 IF
        Vel(Re) = [-1.761,-1.345)
        Vel(Im) = [0.086,0.487)
THEN
        Damage = Edge    100.0%


RULE_19 IF
        Vel(Re) = [-1.345,-0.857)
        Vel(Im) = [-3.609,-2.263)
THEN
        Damage = Center    6.4%
        Damage = Edge      2.1%
        Damage = None     91.5%


RULE_20 IF
        Vel(Re) = [-1.345,-0.857)
        Vel(Im) = [-2.263,-1.435)
THEN
        Damage = Center   46.0%
        Damage = Edge      3.3%
        Damage = None     50.7%


RULE_21 IF
        Vel(Re) = [-1.345,-0.857)
        Vel(Im) = [-1.435,-0.785)
THEN
        Damage = Center   67.6%
        Damage = Edge     32.4%


RULE_22 IF
        Vel(Re) = [-1.345,-0.857)
        Vel(Im) = [-0.785,-0.276)
THEN
        Damage = Edge    100.0%


RULE_23 IF
        Vel(Re) = [-1.345,-0.857)
        Vel(Im) = [-0.276,0.086)
THEN
        Damage = Edge    100.0%


RULE_24 IF
        Vel(Re) = [-0.857,-0.374)
        Vel(Im) = [-2.263,-1.435)
        Thetax = [-2.506,-1.998)
THEN
        Damage = Center    9.1%
        Damage = Edge     63.6%
        Damage = None     27.3%


RULE_25 IF
        Vel(Re) = [-0.857,-0.374)
        Vel(Im) = [-2.263,-1.435)
        Thetax = [-1.998,2.4]
THEN
```

```
        Damage = Center   14.6%
        Damage = Edge      2.4%
        Damage = None     82.9%


RULE_26 IF
        Vel(Re) = [-0.857,-0.374)
        Vel(Im) = [-1.435,-0.785)
        Thetax = [-2.506,2.069)
THEN
        Damage = Center   49.6%
        Damage = Edge      0.8%
        Damage = None     49.6%


RULE_27 IF
        Vel(Re) = [-0.857,-0.374)
        Vel(Im) = [-1.435,-0.785)
        Thetax = [2.069,2.4]
THEN
        Damage = Center   52.9%
        Damage = Edge     17.6%
        Damage = None     29.4%


RULE_28 IF
        Vel(Re) = [-0.857,-0.374)
        Vel(Im) = [-0.785,-0.276)
THEN
        Damage = Center   42.4%
        Damage = Edge     36.4%
        Damage = None     21.2%


RULE_29 IF
        Vel(Re) = [-0.857,-0.374)
        Vel(Im) = [-0.276,0.086)
THEN
        Damage = Edge    100.0%
```

```
RULE_30 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-2.263,-0.785)
        Thetax = [-2.506,-1.998)
THEN
        Damage = Center  100.0%


RULE_31 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-2.263,-0.785)
        Thetax = [0.544,1.561)
THEN
        Damage = None    100.0%


RULE_32 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-2.263,-0.785)
        Thetax = [1.561,2.069)
THEN
        Damage = Center  100.0%


RULE_33 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-2.263,-0.785)
        Thetax = [2.069,2.4]
THEN
        Damage = Center   20.0%
        Damage = Edge     80.0%


RULE_34 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-0.785,-0.276)
        Thetay = [-2.497,-1.965)
THEN
        Damage = Center   20.0%
        Damage = Edge     30.0%
        Damage = None     50.0%
```

```
RULE_35 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-0.785,-0.276)
        Thetay = [-1.965,-1.421)
THEN
        Damage = Center   22.2%
        Damage = None      77.8%


RULE_36 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-0.785,-0.276)
        Thetay = [-1.421,1.804)
THEN
        Damage = Center   62.0%
        Damage = None      38.0%


RULE_37 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-0.785,-0.276)
        Thetay = [1.804,2.707]
THEN
        Damage = Center   26.1%
        Damage = None      73.9%


RULE_38 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [-0.276,0.086)
THEN
        Damage = Center   31.3%
        Damage = Edge      33.6%
        Damage = None      35.1%


RULE_39 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [0.086,0.487)
THEN
```

```
        Damage = Center   35.3%
        Damage = None     64.7%


RULE_40 IF
        Vel(Re) = [-0.374,0.119)
        Vel(Im) = [0.487,0.883)
THEN
        Damage = Edge    100.0%


RULE_41 IF
        Vel(Re) = [0.119,0.677)
        Vel(Im) = [-0.785,0.086)
        Thetay = [-2.497,-1.965)
THEN
        Damage = Center   50.0%
        Damage = Edge     25.0%
        Damage = None     25.0%


RULE_42 IF
        Vel(Re) = [0.119,0.677)
        Vel(Im) = [-0.785,0.086)
        Thetay = [-1.965,2.707]
THEN
        Damage = Edge    100.0%


RULE_43 IF
        Vel(Re) = [0.119,0.677)
        Vel(Im) = [0.086,0.487)
THEN
        Damage = Center   33.0%
        Damage = Edge     28.2%
        Damage = None     38.8%


RULE_44 IF
        Vel(Re) = [0.119,0.677)
        Vel(Im) = [0.487,0.883)
```

```
THEN
        Damage = Center   47.4%
        Damage = Edge      1.3%
        Damage = None     51.3%


RULE_45 IF
        Vel(Re) = [0.119,0.677)
        Vel(Im) = [0.883,2.28)
        Thetax = [-2.506,-1.998)
THEN
        Damage = Center   40.0%
        Damage = Edge     20.0%
        Damage = None     40.0%


RULE_46 IF
        Vel(Re) = [0.119,0.677)
        Vel(Im) = [0.883,2.28)
        Thetax = [-1.998,2.4]
THEN
        Damage = Center    2.0%
        Damage = None     98.0%


RULE_47 IF
        Vel(Re) = [0.677,1.131)
        Vel(Im) = [-0.276,0.086)
THEN
        Damage = Center   50.0%
        Damage = Edge     50.0%


RULE_48 IF
        Vel(Re) = [0.677,1.131)
        Vel(Im) = [0.086,0.487)
        Thetay = [-2.497,-1.965)
THEN
        Damage = Center   40.0%
        Damage = Edge     40.0%
        Damage = None     20.0%
```

```
RULE_49 IF
        Vel(Re) = [0.677,1.131)
        Vel(Im) = [0.086,0.487)
        Thetay = [-1.965,2.707]
THEN
        Damage = Edge     100.0%


RULE_50 IF
        Vel(Re) = [0.677,1.131)
        Vel(Im) = [0.487,0.883)
THEN
        Damage = Center   33.3%
        Damage = Edge     22.2%
        Damage = None     44.4%


RULE_51 IF
        Vel(Re) = [0.677,1.131)
        Vel(Im) = [0.883,1.63)
THEN
        Damage = Center   60.3%
        Damage = Edge      0.9%
        Damage = None     38.8%


RULE_52 IF
        Vel(Re) = [0.677,1.131)
        Vel(Im) = [1.63,2.28)
        Thetay = [-1.965,2.348)
THEN
        Damage = Center    1.1%
        Damage = None     98.9%


RULE_53 IF
        Vel(Re) = [0.677,1.131)
        Vel(Im) = [1.63,2.28)
        Thetay = [2.348,2.707]
THEN
```

