Reconciliation of a Rayleigh-Ritz Beam Model with Experimental Data

by

Brian E. Lindholm

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:

Dr. Robert L. West Jr., Chairman

Dr. Larry D. Mitchell

Dr. Alfred L. Wicks

September, 1994
Blacksburg, Virginia

c.2

LD
5655
V855
1994
L5625
c.2

Reconciliation of a Rayleigh-Ritz Beam Model with Experimental Data

by
Brian E. Lindholm

Dr. Robert L. West Jr., Chairman
Mechanical Engineering

(ABSTRACT)

In order to perform structural optimization and/or modification on a structure, an analytical model which sufficiently describes the behavior of the structure must be developed. Analytical models can be generated for almost any structure, but such a model will generally not effectively predict the behavior of the structure unless the model is somehow reconciled with experimental data taken from the structure. Additionally, the model must also be complete, i.e., it must not only model the structure but also model any suspension system used to support the structure. If the suspension is not included in the model, any attempt to reconcile the model with experimental data will result in a incorrect model. Using this incorrect model to perform structural modification cannot be expected to give correct results.

In this thesis, an approach for estimating the effects of a suspension system on the flexural vibration of a structure is developed. These effects are treated mathematically as variations in boundary conditions. Topics discussed include formulation of an analytical model that includes suspension effects, experimental methods for acquiring mode shapes which exhibit these effects, and reconciliation techniques for matching analytical mode shapes to experimental mode shapes to determine the effective boundary conditions.

Acknowledgements

The writing of this master's thesis has been an excessively lengthy affair, but it was nevertheless a tremendous learning experience. The work would have gone much more smoothly had I known everything at the beginning, but sometimes *doing* is the only way to learn everything you need to know.

Many thanks are owed to my advisor, Dr. Robert L. West Jr., for his willingness to meet with me regularly to help me figure out what I was trying to accomplish with my experimental and analytical work. He is owed thanks for his considerable assistance with revising and proofreading this thesis. And finally, his guidance, patience, and friendship are deeply appreciated.

I wish to thank Dr. Larry D. Mitchell and Dr. Alfred L. Wicks for serving on my committee. Their advise on dealing with some of the trickier experimental issues was also appreciated.

And finally, I wish to thank Mr. Jeff Graffeo for his invaluable assistance with acquiring the data needed for this thesis. Taking 200+ laser-Doppler vibrometer scans is a task too large for a single man to accomplish.

This material is based upon work supported under a National Science Foundation Graduate Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

Acknowledgements iv

List of Illustrations vi

List of Tables vii

Nomenclature x

1. Introduction 1

Background and Motivation 1
Statement of Problem 4
Objectives of Research 5
Scope of Research 8
Summary and Outline of Thesis 8

2. Literature Review 11

Overview of Current Approaches 11
Review of Modification Techniques 14
Structural Modification/Optimization 15
Model Verification/Updating 19

3. Development of Analytical Model 22

Model Development Strategy 22
Basic Beam Theory 23
Continuous Eigensystem Solution Techniques 26
Analytical Model Development Summary 35
Implementation Specifics 36
Validation of Rayleigh-Ritz Analytical Model 44
Perturbation Study 59

4. Study of Suspension 67

Transverse Stiffness Effect 67

5. Techniques for Reconciliation 73

Development of Reconciliation Techniques 73
Update Parameter Specification 74
Least-Squares Comparison Methodology 75
Parameter Updates via Optimization 76
Noise Sensitivity Study 80

6. Experimental Techniques for Obtaining Mode Shapes 87

Overview of Mode Shape Process 87
Experimental Setup 88
Natural Frequencies with Accelerometers 90
Operating Shapes via Scanning Laser-Doppler Vibrometer 90
Correction of Systematic Errors 92
Mode Shape Extraction 97

7. Model Reconciliation with Experimental Data 101

Overview of Model Reconciliation 101
Pre-Analysis Calculations 102
Experimental Design 105
Reconciliation of Model with Experimental Data 107
Model Reconciliation Results 108
Overview of Cases 115

8. Conclusions 117

Review of Objectives 117
Thesis Summary 118
Conclusions Regarding Approach 120
Recommendations for Future Work 123
Concluding Remarks 126

References 127

Appendix A: Jacobi Iteration Based Generalized Eigensystem Solver 130
Appendix B: Results of Reconciliation with Individual Mode Shapes 132
Appendix C: MATLAB Source Code 137
Appendix D: BEAM3.EXE Source Code 148

Vita 171

List of Illustrations

Figure

1 Modeling Techniques 12
2 Model Update/Verification Schemes 14
3 Fourier Sine Functions 30
4 Fourier Cosine Functions 31
5 Suggested Quasi-Comparison Functions (from free-free) 32
6 Suggested Quasi-Comparison Functions (from fixed-fixed) 33
7 Alternate Quasi-Comparison Functions (from pinned-pinned) 34
8 Alternate Quasi-Comparison Functions (from guided-guided) 34
9 Sample Mathematica Results 38
10 Rigid-Body Zero-Frequency Convergence 47
11 Flexural Mode Frequency Convergence 47
12 Free-Free Mode Shapes 48
13 Convergence of Fourth Mode Shape 49
14 RMS Error Convergence of Fourth Mode 50
15 Pinned-Pinned Beam 51
16 Pinned-Pinned Mode Shapes 52
17 Fixed-Free (Cantilever) Beam 53
18 Fixed-Free Mode Shapes 54
19 Tensile Load on Beam 56
20 Convergence of Gravitational Tension Effect 58
21 Effects of Translational Stiffness 62
22 Effects of Rotatory Stiffness 63
23 Effects of Translational Mass 64
24 Effects of Rotatory Inertia 65
25 Effects of Gravitational Inertia 66

26 Shock Cord Elongation and Motion 67
27 Shock Cord Test Set-up 69
28 Shock Cord Load/Elongation Fit 70
29 Shock Cord Load/Elongation Residuals 71
30 Shock Cord Stiffness Property 72
31 Medium Noise Example 82
32 Beam Test Set-up (overview) 89
33 Beam Test Set-up (detail) 89
34 X-Values on Beam 94
35 Asymmetric-Trimmed-Mean (prior to sort) 96
36 Asymmetric-Trimmed-Mean (sorted) 97
37 Frequency Error Effects (Experimental) 98
38 Frequency Error Effects (Analytical) 98
39 Imaginary Parts of Operating Shape 100
40 Effective Mass Experiment 103

List of Tables

Table

1 Issues in Structural Modification 2
2 Literature Review Categorization 14
3 Steps for Developing the Analytical Model 22
4 Rayleigh-Ritz Integration Terms 37
5 Ill-Conditioning of Mass Matrix 40
6 Comparison of Eigensystem Solvers 42
7 Complete Steps for Developing the Model 43
8 Analytical Model Beam Parameters 44
9 Analytical Cases Tested 45
10 Free-free Convergence Study 45
11 RMS Error Convergence of Fourth Mode 50
12 Pinned-pinned Convergence Study 52
13 Fixed-free Convergence Study 54
14 BEAM VI Base Stiffness Study 55
15 Gravitational Convergence Study 57
16 Model Verification Tests 59
17 Perturbation Conditions 60
18 Perturbation Study Convergence 61
19 Transverse Stiffness Assumptions 68
20 Steps in Reconciliation Methodology 73
21 Optimization Techniques 78
22 Noise Study Beam Solution 80
23 Noise Levels 81
24 Individual Noise Fits 83
25 Simultaneous Noise Fits 84

26 Individual Noise Fits (Trans Only) 85
27 Simultaneous Noise Fits (Trans Only) 86
28 Effects of Noise 86
29 Mode Shape Acquisition Process 87
30 Complete Steps for Model Reconciliation 101
31 Beam Parameters 102
32 Point Masses at Beam Tip 104
33 Stinger Properties 104
34 Suspension Cases Tested 106
35 Theoretical Transverse Stiffness 107
36 Reconciliation Models 107
37 Simultaneous Fits (individual $S/N > 8.0$) 110
38 Simultaneous Fits (individual $S/N > 16$) 110
39 Simultaneous Fits ($S/N > 8.0$, Trans Only) 111
40 Simultaneous Fits ($S/N > 16$, Trans Only) 112
41 Simultaneous Fits (individual $S/N > 8.0$, no Gravity) 113
42 Simultaneous Fits (individual $S/N > 16$, no Gravity) 113
43 Simultaneous Fits ($S/N > 8.0$, Trans Only, no Gravity) 114
44 Simultaneous Fits ($S/N > 16$, Trans Only, no Gravity) 115
45 Reconciliation Model Summary 116
46 Issues in Structural Optimization 117
47 Objectives of Research 118
48 Recommendations for Future Work on Model 124
49 Recommendations for Future Work on Reconciliation 125
50 Recommendations for Future Work on Data Gathering 125

Nomenclature

δ	variational operator
T	kinetic energy term
V	potential energy term
t	time
θ_0	phase angle of sinusoid
x	location of point on beam (measured from top)
z	normalized location of point on beam
ρ	structure material density
$A(x)$	beam cross-section area
$r'(x)$	correction factor based on beam cross-section
$w(x,t)$	beam displacement
$\varphi(x,t)$	beam shear displacement
$\dot{w}(x,t)$	beam velocity
$y(x)$	mode shape
L	beam length, or alternately, Lagrangian differential operator
E	structure material modulus of rigidity
G	structure material shear modulus
$I(x)$	beam area moment-of-inertia
W	weight of beam
$P(x)$	tensile load on beam
$k(x)$	beam base stiffness
$K_{t,x}$	external tranverse stiffness applied
$K_{r,x}$	external rotatory stiffness applied
m	discrete mass located at $x = 0$
I_m	discrete rotatory inertia located at $x = 0$
B	boundary condition operator
λ	eigenvalue
n	number of Ritz terms included in Ritz model, or alternately, number of rows of data acquired experimentally from beam

R	Rayleigh's quotient
ϕ_i	Ritz trial functions
$w_n(x)$	beam displacement approximation obtained using n Ritz terms
$y_n(x)$	mode shape approximation obtained using n Ritz terms
a_i	summation coefficients of Ritz terms
$\boldsymbol{K}$	stiffness matrix
$\boldsymbol{M}$	mass matrix
Φ	eigenvector matrix
l_0	initial length (unloaded) of shock cord
l_f	final length (loaded) of shock cord
$f_i(x)$	multiple linear regression regressor functions
β_i	multiple linear regression coefficients
θ	laser deflection angle
d	distance from laser to beam

Chapter 1: Introduction

This thesis addresses the issues involved in building a design model that can be related to a structure under test for the purpose of dynamic modification. In order to perform structural optimization, an analytical model which sufficiently describes the behavior of the structure must be developed. Specifically, this model should not only model the structure but also model any suspension system used to support the structure. If the suspension system is not included in the model, any attempt to reconcile the model with experimental data will result in a model that does not correctly describe the behavior of the system. Using this incorrect model to perform structural modification cannot be expected to give correct results.

In this thesis, an approach for estimating the effects of a suspension system on the flexural vibration of a structure is developed. These effects are treated mathematically as variations in boundary conditions. Topics discussed include formulation of an analytical model that includes suspension effects, experimental methods for acquiring mode shapes which exhibit these effects, and reconciliation techniques for matching analytical mode shapes to experimental mode shapes to determine the effective boundary conditions.

Background and Motivation

The work covered in this thesis develops the underlying modeling scheme needed to support the general problem of structural modification and optimization.

There are four conceptual issues that must be satisfied when performing structural modification, as given in Table 1.

Table 1: Issues in Structural Modification

1. It is necessary to have an analytical model that sufficiently characterizes the behavior of the test structure, including the effects of the suspension. If this is not true, modifications are in effect being performed on the wrong structure.

2. In order to develop a correct model of a structure, there must exist justifiable methods for reconciling the model with experimental data. This reconciliation process should utilize as much prior knowledge about the structure as possible and take into account uncertainty present in the model.

3. In order to correctly update the model, it is necessary to have accurate experimental data that effectively describes the dynamic behavior of the structure.

4. Once the analytical model has been reconciled with the experimental data, it can be used for performing structural modification. There must exist justifiable methods for selecting changes if the structure is to be modified to match certain design criteria.

The most popular analytical tool for structural modification is the finite element method, primarily because of its great flexibility in dealing with structures of arbitrary shape, loading, and boundary conditions. Alternatively, direct analytical solutions can be used with very simple structures. Approximate analytical methods such as the Rayleigh-Ritz method can be used to deal with variations of these basic structures.

Experimental structural dynamic data often comes in the form of results from a modal analysis test. This is commonly accomplished by measuring the response of the structure at many frequencies at a small number of points (spatial degrees-of-freedom), as is done with accelerometers and random excitation. An alternate

approach is to take data with more spatial degrees-of-freedom (DOF) at fewer frequencies, as is done with a scanning laser Doppler vibrometer.

Before the model can be used for structural modification or optimization, it must be reconciled with the experimental data. This reconciliation process consists of comparing the model to the data and updating the model such that it more closely matches the data. Ideally, these updates will be robust, matching the behavior of the structure even at frequencies and spatial points for which experimental data is unavailable.

Comparing and updating analytical models can be done with a wide variety of techniques. Most methods of comparison utilize either the orthogonality property of normal modes or direct mode shape correlations, while others compare dynamic response (operating shapes). Some update methods work with the modal parameters from the test, while others focus more on the mass and stiffness matrices of the finite element model, and still others use a parameter-based technique in which design parameters are varied to update the structure.

The parameter-based optimization approach is most effective when the analyst, prior to the analysis, carefully selects the parameters that are to be updated. Ideally, a small number of parameters will be used, reflecting those parts of the system for which the true values are not known. If an excessive number of parameters is used, the optimization process will generate a model that fits the experimental data closely; however, this model will not be robust and cannot be expected to match data taken at other spatial or frequency degrees-of-freedom. If too fews parameters are chosen, or the wrong ones are selected, the model will not be updated as much as it should be.

However, the updated model will still be more accurate than before. This increase in accuracy will be robust as well if no extraneous parameters are selected.

Statement of Problem

The purpose of this thesis is to investigate and implement a parameter-based approach to updating an analytical model of a structure under test to support the higher function of dynamic modification. To develop the approach, a simple uniform beam is used for the experiments. The beam is an easily manufactured structure for which a number of analytical models are available, including the classic Euler-Bernoulli model and the more complete Timoshenko model.

While both mathematical models are believed to be good approximations to the actual behavior of the beam when the boundary conditions are known, the boundary conditions that actually occur in the laboratory are not known with certainty. Ideally, the beam will have perfectly rigid or infinitely flexible supports, matching the mathematically simple cases of fixed-fixed or free-free boundary conditions. Completely fixing the end of a beam is impossible, although it can be well approximated if the support is made of material much more rigid than the structure material. With steel beams, however, this is not practical. Free-free boundary conditions can be better approximated if the beam is supported in space using highly flexible elastic cords. Even in this case, however, the approximation is not perfect.

Since the interaction of the suspension with the beam is unknown, the estimation of boundary conditions, i.e., spring constants, will be the focus of this study. Specifically, the accuracy with which the elastic support cords approximate free-free

boundary conditions on the beam will be determined. Also, methods for improving the analytical model to account for effects caused by the suspension will be investigated. Since the number of parameters involved in specifying the suspension is small, the scale of the problem is manageable. In the study, an analytical model will be used to back out estimates of the suspension stiffness from the effects the suspension has on experimental mode shapes. The estimated stiffnesses will be compared to independent estimates based on properties of the suspension obtained from an alternate experiment. It is also possible to back out damping coefficients, but there is no way to obtain an independent estimates of what the damping coefficient values might be.

Objectives of Research

Develop Analytical Model

The first task required in this research was to develop an analytical model that included effects caused by the suspension. This model had to accurately model the beam along with any stiffness, mass, and tension effects caused by the suspension and measurement transducers. The Euler-Bernoulli beam theory was used instead of the Timoshenko model since the beam used in the experiment had a very high slenderness ratio. The Rayleigh-Ritz method was used to obtain natural frequencies and mode shapes. When implemented properly, the Rayleigh-Ritz method can be used to obtain solutions quickly and accurately for a variety of boundary conditions. The concept of quasi-comparison functions, developed by Meirovitch [1], was used in this context to accurately solve the Euler-Bernoulli beam equation. The selection of quasi-comparison functions also minimizes the computational time and effort required.

Determine Suspension/Boundary Condition Relationship

The second task was to determine how the effects of the suspension system would be manifested in the boundary conditions of the analytical model. This effect is difficult to predict because the shock cords are stretched in a direction parallel to the length of the beam, while the motion at the end of the beam occurs in a perpendicular, or transverse, direction. The simple axial stiffness of the shock cord is not directly related to the equivalent transverse stiffness caused by the cord. Also, it was necessary to determine how to include the effects of the measurement transducers in the analytical model.

Perform Linearity Check on Suspension

In determining the effects of the suspension system on the structure, it is also necessary to determine whether or not the suspension system is behaving in a linear fashion. An experiment was performed in which the shock cord used in the structure suspension system was subjected to an increasing tensile load. The elongation of the cord was measured at a number of tensile loads. Multiple linear regression was used to fit a load/deflection curve to the data. The derivative of this curve provides a normalized axial stiffness value.

Obtain Mode Shapes

To determine how well the analytical model matched the actual behavior of the beam, analytical and experimental mode shapes were compared. Therefore, experimental mode shapes in which the suspension effects were taking effect had to be obtained. Three different thicknesses of shock cord were used at three different

lengths to suspend the beam. The natural frequencies of the first five flexural modes were determined using sine-dwell excitation and driving point accelerometer measurements. Once these natural frequency estimates were obtained, operating shapes were taken using the laser-Doppler vibrometer at -2%, -1%, 0, 1%, and 2% variations from each natural frequency. The scans were performed using LabVIEW 2 on a Macintosh equipped with several National Instruments data acquisition boards.

After the scans were taken, the data was processed to reduce the effects of drop-outs. Drop-outs are a phenomenon unique to the laser system in which the velocity signal intermittently drops toward or even beyond zero. After drop-out effects were alleviated using an asymmetric-trimmed-mean statistic, mode shape information was obtained from the data. This was done by utilizing the knowledge that the modes of the beam under study are well separated. Phase angle information was used to extract the real components of the operating shapes, which are excellent estimates of the mode shapes. This procedure worked well since the non-resonance modes were excited at phase angles of $\pm 90°$, which meant that they contributed only the imaginary components of the operating shapes. The success of this procecure depended on the fact that the beam was very lightly damped with well separated modes.

Reconcile Model Using Optimization

Once the experimental mode shapes were determined, a parameter-based optimization technique was used to reconcile the model with the data. Stiffnesses in the analytical model were adjusted such that the analytical mode shapes matched those from the experiment as closely as possible. A variety of optimization techniques were used to determine translational and rotational stiffnesses for this purpose. Three

different algorithms for unconstrained optimization were used: the Levenberg-Marquardt variant of Newton's method [2], the Broyden-Fletcher-Goldfarb-Shanno quasi-Newton method [3], and the Nelder-Mead simplex method [4]. Sequential quadratic programming [5] was used to solve the problem when stiffness constraints were required.

Scope of Research

The objective of this research was to examine the effects of a suspension system by using a parameter-based approach to reconcile an analytical model with experimental dynamic data. Ideally, this work could be done quickly using the finite element method with a complex, real-world structure. However, to develop the technique with greater confidence in the results, a simple beam was used as the structure and elastic shock cords were used as the suspension. The mechanics of the beam are well known, and working with the suspension provided a problem of limited and workable scope.

Summary and Outline of Thesis

Chapter 2 contains a review of works in the literature that concern structural modification and optimization, estimation of suspension effects and boundary conditions, analytical and computational models, and analytical and experimental correlation. Some of the papers are included to provide context for the research, while others provide a basis for some of the techniques used in solving the problem.

Chapter 3 develops the analytical model used in the work. The Euler-Bernoulli beam model is used with the Rayleigh-Ritz technique to obtain modal parameters for

the beam. This solution technique is combined with the quasi-comparison function technique to quickly and accurately solve the eigensystem problem for a variety of boundary conditions. Difficulties in the computer-based implementation of this technique caused by ill-conditioning are discussed here. The model was verified via comparisons with known analytical results and results from a second computer-based package.

Chapter 4 develops a simple model of the elastic cord suspension system typically used to approximate free boundary conditions on a test structure. An experiment is also performed in which the linearity assumption made on the cord is tested. The results of the analysis presented here comprised an analytical target value for suspension values to which values extracted from experimental results could be compared.

Chapter 5 discusses the various optimization techniques used in reconciling the analytical model to results obtained from experiments. First covered is the least-squares based criterion by which mode shapes are compared. Next, the problem is formulated in terms of meaningful stiffness parameters which an optimization procedure manipulates to obtain results. This chapter also discusses several techniques for constrained and unconstrained optimization and how these techniques were used in the course of solving this problem. Finally, an analytical study was performed in which varying amounts of noise were added to an analytical solution to determine the noise sensitivity of the results obtained from stiffness optimization.

Chapter 6 outlines the experimental method used to obtain mode shapes under the effects of the suspension. Operating shapes are taken at or near the natural

frequencies of the target modes. The data is then processed to reduce the effects of laser drop-outs and to obtain a single displacement vector. Phase information derived from the force signal is used to extract mode shapes from the operating shapes.

Chapter 7 provides the factorial experiment used to study the effects of the shock-cord suspension on the beam under test. Also discussed in the selection of design parameters used in the model update. Data was acquired for the factorial experiment as described in Chapter 6. This data was used in the model reconciliation process of Chapter 5, the results of which were compared to the analytical suspension study of Chapter 4.

Chapter 8 concludes with a brief summary of the thesis and several conclusions about the approach used here. Several recommendations for improving and continuing this work are discussed as well.

Chapter 2: Literature Review

Overview of Current Approaches

Current Modification Approaches

There are several approaches covered in the literature for modifying and optimizing structures for dynamic behavior. These techniques are often closely related to the technique used to model the structure.

One class of modification techniques is based on modeling the structure as a summation of experimentally obtained normal modes of vibration. This approach can be very powerful while requiring minimal computational resources, but modal truncation and improper degree-of-freedom selection can lead to considerable inaccuracy in modification results [6]. Relating modal modifications to possible changes in the actual physical structure can also be difficult [7].

A second class of modification/optimization techniques is based on analytically derived models which are based on mathematical principles describing the behavior of the structure. This philosophy of modeling provides for the analysis of a structure that may not yet physically exist. For structures of very simple geometry, analytical solutions of the governing differential equations can be obtained with little computational effort. However, for structures of arbitrary shape, a more flexible technique such as the finite element method is required. Modifications made to a structure here are usually quite intuitive, although considerable computational resources may be required.

Model Updating/Verification

Positioned between the purely experimental structural model and the purely analytical model is the task of model updating/verification. In this area, data taken from experiments is combined with analytical modeling techniques to form better models. Figure 1 shows how these philosophies of modeling relate to each other.

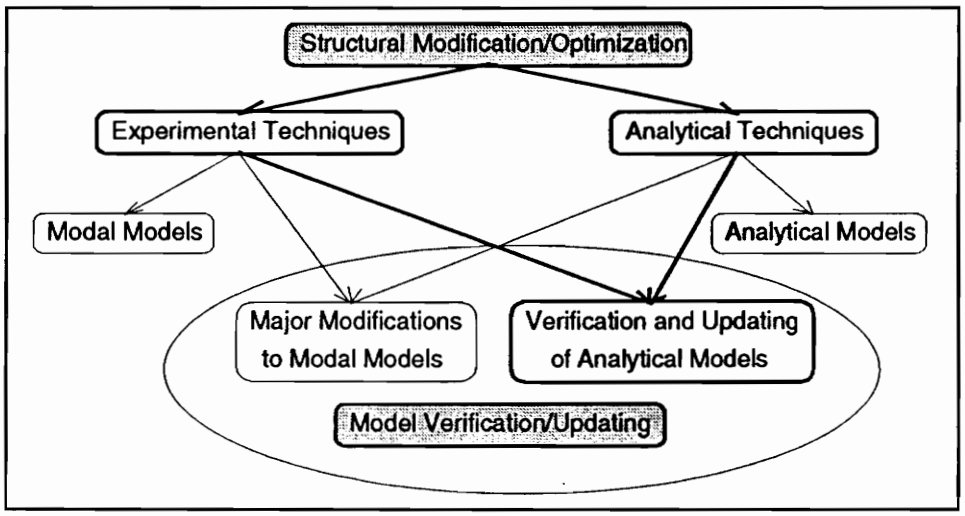

Figure 1: Modeling Techniques

One possible way in which better models can be formed using a combination of experimental and analytical techniques is to combine analytical structural elements with experimentally obtained mode shapes. This allows greater modifications to the modal model without excessive loss of accuracy.

A second way in which better models can be formed using a combination of experimental and analytical techniques is to compare the predictions of the analytical model to results obtained from experiment. The analytical model can then be updated to more accurately match the experimental results. This thesis focuses on this second approach to building better models of a structure. Figure 1 shows where the work

developed here fits into the overall structural modification framework by using bold arrows to show the path researched.

Analytical Model Updating/Verification

There are several tasks that must be accomplished in the process of performing model verification and updating on an analytical. For each of these tasks there are several techniques available. Determining and performing these tasks comprised a major portion of the work presented in this thesis.

The first task that must be done is the acquisition of experimental data. This data can take the form of experimentally derived mode shapes or of experimentally based dynamic response data (operating shapes). These quantities must then be derived from the analytical model as well.

If the mode shapes comprise the choice of data for comparing the analytical model to experimental data, then the comparisons can be made in two ways. One is to utilize the orthogonality property of normal modes, as is done with cross-orthogonality matrices and similar techniques. The other approach is to directly compare the experimental mode shapes with those obtained from the model, usually in a correlation or least-squares sense. If dynamic response data is acquired experimentally as a basis for comparison, then only direct comparisons are possible.

Finally, the model must be updated. Several techniques directly modify the mass, stiffness, and damping matrices of the model. These techniques typically require less computational effort, but the changes made to the model are not always easily related to physical changes in the structure. Other techniques modify design parameters. This approach is more intuitive, but requires more computational effort.

Figure 2 provides a breakdown of the different schemes used to verify and/or update analytical models. The thick black boxes highlight where the work presented in this thesis fits into the overall picture.

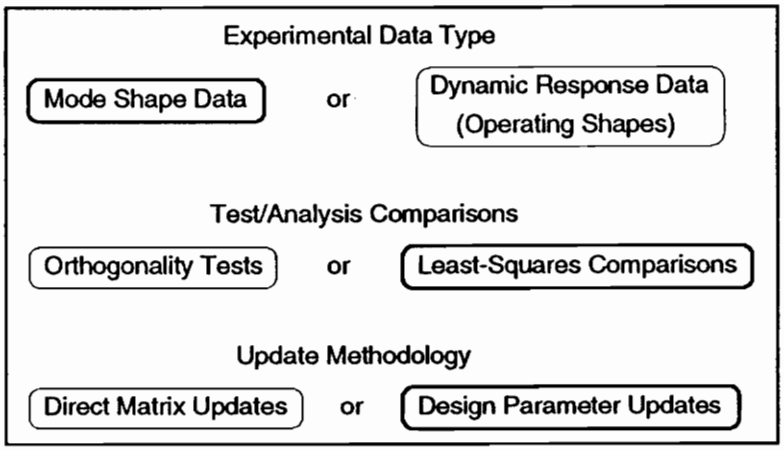

Figure 2: Model Update/Verification Schemes

Review of Modification Techniques

Several papers relating to the previous discussion on modification techniques are presented here. These papers are varied in their philosophies for performing structural modification. The papers are categorized according to Figure 1. Further description of these categories is given in Table 2.

Table 2: Literature Review Categorization

1. Experimental Techniques -- Modal Models: The papers in this category relate to the development of modal models, in which experimentally obtained mode shapes are used as the basis of the model. This type of work does <u>not</u> directly relate to the work of this thesis.

2. Analytical Techniques -- Analytical Models: The papers in this category relate to the development of analytical models, in which models are derived from mathematics describing the governing physics in the structure. This type of work <u>does</u>

relate to the work of this thesis. In particular, Meirovitch's paper [8] describing quasi-comparison functions served as a basis for the analytical model used in this work.

3. Analytical Techniques -- Optimization: The papers in this category focus on a particular application of analytical models, which is structural optimization. This type of work does not directly relate to the work of this thesis, but many of the optimization techniques used here are similar in strategy to the model update process developed in this thesis.

4. Combination Techniques -- Major Modifications to Modal Models: The papers in this category focus on using analytical tools to improve or update experimental modal models. This type of work does not directly relate to the work of this thesis.

5. Combination Techniques -- Verifying and Updating Analytical Models: The papers in this category focus on using experimental data to verify or update an analytical model of a structure. This thesis fits into this category, as the work done here is dedicated to developing a sound methodology for updating analytical models to match experimental dynamic test data.

Structural Modification/Optimization

Experimental Techniques -- Modal Models

In this section, several papers concerning modification techniques are discussed. These papers are concerned with the modal approach to modification, in which the behavior of the structure is modeled using an experimental modal database that describes the dynamics of the structure under test. This original modal information is used in eigenvalue/eigenvector calculations to determine the effects of structural modifications. This approach is fundamentally different from analytical techniques such as the finite element method. Instead of building an analytical model from

scratch, the analyst obtains estimates of the effects of structural modifications in the form of transformations of existing modal data.

Luk and Mitchell [9] developed an extension to the dual-space structural modification by introducing substructuring. With the technique, experimental modal data from two independent substructures could be used to determine the likely modal properties of the structure created when the two substructures are joined together. A large amount of modal truncation could considerably affect the results.

Simultaneously, Elliot and Mitchell [6] more closely examined the effects of modal truncation on the dual-space structural modification technique. They determined that the effects of modal truncation are not simply functions of the number of modes contained in the database, but more functions of which modes were contained in the database. Unfortunately, the means by which modes should be selected was not immediately apparent.

Analytical Techniques -- Analytical Models

Meirovitch and Kwak [8] analyzed the differences in speed of convergence between the classical Rayleigh-Ritz method and the finite element method. The convergence rate of the Rayleigh-Ritz method turned out to be highly dependent on the choice of trial functions used in the series summation. In particular, trial functions can consist of admissible functions, which match only essential boundary conditions and are easy to find. Comparison functions, which match both essential and natural boundary conditions, are often difficult to determine. Certain sets of admissible functions converge very slowly, while comparison functions converge very quickly. A new class of admissible functions known as quasi-comparison functions

was proposed, in which the set of admissible functions contains functions which can be taken in linear combinations to form comparison functions. When the admissible functions used in the series summation are quasi-comparison functions, the rate of convergence is slightly faster than that of true comparison functions. Both sets of functions converge significantly faster than the finite element method. This paper served as a basis for developing the analytical model used in this thesis.

In the area of finite element models, Richard [10] studied a number of mass formulations for beam mass elements for use with dynamic finite element models. Specifically, a generalized mass matrix was used to form four different lumped-mass matrices, a block diagonal mass matrix, a sixth-order mass matrix, and a consistent mass matrix. The consistent mass matrix most accurately matched analytical results obtained from the Euler-Bernoulli beam theory. None of the mass matrices provided results that matched natural frequencies obtained in a small experiment. The finite element method was not used in this thesis.

Singh and Abdelnaser [11] developed a state-space formulation of the Timoshenko beam model for purposes of solving dynamic problems with generalized boundary conditions. Specifically, the state-space approach enables the analyst to avoid explicitly determining orthogonality conditions for the two coupled partial differential equations resulting from Timoshenko's beam theory. Results obtained with Timoshenko's theory were compared with those obtained using the Euler-Bernoulli beam theory. The two theories differ most when predicting natural frequencies and mode shapes for thick beams and for higher mode numbers. The work of this paper related quite closely to the work of the thesis, as it focused on

working with generalized boundary conditions. However, no work was done with experimental data. Additionally, this thesis used the Euler-Bernoulli beam model instead of the Timoshenko beam model because of the high slenderness ratio of the test beam and the fact that only lower modes were studied.

Analytical Models -- Optimization

Another area of interest covered in the literature is optimization of analytical models of structures with respect to vibration. The majority of this work is focused on maximizing natural frequencies or minimizing structural weight with frequency constraints. However, none of the work described here attempts to incorporate experimentally obtained results into the process.

Grandhi and Venkayya [12] developed a design optimization algorithm for structural weight minimization with multiple frequency constraints. The frequency constraints took the form of both equality and inequality constraints, where certain natural frequencies were either fixed at specified values or simply kept above a specified minimum. A Lagrangian-based rescaling algorithm (similar to sequential-quadratic programming) was used to obtain stable solutions to the optimization equations. A number of different finite element truss structure models were used as test structures. The optimization scheme used by Grandhi and Venkayya was similar to that used in this thesis, except that structure mass was minimized instead of sum-of-squares differences in mode shapes.

Olhoff [13] developed an analytical solution for the frequency maximization of a rectangular plate with a fixed mass. A variational formulation of the problem was used to develop a non-linear, fourth-order partial differential eigenvalue problem

describing the solution. The solution to this eigenvalue problem provided a displacement function which specified the plate thickness. The finite difference method was used to solve the partial differential eigenvalue problem. In the analysis, the plate was assumed to be thin. This analysis was completely analytical, in contrast to the numerical optimization performed in the thesis.

Model Verification/Updating

Combination Methods -- Major Modifications to Modal Models

This section discusses a paper in which analytical techniques are combined with experimental techniques to improve the model. Specifically, elements from analytical techniques are used to implement large changes in a modal model of a structure without excessive loss of accuracy.

Elliot and Mitchell [14] discussed an extension of the dual-space structural modification method in which the elements from the analytical transfer-matrix method were used to determine the effects of a beam-type modification to a structure. This was done to better estimate the effects of structural modifications typically made by an engineer. While modifications such as point masses and stiffnesses are easily modeled, beams and braces are a more accurate representation of actual structural modification done in practice. Modal truncation in the modal database was determined to be potentially important, as was the choice of modes put in the database.

Combination Methods: Verifying and Updating Analytical Models

One very important step that must be taken before an analytical model can properly be used for dynamic structural modification is to update the model to match

experimental data. If this is not done, the model will not be reliable for any predictions of a non-qualitative nature. One issue that must be addressed for many of the update techniques in the literature is matching experimental test degrees of freedom with model degrees of freedom.

Baker [7] reviewed methods of comparing finite element models to the results of modal tests. One method of test/analysis is the cross-orthogonality matrix, in which $X = \Phi_{Test}^{T} M \Phi_{FEM}$ ideally yields an identity matrix. Unfortunately, the mode shape vectors need to be of the same length, and the number of data points available from a finite element model usually exceeds the number of points available from an experiment. Additionally, experimental rotational data is normally unavailable. To compensate for this, test-analysis models are used to condense degrees-of-freedom from the analytical model such that they match the degrees-of-freedom from the test. This approach is different than the least-squares methodology used in the thesis.

Another technique for developing analytical models that are representative of experimental data is to build the system mass and stiffness matrices directly from modal test results [7]. Unfortunately, this process is extremely sensitive to noise, which necessitates the use of constrained optimization techniques or penalty weighting schemes to minimize the changes from some predefined model. This technique was not related to any of the work performed in the thesis.

Another topic covered by Baker [7] was the advantages of updating design parameters rather than directly updating the mass and stiffness matrices. Using design parameters maintains direct involvement of engineering judgment, but makes sensitivity calculations more difficult. The approach used in the thesis is an extension

of this methodology, combining the concepts of structural optimization with dynamic structural modification.

Gordis [15] analyzed several of the test-analysis models, placing particular emphasis on the improved-reduced system (IRS) [16] model-reduction procedure. These test-analysis models are used to condense analytical degrees-of-freedom to match those obtained from an experiment. The IRS method is an enhancement of the normal static model-based test analysis model that includes inertia effects. However, the IRS method was found to be very sensitive to the presence of experimental noise in mode shapes, while the static model was much more robust.

Baruh and Boka [17] presented an approach to model and identify boundary conditions in structures, which is equivalent to determining suspension effects. In their approach, orthogonality conditions were used in a system-identification problem to extract estimates of stiffnesses for modeling the suspension. The objective of work done by Baruh and Boka was the similar to that of this thesis.

Conti and Bretl [18] developed an approach for using experimental modal data to determine the rigid-body mass properties in addition to the stiffnesses of the mounts. With this technique, modal data is taken at very low frequencies and is transformed to develop an overdetermined system of equations describing the mount stiffnesses and rigid-body masses. A least-squares analysis is then used to determine the suspension and rigid-body properties. The objective of Conti and Bretl's work was also the same as that of this thesis.

Chapter 3: Development of Analytical Model

Model Development Strategy

Several step were required to develop an analytical model of the beam. These steps are listed below in Table 3.

Table 3: Steps for Developing the Analytical Model

1. Determine theory and partial differential equations governing the behavior of the beam. (The Euler-Bernoulli beam theory and its associated differential equation and boundary conditions were used. The Timoshenko beam theory was also considered.)

2. Determine a method for solving the governing partial differential equation. (The Rayleigh-Ritz discretization technique was chosen. Explicit equations solutions and the finite element method were also considered.)

3. Determine a set of basis functions for use with the Rayleigh-Ritz method. (Quasi-comparison functions based on the pinned-pinned and slide-slide mode shapes of a uniform beam were selected. This was done in comparison with other sets of functions, such as the Fourier series.)

Performing the three steps of Table 3 provided a mathematical basis for obtaining natural frequencies and mode shapes of the beam with a number of possible boundary condition combinations. Once this mathematical framework was developed, it was implemented on a computer, as described later in the chapter, and then verified.

Basic Beam Theory

In order to develop an analytical model of an object, one can develop a set of governing partial differential equations that accurately and completely describe the physics of the problem. For a simple beam in transverse vibration, there are two theories with corresponding differential equations, fourth order in space and second order in time, that describe the physics. The Euler-Bernoulli beam theory is the classic theory since its governing partial differential equation is easily separable and can be explicitly solved using a number of techniques. Timoshenko later derived a more complete formulation that makes fewer assumptions, incorporating the effects of rotatory inertia and shear deformation.

