1995

A compliant model for estimation and optimization of damping in vibrating structures

Thomas Jeffrey Thompson
Iowa State University

A compliant model for estimation and optimization of damping in vibrating structures

by

Thomas Jeffrey Thompson

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Mechanical Engineering
Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1995

DEDICATION

To Ellen, John, and ?

Blessed be the name of God for ever and ever,
to whom belong wisdom and might. Daniel 2:20

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Large space structures proposed for use as space stations, telescopes, or laser platforms require active control systems to suppress vibrations and maintain precise dimensions. The SPICE (Space Integrated Control Experiment) [3] is a scale model of such a structure. Active control strategies such as HAC-LAC (high-authority control, low-authority control) [33, 25, 1] used on such structures are most effective when passive damping is present in the structure to prevent instabilities in the cross-over and spill-over frequency range [32].

Such passive damping is incorporated in strut-built structures such as SPICE by means of inserting struts which introduce viscous (D-Strut) or viscoelastic (V-Strut) damping [32, 28]; these struts normally have high static stiffness, which limits damping effectiveness. It has been proposed [26] that inserting viscous dampers without static stiffness could produce more effective damping. This will be referred to as diagonal damping, since it would span diagonally across node pairs which are not connected by existing structural elements.

The purpose of this dissertation is to evaluate the ability of such diagonal damping to achieve optimal modal damping in several target modes using as few struts as possible. This is done by developing and implementing a procedure to estimate the damping coefficient produced by combinations of struts placed in various loca-

tions. This procedure, which assumes "compliant" models for each of the modes of a structure, is outlined below.

Each of the modes is represented by a unit lumped mass and a spring representing the modal stiffness. Any elements to be added to the structure are scaled and placed in the model parallel to the modal stiffness but in series with another stiffness inherent to the structure. The inclusion of a series stiffness is motivated by the fact that the increase in modal frequency due to the addition of a spring to an existing structure levels off as the added stiffness becomes infinite. The series stiffness helps account for variation in the mode shapes from those of the original structure.

Parameters within the compliant model may be obtained by performing a limited number of normal mode analyses (adding one spring element at a time to the finite element model of the structure being analyzed). In order to characterize the compliant stiffnesses for a structure with m closely-spaced modes, it is advantageous to incorporate all m modes into an m-dimensional "spherical" model in which each added element influences each mode from a certain orientation within the m-space of the spherical model.

Once the compliant model is characterized, spring stiffness, viscous damping, or viscoelastic damping may be inserted into the model in each desired location. The modal damping in the structure due to this set of added damping struts is then estimated by evaluating the simple expression for the loss factor of the elements in the compliant model, instead of performing expensive complex eigenvalue analyses on the original structure.

Use of the spherical compliant model for structural damping optimization eliminates the need to do numerous resolves of the complex eigenvalue problem for each

damping value and for each combination of damping strut locations. Optimization proceeds much more quickly using the simple expression for the loss factor of the compliant model.

The remainder of this work is outlined below. Chapter 2, Background, covers literature review, the specific structure to be studied, the properties of damping struts used in computation, other methods of estimating damping, and optimization methods. Chapter 3, Compliant Model of a Structural Mode, details the development and use of the compliant and spherical models. Chapter 4 discusses optimization and presents results of the optimizations performed on the SPICE structure. Finally, conclusions are presented in Chapter 5.

CHAPTER 2. BACKGROUND

2.1 The SPICE Structure

The structure analyzed in this dissertation was built as a testbed for the Air Force to be used to develop and test structural control technology. It is called SPICE (Space Integrated Control Experiment) [3].

A diagram of the structure itself is shown in Figure 2.1. It is a full-scale model of a "beam expander" structure which could be placed in orbit about the earth; it could be used to focus light. The hexagonal bulkhead portion is a tetrahedron truss containing numerous struts, made of composite material, joined at the ends by node balls. Figure 2.2 shows the numbers of the nodes in the bulkhead. Six petals on the truss represent mirrors. The tripod extends approximately 8 meters above the bulkhead. Eighteen proof mass actuators (PMA's) located throughout the structure provide for active control [33].

The control strategy to be used on the structure is known as high-authority control, low-authority control (HAC-LAC) [25]. High-authority control [33] combines signals from sensors throughout the structure to digitally compute a set of linear quadratic Gaussian feedback control signals to drive the PMA's. Low authority control [1, 25] uses velocity feedback control at the PMA's using collocated sensors to provide low-level dissipative damping in targeted modes. Similarly, the passive