```
        Damage = Center   25.0%
        Damage = Edge      8.3%
        Damage = None     66.7%


RULE_54 IF
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [0.086,0.487)
THEN
        Damage = Edge    100.0%


RULE_55 IF
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [0.487,0.883)
THEN
        Damage = Center   11.1%
        Damage = Edge     88.9%


RULE_56 IF
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [0.883,1.63)
THEN
        Damage = Center  100.0%


RULE_57 IF
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [1.63,2.28)
        Thetay = [-2.497,0.729)
THEN
        Damage = Center   56.8%
        Damage = None     43.2%


RULE_58 IF
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [1.63,2.28)
        Thetay = [0.729,2.707]
THEN
```

```
        Damage = Center   88.1%
        Damage = None      11.9%


RULE_59 IF
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [2.28,3.481]
THEN
        Damage = None     100.0%


RULE_60 IF
        Vel(Re) = [1.508,2.15)
        Vel(Im) = [-0.785,0.487)
THEN
        Damage = Edge     100.0%


RULE_61 IF
        Vel(Re) = [1.508,2.15)
        Vel(Im) = [0.487,0.883)
THEN
        Damage = Edge     100.0%


RULE_62 IF
        Vel(Re) = [1.508,2.15)
        Vel(Im) = [0.883,1.63)
THEN
        Damage = Edge     100.0%


RULE_63 IF
        Vel(Re) = [1.508,2.15)
        Vel(Im) = [1.63,2.28)
        Thetax = [-2.506,-1.998)
THEN
        Damage = Center   50.0%
        Damage = Edge     50.0%
```

```
RULE_64 IF
        Vel(Re) = [1.508,2.15)
        Vel(Im) = [1.63,2.28)
        Thetax = [-1.998,2.4]
THEN
        Damage = Center  100.0%


RULE_65 IF
        Vel(Re) = [1.508,2.15)
        Vel(Im) = [2.28,3.481]
        Thetax = [-2.506,2.069)
THEN
        Damage = Center   90.4%
        Damage = None      9.6%


RULE_66 IF
        Vel(Re) = [1.508,2.15)
        Vel(Im) = [2.28,3.481]
        Thetax = [2.069,2.4]
THEN
        Damage = Center   60.0%
        Damage = None     40.0%


RULE_67 IF
        Vel(Re) = [2.15,3.561]
        Vel(Im) = [-0.785,0.086)
THEN
        Damage = Edge    100.0%


RULE_68 IF
        Vel(Re) = [2.15,3.561]
        Vel(Im) = [0.086,0.487)
THEN
        Damage = Edge    100.0%


RULE_69 IF
```

```
        Vel(Re) = [2.15,3.561]
        Vel(Im) = [0.487,0.883)
THEN
        Damage = Edge     100.0%


RULE_70 IF
        Vel(Re) = [2.15,3.561]
        Vel(Im) = [0.883,2.28)
THEN
        Damage = Edge     100.0%


RULE_71 IF
        Vel(Re) = [2.15,3.561]
        Vel(Im) = [2.28,3.481]
THEN
        Damage = Center  100.0%
```

APPENDIX B

```
% This set of generic rules corresponds to the decision tree in Figures
% 14-17.  Although there are 49 rules listed here, only 6 of them are
% considered to be "interesting."  These represent the "sensitive" areas
% on the plates where damage detection is prominent.


RULE_1 IF
        Thetax = [-2.506,-1.998)
        Thetay = [-1.965,-1.421)
        Vel(Re) = [0.677,1.508)
THEN
        Damage = Center   11.1%
        Damage = None     88.9%


RULE_2 IF
        Thetax = [-2.506,-1.998)
        Thetay = [-1.965,-1.421)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center   88.9%
        Damage = None     11.1%


RULE_3 IF
        Thetax = [-2.506,-1.998)
        Thetay = [-1.965,-1.421)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge    100.0%


RULE_4 IF
```

```
        Thetax = [-2.506,-1.998)
        Thetay = [-1.421,-0.89)
        Vel(Im) = [0.883,1.63)
THEN
        Damage = Edge     100.0%


RULE_5 IF
        Thetax = [-2.506,-1.998)
        Thetay = [-1.421,-0.89)
        Vel(Im) = [1.63,3.481]
        Vel(Re) = [1.131,1.508)
THEN
        Damage = None     100.0%


RULE_6 IF
        Thetax = [-2.506,-1.998)
        Thetay = [-1.421,-0.89)
        Vel(Im) = [1.63,3.481]
        Vel(Re) = [1.508,3.561]
THEN
        Damage = Center   81.8%
        Damage = None      18.2%


RULE_7 IF
        Thetax = [-1.998,-1.49)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [0.677,1.508)
THEN
        Damage = None     100.0%


RULE_8 IF
        Thetax = [-1.998,-1.49)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center  100.0%
```

```
RULE_9 IF
        Thetax = [-1.998,-1.49)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Center   10.0%
        Damage = Edge     90.0%


RULE_10 IF
        Thetax = [-1.49,-0.981)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [0.677,1.508)
THEN
        Damage = None    100.0%


RULE_11 IF
        Thetax = [-1.49,-0.981)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center  100.0%


RULE_12 IF
        Thetax = [-1.49,-0.981)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Center   10.0%
        Damage = Edge     90.0%


RULE_13 IF
        Thetax = [-0.981,-0.473)
THEN
        Damage = Center   33.3%
        Damage = Edge     33.3%
        Damage = None     33.3%
```

```
RULE_14 IF
        Thetax = [-0.473,0.035)
        Thetay = [-0.346,0.185)
        Vel(Re) = [0.677,1.508)
THEN
        Damage = Center   10.0%
        Damage = None     90.0%


RULE_15 IF
        Thetax = [-0.473,0.035)
        Thetay = [-0.346,0.185)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center  100.0%


RULE_16 IF
        Thetax = [-0.473,0.035)
        Thetay = [-0.346,0.185)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge    100.0%


RULE_17 IF
        Thetax = [-0.473,0.035)
        Thetay = [0.185,0.729)
        Vel(Re) = [0.119,1.508)
THEN
        Damage = Center   25.0%
        Damage = None     75.0%


RULE_18 IF
        Thetax = [-0.473,0.035)
        Thetay = [0.185,0.729)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center  100.0%
```

```
RULE_19 IF
        Thetax = [-0.473,0.035)
        Thetay = [0.185,0.729)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge     100.0%


RULE_20 IF
        Thetax = [0.035,0.544)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [0.119,0.677)
THEN
        Damage = None     100.0%


RULE_21 IF
        Thetax = [0.035,0.544)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [0.677,2.15)
        Vel(Im) = [0.086,0.883)
THEN
        Damage = Edge     100.0%


RULE_22 IF
        Thetax = [0.035,0.544)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [0.677,2.15)
        Vel(Im) = [0.883,3.481]
THEN
        Damage = Center   64.3%
        Damage = None     35.7%


RULE_23 IF
        Thetax = [0.035,0.544)
        Thetay = [-1.421,-0.89)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge     100.0%
```

```
RULE_24 IF
        Thetax = [0.035,0.544)
        Thetay = [-0.346,0.185)
        Vel(Re) = [0.677,1.508)
THEN
        Damage = None     100.0%