Euler-Bernoulli Beam Model

The Euler-Bernoulli equation can be easily derived using Eq. 1, which is Hamilton's variational equation of motion for a conservative system [19]. The kinetic and potential energy terms are given in Eqs. 2 and 3. Contributions of rotatory inertia and shear deformation to the kinetic and potential energy terms are neglected in the Euler-Bernoulli formulation. Neglecting these terms makes the model less accurate for beams with a low slenderness ratio (length/radius of gyration) or at very high frequencies [20], which result in small spatial wavelengths.

$$\int_{t_1}^{t_2} (\delta T - \delta V) dt = 0 \tag{1}$$

$$T = \int_0^L \frac{1}{2} \rho A(x) \dot{w}^2 dx \tag{2}$$

$$V = \int_0^L \frac{1}{2}\left[EI(x)\left(\frac{\partial^2 w}{\partial x^2}\right)^2 + P(x)\left(\frac{\partial w}{\partial x}\right)^2 + k(x)w^2\right]dx$$
$$+ \frac{1}{2}\left[K_{t,0}w^2 + K_{r,0}\left(\frac{\partial w}{\partial x}\right)^2\right]_{x=0} + \frac{1}{2}\left[K_{t,L}w^2 + K_{r,L}\left(\frac{\partial w}{\partial x}\right)^2\right]_{x=L} \tag{3}$$

The energy terms of Eq. 2 and 3, used in the context of Hamilton's equation, will generate the equation of motion and boundary conditions given in Eqs. 4 and 5.

$$\frac{\partial^2}{\partial x^2}\left(EI(x)\frac{\partial^2 w}{\partial x^2}\right) - \frac{\partial}{\partial x}\left(P(x)\frac{\partial w}{\partial x}\right) + k(x)w + \rho A(x)\frac{\partial^2 w}{\partial t^2} = 0 \tag{4}$$

$$\left[\frac{\partial}{\partial x}\left(EI\frac{\partial^2 w}{\partial x^2}\right) - P\frac{\partial w}{\partial x} + K_{t,0}w\right]_{x=0} = 0 \qquad \left[EI\frac{\partial^2 w}{\partial x^2} - K_{r,0}\frac{\partial w}{\partial x}\right]_{x=0} = 0$$
$$\left[\frac{\partial}{\partial x}\left(EI\frac{\partial^2 w}{\partial x^2}\right) - P\frac{\partial w}{\partial x} + K_{t,L}w\right]_{x=L} = 0 \qquad \left[EI\frac{\partial^2 w}{\partial x^2} - K_{r,L}\frac{\partial w}{\partial x}\right]_{x=L} = 0 \tag{5}$$

This formulation for conservative systems is very general, including the effects of tension, base stiffness, and transverse and rotatory springs at both ends of the beam. Including these effects makes the equation of motion and boundary conditions suitable for use in a variety of problems. Damping was not included since the model was to be used only for computation of undamped mode shapes.

Timoshenko Beam Model

The beam model developed by Timoshenko makes fewer assumptions than the Euler-Bernoulli model, as it includes the effects of shear deformation and rotatory inertia [20]. This increases the accuracy of the model, particularly if the beam has a slenderness ratio (length/thickness) less than 10 or is excited at higher frequencies.

While the formulation is more complete, the model results in two differential equations, making it more difficult to use. Equations 6 and 7 give the kinetic and potential energy terms that, when used in Hamilton's principle, Eq. 1, provide the equations of motion and boundary conditions given in Eqs. 8 and 9 [20].

$$T = \int_0^L \frac{1}{2}\left[\rho A(x)\dot{w}^2 + \rho I(x)\dot{\varphi}^2\right]dx \tag{6}$$

$$\begin{aligned} V = \int_0^L \frac{1}{2}\left[EI(x)\left(\frac{\partial^2\varphi}{\partial x^2}\right)^2 + r'GA(x)\left(\frac{\partial w}{\partial x} - \varphi\right)^2 + P(x)\left(\frac{\partial w}{\partial x}\right)^2 + k(x)w^2\right]dx \\ + \frac{1}{2}\left[K_{t,0}w^2 + K_{r,0}\left(\frac{\partial w}{\partial x}\right)^2\right]_{x=0} + \frac{1}{2}\left[K_{t,L}w^2 + K_{r,L}\left(\frac{\partial w}{\partial x}\right)^2\right]_{x=L} \end{aligned} \tag{7}$$

$$\begin{aligned} \frac{\partial}{\partial x}\left(r'GA(x)\left(\frac{\partial w}{\partial x} - \varphi\right)\right) + \frac{\partial}{\partial x}\left(P(x)\frac{\partial w}{\partial x}\right) - k(x)w - \rho A(x)\frac{\partial^2 w}{\partial t^2} = 0 \\ \frac{\partial}{\partial x}\left(EI(x)\frac{\partial\varphi}{\partial x}\right) + r'GA(x)\left(\frac{\partial w}{\partial x} - \varphi\right) - \rho I(x)\frac{\partial^2\varphi}{\partial t^2} = 0 \end{aligned} \tag{8}$$

$$\begin{aligned} \left[r'GA\left(\frac{\partial w}{\partial x} - \varphi\right) + P\frac{\partial w}{\partial x} - K_{t,0}w\right]_{x=0} = 0 \qquad \left[EI\frac{\partial\varphi}{\partial x} - K_{r,0}\frac{\partial w}{\partial x}\right]_{x=0} = 0 \\ \left[r'GA\left(\frac{\partial w}{\partial x} - \varphi\right) + P\frac{\partial w}{\partial x} - K_{t,L}w\right]_{x=L} = 0 \qquad \left[EI\frac{\partial\varphi}{\partial x} - K_{r,L}\frac{\partial w}{\partial x}\right]_{x=L} = 0 \end{aligned} \tag{9}$$

If the slenderness ratio of the beam is very high, there is very little difference between the results of the Euler-Bernoulli theory and the Timoshenko theory. Because the beam used in the experiment has a very high slenderness ratio of 210, the Euler-Bernoulli theory was used. The negligible differences between the two theories at low frequencies with a slenderness ratio of 210 did not warrant the additional complexity of using the Timoshenko theory.

Continuous Eigensystem Solution Techniques

It is possible to explicitly solve the Euler-Bernoulli equation for a number of different boundary condition combinations. These solutions are typically solved by separation of variables, which results in a distributed-parameter system yielding a continuous eigenvalue problem. Several examples of this are presented by Meirovitch [21] and Gorman [22]. However, for the situation in which springs are attached to the beam, closed-form analytical solutions cannot always be generated. The problem must then be solved with a discretization technique.

In practice, discretization techniques are almost always used in problem solutions. Explicit solutions can only be generated for structures with very simple geometries, loading, and boundary conditions. These ideal problems rarely occur in practice.

Finite Element Method

The most popular method for discretizing a continuous system is the finite element method. In this method, simple approximating polynomials are used to approximate the response of the structure over small regions on the structure. This technique is very powerful, primarily because of its flexibility in dealing with structures of arbitrary shape. However, using the finite element method for vibration problems derived from static stiffness can require consider computational power to solve accurately for vibrational modes. The finite element method was not used in this work.

Rayleigh-Ritz Method

Another approach for discretizing continuous vibrational systems is to use a complex approximating function over the entire domain of the structure, which is fundamentally different from the finite element method's use of simple functions over small portions of the structure. One implementation of this second technique is known as the Rayleigh-Ritz method, which can be applied to any system that can be characterized by a self-adjoint differential operator L and boundary condition operators B, as given in Eqs. 10 and 11 [23]. L is of order $2p$, and B is of order p. For the fourth order Euler-Bernoulli equation, p equals 2.

$$L = \frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2}{\partial x^2}\right) - \frac{\partial}{\partial x}\left(P\frac{\partial}{\partial x}\right) + k \tag{10}$$

$$\begin{aligned} B_{1,0} &= \left[\frac{\partial}{\partial x}\left(EI\frac{\partial^2}{\partial x^2}\right) - P\frac{\partial}{\partial x} + K_{t,0}\right]_{x=0} & B_{2,0} &= \left[EI\frac{\partial^2}{\partial x^2} - K_{r,0}\frac{\partial}{\partial x}\right]_{x=0} \\ B_{1,L} &= \left[\frac{\partial}{\partial x}\left(EI\frac{\partial^2}{\partial x^2}\right) - P\frac{\partial}{\partial x} + K_{t,L}\right]_{x=L} & B_{2,L} &= \left[EI\frac{\partial^2}{\partial x^2} - K_{r,L}\frac{\partial}{\partial x}\right]_{x=L} \end{aligned} \tag{11}$$

These operators are used to specify the continuous generalized eigenvalue problem given in Eq. 12, which provides a solution to the free vibration problem given in Eqs. 4 and 5, as shown in Eq. 13.

$$\begin{aligned} Ly &= \lambda \rho A y, \quad 0 < x < L \\ B_{i,x}y &= 0, \quad i = 1, 2, \ldots, p, \quad \text{for } x = 0, L \end{aligned} \tag{12}$$

$$w(x, t) = y(x)\cos\left(\omega_0 t + \theta_0\right) \tag{13}$$

In the Rayleigh-Ritz method, a series of functions are added together to approximate the mode shapes of the system, as shown in Eq. 14. When this technique is derived using Rayleigh's method as a basis, these functions have strict differentiability and boundary condition requirements. Specifically, the functions must be $2p$-times differentiable over the entire domain and must satisfy all boundary conditions of the original problem exactly [23]. These functions are known as comparison functions. Deriving comparison functions for a system with generalized boundary conditions can be quite difficult.

$$y_n(x) = \sum_{i=1}^{n} a_i \phi_i(x), \quad 0 < x < L \tag{14}$$

The coefficients of the functions must be selected to find a stationary point of Rayleigh's quotient for the system, as defined in Eq. 15. This minimization process is equivalent to the generalized algebraic eigenvalue problem of Eq. 16, with terms defined in Eqs. 17 and 18. The $\vec{a}$ vector gives the coefficients of the functions used in the summation. There are n solutions to the problem of Eq. 16, each of which corresponds to a natural mode of vibration of the system. Because the system is self-adjoint, the ω_0^2 terms are always real-valued. Additionally, since all of the energy terms used to define the problem are non-negative, then the ω_0^2 terms will also always be zero or positive-valued [19].

$$R = \frac{(yLy)}{(\rho A y^2)} = \frac{\sum_i \sum_j a_i a_j \int_0^L \phi_i L(\phi_j) dx + K_{t,0} w_0^2 + K_{t,L} w_L^2 + K_{r,0} \left(\frac{dw}{dx}\right)_0^2 + K_{r,L} \left(\frac{dw}{dx}\right)_L^2}{\sum_i \sum_j a_i a_j \int_0^L \rho A \phi_i \phi_j dx} \tag{15}$$

$$[K]\vec{a} = \omega_0^2 [M]\vec{a} \tag{16}$$

$$\begin{aligned} k_{ij} = \int_0^L \phi_i L \phi_j dx \; + \; K_{t,0}\phi_{i,0}\phi_{j,0} + K_{t,L}\phi_{i,L}\phi_{j,L} \\ + K_{r,0}\left(\frac{d\phi_i}{dx}\right)_0 \left(\frac{d\phi_j}{dx}\right)_0 + K_{r,L}\left(\frac{d\phi_i}{dx}\right)_L \left(\frac{d\phi_j}{dx}\right)_L \end{aligned} \tag{17}$$

$$m_{ij} = \int_0^L \rho A \, \phi_i \phi_j dx \tag{18}$$

A derivation based on energy terms, however, reveals that the solution technique of Eq. 16 will work even if the functions used in the summation are only *p*-times differentiable and match only the essential boundary conditions, i.e., those boundary conditions in which displacements or slopes are fixed [23]. These functions are known as admissible functions. This leads to a difficulty of a different nature. The set of choices for admissible functions is so broad that determining which functions will work best is difficult.

Admissible Functions

While the general requirements for a function to be admissible are weak, there are strong requirements that a set of admissible functions must meet if they are to accurately approximate the response of the structure. The set of admissible functions must be linearly independent and complete, and it must be possible to form linear combinations of the functions that match the problem boundary conditions exactly. If these conditions are not met, an inaccurate or very slowly converging solution may result.

An example of a set of functions that satisfies the linear independence and completeness requirements, but not the boundary condition requirement, is the Fourier series. This series can exactly match a given function if enough terms are included in the series. Specifically, the Fourier series converges to the solution in the limit. Unfortunately, this requires solving Eqs. 16 and 17 with an infinite number terms in the Ritz series. If the series is used with only a finite number of terms, the end conditions of the solution may be matched poorly. For example, the first flexural mode of a free-free beam has a positive slope at one end and a negative slope at the other. All of the sine terms have the same slope at both ends, while the cosine terms have the same displacement at both ends, as shown in Figures 3 and 4.

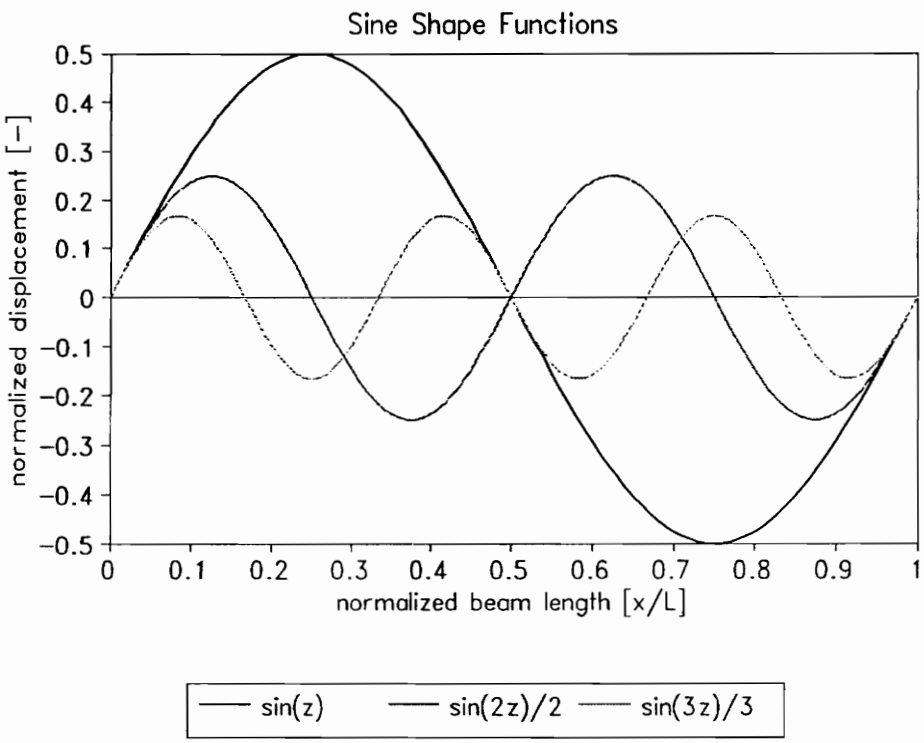

Figure 3: Fourier Sine Terms

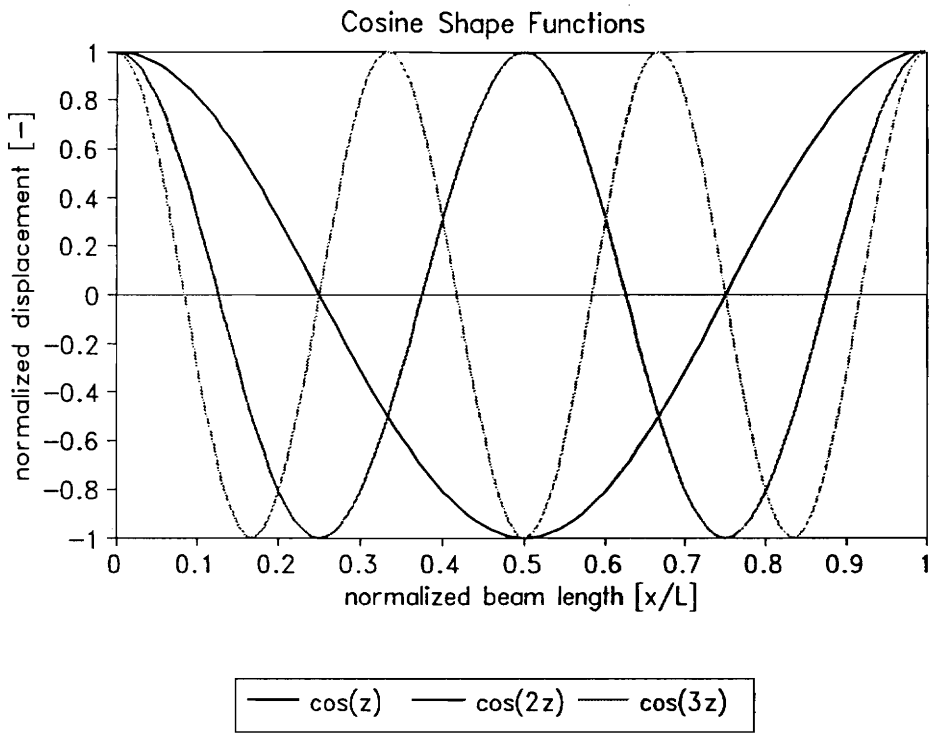

Figure 4: Fourier Cosine Terms

Adding a finite number of these terms will never result in different slopes at the end. Similarly, the second flexural mode of the beam has a positive displacement at one end and a negative displacement at the other. Since the sine terms have no displacement at either end and the cosine terms all have a displacement of +1 at both ends, a finite number of terms will again fail to generate a function with different displacements at each end.

Meirovitch [8] proposed the concept of quasi-comparison functions to resolve the problem of slow convergence. In this technique, admissible functions are specifically selected such that linear combinations can be used to match any possible linear boundary condition. Meirovitch's example was a rod experiencing axial vibration. The variable cross-section rod was fixed at one end and spring loaded at the other.

The admissible functions used to model the response of the rod were the mode shapes of a uniform rod with fixed-fixed ends and of a rod with fixed-free ends. A linear combination of a fixed-fixed mode shape and a fixed-free mode shape can be used to match any linear boundary condition caused by the spring, thus forming the equivalent of a comparison function with only two terms. The quasi-comparison functions were compared in convergence rate to actual comparison functions and to the finite element method. In all cases it slightly outperformed the comparison function method and significantly outperformed the finite element method in terms of convergence rate.

The beam modeled in this study was modeled with quasi-comparison functions. In an interview, Meirovitch suggested using combinations of the modes shapes generated by free-free and fixed-fixed boundary conditions, examples of which are given by Figs. 5 and 6 and specified in Eq. 19.

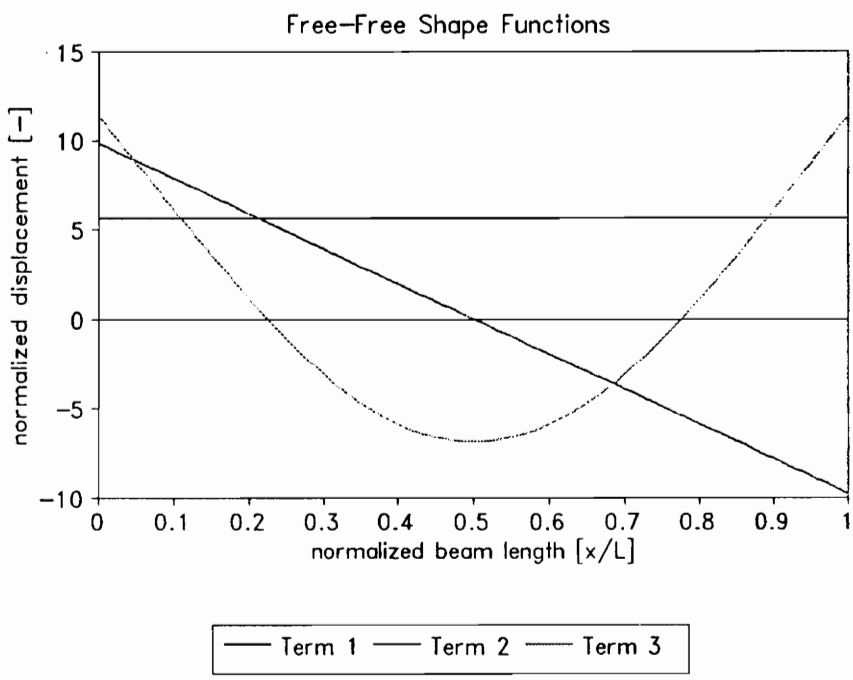

Figure 5: Suggested Quasi-Comparison Functions (from free-free)

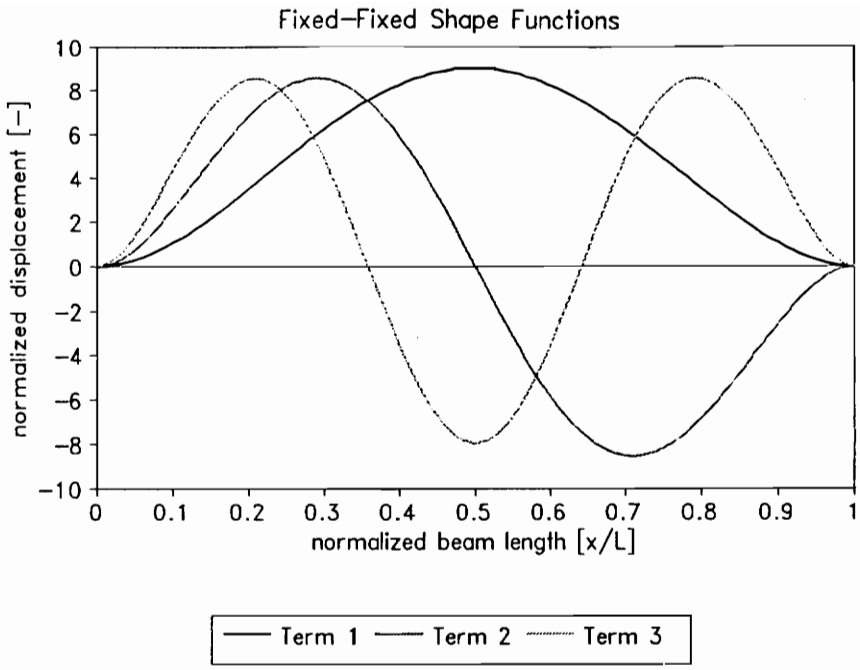

Figure 6: Suggested Quasi-Comparison Functions (from fixed-fixed)

$$
\begin{aligned}
\phi_1 &= 1 \\
\phi_2 &= \frac{x}{L} - \frac{1}{2} \\
\phi_3 &= 0.8568\left(\sin 1.875\frac{x}{L} + \sinh 1.875\frac{x}{L}\right) + 1.1671\left(\cos 1.875\frac{x}{L} + \cosh 1.875\frac{x}{L}\right) \\
\phi_4 &= 1.0092\left(\sin 4.694\frac{x}{L} + \sinh 4.694\frac{x}{L}\right) + 0.9909\left(\cos 4.694\frac{x}{L} + \cosh 4.694\frac{x}{L}\right) \\
\phi_5 &= 0.9912\left(\sin 4.730\frac{x}{L} + \sinh 4.730\frac{x}{L}\right) - 1.0089\left(\cos 4.730\frac{x}{L} + \cosh 4.730\frac{x}{L}\right) \\
\phi_6 &= 1.0004\left(\sin 7.853\frac{x}{L} + \sinh 7.853\frac{x}{L}\right) - 0.9996\left(\cos 7.853\frac{x}{L} + \cosh 7.853\frac{x}{L}\right)
\end{aligned}
\tag{19}
$$

A minimum of four terms are needed to match any possible boundary condition. While these mode shapes can be obtained analytically, they contain hyperbolic trigonometric terms that result in difficult integrations over the beam. Therefore, linear combinations of the modes resulting from pinned-pinned and slide-slide boundary conditions were used instead, examples of which are shown in Figs. 7 and 8.

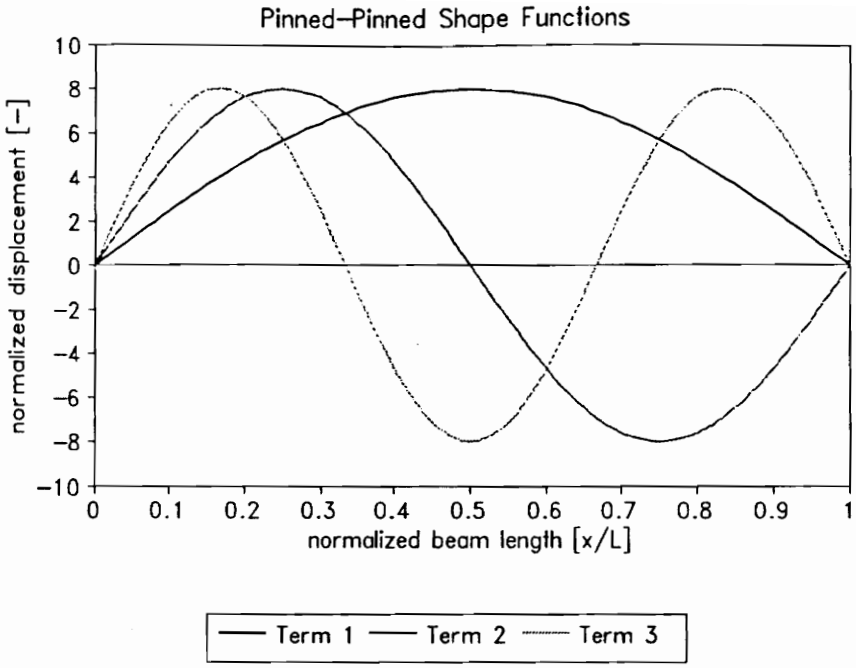

Figure 7: Alternate Quasi-Comparison Functions (from pinned-pinned)

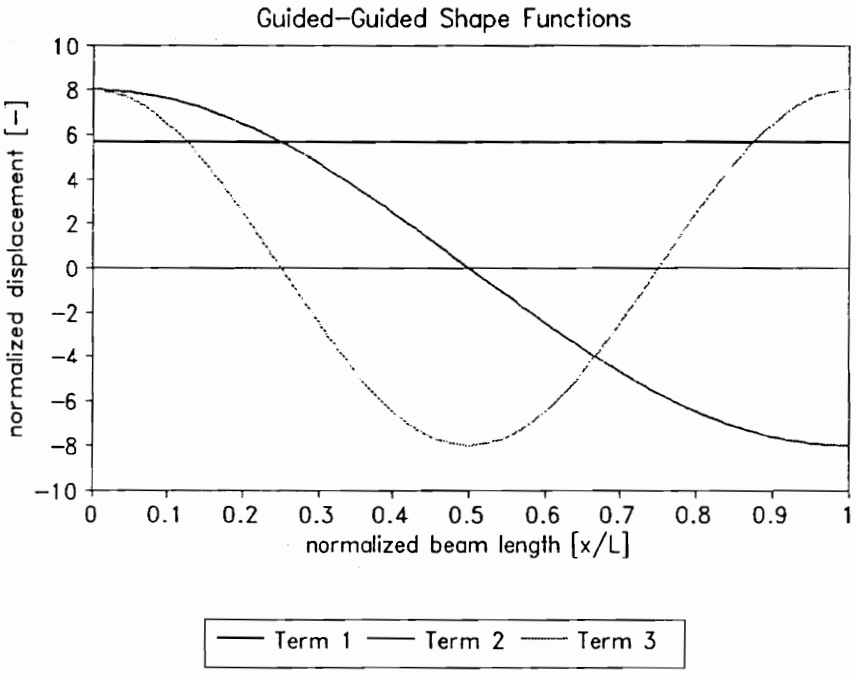

Figure 8: Alternate Quasi-Comparison Functions (from guided-guided)

With these mode shapes, boundary conditions can still be modeled with four terms, but all of the mode shapes are composed of simple sine and cosine terms. This greatly simplifies the integrations required while retaining the power of the quasi-comparison function method. Using these modes shapes as basis functions is equivalent to a Fourier series with half-periodic terms included, as seen in Eq. 20.

$$\begin{aligned} \phi_1 &= 1 \\ \phi_i &= \sin\left[\left(\frac{i+1}{2}\right)\frac{\pi x}{L}\right] \text{ for } i = 3, 5, 7, \ldots \\ \phi_i &= \cos\left[\left(\frac{i}{2}\right)\frac{\pi x}{L}\right] \text{ for } i = 2, 4, 6, \ldots \end{aligned} \tag{20}$$

Analytical Model Development Summary

The previous discussion on the Euler-Bernoulli beam theory and the Rayleigh-Ritz solution technique are summarized below. The Euler-Bernoulli beam theory was selected as a theoretical basis for the analytical model. This theory, when used in the context of Hamilton's principle, Eq. 1, results in the partial differential equation and boundary conditions of Eqs. 4 and 5. The Rayleigh-Ritz technique for solving partial differential equations was chosen for the purpose of solving the equation of motion.

The quasi-comparison function technique was used to select a set of admissible functions, Eq. 20, to be used in the Rayleigh-Ritz series. These terms were used to build mass and stiffness matrices, as shown in Eq. 17, which were then used in the eigenvalue problem of Eq. 16. The building of the system matrices and the solution of the eigenvalue problem was performed on a computer, as described in the computer-based implementation that follows. The solution of this eigenvalue problem yielded the natural frequencies of the system. For each natural frequency there is an

associated vector of coefficients $\vec{a}$ which, when used in the context of Eq. 14, yielded the mode shape corresponding to the natural frequency.

Implementation Specifics

Development of BEAM3.EXE

Since the ultimate objective of this work was to perform optimization experiments on the beam and its boundary conditions, it was necessary to develop a model solution technique with which a large number of model solutions could be calculated very quickly. In accomplishing this, the Rayleigh-Ritz method was used with quasi-comparison functions, as discussed in the previous sections, to approximate the mode shapes and natural frequencies of the beam.

The Rayleigh-Ritz integrations, from Eq. 17, were performed symbolically using Mathematica [24] with the Ritz terms of Eq. 14. A typical integration might look like the one found in Eq. 21. A change of variables $z = \frac{\pi x}{L}$ could be used to simplify this equation, yielding Eq. 22.

$$k_{ij} = \frac{ij\pi^2}{L^2}\int_0^L EI(x)\sin\left(\frac{i\pi x}{L}\right)\cos\left(\frac{j\pi x}{L}\right)dx \tag{21}$$

$$k_{ij} = \frac{ij\pi}{L}\int_0^\pi EI\left(\frac{Lz}{\pi}\right)\sin(iz)\cos(jz)\,dz \tag{22}$$

Since the $EI(x)$ parameter was treated as a constant across the entire beam, the problem was reduced to integrating mixed trigonometric terms across the domain of the beam. However, if the gravitational tension $P(x) = W\left(1-\frac{x}{L}\right)$ was to be included in

the model, additional terms including multiplication by z had to be included. Table 4 lists all of the combinations that were in the model, minus any leading coefficients.

Table 4: Rayleigh-Ritz Integration Terms

integration terms $(i \neq j)$	
$\int_0^{\pi} 1\, dz$	$\int_0^{\pi} \sin(iz)\cos(iz)\, dz$
$\int_0^{\pi} z\, dz$	$\int_0^{\pi} \cos(iz)\cos(jz)\, dz$
$\int_0^{\pi} \sin(iz)\, dz$	$\int_0^{\pi} \cos^2(iz)\, dz$
$\int_0^{\pi} \cos(iz)\, dz$	$\int_0^{\pi} z\sin(iz)\sin(jz)\, dz$
$\int_0^{\pi} z\sin(iz)\, dz$	$\int_0^{\pi} z\sin^2(iz)\, dz$
$\int_0^{\pi} z\cos(iz)\, dz$	$\int_0^{\pi} z\sin(iz)\cos(jz)\, dz$
$\int_0^{\pi} \sin(iz)\sin(jz)\, dz$	$\int_0^{\pi} z\sin(iz)\cos(iz)\, dz$
$\int_0^{\pi} \sin^2(iz)\, dz$	$\int_0^{\pi} z\cos(iz)\cos(jz)\, dz$
$\int_0^{\pi} \sin(iz)\cos(jz)\, dz$	$\int_0^{\pi} z\cos^2(iz)\, dz$

The integrals listed in Table 4 were computed symbolically using Mathematica. Figure 9 provides a sample of this work, giving the Mathematica results for two of the integration terms.

```
In[3]:= Simplify[Integrate[Sin[i z] Sin[j z], {z, 0, pi}]]

Out[3]= (i Sin[(i - j) pi] + j Sin[(i - j) pi] - i Sin[(i + j) pi] +
                                                   2    2
>       j Sin[(i + j) pi]) / (2 (i  - j ))

In[4]:= Simplify[Integrate[z Sin[i z] Cos[j z], {z, 0, pi}]]

                 3                                  2
Out[4]= (-(i  pi Cos[(i - j) pi]) - i  j pi Cos[(i - j) pi] +
                 2                                  3
>       i j  pi Cos[(i - j) pi] + j  pi Cos[(i - j) pi] -
          3                                  2
>       i  pi Cos[(i + j) pi] + i  j pi Cos[(i + j) pi] +
          2                                  3                                  2
>       i j  pi Cos[(i + j) pi] - j  pi Cos[(i + j) pi] + i  Sin[(i - j) pi] +
                                                   2                   2
>       2 i j Sin[(i - j) pi] + j  Sin[(i - j) pi] + i  Sin[(i + j) pi] -
                                                   2                                  2    2 2
>       2 i j Sin[(i + j) pi] + j  Sin[(i + j) pi]) / (2 (i  - j ) )
```

Figure 9: Sample Mathematica Results

The results of Figure 9 appear quite complicated, but they can be simplified since the indices i and j are always non-zero integers. The relations given by Eq. 23 were used to substantially simplify the results, providing the integrals in closed form. The sample Mathematica results from Figure 9 are equivalent to the expressions given in Eq. 24.

$$\begin{aligned} \sin(i\,\pi) &= 0 \\ \cos(i\,\pi) &= +1 \quad \text{if } i \text{ is odd} \\ \cos(i\,\pi) &= -1 \quad \text{if } i \text{ is even} \end{aligned} \tag{23}$$

$$\begin{aligned} &\int_0^{\pi} \sin(iz)\sin(jz)\,dz = 0 \\ &\int_0^{\pi} z\sin(iz)\cos(jz)\,dz = \frac{\pi(j^2-i^2)\cos(\pi(i+j))}{(i^2-j^2)^2} \qquad \text{for } i \neq j \end{aligned} \tag{24}$$

The integration results were implemented in a IBM-PC based matrix assembly program (see Appendix D) designed to work with MATLAB [25] MAT files, in

which matrices are stored in IEEE double-precision format. The beam and suspension parameters were sent to the program using a MAT file, and the mass and stiffness matrices were returned to MATLAB using a second MAT file. Using this system for transferring matrices prevented the loss of any precision between the matrix assembly program and MATLAB. The use of MAT files also enabled MATLAB to call the matrix assembly program directly. This was specifically done to allow the MATLAB Optimization Toolbox [26] to be used in optimization experiments with the model.

Difficulties Encountered and Solved

The matrix assembly program went through four major revisions before being used for the final work. The first version transferred data with ASCII files rather than MAT files, and this relatively inaccurate data storage format combined with poorly conditioned matrices caused imaginary frequencies to be generated for mass and stiffness matrices larger than 11 by 11. The second version, which worked directly with MAT files, was useful with systems as large as 19 by 19. The third version of the program was formulated with direct design parameters, such as length and thickness, rather than the traditional combined beam parameters, such as EI and m. Finally, the fourth version solved the eigenvalue problem directly using a Jacobi-iteration based symmetric generalized eigensystem solver (see Appendix A) rather than using the solver built into MATLAB. This routine specifically utilized symmetry and increased numeric precision, enabling the program to work with systems as large as 27 by 27. The source code for this program is given in Appendix D.

The primary source of difficulty in solving the beam problem was ill-conditioning that occurred in the mass and stiffness matrices. According to the Rayleigh-Ritz theory, the discretized mass matrix is positive definite, and the stiffness matrix is positive definite if there are no rigid-body modes; it is otherwise positive semi-definite [27]. However, if the series of functions used for the approximation is not linearly independent, the positive definite character of the matrix can be lost. A look at the eigenvalues and condition number of the mass matrix (which remained constant for a given beam with a given number of summation terms) shows how the matrix eigenvalues approach zero as the number of terms is increased. This is indicative of a matrix approaching singularity. Table 5 shows this trend.

Table 5: Ill-Conditioning of Mass Matrix

terms	$n = 7$	$n = 9$	$n = 11$
eigen-values	--	--	0.0000000036
	--	0.0000001101	0.0000003242
	0.0000032794	0.0000076425	0.0000153275
	0.0001578016	0.0002632472	0.0003195041
	0.0031573992	0.0032649727	0.0037480720
	0.0156133142	0.0163376762	0.0156133142
	0.0290391704	0.0279616558	0.0281048567
	0.0310688269	0.0310595565	0.0309071244
	0.0458667222	0.0312189859	0.0312170274
	--	0.0460192954	0.0312263043
	--	--	0.0462079125
condition number	1.39×10^4	4.18×10^5	1.28×10^7

The reason for this tendency towards singularity is not immediately apparent. It may be due to the fact that whole wavelength terms comprise the equivalent of a

truncated Fourier series, which can be used to approximate functions. If enough whole wavelength terms are used, any possible mode shape can be approximated. The half-wavelength terms comprise additional terms in the series. Adding terms to a nearly complete truncated series is certain to cause lack of linear independence, which will cause the matrices to become singular.

Need for Utilization of Symmetry

A second difficulty that occurred during the quasi-comparison function modeling of the beam mode shapes was the presence of large, negative eigenvalues in the eigensystem solution. This occurred when the MATLAB solver was used with a high number of terms in the series, typically 21 or more.

Due to the Rayleigh-Ritz formulation of the beam, the eigensystem should be either positive definite or positive semi-definite. This implies that negative eigenvalues should not occur. Therefore, when negative eigenvalues did occur, the results of the entire solution were suspect. The ill-conditioning of the matrices discussed in the previous section is the most probable cause for this problem. This was a second motivation for incorporating a Jacobi-iteration based solver into the final version of the program. In addition to providing additional accuracy, the algorithm is specific to symmetric, positive semi-definite systems [28]. Table 6 shows how the new solver performed in comparison to the MATLAB eigensolver. The imaginary frequencies generated by MATLAB are highlighted to emphasize where it failed to calculate correct results for a positive semi-definite 27 term system. This solver was useful for up to 29 terms, at which point the mass matrix lost positive definiteness because of matrix ill-conditioning.

Table 6: Comparison of Eigensystem Solvers

Mode Number	MATLAB natural frequencies [Hz]	Jacobi natural frequencies [Hz]
r(1)	0	2.6e-13
r(2)	**1.7e-11i**	2.2e-12
1	17.5	17.5
2	48.2	48.2
3	94.5	94.5
4	156	156
5	233	233
6	326	326
7	434	434
????	*** 536i ***	--
8	558	558
9	697	697
10	851	851
11	1021	1021
12	1206	1206
13	1407	1407
14	1623	1623
15	1855	1855
16	2107	2106
17	2432	2380
18	2825	2791
19	3719	3223
20	4699	4449
21	8791	5378
22	13129	10902
23	51569	13880
????	*** 183554i ***	--
24	--	65016
25	--	89862

Small negative eigenvalues sometimes resulted from the use of the Jacobi-based eigensolver, but these were always small perturbations from zero, probably caused by rounding and truncation error during the solution process. These small negative values were safely rounded up to zero, as they were clearly errors occurring at the limits of machine precision.

Model Summary

Table 7 below recounts the three steps required for the theoretical development of the model, as given in Table 3, in addition to including the steps needed to implement the model on the computer.

Table 7: Complete Steps for Developing the Model

1. The Euler-Bernoulli beam theory was selected as the theory describing the behavior of the beam.

2. The Rayleigh-Ritz discretization technique was selected as the method used to solve the governing partial differential equations and boundary conditions.

3. Quasi-comparison functions based on the pinned-pinned and slide-slide mode shapes of a uniform beam were selected as the set of basis functions to use with the Rayleigh-Ritz method.

4. The closed form of the Rayleigh-Ritz mass and stiffness integrations were determined using Mathematica.

5. The closed form of the Rayleigh-Ritz integrations were placed in a program that assembled the mass and stiffness matrices.

6. The mass and stiffness matrices generated by the program were used in a generalized eigenvalue problem. This problem was solved using a Jacobi-iteration based generalized eigenvalue solver.

7. The results of the eigenvalue problem directly provided the natural frequencies of the beam along with eigenvectors of coefficients that, when combined with the Rayleigh-Ritz basis functions, provided the corresponding mode shapes.