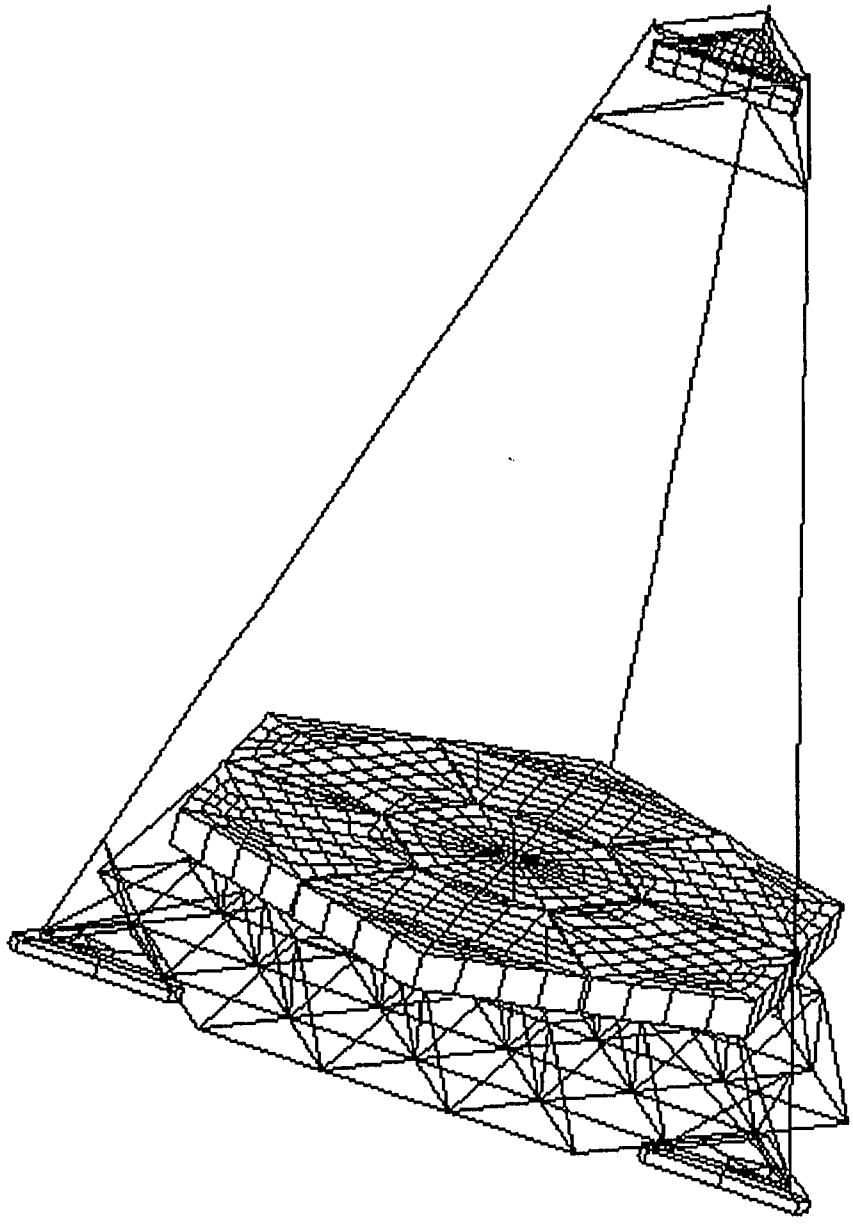

Figure 2.1: SPICE

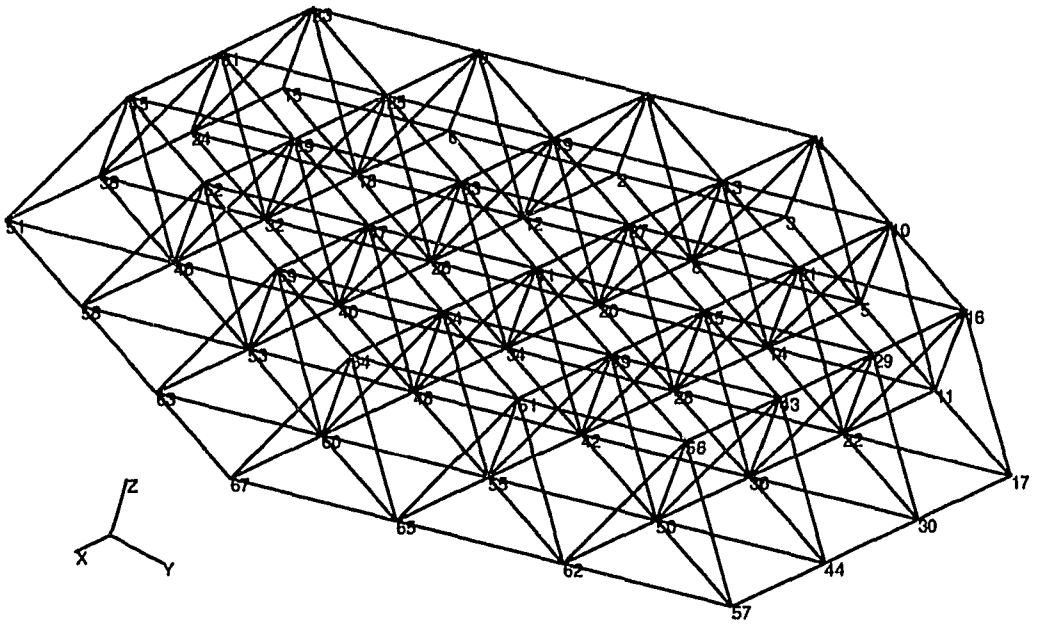

Figure 2.2: The SPICE Bulkhead Nodes

vibration control [33] proposed would use viscous or viscoelastic energy-dissipating struts in the bulkhead truss to achieve specified damping in several target modes and enhance control system stability in the loop cross-over and spill-over frequency ranges. Target modes for passive damping have been identified based on the active control system design; however, it has also been suggested that moderate passive damping (2%) in all modes within the control range would be very beneficial to the active control effort [4].

A NASTRAN finite element model of the SPICE bulkhead has been correlated to modal experimental data [22]. Because the model contains 384 degrees of freedom, ways of accurately simplifying the model produce large savings in computing modal damping.