RULE_25 IF
        Thetax = [0.035,0.544)
        Thetay = [-0.346,0.185)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center  100.0%


RULE_26 IF
        Thetax = [0.035,0.544)
        Thetay = [-0.346,0.185)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge     100.0%


RULE_27 IF
        Thetax = [0.035,0.544)
        Thetay = [0.185,0.729)
        Vel(Re) = [0.677,1.508)
THEN
        Damage = Center   10.0%
        Damage = None     90.0%


RULE_28 IF
        Thetax = [0.035,0.544)
        Thetay = [0.185,0.729)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center  100.0%
```

```
RULE_29 IF
        Thetax = [0.035,0.544)
        Thetay = [0.185,0.729)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge    100.0%


RULE_30 IF
        Thetax = [0.544,1.052)
        Thetay = [0.185,0.729)
        Vel(Re) = [0.119,1.131)
THEN
        Damage = None    100.0%


RULE_31 IF
        Thetax = [0.544,1.052)
        Thetay = [0.185,0.729)
        Vel(Re) = [1.131,2.15)
        Vel(Im) = [0.487,0.883)
THEN
        Damage = Edge    100.0%


RULE_32 IF
        Thetax = [0.544,1.052)
        Thetay = [0.185,0.729)
        Vel(Re) = [1.131,2.15)
        Vel(Im) = [1.63,3.481]
THEN
        Damage = Center   81.8%
        Damage = None     18.2%


RULE_33 IF
        Thetax = [0.544,1.052)
        Thetay = [0.185,0.729)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge    100.0%
```

```
RULE_34 IF
        Thetax = [0.544,1.052)
        Thetay = [0.729,1.261)
        Vel(Re) = [0.677,1.131)
THEN
        Damage = None    100.0%


RULE_35 IF
        Thetax = [0.544,1.052)
        Thetay = [0.729,1.261)
        Vel(Re) = [1.131,1.508)
THEN
        Damage = Center   81.8%
        Damage = None     18.2%


RULE_36 IF
        Thetax = [0.544,1.052)
        Thetay = [0.729,1.261)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge    100.0%


RULE_37 IF
        Thetax = [1.052,1.561)
        Thetay = [0.729,1.261)
        Vel(Re) = [0.677,1.131)
THEN
        Damage = None    100.0%


RULE_38 IF
        Thetax = [1.052,1.561)
        Thetay = [0.729,1.261)
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [1.63,2.28)
THEN
        Damage = Center   81.8%
        Damage = None     18.2%


RULE_39 IF
        Thetax = [1.052,1.561)
        Thetay = [0.729,1.261)
        Vel(Re) = [1.131,1.508)
        Vel(Im) = [2.28,3.481]
THEN
        Damage = None    100.0%


RULE_40 IF
        Thetax = [1.052,1.561)
        Thetay = [0.729,1.261)
        Vel(Re) = [1.508,3.561]
THEN
        Damage = Edge    100.0%


RULE_41 IF
        Thetax = [1.561,2.069)
        Thetay = [-0.89,-0.346)
        Vel(Im) = [-3.609,-2.263)
THEN
        Damage = Center   25.0%
        Damage = None     75.0%


RULE_42 IF
        Thetax = [1.561,2.069)
        Thetay = [-0.89,-0.346)
        Vel(Im) = [-2.263,-1.435)
THEN
        Damage = Center  100.0%


RULE_43 IF
        Thetax = [1.561,2.069)
        Thetay = [-0.89,-0.346)
        Vel(Im) = [-0.276,0.487)
```

```
THEN
        Damage = Edge     100.0%


RULE_44 IF
        Thetax = [1.561,2.069)
        Thetay = [1.261,1.804)
        Vel(Re) = [1.131,1.508)
THEN
        Damage = Center   12.5%
        Damage = None     87.5%


RULE_45 IF
        Thetax = [1.561,2.069)
        Thetay = [1.261,1.804)
        Vel(Re) = [1.508,2.15)
THEN
        Damage = Center   80.0%
        Damage = None     20.0%


RULE_46 IF
        Thetax = [1.561,2.069)
        Thetay = [1.261,1.804)
        Vel(Re) = [2.15,3.561]
THEN
        Damage = Edge     100.0%


RULE_47 IF
        Thetax = [2.069,2.4]
        Thetay = [1.261,1.804)
        Vel(Im) = [0.086,1.63)
THEN
        Damage = Edge     100.0%


RULE_48 IF
        Thetax = [2.069,2.4]
        Thetay = [1.261,1.804)
        Vel(Im) = [1.63,2.28)
THEN
        Damage = Center  100.0%


RULE_49 IF
        Thetax = [2.069,2.4]
        Thetay = [1.261,1.804)
        Vel(Im) = [2.28,3.481]
THEN
        Damage = Center   30.8%
        Damage = None     69.2%
```

APPENDIX C

```
% Learn1i.m


% This program will use BOXES to optimize inputs to control vibration of a
% 1 mass, single degree-of-freedom system.  The mass is controlled in this
% simulation in real time.  There are 9 discretizations of disp and U.
% Excitation is an impulse disturbance.
clear;
!del learn1i.met

% Define the 3-mass system and discretize.

A = [0 1; -2 0];
B2 = [0 1]';          % disturbance input
B1 = [0 1]';          % control input
C = [0 0];
D = [0];
T = 1;
[phi,gam1] = c2d(A,B1,T);
[phi,gam2] = c2d(A,B2,T);

% Initialize variables

for f = 1:1000

Xt = 5;   % analogous to initial disp of 0.0
Xt1 = 5;
Xt2 = 5;
XtnU = 41;
XtnU3 = 41;
XtnU2 = 41;
XtnU1 = 41;
Xt1nXt2 = 41;
Xt1nXt21 = 41;
```

```
Xt1nXt22 = 41;
Xt1nXt23 = 41;
k = 0;
D = 50;  %  set impulse disturbance.
u = 5;  %  analogous to U=0 or input of 0.
U = 0;
X(1,:) = [0 0];
R(1,:) = [0 0];
res(1,1)=0;
cont(1,1)=0;
disp = 0;
disp1 = 0;
disp2 = 0;
disp3 = 0;
totx = 0;
%load avex; load n;
%addx = n;
%load opt;
load uiavex; load uin;
load uiaddx;
load uiopt;

% Perform simulation

for k = 1:100

X(2,:) = (phi*X(1,:)' + gam1*U + gam2*D)';
R(k+1,:) = (phi*R(k,:)' + gam2*D)';
D = 0;
X(1,:) = X(2,:);
disp3 = disp2;
disp2 = disp1;
disp1 = disp;
disp = X(2,1);

res(k+1,1) = disp;              % evaluator (this should converge to zero).
cont(k+1,1) = U;

%  Update learning matrices

totx = abs(disp) + abs(disp1) + abs(disp2) + abs(disp3);
```

```
n(XtnU3,Xt1nXt23) = n(XtnU3,Xt1nXt23) + 1;
addx(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23) + totx;
avex(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23)/n(XtnU3,Xt1nXt23);