Validation of Rayleigh-Ritz Analytical Model

In order to use the quasi-comparison analytical model to approximate the behavior of the beam, it had to be validated. This was done by comparing results from the model to known analytical solutions for certain combinations of simple boundary conditions. Natural frequencies were used as the quantitative basis for comparison, and the mode shapes were examined qualitatively to see if they matched expected results as well.

For all cases, the beam parameters of Table 8 were used. These are the same as the parameters used in the actual model reconciliation of Chapter 7, as given in Table 31.

Table 8: Analytical Model Beam Parameters

1. Length = 54.75 in.
2. Thickness = 0.2605 in.
3. Width = 3.00 in.
4. Density = 0.282 lbf/in^3
5. Elastic modulus = 28.0×10^6 lbf/in^2

Closed-Form Analytical Solutions

Analytical results were available for three different sets of boundary conditions, as found in Meirovitch [21]. These combinations are given in Table 9.

Table 9: Analytical Cases Tested

1. free-free (no external stiffnesses)
2. pinned-pinned (infinite translational stiffnesses at both ends)
3. fixed-free (infinite translational and rotational stiffnesses at the left end)

Free-free Beam Case

The free-free end condition was the one most heavily examined, as it most closely approximates the conditions likely to be found in the study. It was therefore used as the basis of a convergence study. By Theorem 1, given below, the Rayleigh-Ritz method converges to the true solution from above as terms are added to the series. This occurs because each additional term creates another row and column to the system matrices. This pushes all of the frequencies down except for the new one added. The frequencies never drop below the actual solution frequencies.

Theorem 1: Inclusion Principle

If $A_{n\times n}$ is a symmetric matrix and $B_{n+1\times n+1} = \begin{bmatrix} A & \vec{x} \\ \vec{x}^T & x_0 \end{bmatrix}$, then the eigenvalues λ of A and B meet the following inequalities:

$$\lambda_{B,1} \le \lambda_{A,1} \le \lambda_{B,2} \le \lambda_{A,2} \le \ldots \le \lambda_{B,n} \le \lambda_{A,n} \le \lambda_{B,n+1}$$

Table 10 shows the quickness with which convergence is reached using the quasi-comparison functions. For the free-free boundary conditions, it appears that only 15 terms are required to accurately model the first four flexural modes of the beam. At 19 terms, the natural frequencies converge to machine precision. The first two modes listed, $r(1)$ and $r(2)$, are the two rigid-body modes of the free-free beam.

Table 10: Free-free Convergence Study

n	Natural frequency of mode [Hz]					
	r(1)	r(2)	1	2	3	4
5	0.00	4.89	17.53	48.69	127.58	--
7	0.00	1.67	17.50	48.53	95.56	204.70
9	0.00	0.44	17.50	48.28	94.58	158.11
11	0.00	0.10	17.50	48.23	94.55	156.36
13	0.00	0.02	17.50	48.23	94.55	156.31
15	0.00	0.00	17.50	48.23	94.55	156.30
theory	0.00	0.00	17.50	48.23	94.55	156.30

Figures 10 and 11 show that the error in the natural frequency estimates, defined in Eq. 25, decays exponentially after approximately 9 Ritz terms are used in the model. When exponential decay is plotted on a semi-log plot, the result is a straight line with a negative slope. This type of convergence is desirable, since a small increase in model complexity, i.e., one or two addition terms, results in a substantial accuracy gain. The higher modes have higher errors initially, but they all appear to converge at the same rate after initial variations.

$$err_f = f_{est,n} - f_{theory} \tag{25}$$

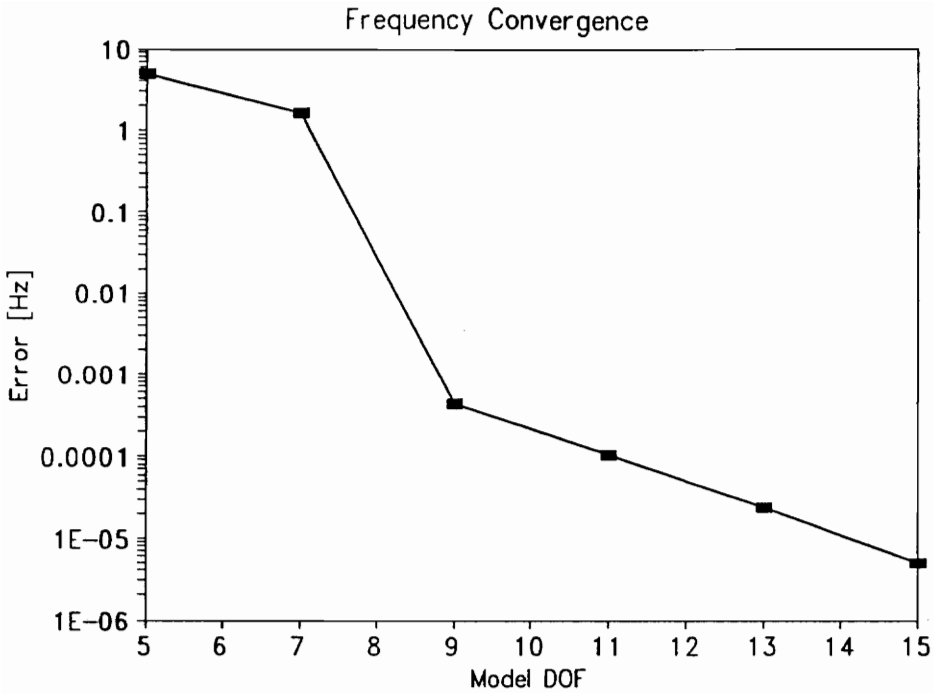

Figure 10: Rigid-Body Zero-Frequency Convergence

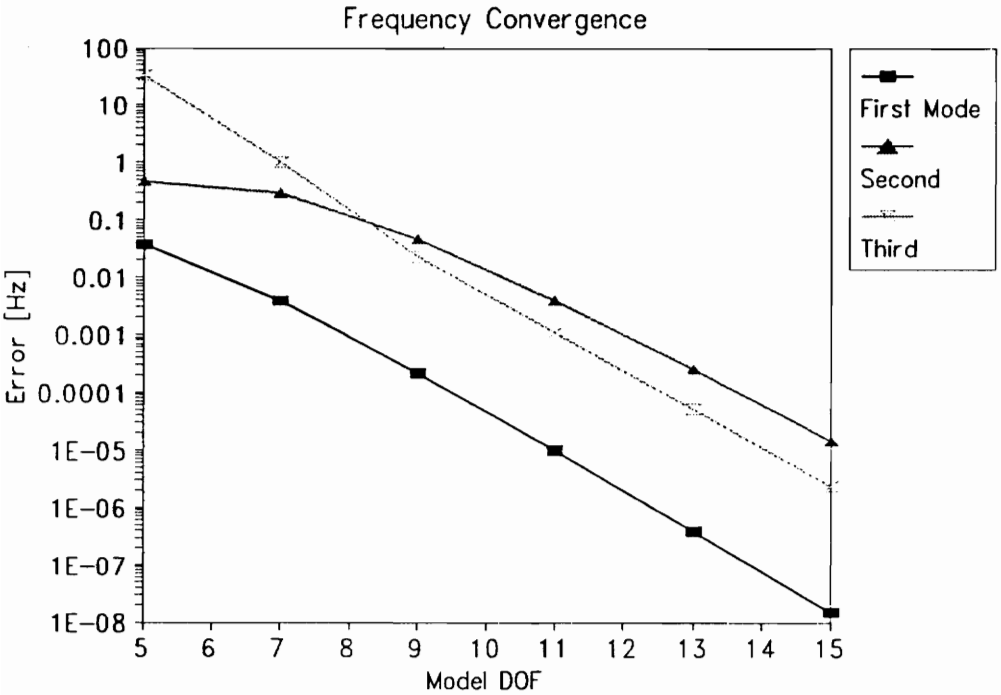

Figure 11: Flexural Mode Frequency Convergence

Another indication of whether or not the model is valid is the accuracy of the mode shapes. Figure 12 shows the first four mode shapes obtained from the model, where all mode shapes were normalized such that RMS displacement was unity. These mode shape estimates visually match what is expected from theory.

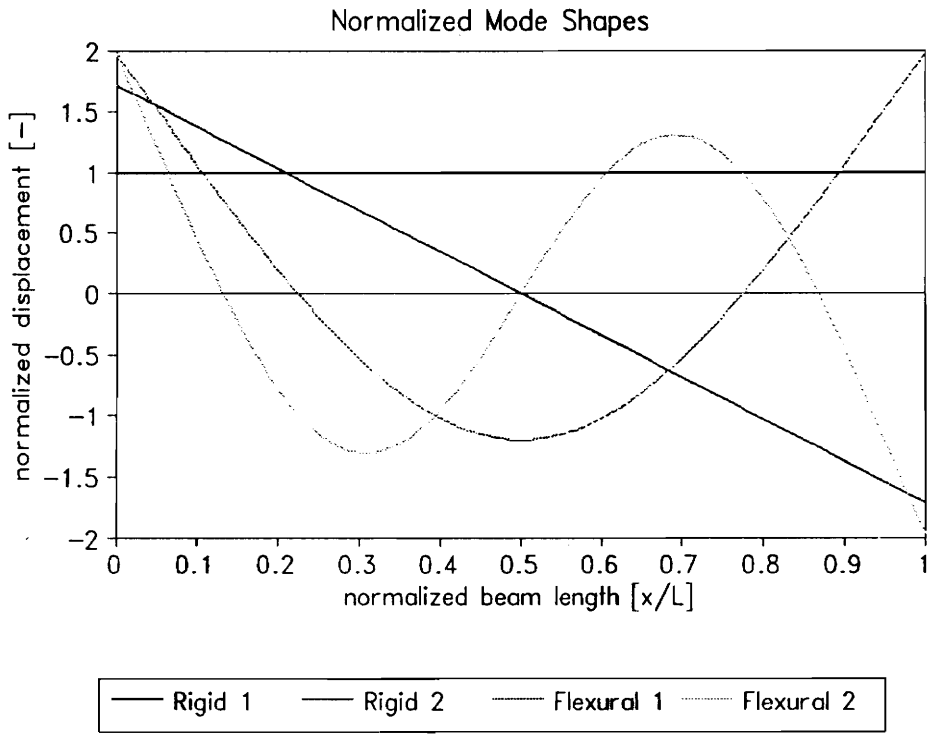

Figure 12: Free-free Mode Shapes

The mode shapes for the free-free case were also compared to analytical mode shapes quantitatively. For an example, the fourth flexural mode of the free-free beam is examined here. Figure 13 below shows the difference between a theoretical mode shape and estimates of mode shape obtained with a varying number of Ritz terms. Equation 26 defines the error relationship.

$$\Delta y_i = y_{m,i} - y_{theory,i} \tag{26}$$

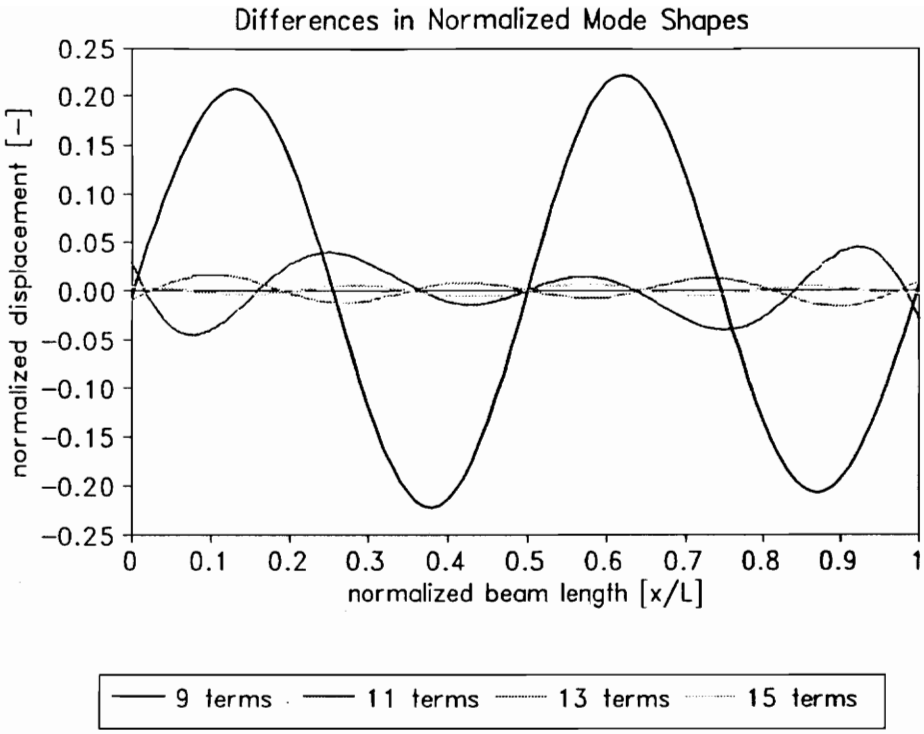

Figure 13: Convergence of Fourth Mode Shape

Equation 27 defines the RMS error in the mode shape, using the error relationship defined in Eq. 26. Table 11 below shows how the RMS error in the mode shape decreases with an increasing number of Ritz terms. This data is also plotted in Fig. 14, which shows that the decrease in error is similar to that of the natural frequencies, as seen in Figs. 10 and 11.

$$err_{RMS} = \sqrt{\sum_{i=1}^{n} \Delta y_i^2 / n} \tag{27}$$

Table 11: RMS Error Convergence of Fourth Mode

Ritz Terms	RMS error
9	0.1517
11	0.02541
13	0.008866
15	0.003644

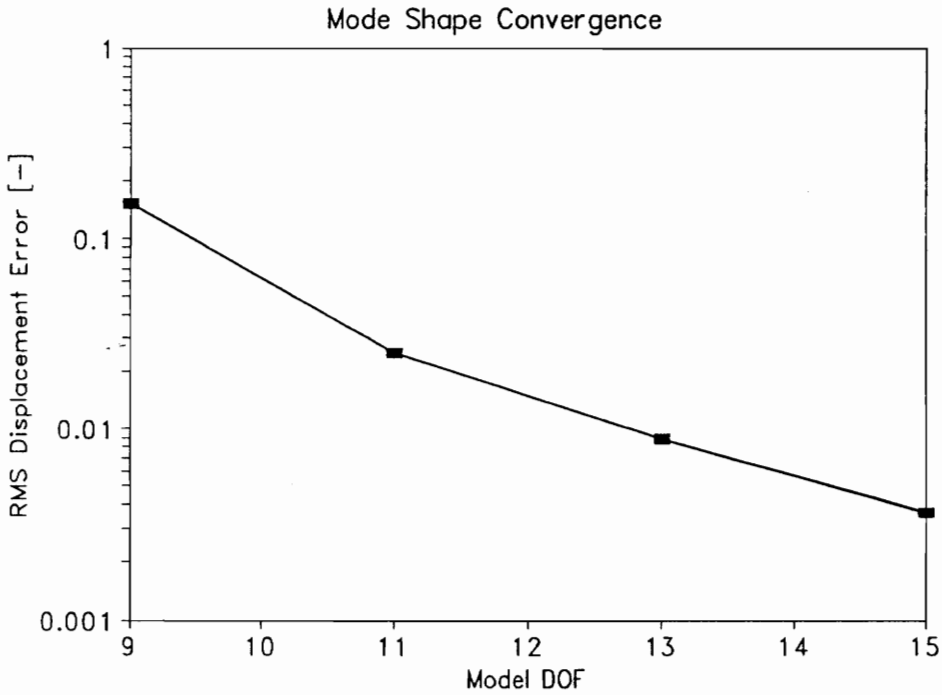

Figure 14: RMS Error Convergence of Fourth Mode

Pinned-pinned Beam Case

The model was studied next by approximating pinned-pinned boundary conditions on the beam, as shown in Figure 15. This was done to confirm that transverse stiffnesses added to the model gave meaningful results.

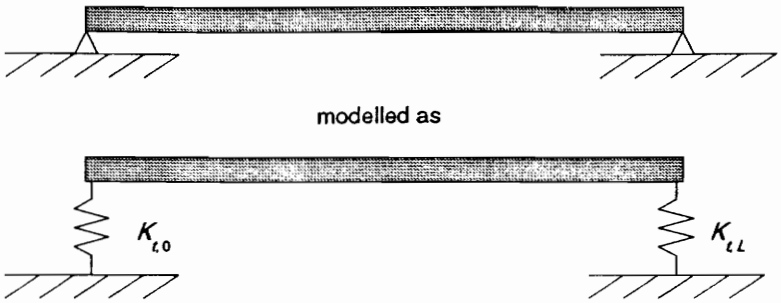

Figure 15: Pinned-pinned Beam

This setup was studied in a slightly different manner than the free-free beam, focusing on convergence of natural frequencies as a function of stiffness rather than as a function of the number of terms in the Ritz series as was done with the free-free beam in Table 10. Varying the number of Ritz terms results in frequency convergence much like that of Table 10, except that the final frequencies obtained are a function of the spring stiffness values used. The nineteen Ritz terms used here are sufficient to generate accurate results. Table 12 shows how the natural frequencies of the pinned-pinned beam converged to theoretical pinned-pinned beam results as stiffnesses at both ends of the beam were increased. Figure 16 shows the mode shapes for stiffness values of 10,000 lbf/in. The dashed lines near zero represent the error when compared to the theoretical shapes.

Table 12: Pinned-pinned Convergence Study

stiffness [lbf/in]	Natural frequency of modes [Hz]					
	r(1)	r(2)	1	2	3	4
0	0.00	0.00	17.50	48.23	94.55	156.30
10	3.63	6.87	19.32	48.91	94.90	156.51
1e2	6.73	18.87	31.60	55.34	98.16	158.44
1e3	7.61	29.07	60.12	94.50	131.84	181.21
1e4	7.71	30.69	68.53	120.47	185.28	261.03
1e6	7.72	30.87	69.46	123.47	192.89	277.72
theory	7.72	30.87	69.47	123.50	192.97	277.87

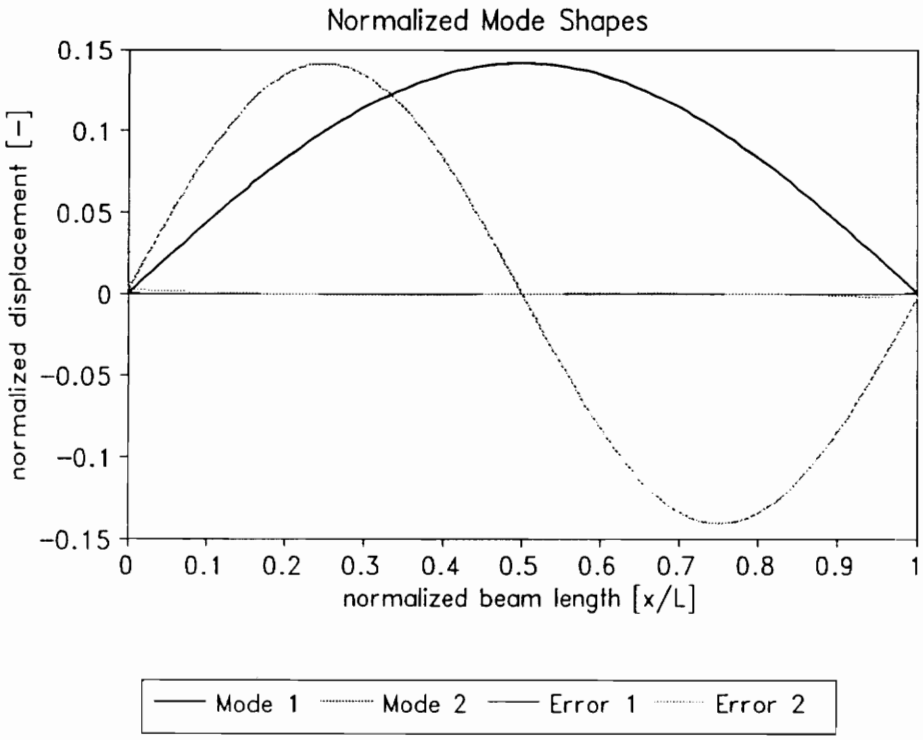

Figure 16: Pinned-pinned Mode Shapes

Fixed-free Beam Case

Finally, the fixed-free boundary conditions were studied in a manner similar to the pinned-pinned boundary conditions. Fixed-free boundary conditions result from a cantilever beam setup, as shown in Figure 17.

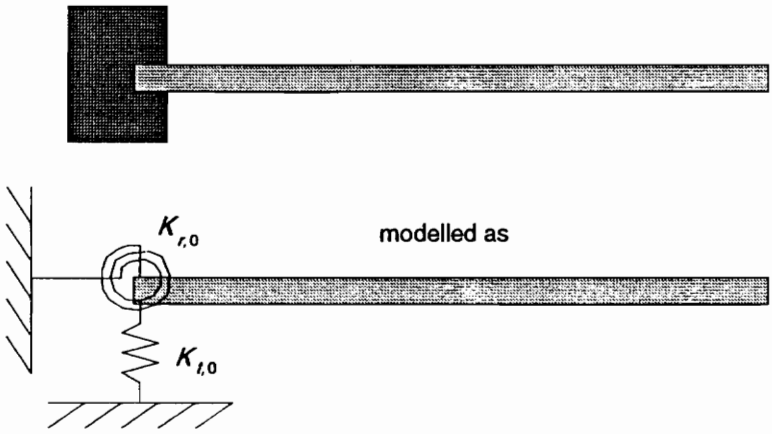

Figure 17: Fixed-free (Cantilever) Beam

The translational stiffness was held near infinity while the rotatory stiffness was increased from zero to near infinity. Varying the number of Ritz terms results in frequency convergence much like that of Table 10, except that the final frequencies obtained are again functions of the spring stiffness values used. The nineteen terms used here are more than sufficient to generate accurate results. The rotatory stiffness convergence for the fixed-free case is tabulated in Table 13. Figure 18 shows the mode shapes for a rotatory stiffness of 10^6 lbf·in/rad, with thin lines near zero representing error in the mode shape estimates, Eq. 26.

Table 13: Fixed-free Convergence Study

stiffness [lbf·in]	Natural frequency of modes [Hz]		
	r(1)	r(2)	1
0	0.00	12.06	39.08
1e2	0.28	12.09	39.11
1e4	1.99	14.09	41.65
1e6	2.74	17.16	48.04
1e8	2.75	17.25	48.24
1e10	2.75	17.25	48.25
theory	2.75	17.25	48.25

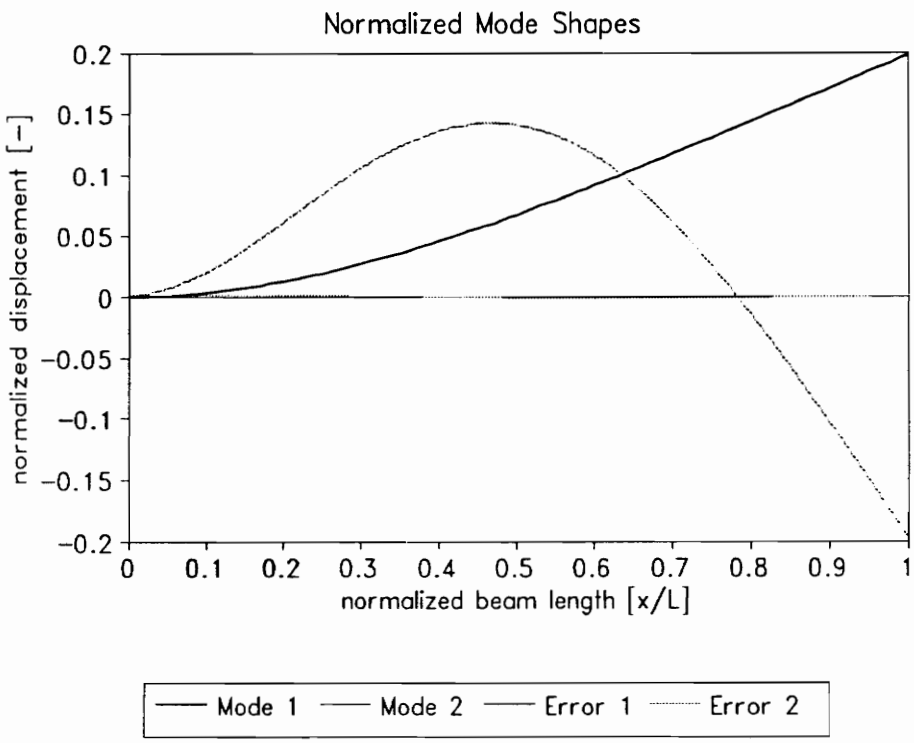

Figure 18: Fixed-free Mode Shapes

Continuum Mechanics Solution (BEAM VI)

While zero and infinite stiffness boundary conditions are easily tested analytically, intermediate values are not. To validate the model on intermediate values, a

numerical package known as BEAM VI [29] was used. BEAM VI is a program based on transfer-matrix techniques for solving beam problems. In this approach, a finite number of uniform sections are joined and analyzed together.

The finite stiffness cases from Tables 12 and 13 were run on BEAM VI, where they generated identical results, matching to at least 8 significant figures. Also, BEAM VI was used to validate the analytical model on cases in which base stiffness was present. Table 14 shows the effects of several base stiffness values for the base stiffness problem. Again, results were identical for BEAM VI and the analytical model.

Table 14: BEAM VI Base Stiffness Study

base stiffness [lbf/in^2]	Natural frequency of modes [Hz]				
	r(1)	r(2)	1	2	3
0	0.00	0.00	17.50	48.23	94.55
10	21.07	21.07	27.39	52.64	96.87
100	66.64	66.64	68.91	82.26	115.68
1000	210.74	210.74	211.47	216.19	230.98

Since the results from the quasi-comparison function based Rayleigh-Ritz model closely matched the results from known analytical solutions and results from a continuum mechanics based BEAM VI solution, it can be concluded that the model is effective for modeling a beam with external stiffnesses added to the ends of the beam.

Gravitational Tension Convergence

The convergence and validation studies discussed above focused on variations of the free-free beam problem for which independent solutions were available. However, in the physical experiment in which the beam was tested, there was a tension effect caused by gravity loading. This tension, P, is described in Eq. 28, where W is the weight of the beam and x is the distance measured from the top of the beam. Figure 19 shows how the tension varies down the beam.

$$P(x) = W\left(1 - \frac{x}{L}\right) \tag{28}$$

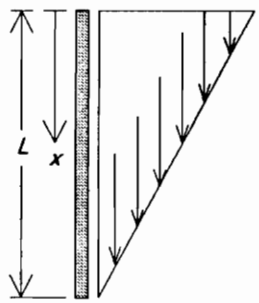

Figure 19: Tensile Load on Beam

The natural frequencies and mode shapes computed under the tension effect could not be compared to other solutions, since no other solutions were available. However, it could still be tested for convergence. Table 15 shows that the increase in natural frequency caused by the gravity effect had not converged within 0.01 Hz even with 27 terms in the Rayleigh-Ritz model. In contrast, a beam with only external stiffnesses but no gravity effect required only 15 terms. Ill-conditioning in the mass and stiffness matrices prevented the use of more Ritz terms for the model that included gravity. The reasons for the slow convergence are not known. The zero

gravity results are included only for comparison purposes. It is *not* expected that the results would converge to the zero gravity values.

Table 15: Gravitational Convergence Study

n	Natural frequency of mode [Hz]					
	r(1)	r(2)	1	2	3	4
7	0.00	2.19	17.86	48.85	95.90	205.52
11	0.00	1.61	17.78	48.64	94.80	156.60
15	0.00	1.39	17.76	48.52	94.78	156.55
19	0.00	1.31	17.74	48.47	94.76	156.52
23	0.00	1.27	17.72	48.45	94.74	156.50
27	0.00	1.25	17.70	48.44	94.72	156.49
zero gravity	0.00	0.00	17.50	48.23	94.55	156.30

There is, however, evidence that the results were converging to a fixed solution, primarily in that the rate of decrease in natural frequencies dropped as the number of Ritz terms was increased. Figure 20 shows how estimates of the natural frequencies of the second rigid-body mode and the first flexural mode converged. When compared to the results of Figures 10 and 11, it is apparent that the natural frequency estimates converge very slowly when gravitational tension is present. Approximately 12 terms are required in the series to cut the natural frequency errors by a factor of ten. In a model in which a gravitational tension load is not considered, about 3 terms are required to achieve a similar drop in rigid-body mode natural frequency estimates, while only 1.5 terms are required for flexural modes.

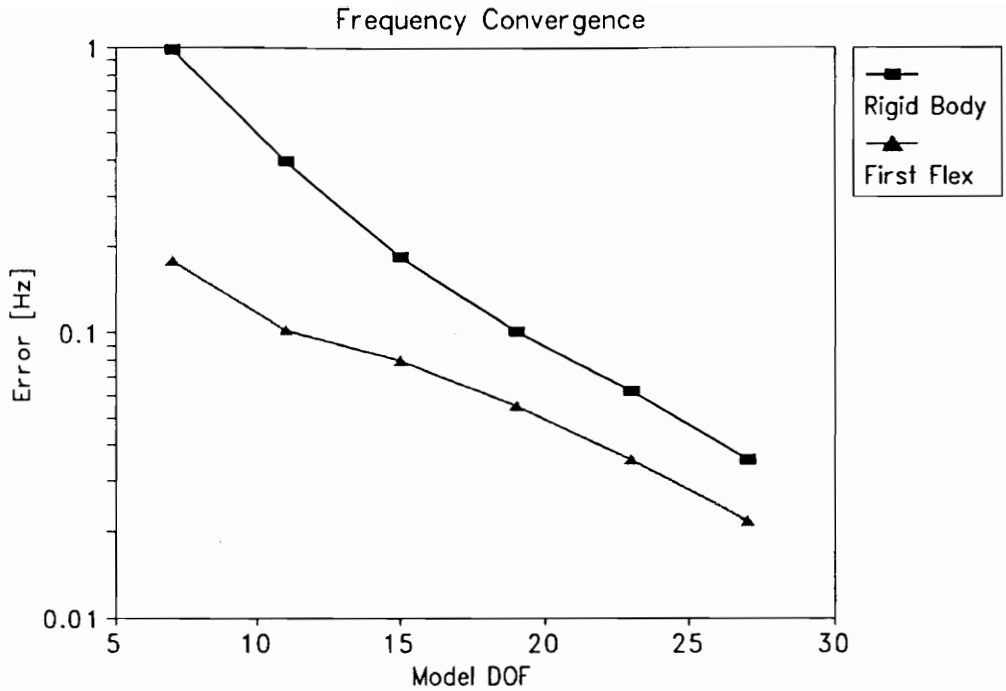

Figure 20: Convergence of Gravitational Tension Effect

The difficulty with convergence here led to the question of whether or not the quasi-comparison function analytical model was appropriate for modeling the gravitational body load. The effect was left in the model because it is known to exist physically, although it appears to change the natural frequencies of the beam by at most 0.20 Hz. It is expected that an incompletely converged gravity effect will more accurately model the beam than no gravity effect at all. However, comparing results with a model <u>not</u> including the gravity effect may provide insight as to whether or not the gravity effect should be included.

Model Verification Summary

The Rayleigh-Ritz model, based on quasi-comparison functions and implemented on the computer, was run through several tests, as listed in Table 16.

Table 16: Model Verification Tests

1. Rayleigh-Ritz natural frequencies must converge from above (as a function of number of Ritz terms) to fixed frequencies, as implied by the inclusion principle of Theorem 1.

2. Natural frequencies for the free-free, pinned-pinned, and fixed-free beam cases must match theoretical values exactly, given a sufficient number of Ritz terms.

3. Natural frequencies for the free-free, pinned-pinned, fixed-free, and base stiffness cases must exactly match results from another computer-based beam solution implementation, BEAM VI.

4. Since no theoretical solution for a beam with gravitationally induced tension was available, it was decided that natural frequencies must converge to fixed frequencies, given a sufficient number of Ritz terms.

The model implementation easily passed the first three items of Table 16. The fourth item, however, was not strictly passed. It appeared that the natural frequencies obtained from the model were converging, but not quickly enough to provide natural frequency estimates accurate to 0.01 Hz before matrix ill-conditioning prevented the use of additional Ritz terms.

Perturbation Study

The convergence studies discussed earlier focused primarily on the natural frequencies of the beam. However, additional insight into the system can be gained

by examining the mode shapes. A second analytical study was performed in which the boundary conditions of the beam were perturbed slightly by adding a small stiffness or mass to the top of the beam. A case in which gravity was added was also studied. Natural frequencies for each condition were computed at 17 and 25 terms to see how well the effects were converged. The convergence study of Table 10 demonstrated that only 15 terms are necessary to model the beam without any added external masses or stiffnesses. Table 17 lists the five conditions studied.

Table 17: Perturbation Conditions

1. tranverse spring at top: $k = 1.0$ lbf/in
2. rotatory spring at top: $k = 100$ lbf·in
3. linear mass at top: $m = 0.0001$ lbf·sec^2/in
4. rotational inertia at top: $m = 0.0001$ lbf·sec^2
5. gravitational tension at top: $g = 386.4$ in/sec^2

Examination of the natural frequencies given in Table 18 show that the effects of both springs and both masses are well converged, matching to at least six significant digits. As expected from the study of Table 15, the natural frequencies obtained with the gravity load did not match for the differing number of Ritz terms, which was evidence of incomplete convergence.

Table 18: Perturbation Study Convergence

effect	first mode natural frequency [Hz]		difference in freq. [Hz]
	17 terms	25 terms	--
transverse spring	17.5906	17.5906	0.0000
rotatory spring	17.5635	17.5635	0.0000
translational mass	17.3874	17.3874	0.0000
rotational inertia	17.4965	17.4965	0.0000
gravitational tension	17.7355	17.7084	0.0271

Figures 21 through 25 show the effect that each of the five perturbations of Table 9 has on the mode shapes of the beam. The perturbation study was performed to provide insight into how the beam suspension and any attached measurement transducers might affect the mode shapes of the beam. Equation 29 gives the formula used to generate the perturbation effect plots, where the normalized mode shapes came from the 25-term perturbed and unperturbed models.

$$\Delta y = y_{perturb} - y_{unperturbed} \tag{29}$$

Transverse Stiffness Effects

Figure 21 shows that the mode shapes of the lower modes were affected most by a transverse spring. Additionally, it was discovered that adding a transverse stiffness to the top of the beam increased the mode shape displacement at the top, which was a counter-intuitive result. This effect was confirmed with BEAM VI. The system suspension is likely to be source of the stiffness causing the mode shape differences seen in Fig. 21.

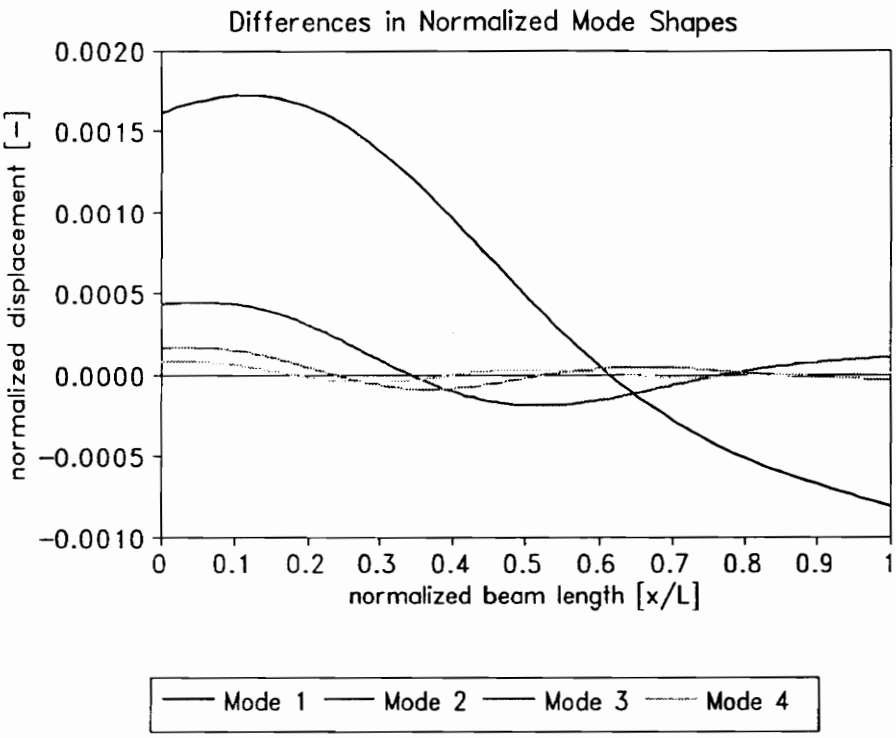

Figure 21: Effects of Translational Stiffness

Rotatory Stiffness Effects

Figure 22 shows how the first four mode shapes were affected by the addition of a rotatory stiffness at the beam tip. There was no apparent trend characterizing which mode shapes were affected "the most." The stinger used to connect the beam to an electromagnetic shaker is a possible source of rotatory stiffness.

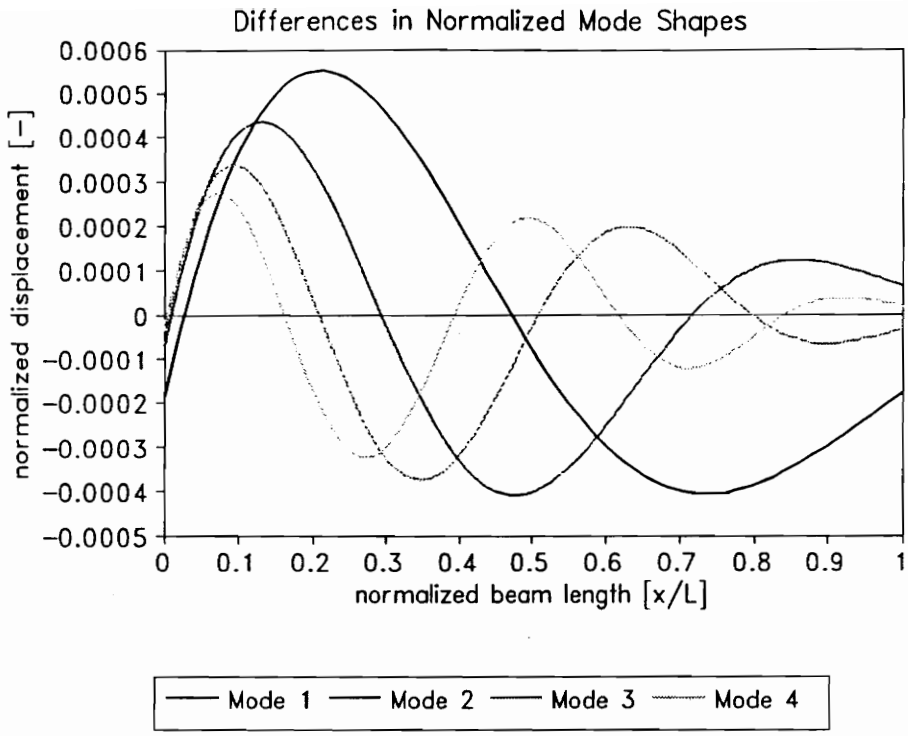

Figure 22: Effects of Rotatory Stiffness

Translational Mass Effects

Figure 23 shows how a translational mass added at the top of the beam alters its mode shapes. In particular, the tip displacement is decreased substantially for the higher modes. The lower mode shapes are affected less. An examination of Fig. 23 suggests that the amount of tip displacement decrease is almost directly proportional to the mode number. A quantitative analysis could confirm this. The force gauge used to measure the excitation force contributes partly to this translational mass.