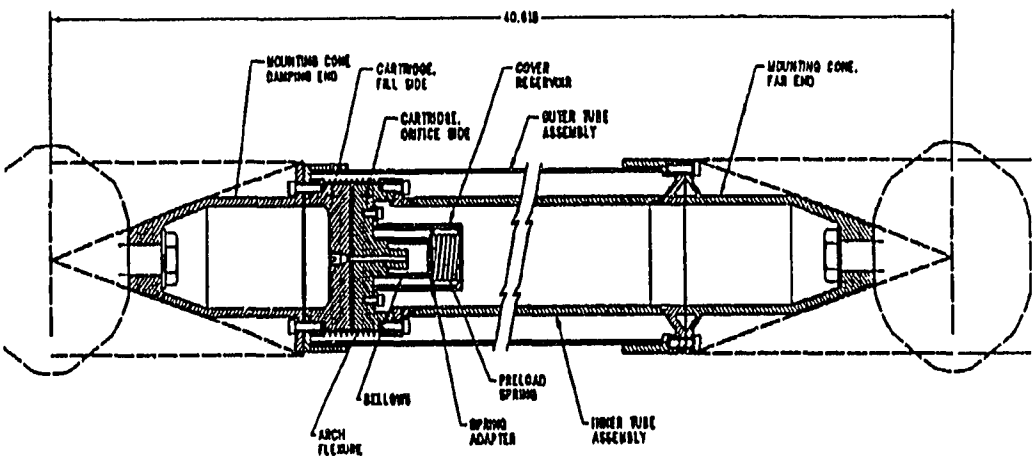

Figure 2.3: Diagram of a D-Strut

2.2 Loss Factor of a Damping Strut

2.2.1 Description of a Damping Strut

Various damping treatments have been used in damping structures such as SPICE. Constrained layer damping [15] is accomplished by laminating a structure with a viscoelastic material, which has a complex stiffness. Viscoelastic materials may also be incorporated into a cylindrical damping device, or V-Strut [32], which uses the complex modulus of the viscoelastic material to dampen relative motion between two structural nodes. The damping device studied in this dissertation achieves viscous damping between structural node points by forcing a liquid through a small orifice within its cylindrical body. It is called the D-Strut [7].

Figure 2.3 is a diagram of a D-Strut , and Figure 2.4 shows its lumped-parameter model. When relative axial motion between the two end nodes of the D-Strut occurs, force is transmitted through the inner rod to a flexure, which drives the liquid through the orifice, dissipating energy.

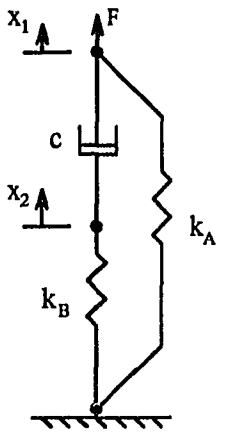

Figure 2.4: Model of a D-Strut

The lumped-parameter model comprises the dashpot placed in series with stiffness k_B and in parallel with stiffness k_A. The series stiffness must have some finite value not only because the inner rod of the D-Strut is made of some real material, but also because of compliance in the flexure and the bellows which contain the liquid. The outer shell of the D-Strut is represented by the parallel, or load-bearing stiffness k_A. If the rest of the structure is sufficiently strong, k_A could be eliminated, increasing damping effectiveness [7].

2.2.2 Definition of Loss Factor

Loss factor or loss coefficient is a measure of the proportion of the total strain energy being dissipated by a structural component under sinusoidal loading. It is defined by Thomson [29], p. 70, as the ratio of damping energy loss per radian divided by the peak strain energy.

For a component made of a material with complex modulus, the loss factor is

equal to the internal friction, which can be measured as the negative tangent of the phase angle between the stress and the strain in the component [10]. In analyzing damping struts in this dissertation, a similar method will be used. Beginning with the known Laplace transfer function relating force to displacement, a sinusoidal input ($s = i\omega$) is assumed. The loss factor is then taken to be the imaginary part over the real part of the frequency response function at frequency ω.

In the next section, the above approach is applied to the damping struts studied in this dissertation.

2.2.3 Modeling

The loss factor of the lumped-parameter model of a D-Strut will now be presented (see also [7, 27]). The transfer function relating force to displacement at the free end of the model in Figure 2.4 is

$$\frac{F}{X}(s) = k_A + \frac{k_B cs}{k_B + cs} . \tag{2.1}$$

Here s denotes the Laplace transform.

In order to simplify analysis, define the stiffness ratio

$$\kappa = \frac{k_B}{k_A} ,$$

the damper constant

$$\omega_B = \frac{k_B}{c} ,$$

the damper ratio

$$r = \frac{\omega}{\omega_B} = \frac{\omega c}{k_B} ,$$

and

$$j = \sqrt{-1}$$

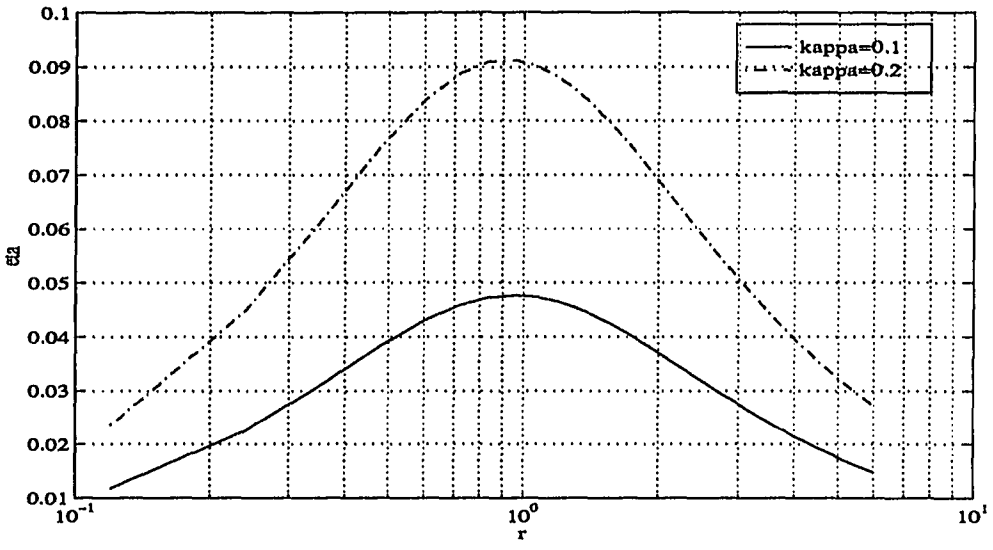

Figure 2.5: Shape of the Loss Factor Curve

where ω is the forcing frequency. It is apparent that r increases proportionally with either ω or c.