% Identify box associated with last run

Xt2 = Xt1;
Xt1 = Xt;
if disp < -10.0 Xt = 1;
  elseif disp >= -10.0 & disp < -5.0 Xt = 2;
  elseif disp >= -5.0 & disp < -3.0 Xt = 3;
  elseif disp >=  -3.0 & disp < -1.0 Xt = 4;
  elseif disp >=  -1.0 & disp <  1.0 Xt = 5;
  elseif disp >=   1.0 & disp < 3.0 Xt = 6;
  elseif disp >=  3.0 & disp <  5.0 Xt = 7;
  elseif disp >=  5.0 & disp < 10.0 Xt = 8;
  elseif disp >=  10.0 Xt = 9;
end

% Choose subsequent optimum input from BOXES

XtnU3 = XtnU2;
XtnU2 = XtnU1;
XtnU1 = XtnU;
Xt1nXt23 = Xt1nXt22;
Xt1nXt22 = Xt1nXt21;
Xt1nXt21 = Xt1nXt2;
Xt1nXt2 = (Xt2-1)*9 + Xt1;
XtnUa = (Xt-1)*9 + 1;
XtnUb = XtnUa + 8;
for l = XtnUa:XtnUb
  if avex(l,Xt1nXt2)==min(avex((XtnUa:XtnUb),Xt1nXt2))
     u=l+1-XtnUa;
     U=(u-5);
  end
end
XtnU = (Xt-1)*9 + u;

% Save the most recent optimum input to BOXES matrix

opt(Xt,Xt1nXt2)=U;
```

```
end

save uIn n;
save uIaddx addx;
save uIavex avex;
save uIopt opt;

clg;
!del learn1i.met
t=(0:.01:k/100);
axis([0 1 -4 4])
subplot(224), plot(t,cont)
xlabel('Time, sec.')
ylabel('Force')
title('Controller Input')
axis([0 1 -40 40])
subplot(221), plot(t,R(:,1))
xlabel('Time, sec.')
ylabel('Disp')
title('Uncontrolled Response')
axis([0 1 -40 40])
subplot(222), plot(t,res)
xlabel('Time, sec.')
ylabel('Disp')
title('Controlled Response')
meta learn1i
end


% Learn1CS.m

% This program will use BOXES to optimize inputs to control vibration of a
% 1 mass, single degree-of-freedom system.  The mass is controlled in this
% simulation in real time.  There are 9 discretizations of disp and U.
% Excitation is an constant disturbance.  The vibration is controlled to the
% steady-state value of the system, i.e., 5.
clear;
!del learn1cs.met

% Define the 1-mass system and discretize.
```

```
A = [0 1; -2 0];
B2 = [0 1]';            % disturbance input
B1 = [0 1]';            % control input
C = [0 0];
D = [0];
T = 2;
[phi,gam1] = c2d(A,B1,T);
[phi,gam2] = c2d(A,B2,T);

% Initialize variables

for f = 1:1000

Xt = 3;   % analogous to initial disp of 0.0
Xt1 = 3;
Xt2 = 3;
XtnU = 23;
XtnU3 = 23;
XtnU2 = 23;
XtnU1 = 23;
Xt1nXt2 = 21;
Xt1nXt21 = 21;
Xt1nXt22 = 21;
Xt1nXt23 = 21;
k = 0;
D = 10;  %  set constant disturbance.
u = 5;  %  analogous to U=0 or input of 0.
U = 0;
X(1,:) = [0 0];
R(1,:) = [0 0];
res(1,1)=0;
cont(1,1)=0;
disp = 0;
disp1 = 0;
disp2 = 0;
disp3 = 0;
totx = 0;
%load avex; load n;
%addx = n;
%load opt;
load ucsavex; load ucsn;
load ucsaddx;
load ucsopt;

%if f == 5
%addx = avex; load n;
%end;

% Perform simulation

for k = 1:100

X(2,:) = (phi*X(1,:)' + gam1*U + gam2*D)';
R(k+1,:) = (phi*R(k,:)' + gam2*D)';
D = 10;
X(1,:) = X(2,:);
disp3 = disp2;
disp2 = disp1;
disp1 = disp;
disp = X(2,1);

res(k+1,1) = disp;            % evaluator (this should converge to zero).
cont(k+1,1) = U;

% Update matrices

totx = abs(abs(disp)-5) + abs(abs(disp1)-5) + abs(abs(disp2)-5) + abs(abs(disp3)-5);
n(XtnU3,Xt1nXt23) = n(XtnU3,Xt1nXt23) + 1;
addx(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23) + totx;
avex(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23)/n(XtnU3,Xt1nXt23);

% Identify box associated with last run

Xt2 = Xt1;
Xt1 = Xt;
if disp < -5.0 Xt = 1;
  elseif disp >= -5.0 & disp <  0.0 Xt = 2;
  elseif disp >=  0.0 & disp <  2.0 Xt = 3;
  elseif disp >=  2.0 & disp <  4.0 Xt = 4;
  elseif disp >=  4.0 & disp <  6.0 Xt = 5;
  elseif disp >=  6.0 & disp <  8.0 Xt = 6;
  elseif disp >=  8.0 & disp < 10.0 Xt = 7;
  elseif disp >=  10.0 & disp < 15.0 Xt = 8;
  elseif disp >=  15.0 Xt = 9;
end

% Choose subsequent optimum input from BOXES

XtnU3 = XtnU2;
XtnU2 = XtnU1;
XtnU1 = XtnU;
Xt1nXt23 = Xt1nXt22;
Xt1nXt22 = Xt1nXt21;
Xt1nXt21 = Xt1nXt2;
Xt1nXt2 = (Xt2-1)*9 + Xt1;
XtnUa = (Xt-1)*9 + 1;
XtnUb = XtnUa + 8;
for l = XtnUa:XtnUb
  if avex(l,Xt1nXt2)==min(avex((XtnUa:XtnUb),Xt1nXt2))
     u=l+1-XtnUa;
     U=(u-5);
  end
end
XtnU = (Xt-1)*9 + u;

% Save the most recent optimum input to BOXES matrix

opt(Xt,Xt1nXt2)=U;

end

save ucsn n;
save ucsaddx addx;
save ucsavex avex;
save ucsopt opt;

clg;
!del learn1cs.met
t=(0:.01:k/100);
axis([0 1 -4 4])
subplot(224), plot(t,cont)
xlabel('Time')
ylabel('Force')
```

```
title('Controller Input')
axis([0 1 -5 15])
subplot(221), plot(t,R(:,1))
xlabel('Time')
ylabel('Disp')
title('Uncontrolled Response')
axis([0 1 -5 15])
subplot(222), plot(t,res)
xlabel('Time')
ylabel('Disp')
title('Controlled Response')
meta learn1cs
end



% Learn1IO.m

% This program will use BOXES to optimize inputs to control vibration of a
% 1 mass, single degree-of-freedom system. The mass is controlled in this
% simulation in real time. There are 9 discretizations of disp and U.
% Excitation is an sinusoidal disturbance.
clear;
!del learn1iO.met

% Define the 1-mass system and discretize.