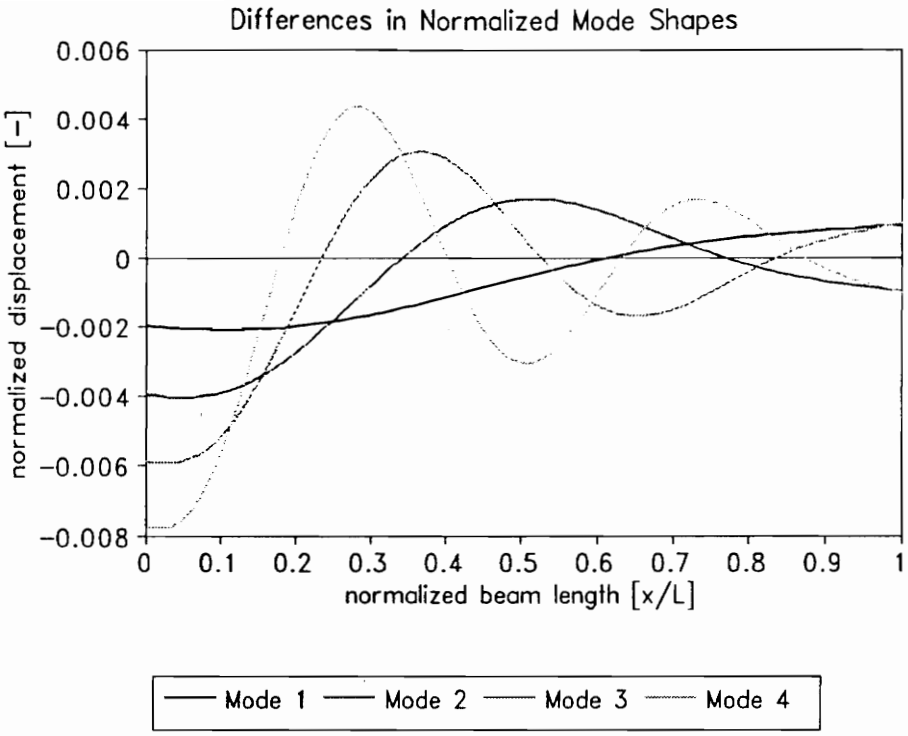

Figure 23: Effects of Translational Mass

Rotational Inertia Effects

Figure 24 shows how the addition of rotatory inertia at the top of the beam affects the mode shapes. As with the addition of translational mass, the addition of rotatory inertia affects the mode shapes of higher modes significantly more than it does the mode shapes of lower modes. However, these changes are not manifested as variations in tip deflections, as seen in Fig. 23, but rather in tip rotations. The force gauge used to measure excitation force contributes to this rotatory inertia in addition to being a partial source of translational mass.

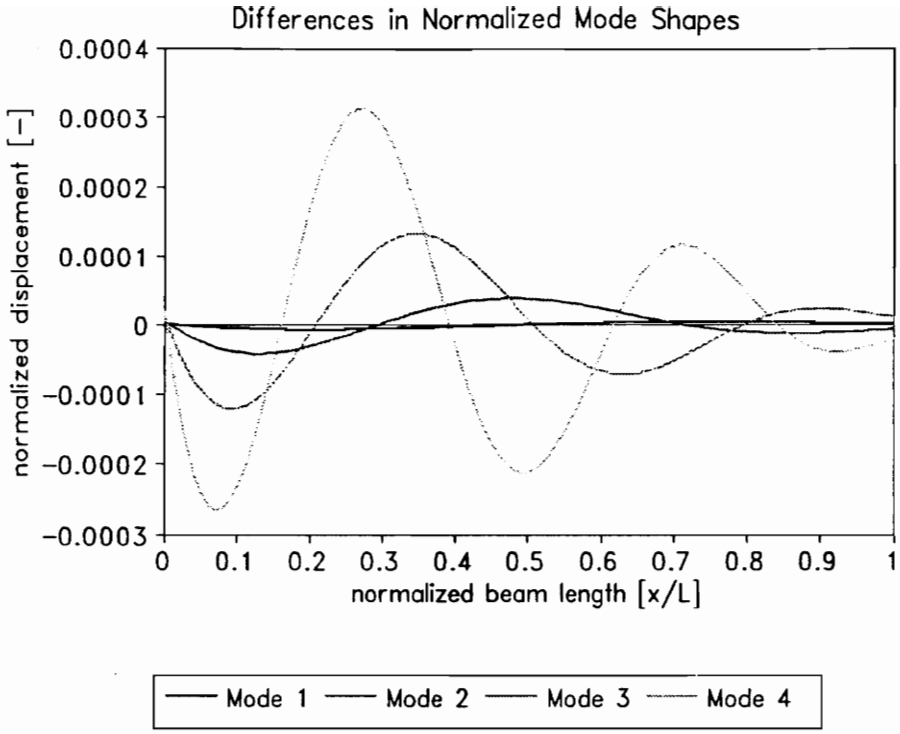

Figure 24: Effects of Rotatory Inertia

Gravitational Tension Effects

Figure 25 clearly shows the incomplete convergence of the gravitational tension effect caused by the weight of the beam. All of the mode shape differences have an spatial oscillatory component with a wavelength of approximately 0.18 times the length of the beam (5.5 wavelengths total). This corresponds to twice the wavelength of the last two sine and cosine terms used in the Ritz series, possibly indicating a convergence problem. Despite this problem, the effect was left in the model. Although it is difficult to see on the plot, the tip of the first mode shape is being pulled down. Leaving it out would artificially reduce the translational stiffness values obtained.

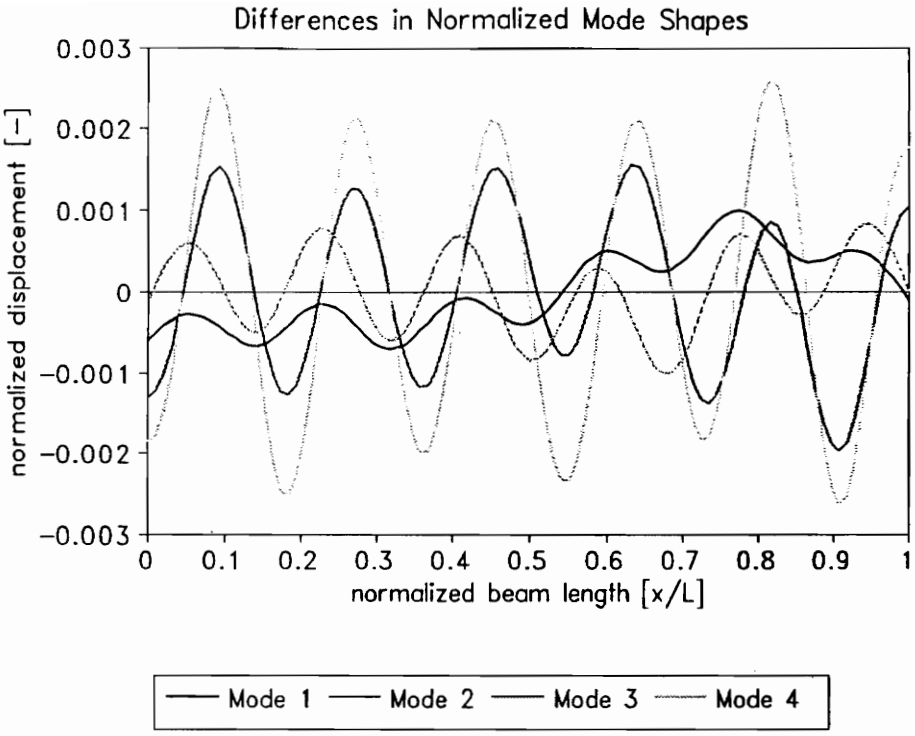

Figure 25: Effects of Gravitational Tension

Perturbation Study Summary

The perturbation study was performed to provide a "feel" for the effects that the presence of the system suspension and measurement transducers might have on the mode shapes of the beam. The perturbation study also suggests the quality of the data that must be acquired if one is to extract these terms from experimental data.

It is important to note that these results are only valid for the first few modes of the beam. At higher modes, the Euler-Bernoulli model loses validity because it neglects shear effects and the rotatory inertia of the beam itself. Fortunately, the lower modes of a structure are the ones that usually dominate its behavior.

Chapter 4: Study of Suspension

Transverse Stiffness Effect

In many modal tests, shock cord is used to suspend a test structure in mid-air. Ideally, the structure would be vibrating under free boundary conditions, but in reality the cord exhibits a transverse stiffness effect when the attachment point is moved in a direction perpendicular to the shock cord. In the case of a vertically suspended simple beam, the beam tip moves in this manner, as shown in Fig. 26.

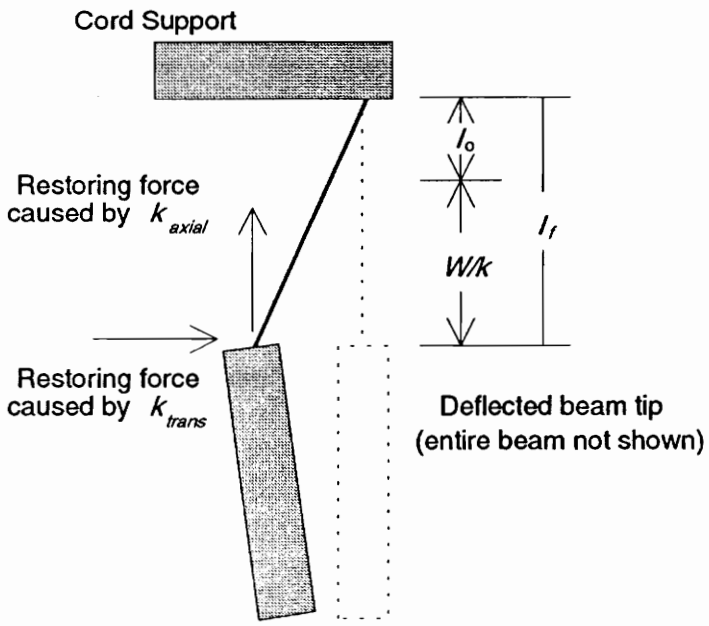

Figure 26: Shock Cord Elongation and Motion

If the assumptions of Table 19 are made, an equivalent transverse stiffness can be calculated if the axial stiffness of the shock cord is known.

Table 19: Transverse Stiffness Assumptions

1. The tip of the beam experiences only horizontal motion. The beam is quite heavy, and forcing it to vibrate up and down significantly at the excitation frequency would require large variations in the tension of the cord.

2. The tip deflections are small enough such that additional elongation caused by the deflection does not significantly change the tension.

3. The shock cord has negligible mass and remains in a straight line connecting the cord clamp and the tip of the beam.

4. The shock cord behaves as a linear spring with an equivalent zero-load length. Again, the small changes in the length of the cord (described in assumption 2) should make this assumption reasonable.

Equation 30 results from the application of these assumptions. If the equation is expressed in terms of the final cord length instead of the equivalent initial length, a much simpler equation results, as seen in Eq. 31.

$$K_{trans} = \left[1 - \frac{l_0}{l_0 + W/K_{axial}}\right] \tag{30}$$

$$K_{trans} = K_{axial}\left[1 - \frac{l_f - W/K_{axial}}{l_f}\right] = \frac{W}{l_f} \tag{31}$$

It is important to note that if the assumptions of Table 19 are correct, the transverse stiffness effect of the shock cord is independent of the cord's axial stiffness, except in the fact that the final length of the cord will vary with the cord's axial stiffness.

Linearity Test

An experiment was conducted to check assumption 4 of Table 19. In the experiment, a single shock cord of known zero-load length was tied to a rod at the top and a lightweight bucket at the bottom, as shown in Figure 27.

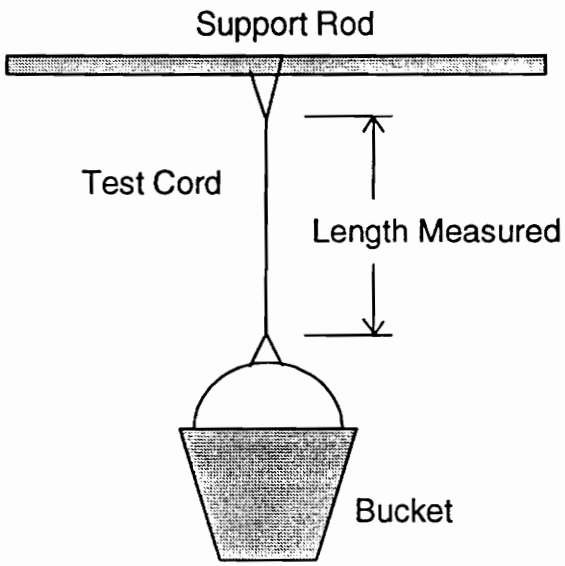

Figure 27: Shock Cord Test Set-up

Water was added in small, fixed volumes to the bucket, increasing the load on the shock cord and stretching it further. The cord was lightly plucked at each water level to alleviate any hysteresis effect. The length was recorded at each step, yielding points on a volume/displacement curve. The displacements were then divided by the original length and the volume multiplied by the specific weight of water to yield a load/elongation curve. Once this was done, multiple linear regression was used to fit the functions of Equation 32 to the curve with an r^2 value of 0.9997. These three functions comprise a straight line and two smoothed ramp functions.

$$
\begin{aligned}
f_1(x) &= x \\
f_2(x) &= (x - a_2)\left[\frac{1}{2} - \frac{\arctan((x - a_2)/b_2)}{\pi}\right] \\
f_3(x) &= (x - a_3)\left[\frac{1}{2} + \frac{\arctan((x - a_3)/b_3)}{\pi}\right]
\end{aligned}
\tag{32}
$$

The a parameters specifies the points at which the "bend" in the ramp is centered, and the b parameters specifies how sharp the "bends" are. The multiple linear regression was repeated several times with the a and b parameters being adjusted manually to provide the best fit. Figure 28 shows the load/elongation data with the fitted curve, with Fig. 29 showing the residuals of this fit. The residuals have a standard deviation of 0.082, which is quite small compared to typical linear load values (2 to 8 lbf).

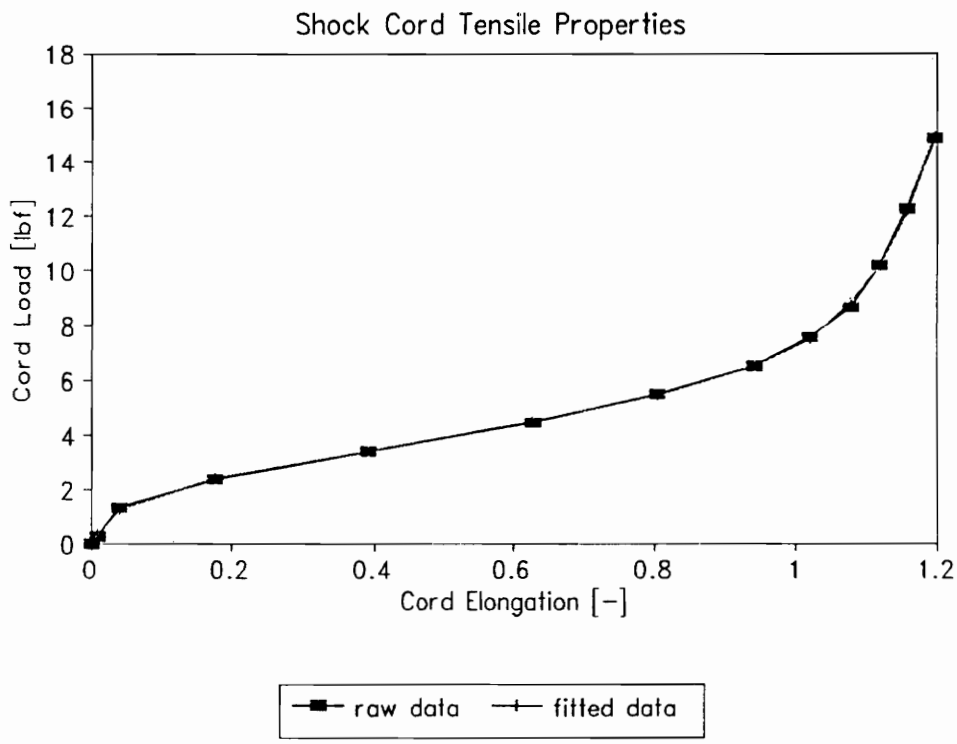

Figure 28: Shock Cord Load/Elongation Fit

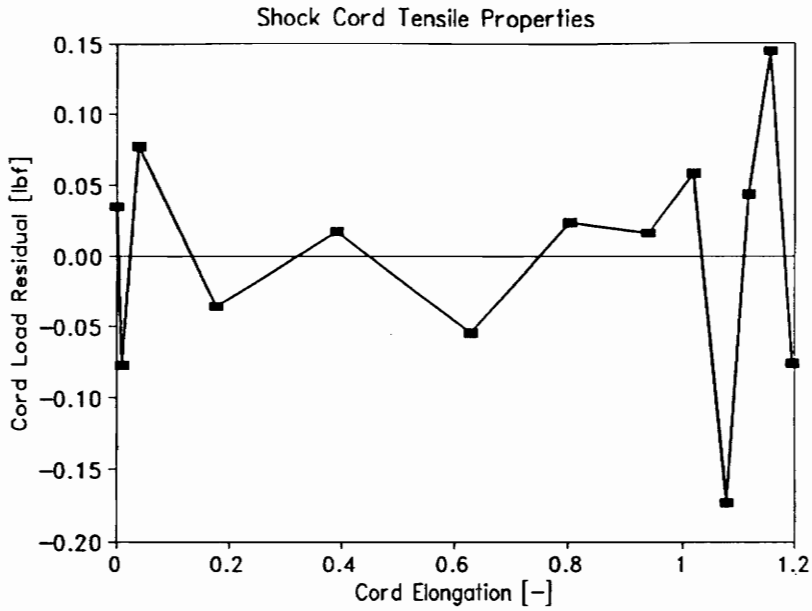

Figure 29: Shock Cord Load/Elongation Residuals

Once the load/elongation curve had been fitted with the functions, the regression equation was differentiated to yield a normalized stiffness/elongation curve, as shown in Eq. 33.

$$KL = \beta_1 \frac{\partial f_1(x)}{\partial x} + \beta_2 \frac{\partial f_2(x, a_2, b_2)}{\partial x} + \beta_3 \frac{\partial f_3(x, a_3, b_3)}{\partial x} \tag{33}$$

To find the stiffness of the cord, the elongation x must be determined first. This can be directly read from the load/elongation curve, which is shown in Fig. 30. This elongation value can be used with Eq. 33 to determine a normalized stiffness value. Dividing the normalized stiffness by the original length of the cord yields the axial stiffness of the cord. Figure 30 shows normalized stiffness values over the elongation range tested. Over an elongation range of 20% to 80%, the assumption of linearity is shown to be quite reasonable. The non-linearity at high elongations was

anticipated, but the non-linearity at low elongations was unexpected. It is perhaps due to the collapse of the nylon sheath surrounding the elastic core of the shock cord.

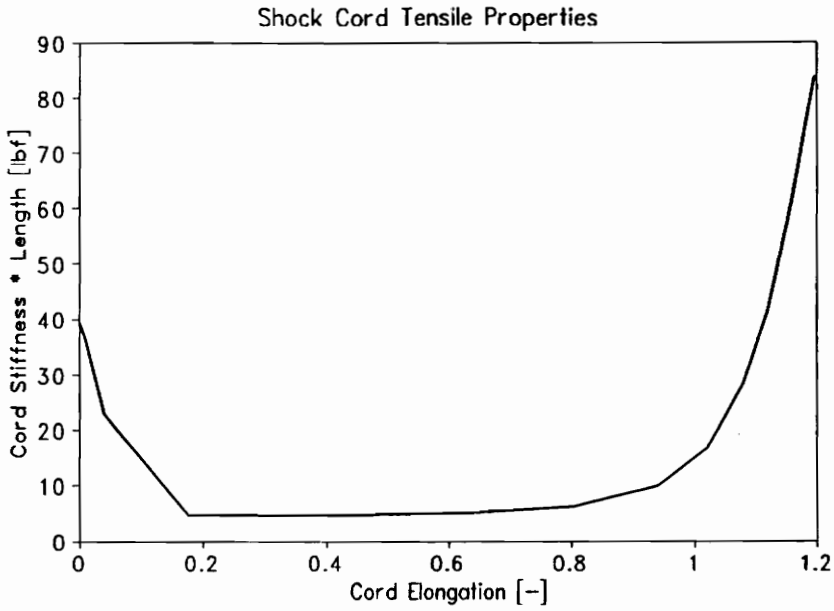

Figure 30: Shock Cord Stiffness Property

In the modal tests, as discussed later, the beam was suspended with either 1, 2, or 3 cords. Since the beam weighed about 12 lbs, dividing the load up between 2 or 3 cords kept the cords in the linear behavior region in the center of Figure 30. Placing the entire load on a single cord caused it to enter the non-linear behavior region on the right.

If the assumptions in Table 19 do not hold, motion of a non-linear nature may occur. If this is the case, the use of modal summation techniques to model linear beam behavior is no longer valid since the entire system is being modeled, not just the beam. A proper analysis of the problem becomes considerably more complex in this case.

Chapter 5: Techniques for Reconciliation

Development of Reconciliation Technique

The objective of the research presented in this thesis is to reconcile an analytical model of a structure with data obtained in dynamic experiments. This required the development of a model reconciliation methodology. The steps for this methodology are given in Table 20.

Table 20: Steps in Reconciliation Methodology

1. Determine the location of uncertainty in the model. In this case, the shock cord used to support the beam was the least understood part of the system.

2. Determine update parameters for the model that will reduce uncertainty in the model when accurately specified. Here, transverse and rotatory stiffness parameters were used to specify the effect the shock cord had on the beam.

3. Determine a means for comparing the analytical model to data obtained from experiments. Here, analytical mode shapes were compared to experimental mode shapes in a least-squares fashion.

4. Determine a means for updating the parameters to minimize the difference between the analytical model and the results obtained from experiments. In this case, optimization (function minimization) was selected as the methodology of choice. Several algorithms were investigated here.

5. Determine how sensitive the parameter estimates are to the presence of experimental noise. A small-scale Monte-Carlo study was performed here to provide information on how the stiffness estimates varied with noise.

Update Parameter Specification

To obtain a meaningful update of the analytical model, strict control must be exercised over which parameters were varied. For example, the beam thickness, material stiffness, material density, gravity load, top and bottom translational and rotatory stiffnesses could all be changed to update the model, minimizing any error in the model when compared to experimental results. However, the effects of some of these parameters are highly correlated with the effects of other parameters. For example, thickening the beam and increasing the beam material modulus both increase natural frequencies. If both of these parameters are included in a parameter update scheme, the estimates obtained are unlikely to match the true values. This happens because of the collinearity of the effects, i.e., the increase in natural frequency caused by thickening the beam can be counteracted by decreasing the material modulus.

This problem often occurs in multiple linear regression. One unfortunate symptom of including too many parameters with correlated effects is that the resulting model is unlikely to match data taken at new points, i.e., the updated beam model will not match data taken at other frequencies or scan points on the beam.

A better approach is to utilize all information known about the beam and to keep the number of optimization parameters to a minimal but usable number of target parameters. Only parameters with significant amounts of uncertainty should be updated. For the beam suspended with shock cords, the only part of the system that was not known with some confidence was the suspension. Therefore, updates were only performed on equivalent transverse and rotatory stiffnesses representing the suspension. There are not actually two springs attached to the top of the beam, but it

is reasonable to expect that the effects caused by the shock cord can effectively be modeled with these two stiffnesses.

Chapter 4 contains an analysis of the shock cord and its expected stiffness effect. The cord is not expected to contribute to any rotatory stiffness effect, but the "stinger" used to attach the electromagnetic shaker to the top of the beam will have a rotatory stiffness effect.

Least-Squares Comparison Methodology

In order to reconcile the analytical model with experimental data, theoretical results from the analytical model must be compared to results from the experiment. For this thesis, comparisons of mode shapes were selected as the basis for quantifying differences between the analytical model and the experimental results. Unlike natural frequencies, the mode shapes of a beam are not functions of material modulus or beam thickness. Therefore, inaccurate estimates of these values will not affect the study of mode shapes. This property does not hold for a study of natural frequencies.

With this comparison technique, each of the mode shapes was normalized such that the root-mean-square (RMS) displacement was unity, as shown in Eq. 34. The shape of the first mode from the test was then subtracted from the shape of the first mode of the model, and a sum-of-squares total was taken over the entire structure. The sum-of-squares (SS) value given in Eq. 35 is a measurement of spatial error in the mode shapes and was the quantity minimized using the optimization methodology presented in the section below. Another useful quantity that can be derived using the sum-of-squares error value given in Eq. 35 is an estimated signal-to-noise ratio, given

in Eq. 36. This simple relationship can be derived using the mode shape normalization that is performed in Eq. 34. The normalization very nearly makes the RMS value of the signal unity.

$$RMS_{orig} = \sqrt{\left(\sum_{i=1}^{n} y_{old,i}^2\right)/n}, \quad y_i = \frac{y_{old,i}}{RMS_{orig}} \quad \rightarrow \quad RMS_{disp} = \sqrt{\left(\sum_{i=1}^{n} y_i^2\right)/n} = 1.0 \tag{34}$$

$$SS_{err} = \sum_{i=1}^{n} \left(y_{i,\,meas} - y_{i,\,model}\right)^2 \tag{35}$$

$$S/N \equiv \frac{RMS_{signal}}{RMS_{noise}} \approx \frac{1.0}{\sqrt{SS_{err}/n}} \tag{36}$$

Parameter Updates via Optimization

Update Methodology

A non-linear least-squares approach was used to update the stiffnesses in the model such that the analytical mode shapes matched the mode shapes obtained from experiment as closely as possible. Specifically, an optimization routine was used to update the stiffness parameters to achieve a minimum least-squares difference between the analytical mode shapes and the experimental mode shapes. Equation 37 gives the non-linear programming formulation used to update the analytical model with a single mode shape, while Eq. 38 shows the formulation used to update the model with several mode shapes simultaneously. The analytical mode shape, y_{model}, comes from the Rayleigh-Ritz beam model, described in Chapter 3, which incorporates the

transverse and rotatory stiffness parameters, while the experimental mode shape, y_{meas}, comes from the experimental procedure described in Chapter 6.

$$\min_{K_{t,0},K_{r,0}} \sum_{i=1}^{n} \left(y_{meas,i} - y_{model,i}\right)^2 \quad \text{such that } K_{t,0}, K_{r,0} \geq 0 \tag{37}$$

$$\min_{K_{t,0},K_{r,0}} \sum_{r=1}^{m} \sum_{i=1}^{n} \left(y_{meas,r,i} - y_{model,r,i}\right)^2 \quad \text{such that } K_{t,0}, K_{r,0} \geq 0 \tag{38}$$

Optimization Techniques Used

To solve the minimization problem of Eqs. 37 and 38, two types of optimization algorithms were considered. Unconstrained optimization algorithms are very effective at finding function minimums but are poorly suited for imposing constraints on the problem. However, the requirement for non-negative stiffness estimates was a constraint that could not be ignored. Therefore, a constrained optimization algorithm was considered as well.

Although only two optimization techniques were used for the final analysis, a total of four different techniques were used in the course of the study. The first three methods discussed are techniques for unconstrained optimization, in which the parameters are freely adjusted to minimize an objective (or error) function. The fourth is a constrained optimization technique in which the parameters have to satisfy constraint equations as well. The four techniques are summarized below in Table 21.

Table 21: Optimization Techniques

1. The Levenberg-Marquardt method [2] is a technique which uses direct estimates of first and second derivative information to adjust the optimization parameters. These derivatives are usually estimated using the finite difference method for all but the simplest problems. Each step taken by this technique represents a weighted mean of the Gauss-Newton method, which is very fast but often unstable, and the steepest descent method, which is very slow but extremely stable. The net result is a fairly fast algorithm which is quite stable, although the finite difference calculations can be time consuming.

2. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [3] is a particular implementation of a class of algorithms known as quasi-Newton methods. These methods use first-derivative information to update an estimate of the second derivative as each step is taken. Again, these derivatives are usually estimated using the finite difference. The second derivative estimate is updated as the algorithm proceeds. It is not as accurate as the estimate used in the Levenberg-Marquardt method, but it is much less expensive to calculate. This algorithm is also fast and fairly stable.

3. The Nelder-Mead simplex method [4] takes a different approach to changing the design parameters. Instead of working with estimates of derivatives, this method manipulates a minimal-sized triangle in the parameter space. The triangle is expanded, contracted, and folded until it converges on the minimum objection function evaluation point. The simplex method is slower than the two derivative methods discussed above, but is very stable, dealing well with discontinuities in slope.

4. Sequential-quadratic programming [5] utilizes a fundamentally different approach than the three unconstrained methods. At each step, the objective function is approximated as a quadratic function, while the constraints are approximated as linear functions. This approximate constrained problem is solved, and the process is repeated from the solution design point. Often the BFGS method of updating second derivative information is used to form the quadratic function.

Ideally, sequential-quadratic programming (SQP) could have been used alone to solve the constrained minimization problem of Eqs. 37 and 38. Unfortunately, the SQP algorithm converged poorly when neither of the non-negativity constrains were active. Of course, when neither constraint was active, the problem effectively became an unconstrained optimization problem, for which the Levenberg-Marquardt, BFGS, and simplex methods could be used.

The Levenberg-Marquardt and BFGS methods were found to be unsuitable in this case as well, as they required extremely tight convergence parameters to prevent premature termination of the algorithm. Additionally, in some of the cases the line search used in the Matlab Optimization Toolbox [26] implementation went into an endless loop, locking up the optimization algorithm entirely. The Nelder-Mead simplex method was much more robust, always converging to the exact location of the function minimum without requiring excessively tight convergence parameters.

The final algorithm consisted of using the simplex algorithm as a first attempt at solving both the translational and rotatory stiffnesses at the top of the beam. If negative stiffness estimates resulted from the use of this algorithm, sequential-quadratic programming was used to impose constraints that kept the stiffnesses positive. The combination of these two optimization algorithms was effectively used to solve the optimization problem.

Later in the study, the optimization was restricted to solving for the optimum translational stiffness, but not the rotational stiffness. The rotational stiffness proved to be quite sensitive to noise in the experimental mode shapes, as discussed later in this chapter, and in some cases it was decided that simply using a good estimate of

the rotatory stiffness was a better approach than allowing the optimization routine to generate meaningless results. MATLAB code for the model reconciliation process is given in Appendix C.

Noise Sensitivity Study

Monte-Carlo Methodology

Before the reconciliation was performed with actual experimental data, a study was performed to determine how sensitive the reconciliation process was to the presence of noise in the data. In order to accomplish this, the analytical model was used to simulate "experimental" mode shapes for a beam with known suspension stiffnesses, as listed in Table 22. The "known" values used in the simulation were fixed at reasonable estimates of the translational and rotational stiffness values. The stiffness estimates were added to the Rayleigh-Ritz model of Chapter 3, which was then used to generate mode shapes. All of the modes shapes were normalized to an RMS amplitude of unity. The beam parameters from Table 8 of Chapter 3 were used to complete the specification of the analytical model.

Table 22: Noise Study Beam Solution

1. translational stiffness $K_{t,0}$ = 3.0 lbf/in
2. rotatory stiffness $K_{r,0}$ = 12.0 lbf·in/rad
3. translational mass, rotatory inertia, and gravity = zero

The first five flexural modes were generated for this "beam," and four levels of noise were added to the mode shapes. Each case was repeated three times to provide

some indication of solution variability caused by the noise. A proper Monte-Carlo would have repeated each case several hundred times, but performing model reconciliation on so many mode shapes would have required months of computer time.

Table 23 lists the four levels of normal noise added to each mode shape along with estimates of the corresponding signal-to-noise ratio, which was defined in Eq. 35. The number of points used in the noise study was the same as the number of points taken experimentally going down the beam, as is described in Chapter 6. Figure 31 shows an example of the medium noise level.

Table 23: Noise Levels

1. Zero: normalized shape alone
2. Low: normalized shape + 0.01 N[0,1], $S/N = 100$
3. Medium: normalized shape + 0.07 N[0,1], $S/N = 14.3$
4. High: normalized shape + 0.50 N[0,1], $S/N = 2.0$

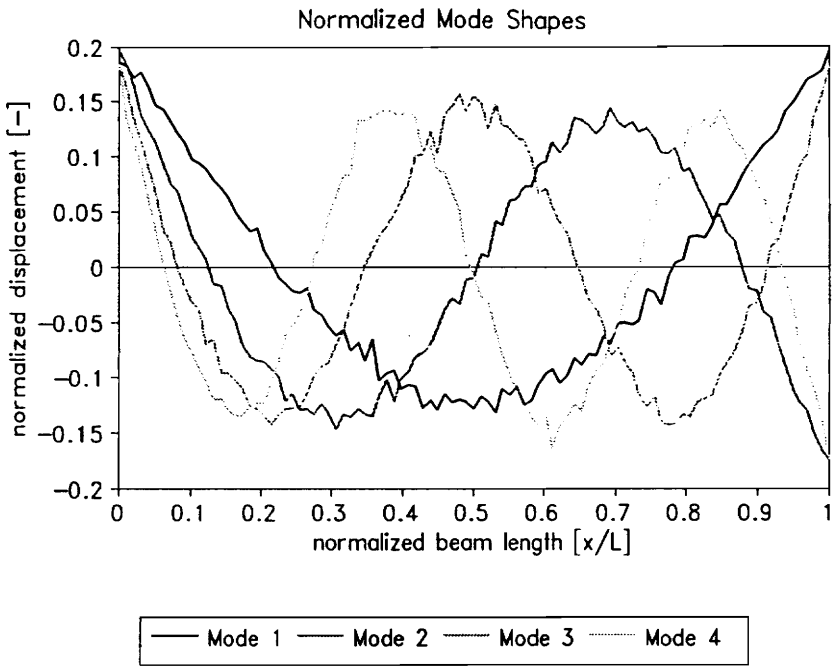

Figure 31: Medium Noise Example

Limited Monte-Carlo Results

For the first set of trials, the model was reconciled with each of the "experimental" mode shapes individually. This was done to determine if estimates based on higher modes were more or less sensitive to estimates based on mode shapes from lower modes. These results are listed in Table 24. Table 25 gives the results for model reconciliation when all five shapes were used simultaneously. It should be noted that mean and variance statistics used to describe the results are sometimes biased by the constrained nature of the optimization algorithm. The statistical distribution of the stiffness estimates will not be normal.

Table 24: Individual Noise Fits

noise/ mode	Trial 1		Trial 2		Trial 3		mean		variance	
	trans	rot	trans	rot	trans	rot	trans	rot	trans	rot
Zero/1	3.00	12.0	3.00	12.0	3.00	12.0	3.00	12.0	0.00	0.00
2	3.00	12.0	3.00	12.0	3.00	12.0	3.00	12.0	0.00	0.00
3	3.00	12.0	3.00	12.0	3.00	12.0	3.00	12.0	0.00	0.00
4	3.00	12.0	3.00	12.0	3.00	12.0	3.00	12.0	0.00	0.00
5	3.00	12.0	3.00	12.0	3.00	12.0	3.00	12.0	0.00	0.00
Low/1	2.99	29.9	2.87	4.66	3.5	18.3	3.12	17.6	0.33	12.6
2	2.95	22.7	3.37	24.1	3.69	0	3.33	15.6	0.37	13.5
3	3.86	29.8	3.68	5.92	3.57	0	3.70	11.9	0.15	15.8
4	5.48	13.8	2.09	19.6	0	0	2.52	11.1	2.76	10.1
5	8.29	11.3	8.49	0	0.67	0	5.82	3.77	4.46	6.52
Med/1	2.13	14.0	1.27	459	2.55	0	1.98	158	0.65	261
2	5.47	82.2	7.52	0	10.6	0	7.85	27.4	2.56	47.4
3	4.60	29.2	0	333	11.2	0	5.28	121	5.65	184
4	0.45	158	0	176	58.8	54.9	19.8	130	33.8	65.4
5	0	0	86.5	0	30.5	0	28.8	0.00	49.9	0.00
High/1	0	0	0	519	0	175	0.00	231	0.00	264
2	0	299	0	0	0	137	0.00	556	0.00	719
3	0	0	0	0	114	312	38.0	104	65.8	180
4	171.2	396	229	0	73.9	0	158	132	78.4	229
5	0	516	0	0	0	1051	0.00	522	0.00	526
Actual	3.00	12.0	3.00	12.0	3.00	12.0	--	--	--	--

Clearly, the process is quite sensitive to noise, particularly for the rotatory stiffness estimates, where the estimate variances were substantially larger than the stiffness value itself. The estimates for the simultaneous mode reconciliation of Table 25 were not as sensitive, but still shows the significant solution variability caused by the presence of noise.

Table 25: Simultaneous Noise Fits

noise	Trial 1		Trial 2		Trial 3		mean		variance	
	trans	rot	trans	rot	trans	rot	trans	rot	trans	rot
Zero	3.00	12.0	3.00	12.0	3.00	12.0	3.00	12.0	0.00	0.00
Low	3.05	9.58	2.89	8.75	3.09	7.75	3.01	8.69	0.11	0.92
Med	2.62	0	2.52	0	2.73	79.7	2.62	26.6	0.11	46.0
High	0	0	0	251	0	3.79	0.00	84.8	0.00	144
Actual	3.00	12.0	3.00	12.0	3.00	12.0	--	--	--	--

Results with Translational Stiffness Only

Early results in the study showed that the rotational stiffness estimates were extremely sensitive to noise. Therefore, as a second approach, the known rotational stiffness was always used instead of extracting a stiffness estimate using the optimization algorithm. Table 26 contains the results when mode shapes were fitted individually. The results closely match those of Table 24, which further indicates that the effects caused by the presence of rotatory stiffness are exceeded by the effects of noise in the data.

Table 26: Individual Noise Fits (Trans Only)

noise/ mode	Trial 1	Trial 2	Trial 3	mean	variance
	trans	trans	trans	trans	trans
Zero/1	3.00	3.00	3.00	3.00	0.00
2	3.00	3.00	3.00	3.00	0.00
3	3.00	3.00	3.00	3.00	0.00
4	3.00	3.00	3.00	3.00	0.00
5	3.00	3.00	3.00	3.00	0.00
Low/1	3.03	2.86	3.06	2.98	0.11
2	3.07	3.52	3.59	3.39	0.28
3	4.06	3.59	3.38	3.68	0.35
4	6.19	3.36	0	3.18	3.10
5	12.0	9.13	1.32	7.48	5.53
Med/1	2.43	2.32	2.52	2.42	0.10
2	6.07	7.42	10.5	8.00	2.27
3	4.82	0	11.0	5.27	5.51
4	1.33	0	59.0	20.1	33.7
5	0	87.2	31.1	39.4	44.2
High/1	0	0	0	0.00	0.00
2	12.3	0	0	4.10	7.10
3	0	0	118	39.3	68.1
4	171	229	73.8	158	78.4
5	0	0	0	0.00	0.00
Actual	3.00	3.00	3.00	--	--

When all of the first five modes were fitted simultaneously for the translational stiffness only, the results of Table 27 were obtained. Again, these results were very similar to those of Table 25.