Substituting $s = j\omega$ into the transfer function results in

$$\frac{F}{X}(\omega) = k_A \left(1 + \kappa \frac{j\omega}{\omega_B + j\omega}\right) \tag{2.2}$$

$$\frac{F}{X}(r) = k_A \left(1 + \kappa \frac{r^2}{1+r^2} + \kappa \frac{jr}{1+r^2}\right) . \tag{2.3}$$

The loss factor, written as the ratio of the imaginary to the real part of Equation 2.3, is then

$$\eta(r) = \frac{\kappa \frac{r}{1+r^2}}{1 + \kappa \frac{r^2}{1+r^2}}. \tag{2.4}$$

The shape of this function is illustrated in Figure 2.5. At r (i.e. c or ω) $= 0$, the loss factor $\eta = 0$; as r is increased, η takes on some positive value which diminishes to zero as $r \rightarrow \infty$.

Setting the derivative of η with respect to r equal to zero results in the maximum loss factor

$$\eta^* = \frac{\kappa}{2\sqrt{1+\kappa}} \tag{2.5}$$

which occurs at

$$r^* = \frac{1}{\sqrt{1+\kappa}} \,. \tag{2.6}$$

Neither the optimal value of r nor the maximum loss factor attainable by a given strut depends upon the value of c, but upon κ, the ratio of the series stiffness k_B to the static stiffness k_A. This implies that the strut effectiveness increases as k_A decreases. The value of c may be specified by the designer to place the peak loss factor at a desirable frequency.

2.3 Diagonal Damping

Thompson and Baumgarten [26, 28] investigated the damping potential of diagonal dampers by analyzing their effect on an octahedral cell of a tetrahedron truss. A diagonal damper is defined as one which spans between two nodes of a truss which are not originally directly connected by a stiff element (strut). In a tetrahedron truss such as the SPICE bulkhead, potential locations for diagonal damping connect any two opposite corners of one of the octahedral cells which comprise the tetrahedron truss. Such a location is illustrated in Figure 2.6, which shows the cell Thompson and Baumgarten analyzed.

Thompson and Baumgarten reasoned that since the loss factor of a given damping strut is limited by the ratio of its series stiffness to its parallel stiffness, the placement of a spring-dashpot series in a diagonal location containing no parallel stiffness could result in a high loss factor. They used loss factor of an octahedral cell

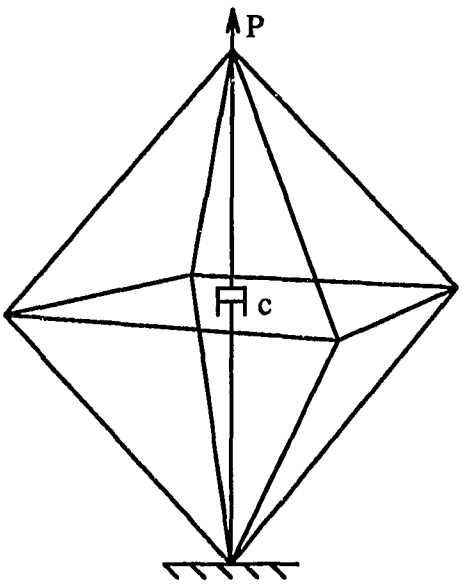

Figure 2.6: Octahedral Cell with Diagonal Damper

with harmonic excitation as a means of comparing damping potential of a diagonal location to that of in-line spring-dashpot series collocated with existing struts.

The analysis showed that for the orientation shown diagonal damping indeed provides higher loss factor for an octahedral cell than in-line damping. The authors hypothesized that diagonal damping would also be more effective in providing modal damping in large truss structures and suggested that optimization of the required damping criteria relative to damping locations is the true test of this hypothesis.

2.4 Modal Analysis and Damping

2.4.1 Normal Mode (Real Eigenvalue) Analysis

One of the keys to analyzing vibrating structures is identifying the undamped modes of vibration through real eigenvalue analysis [19, 29, 30, 34], described in this section. The equations of motion of the finite element model of a vibratory system can be expressed as

$$M\ddot{x} + C\dot{x} + Kx = f \tag{2.7}$$

where M, C, and K are the symmetrically square mass, damping, and stiffness matrices, x is an m-vector of displacements, and f is an m-vector of forces. The normal modes of the system are found by setting C and f equal to zero and assuming solutions of the homogeneous equation to be of the form

$$x = q_j(t)\phi_j = q_j(0)\phi_j e^{st} \tag{2.8}$$

where ϕ_j is a vector of displacements for the jth mode, and q_j is its generalized coordinate. Equation 2.7 then reduces to the form

$$K\phi_j = \lambda_j M\phi_j \ , \quad j = 1, 2, 3, \ldots m \tag{2.9}$$

where

$$\lambda_j = -s^2 \tag{2.10}$$

or

$$[\lambda_j M - K]\phi_j = \{0\} \ , \quad j = 1, 2, 3, \ldots m \ . \tag{2.11}$$

This is the non-standard form of eigenvalue problem. To get the eigenvalue problem into standard form, the transformation

$$\phi_j = Q^{-1}v \tag{2.12}$$

is introduced, where

$$Q^T Q = M \tag{2.13}$$

and v is an m-vector. This results in the standard form of the eigenvalue problem,

$$[I\lambda_j - A]v_j = \{0\} \tag{2.14}$$

in which

$$A = (Q^T)^{-1} K Q^T \ .$$

The eigenvalue problem may be solved by one of many algorithms using the Jacobi, Givens', Householder's [19], or Lanczos method, resulting in eigenvalues

$$\lambda_j \ , \quad j = 1, 2, 3, \ldots m$$

and eigenvectors,

$$v_j \ , \quad j = 1, 2, 3, \ldots m \ .$$

With the vectors v_j normalized to unit length,

$$v_i{}^T v_j = \delta_{ij} \equiv \begin{cases} 0 & \text{for } i = j \\ 1 & \text{for } i \neq j \end{cases} \ ,$$

the eigenvectors ϕ_j associated with the non-standard problem have the property

$$\phi_i{}^T M \phi_j = \phi_i{}^T Q^T Q \phi_j = \delta_{ij},$$

that is, they are mass-normalized, and

$$\phi_i{}^T K \phi_j = \lambda_j \phi_i{}^T M \phi_j = \lambda_j \delta_{ij} \ .$$