A = [0 1; -1 -1];
B2 = [0 1]';           % disturbance input
B1 = [0 1]';           % control input
C = [0 0];
D = [0];
T = 1;
[phi,gam1] = c2d(A,B1,T);
[phi,gam2] = c2d(A,B2,T);

% Initialize variables

for f = 1:1000

Xt = 5;   % analogous to initial disp of 0.0
```

```
Xt1 = 5;
Xt2 = 5;
XtnU = 41;
XtnU3 = 41;
XtnU2 = 41;
XtnU1 = 41;
Xt1nXt2 = 41;
Xt1nXt21 = 41;
Xt1nXt22 = 41;
Xt1nXt23 = 41;
k = 0;
D = 0;  %  set initial disturbance.
u1 = 5;
u = 5;  %  analogous to U=0 or input of 0.
U = 0;
X(1,:) = [0 0];
R(1,:) = [0 0];
res(1,1)=0;
cont(1,1)=0;
Din(1,1) = 0;
disp = 0;
disp1 = 0; disp2 = 0;
disp3 = 0; disp4 = 0;
disp5 = 0; disp6 = 0;
totx = 0;
%load avex; load n;
%addx = n;
%load opt;
load uiOavex; load uiOn;
load uiOaddx;
load uiOopt;
%addx = avex;
%load n;
%n = n(:,:) + 1;

% Perform simulation

for k = 1:200

D = 15*sin(2*pi/10*k);
X(2,:) = (phi*X(1,:)' + gam1*U + gam2*D)';
```

```
R(k+1,:) = (phi*R(k,:)' + gam2*D)';
X(1,:) = X(2,:);
disp6 = disp5;
disp5 = disp4;
disp4 = disp3;
disp3 = disp2;
disp2 = disp1;
disp1 = disp;
disp = X(2,1);

res(k+1,1) = disp;          % evaluator (this should converge to zero).
cont(k+1,1) = U;
Din(k+1,1) = 10*sin(2*pi/10*(k+1));

% Update learning matrices

totx = abs(disp) + abs(disp1) + abs(disp2) + abs(disp3) + abs(disp4) + abs(disp5) +
abs(disp6);
n(XtnU3,Xt1nXt23) = n(XtnU3,Xt1nXt23) + 1;
addx(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23) + totx;
avex(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23)/n(XtnU3,Xt1nXt23);

% Identify box associated with last run

Xt2 = Xt1;
Xt1 = Xt;
if disp < -10.0 Xt = 1;
  elseif disp >= -10.0 & disp < -5.0 Xt = 2;
  elseif disp >= -5.0 & disp < -3.0 Xt = 3;
  elseif disp >=  -3.0 & disp < -1.0 Xt = 4;
  elseif disp >=  -1.0 & disp <  1.0 Xt = 5;
  elseif disp >=   1.0 & disp < 3.0 Xt = 6;
  elseif disp >=  3.0 & disp <  5.0 Xt = 7;
  elseif disp >=  5.0 & disp < 10.0 Xt = 8;
  elseif disp >=  10.0 Xt = 9;
end

% Choose subsequent optimum input from BOXES

u1 = u;
XtnU3 = XtnU2;
```

```
XtnU2 = XtnU1;
XtnU1 = XtnU;
Xt1nXt23 = Xt1nXt22;
Xt1nXt22 = Xt1nXt21;
Xt1nXt21 = Xt1nXt2;
Xt1nXt2 = (Xt2-1)*9 + Xt1;
XtnUa = (Xt-1)*9 + 1;
XtnUb = XtnUa + 8;
for l = XtnUa:XtnUb
  if avex(l,Xt1nXt2)==min(avex((XtnUa:XtnUb),Xt1nXt2))
     u=l+1-XtnUa;
     U=(u-5)*5;
  end
end
XtnU = (Xt-1)*9 + u;

% A smoothing filter of sorts

%if u ~= (u1 | (u1-1) | (u1+1))
%  addx(XtnU1,Xt1nXt21) = addx(XtnU1,Xt1nXt21) + 100;
%end

% Save the most recent optimum input to BOXES matrix

opt(Xt,Xt1nXt2)=U;



end

save uIOn n;
save uIOaddx addx;
save uIOavex avex;
save uIOopt opt;

clg;
!del learn1iO.met
t=(0:k);
axis([0 200 -12 12])
subplot(223), plot(t,Din(:,1))
xlabel('Time, sec.')
```

```
ylabel('Force')
title('Disturbance')
axis([0 200 -20 20])
subplot(224), plot(t,cont)
xlabel('Time, sec.')
ylabel('Force')
title('Controller Input')
axis([0 200 -20 20])
subplot(221), plot(t,R(:,1))
xlabel('Time, sec.')
ylabel('Disp')
title('Uncontrolled Response')
axis([0 200 -20 20])
subplot(222), plot(t,res)
xlabel('Time, sec.')
ylabel('Disp')
title('Controlled Response')
meta learn1iO
end



% Learn2I.m

% This program will use BOXES to optimize inputs to control vibration of a
% 2 mass, two degree-of-freedom system.  The mass is controlled in this
% simulation in real time.  There are 9 discretizations of disp and U.
% Excitation is an impulse disturbance.
clear;
!del learn2i.met

% Define the 2-mass system and discretize.

A = [0 0 1 0; 0 0 0 1; -4 2 0 0; 1 -1 0 0];
B2 = [0 0 0 1]';          % disturbance input
B1 = [0 0 0 1]';          % control input
C = [0 0 0 0];
D = [0];
T = 1;
[phi,gam1] = c2d(A,B1,T);
[phi,gam2] = c2d(A,B2,T);
```

```
% Initialize variables

for f = 1:1000

Xt = 5;   % analogous to initial disp of 0.0
Xt1 = 5;
Xt2 = 5;
XtnU = 41;
XtnU3 = 41;
XtnU2 = 41;
XtnU1 = 41;
Xt1nXt2 = 41;
Xt1nXt21 = 41;
Xt1nXt22 = 41;
Xt1nXt23 = 41;
k = 0;
D = 50;  %  set impulse disturbance.
u = 5;  %  analogous to U=0 or input of 0.
U = 0;
X(1,:) = [0 0 0 0];
R(1,:) = [0 0 0 0];
res(1,1)=0;
cont(1,1)=0;
disp = 0;
disp1 = 0;
disp2 = 0;
disp3 = 0;
totx = 0;
%load avex; load n;
%addx = n;
%load opt;
load uiavex; load uin;
load uiaddx;
load uiopt;

if f == 5
addx = avex; load n;
end

% Perform simulation
```

```
for k = 1:100

X(2,:) = (phi*X(1,:)' + gam1*U + gam2*D)';
R(k+1,:) = (phi*R(k,:)' + gam2*D)';
D = 0;
X(1,:) = X(2,:);
disp3 = disp2;
disp2 = disp1;
disp1 = disp;
disp = X(2,2);

res(k+1,1) = disp;          % evaluator (this should converge to zero).
cont(k+1,1) = U;

% Update matrices

totx = abs(disp) + abs(disp1) + abs(disp2) + abs(disp3);
n(XtnU3,Xt1nXt23) = n(XtnU3,Xt1nXt23) + 1;
addx(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23) + totx;
avex(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23)/n(XtnU3,Xt1nXt23);

% Identify box associated with last run

Xt2 = Xt1;
Xt1 = Xt;
if disp < -10.0 Xt = 1;
  elseif disp >= -10.0 & disp < -5.0 Xt = 2;
  elseif disp >= -5.0 & disp < -3.0 Xt = 3;
  elseif disp >=  -3.0 & disp < -1.0 Xt = 4;
  elseif disp >=  -1.0 & disp <  1.0 Xt = 5;
  elseif disp >=   1.0 & disp < 3.0 Xt = 6;
  elseif disp >=  3.0 & disp <  5.0 Xt = 7;
  elseif disp >=  5.0 & disp < 10.0 Xt = 8;
  elseif disp >=  10.0 Xt = 9;
end