Table 27: Simultaneous Noise Fits (Trans Only)

noise	Trial 1	Trial 2	Trial 3	mean	variance
	trans	trans	trans	trans	trans
Zero	3.00	3.00	3.00	3.00	0.00
Low	3.04	2.89	3.08	3.00	0.10
Med	2.58	2.48	2.92	2.66	0.23
High	0	0	0	0.00	0.00
Actual	3.00	3.00	3.00	--	--

When all of the results of the noise study are examined carefully, certain "rules-of-thumb" become evident. Table 28 lists these observations.

Table 28: Effects of Noise

1. Estimates of translational stiffness are somewhat sensitive to noise.

2. Estimates of rotatory stiffness are very sensitive to noise.

3. Estimates of stiffnesses based on higher modes are more sensitive to noise that those based on lower modes.

4. Using several mode shapes simultaneously improves the quality of the estimate. In fact, model reconciliation should always be done with all mode shapes, as the updated model should successfully describe <u>all</u> of the mode shapes at once.

5. Noise of a sufficient magnitude can render stiffness estimates meaningless. This was particularly evident with rotatory stiffness estimates, which required low noise levels with simultaneous mode shape fits to obtain reasonable results.

Chapter 6: Experimental Techniques for Obtaining Mode Shapes

Overview of Mode Shape Process

The entire purpose of developing an analytical model of a structure is to predict the dynamic behavior of an actual physical object. This model can then be used for the intended function of structural modification and/or optimization. The quasi-comparison function based Rayleigh-Ritz model was developed to model beams, and the reconciliation process of Chapter 5 is based on comparisons of mode shapes. Therefore, experimental mode shapes had to be acquired from an actual beam to perform system identification on the model. To accomplish this task, several steps had to be completed. These steps are listed in Table 29.

Table 29: Mode Shape Acquisition Process

1. Determine the lab set-up that can best be used to acquire experimental data. For this step, a stainless steel beam was suspended such that the effects of the shock cord suspension were emphasized.

2. Determine natural frequencies for the structure. This step is necessary if mode shapes are to be acquired. This was done by sinusoidally exciting the beam at the top and measuring its response at the driving point with an accelerometer and a phase meter (implemented in LabVIEW 2.21 [30] using Eqs. 39 through 41, as presented later in the chapter).

3. Acquire operating shape information at frequencies near the natural frequencies. This step is also necessary if mode shapes are to be acquired. This was done by sinusoidally

exciting the beam at the top and measuring the response of the entire structure with a scanning laser Doppler vibrometer.

4. Make corrections in the operating shape data to account for any systematic errors. The two sources of error in the operating shape data were velocity reductions caused by the angle of the laser beam angle with respect to the beam and velocity signal dropouts caused by surface roughness.

5. Extract mode shape information from operating shape data. This step was accomplished by utilizing the well-separated modes of the beam with knowledge of how modes behave when excited at near-resonance frequencies.

Experimental Setup

In normal modal analysis tests, structures are suspended using long lengths of shock cord. This is done to approximate free-free boundary conditions. An often used rule-of-thumb states that the natural frequency of the suspension should be a tenth or less of the natural frequency of the lowest flexural mode of interest. The stainless steel beam, with a first mode natural frequency of approximately 17.5 Hz, would require, by rule-of-thumb a suspension with frequencies of less than 1.75 Hz. If the suspension frequency is higher, the results of the test may be affected by beam/ suspension coupling.

In this particular set-up, however, the lengths of the shock cord were deliberately kept short in order to emphasize any effect they may have on the vibration of the beam. Figures 32 and 33 shows the configuration used to gather data from the beam. A clamp was used to firmly attach the shock cord to the support beam.

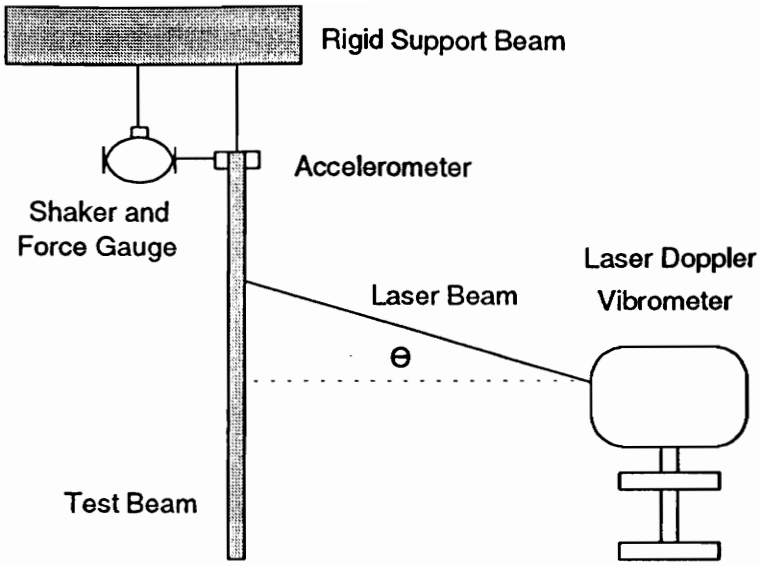

Figure 32: Beam Test Set-up (overview)

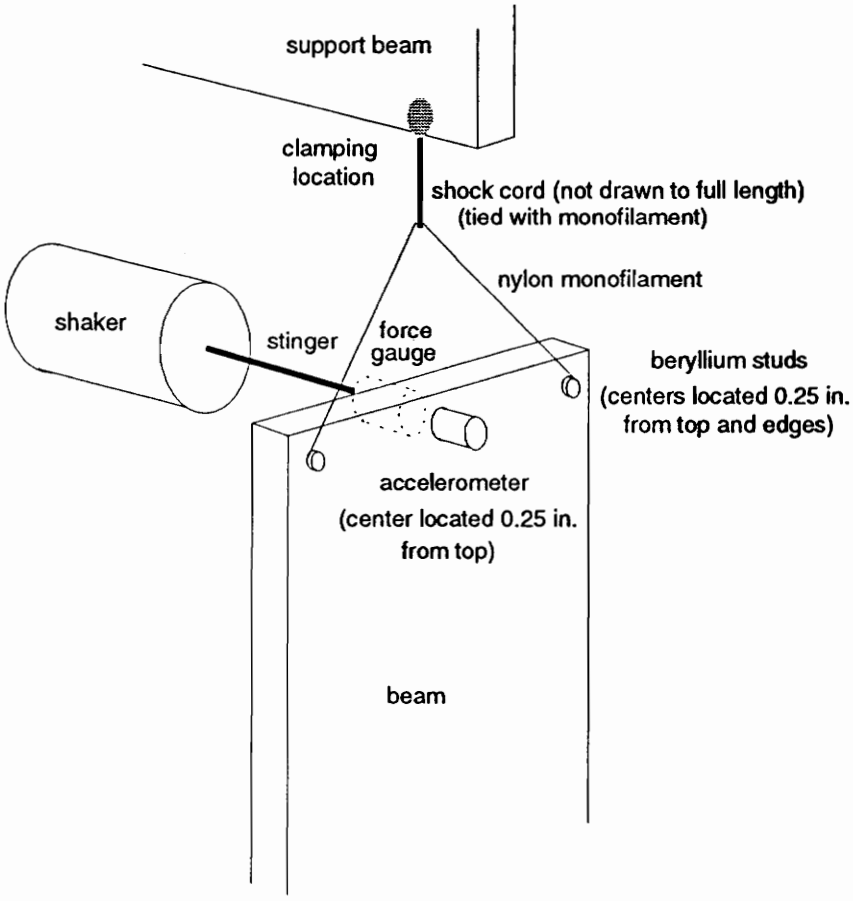

Figure 33: Beam Test Set-up (detail)

Natural Frequencies with Accelerometers

The perturbation studies of Chapter 3 indicated that the lower vibrational modes of the beam would be affected most by an added translational stiffness. Therefore, the first five flexural modes were studied in the experiment. Figure 21 of the perturbation study of Chapter 3 indicates that the first mode was expected to be affected the most, while the fifth mode was expected to be affected the least.

Before operating shapes could be recorded using the laser, the natural frequencies of the first five flexural modes were determined using an accelerometer attached at the driving point. Sine-dwell testing was used with a LabVIEW 2.21-based phase meter to find the frequency at which the phase angle was as close to 90° as possible. For a lightly damped structure with well separated modes, such as the beam, this frequency is very close to the natural frequency, as indicated by modal theory. This frequency search was done for the first five modes for each boundary condition case. In the case of a structure with heavy damping or nearly coupled modes, modal decomposition must first be done in the frequency domain before natural frequencies can be determined.

Operating Shapes via Scanning Laser-Doppler Vibrometer

Raw Data Acquisition

Once the natural frequencies had been determined with the accelerometer, five operating shapes were taken for each mode with the scanning laser-Doppler vibrometer. The sine-dwell frequencies were perturbed from the estimate of the natural frequency by -2%, -1%, 0, +1%, and +2%. The operating shape at the natural

frequency most closely matches the mode shape, but the shapes at the other frequencies were taken since the force signal is nearly zero at the natural frequency, which can allow noise in the force signal to dominate the results. Taking data at other frequencies resulted in a stronger force signal, which would be less affected by a fixed amount of noise. These other shapes were also taken in case some frequency dependent effect was present. With 9 boundary conditions cases, 5 modes per case, and 5 operating shapes per mode, a total of 225 tests were recorded with the laser.

Each operating shape was recorded using sine-dwell testing. A Macintosh IIfx equipped with National Instruments data acquisition boards and running LabVIEW 2.21 [30] was used to aim the laser and to acquire velocity and force data from the beam. Data was acquired on a 129 by 17 spatial grid on the beam, with 256 time samples being taken of the force and velocity signals at each point. The sampling rate was as close to eight times the driving frequency as possible.

Data Preprocessing

The LabVIEW program performed a significant amount of data processing at each point, using multiple linear regression to fit a sine wave, cosine wave, and DC offset to the data, as specified in Eq. 39. The results of this regression provided a magnitude and phase angle for each of the signals. This information was then combined to obtain the relative phase angle between the two signals, as given in Eq. 40. This relationship enables the force and velocity signals of Eq. 39 to be written in a simpler form, as shown in Eq. 41.

$$\text{least-squares fit} \quad \begin{aligned} f(t) &= \alpha_0 + \alpha_1 \sin(\omega t) + \alpha_2 \cos(\omega t) \\ v(t) &= \beta_0 + \beta_1 \sin(\omega t) + \beta_2 \cos(\omega t) \end{aligned} \tag{39}$$

$$\theta_0 = \arctan(\alpha_2, \alpha_1) - \arctan(\beta_2, \beta_1) \tag{40}$$

$$\begin{aligned} f(t) &= \cos(\omega t) \\ v(t) &= \cos(\omega t + \theta_0) \end{aligned} \tag{41}$$

The force amplitude, velocity amplitude, and relative phase angle for each point were recorded in ASCII files on the Macintosh. These files were then converted to MATLAB MAT files and transferred to an HP 380 workstation and then to a 80386-based PC. From the PC further data processing was performed, as is described in the sections below.

Correction of Systematic Errors

Correction for Angles

In order to compare the experimental data to the quasi-comparison function Rayleigh-Ritz model, the data had to be corrected for any systematic errors that occurred during the test. One major source of error was the deflection angle of the laser beam during the test. The motion of the beam during vibration is believed to be almost entirely out-of-plane. The laser, however, does not normally hit the beam at right angles. It instead hits the beam at some angle, as shown in Figure 32.

Since the laser-Doppler vibrometer can only record velocities along the line of the laser beam, it will not measure the full velocity of the beam when it is not perpendicular. Instead, a smaller component of the true velocity is measured. To

compensate for this, the recorded velocity signal must be corrected with a simple trigonometric calculation, as given in Eq. 42.

$$\dot{w}_{adjust} = \frac{\dot{w}_{meas}}{\cos\theta_{up-down}\cos\theta_{left-right}} \tag{42}$$

However, since the left-right angle deflection is very small, its resulting cosine term can effectively be treated as unity, yielding Eq. 43. Making this assumption will cause at most a 0.01% error in the velocity measurements.

$$\dot{w}_{adjust} = \frac{\dot{w}_{meas}}{\cos\theta_{up-down}} \tag{43}$$

Unfortunately, the LabVIEW program used to gather the data was not set up to record the angle voltages sent to the laser-Doppler vibrometer. The vertical deflection angles were, therefore, unavailable for use in angle correction. However, scaled x and z coordinates for each point on the beam were recorded. Since the laser angle between adjacent points was a constant and the length of the beam was known, the laser angles and working distance could be backed out using an optimization routine.

Specifically, the x points on the beam follow a tangent curve, as shown in Figure 34. The straight tangent lines were placed on the figure to help emphasize how a slight amount of curve is present in the x-values.

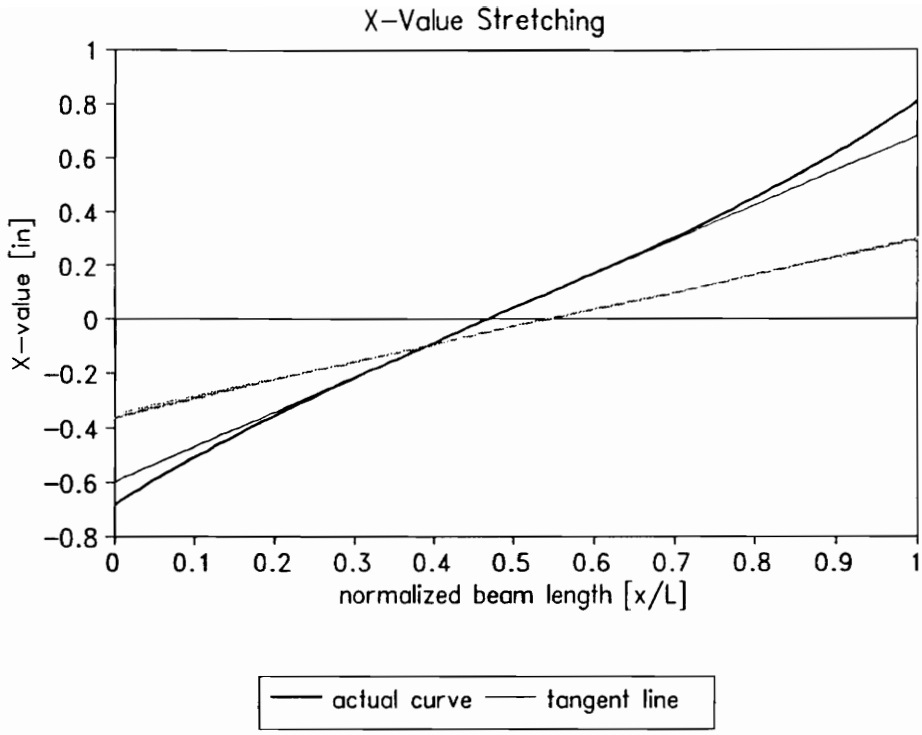

Figure 34: X-Values on Beam

Estimates for the top angle, bottom angle, and distance were used to calculate a similar curve, as shown in Eq. 44. A Gauss-Newton based optimization routine found in the MATLAB Optimization Toolbox [26] was used to update these estimates such that the least-squares difference between the angles and distance-based x values and the recorded x values was minimized, as given in Eq. 45. The updated estimated were then used to calculate the angles at each point and to correct for the angle errors. Appendix C contains a listing of the code used to perform this task.

$$\hat{x}_{est,i} = d\cos\left(\theta_{\min} + \frac{i(\theta_{\max} - \theta_{\min})}{n_x}\right) \qquad x_{est,i} = \hat{x}_{est,i} - \hat{x}_{est,1} \tag{44}$$

$$\min_{\theta_{\min},\theta_{\max},d} \sum_{i=1}^{n_x} \left(x_{est,i} - x_{rec,i}\right)^2 \tag{45}$$

Removal of Dropouts

In addition to angle effects, a second source of error in the experimental data was the presence of drop-outs. A drop-out occurs when the laser beam strikes an optically rough point on the surface of the beam. The strength of the returning Doppler signal is significantly weakened, and the demodulator in the laser-Doppler vibrometer system is unable to interpret the signal from the laser and outputs a zero velocity signal instead. This source of noise occurs sporadically and is clearly biased towards zero. This bias prohibits the use of standard statistical tools for dealing with noise. The best method of dealing with this problem is to simply remove data in which a drop-out has occurred. Unfortunately, the regression that was done within the LabVIEW program before the data reached storage made it difficult to determine whether or not a drop-out had occurred.

The velocities at points on any given line going across the beam horizontally are expected to be the same, since the beam had a very high aspect ratio and behaved much like the infinitely thin Euler-Bernoulli mathematical beam. Therefore, an asymmetric-trimmed-mean [31] was applied to each row of data. Upon completion of this operation, a single column vector remained which contained an estimate of the operating shape at the excitation frequency. Because of the biased nature of the noise present in the data, this asymmetric-trimmed-mean estimate is expected to be more accurate than an estimate based on a simple mean.

Figure 35 shows what a typical row of data looked like on the beam. To calculate the asymmetric-trimmed-mean, the data was sorted in numerical order. Points containing the smallest and largest response values were thrown out. Because of the biased nature of drop-out noise, more small-valued points were thrown out than large-valued points. The final statistic computed was an average of the remaining points. Figure 36 shows how a drop-out, originally at point 15, is eliminated from the data set, along with several other points. An average not including the drop-out data point will be more accurate than a mean that contains the drop-out. Appendix C contains the MATLAB code used to perform this task.

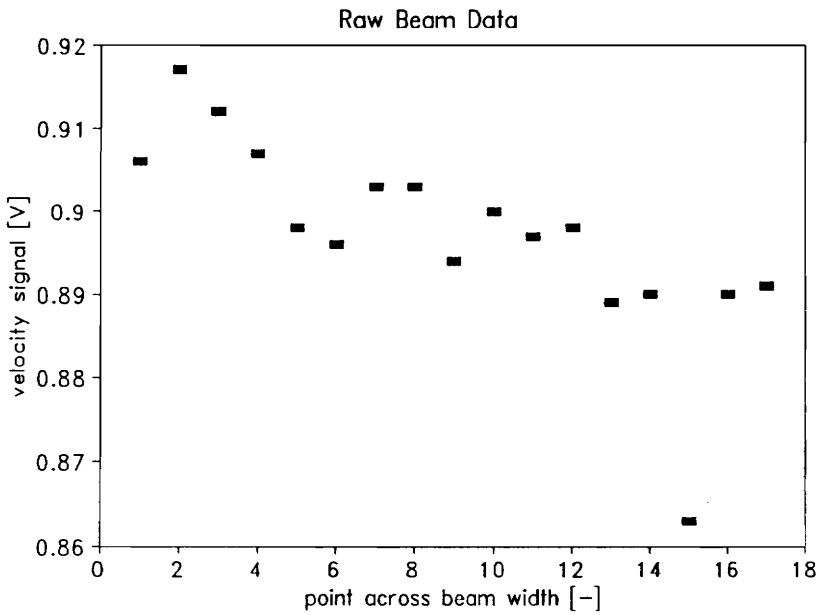

Figure 35: Aymmetric-Trimmed-Mean (prior to sort)

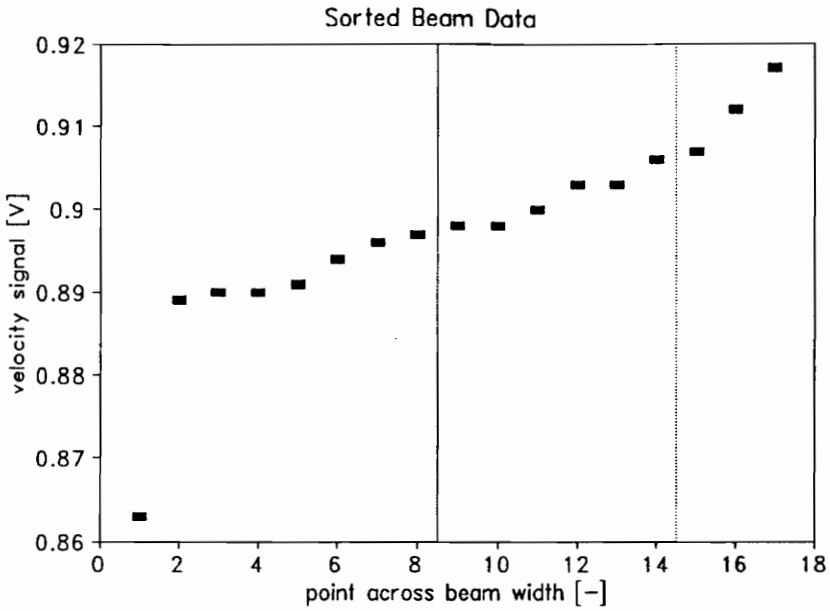

Figure 36: Asymmetric-Trimmed-Mean (sorted)

Mode Shape Extraction

A final step that was necessary in preparing the data was to extract mode shapes from the operating shape data taken from the beam. If a structure has well separated modes and is lightly damped, the operating shape at the natural frequency will closely match the mode shape, as the modal participation factor of the target mode is much higher than that of other modes. However, the excitation frequency must match the natural frequency exactly if this is to yield the mode shape. Figures 37 and 38 show how the operating shapes vary with one and two percent variations in the excitation frequency. This happens with both experimental data, taken as described earlier in this chapter, and with analytical operating shapes built from mode shapes created with the analytical model of Chapter 3. The tip deflections of the $\pm 2\%$ frequency shifts varied by more than $\pm 5\%$ from the tip deflection of the central excitation frequency.

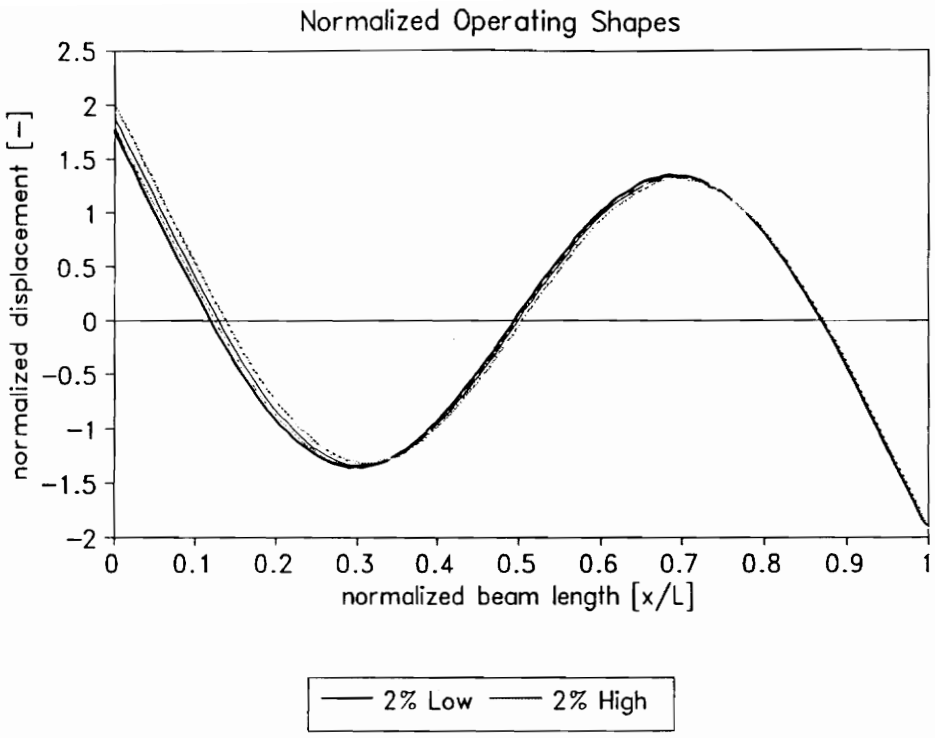

Figure 37: Frequency Error Effects (Experimental)

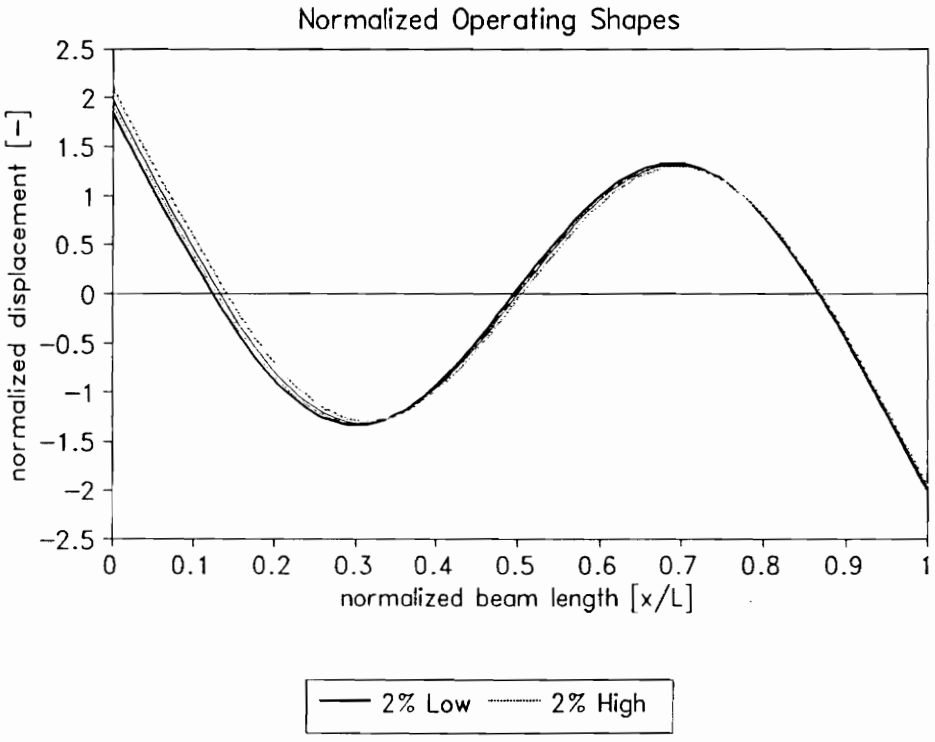

Figure 38: Frequency Error Effects (Analytical)

In order to obtain an estimate of the mode shape that was not sensitive to the excitation frequency, phase angle information derived using Eq. 40 was used. Near the natural frequency, the mode of interest is excited with a phase angle near 0°, depending on how close the excitation frequency is to the natural frequency. The other modes are excited as well, but at phase angles that are very close to -90° or 90° instead. If viewed in terms of complex numbers, the contribution to the operating shape of the mode of interest is the only one containing a real velocity component. All of the other mode shapes are excited such that their response is almost purely imaginary. Thus, the real part of the operating shape (represented by velocity) contains the desired mode shape, while the imaginary part contains a combination of several mode shapes. However, if the excitation frequency is too far from the natural frequency, all of the modes will be excited at $\pm 90°$ and no meaningful real part will be present. Figure 39 shows the real parts of the operating shapes at -2%, -1%, 0, and +1%. None of the frequency dependent variation is evident. The level of noise on the mode shape estimates, however, was significantly increased.

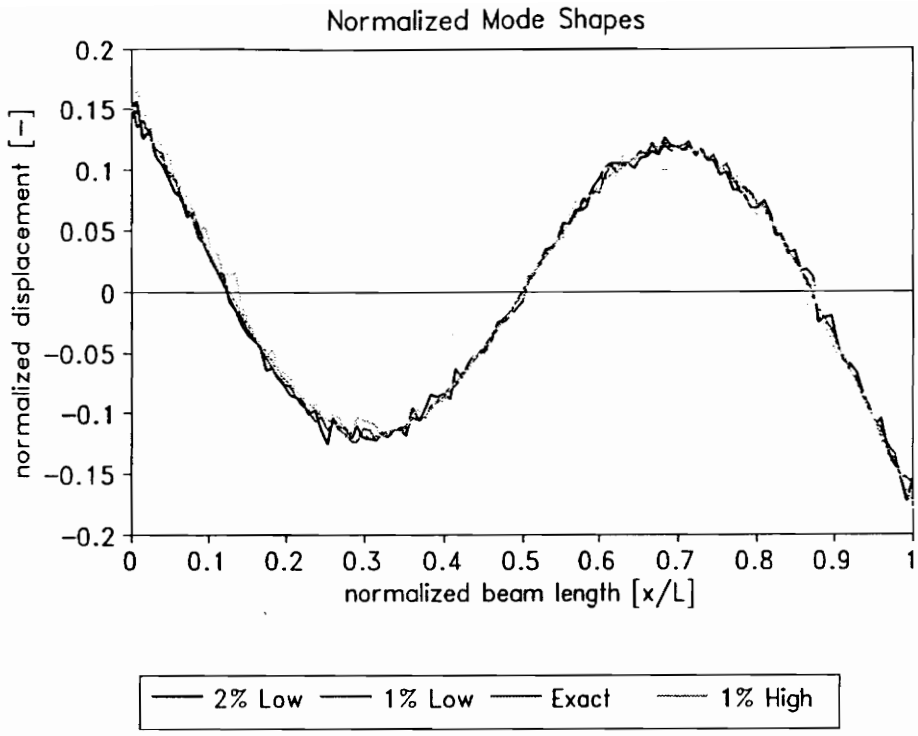

Figure 39: Real Parts of Operating Shape

For each mode, an average of the mode shapes from the middle three excitation frequencies, -1%, 0, and +1%, was used as an estimate of the mode shape. The results of this data processing served as the basis for use of model reconciliation in Chapter 7.

Chapter 7: Model Reconciliation with Experimental Data

Overview of Model Reconciliation

This chapter contains the heart of the thesis. Here, the various elements of the thesis are brought together into a cohesive package. Table 30 below provides a complete list of steps needed to perform the work of this thesis.

Table 30: Complete Steps for Model Reconciliation

1. Develop an analytical model capable of modeling the structure and its suspension. The Rayleigh-Ritz model developed in Chapter 3 is the product of this objective.

2. Determine a method for updating the model using experimental data obtained from dynamic tests. Chapter 5 discusses the optimization-based reconciliation technique developed for this task.

3. Determine how to acquire the experimental data needed to perform the model reconciliation. Chapter 6 discusses the experimental techniques used to obtain high spatial resolution operating shape data and the analysis techniques required to reduce it to mode shape data.

4. Perform any pre-analysis calculations needed for model reconciliation. These calculations included determining the beam parameters needed to complete the Rayleigh-Ritz model, determining the masses of the transducers used to acquire data, and determining any stiffness effects not caused by the suspension.

5. Set up a designed experiment to analyze effects caused by variations in the suspension and determine independently likely results. A small two-factor factorial experiment was

set up to study the effects of varying shock cord lengths and thicknesses.

6. Acquire the data, perform model reconciliation for each case, and compare to results from theory. The stiffnesses obtained from model reconciliation were compared to results from the shock cord theory developed in Chapter 4. The comparisons were done in light of the results from the noise sensitivity study of Chapter 6.

These six steps comprise the work of the thesis. Items 4 through 6 are described in detail in this chapter.

Pre-Analysis Calculations

Before the actual model reconciliation could be performed using data obtained experimentally, a number of parameters had to be specified to complete the Rayleigh-Ritz model. These parameters included beam dimensions, beam material properties, transducer mass estimates, and an estimate of the rotatory stiffness that comes from the stinger.

Beam Parameters

Table 31 gives the best estimates available for the specific parameters needed to accurately describe the beam. These model parameters were needed to complete the Rayleigh-Ritz beam model. Also listed are the means by which the estimates were obtained, where the instrument was measured to the nearest half-division.

Table 31: Beam Parameters

1. Length = 54.75 ± 0.03 in. (tape measure)
2. Thickness = 0.2605 ± 0.0005 in. (micrometer)

3. Width = 3.00 ± 0.03 in. (tape measure)

4. Density = 0.282 lbf/in^3 (table value for 304 stainless, an austenitic stainless steel) [32]

5. Elastic modulus = 28.0 x 10^6 (table value for 304 stainless)

Improved parameters for density and elastic modulus could be obtained with more accurate material identification and/or actual property testing performed on the beam itself. However, the beam exhibited the classic non-ferromagnetic behavior of an austenitic stainless steel and was identified by a 304 stamp located on its face.

Transducer Inertias

A brief experiment was run to determine the effective mass of the accelerometer and force gauge at the top of the beam (see Fig. 32). In the experiment, both transducers were directly attached to the shaker, and force and acceleration measurements were taken, as shown in Fig. 40. The simple relationship $F = ma$ provided the effective mass of the two transducers, including the attached wiring.

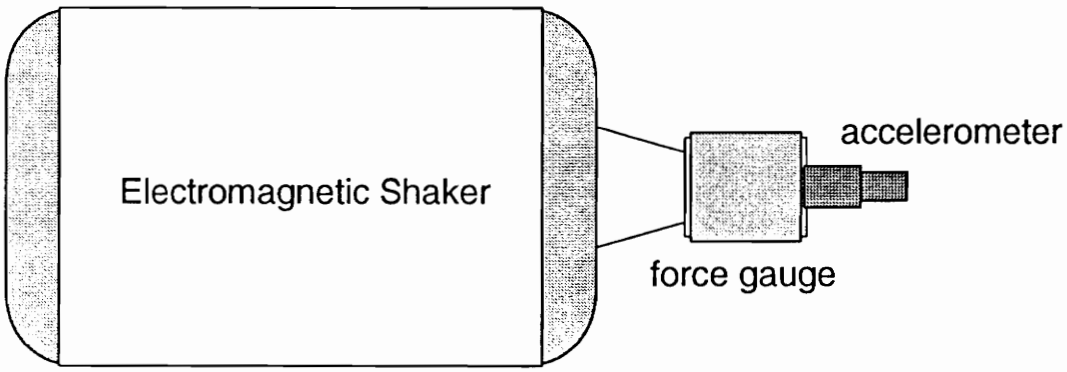

Figure 40: Effective Mass Experiment

To find the rotatory inertia effect, the *total* weights and dimensions of the accelerometer and force gauge were carefully measured. They were treated as cylinders displaced from the center of rotation, which was the neutral axis of the beam, for purposes of determining the mass moment of inertia. Table 32 gives the results of the mass experiment and the rotatory inertia calculations.

Table 32: Point Masses at Beam Tip

1. Translational mass: $m = 7.101 \times 10^{-5} \frac{\text{lbf} \cdot \text{sec}^2}{\text{in}} = 12.44 \text{ g}$.
2. Rotatory inertia: $I_m = 3.76 \times 10^{-5} \text{ lbf} \cdot \text{sec}^2 \cdot \text{in}$.

Stinger Rotatory Stiffness

In Chapter 4, the behavior of the elastic shock cord was carefully studied to determine its likely contribution to a transverse stiffness effect at the tip of the beam. Similarly, the "stinger" used to connect the electromagnetic shaker to the top of the beam was also carefully examined. This slender and flexible steel rod was carefully measured and used to calculate its likely contribution to a rotatory stiffness effect at the top of the beam. Table 33 contains the measured dimensions of the stinger along with the rotatory stiffness effects it creates when treated as a cantilever beam with a moment applied at the free end.

Table 33: Stinger Properties

1. Length = 1.25 ± 0.03 in. (tape measure)
2. Diameter = 0.0400 ± 0.0005 in. (micrometer)
3. Elastic Modulus = 30.0×10^6 (table value for steel) [33]
4. Rotatory stiffness = 2.9 lbf·in/rad

This estimate of rotatory stiffness was also used as a fixed rotatory stiffness value when the reconciliation algorithm was constrained to provide only transverse stiffness estimates.

Experimental Design

Once the pre-test calculations had been completed, all that remained was to determine the transverse stiffness effect of the shock cord used to suspend the beam. This, of course, is the main point of the research presented in this thesis.

Suspension with 3 Thicknesses at 3 Lengths

Normally, when an object is suspended with shock cord, the cord is stretched from the ceiling in order to maximize its length, hopefully minimizing any effect on the structure. For this project, however, the cord was deliberately kept shorter in order to emphasize any effect caused by the suspension, which would be manifested by increases in the effective translational and rotatory stiffnesses supporting the beam. In addition, the thickness of the cord might also have an effect on the behavior of the beam, although this was not predicted by the shock cord study of Chapter 4.

The two parameters of shock cord length and thickness were the main factors studied in the model reconciliation. They were used in a factorial experiment in which three different thicknesses of shock cord were used at three different fixed lengths to suspend the beam. To accommodate the effect of increasing the thickness, multiple lengths of shock cord were used. Table 34 shows the combinations tested.

Table 34: Suspension Cases Tested

Cases		thickness		
		1 cord	2 cords	3 cords
length	5 inches	5 modes	5 modes	5 modes
	10 inches	5 modes	5 modes	5 modes
	14 inches	5 modes	5 modes	5 modes

The result developed in Eq. 31 of Chapter 4 indicates that the effective stiffness of the shock cord should be independent of its axial stiffness. This assumption was tested by using three different "thicknesses" of cord. The first thickness of cord was a single strand of 0.25". The second thickness of cord was comprised of two strands of the same cord, and the third was comprised of three strands. Since the beam weighed 12.1 lbs, dividing its weight between two or three cords kept the cords in the linear region of the cord's load/deflection curve, as shown in Fig. 30 of Chapter 4. The single thickness was operating in the non-linear region. This violated the assumptions of Table 19, possibly invalidating the results of Eqs. 30 and 31. Each of these three cord types was tested at three different lengths.

Target Stiffness Estimates

The results of model reconciliation can be compared to the theoretical transverse stiffness based on the cord's length. Equation 31 from Chapter 4 yields the stiffnesses given in Table 35, although the stiffnesses for the single cord cases may vary since the cord strain is operating in the non-linear region. Estimates of the rotatory stiffness can be compared to stinger rotatory stiffness value of 2.9 lbf·in/rad from Table 33. The theoretical values may be invalid for the single cord case.

Table 35: Theoretical Transverse Stiffness

Length [in]	Stiffness [lbf/in]
5	2.4
10	1.2
14	0.86

Reconciliation of Model with Experimental Data

Chapter 5 described the process by which the data was gathered and processed to obtain the first five flexural mode shapes for each of the test cases of Table 36. Once this was accomplished, the mode shapes were then run through the reconciliation process of Chapter 6 to obtain estimates of the stiffnesses representing the suspension.

The experimental data was reconciled to a total of five different models, as listed in Table 36. The reasons for performing model reconciliation with each of these models is discussed in the results section for each model.

Table 36: Reconciliation Models

1. The first set of results was obtained using a model that included gravity-induced tension but did not include transducer mass. Both transverse and rotatory stiffness estimates were obtained.

2. The second set of results was obtained using a model that contained both gravity and transducer mass effects. Both transverse and rotatory stiffness estimates were obtained.

3. The third set of results was also obtained using a model that contained both gravity and transducer mass effects. Only transverse stiffnesses estimates were obtained.

4. The fourth set of results was obtained using a model that included transducer mass but did not include gravity-induced tension. Both transverse and rotatory stiffness estimates were obtained.

5. The fifth set of results was also obtained using a model that included transducer mass but did not include gravity-induced tension. Only transverse stiffness estimates were obtained.

Model Reconciliation Results

Case 1: Model with Gravity Only

The Rayleigh-Ritz model that included only gravity-induced tension was performed relatively early in the research. Listed in Appendix B, the stiffness estimates obtained from the model reconciliation process of Chapter 6 contained an unusually large number of zero stiffness values. This made little sense in light of the results of Chapter 4, in which the transverse stiffness effect of the suspending shock cord was found to be a *positive* value based on the cord length and beam weight.