With the orthogonal vectors ϕ_j compiled into the square modal matrix

$$\Phi = [\ \phi_1 \quad \phi_2 \quad \cdots \quad \phi_m\ ] \ ,$$

the solutions to the eigenvalue problem obey

$$K\Phi = M\Phi\Lambda \ ,$$

where Λ is the diagonal matrix of eigenvalues. Furthermore, the position of the undamped system can be expressed as the summation of contributions from all of the modes,

$$x = \sum_{j=1}^{m} q_j \phi_j = \Phi q \tag{2.15}$$

where q is a vector of the generalized coordinates. Now the undamped, decoupled equations of motion can be expressed as

$$\Phi^T M \Phi \ddot{q} + \Phi^T K \Phi q = \Phi^T f \ , \tag{2.16}$$

or

$$\phi_j{}^T M \phi_j \ddot{q}_j + \phi^T{}_j K \phi_j q_j = \phi^T{}_j f \ . \tag{2.17}$$

2.4.2 Modal Damping and Complex Eigenvalue Analysis

2.4.2.1 Definition of Modal Damping The modal damping ratio of a structural mode is a measure of how quickly free vibrations in that structure diminish in magnitude. The definition of modal damping ratio is developed below.

A finite element or lumped-mass model of a freely vibrating structure can be written as

$$M\ddot{x} + C\dot{x} + Kx = [0] \tag{2.18}$$

where M is the mass matrix, C is the damping matrix, K is the stiffness matrix, and $x(t)$ is the vector of displacements. Dot notation represents differentiation with respect to time.

Without damping, $C = [0]$, and a real eigenvalue analysis results in a modal matrix Φ of eigenvectors ϕ_j and a set of eigenvalues λ_j which decouple the modes of the system into equations of the form [34] (see Section 2.4.1)

$$\phi_j^T M \phi_j \ddot{q}_j + \phi_j^T K \phi_j q_j = 0 \tag{2.19}$$

where q_j is the jth member of the vector q containing generalized coordinates, and

$$x = \Phi q \ . \tag{2.20}$$

If ϕ_j are mass-normalized,

$$\ddot{q}_j + {\omega_j}^2 q_j = 0 \tag{2.21}$$

and

$$\lambda_j = {\omega_j}^2 \ .$$

When C can be written as a linear combination of M and K, it is proportional (p. 197 of [29]), and the modal matrix Φ diagonalizes not only the M and K matrices, but also the C matrix, resulting in real equations of the form

$$\ddot{q}_j + 2\zeta_j \omega_j \dot{q}_j + {\omega_j}^2 q_j = 0 \tag{2.22}$$

where ζ_j is the equivalent viscous modal damping ratio. This can be computed from the complex eigenvalues of the above system by (p. 27, [29])

$$\zeta_j = -\frac{\Re(\lambda_j)}{|\lambda_j|} \ . \tag{2.23}$$

If $C \neq 0$ and is not proportional, then equation 2.18 leads to the eigenvalue problem [34]

$$({\tilde{\lambda}_j}^2 M + \tilde{\lambda}_j C + K)\tilde{\phi}_j = 0 \tag{2.24}$$

where $\tilde{\lambda}_j$, $\tilde{\phi}_j$, and $\tilde{q}_j$ are, in general, complex. The modal damping can then be computed as

$$\zeta_j = -\frac{\Re(\tilde{\lambda}_j)}{|\tilde{\lambda}_j|} . \tag{2.25}$$

Since local damping treatments such as those used in SPICE produce non-proportional damping, this work focuses on computing or estimating the value of ζ_j in equation 2.25.

2.4.2.2 Complex Eigenvalue Analysis of a Damped System This description of complex eigenvalue analysis amplifies that given in Section 2.4.2.1. Beginning with the homogeneous equations of motion of an underdamped $m \times m$ system

$$M\ddot{x} + C\dot{x} + Kx = [0] , \tag{2.26}$$

if a solution of the form

$$x = \tilde{\phi}_j q_j(0) e^{\tilde{\lambda}_j t} = \tilde{\phi}_j q_j(t)$$

is assumed, then

$$[\tilde{\lambda}_j^2 M + \tilde{\lambda}_j C + K]\tilde{\phi}_j = \{0\} . \tag{2.27}$$

The eigenvalues $\tilde{\lambda}_j$, eigenvectors $\tilde{\phi}_j$, and generalized coordinates q_j of the above system are all complex in general. The eigenvalues and eigenvectors may be computed by reducing Equation 2.27 to a first-order $2m \times 2m$ system of the form [8, 34]

$$[I\tilde{\lambda}_j - A]\psi_j = \{0\} . \tag{2.28}$$

The eigenvalues $\tilde{\lambda}_j$ of the reduced system occur as complex conjugate pairs, and are also the solutions of Equation 2.27. The eigenvectors also occur in complex conjugate

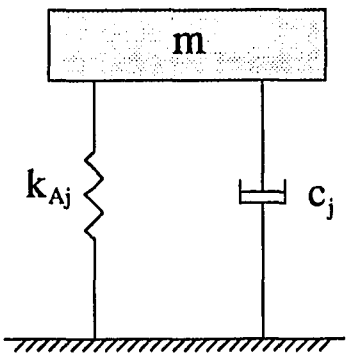

Figure 2.7: Model of a Damped Structural Mode

pairs and are related to the eigenvectors of second-order Equation 2.27 by expressions such as

$$\psi_j = \left\{ \begin{array}{c} \tilde{\phi}_j \\ \tilde{\lambda}_j \tilde{\phi}_j \end{array} \right\}, \quad j = 1, 2, 3 \ldots 2m \ .$$

The equivalent viscous modal damping ratio can then be computed from the location of the eigenvalue by Equation 2.25. In the special case of proportional damping [29], the equations of motion are uncoupled by the normal modes of the system, so that the values of $\tilde{\phi}_j$ are real.