% Choose subsequent optimum input from BOXES

XtnU3 = XtnU2;
XtnU2 = XtnU1;
XtnU1 = XtnU;
```

```
Xt1nXt23 = Xt1nXt22;
Xt1nXt22 = Xt1nXt21;
Xt1nXt21 = Xt1nXt2;
Xt1nXt2 = (Xt2-1)*9 + Xt1;
XtnUa = (Xt-1)*9 + 1;
XtnUb = XtnUa + 8;
for l = XtnUa:XtnUb
  if avex(l,Xt1nXt2)==min(avex((XtnUa:XtnUb),Xt1nXt2))
     u=l+1-XtnUa;
     U=(u-5);
  end
end
XtnU = (Xt-1)*9 + u;

% Save the most recent optimum input to BOXES matrix

opt(Xt,Xt1nXt2)=U;

end

save uln n;
save uladdx addx;
save ulavex avex;
save ulopt opt;

clg;
!del learn2i.met
t=(0:.01:k/100);
axis([0 1 -4 4])
subplot(224), plot(t,cont)
xlabel('Time, sec.')
ylabel('Force')
title('Controller Input')
axis([0 1 -50 50])
subplot(221), plot(t,R(:,1))
xlabel('Time, sec.')
ylabel('Disp')
title('Uncontrolled Response')
axis([0 1 -50 50])
subplot(222), plot(t,res)
xlabel('Time, sec.')
```

```
ylabel('Disp')
title('Controlled Response')
meta learn2i
end




% Learn2CS.m

% This program will use BOXES to optimize inputs to control vibration of a
% 2 mass, two degree-of-freedom system.  The mass is controlled in this
% simulation in real time.  There are 9 discretizations of disp and U.
% Excitation is an constant disturbance.  The vibration is controlled to
% the steady state value, i.e., 5.
clear;
!del learn2cs.met

% Define the 2-mass system and discretize.

A = [0 0 1 0; 0 0 0 1; -4 2 0 0; 1 -1 0 0];
B2 = [0 0 1 0]';          % disturbance input
B1 = [0 0 0 1]';          % control input
C = [0 0 0 0];
D = [0];
T = 1;
[phi,gam1] = c2d(A,B1,T);
[phi,gam2] = c2d(A,B2,T);

% Initialize variables

for f = 1:1000

Xt = 2;   % analogous to initial disp of 0.0
Xt1 = 2;
Xt2 = 2;
XtnU = 14;
XtnU3 = 14;
XtnU2 = 14;
XtnU1 = 14;
```

```
Xt1nXt2 = 11;
Xt1nXt21 = 11;
Xt1nXt22 = 11;
Xt1nXt23 = 11;
k = 0;
D = 10;  %  set constant disturbance.
u = 5;  %  analogous to U=0 or input of 0.
U = 0;
X(1,:) = [0 0 0 0];
R(1,:) = [0 0 0 0];
res(1,1)=0;
cont(1,1)=0;
disp = 0;
disp1 = 0;
disp2 = 0;
disp3 = 0;
totx = 0;
%load avex;
%load n;
%addx = n; load opt;
load ucsavex; load ucsopt;
load ucsn;
load ucsaddx;

if f == 1
addx = avex; load n;
end;

% Perform simulation

for k = 1:100

X(2,:) = (phi*X(1,:)' + gam1*U + gam2*D)';
R(k+1,:) = (phi*R(k,:)' + gam2*D)';
D = 10;
X(1,:) = X(2,:);
disp3 = disp2;
disp2 = disp1;
disp1 = disp;
disp = X(2,2);
```

```
res(k+1,1) = disp;            % evaluator (this should converge to zero).
cont(k+1,1) = U;

% Update matrices

totx = abs(abs(disp)-5) + abs(abs(disp1)-5) + abs(abs(disp2)-5) + abs(abs(disp3)-5);
n(XtnU3,Xt1nXt23) = n(XtnU3,Xt1nXt23) + 1;
addx(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23) + totx;
avex(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23)/n(XtnU3,Xt1nXt23);

% Identify box associated with last run

Xt2 = Xt1;
Xt1 = Xt;
if disp < -5.0 Xt = 1;
  elseif disp >=  -5.0 & disp <  0.0 Xt = 2;
  elseif disp >=   0.0 & disp <  2.0 Xt = 3;
  elseif disp >=   2.0 & disp <  4.0 Xt = 4;
  elseif disp >=   4.0 & disp <  6.0 Xt = 5;
  elseif disp >=   6.0 & disp <  8.0 Xt = 6;
  elseif disp >=   8.0 & disp < 10.0 Xt = 7;
  elseif disp >=  10.0 & disp < 15.0 Xt = 8;
  elseif disp >=  15.0 Xt = 9;
end

% Choose subsequent optimum input from BOXES

XtnU3 = XtnU2;
XtnU2 = XtnU1;
XtnU1 = XtnU;
Xt1nXt23 = Xt1nXt22;
Xt1nXt22 = Xt1nXt21;
Xt1nXt21 = Xt1nXt2;
Xt1nXt2 = (Xt2-1)*9 + Xt1;
XtnUa = (Xt-1)*9 + 1;
XtnUb = XtnUa + 8;
for l = XtnUa:XtnUb
  if avex(l,Xt1nXt2)==min(avex((XtnUa:XtnUb),Xt1nXt2))
     u=l+1-XtnUa;
     U=(u-5);
  end
```

```
end
XtnU = (Xt-1)*9 + u;

% Save the most recent optimum input to BOXES matrix

opt(Xt,Xt1nXt2)=U;

end

save ucsn n;
save ucsaddx addx;
save ucsavex avex;
save ucsopt opt;

clg;
!del learn2cs.met
t=(0:.01:k/100);
axis([0 1 -4 4])
subplot(224), plot(t,cont)
xlabel('Time')
ylabel('Force')
title('Controller Input')
axis([0 1 -5 15])
subplot(221), plot(t,R(:,2))
xlabel('Time')
ylabel('Disp')
title('Uncontrolled Response')
axis([0 1 -5 15])
subplot(222), plot(t,res)
xlabel('Time')
ylabel('Disp')
title('Controlled Response')
meta learn2cs
end



% Learn2IO.m


% This program will use BOXES to optimize inputs to control vibration of a
% 2 mass, two degree-of-freedom system.  The mass is controlled in this
% simulation in real time.  There are 9 discretizations of disp and U.
% Excitation is an sinusoidal disturbance.
clear;
!del learn2iO.met

% Define the 2-mass system and discretize.