Further study of the set-up revealed that the mass of the transducers needed to be included in the model. This was the motivation for performing the perturbation study of Chapter 3, which showed how the effect of a mass located at the top of the beam can counteract the effect of a transverse stiffness. If the mass is not included in the model, a reduced or even zero stiffness estimate can be obtained from the reconciliation process.

The results of reconciliation with this model also provided the motivation for performing the noise study described in Chapter 6. When the transverse stiffness results did not match those of Table 35, it was not apparent whether the discrepancies

were due to noise or due to other unmodeled factors. The noise study revealed that the variations seen in at least the rotatory stiffness effects were probably due to noise.

Case 2: Model with Gravity and Tip Masses

With the knowledge obtained from reconciliation of the model with gravity-induced tension only, the model was updated with the tip masses given in Table 32. Model reconciliation was first performed with the mode shapes individually to make an estimate of how much noise was present on each shape. These results are tabulated in Appendix B. Next, model reconciliation was performed with several mode shapes simultaneously, which should result in a better model as discussed in Chapter 6. Simultaneous fits will also alleviate the effects of noise, especially if only the mode shapes containing relatively low amounts of noise are included. This process was run twice, the first time including shapes with a S/N ratio greater than 8.0, and the second time including only shapes with a S/N ratio greater than 16.0. The individual S/N ratio estimates in Appendix B were used to determine whether or not a mode shape was included in the simultaneous fit. Tables 37 and 38 give the results for the two noise level cases.

Table 37: Simultaneous Fits (individual $S/N > 8.0$)

Length	# of Cords	trans [lbf/in]	rot [lbf·in/rad]	modes included
Long	1	5.17	65.7	2-5
	2	0	353	1-3
	3	21.6	0	2-5
Medium	1	0	40.7	1-5
	2	0	88.0	1-5
	3	0	564	1-5
Short	1	0	703	2-5
	2	2.03	354	1-5
	3	0	252	1-5

Table 38: Simultaneous Fits (individual $S/N > 16$)

Length	# of Cords	trans [lbf/in]	rot [lbf·in/rad]	modes included
Long	1	5.17	65.7	2-5
	2	0	748	2-3
	3	21.4	0	2-4
Medium	1	0	14.1	2-3
	2	0	0	1-2
	3	0	1060	2-4
Short	1	0	1090	2,4-5
	2	1.89	442	1
	3	0	252	1-5

Case 3: Model with Gravity and Tip Masses (Translational Only)

As with the noise study of Chapter 5, the presence of noise in the previous model case caused the estimates of the rotatory stiffness to vary wildly. To possibly obtain better estimates of the translational stiffness, the rotatory stiffness was held to the fixed value given in Table 33.

Fixing the rotatory stiffness seemed to improve the translational stiffness estimates, making more of them positive and bringing them closer to the expected value. However, there is no way to determine whether or not these values are actually more correct. Tables 39 and 40 contain results when mode shapes meeting certain noise criteria were used simultaneously to update the model.

Again, individual mode shape fits were used to determine which shapes should be included in the simultaneous analysis. Appendix B contains a listing of these results.

Table 39: Simultaneous Fits ($S/N > 8.0$, Trans Only)

Length	# of Cords	trans [lbf/in]	modes included
Long	1	5.99	2-5
	2	0.66	1-3
	3	21.6	2-5
Medium	1	0	1-5
	2	0	1-5
	3	0.52	1-5
Short	1	6.40	2-5
	2	3.02	1-5
	3	0.18	1-5

Table 40: Simultaneous Fits ($S/N > 16$, Trans Only)

Length	# of Cords	trans [lbf/in]	modes included
Long	1	5.99	2-5
	2	7.86	2-3
	3	21.4	2-4
Medium	1	0	2-3
	2	0	1-2
	3	9.77	2-4
Short	1	10.5	2,4-5
	2	2.94	1
	3	0.18	1-5

These results do not match the theoretical results of Table 35 very closely, but they are not grossly inaccurate when the level of noise in the data is considered. Specifying exactly how closely the results match the theoretical results is difficult. An expansion of the noise study of Chapter 5 might yield a relationship between noise levels on the mode shape estimates and the variance of the stiffness estimates. This would enable the theoretical stiffnesses to be statistically compared with the estimates obtained from the reconciliation process.

Case 4: Model with Tip Masses, no Gravity

In Chapter 3, the convergence of the gravity effect was examined, as was shown in Table 15 and Fig. 20. This study revealed that the gravity effect converged very slowly. Indeed, it did not converge adequately even when the maximum possible number of Ritz terms was used.

Because of this, cases were run in which gravity was left out. This represented a less complete model than before, but it was hoped that leaving out an incompletely

converged effect might reduce the variability in the results. Tables 41 and 42 contain the results of the model reconciliation performed with this model. Appendix B contains the results of the individual mode shape fits used to determine which mode shapes were included in the simultaneous analysis.

Table 41: Simultaneous Fits (individual $S/N > 8.0$, no Gravity)

Length	# of Cords	trans [lbf/in]	rot [lbf·in/rad]	modes included
Long	1	4.23	193	2-5
	2	0	303	1-3
	3	21.1	0	2-5
Medium	1	0	48.2	1-5
	2	0	95.5	1-5
	3	0	573	1-5
Short	1	0	81.6	2-5
	2	1.50	416	1-5
	3	0	260	1-5

Table 42: Simultaneous Fits (individual $S/N > 16$, no Gravity)

Length	# of Cords	trans [lbf/in]	rot [lbf·in/rad]	modes included
Long	1	4.23	193	2-5
	2	0	77.4	2-3
	3	20.8	0	2-4
Medium	1	0	39.5	2-3
	2	0	0	1-2
	3	0	109	2-5
Short	1	0	122	2,4-5
	2	1.70	34.6	1
	3	0	260	1-5

Removing gravity-induced tension from the model had two noticeable effects. The first was that the transverse stiffness values obtained were always smaller. This effect was anticipated based on the results of the perturbation study of Chapter 3. The second effect was that rotatory stiffness values obtained were also smaller, although they still varied significantly from the estimated value of 2.9 lbf·in/rad.

Model with Tip Masses, no Gravity (Trans Only)

Because of the extreme sensitivity of rotatory stiffness estimates to noise, the rotatory stiffness in the gravity-less model was fixed, just as it was in the model that included gravity.

Tables 43 and 44 give the results of model reconciliation when only transverse stiffness is fitted. Appendix B contains the individual mode shape fits used to select mode shapes for this fit.

Table 43: Simultaneous Fits ($S/N > 8.0$, Trans Only, no Gravity)

Length	# of Cords	trans [lbf/in]	modes included
Long	1	6.59	2-5
	2	0.38	1-3
	3	21.1	2-5
Medium	1	0	1-5
	2	0	1-5
	3	0.27	1-5
Short	1	7.09	2-5
	2	2.64	1-5
	3	0	1-5

Table 44: Simultaneous Fits ($S/N > 16$, Trans Only, no Gravity)

Length	# of Cords	trans [lbf/in]	modes included
Long	1	6.59	2-5
	2	8.03	2-3
	3	20.7	2-4
Medium	1	0	2-3
	2	0	1-2
	3	10.4	2-4
Short	1	10.7	2,4-5
	2	2.50	1
	3	0	1-5

The effects of removing gravity from the model in which only transverse stiffnesses were fitted was not as significant as it was in the previous model case, where both transverse and rotatory stiffnesses were fitted. Additionally, no pattern to the effects was evident. Transverse stiffness values went both up and down here, while they only went down in the previous model case.

Overview of Cases

Table 45 below shows the different models used in the reconciliation process. Direct comparisons between the results of these five cases would have been useful, but the amount of variation caused by noise kept any meaningful comparison from being made.

The first three models provided significant insight into the reconciliation problem. The first case demonstrated the importance of making the model as complete as possible. The second case demonstrated the value of determining what

effect noise might have on the results of the process. The third case represented a best effort at solving the problem.

The incompletely converged gravity-effect provided the motivation for the last two models used in the reconciliation problem. In these two models, gravity was left out, providing stiffness values that were not significantly different.

Table 45: Reconciliation Model Summary

1. Model including gravity but not transducer mass; both transverse and rotatory stiffness estimated: In this case, an unusually large number of zero stiffnesses were obtained. This provided the motivation for performing the perturbation study at the end of Chapter 3, which showed that neglecting mass was approximately equivalent to reducing stiffness.

2. Model including both gravity and transducer mass; both transverse and rotatory stiffness estimated: More positive stiffness values were obtained, but values (especially rotatory) varied wildly. This provided the motivation for performing the noise study, which showed that rotatory stiffness estimates could not be accurately estimated with the amount of noise present in the data.

3. Model including both gravity and transducer mass; only transverse stiffnesses estimated: The stiffness values generated here came closer to the independent stiffness estimate derived in Chapter 4.

4. Model including transducer mass but not gravity; both transverse and rotatory stiffness estimated: The stiffness values here were slightly lower than those generated when gravity was included in the model; however, they still showed significant variations caused by noise.

5. Model including transducer mass but not gravity; only transverse stiffness estimated: The stiffness values here varied only slightly from those obtained from the model with gravity.

Chapter 8: Conclusions

Review of Objectives

As stated in Chapter 1, the goal of the research presented in this thesis was to investigate and implement a parameter-based approach for updating an analytical model of a structure under test conditions. Specifically, the analytical model was to be updated such that it included the effects of the suspension in the model of the structure. This was done in accordance with the issues of Table 1 of Chapter 1. The first three items of that list are repeated below in Table 46.

Table 46: Issues in Structural Modification

1. It is necessary to have an analytical model that sufficiently characterizes the behavior of the test structure, including the effects of the suspension. If this is not true, modifications are in effect being performed on the wrong structure.

2. In order to develop a correct model of a structure, there must exist justifiable methods for reconciling the model with experimental data. This reconciliation process should utilize all known information about the model and take into account uncertainty present in the model.

3. In order to correctly update the model, it is necessary to have accurate experimental data that effectively describes the dynamic behavior of the structure.

The issues of Table 46 were addressed by the objectives of the research, which were also listed in Chapter 1. These objectives are listed again in Table 47.

Table 47: Objectives of Research

1. Developing an analytical model that includes suspension and measurement transducer effects. This is the model to be updated to better represent the structure under actual test conditions.

2. Determining the likely relationship between the suspension and its effect on the boundary conditions of the structure. This is done to ensure that the suspension is incorporated into the model correctly.

3. Determine whether or not the structure and its suspension behave linearly. This is done to determine whether or not it is appropriate to use linear model theory.

4. Obtain experimental mode shapes from the structure under actual test conditions. These mode shapes comprise the experimental basis for updating the analytical model.

5. Reconcile the model with the experimental data using optimization techniques.

The scope of the research was intentionally kept small in order to better develop a model reconciliation methodology. The structure used in the experiment was a simple uniform beam, which can easily be modeled using either the finite element method or the Rayleigh-Ritz discretization technique. Including the suspension in the model was accomplished by adding external springs of equivalent effect at the top of beam. This kept the number of parameters involved in the model reconciliation reasonably small.

Thesis Summary

Chapter 3 focused on the first objective of Table 47. This chapter was dedicated to the process of developing and verifying an analytical model that could be

used to model the test beam used in the laboratory. Specifically, a Rayleigh-Ritz model (with quasi-comparison functions) of the beam was developed, implemented, and compared to independent theoretical results.

The second and third objectives of Table 47 were addressed in Chapter 4, where a model of the suspension system was developed. Specifically, the shock cord used to suspend the beam was studied to determine the relationship between the length and stiffness of the cord and its effective stiffness. This study revealed that the effective transverse stiffness caused by the cord was only a function of the length of the cord and the weight it was suspending. The axial stiffness of the cord dropped out of the equations. Linearity of the suspension was also confirmed here.

The fourth objective of Table 47 was addressed by Chapter 6, in which a system for gathering accurate data was discussed. Specifically, modal data was extracted from operating shape information acquired from the beam using a scanning laser-Doppler vibrometer. This was accomplished by acquiring operating shape information at frequencies near the natural frequencies of interest. Phase angle information was used to separate the operating shape into real and imaginary parts. As predicted by modal theory, the real part of a velocity-based operating shape is very close to the true mode shape. Two sources of systematic error, drop-outs and angle effects, were alleviated or removed before the data could be used.

Chapters 5 and 7 addressed the fifth objective of Table 47. In Chapter 5, the methodology by which the model and the data could be reconciled was developed. Transverse and rotatory stiffnesses at the tip of the were selected as the design parameters used to model the suspension, which was the main source of uncertainty in

the model. The parameters were updated using optimization techniques to minimize the least-squares difference between the experimentally obtained mode shapes and mode shapes predicted by the analytical model. A Monte-Carlo type study was performed to determine how sensitive the parameter update process was to the presence of noise in the experimental mode shapes.

Chapter 7 brought all of the techniques developed in the previous chapters together on a structure under test in the laboratory. This structure was a stainless steel beam suspended with the elastic shock cords used in typical modal tests. The model update procedure was used with the experimental data acquired from beam as described in Chapter 6. The stiffness estimates obtained from the model update procedure were compared to values based on the suspension model developed in Chapter 4.

Conclusions Regarding Approach

The Analytical Model

The Rayleigh-Ritz method, when combined with the technique of quasi-comparison functions, can be used to quickly and accurately solve the equations resulting from the application of the Euler-Bernoulli beam model. Convergence studies confirmed that the implementation of the model converged to correct frequencies from above, as predicted by theory. The frequencies were also compared to results from analysis and other computational models, where exact agreement was obtained.

There was evidence of considerable ill-conditioning present in the mass and stiffnesses built in the Rayleigh-Ritz method. This problem was the result of near

lack of linear independence, which could cause errors in the eigensystem solution and thus limit the useable number of terms in the Rayleigh-Ritz series. The problem was partially alleviated using a Jacobi-iteration based eigensystem solver, which specifically utilized the known matrix properties of symmetry and positive definiteness. Using this solver allowed the use of up to 27 Ritz terms instead of 19.

Convergence with the Rayleigh-Ritz model was usually obtained with as few as 15 terms. Convergence was achieved when additional stiffnesses were added in either the form of translational and rotatory springs at the ends or of a base stiffness. This provided further confirmation that the model was accurately predicting the behavior of the beam with mixed boundary conditions. Again, these results were successfully compared with results of the continuum mechanics computational model implemented in BEAM VI.

Gravity-induced tension effects, however, were not as accurately modeled because of incomplete Rayleigh-Ritz convergence. This was the case with even 27 terms, which was the maximum number possible without matrix ill-conditioning effects invalidating the results. It appears that the natural frequencies were converging to true values, although slowly. If substantially increased arithmetic precision were available, the problem should be solvable. Despite this lack of complete convergence, the effect was kept in the model with the intent of making it more complete. Comparisons with other models that included gravity tension effects would have helped, but none were available.

The Reconciliation Process

The first, and perhaps most important, stage in reconciling differences between the results of the analytical model and experimental data was determining which model parameters were to be updated. This was done by considering levels of uncertainty in the model. All of the dimensions of the beam were very carefully measured, so there was no justification for changing them. Similarly, there was no reason to change the material modulus or density, since the reconciliation process was based on mode shapes, which are invariant to such properties as long as there are no variations across the beam. The properties of the suspension, however, were almost completely unknown. The study therefore focused on improving estimates of the stiffnesses used to model the cord suspending the beam.

Once the choices for parameters were made, a procedure for updating the model was implemented using an optimization-based methodology. A non-linear least-squares process served as the basis for obtaining estimates of the translational and rotatory stiffnesses at the beam tip. A noise study was performed to determine how sensitive the process was to the presence of experimental noise. In this study, fixed amounts of Gaussian noise were added to the analytical solution and then run through the reconciliation process. The rotatory stiffness estimates were extremely sensitive to noise, while the translational stiffness estimates were less sensitive. In keeping with the idea of updating only meaningful parameters, a second set of calculations was performed in which only estimates of the translational stiffness were updated.

Reconciliation with Experimental Data

Before the model could be updated to more closely match the physical beam in the laboratory, experimental data from the beam had to be acquired. This was accomplished with LabVIEW 2.21-based system that used a laser-Doppler vibrometer to acquire data. Once the data collection was accomplished, the effects of systematic error had to be eliminated. The first of these was an angle effect, which was easily corrected with a trigonometric transformation. Determining the angles for this transformation, however, required solving a relatively complex optimization problem. The second source of error was the presence of drop-outs in the laser-Doppler vibrometer data. The effects of this were alleviated, though not eliminated, through the use of the relatively robust asymmetric-trimmed-mean statistic.

Once all of the data had been gathered and prepared, it was processed using the reconciliation procedure. Noise levels turned out to be relatively high, making it difficult to compare the estimates obtained with those predicted by the shock cord theory. The process was repeated with only translational stiffness estimates being updated. This somewhat improved the estimates obtained, but they still varied too much to be directly compared to the theoretical values. However, it was apparent that the reconciliation process had resulted in a more accurate model than the free-free beam, even in light of the uncertainty of the stiffness estimates.

Recommendations for Future Work

While the work presented in this thesis is fairly complete, there are three areas in which improvements could be made: the analytical model, the reconciliation

process, and data collection. Some of the recommendations are simple and could be implemented rather easily. Others would require a significant amount of work to be accomplished. Recommendations for the analytical model are discussed in Table 48.

Table 48: Recommendations for Future Work on Model

1. Replace the quasi-comparison functions with a complete and independent set of functions that allow for varying displacements and slopes at the ends. Possible sets of functions that might serve here are the Forsythe or Chebyshev polynomials. If the functions meet these requirements, the ill-conditioning problem would be eliminated.

2. Replace the Jacobi-based eigensolver of the Pascal program with a faster QR/tridiagonalization-based routine. This would increase the speed with which solutions are obtained and very slightly increase the accuracy of the results.

3. Reformulate the problem using the equations resulting from the Timoshenko beam model. This would allow the model to be used with thicker beams and at higher modes.

4. More completely model the suspension. Incorporating the mass of the cord into a suspension model should provide more accurate results.

5. Compare the model with results obtained using the finite-element method or a continuum mechanics solution. Ideally, this new model would contain gravitational tension effects in addition to the stiffness effects at the beam tip.

Next discussed are possible improvements to the reconciliation process, as listed in Table 49. Most of these involve a significant amount of work and would be best implemented in a separate, larger project.

Table 49: Recommendations for Future Work on Reconciliation

1. Incorporate natural frequency estimates into the sum-of-squares error term used in the optimization process. This would be especially useful for updating parameters such as beam thickness or material modulus.

2. Increase the size and scope of the Monte-Carlo noise study to obtain variance estimates on the suspension stiffness estimates.

3. Develop a better model of the suspension. Including the mass of the shock cord in the cord model would increase its accuracy, although this would probably require treating the cord as a second vibrating structure attached to the beam.

4. Expand the technique to work with more explicitly discretized models, such as the finite element method. A properly implemented parameter-based reconciliation technique should be useable for shape optimization as well as incorporation of suspension effects.

Finally, methods for improving the collection of data are discussed in Table 50. These suggestions are particularly important, since the noise levels present in the data taken for this thesis caused significant problems.

Table 50: Recommendations for Future Work in Data Gathering

1. Perform drop-out removal and any signal conditioning <u>before</u> the multiple-linear-regression estimates of magnitude and phase are obtained. This would significantly reduce the amount of noise in the operating shape data and eliminate the need to perform asymmetric-trimmed-means.

2. Use robust regression techniques to obtain better sine wave amplitudes and phase angles in the presence of drop-outs.

3. Record regression statistics along with regression results. This would enable the analyst to make better judgements on how to weight or use the data.

Concluding Remarks

In this thesis, the goal of developing a methodology for reconciling analytical models with data obtained in experiments was achieved. In the course of performing the work, the importance of having a complete model of the structure became apparent, as did the importance of understanding the effects of noise on the reconciliation process.

In particular, the presence of noise prevented a quantitative analysis of the results of the reconciliation process developed in the thesis. However, the methodology was demonstrated to be valid by analytical results. Combined with certain statistical tools, it could provide the basis for very powerful model update techniques.

References

[1] Meirovitch, Leonard, and Kwak, Moon K., "Convergence of the Classical Rayleigh-Ritz Method and the Finite Element Method," **AIAA Journal**, Vol. 28, No. 9, August 1990, pp. 1509-1516.

[2] Marquardt, D., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," **SIAM Journal on Applied Mathematics**, 1963, Vol. 11, pp. 431-441.

[3] Fletcher, R. "A New Approach to Variable Metric Algorithms," **Computer Journal**, 1970, Vol. 13, No. 3, pp. 317-322.

[4] Nelder, J. A., and Mead, R., "A Simplex Method for Function Minimization," **Computer Journal**, 1965, Vol. 7, pp. 308-313.

[5] Gill, P. E., and Murray, W., "Newton-Type Methods for Unconstrained and Linearly Constrained Optimization," **Mathematical Programming**, 1974, Vol. 7, pp. 311-350.

[6] Elliot, K. B., and Mitchell, L. D., "The Effect of Modal Truncation on Modal Modification," **Proceedings of the Fifth International Modal Analysis Conference**, Imperial College of Science and Technology, London, England, April 6-9, 1987, pp. 72-78.

[7] Baker, Mary, "Review of Test/Analysis Correlation Methods and Criteria for Validation of Finite Element Models for Dynamic Analysis," **Proceedings of the Tenth International Modal Analysis Conference**, San Diego, CA, February 3-7, 1992, pp. 985-991.

[8] Meirovitch, Leonard, and Kwak, Moon K., "Convergence of the Classical Rayleigh-Ritz Method and the Finite Element Method," **AIAA Journal**, Vol. 28, No. 8, August, 1990, pp. 1509-1516.

[9] Luk, Y. W. and Mitchell, L. D., "Analytical and Geometric Substructuring via Dual Modal Space Structural Modification," **Proceedings of the Second International Modal Analysis Conference**, Orlando, FL, February 6-9, 1984, Union College, Schenectady, NY, pp. 50-57.

[10] Richard, John, **Graduate Assistantship Research** (final report for Robert L. West), Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1992.

[11] Singh, M. P., and Abdelnaser, A. S., "Random Vibrations of Timoshenko Beams with Generalized Boundary Conditions," **Second International Conference on Stochastic Structural Dynamics**, Boca Raton, FL, May 9-11, 1990, pp. 173-192.

[12] Grandhi, R. V., and Venkayya, V. B., "Structural Optimization with Frequency Constraints," **AIAA Journal**, Vol. 26, No. 7, July, 1988, pp. 858-866.

[13] Olhoff, Niels, "Optimal Design of Vibrating Rectangular Plates," **International Journal of Solids and Structures**, No. 10, 1974, pp. 93-109.

[14] Elliot, K. B., and Mitchell, L. D., "Structural Modification Utilizing Beam Elements," **Proceedings of the Fifth International Modal Analysis Conference**, Imperial College of Science and Technology, London, England, April 6-9, 1987, pp. 956-965.

[15] Gordis, Joshua H., "An Analysis of the Improved Reduced System (IRS) Modal Reduction System," **Proceedings of the Tenth International Modal Analysis Conference**, San Diego, CA, February 3-7, 1992, pp. 471-479.

[16] O'Callahan, J., "A Procedure for an Improved Reduced System (IRS) Model," **Proceedings of the 7th International Modal Analysis Conference**, Kissimmee, Florida, 1989.

[17] Baruh, H., and Boka, J. B., "Modeling and Identification of Boundary Conditions in Flexible Structures," **The International Journal of Analytical and Experimental Modal Analysis**, Vol. 8, No. 2, April, 1993, pp. 107-117.

[18] Conti, P., and Bretl, J., "Mount Stiffness and Inertia Properties for Modal Test Data," **Modal Testing and Analysis, ASME**, Vol. 3, December, pp. 11-16.

[19] Meirovitch, **Computational Methods in Structural Dynamics**, Sijthoff & Noordhoff, Rockville, Maryland, 1980, pp. 230-232.

[20] Singh, M. P., and Abdelnaser, A. S., **Random Vibrations of Timoshenko Beams with Generalized Boundary Conditions**, 2nd International Conference of Stochastic Structural Dynamics, Boca Raton, Florida, 1990, pp. 173-192.

[21] Meirovitch, Leonard, **Analytical Methods in Vibrations**, Macmillan Publishing, New York, 1967, pp. 161-166.

[22] Gorman, Daniel J., **Free Vibration Analysis of Beams and Shafts**, Wiley, New York, 1975.

[23] Meirovitch, Leonard, **Analytical Methods in Vibrations**, Macmillan Publishing, New York, 1967, pp. 211-233.

[24] **Mathematica 2.1**, Wolfram Research, Urbana, Illinois, 1993.

[25] **Matlab-386 3.5k**, The Mathworks, South Matick, Massachusetts, 1991.

[26] Andrew Grace, **MATLAB Optimization Toolbox 1.0b**, The Mathworks, South Natick, Massachusetts, 1991.

[27] Meirovitch, Leonard, **Computational Methods in Structural Dynamics**, Sijthoff & Noordhoff, Rockville, Maryland, 1980, p. 250.

[28] Golub, Gene H., and Van Loan, Charles F., **Matrix Computations, Second Edition**, Johns Hopkins University Press, Baltimore, Maryland, 1989, pp. 469-470.

[29] Larry D. Mitchell, et al., **BEAM VI** (software), Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1992.

[30] **LabVIEW 2.21**, National Instruments, Austin, Texas, 1991.

[31] Birch, Jeffrey B., **Exploratory and Robust Data Analysis Notes** (for Stat 5334), Department of Statistics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1989, pp. 116-117.

[32] Beer, Ferdinand P., and Johnston, E. Russell, Jr., **Mechanics of Materials**, McGraw-Hill Book Co., New York, 1981, p. 584.

[33] Beer, Ferdinand P., and Johnston, E. Russell, Jr., **Mechanics of Materials**, McGraw-Hill Book Co., New York, 1981, p. 584.

Appendix A: Jacobi Iteration Based Generalized Eigensystem Solver

Difficulties with the eigenvalue solver built into Matlab (see Chapter 3) provided the motivation for developing an generalized eigenvalue solver that specifically utilized the knowledge that the system matrices were symmetric and positive semi-definite. The solver was based on an altermative algorithm was found in Golub and van Loan. This algorithm is summarized in Table A1 below.

Table A1: Steps for Symmetric Definite Algorithm

1. Define problem as $A\vec{x} = \lambda B\vec{x}$, where A and B are symmetric and B is positive definite.

2. Compute the Schur decomposition $B = VDV^T$, which is equivalent to solving the symmetric eigenvalue problem $B\vec{v} = d\vec{v}$, where the eigenvalues d are located on the main diagonal of D.

3. Sort eigenvalues d and corresponding eigenvectors $\vec{v}$ in matrices D and V such that the eigenvalues are ordered from smallest to largest.

4. Compute $G = V^T D^{-1/2}$, and then compute $C = GAG^T$.

5. Compute the Schur decomposition $\Lambda = QCQ^T$, which is equivalent to solving the symmetric eigenvalue problem $C\vec{q} = \lambda\vec{q}$, where the eigenvalues λ are on the main diagonal of Λ. These eigenvalues are also the eigenvalues of $A\vec{x} = \lambda B\vec{x}$.

6. Compute $X = G^T Q$. The $\vec{x}$ vectors coming from this matrix are the eigenvectors of $A\vec{x} = \lambda B\vec{x}$.

Solving this problem requires the solution of two symmetric problems, as seen in steps 2 and 5 of Table A1. Each of these subproblems was solved using a Jacobi iteration based eigensystem solver that utilized matrix symmetry. The algorithm was found in **Numerical Recipes in C.**[1] The code presented there was recoded in Pascal with as little change as possible.

---

[1]Press, William H., et al., **Numerical Recipes in C, the Art of Scientific Computing**, Cambridge University Press, New York, 1988, pp. 360-366.

Appendix B: Results of Reconciliation with Individual Mode Shapes

These results given in this appendix correspond to the simultaneous mode shape fits of Chapter 7. The individual fits were used to obtain estimates of noise levels on each shape. This information about noise was then used to determine which mode shapes were to be included in the simultaneous mode shape fits and which shapes were to be left out.

The results of the individual mode shape fits in which with both transverse and rotatory were fitted in the presence of gravity are given in Table A2. These were the first cases run, where the transducer masses had been neglected.

Table A2: Individual Fits (Trans, Rot, and Grav)

length/ mode	1 cord			2 cords			3 cord		
	trans	rot	S/N	trans	rot	S/N	trans	rot	S/N
long/1	0	126	5.2	0	0	13	0	0	3.6
2	0	0	19	0	0	36	12.6	65.6	68
3	0	0	15	0	255	18	0	0	23
4	0	0	19	0	0	2.8	0	0	16
5	0	0	18	16.9	0	4.7	0	0	9.3
medium/1	0	0	8.9	0	0	24	0	0	11
2	0	0	15	0	0	21	0	0	27
3	0	30.7	22	0	0	9.3	4.81	24.2	19
4	0	75.0	11	0	55.6	9.9	8.58	8.57	23
5	0	0	9.5	0	84.9	10	0	0	18
short/1	0	0	2.6	0.29	26.8	46	0	0	20
2	0	0	20	0	0	14	0	0	42
3	0	124	10	0	0	10	0	0	21
4	5.55	32.3	83	0	0	12	0	30.5	20
5	0	0	15	0	0	9.7	0	0	50

Because so many zero stiffness values were generated in this case, the effects of transducer mass was investigated. This was the incentive for the perturbation study of Chapter 3. In that study, it was determined that leaving the mass terms out was approximately equivalent to reducing the transverse stiffness value.

In light of the perturbation study results, the model reconciliation process was run again with transducer mass included. The results generated, as shown in Table 3, contained many more positive stiffness values. Also, the signal-to-noise ratios provided the basis for determine whether or not to include a particular mode shape in the simultaneous mode shape fits of Chapter 7.

Table A3: Individual Fits (Trans, Rot, Grav, and Mass)

length/ mode	1 cord			2 cords			3 cord		
	trans	rot	S/N	trans	rot	S/N	trans	rot	S/N
long/1	0	448	5.2	0.36	0	13	0	0	3.6
2	5.79	0	20	3.52	103	46	23.0	316	80
3	0	340	16	12.2	109	18	0	0	57
4	10.0	148	24	--	--	--	0	0	20
5	0.19	185	26	--	--	--	71.4	0	9.8
medium/1	0	0	9.2	0	0	36	0.12	0	12
2	0	0	18	0	104	27	0.15	240	33
3	0.22	11.1	24	0	0	9.8	46.5	215	19
4	0	495	12	87.5	0	10	137	0	23
5	0	919	10	0	2170	11	0	430	24
short/1	0	763	2.5	1.89	442	46	0	0	22
2	1.88	591	21	0	429	15	11.1	0	43
3	0	0	12	0	1020	10	0	0	28
4	88.7	997	110	7.63	0	13	4.48	250	25
5	0	478	19	0	0	11	82.7	1470	57

Table A4 shows the results of individual mode shape fits for each mode shape, with gravity and transducer masses, when only tranverse stiffness was fitted. The rotatory stiffness value was fixed at 2.9 lbf·in/rad, as was done in Chapter 7 with the transverse-only simulta-

neous mode shape fits. Again, the signal-to-noise ratios given here were used to determine which mode shapes were included in the simultaneous mode shape fits.

Table A4: Individual Fits (Trans, Grav, and Mass)

length/ mode	1 cord		2 cords		3 cords	
	trans	S/N	trans	S/N	trans	S/N
long/1	0	5.2	0.36	13	--	3.6
2	5.76	20	4.58	46	25.9	80
3	5.91	16	38.6	18	0	57
4	17.3	24	--	--	0	20
5	9.78	26	--	--	71.1	9.8
medium/1	0	9.2	0	36	0.11	12
2	0	18	0	27	2.68	33
3	17.7	24	0	9.8	51.6	19
4	4.15	12	87.4	10	136	23
5	65.6	10	137	11	34.8	23
short/1	--	--	2.94	43	0	22
2	8.00	20	3.13	15	11.0	43
3	0	12	24.5	10	0	28
4	114	80	7.48	13	16.7	25
5	22.6	18	0	11	197	51

Table A5 shows the results of individual mode shape fits for each mode shape, with transducer masses, when only tranverse stiffness was fitted. In this case, gravity was deliberately left out in case the incompletely converged effect was having a negative impact on the results. Signal-to-noise ratios were provided again.

Table A5: Individual Fits (Trans, Rot, and Mass)

length/ mode	1 cord			2 cords			3 cord		
	trans	rot	S/N	trans	rot	S/N	trans	rot	S/N
long/1	0	327	5.2	0.05	0	13	0	0	3.6
2	5.67	0	20	4.48	11.3	46	22.9	201	80
3	0	422	16	16.6	1000	18	0	0	57
4	23.8	92.2	24	98.7	0	--	0	0	20
5	6.54	519	26	267	0	--	97.9	0	9.8
medium/1	0	0	9.2	0	0	36	0	0	12
2	0	0	18	0	97.6	27	1.31	147	33
3	16.4	145	24	0.97	0	9.8	49.1	87.7	19
4	0	710	12	9.03	0	10	134	0	23
5	0	1380	10	0	2650	11	14.3	707	24
short/1	0	612	2.5	1.70	346	46	0	0	22
2	2.74	612	21	0	410	15	10.7	0	43
3	0	0	12	4.63	957	10	0	0	28
4	96.2	887	110	18.9	0	13	17.2	231	25
5	0	897	19	0	246	11	101	1570	57

Table A6 shows the results of individual mode shape fits for each mode shape, with transducer masses, when only tranverse stiffness was fitted. The rotatory stiffness value was again fixed at 2.9 lbf·in/rad. Signal-to-noise ratios are also given here.

Table A6: Individual Fits (Trans and Mass)

length/ mode	1 cord		2 cords		3 cords	
	trans	S/N	trans	S/N	trans	S/N
long/1	0	5.2	0.04	13	0	3.6
2	5.65	20	4.56	46	24.7	80
3	9.04	16	39.2	18	0	57
4	27.9	24	98.6	--	0	20
5	44.9	26	267	--	97.7	9.8
medium/1	0	9.2	0	36	0	12
2	0	18	0	27	2.77	33
3	19.8	24	0.90	9.8	51.0	19
4	17.4	12	90.3	10	134	23
5	93.1	10	159	11	65.9	23
short/1	0.29	--	2.50	43	0	22
2	7.78	20	3.20	15	10.3	43
3	0	12	26.0	10	0	28
4	132	80	18.7	13	27.5	25
5	57.2	18	0	11	208	51

Appendix C: MATLAB Source Code

This appendix contains all of the source code used in the research presented in this thesis. The code can be broken into three sections. The first section is comprised of MATLAB routines that were used in preprocessing the data. The calling sequence of these routines is shown in Fig. A1.

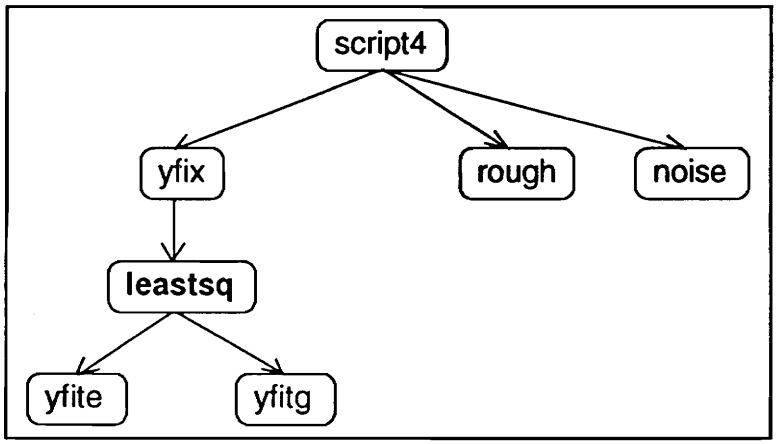

Figure A1: Preprocessing Routines

The second section of code is comprised of MATLAB routines that were used to perform the model reconciliation. The calling sequence of these routines is shown in Fig. A2. The bold faced routines **fmins**, **fminu**, **constr**, and **leastsq** are M-files that are part of the MATLAB Optimization Toolbox.

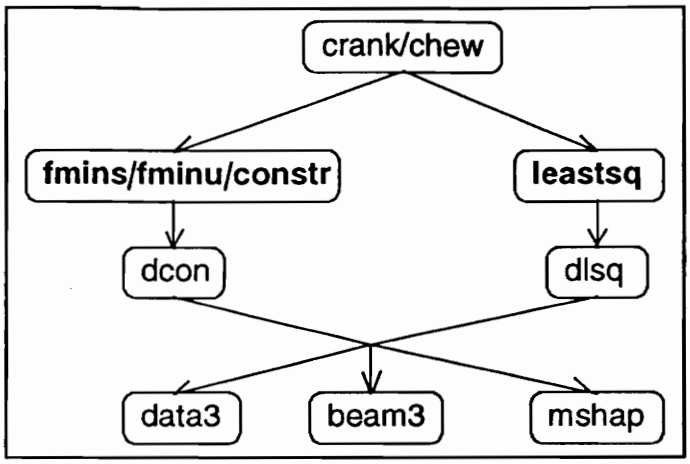

Figure A2: Reconciliation Routines

The third and final section of code is composed of side routines not used in the main work of the thesis. These routines were used to build operating shapes from mode shape data and to test for the presence of anti-clastic curvature in the experimental data. Figure A3 shows how these routines were organized.

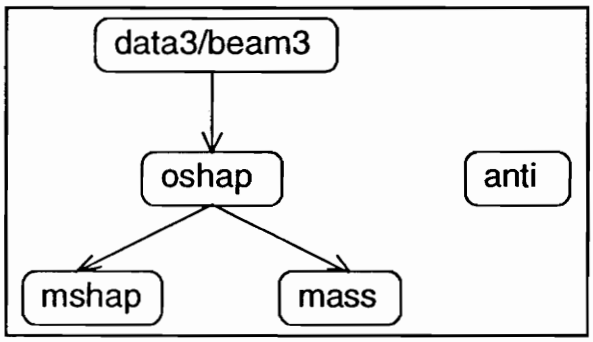

Figure A3: Miscellaneous Routines

Note: All of the actual m-file names were preceded with the three letters **"bel,"** the initial of the authors of this thesis. This was done to distinguish the m-files used her from those of other users when laboratory computers were used.