2.4.3 Modal Loss Factor

In Section 2.2.2, it is stated that loss factor may be computed by dividing the ratio of the imaginary to real parts of the forced frequency response of a massless system. In this section, it is shown that a similar concept may be used to describe damping in structural modes.

The equation of a proportionally-damped structural mode (see Figure 2.7) is

$$m_j \ddot{q} + c_j \dot{q} + k_j q = f_j \ , \tag{2.29}$$

where m_j, c_j, and k_j are modal mass, damping, and stiffness. This may be expressed as

$$\ddot{q} + \zeta_j \omega_j \dot{q} + {\omega_j}^2 q = \frac{f_j}{m_j} .$$

where

$$\omega_j^2 = \frac{k_j}{m_j}$$

and

$$\zeta_j = \frac{c_j}{2\sqrt{m_j k_j}} .$$

Taking the Laplace transform of Equation 2.29 and dividing through by m_j, an impedance $Z(s)$ [13] may be defined by

$$Z(s) \equiv s^2 + \frac{c_j}{m_j} s + \frac{k_j}{m_j} = \frac{f_j}{m_j q}(s) . \tag{2.30}$$

Leaving on the right only the terms of the impedance related to stiffness and damping,

$$m_j(Z(s) - s^2) = c_j s + k_j . \tag{2.31}$$

With sinusoidal excitation, $s = j\omega$. Then the viscous damping ζ_j is related to an equivalent structural damping factor (loss factor) η_j by

$$m_j(Z(s) + \omega^2) = j c_j \omega + k_j = k_j(1 + j\eta_j(\omega)) , \tag{2.32}$$

where the loss factor is the following ratio of the imaginary to real parts of the complex stiffness

$$\eta_j = \frac{c_j \omega}{k_j} = \frac{2\zeta_j \omega_j \omega}{\omega_j^2} . \tag{2.33}$$

At resonance, the modal loss factor is twice the modal damping ratio [30, 29, 14],

$$\eta_j(\omega_j) = 2\zeta_j . \tag{2.34}$$

Indeed, Johnson and Keinholtz [15] utilize the modal loss factor in writing the equations of motion of damped modes of a system in the form

$$\ddot{q}_j + \eta_j \omega_j \dot{q}_j + {\omega_j}^2 q_j = \frac{f_j}{m_j} ,$$

where η_j is the loss factor for the jth mode. In this dissertation, modal damping will be estimated by considering each mode to be a two-degree-of-freedom compliant model and computing the loss factor for that simplified model.

2.5 Methods of Computing or Estimating Modal Damping

2.5.1 Direct Complex Eigenvalue Analysis

The modal damping ratio ζ_j of Equation 2.25 can be computed directly by means of complex eigenvalue analysis. This is the method used by MSC/NASTRAN [12]. First, complex eigenvalues of a damped structure are computed by one of a variety of methods, such as the complex Lanczos, inverse power, or determinant method. Then an approximation of Equation 2.25 is used to compute ζ_j as

$$\eta_j = 2\zeta_j \approx -\frac{2\Re(\tilde{\lambda}_j)}{|\Im(\tilde{\lambda}_j)|} . \tag{2.35}$$

However, this method is computationally expensive. While normal mode analysis requires eigenvalue solution of an $m \times m$ symmetric system with real roots, direct complex eigenvalue analysis involves solution of a $2m \times 2m$ asymmetric system with complex roots. Because of this expense, various methods of estimating modal damping ratios have been used. These will now be reviewed.

2.5.2 Reduced Vector Subspace

One method of computing the complex eigenvalues of a damped system is to use a subset of all of the normal modes of the undamped structure as a "reduced basis" for solving the complex eigenvalue problem. It is hoped that a very few of the lowest-frequency modes will adequately represent the new mode shapes of the damped structure so that accurate complex eigenvalues may be computed. Yiu and Weston examine some problems with this method along with improvements which overcome these problems [34].

In this paper, Yiu and Weston show by example that this method alone may inaccurately predict modal damping ratio for a structure which is damped locally by a damper represented by a viscous element and a spring element in series. The normal modes of the structure are inadequate to represent the modal displacement of each damper's interior node, which is heavily influenced by the viscous element.

Yiu and Weston then describe the use of constraint and constrained normal subspace to better describe the modes of the undamped system and to accurately compute the complex eigenvalues. Instead of using the normal modes of the original structure, they constrain it against axial displacements across the dashpot locations and compute constrained normal modes $\hat{\Phi}$ and eigenvalues $\hat{\Lambda}$. To this subspace are added for each dashpot two static vectors B to represent the motion of the dashpots. The solution vector u is computed in the subspace

$$u = \hat{\Phi}\hat{q} + Ba \tag{2.36}$$

where u represents the vector of displacements, $\hat{q}$ represents the vector of generalized coordinates for $\hat{\Phi}$, and a represents the coordinates of the added static vectors.

If n dashpots and a subset of m modes are being considered, this results in matrices of the size $(m+2n) \times (m+2n)$. Examples show this method to be accurate.

The use of constraint subspace is significant to the work contained in this dissertation in that information regarding how the modified structure would behave is built into the reduced subspace by performing a real eigenvalue analysis on a modified structure.