A = [0 0 1 0; 0 0 0 1; -2 1 -2 1; 1 -1 1 -1];
B2 = [0 0 1 0]';             % disturbance input
B1 = [0 0 0 1]';             % control input
C = [0 0 0 0];
D = [0];
T = 1;
[phi,gam1] = c2d(A,B1,T);
[phi,gam2] = c2d(A,B2,T);

% Initialize variables

for f = 1:1000

Xt = 5;   % analogous to initial disp of 0.0
Xt1 = 5;
Xt2 = 5;
XtnU = 41;
XtnU3 = 41;
XtnU2 = 41;
XtnU1 = 41;
Xt1nXt2 = 41;
Xt1nXt21 = 41;
Xt1nXt22 = 41;
Xt1nXt23 = 41;
k = 0;
D = 0;  %  set initial disturbance.
u1 = 5;
u = 5;  %  analogous to U=0 or input of 0.
U = 0;
X(1,:) = [0 0 0 0];
R(1,:) = [0 0 0 0];
res(1,1)=0;
cont(1,1)=0;
```

```
Din(1,1) = 0;
disp = 0;
disp1 = 0; disp2 = 0;
disp3 = 0; disp4 = 0;
disp5 = 0; disp6 = 0;
totx = 0;
load avex; load n;
addx = n;
load opt;
%load ui0avex; load ui0n;
%load ui0addx;
%load ui0opt;
%addx = avex;
%load n;
%n = n(:,:) + 1;

% Perform simulation

for k = 1:200

D = 15*sin(2*pi/10*k);
X(2,:) = (phi*X(1,:)' + gam1*U + gam2*D)';
R(k+1,:) = (phi*R(k,:)' + gam2*D)';
X(1,:) = X(2,:);
disp6 = disp5;
disp5 = disp4;
disp4 = disp3;
disp3 = disp2;
disp2 = disp1;
disp1 = disp;
disp = X(2,2);

res(k+1,1) = disp;            % evaluator (this should converge to zero).
cont(k+1,1) = U;
Din(k+1,1) = 15*sin(2*pi/10*(k+1));

% Update learning matrices

totx = abs(disp) + abs(disp1) + abs(disp2) + abs(disp3) + abs(disp4) + abs(disp5) +
abs(disp6);
n(XtnU3,Xt1nXt23) = n(XtnU3,Xt1nXt23) + 1;
```

```
addx(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23) + totx;
avex(XtnU3,Xt1nXt23) = addx(XtnU3,Xt1nXt23)/n(XtnU3,Xt1nXt23);

% Identify box associated with last run

Xt2 = Xt1;
Xt1 = Xt;
if disp < -10.0 Xt = 1;
  elseif disp >= -10.0 & disp < -5.0 Xt = 2;
  elseif disp >= -5.0 & disp < -3.0 Xt = 3;
  elseif disp >=  -3.0 & disp < -1.0 Xt = 4;
  elseif disp >=  -1.0 & disp <  1.0 Xt = 5;
  elseif disp >=   1.0 & disp < 3.0 Xt = 6;
  elseif disp >=  3.0 & disp <  5.0 Xt = 7;
  elseif disp >=  5.0 & disp < 10.0 Xt = 8;
  elseif disp >=  10.0 Xt = 9;
end

% Choose subsequent optimum input from BOXES

u1 = u;
XtnU3 = XtnU2;
XtnU2 = XtnU1;
XtnU1 = XtnU;
Xt1nXt23 = Xt1nXt22;
Xt1nXt22 = Xt1nXt21;
Xt1nXt21 = Xt1nXt2;
Xt1nXt2 = (Xt2-1)*9 + Xt1;
XtnUa = (Xt-1)*9 + 1;
XtnUb = XtnUa + 8;
for l = XtnUa:XtnUb
  if avex(l,Xt1nXt2)==min(avex((XtnUa:XtnUb),Xt1nXt2))
     u=l+1-XtnUa;
     U=(u-5)*5;
  end
end
XtnU = (Xt-1)*9 + u;

% A smoothing filter of sorts

%if u ~= (u1 | (u1-1) | (u1+1))
```

```
% addx(XtnU1,Xt1nXt21) = addx(XtnU1,Xt1nXt21) + 100;
%end

% Save the most recent optimum input to BOXES matrix

opt(Xt,Xt1nXt2)=U;

end

save uIOn n;
save uIOaddx addx;
save uIOavex avex;
save uIOopt opt;

clg;
!del learn2iO.met
t=(0:k);
axis([0 200 -20 20])
subplot(223), plot(t,Din(:,1))
xlabel('Time, sec.')
ylabel('Force')
title('Disturbance')
axis([0 200 -20 20])
subplot(224), plot(t,cont)
xlabel('Time, sec.')
ylabel('Force')
title('Controller Input')
%axis([0 200 -20 20])
subplot(221), plot(t,R(:,2))
xlabel('Time, sec.')
ylabel('Disp')
title('Uncontrolled Response')
axis([0 200 -20 20])
subplot(222), plot(t,res)
xlabel('Time, sec.')
ylabel('Disp')
title('Controlled Response')
meta learn2iO
end
```

APPENDIX D

```
/************************************************************************
*   Box_t8.C      "BOXES" controller code for a clamped-clamped
*              beam (Gary's Beam).
*
*              This code is derived from SISO_5M.c.
*              The code writes the following binary files:
*
*                  strain.mat  Measure strain voltages from beam.
*                  U_cont.mat  Output control voltages to beam.
*
*
*            Original Code written
*            3/18/92
*            GKE
*            "BOXES modifications
*            4/15/93
*            DHK
*
*************************************************************************
*/
/*           To Communicate with the T2 data acquisition node, use LINK3
*
*          NOTE:   T8 integers are 32 bits, T2 integers are 16 bits
*/

#include <stdio.h>
#include <math.h>
#include <conc.h>
#include "pzt_cal.h"
#include "savemat.c"

#define   ADCGAIN  3.005f           /* board amplifier gain */
#define   AD_SCALE (4096.0f/(20.0f/ADCGAIN))
#define   CAL      1.0f/AD_SCALE     /* 1.624896006655574e-3 */
```

```
#define   NLOOPS   5000         /* number of control loops to perform */

#define UTOINT16(x)              ((x)*204.7f + 2047)
#define INT16TOVOLT(x, mean)    (((x)-(mean)) * CAL)

int savemat(FILE *fp, int type, char *pname, int mrows, int ncols, int imagf,
             double *preal, double *pimag);
int control(void);

float Yout[1000];
float U_cont[1000];
double dummy[1000];         /* scratch space for MATLAB data output */

void main(void)
{
   int i, r;
   int start_time, stop_time;
   float freq;

   FILE *dataout1;
   FILE *dataout2;

   r = NLOOPS;

   dataout1 = fopen("strain.mat","wb");
   dataout2 = fopen("U_cont.mat","wb");

   start_time = control();
   stop_time = Time();

   printf("Number of Low-Pri ticks = %d\n", stop_time - start_time);
   printf(" k = %d\n",r);
   freq = r/((stop_time - start_time)*64e-6);
   printf("Frequency %4.1f Hz\n", freq);
   printf("Writing Data File....\n");

   for (i=0;i<1000;i++) {
     dummy[i] = (double) Yout[i];
   }
   savemat(dataout1, 0, "y", 1000, 1, 0, (double *) dummy, (double *) 0);
   for (i=0;i<1000;i++) {
     dummy[i] = (double) U_cont[i];
   }
   savemat(dataout2, 0, "u", 1000, 1, 0, (double *) dummy, (double *) 0);

   fcloseall();
   }

/* ............. control loop ................*/

#pragma asm
        .MOD 1
#pragma endasm

int control(void)
{

        int  u16[1];   /* Control voltage in correct DAC format   */
        int y[1];      /* Integer accelerometer data from T2              */
        float y1;      /* Real, calibrated strain voltages          */
        float U;       /* Control output */
        int u;         /* Discretized control values */

        int a, b, c, d, s, p;
        int S, S1, S2, Sa, Sb, Sc, S1a, S1b, S1c, S2a, S2b, S2c;
        float str, str1, str2;
        int nloops, j, k, indx;
        int start_time, pause_time, new_time;
        float totx, MOMMA;

        int N[9][9][9][15];       /* Learning count variable */
        float ADDX[9][9][9][15];  /* Learning sum variable */
        float AVEX[9][9][9][15];  /* Learning average variable */