Preprocessing Routines

script4.m: This was the overall routine used to perform the data preprocessing. This preprocessing included correction of angle effects and dropouts, as discussed in Chapter 6.

```
bot=input('What is the bottom for cleaning?  ');
top=input('What is the top for cleaning?     ');
force=[f1;f2;f3;f4;f5]; clear f1 f2 f3 f4 f5
forcestd=[fs1;fs2;fs3;fs4;fs5]; clear fs1 fs2 fs3 fs4 fs5
gains=[g1;g2;g3;g4;g5]; clear g1 g2 g3 g4 g5
freqs=[hz1;hz2;hz3;hz4;hz5]; clear hz1 hz2 hz3 hz4 hz5
[y,unangle,dist]=belyfix(y1+y2+y3+y4+y5); y=y/54.75;
clear y1 y2 y3 y4 y5 x1 x2 x3 x4 x5
mag=zeros(length(y),5); pha=mag; rsh=mag; ish=mag;
if exist('rough')
  [mag(:,1),pha(:,1),rsh(:,1),ish(:,1)]=belrough(re1,im1,bot,top,unangle);
  [mag(:,2),pha(:,2),rsh(:,2),ish(:,2)]=belrough(re2,im2,bot,top,unangle);
  [mag(:,3),pha(:,3),rsh(:,3),ish(:,3)]=belrough(re3,im3,bot,top,unangle);
  [mag(:,4),pha(:,4),rsh(:,4),ish(:,4)]=belrough(re4,im4,bot,top,unangle);
  [mag(:,5),pha(:,5),rsh(:,5),ish(:,5)]=belrough(re5,im5,bot,top,unangle);
else
  [mag(:,1),pha(:,1),rsh(:,1),ish(:,1)]=belnoise(re1,im1,bot,top,unangle);
  [mag(:,2),pha(:,2),rsh(:,2),ish(:,2)]=belnoise(re2,im2,bot,top,unangle);
  [mag(:,3),pha(:,3),rsh(:,3),ish(:,3)]=belnoise(re3,im3,bot,top,unangle);
  [mag(:,4),pha(:,4),rsh(:,4),ish(:,4)]=belnoise(re4,im4,bot,top,unangle);
  [mag(:,5),pha(:,5),rsh(:,5),ish(:,5)]=belnoise(re5,im5,bot,top,unangle);
end
clear re1 re2 re3 re4 re5 im1 im2 im3 im4 im5
mav=mean(mag')'; mav=mav/norm(mav)*sqrt(length(mav));
rnorm=zeros(length(y),5);
for i = [1:5]
  rnorm(:,i) = rsh(:,i)*sign(rsh(1,i))/norm(rsh(:,i))*sqrt(length(rsh));
end
rav=mean(rnorm')'; rav=rav/norm(rav)*sqrt(length(rav));
plot(y,rsh); title('real part'); pause;
plot(y,ish); title('imaginary part'); pause;
plot(y,pha); title('phase angle'); pause;
plot(y,mag); title('normalized magnitude'); pause;
plot(y,mag-mav*ones(1,5)); title('shift magnitude'); pause;
plot(y,rnorm); title('normalized real part'); pause;
plot(y,rnorm-rav*ones(1,5)); title('shift real part'); pause;
which=input('Which frequencies to you want?  ');
rnorm=rnorm(:,which);
plot(y,rnorm); title('normalized real part');
if (length(which) ~= 1)
  rav=mean(rnorm')'; rav=rav/norm(rav)*sqrt(length(rav)); pause;
  plot(y,rav); title('average normalized real part'); pause;
  plot(y,rnorm-rav*ones(1,length(which))); title('shift real part');
else
  rav=rnorm;
end;
```

yfix.m: This routine was used to generate a correction for the angle error effect. It used the Optimization Toolbox routine **leastsq.m** to implement Gauss-Newton error minimization.

```
function [newy,unangle,dist,esty]=belyfix(oldy);
%BELYFIX  Eliminates errors in y scales that may have occurred during the
%         scanning process.  OLDY is the column vector containing the y
%         coordinates of scan points on the beam.  NEWY contains a normalized
%         and corrected vector, UNANGLE contains a vector that will remove
%         angle effects when multiplied by a velocity vector, DIST is the
%         estimated distance, and ESTY holds the estimated vector used to
%         generate the angles.
%
%USAGE:   [newy, unangle, dist, esty] = belyfix(oldy);
%         Copyright, May 14, 1993:  Brian E. Lindholm.
if (nargin ~= 1)
  error('Incorrect number of input arguments.  1 is required.');
elseif (nargout > 4)
```

```
  error('Incorrect number of output arguments.  1, 2, 3, or 4 are required.');
end
ylen = [54.75 0.25]; opt0 = [0.2 -0.2 120];
newy = flipud(oldy)*(ylen(1)-2*ylen(2))/(oldy(length(oldy))-oldy(1));
newy = -newy + newy(1)+ylen(2);
if (nargout > 1)
  myopt = foptions; myopt(2) = 1e-8; myopt(3) = 1e-8; myopt(5)=1; myopt(7)=1;
  opt = leastsq('belyfite', opt0, myopt, 'belyfitg', oldy, ylen);
  unangle = ones(oldy)./cos([opt(1):(opt(2)-opt(1))/(length(oldy)-1):opt(2)]');
end
if (nargout > 2)
  dist = opt(3);
end
if (nargout > 3)
  esty = opt(3) * tan([opt(1):(opt(2)-opt(1))/(length(oldy)-1):opt(2)]');
  esty = -esty + esty(1)+ylen(2);
end
```

yfite: This routine provided error values that specified the differences between the z-values recorded by the LabVIEW-based data acquisition program and the z-values computed using the estimated minimum and maximum laser angles and distance.

```
function error=belyfite(alpha,y,len);
%BELYFITE  Calculates an error vector for an estimated set of y values.
%          ALPHA contains THETA1, THETA2, and D.  Y contains the target set
%          of y values.  LEN contains D and SPACING.
%
%USAGE     error = belyfite(alpha, y, len);
%          Copyright, March 28, 1993:  Brian E. Lindholm
ynorm=(len(1)-2*len(2))*flipud(y(:))/(y(length(y))-y(1));
%  Note:  The FLIPUP was required due to an error in the LabVIEW code.
yest=alpha(3)*tan([alpha(1):(alpha(2)-alpha(1))/(length(y)-1):alpha(2)]');
error=yest-ynorm;
```

yfitg: This routine provided gradients of the error values generated in the routines **yfite**.

```
function grad=belyfitg(alpha,y,len);
%BELYFITG  Calculates gradient vectors for an estimated set of y values.
%          ALPHA contains THETA1, THETA2, and D.  Y contains the target set
%          of y values.  LEN contains D and SPACING.
%
%USAGE     grad = belyfitg(alpha, y, len);
%          Copyright, May 14, 1993:  Brian E. Lindholm.
grad=zeros(3,length(y));
sec=(1.0)./cos([alpha(1):(alpha(2)-alpha(1))/(length(y)-1):alpha(2)]);
grad(1,:)=alpha(3)*[1.0:-1/(length(y)-1):0.0].*sec.*sec;
grad(2,:)=alpha(3)*[0.0:1/(length(y)-1):1.0].*sec.*sec;
grad(3,:)=tan([alpha(1):(alpha(2)-alpha(1))/(length(y)-1):alpha(2)]);
```

rough: This routine performs an asymmetric-trimmed beam on each of the real and imaginary parts acquired from LabVIEW before combining them into magnitude and phase angle parts. Use of this routine was dropped in favor of **noise**.

```
function [mag,phase,rshape,ishape]=belrough(re,im,lowi,hii,unangle);
%BELROUGH  Throws away bad data points from experimental beam data and combines
%          the remaining points into a smooth centerline value.  RE is the real
%          part of the data, IM is the imaginary, LOWI is the low index used in
%          the sorted-magnitude based noise reduction, HII is the high index,
%          and UNANGLE contains an angle correction factors.
%
%USAGE:    [mag, phase, rshape, ishape] = belrough(re, im, lowi, hii, unangle);
%          Copyright, June 2, 1993:  Brian E. Lindholm.
if (nargin ~= 5)
  error('Incorrect number of input arguments.  5 are required.');
```

```
elseif (nargout ~= 4)
  error('Incorrect number of output arguments.  4 are required.');
end
tsign = sign(mean(re'))'; tmag = sort(abs(re'));
rshape = mean(tmag(lowi:hii,:))'.*unangle(:).*tsign;
tsign = sign(mean(im'))'; tmag = sort(abs(im'));
ishape = mean(tmag(lowi:hii,:))'.*unangle(:).*tsign;
phase = angle(rshape+i*ishape); tmag = abs(rshape+i*ishape);
if (mean(mean(abs(re)))/mean(mean(abs(im)))) <= 1
  mag = tmag.*sign(ishape)*sign(im(1))/norm(tmag)*sqrt(length(tmag));
else
  mag = tmag.*sign(rshape)*sign(re(1))/norm(tmag)*sqrt(length(tmag));
end
```

noise: This routine combined the real and imaginary parts acquired from LabVIEW into magnitudes <u>before</u> performing the asymmetric-trimmed beam.

```
function [mag,phase,rshape,ishape]=belnoise(re,im,lowi,hii,unangle);
%BELNOISE  Throws away bad data points from experimental beam data and combines
%          the remaining points into a smooth centerline value.  RE is the real
%          part of the data, IM is the imaginary, LOWI is the low index used in
%          the sorted-magnitude based noise reduction, HII is the high index,
%          and UNANGLE contains an angle correction factors.
%
%USAGE:    [mag, phase, rshape, ishape] = belnoise(re, im, lowi, hii, unangle);
%          Copyright, June 7, 1993:  Brian E. Lindholm.
if (nargin ~= 5)
  error('Incorrect number of input arguments.  5 are required.');
elseif (nargout ~= 4)
  error('Incorrect number of output arguments.  4 are required.');
end
rshape = zeros(length(re),1); ishape = rshape; mag = rshape; phase = rshape;
[tmag,index] = sort(abs(re+sqrt(-1)*im)');
for i = [1:length(re)]
  rshape(i) = mean(re(i,index(lowi:hii,i))) * unangle(i);
  ishape(i) = mean(im(i,index(lowi:hii,i))) * unangle(i);
  mag(i) = abs(rshape(i)+sqrt(-1)*ishape(i));
  phase(i) = angle(rshape(i)+sqrt(-1)*ishape(i));
end
if (mean(abs(rshape))/mean(abs(ishape))) < 1
  mag = mag.*sign(ishape)*sign(ishape(1))/norm(mag)*sqrt(length(mag));
else
  mag = mag.*sign(rshape)*sign(rshape(1))/norm(mag)*sqrt(length(mag));
end
```

Reconciliation Routines

crank: The was the high level routine used to perform model reconciliation in which both tranverse and rotatory stiffness estimates were obtained. A grid of starting points for the optimization routines could be specified, but this was unnecesssary when the simplix/SQP combination was used.

```
function [trans,rot,res,n,zk]=belcrank(modes,y,shape,tk,rk,count);
%BELCRANK  Cranks out several optimization cases based on the users choice.
%          TRANS are the translational spring value, ROT the rotational spring
%          values, RES the least-squares residuals, N the number of function
%          evaluations, and ZK the zero-stiffness residual.  MODES is a list
%          of modes being investigated, Y is a set of y-vectors, SHAPE is a
%          set of experimental mode shapes, TK is a central guess at TRANS, RK
%          is a guess at ROT, and COUNT is the order of the trial grid.
%
%USAGE:    [trans, rot, res, n, zk] = belcrank(modes, y, shape, tk, rk, count);
%          Copyright, July 11, 1993:  Brian E. Lindholm.
myopt=foptions; myopt(1)=1; myopt(2)=1e-12; myopt(3)=1e-12; myopt(14)=100;
powers=[1:count]-(count+1.0)/2.0; powers=(10*ones(1,length(count))).^powers;
```

```
k1=tk*powers; k2=rk*powers; data=beldata3(28,1); t0=clock;
[zk,temp]=beldcon([0 0],modes,y,shape,data);
trans=zeros(count,count); rot=trans; res=trans; n=trans; usecon=-1;
for i = [1:count]
  for j = [1:count]
    if (usecon < 0)
      K=[k1(i); k2(j)];
      [K,out]=fmins('beldcon',K,myopt,[],modes,y,shape,data);
%     [K,out]=leastsq('beldlsq',K,myopt,[],modes,y,shape,data);
      if (min(K) < 0.0)
        usecon=1; disp('Bad Result:  Switching to Constrained Algorithm');
      end
    end
    if (usecon > 0)
      K=[k1(i); k2(j)];
      [K,out]=constr('beldcon',K,myopt,[],[],[],modes,y,shape,data);
      if (min(K) > 1e-4)
        usecon=-1; disp('Good Result:  Returning to Unconstrained Algorithm');
      end
    end
    trans(i,j)=K(1); rot(i,j)=K(2); res(i,j)=out(8); n(i,j)=out(10);
    disp('Finished for Trial:'); disp([i j]);
  end
end
disp('Elapsed Time (min)');
disp(etime(clock,t0)/60);
```

chew: This routine was identical to **crank**, except that only transverse stiffness estimates were obtained. The rotatory stiffness values were held at a fixed value.

```
function [trans,res,n,zk]=belchew(modes,y,shape,tk,count);
%BELCHEW  Cranks out several optimization cases based on the users choice.
%         TRANS are the translational spring value,  RES the least-squares
%         residuals, N the number of function evaluations, and ZK the zero-
%         stiffness residual.  MODES is a list of modes being investigated,
%         Y is a set of y-vectors, SHAPE is a set of experimental mode
%         shapes, TK is a central guess at TRANS, and COUNT is the order
%         of the trial vector.
%
%USAGE:   [trans, res, n, zk] = belchew(modes, y, shape, tk, count);
%         Copyright, July 11, 1993:  Brian E. Lindholm.
myopt=foptions; myopt(1)=1; myopt(2)=1e-12; myopt(3)=1e-12; myopt(14)=75;
powers=[1:count]-(count+1.0)/2.0; powers=(10*ones(1,length(count))).^powers;
k1=tk*powers; data=beldata3(28,1); t0=clock;
[zk,temp]=beldcon([0],modes,y,shape,data);
trans=zeros(1,count); res=trans; n=trans;
for i = [1:count]
  K=k1(i);
  [K,out]=fmins('beldcon',K,myopt,[],modes,y,shape,data);
% [K,out]=leastsq('beldlsq',K,myopt,[],modes,y,shape,data);
  if (K < 0.0)
    K=0; out(8)=zk;
  end
  trans(i)=K; res(i)=out(8); n(i)=out(10);
  disp('Finished for Trial:'); disp(i);
end
disp('Elapsed Time (min)');
disp(etime(clock,t0)/60);
```

dcon: This routine computed a sum-of-squares error term in comparisons of one or more experimental mode shapes with corresponding analytical mode shapes.

```
function [diff,constr]=beldcon(k,modes,y,expshape,data);
%BELDCON  Calculates the difference between experiment and analytical
%         mode shapes.  K is the square roots of the lateral and rotational
%         stiffnesses caused by the bungee cord, MODES is a list of the modes
%         under investigation, Y has sets of y-coordinates, EXPSHAPE has the
%         experimental mode shapes, and DATA is the beam parameter set.
%
%USAGE:   [diff,constr] = beldcon(k, modes, y, expshape, data);
%         Copyright, June 8, 1993:  Brian E. Lindholm.
```

```
if (nargin ~= 5)
  disp('Incorrect number of input parameters.');
  error('4 in addition to K are required.');
elseif (nargout > 2)
  error('Incorrect number of output parameters.  2 are required.');
end
x=0.25;
if (length(k) == 1)
  [vals,vect] = belbeam3(data,2,[x k],[x 2.815],[x 1.3e-4],[x 3.76e-5]);
else
  [vals,vect] = belbeam3(data,2,[x k(1)],[x k(2)],[x 1.3e-4],[x 3.76e-5]);
end
texp = expshape; tana = zeros(length(y),length(modes));
for i = [1:length(modes)]
  texp(:,i) = texp(:,i)/norm(texp(:,i))*sqrt(length(texp))*sign(texp(1,i));
  tana(:,i) = belmshap(modes(i)+2,y(:,i),vect);
  tana(:,i) = tana(:,i)/norm(tana(:,i))*sqrt(length(tana))*sign(tana(1,i));
end;
texp = reshape(texp,length(texp)*length(modes),1);
tana = reshape(tana,length(tana)*length(modes),1);
diff = (tana-texp)'*(tana-texp); constr = -k;
```

dlsq: This routine was identical to **dcon**, except that the vector of error values was returned instead of a sum-of-squares value.

```
function [diff,constr]=beldlsq(k,modes,y,expshape,data);
%BELDLSQ  Calculates the difference between experiment and analytical
%         mode shapes.  K is the square roots of the lateral and rotational
%         stiffnesses caused by the bungee cord, MODES is a list of the modes
%         under investigation, Y has sets of y-coordinates, EXPSHAPE has the
%         experimental mode shapes, and DATA is the beam parameter set.
%
%USAGE:   diff = beldiff3(k, modes, y, expshape, data);
%         Copyright, June 9, 1993:  Brian E. Lindholm.
if (nargin ~= 5)
  disp('Incorrect number of input parameters.');
  error('4 in addition to K are required.');
elseif (nargout > 2)
  error('Incorrect number of output parameters.  2 are required.');
end
x=0.25;
if (length(k) == 1)
  [vals,vect] = belbeam3(data,2,[x k],[x 2.815],[x 1.3e-4],[x 3.76e-5]);
else
  [vals,vect] = belbeam3(data,2,[x k(1)],[x k(2)],[x 1.3e-4],[x 3.76e-5]);
end
texp = expshape; tana = zeros(length(y),length(modes));
for i = [1:length(modes)]
  texp(:,i) = texp(:,i)/norm(texp(:,i))*sqrt(length(texp))*sign(texp(1,i));
  tana(:,i) = belmshap(modes(i)+2,y(:,i),vect);
  tana(:,i) = tana(:,i)/norm(tana(:,i))*sqrt(length(tana))*sign(tana(1,i));
end;
texp = reshape(texp,length(texp)*length(modes),1);
tana = reshape(tana,length(tana)*length(modes),1);
diff = tana-texp;
```

data3: This routine provided default values for a number of parameters specifying the properties of the beam. It was specifically set up to allow easy modification of the number of Ritz terms and gravity load without worrying about other values.

```
function data=beldata3(num,gravs);
%BELDATA3  Gives standard parameters for the stainless steel beam problem.
%          NUM determines how many degrees-of-freedom are available for the
%          eigenvalue problem.  GRAVS gives the number of gravities
%
%USAGE:    data = beldata2(num)
%          Copyright, June 1, 1993:  Brian E. Lindholm.
%
```

```
%NOTES:     The DATA vector contains:  (1) L    (5) rho    (9)  k1
%                                      (2) w    (6) p0     (10) g
%                                      (3) t    (7) g's    (11) num
%                                      (4) E    (8) ko
%           It defaults to:
%             [54.75, 3.0, 0.2605, 28e6, 0.282/386.4, 0, 1, 0, 0, 386.4, 26]'
if (nargin > 2)
  error('Incorrect number of input arguments.  0, 1, or 2 are required.');
elseif (nargout > 1)
  error('Incorrect number of output arguements.  1 is required.');
end
data=zeros(11,1); data(1)=54.75; data(2)=3.0; data(3)=0.2605; data(4)=28.0e6;
data(5)=0.282/386.4; data(10)=386.4;
if (nargin > 0)
  data(11)=num;
else
  data(11)=26;
end
if (nargin > 1)
  data(7)=gravs;
else
  data(7)=1.0;
end
```

beam3: This routine was the heart of the analytical model. It was the M-file that interacted with the **BEAM3.EXE** program. Beam parameters, along with beam tip stiffnesses and masses, are saved in a MAT-file by this routine, which then calls **BEAM3.EXE**. The program reads in the parameters, builds the system matrices, and computes the eigenvalues and eigenvectors, saving the results in a second MAT-file. The **beam3** routine then reads these results back into MATLAB.

```
function [out1,out2,out3,out4]=belbeam3(dataset,output,st,sr,mt,mr);
%BELBEAM3:  Creates natural frequencies and eigenvectors for a specified set of
%           modes.   DATASET contains the parameters characterizing the beam,
%           OUTPUT specified what output will be provided by the program, ST
%           contains columns of translational spring location(:,1)/values(:,2),
%           SR rotational springs, MT translational masses, and MR rotational
%           masses.  VALS are the natural frequencies, VECTS the eigenvectors,
%           K the stiffness matrix, and M the mass matrix.
%
%USAGE:     [vals, vect, K, M] = belbeam3(dataset, 3, st, sr, mt, mr);
%           [vals, vect] =       belbeam3(dataset, 2, st, sr, mt, mr);
%           [K, M] =             belbeam3(dataset, 1, st, sr, mt, mr);
%           Copyright, June 1, 1993:  Brian E. Lindholm.
%
%NOTES:     The DATASET vector contains:  (1) L    (5) rho    (9)  k1
%                                         (2) w    (6) p0     (10) g
%                                         (3) t    (7) g's    (11) num
%                                         (4) E    (8) ko
if (nargin < 2)
  error('Incorrect number of input arguments.  At least 2 are required.');
elseif ((output < 1) | (output > 3))
  error('Incorrect output argument.  1, 2, or 3 must be selected');
elseif (((output == 1) | (output == 2)) & (nargout ~= 2))
  error('Incorrect number of output arguments.  2 are required.');
elseif ((output == 3) & (nargout ~= 4))
  error('Incorrect number of output arguments.  4 are required.');
end
data = zeros(12,1); data(1:11) = dataset; data(12) = output;
if (nargin == 2)
  save beamin data;
elseif (nargin == 3)
  save beamin data st;
elseif (nargin == 4)
  save beamin data st sr;
elseif (nargin == 5)
  save beamin data st sr mt;
elseif (nargin == 6)
  save beamin data st sr mt mr;
```

```
end
!beam3
load beamout
if (output == 1)
  out1 = K3; out2 = M3;
elseif (output == 2)
  out1 = vals3; out2 = vect3;
elseif (output == 3)
  out1 = vals3; out2 = vect3; out3 = K3; out4 = M3;
end
```

mshap: This routine used the eigenvectors obtained by **beam3** along with the trial functions used in the Rayleigh-Ritz integrations to build spatial mode shape vectors.

```
function mshapes=belmshap(modes,x,vect);
%BELMSHAP:  Creates mode shapes for a specified set of modes.  MODES is a list
%           of mode numbers, X is a column of data points on the beam between
%           0 and 1, and VECT is a matrix of eigenvectors.
%
%USAGE:     mshapes = belmshap(modes, x, vect);
%           Copyright, May 17, 1993:  Brian E. Lindholm.
if (nargin ~= 3)
  error('Incorrect number of input arguments.  3 are required.');
elseif (nargout > 1)
  error('Incorrect number of output arguments.  1 is required.');
end
phi = zeros(length(x),length(vect)); phi(:,1) = ones(length(x),1);
for j = 1:((length(vect)-2)/2)
  phi(:,(2*j):(2*j+1)) = [sin(pi*j*x(:)) cos(pi*j*x(:))];
end
mshapes = phi*vect(:,modes);
```

Miscellaneous Routines

oshap: This routines used mode shape information obtained from **mshap** (along with natural frequency information) to build operating shape vectors at a given set of frequencies.

```
function [mag,phase,rshape,ishape]=beloshap(modes,x,vals,vect,damp,xd,drivef);
%BELOSHAP:  Creates a set of operating shapes based on a set of mode shapes
%           for a specified set of modes.  MODES is a list of mode numbers used
%           in the approximation, X is a column of data points on the beam
%           between 0 and 1, VALS is a list of natural frequencies, VECT is a
%           matrix of eigenvectors, DAMP is a damping coefficient, XD is the
%           location of the driving point, and DRIVEF is a list of driving
%           frequencies.
%
%USAGE:     [mag, phase, rshape, ishape] =
%              beloshap(modes, x, vals, vect, damp, xd, drivef);
%           Copyright, June 2, 1993:  Brian E. Lindholm.
if (nargin ~= 7)
  error('Incorrect number of input arguments.  7 are required.');
elseif (nargout ~= 4)
  error('Incorrect number of output arguments.  4 are required.');
end
mass = belmass(modes,vect,[5 250]);
mshapes = belmshap(modes,x,vect)*diag(mass);
drives = belmshap(modes,xd,vect)*diag(mass);
if length(vals) == min(size(vect))
  omegan = vals(modes)*2.0*pi;
elseif length(vals) == length(modes)
  omegan = vals*2.0*pi;
else
  error('Incorrect number of vals.  n-freq = n-modes or n-vects');
end
mag = zeros(length(x),length(drivef)); phase = mag; rshape = mag; ishape = mag;
for i = 1:length(drivef)
  omega = drivef(i)*2.0*pi;
```

```
  H = drives(:)./(omegan.*omegan-omega*omega+2.0*sqrt(-1)*damp*omega*omegan);
  temp = sqrt(-1)*mshapes*H;
  rshape(:,i) = real(temp); ishape(:,i) = imag(temp);
  phase(:,i) = angle(temp); tmag = abs(temp);
  if (sum(abs(real(temp)))/sum(abs(imag(temp)))) <= 1.0
    tmag = tmag.*sign(ishape(:,i))*sign(ishape(1,i));
  else
    tmag = tmag.*sign(rshape(:,i))*sign(rshape(1,i));
  end
  mag(:,i) = tmag/norm(tmag)*sqrt(length(tmag));
end
```

mass: This routine computed a modal mass for each mode specified. These results were used by **oshap** to build operating shapes.

```
function mass=belmass(modes,vect,options);
%BELMASS:  Determines modal mass for a specified set of modes.  MODES is a list
%          of mode numbers, VECT is a matrix of eigenvectors, and the optional
%          OPTIONS specifies the method of integration (1=reimann, 2=trapezoid,
%          3=simpson, 4=Legendre2, 5=Legendre3) type and the block count.
%
%USAGE:    mass = belmass(modes, vect, options);
%          Copyright, May 14, 1993:  Brian E. Lindholm.
if ((nargin ~= 2) & (nargin ~= 3))
  error('Incorrect number of input arguments.  2 or 3 are required.');
elseif (nargout > 1)
  error('Incorrect number of output arguments.  1 is required.');
end
if (nargin == 3)
  itype = options(1); blocks=options(2);
else
  itype = 5; blocks = 250;
end
len = 1.0/blocks; hlen = 0.5/blocks;
if (itype == 1)
  x = ([1:blocks]'-0.5)*len; wts = len*ones(blocks,1);
elseif (itype == 2)
  x = [0:blocks]'*len; wts = len*ones(blocks+1,1);
  wts(1)=hlen; wts(blocks+1)=hlen;
elseif (itype == 3)
  x = [0:blocks*2]'*hlen; wts = zeros(blocks*2+1,1);
  wts(3:2:blocks*2-1) = len*ones(blocks-1,1)/3.0;
  wts(2:2:blocks*2) = 2.0*len*ones(blocks,1)/3.0;
  wts(1) = hlen/3.0; wts(blocks*2+1) = hlen/3.0;
elseif (itype == 4)
  x = zeros(2*blocks,1); wts = x;
  x(1:2:blocks*2-1) = ([1:2:blocks*2-1]'-sqrt(1/3))*hlen;
  x(2:2:blocks*2) = ([1:2:blocks*2-1]'+sqrt(1/3))*hlen;
  wts(1:2:blocks*2-1) = ones(blocks,1)*hlen;
  wts(2:2:blocks*2) = ones(blocks,1)*hlen;
elseif (itype == 5)
  x = zeros(3*blocks,1); wts = x;
  x(1:3:blocks*3-2) = ([1:2:blocks*2-1]'-sqrt(0.6))*hlen;
  x(2:3:blocks*3-1) = [1:2:blocks*2-1]'*hlen;
  x(3:3:blocks*3) = ([1:2:blocks*2-1]'+sqrt(0.6))*hlen;
  wts(1:3:blocks*3-2) = ones(blocks,1)*(5/9)*hlen;
  wts(2:3:blocks*3-1) = ones(blocks,1)*(8/9)*hlen;
  wts(3:3:blocks*3) = ones(blocks,1)*(5/9)*hlen;
end
mshapes=belmshap(modes,x,vect);
mass=ones(modes(:))./sqrt((mshapes.*mshapes)'*wts);
```

anti: This routine was used at the very beginning of the research to search for anti-clastic effects in the raw data. None were ever found.

```
function clast=belanti(re,im);
%BELANTI  Checks an experimental mode shape for anticlastic effects by
%         fitting parabolae across the beam.  RE is the real part of the
%         mode shape data, IM is the imaginary, and CLAST is a column
%         vector of parabola coefficients.
%
```

```
%USAGE:   clast = belanti(re,im);
%         Copyright, March 28, 1993:  Brian E. Lindholm.
if (nargin ~= 2)
  error('Incorrect number of input arguments.  2 are required.');
elseif (nargout > 1)
  error('Incorrect number of output arguments.  1 is required.');
end
phase = mean(mean(abs(re)))/mean(mean(abs(im))); psize = min(size(re));
para = [-1.0:(2/(psize-1)):1.0].^2; para = para-para*ones(psize,1)/psize;
if phase <= 1.0
  tmag = im*sign(mean(im(1,:)));
else
  tmag = re*sign(mean(re(1,:)));
end
clast = tmag*para';
```

Appendix D: BEAM3.EXE Source Code

The program **BEAM3.EXE** was written in Turbo Pascal. It was written in three main parts, along with a small fourth part containing a couple of minor routines. The first part of the program is **BEAM3.PAS**, which contained the main body of the program. It is also contains the set of routines used for matrix assembly. The second part of the program, **MATLAB.PAS**, is a unit (similar to a pre-compiled header/ library file in C) containing routines for interfacing with MATLAB. The third part of the program, **MATVECT.PAS**, contains routines for manipulating matrices and vectors. The Jacobi iteration based generalized eigenvalue solver can be found here. Finally, **MISC.PAS** contains two additional routines used in the main body of the program.

BEAM3.PAS

```
program beam3;
  { Copyright, June 1, 1993.  Brian E. Lindholm }

  uses
    matvect, matlab, misc;

  type
    delta = array[1..20,1..2] of extended;

  var
    length, m, ei, p0, p1, k0, k1: extended;
    mslin, msrot, sprlin, sprrot: delta;
    num, out, i, j, ml, mr, sl, sr: integer;
    Kmat, Mmat, Vect: matrix; vals: vector;
    realp: bufptr; matfile: file;
    valid: boolean;


function qc(a: integer): integer;

begin
  { qc := cos(pi*a) }
  if odd(a) then
    qc := -1
  else
    qc := 1
end;
```

```
function one: extended;

begin
  one := pi
end;


function x: extended;

begin
  x := pi/2.0
end;


function xx: extended;

begin
  xx := pi/3.0
end;


function xxx: extended;

begin
  xxx := pi/4.0
end;


function sinax(a: integer): extended;

begin
  { sinax := (1-cos(pi*a))/a }
  sinax := (1-qc(a))/a
end;


function cosax(a: integer): extended;

begin
  { cosax := sin(pi*a)/a }
  cosax := 0.0;
end;


function xsinax(a: integer): extended;

begin
  { xsinax := (sin(pi*a)/a - pi*cos(pi*a))/(pi*a) }
  xsinax := -qc(a)/a
end;


function xcosax(a: integer): extended;

begin
  { xcosax := (pi*sin(pi*a) + (cos(pi*a)-1)/a)/(pi*a) }
  xcosax := pred(qc(a))/(pi*sqr(a))
end;


function xxsinax(a: integer): extended;

begin
  { xxsinax := ((((2-sqr(pi))*cos(pi*a))-2)/a + 2*pi*sin(pi*a))/sqr(pi*a) }
  xxsinax := (((2-sqr(pi))*qc(a))-2)/(a*sqr(pi*a))
end;


function xxcosax(a: integer): extended;

begin
  { xxcosax := ((sqr(pi*a)-2)*sin(pi*a)/a + 2*pi*cos(pi*a))/sqr(pi*a) }
  xxcosax := 2.0*qc(a)/(pi*sqr(a))
end;
```

```
function sinaxsinbx(a, b: integer): extended;

begin
  { sinaxsinbx := (b*sin(pi*a)*cos(pi*b) - a*cos(pi*a)*sin(pi*b))
                  /(sqr(a)-sqr(b)) }
  if a = b then
    sinaxsinbx := 0.5*pi
  else
    sinaxsinbx := 0.0;
end;


function sinaxcosbx(a, b: integer): extended;

begin
  { sinaxcosbx := (b*sin(pi*a)*sin(pi*b) + a*(cos(pi*a)*cos(pi*b)-1))
                  /(sqr(b)-sqr(a)) }
  if a = b then
    sinaxcosbx := 0.0
  else
    sinaxcosbx := a*pred(qc(a+b))/(sqr(b)-sqr(a))
end;


function cosaxcosbx(a, b: integer): extended;

begin
  { cosaxcosbx := (b*cos(pi*a)*sin(pi*b) - a*sin(pi*a)*cos(pi*b))
                  /(sqr(b)-sqr(a)) }
  if a = b then
    cosaxcosbx := 0.5*pi
  else
    cosaxcosbx := 0.0;
end;


function xsinaxsinbx(a, b: integer): extended;

begin
  { xsinaxsinbx := (pi*(b*(sqr(a)-sqr(b))*sin(pi*a)*cos(pi*b)
                      +a*(sqr(b)-sqr(a))*cos(pi*a)*sin(pi*b))
                    +2*a*b*(cos(pi*a)*cos(pi*b)-1)
                    +(sqr(a)+sqr(b))*sin(pi*a)*sin(pi*b))
                    /(pi*sqr(sqr(a)-sqr(b))) }
  if a = b then
    xsinaxsinbx := 0.25*pi
  else
    xsinaxsinbx := 2.0*a*b*pred(qc(a+b))/(pi*sqr(sqr(a)-sqr(b)))
end;


function xsinaxcosbx(a, b: integer): extended;

begin
  { xsinaxcosbx := (pi*(b*(sqr(b)-sqr(a))*sin(pi*a)*sin(pi*b)
                      +a*(sqr(b)-sqr(a))*cos(pi*a)*cos(pi*b))
                    +2*a*b*cos(pi*a)*sin(pi*b)
                    +(sqr(a)+sqr(b))*sin(pi*a)*cos(pi*b))
                    /(pi*sqr(sqr(a)-sqr(b))) }
  if a = b then
    xsinaxcosbx := -0.25/a
  else
    xsinaxcosbx := a*(sqr(b)-sqr(a))*qc(a+b)/sqr(sqr(a)-sqr(b))
end;


function xcosaxcosbx(a, b: integer): extended;

begin
  { xcosaxcosbx := (pi*(b*(sqr(b)-sqr(a))*cos(pi*a)*sin(pi*b)
                      +a*(sqr(a)-sqr(b))*sin(pi*a)*cos(pi*b))
                    +2*a*b*sin(pi*a)*sin(pi*b)
                    +(sqr(a)+sqr(b))*(cos(pi*a)*cos(pi*b)-1))
                    /(pi*sqr(sqr(a)-sqr(b))) }
  if a = b then
    xcosaxcosbx := 0.25*pi
  else
    xcosaxcosbx := (sqr(a)+sqr(b))*pred(qc(a+b))/(pi*sqr(sqr(a)-sqr(b)))
end;
```

```
function Vzero(x, length: extended; i, j: integer): extended;

  var
    a,b: integer;
    z: extended;

begin
  a := i div 2;
  b := j div 2;
  z := pi * x / length;
  if i = 0 then
    if j = 0 then
      Vzero := 1.0
    else if j = 1 then
      Vzero := z / pi
    else if odd(j) then
      Vzero := cos(b*z)
    else
      Vzero := sin(b*z)
  else if i = 1 then
    if j = 0 then
      Vzero := z / pi
    else if j = 1 then
      Vzero := sqr(z/pi)
    else if odd(j) then
      Vzero := z * cos(b*z) / pi
    else
      Vzero := z * sin(b*z) / pi
  else if odd(i) then
    if j = 0 then
      Vzero := cos(a*z)
    else if j = 1 then
      Vzero := z * cos(a*z) / pi
    else if odd(j) then
      Vzero := cos(a*z) * cos(b*z)
    else
      Vzero := cos(a*z) * sin(b*z)
  else
    if j = 0 then
      Vzero := sin(a*z)
    else if j = 1 then
      Vzero := z * sin(a*z) / pi
    else if odd(j) then
      Vzero := sin(a*z) * cos(b*z)
    else
      Vzero := sin(a*z) * sin(b*z)
  end;


function Vone(x, length: extended; i, j: integer): extended;

  var
    a,b: integer;
    z,a1,b1: extended;

begin
  a := i div 2;
  b := j div 2;
  a1 := a*pi/length;
  b1 := b*pi/length;
  z := pi * x / length;
  if (i = 0) or (j = 0) then
    Vone := 0.0
  else if i = 1 then
    if j = 1 then
      Vone := 1.0 / sqr(length)
    else if odd(j) then
      Vone := -b1 * sin(b*z) / length
    else
      Vone := b1 * cos(b*z) / length
  else if odd(i) then
    if j = 1 then
      Vone := -a1 * sin(a*z) / length
    else if odd(j) then
      Vone := a1*b1 * sin(a*z) * sin(b*z)
    else
      Vone := -a1*b1 * sin(a*z) * cos(b*z)
  else
    if j = 1 then
      Vone := a1 * cos(a*z) / length
```

```
    else if odd(j) then
      Vone := -a1*b1 * cos(a*z) * sin(b*z)
    else
      Vone := a1*b1 * cos(a*z) * cos(b*z)
  end;


function Izero(length, c0, c1: extended; i, j: integer): extended;

  var
    a,b: integer;
    c: extended;

begin
  a := i div 2;
  b := j div 2;
  c := length/pi;
  if i = 0 then
    if j = 0 then
      Izero := c * (c0*one+c1*x)
    else if j = 1 then
      Izero := c * (c0*x+c1*xx)
    else if odd(j) then
      Izero := c * (c0*cosax(b)+c1*xcosax(b))
    else
      Izero := c * (c0*sinax(b)+c1*xsinax(b))
  else if i = 1 then
    if j = 0 then
      Izero := c * (c0*x+c1*xx)
    else if j = 1 then
      Izero := c * (c0*xx+c1*xxx)
    else if odd(j) then
      Izero := c * (c0*xcosax(b)+c1*xxcosax(b))
    else
      Izero := c * (c0*xsinax(b)+c1*xxsinax(b))
  else if odd(i) then
    if j = 0 then
      Izero := c * (c0*cosax(a)+c1*xcosax(a))
    else if j = 1 then
      Izero := c * (c0*xcosax(a)+c1*xxcosax(a))
    else if odd(j) then
      Izero := c * (c0*cosaxcosbx(b,a)+c1*xcosaxcosbx(b,a))
    else
      Izero := c * (c0*sinaxcosbx(b,a)+c1*xsinaxcosbx(b,a))
  else
    if j = 0 then
      Izero := c * (c0*sinax(a)+c1*xsinax(a))
    else if j = 1 then
      Izero := c * (c0*xsinax(a)+c1*xxsinax(a))
    else if odd(j) then
      Izero := c * (c0*sinaxcosbx(a,b)+c1*xsinaxcosbx(a,b))
    else
      Izero := c * (c0*sinaxsinbx(a,b)+c1*xsinaxsinbx(a,b))
end;


function Ione(length, c0, c1: extended; i, j: integer): extended;

  var
    a,b: integer;
    c,a1,b1: extended;

begin
  a := i div 2;
  b := j div 2;
  c := length/pi;
  a1 := a*pi/length;
  b1 := b*pi/length;
  if (i = 0) or (j = 0) then
    Ione := 0.0
  else if i = 1 then
    if j = 1 then
      Ione := c * (c0*one/sqr(length)+c1*x/sqr(length))
    else if odd(j) then
      Ione := c * (-b1*c0*sinax(b)/length-b1*c1*xsinax(b)/length)
    else
      Ione := c * (b1*c0*cosax(b)/length+b1*c1*xcosax(b)/length)
  else if odd(i) then
    if j = 1 then
      Ione := c * (-a1*c0*sinax(a)/length-a1*c1*xsinax(a)/length)
```

```
    else if odd(j) then
      Ione := c * (a1*b1*c0*sinaxsinbx(b,a)+a1*b1*c1*xsinaxsinbx(b,a))
    else
      Ione := c * (-a1*b1*c0*sinaxcosbx(a,b)-a1*b1*c1*xsinaxcosbx(a,b))
  else
    if j = 1 then
      Ione := c * (a1*c0*cosax(a)/length+a1*c1*xcosax(a)/length)
    else if odd(j) then
      Ione := c * (-a1*b1*c0*sinaxcosbx(b,a)-a1*b1*c1*xsinaxcosbx(b,a))
    else
      Ione := c * (a1*b1*c1*cosaxcosbx(a,b)+a1*b1*c1*xcosaxcosbx(a,b))
end;


function Itwo(length, c0, c1: extended; i, j: integer): extended;

  var
    a,b: integer;
    c,a2,b2: extended;

begin
  a := i div 2;
  b := j div 2;
  c := length/pi;
  a2 := sqr(a*pi/length);
  b2 := sqr(b*pi/length);
  if (i = 0) or (i = 1) or (j = 0) or (j = 1) then
    Itwo := 0.0
  else if odd(i) then
    if odd(j) then
      Itwo := c * (a2*b2*c0*cosaxcosbx(b,a)+a2*b2*c1*xcosaxcosbx(b,a))
    else
      Itwo := c * (a2*b2*c0*sinaxcosbx(b,a)+a2*b2*c1*xsinaxcosbx(b,a))
  else
    if odd(j) then
      Itwo := c * (a2*b2*c0*sinaxcosbx(a,b)+a2*b2*c1*xsinaxcosbx(a,b))
    else
      Itwo := c * (a2*b2*c0*sinaxsinbx(a,b)+a2*b2*c1*xsinaxsinbx(a,b))
end;


function mij(length, m: extended; ml, mr: integer;
             mslin, msrot: delta; i, j: integer): extended;

  var
    k: integer;
    temp: extended;

begin
  temp := Izero(length,m,0.0,i,j);
  for k := 1 to ml do
    temp := temp + mslin[k,2] * Vzero(mslin[k,1],length,i,j);
  for k := 1 to mr do
    temp := temp + msrot[k,2] * Vone(msrot[k,1],length,i,j);
  mij := temp;
end;


function kij(l, ei, p0, p1, k0, k1: extended; sl, sr: integer;
             sprlin, sprrot: delta; i, j: integer): extended;

  var
    k: integer;
    temp: extended;

begin
  temp := Izero(length,k0,k1,i,j) + Ione(length,p0,p1,i,j) +
          Itwo(length,ei,0.0,i,j);
  for k := 1 to sl do
    temp := temp + sprlin[k,2] * Vzero(sprlin[k,1],length,i,j);
  for k := 1 to sr do
    temp := temp + sprrot[k,2] * Vone(sprrot[k,1],length,i,j);
  kij := temp;
end;


procedure datafile(var matfile: file; var valid: boolean;
                   var length, m, ei, p0, p1, k0, k1: extended;
                   var ml, mr, sl, sr, num, out: integer;
                   var mslin, msrot, sprlin, sprrot: delta);
```

```
  var
    width, thick, E, rho, gravs, g: extended;
    i, row, col: integer;
    realmat, imagmat: bufptr;
    name: string;
    complex: boolean;

begin
  new(realmat);
  new(imagmat);
  valid := false;
  ml := 0; mr := 0; sl := 0; sr := 0;
  while not eof(matfile) do begin
    loadmat(matfile, name, complex, row, col, realmat, imagmat);
    if caps(name) = 'DATA' then
      if (row*col <> 12) or (complex) then
        writeln('Invalid data matrix.')
      else begin
        valid := true;
        length := abs(realmat^[1]);
        width := abs(realmat^[2]);
        thick := abs(realmat^[3]);
        E := abs(realmat^[4]);
        rho := abs(realmat^[5]);
        p0 := realmat^[6];
        gravs := realmat^[7];
        k0 := abs(realmat^[8]);
        k1 := realmat^[9];
        g := abs(realmat^[10]);
        num := abs(round(realmat^[11]));
        out := abs(round(realmat^[12]));
        m := width * thick * rho;
        ei := E * width * thick * sqr(thick) / 12;
        p1 := length * width * thick * rho * g * gravs end
    else if caps(name) = 'MT' then
      if (col <> 2) or (complex) then
        writeln('Invalid MT matrix.')
      else begin
        ml := row;
        for i := 1 to row do begin
          mslin[i,1] := realmat^[i];
          mslin[i,2] := realmat^[i+row] end end
    else if caps(name) = 'MR' then
      if (col <> 2) or (complex) then
        writeln('Invalid MR matrix.')
      else begin
        mr := row;
        for i := 1 to row do begin
          msrot[i,1] := realmat^[i];
          msrot[i,2] := realmat^[i+row] end end
    else if caps(name) = 'ST' then
      if (col <> 2) or (complex) then
        writeln('Invalid ST matrix.')
      else begin
        sl := row;
        for i := 1 to row do begin
          sprlin[i,1] := realmat^[i];
          sprlin[i,2] := realmat^[i+row] end end
    else if caps(name) = 'SR' then
      if (col <> 2) or (complex) then
        writeln('Invalid SR matrix.')
      else begin
        sr := row;
        for i := 1 to row do begin
          sprrot[i,1] := realmat^[i];
          sprrot[i,2] := realmat^[i+row] end end end
end;


procedure MtoB(mat: matrix; buff: bufptr);

  var
    i,j,k: integer;

begin
  k := 0;
  for j := 1 to mat^.col do
    for i := 1 to mat^.row do begin
      inc(k);
```

```
      buff^[k] := mat^.xx[i,j] end
end;


procedure VtoB(vect: vector; buff: bufptr);

  var
    i: integer;

begin
  for i := 1 to vect^.len do
    buff^[i] := vect^.xx[i]
end;


begin
      { Load BEAMIN.MAT }
  assign(matfile, 'beamin.mat');
  {$I-} reset(matfile, 1); {$I+}
  if ioresult <> 0 then begin
    writeln('BEAMIN.MAT missing.  Procedure aborted.');
    halt end
  else begin
    datafile(matfile,valid,length,m,ei,p0,p1,k0,k1,ml,mr,sl,sr,num,out,
             mslin,msrot,sprlin,sprrot);
    close(matfile) end;
  if not valid then begin
    writeln('Invalid BEAMIN.MAT.  Procedure aborted.');
    halt end;

      { Perform calculations with stripping }
  new(Kmat); new(Mmat);
  minit(Kmat,pred(num),pred(num));
  minit(Mmat,pred(num),pred(num));
  if (out and 2) = 2 then begin
    new(Vect); new(vals);
    minit(Vect,pred(num),pred(num));
    vinit(vals,pred(num)) end;

  Kmat^.xx[1,1] := kij(length,ei,p0,p1,k0,k1,sl,sr,sprlin,sprrot,0,0);
  Mmat^.xx[1,1] := mij(length,m,ml,mr,mslin,msrot,0,0);
  for i := 2 to pred(num) do begin
    Kmat^.xx[i,1] := kij(length,ei,p0,p1,k0,k1,sl,sr,sprlin,sprrot,i,0);
    Kmat^.xx[1,i] := kij(length,ei,p0,p1,k0,k1,sl,sr,sprlin,sprrot,0,i);
    Mmat^.xx[i,1] := mij(length,m,ml,mr,mslin,msrot,i,0);
    Mmat^.xx[1,i] := mij(length,m,ml,mr,mslin,msrot,0,i);
    for j := 2 to pred(num) do begin
      Kmat^.xx[i,j] := kij(length,ei,p0,p1,k0,k1,sl,sr,sprlin,sprrot,i,j);
      Mmat^.xx[i,j] := mij(length,m,ml,mr,mslin,msrot,i,j) end end;

  if (out and 2) = 2 then begin
    gensymmeig(vals,Vect,Kmat,Mmat);
    sorteig(vals,Vect,vals,Vect,true);
    for i := 1 to pred(num) do
      vals^.xx[i] := signr(vals^.xx[i])*sqrt(abs(vals^.xx[i]))*0.5/pi end;

      { Save BEAMOUT.MAT }
  new(realp);
  assign(matfile, 'beamout.mat');
  rewrite(matfile, 1);
  if (out and 1) = 1 then begin
    MtoB(Kmat,realp);
    savemat(matfile, 'K3', false, pred(num), pred(num), realp, nil);
    MtoB(Mmat,realp);
    savemat(matfile, 'M3', false, pred(num), pred(num), realp, nil) end;
  if (out and 2) = 2 then begin
    VtoB(vals,realp);
    savemat(matfile, 'vals3', false, pred(num), 1, realp, nil);
    MtoB(Vect,realp);
    savemat(matfile, 'vect3', false, pred(num), pred(num), realp, nil) end;
  close(matfile);

      { Clean up }
  dispose(Kmat); dispose(Mmat); dispose(realp);
  if (out and 2) = 2 then begin
    dispose(vals); dispose(Vect) end;
end.
```