2.5.3 Rayleigh Damping

Rayleigh or proportional damping exists if the damping matrix C can be diagonalized by the same modal matrix which diagonalizes the M and K matrices [24, 14]. This is true if and only if [5]

$$CM^{-1}K = KM^{-1}C \ , \tag{2.37}$$

and is true in the case that C can be expressed as a linear combination of M and K. Then the equations of motion

$$\Phi^T M \Phi \ddot{q}_j + \Phi^T C \Phi \dot{q}_j + \Phi^T K \Phi q_j = 0 \tag{2.38}$$

can be expressed as uncoupled equations of the form

$$\ddot{q}_j + 2\zeta_j \omega_j \dot{q}_j + {\omega_j}^2 q_j = 0 \ . \tag{2.39}$$

The modal damping ratio for mode j is

$$\zeta_j = \frac{{\phi_j}^T C \phi_j}{2\omega_j} \ . \tag{2.40}$$

When localized damping is applied to a structure, the C matrix is not generally diagonalized by the normal modes of the undamped system. As Rayleigh wrote (pp. 199-200 of [24]),

> The law of friction assumed in the preceding investigation is the only one whose results can be easily followed deductively, and it is sufficient to give a general idea of the effects of dissipative forces on the motion of a string. But in other respects the conclusions drawn from it possess a fictitious simplicity, depending on the fact that F—the dissipation-function—is similar in form to T, which makes the normal co-ordinates independent of each other. In almost any other case (for example, when but a single point of the string is retarded by friction) there are no normal co-ordinates properly so called. ... Special cases excepted, no linear transformation of the co-ordinates (with real coefficients) can reduce T, F, and V together to a sum of squares.

(In the above quotation, T represents kinetic energy, V potential energy, and F the energy dissipation rate.) However, in the case of non-proportional damping, the diagonal terms of $\Phi^T C \Phi$ may still be used to approximate the modal damping ratios [15]. This is the basis for the method described in the next section.

2.5.4 Modal Strain Energy Method

Johnson and Keinholtz [15] describe a method of computing a modal loss factor based on summing over all components the loss factor of a component multiplied by its proportion of the total undamped modal strain energy. This is called the modal strain energy method.

At each of the resonant frequencies j of an undamped structure, each of the damping components (viscous or viscoelastic struts or laminates) i would possess a loss factor η_{ij}. The loss factor is a measure of the proportion of its strain energy which is dissipated each cycle. For mode j, the real part of the stiffness of this component

possesses the fraction

$$\frac{V_{ij}}{V_j}$$

of the total strain energy for mode j. The loss factor for a given damped mode is then estimated

$$\eta_j = \sum_{i=1}^{n} \left(\frac{V_{ij}}{V_j} \right) \eta_{ij} \ . \tag{2.41}$$

Then it is assumed that the damping term based on this loss factor can be inserted into the uncoupled modal equations of motion so that

$$\ddot{q}_j + \eta_j \omega_j \dot{q}_j + {\omega_j}^2 q_j = \ell_j(t) \tag{2.42}$$

and

$$u = \Phi q_j(t) \ . \tag{2.43}$$

The loss factor in this equation is related to the modal damping ratio in Equation 2.22 by $\eta_j \approx 2\zeta_j$.

The implicit assumption is that the damped modes can be adequately represented by the normal modes of the undamped structure, that is, that the damping matrix can be diagonalized by the same matrix Φ which diagonalizes K and M.

The advantage of this method is that it only requires the normal modes analysis of the original structure to arrive at good estimates of modal damping achieved by applied damping treatments. The disadvantage stated is that approximation is required to accommodate frequency-dependent material properties. Also, there is no way to assign a strain energy to a component, such as a viscous element in series with a spring element, which is to be placed in a location where the original structure possesses no strained element.

2.5.5 Perturbation Analysis

Perturbation analysis is a method by which Taylor series are used to approximate the value of a function near a point where its value is known [14, 17, 19]. In the case of eigenvalue analysis of a structure, perturbation analysis is used to estimate the eigenvalues and eigenvectors as functions of design changes, or perturbations ε, in a structure for which normal modes are known.

The equations of motion of a structure are expressed as

$$\left[I{s_j}^2 + A\right]\phi_j = [0] \tag{2.44}$$

or

$$[I\lambda_j - A]\,\phi_j = [0] \tag{2.45}$$

where $A = M^{-1}K$. The shift in the eigenvalue λ_j is estimated by

$$\lambda_j(\varepsilon_{ij}) = \lambda_j(0) + \varepsilon_{ij}{\lambda_j}^{(1)} + {\varepsilon_{ij}}^2{\lambda_j}^{(2)} + \ldots \tag{2.46}$$

where ${\lambda_j}^{(k)}$ are functions of the kth derivative of the eigenvalue [14]

$${\lambda_j}^{(k)} = \left(\frac{1}{k!}\right)\left[\frac{d^k\lambda_j}{d{\varepsilon_{ij}}^k}\right]\Bigg|_{\varepsilon=0}. \tag{2.47}$$

If the matrix A is perturbed according to

$$A_p = A + \varepsilon_{ij}B_i \tag{2.48}$$

where B_i is the matrix of changes, then

$${\lambda_j}^{(1)} = {\phi_j}^T B_i \phi_j \tag{2.49}$$

and the eigenvalue may be estimated by

$$\lambda_j(\varepsilon_{ij}) \approx \lambda_j(0) + \varepsilon_{ij}{\phi_j}^T B_i \phi_j\ . \tag{2.50}$$

This may be interpreted as the influence of the new element on the modes of the system assuming the eigenvectors remain unchanged. Higher-order perturbation models are useful, but will not be addressed in this dissertation.