/* Initialize variables */
  u = 0; s = 0; d = 0; p = 0;
  Yout[0] = 0.0f;
  U_cont[0] = 0.0f;

  S = 5; S1 = 5; S2 = 5; S1a = 5; S1b = 5; S1c = 5;
  S2a = 5; S2b = 5; S2c = 5;
  str = 0.0f; str1 = 0.0f; str2 = 0.0f; str3 = 0.0f;
  totx = 0.0f;

/* Send the number of control loops to the data acquisiton T2 */
  nloops = NLOOPS;
  ChanOut(LINK3OUT, (char *) &nloops, 2);      /* send 2 bytes to T2          */

  printf("Press any key to start\n");
  getch();

/*----------------------------------------------------------------------
                    M A I N                   L O O P
 -----------------------------------------------------------------------
*/
  k = -1;              /* set loop counter */
  start_time = Time();
  while (NLOOPS > k++)
    {
  new_time = Time();
   /* Read in the acceleration values from the T2           */
   /* We are reading this in as two non-zero bytes, and */
   /* then 2 zero bytes to simplify the code on this        */
   /* transputer.  We expect a LSB -> MSB transfer          */

   ChanIn(LINK3IN, (char *) y, sizeof(y));

/* convert the integer value to voltage */
   y1  = INT16TOVOLT(y[0], ADC0_MEAN);

/* Evaluators */
   s = s + 1;
   p = p + 1;
   if (s <= 5000) {
     if (p == 5) {
   p = 0;
   d = d + 1;
   Yout[d] = y1;
   U_cont[d] = U;
     }
   }
```

```
/* Update voltage (strain) ouput values */
    str3 = str2; str2 = str1; str1 = str; str = y1;

/* Update learning arrays */
    totx = fabs(str) + fabs(str1) + fabs(str2);
    N[Sc][S1c][S2c][uc] = N[S][S1][S2][u] + 1;
    ADDX[Sc][S1c][S2c][uc] = ADDX[Sc][S1c][S2c][uc] + totx;
    AVEX[Sc][S1c][S2c][uc] = ADDX[Sc][S1c][S2c][uc]/N[Sc][S1c][S2c][uc];

/* Identify box associated with last loop */
    S2c = S2b; S2b = S2a; S2a = S2;
    S1c = S1b; S1b = S1a; S1a = S1;
    Sc = Sb; Sb = Sa; Sa = S;
    S2 = S1;
    S1 = S;
    if (str < -0.5) S = 1;
      else if (str >= -0.5 & str < -0.3) S = 2;
      else if (str >= -0.3 & str < -0.2) S = 3;
      else if (str >= -0.2 & str < -0.1) S = 4;
      else if (str >= -0.1 & str <  0.1) S = 5;
      else if (str >=  0.1 & str <  0.2) S = 6;
      else if (str >=  0.2 & str <  0.3) S = 7;
      else if (str >=  0.3 & str <  0.5) S = 8;
      else if (str >=  0.5) S = 9;

/* Calculate control associated with optimum input to date */
    MOMMA = AVEX[S][S1][S2][1];
    indx = 1;
    for (j = 2; j <= 15; j++) {
      if (AVEX[S][S1][S2][j] <= MOMMA) {
         MOMMA = AVEX[S][S1][S2][j];
         indx = j;
      }
    }
    u = indx;
    U = (u-8.0)*0.15;

/* send control voltage to T2. Note, upnt points to u16.    */
      u16[0] = UTOINT16(U);
      pause_time = Time();
/*      printf("%d\n",pause_time-new_time); */
```

```
    while (pause_time-new_time < 4) { /* Wait to avoid time variations */
      pause_time = Time();
    }
    ChanOut(LINK3OUT, (char *) &u16[0], 2);          /* send 2 bytes        */

  } /* end control loop */

  return(start_time);
}




/***********************************************************************
*
*       box_t2.c
*
*       Data Acquitsion code for "BOXES" controller
*
*       Programmed By:
*       GKE
*       12/27/90
*       BOXES update By:
*       DHK
*       4/24/93
*
************************************************************************
*/

#include <conc.h>
#include <inline.h>

#define HEAP        1024
#define DELAY

#define ADC         0x6000
#define SH          0x6100
#define MUX                 0x6200
#define DAC         0x6828

#define ADCMASK  0x0fff         /* Since we have a 12-bit converter          */
```

```
                                        /* mask off the high nibble.  */

inp(Process *);
outp(Process *, int);

void main(void)
{
        int nloops;
        Process *ain, *dout;

        /* read in the number of control loops from the root transputer    */
        nloops = ChanInInt(_boot_chan_in);

        /* allocate the processes       */
        ain = ProcAlloc(inp, HEAP, NULL);
        dout = ProcAlloc(outp, 256, 1, nloops+1);

        ProcPriPar(ain, dout);
}

inp(Process *p)
{

        int i;
        int *adc, *sh, *mux;  /* Hardware addresses                        */
        int outdat[2];        /* we need 8 bytes to pad each word with     */
                              /* 2 zero bytes to make it look like         */
                              /* 32-bit ints we are sending back to        */
                              /* control node.                             */

        /* zero out the outdat[2i] */

        for(i=0; i<sizeof(outdat)/sizeof(int); i++) {
                outdat[i] = 0;
        }

        *sh = 0;              /* track mode          */
        adc = (int *) ADC;    /* assign the hardware pointers        */
        sh  = (int *) SH;
        mux = (int *) MUX;
```

```
	while(1) {
		*mux = 0;			/* channel 0 */
		*sh = 1;			/* hold the data */
		ProcWait(1);
		*adc = 0;			/* start the conversion, channel 0 */
		ChanInInt(EVENT);		/* are we done? */
		outdat[0] = *adc & ADCMASK;	/* save the data */

		ChanOut(_boot_chan_out, (char *) outdat, sizeof(outdat));
		*sh = 0;		/* track */
		ProcWait(2);
	}
}


outp(Process *p, int nloops)
{
	int data;
	int *dac;

	dac = (int *) DAC;
	*dac = 2047;		/* zero the output */

	/* control the number of loops in this routine */

	while(nloops--) {
/*	while(1) {  */
	    data = ChanInInt(_boot_chan_in);
	    *dac = data;
	}
	*dac = 2047;
}
```

Vita

David Harlan Kiel was born in Monterey, California on March 12, 1969. He grew up in Annandale, Va. where he graduated from W.T. Woodson High School. With the support of his parents (both morally and financially), he left home in August, 1987 to attend Virginia Tech and major in Mechanical Engineering. He went through the Co-op program by working with Grumman Corporation on the Space Station Freedom Program. In December, 1991 he received his B.S. degree and decided to continue at Virginia Tech for his graduate work. David received his Masters of Science degree in Mechanical Engineering in June, 1993. He has proudly accepted a product development position with Siecor Corporation in Hickory, N.C. and plans to be successful.