MATLAB.PAS

```
unit matlab;
  { Copyright, June 1, 1993.  Brian E. Lindholm }

interface

  type
    buffer = array[1..4096] of double;
    bufptr = ^buffer;

  procedure loadmat(var infile: file; var name: string; var complex: boolean;
                    var rows, cols: integer; realp, imagp: bufptr);

  procedure savemat(var outfile: file; name: string; complex: boolean;
                    rows, cols: integer; realp, imagp: bufptr);

implementation

  type
    headrec = record
      mattype, matrows, matcols, imagf, namelen: longint;
    end;
    namebuf = array[1..255] of char;

  procedure loadmat(var infile: file; var name: string; var complex: boolean;
                    var rows, cols: integer; realp, imagp: bufptr);

    var
      header: headrec;
      zero: byte;
      temp: namebuf;
      i: integer;

  begin
    blockread(infile, header, 20);
    blockread(infile, temp, header.namelen);
    blockread(infile, realp^, 8*header.matcols*header.matrows);
    complex := (header.imagf = 1);
    if complex then
      blockread(infile, imagp^, 8*header.matcols*header.matrows);
    name[0] := chr(pred(header.namelen));
    for i := 1 to pred(header.namelen) do
      name[i] := temp[i];
    rows := header.matrows;
    cols := header.matcols;
  end;

  procedure savemat(var outfile: file; name: string; complex: boolean;
                        rows, cols: integer; realp, imagp: bufptr);

    var
      header: headrec;
      temp: namebuf;
      i: integer;

  begin
    header.mattype := 0;
    header.matrows := rows;
    header.matcols := cols;
    if complex then
      header.imagf := 1
    else
      header.imagf := 0;
    header.namelen := succ(length(name));
    blockwrite(outfile, header, 20);
    for i := 1 to length(name) do
      temp[i] := name[i];
    temp[header.namelen] := chr(0);
    blockwrite(outfile, temp, header.namelen);
    blockwrite(outfile, realp^, 8*header.matcols*header.matrows);
    if complex then
      blockwrite(outfile, imagp^, 8*header.matcols*header.matrows);
  end;

begin
end.
```

MATVECT.PAS

```
unit matvect;
  { Copyright, June 1, 1993.  Brian E. Lindholm }

  { References:

  Golub, Gene H., and Van Load, Charles F., Matrix Computations, Second
    Edition, Johns Hopkins University Press, Baltimore, 1989.

  Press, W. H., et al., Numerical Recipes in C, Cambridge University
    Press, Cambridge, 1990, pp. 360-366. }


interface

  const
    maxmatrix = 32;

  type
    mattype = record
      row, col: integer;
      xx: array[1..maxmatrix, 1..maxmatrix] of extended;
    end;
    matrix = ^mattype;

    vecttype = record
      len: integer;
      xx: array[1..maxmatrix] of extended;
    end;
    vector = ^vecttype;

  var
    matvecttol: extended;

    { output procedures }

  procedure printmat(A: matrix; title: string; l,d: integer);
  procedure printvect(x: vector; title: string; l,d: integer; isrow: boolean);

    { simple assignment procedures }

  procedure minit(C: matrix; m,n: integer);
  procedure vinit(z: vector; n: integer);
  procedure mzero(C: matrix; m,n: integer);
  procedure vzero(z: vector; n: integer);
  procedure mident(C: matrix; n: integer);
  procedure vone(z: vector; n: integer);
  procedure mequal(C: matrix; A: matrix);
  procedure vequal(z: vector; x: vector);

    { elementary arithmetic operations }

  procedure msum(C,A,B: matrix; d,e: extended);
  procedure vsum(z,x,y: vector; d,e: extended);
  procedure mprod(C,A,B: matrix; d: extended);
  function vdot(x,y: vector; d: extended): extended;
  procedure vcross(C: matrix; x,y: vector; d: extended);
  procedure mvprod(z: vector; A: matrix; x: vector; d: extended);
  procedure vmprod(z,x: vector; A: matrix; d: extended);
  procedure mvdiagprod(C,A: matrix; x: vector);
  procedure vmdiagprod(C: matrix; x: vector; A: matrix);
  procedure msymmprod(C,A,B: matrix; d: extended);
  procedure mscale(C,A: matrix; d: extended);
  procedure vscale(z,x: vector; d: extended);
  procedure mtrans(C,A: matrix);

    { elementary mixed arithmetic operations }

  procedure getdiag(z: vector; A: matrix);
  procedure makediag(C: matrix; x: vector);
  procedure putdiag(C,A: matrix; x: vector);
  procedure getvect(z: vector; A: matrix; n: integer; isrow: boolean);
  procedure makevect(C: matrix; x: vector; isrow: boolean);
  procedure putvect(C,A: matrix; x: vector; n: integer; isrow: boolean);
```

```
    { various boolean checks }

  function missymm(A: matrix): boolean;
  function misdiag(A: matrix): boolean;
  function miszero(A: matrix): boolean;
  function viszero(x: vector): boolean;
  function misequal(A,B: matrix): boolean;
  function visequal(x,y: vector): boolean;

    { descriptors:  sizes and norms }

  function mrows(A: matrix): integer;
  function mcols(A: matrix): integer;
  function vlen(x: vector): integer;
  function monenorm(A: matrix): extended;
  function minfnorm(A: matrix): extended;
  function mfronorm(A: matrix): extended;
  function vonenorm(x: vector): extended;
  function vinfnorm(x: vector): extended;
  function vtwonorm(x: vector): extended;

    { more complex operations}

  procedure symmeig(vals: vector; vects, A: matrix);
  procedure gensymmeig(vals: vector; vects, A, B: matrix);
  procedure sorteig(vals: vector; vects: matrix;
                    oldvals: vector; oldvects: matrix; ascend: boolean);

implementation


  procedure printmat(A: matrix; title: string; l,d: integer);

    var
      i, j: integer;

  begin
    if (title <> '') then
      writeln(title);
    for i := 1 to A^.row do begin
      for j := 1 to A^.col do
        write(A^.xx[i,j]:l:d);
      writeln end
  end;


  procedure printvect(x: vector; title: string; l,d: integer; isrow: boolean);

    var
      i: integer;

  begin
    if (title <> '') then
      writeln(title);
    if isrow then begin
      for i := 1 to x^.len do
        write(x^.xx[i]:l:d);
      writeln end
    else
      for i := 1 to x^.len do
        writeln(x^.xx[i]:l:d)
  end;


  procedure minit(C: matrix; m,n: integer);

  begin
    if (m < 0) or (m > maxmatrix) or (n < 0) or (n > maxmatrix) then begin
      writeln('minit:  matrix size out of bounds');
      halt end;
    C^.row := m; C^.col := n
  end;


  procedure vinit(z: vector; n: integer);

  begin
    if (n < 0) or (n > maxmatrix) then begin
      writeln('vinit:  vector length out of bounds');
      halt end;
```

```
  z^.len := n
end;


procedure mzero(C: matrix; m,n: integer);

  var
    i,j: integer;

begin
  if (m < 1) or (m > maxmatrix) or (n < 1) or (n > maxmatrix) then begin
    writeln('mzero:  matrix size out of bounds');
    halt end;
  for i := 1 to m do
    for j := 1 to n do
      C^.xx[i,j] := 0.0;
  C^.row := m; C^.col := n
end;


procedure vzero(z: vector; n: integer);

  var
    i: integer;

begin
  if (n < 1) or (n > maxmatrix) then begin
    writeln('vzero:  vector length out of bounds');
    halt end;
  for i := 1 to n do
    z^.xx[i] := 0.0;
  z^.len := n
end;


procedure mident(C: matrix; n: integer);

  var
    i,j: integer;

begin
  if (n < 1) or (n > maxmatrix) then begin
    writeln('ident:  matrix size out of bounds');
    halt end;
  for i := 1 to n do begin
    C^.xx[i,i] := 1.0;
    for j := succ(i) to n do begin
      C^.xx[i,j] := 0.0;
      C^.xx[j,i] := 0.0 end end;
  C^.row := n; C^.col := n
end;


procedure vone(z: vector; n: integer);

  var
    i: integer;

begin
  if (n < 1) or (n > maxmatrix) then begin
    writeln('vones:  vector length out of bounds');
    halt end;
  for i := 1 to n do
    z^.xx[i] := 1.0;
  z^.len := n
end;


procedure mequal(C: matrix; A: matrix);

  var
    i,j: integer;

begin
  if (C <> A) then begin
    for i := 1 to A^.row do
      for j := 1 to A^.col do
        C^.xx[i,j] := A^.xx[i,j];
    C^.row := A^.row; C^.col := A^.col end
end;
```

```
procedure vequal(z: vector; x: vector);

  var
    i: integer;

begin
  if (z <> x) then begin
    for i := 1 to x^.len do
      z^.xx[i] := x^.xx[i];
    z^.len := x^.len end
end;


procedure msum(C,A,B: matrix; d,e: extended);

  var
    i,j: integer;

begin
  if (A^.row <> B^.row) or (A^.col <> B^.col) then begin
    writeln('msum:  matrix sizes are different');
    halt end;
  for i := 1 to A^.row do
    for j := 1 to A^.col do
      C^.xx[i,j] := d * A^.xx[i,j] + e * B^.xx[i,j];
  C^.row := A^.row; C^.col := A^.col
end;


procedure vsum(z,x,y: vector; d,e: extended);

  var
    i: integer;

begin
  if (x^.len <> y^.len) then begin
    writeln('vsum:  vector lengths are not equal');
    halt end;
  for i := 1 to x^.len do
    z^.xx[i] := d * x^.xx[i] + e * y^.xx[i];
  z^.len := x^.len
end;


procedure mprod(C,A,B: matrix; d: extended);

  var
    i,j,k: integer;
    temp: extended;
    tempC: matrix;
    usetemp: boolean;

begin
  if (A^.col <> B^.row) then begin
    writeln('mprod:  first matrix cols not equal to second matrix rows');
    halt end;
  usetemp := ((C = A) or (C = B));
  if usetemp then
    new(tempC)
  else
    tempC := C;
  for i := 1 to A^.row do
    for j := 1 to B^.col do begin
      temp := 0.0;
      for k := 1 to A^.col do
        temp := temp + A^.xx[i,k] * B^.xx[k,j];
      tempC^.xx[i,j] := d * temp end;
  tempC^.row := A^.row; tempC^.col := B^.col;
  if usetemp then begin
    mequal(C,tempC);
    dispose(tempC) end
end;


function vdot(x,y: vector; d: extended): extended;

  var
    i: integer;
    temp: extended;
```

```
begin
  if (x^.len <> y^.len) then begin
    writeln('vdot:  vectors are different lengths');
    halt end;
  temp := 0.0;
  for i := 1 to x^.len do
    temp := temp + x^.xx[i] * y^.xx[i];
  vdot := d * temp
end;


procedure vcross(C: matrix; x,y: vector; d: extended);

  var
    i,j: integer;

begin
  for i := 1 to x^.len do
    for j := 1 to y^.len do
      C^.xx[i,j] := d* x^.xx[i] * y^.xx[j];
  C^.row := x^.len; C^.col := y^.len
end;


procedure mvprod(z: vector; A: matrix; x: vector; d: extended);

  var
    i,j: integer;
    temp: extended;
    usetemp: boolean;
    tempz: vector;

begin
  if (A^.col <> x^.len) then begin
    writeln('mvprod:  matrix columns not equal to vector length');
    halt end;
  usetemp := (z = x);
  if usetemp then
    new(tempz)
  else
    tempz := z;
  for i := 1 to A^.row do begin
    temp :=  0.0;
    for j := 1 to x^.len do
      temp := temp + A^.xx[i,j] * x^.xx[j];
    tempz^.xx[i] := d * temp end;
  tempz^.len := A^.row;
  if usetemp then begin
    vequal(z,tempz);
    dispose(tempz) end
end;


procedure vmprod(z,x: vector; A: matrix; d: extended);

  var
    i,j: integer;
    temp: extended;
    usetemp: boolean;
    tempz: vector;

begin
  if (A^.row <> x^.len) then begin
    writeln('mvprod:  vector length not equal to matrix rows');
    halt end;
  usetemp := (z = x);
  if usetemp then
    new(tempz)
  else
    tempz := z;
  for j := 1 to A^.col do begin
    temp :=  0.0;
    for i := 1 to x^.len do
      temp := temp + A^.xx[i,j] * x^.xx[j];
    tempz^.xx[i] := d * temp end;
  tempz^.len := A^.col;
  if usetemp then begin
    vequal(z,tempz);
    dispose(tempz) end
end;
```

```
procedure mvdiagprod(C,A: matrix; x: vector);

  var
    i,j: integer;

begin
  if (A^.col <> x^.len) then begin
    writeln('mvprod:  matrix columns not equal to vector length');
    halt end;
  for i := 1 to A^.row do
    for j := 1 to A^.col do
      C^.xx[i,j] := A^.xx[i,j] * x^.xx[j];
  C^.row := A^.row; C^.col := A^.col
end;


procedure vmdiagprod(C: matrix; x: vector; A: matrix);

  var
    i,j: integer;

begin
  if (A^.row <> x^.len) then begin
    writeln('mvprod:  vector length not equal to matrix rows');
    halt end;
  for i := 1 to A^.row do
    for j := 1 to A^.col do
      C^.xx[i,j] := A^.xx[i,j] * x^.xx[i];
  C^.row := A^.row; C^.col := A^.col
end;


procedure msymmprod(C,A,B: matrix; d: extended);

  var
    i,j,k: integer;
    temp: extended;
    tempC: matrix;
    usetemp: boolean;

begin
  if (A^.row <> A^.col) or (B^.row <> B^.col) then begin
    writeln('msymmprod:  matrices are not square');
    halt end
  else if (A^.row <> B^.row) then begin
    writeln('msymmprod:  matrix sizes are different');
    halt end
  else if not (missymm(A) and missymm(B)) then begin
    writeln('msymmprod:  matrices are not symmetric');
    halt end;
  usetemp := ((C = A) or (C = B));
  if usetemp then
    new(tempC)
  else
    tempC := C;
  for i := 1 to A^.row do begin
    temp := 0.0;
    for k := 1 to A^.col do
      temp := temp + A^.xx[i,k] * B^.xx[k,i];
    tempC^.xx[i,i] := d * temp;
    for j := succ(i) to A^.row do begin
      temp := 0.0;
      for k := 1 to A^.col do
        temp := temp + A^.xx[i,k] * B^.xx[k,j];
      temp := d * temp; tempC^.xx[i,j] := temp;
      tempC^.xx[j,i] := temp end end;
  tempC^.row := A^.row; tempC^.col := A^.col;
  if usetemp then begin
    mequal(C,tempC);
    dispose(tempC) end
end;


procedure mscale(C,A: matrix; d: extended);

  var
    i,j: integer;
```

```
begin
  for i := 1 to A^.row do
    for j := 1 to A^.col do
      C^.xx[i,j] := d * A^.xx[i,j];
  C^.row := A^.row; C^.col := A^.col
end;


procedure vscale(z,x: vector; d: extended);

  var
    i: integer;

begin
  for i := 1 to x^.len do
    z^.xx[i] := x^.xx[i];
  z^.len := x^.len
end;


procedure mtrans(C: matrix; A: matrix);

  var
    i,j: integer;
    usetemp: boolean;
    tempC: matrix;

begin
  usetemp := (A = C);
  if usetemp then
    new(tempC)
  else
    tempC := C;
  for i := 1 to A^.row do
    for j := 1 to A^.col do
      tempC^.xx[j,i] := A^.xx[i,j];
  tempC^.row := A^.col; tempC^.col := A^.row;
  if usetemp then begin
    mequal(C,tempC);
    dispose(tempC) end
end;


procedure getdiag(z: vector; A: matrix);

  var
    i: integer;

begin
  if (A^.row <> A^.col) then begin
    writeln('getdiag:  matrix not square');
    halt end;
  for i := 1 to A^.row do
    z^.xx[i] := A^.xx[i,i];
  z^.len := A^.row
end;


procedure makediag(C: matrix; x: vector);

  var
    i,j: integer;

begin
  for i := 1 to x^.len do begin
    C^.xx[i,i] := x^.xx[i];
    for j := succ(i) to x^.len do begin
      C^.xx[i,j] := 0.0;
      C^.xx[j,i] := 0.0 end end;
  C^.row := x^.len; C^.col := x^.len
end;


procedure putdiag(C: matrix; A: matrix; x: vector);

  var
    i: integer;

begin
  if (A^.row <> A^.col) then begin
```

```
    writeln('putdiag:  incorrectly sized matrix');
    halt end;
  if (C <> A) then
    mequal(C,A);
  for i := 1 to x^.len do
    C^.xx[i,i] := x^.xx[i]
end;


procedure getvect(z: vector; A: matrix; n: integer; isrow: boolean);

  var
    i: integer;

begin
  if (n < 1) or ((n > A^.row) and isrow) or ((n > A^.col) and not isrow) then begin
    writeln('getvect:  incorrect vector location');
    halt end;
  if isrow then begin
    for i := 1 to A^.col do
      z^.xx[i] := A^.xx[n,i];
    z^.len := A^.col end
  else begin
    for i := 1 to A^.row do
      z^.xx[i] := A^.xx[i,n];
    z^.len := A^.row end
end;


procedure makevect(C: matrix; x: vector; isrow: boolean);

  var
    i: integer;

begin
  if isrow then begin
    for i := 1 to x^.len do
      C^.xx[1,i] := x^.xx[i];
    C^.row := 1; C^.col := x^.len end
  else begin
    for i := 1 to x^.len do
      C^.xx[i,1] := x^.xx[i];
    C^.row := x^.len; C^.col := 1 end
end;


procedure putvect(C: matrix; A: matrix; x: vector; n: integer;
             isrow: boolean);

  var
    i: integer;

begin
  if (n < 1) or ((n > C^.row) and isrow) or ((n > C^.col) and not isrow) then begin
    writeln('getvect:  incorrect vector location');
    halt end
  else if ((x^.len <> C^.col) and isrow) or ((x^.len <> C^.row) and not isrow) then
    begin writeln('getvect:  incorrectly sized matrix and vector');
    halt end;
  if (C <> A) then
    mequal(C,A);
  if isrow then
    for i := 1 to x^.len do
      C^.xx[n,i] := x^.xx[i]
  else
    for i := 1 to x^.len do
      C^.xx[i,n] := x^.xx[i]
end;


function missymm(A: matrix): boolean;

  var
    i,j: integer;
    sum, diff: extended;
    temp: boolean;

begin
  if (A^.row = 1) and (A^.col = 1) then
    temp := true
```

```
  else begin
    temp := (A^.row = A^.col);
    if temp then begin
      sum := 0.0; diff := 0.0;
      for i := 1 to A^.row do
        for j := succ(i) to A^.row do begin
          sum := sum + abs(A^.xx[i,j]) + abs(A^.xx[j,i]);
          diff := diff + abs(A^.xx[i,j] - A^.xx[j,i]) end;
      if (sum = 0.0) then
        temp := true
      else
        temp := ((diff/sum) < matvecttol) end end;
  missymm := temp
end;


function misdiag(A: matrix): boolean;

  var
    i,j: integer;
    onsum, offsum: extended;
    temp: boolean;

begin
  if (A^.row = 1) and (A^.col = 1) then
    temp := true
  else begin
    temp := (A^.row = A^.col);
    if temp then begin
      onsum := 0.0; offsum := 0.0;
      for i := 1 to A^.row do begin
        onsum := onsum + abs(A^.xx[i,i]);
        for j := succ(i) to A^.row do
          offsum := offsum + abs(A^.xx[i,j]) + abs(A^.xx[j,i]) end;
      if (onsum = 0.0) and (offsum <> 0.0) then
        temp := false
      else if (onsum = 0.0) and (offsum = 0.0) then
        temp := true
      else
        temp := ((offsum/onsum) < matvecttol) end end;
  misdiag := temp
end;


function miszero(A: matrix): boolean;

  var
    i,j: integer;
    sum: extended;

begin
  sum := 0.0;
  for i := 1 to A^.row do
    for j := 1 to A^.col do
      sum := sum + abs(A^.xx[i,j]);
  miszero := (sum < (matvecttol * sqrt(A^.row * A^.col)))
end;


function viszero(x: vector): boolean;

  var
    i: integer;
    sum: extended;

begin
  sum := 0.0;
  for i := 1 to x^.len do
    sum := sum + abs(x^.xx[i]);
  viszero := (sum < matvecttol * sqrt(x^.len))
end;


function misequal(A,B: matrix): boolean;

  var
    i,j: integer;
    sum,diff: extended;
    temp: boolean;
```

```
begin
  if (A^.row <> B^.row) or (A^.col <> B^.col) then
    temp := false
  else begin
    diff := 0.0; sum := 0.0;
    for i := 1 to A^.row do
      for j := 1 to A^.col do begin
        sum := sum + abs(A^.xx[i,j]) + abs(B^.xx[i,j]);
        diff := diff + abs(A^.xx[i,j] - B^.xx[i,j]) end;
    if (sum = 0.0) then
      temp := true
    else
      temp := ((diff/sum) < matvecttol) end;
  misequal := temp
end;


function visequal(x,y: vector): boolean;

  var
    i: integer;
    sum,diff: extended;
    temp: boolean;

begin
  if (x^.len <> y^.len) then
    temp := false
  else begin
    diff := 0.0; sum := 0.0;
    for i := 1 to x^.len do begin
      sum := sum + abs(x^.xx[i]) + abs(y^.xx[i]);
      diff := diff + abs(x^.xx[i] - y^.xx[i]) end;
    if (sum = 0.0) then
      temp := true
    else
      temp := ((diff/sum) < matvecttol) end;
  visequal := temp
end;


function mrows(A: matrix): integer;

begin
  mrows := A^.row
end;


function mcols(A: matrix): integer;

begin
  mcols := A^.col
end;


function vlen(x: vector): integer;

begin
  vlen := x^.len
end;


function monenorm(A: matrix): extended;

  var
    i,j: integer;
    temp,sum: extended;

begin
  temp := 0.0;
  for j := 1 to A^.col do begin
    sum := 0.0;
    for i := 1 to A^.row do
      sum := sum + abs(A^.xx[i,j]);
    if sum > temp then
      temp := sum end;
  monenorm := temp
end;
```

```
function minfnorm(A: matrix): extended;

  var
    i,j: integer;
    temp,sum: extended;

begin
  temp := 0.0;
  for i := 1 to A^.row do begin
    sum := 0.0;
    for j := 1 to A^.col do
      sum := sum + abs(A^.xx[i,j]);
    if sum > temp then
      temp := sum end;
  minfnorm := temp
end;


function mfronorm(A: matrix): extended;

  var
    i,j: integer;
    temp: extended;

begin
  temp := 0.0;
  for i := 1 to A^.row do
    for j := 1 to A^.col do
      temp := temp + sqr(A^.xx[i,j]);
  mfronorm := sqrt(temp)
end;


function vonenorm(x: vector): extended;

  var
    i: integer;
    temp: extended;

begin
  temp := 0.0;
  for i := 1 to x^.len do
    temp := temp + abs(x^.xx[i]);
  vonenorm := temp
end;


function vinfnorm(x: vector): extended;

  var
    i: integer;
    temp: extended;

begin
  temp := 0.0;
  for i := 1 to x^.len do
    if x^.xx[i] > temp then
      temp := x^.xx[i];
  vinfnorm := temp
end;


function vtwonorm(x: vector): extended;

  var
    i: integer;
    temp: extended;

begin
  temp := 0.0;
  for i := 1 to x^.len do
    temp := temp + sqr(x^.xx[i]);
  vtwonorm := sqrt(temp)
end;
```

```
procedure symmeig(vals: vector; vects, A: matrix);
  { Numerial Recipes in C, pp. 360-366. }

  const
    maxiters = 50;

  var
    i,j,ip,iq,n: integer;
    thresh,theta,tau,t,sum,s,h,g,c: extended;
    b,z: vector;
    tempA: matrix;
    done: boolean;

  procedure rotate(C: matrix; i,j,k,l: integer);

  begin
    g := C^.xx[i,j]; h := C^.xx[k,l];
    C^.xx[i,j] := g - s * (h + g * tau);
    C^.xx[k,l] := h + s * (g - h * tau);
  end;

begin
  if (A^.row <> A^.col) then begin
    writeln('jacobi:  matrix are not square');
    halt end
  else if not missymm(A) then begin
    writeln('jacobi:  matrix are not symmetric');
    halt end;
  new(tempA); mequal(tempA,A);
  n := tempA^.row;
  mident(vects,n);
  getdiag(vals,tempA);
  new(b); vequal(b,vals);
  new(z); vzero(z,n);
  sum := 0.0;
  for ip := 1 to pred(n) do
    for iq := succ(ip) to n do
      sum := sum + abs(tempA^.xx[ip,iq]);
  done := (sum = 0.0);
  i := 0;
  while (i <= maxiters) and (not done) do begin
    inc(i);
    if (i < 4) then
      thresh := 0.2 * sum / sqr(n)
    else
      thresh := 0.0;
    for ip := 1 to pred(n) do
      for iq := succ(ip) to n do begin
        g := 100.0 * abs(tempA^.xx[ip,iq]);
        if (i > 4) and ((abs(vals^.xx[ip]) + g) = abs(vals^.xx[ip])) and
        ((abs(vals^.xx[iq]) + g) = abs(vals^.xx[iq])) then
          tempA^.xx[ip,iq] := 0.0
        else if (abs(tempA^.xx[ip,iq]) > thresh) then begin
          h := vals^.xx[iq] - vals^.xx[ip];
          if ((abs(h) + g) = abs(h)) then
            t := tempA^.xx[ip,iq]/h
          else begin
            theta := 0.5 * h / tempA^.xx[ip,iq];
            t := 1.0 / (abs(theta) + sqrt(1.0 + sqr(theta)));
            if (theta < 0) then
              t := -t end;
          c := 1.0 / sqrt(1 + sqr(t)); s := t * c;
          tau := s / (1.0 + c); h := t * tempA^.xx[ip,iq];
          z^.xx[ip] := z^.xx[ip] - h;
          z^.xx[iq] := z^.xx[iq] + h;
          vals^.xx[ip] := vals^.xx[ip] - h;
          vals^.xx[iq] := vals^.xx[iq] + h;
          tempA^.xx[ip,iq] := 0.0;
          for j := 1 to pred(ip) do
            rotate(tempA,j,ip,j,iq);
          for j := succ(ip) to pred(iq) do
            rotate(tempA,ip,j,j,iq);
          for j := succ(iq) to n do
            rotate(tempA,ip,j,iq,j);
          for j := 1 to n do
            rotate(vects,j,ip,j,iq) end end;
    for ip := 1 to n do begin
      b^.xx[ip] := b^.xx[ip] + z^.xx[ip];
      vals^.xx[ip] := b^.xx[ip];
      z^.xx[ip] := 0.0 end;
```

```
    sum := 0.0;
    for ip := 1 to pred(n) do
      for iq := succ(ip) to n do
        sum := sum + abs(tempA^.xx[ip,iq]);
    done := (sum = 0.0) end;
  dispose(tempA); dispose(b); dispose(z);
  if (not done) then begin
    writeln('jacobi:  too many iterations');
    halt end
end;


procedure gensymmeig(vals: vector; vects, A, B: matrix);
  { Golub and Van Loan, pp. 469-470, with Jacobi's method and adjustments
  for an ill-conditioned B matrix. }

  var
    d: vector;
    C,Gi,Git,V: matrix;
    i: integer;

begin
  if (A^.row <> A^.col) or (B^.row <> B^.col) then begin
    writeln('geneig:  matrices are not square');
    halt end
  else if (A^.row <> B^.row) then begin
    writeln('geneig:  matrices are not the same size');
    halt end
  else if ((not missymm(A)) or (not missymm(B))) then begin
    writeln('geneig:  matrices are not symmetric');
    halt end;
  new(d); new(C); new(Gi); new(Git); new(V);
  symmeig(d,V,B);
  sorteig(d,V,d,V,true);
  for i := 1 to d^.len do
    if d^.xx[i] <= 0.0 then begin
      writeln('geneig:  second matrix is not positive definite');
      halt end
    else
      d^.xx[i] := 1.0/sqrt(d^.xx[i]);
  mvdiagprod(Git,V,d); mtrans(Gi,Git);
  mequal(C,A); mprod(C,C,Git,1.0); mprod(C,Gi,C,1.0);
  symmeig(vals,V,C);
  mprod(vects,Git,V,1.0);
  dispose(d); dispose(C); dispose(Gi); dispose(Git); dispose(V);
end;


procedure sorteig(vals: vector; vects: matrix;
                  oldvals: vector; oldvects: matrix; ascend: boolean);
  { Numerical Recipies in C, p. 366. }

  var
    i,j,k: integer;
    temp: extended;

begin
  if (vects^.row <> vects^.col) or (vals^.len <> vects^.row) then begin
    writeln('eigsort:  matrix and vector are the wrong sizes');
    halt end;
  if (oldvals <> vals) then
    vequal(vals,oldvals);
  if (oldvects <> vects) then
    mequal(vects,oldvects);
  for i := 1 to vals^.len do begin
    k := i;
    temp := vals^.xx[k];
    for j := succ(i) to vals^.len do
      if ascend then begin
        if (vals^.xx[j] < temp) then begin
          k := j;
          temp := vals^.xx[k] end end
      else begin
        if (vals^.xx[j] > temp) then begin
          k := j;
          temp := vals^.xx[k] end end;
    if (k <> i) then begin
      vals^.xx[k] := vals^.xx[i];
      vals^.xx[i] := temp;
      for j := 1 to vals^.len do begin
```

```
          temp := vects^.xx[j,i];
          vects^.xx[j,i] := vects^.xx[j,k];
          vects^.xx[j,k] := temp end end end
  end;

begin
  matvecttol := 1.0e-8;
end.
```

MISC.PAS

```
unit misc;
  { Copyright, June 1, 1993.  Brian E. Lindholm }

interface

  function caps(data: string): string;
  function signr(x: extended): extended;

implementation

  function caps(data: string): string;

    var
      i: integer;

  begin
    caps[0] := chr(length(data));
    for i := 1 to length(data) do
      caps[i] := upcase(data[i])
  end;

  function signr(x: extended): extended;

  begin
    if x = 0.0 then
      signr := 0.0
    else
      if x > 0.0 then
        signr := 1.0
      else
        signr := -1.0
  end;

begin
end.
```

Vita

Brian Eric Lindholm was born on November 9, 1967 in Jacksonville, Arkansas. He spent his childhood in Chattanooga, Tennessee and completed his high school education in Chesterfield County, Virginia. In September of 1986, he enrolled at Virginia Polytechnic Institute and State University, where in May of 1991 he received a Bachelor of Science degree in Mechanical Engineering. In August of 1991, he enrolled in the graduate program in Mechanical Engineering at Virginia Polytechnic Institute and State University. This thesis is part of the requirements for a Master of Science degree in Mechanical Engineering.

Brian E. Lindholm