If the change to the structure is a spring-damper series c_i—k_i, then

$$\varepsilon_{ij} = \frac{c_i s_j k_i}{c_i s_j + k_i} \ . \tag{2.51}$$

The complex root is then approximated as

$$s_j(\varepsilon_{ij}) \approx s_j(0) + \varepsilon_{ij} s_j{}' \tag{2.52}$$

where

$$s_j{}' = -\frac{\phi_j{}^T B_i \phi_j}{2 s_j \phi_j{}^T \phi_j} \ . \tag{2.53}$$

The square B_i matrix contains only zeroes except for ± 1's according to the locations of the added elements. The $2s_j$ in the denominator arises because the perturbation is on the function s_j, not $\lambda_j = -s_j{}^2$ at $\varepsilon_{ij} = 0$.

For low levels of damping, $s_j \approx i\omega_j$. Setting

$$\kappa_p = k_i \frac{\phi_j{}^T B_i \phi_j}{\omega_j{}^2 \phi_j{}^T \phi_j} \tag{2.54}$$

and

$$r_{ij} = \frac{\omega_j c_i}{k_i} \tag{2.55}$$

results in

$$s_j \approx \omega_j \left[i + \frac{\kappa_p}{2} \left(\frac{-r_{ij} + i r_{ij}{}^2}{r_{ij}{}^2 + 1} \right) \right] \ . \tag{2.56}$$

Also for low damping, the damping ratio is approximately

$$\zeta_j \approx -\frac{\Re(s_j)}{\Im(s_j)} = \frac{\frac{1}{2}\kappa_p \frac{r_{ij}}{r_{ij}{}^2+1}}{1 + \frac{1}{2}\kappa_p \frac{r_{ij}{}^2}{r_{ij}{}^2+1}} \ . \tag{2.57}$$

When multiple changes are made to the original system, the resulting eigenvalues may be estimated by a summation of the first-order effects of all the elements on the unperturbed system. This is done for an example structure in Section 3.6.2.6.

2.6 Methods of Optimizing Damping

2.6.1 Simulated Annealing

Chen, Bruno, and Salama [16] use a simulated annealing technique to seek optimal actuator placement for active and passive structural control. This involves choosing new candidate solutions at random and accepting improving along with random non-improving solutions according to a diminishing probability.

First, they define an optimization function based on energy dissipation by passive and active elements during a finite time interval from a set of given initial conditions. The energy dissipation function is a weighted summation of the dissipation due to active and passive components. The passive energy dissipation approximation is based upon the modal strain energy method [15], and the energy dissipation due to active members is based on a constant-output velocity feedback control law.

The search for the optimal combination proceeds from the initial candidate solution as follows. At random, one of the current control locations is replaced by one of the remaining locations, and the optimization function is computed for the new candidate solution. If the function evaluation has improved, the new candidate solution is accepted. If it has not improved, the new candidate solution is only occasionally accepted according to a modified Boltzmann probability function

$$P = \exp \frac{\Delta E_d}{\theta} \tag{2.58}$$

where ΔE_d is the change in the energy dissipation function evaluation, and θ is a pseudotemperature, which decreases as the search progresses. This results in the acceptance of fewer and fewer non-improving solutions as the algorithm approaches a near-optimal solution.

Chen, Bruno, and Salama apply this algorithm to two example structures: a 150-member tetrahedron truss and a cantilever boom. Since the truss contained more member locations, various initial solutions resulted in distinct but similar sub-optimal solutions. Certain locations appeared in most cases. The algorithm repeatedly converges to the globally optimal solution in the case of the simpler cantilever boom problem.

Because the near-optimal solution to the tetrahedron truss problem requires about the same number of iterations as the solution to the cantilever boom problem, the simulated annealing approach appears to be particularly effective in attacking combinatorial optimization problems (such as that of SPICE) containing large numbers of combinations. In most cases the 150-member tetrahedron truss problem required between 300 and 500 iterations to converge.

2.6.2 Search by Golden Section

Search by golden section [31] is a non-gradient technique for finding a local maximum (or minimum) of a bounded, one-dimensional function. With this method, the region is sectioned and reduced by approximately 38.2% after each function evaluation. It is useful for finding the maximum of a performance function based on loss factors for a structure containing one or several added dampers with the same damping coefficient.

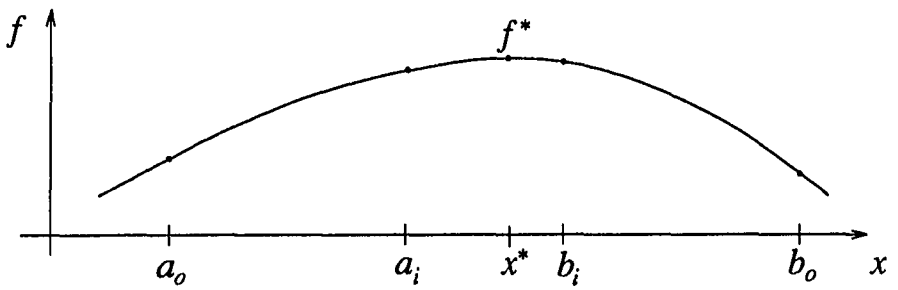

Figure 2.8: Region for Search by Golden Section

The method starts with a bounded region of a function which contains a unimodal optimum f^*. A performance function comprising the summation of modal loss factors of a damped structure often fits this pattern. The value of the performance function is to be optimized which respect to damping coefficient c (same for all the dampers). The region is shown in Figure 2.8, in which $a_i - a_o = b_o - b_i = 0.381966(b_o - a_o)$. Beginning with the function evaluation at the points a_o, a_i, b_i, and b_o shown, the portion of the region beyond the interior point with the "worse" function evaluation is discarded, and that point becomes a new exterior point. A function evaluation is then performed at a new interior point, which is located such that the spacing of the interior points and exterior points is again that of Figure 2.8. This sectioning process continues until convergence, which occurs when the width of the region or the change in each function evaluation is sufficiently small.

2.6.3 Sensitivity of Resonant Response

Li [18] develops methods to minimize peak structural response to harmonic excitation. He uses sensitivity information to arrive at directions of search within the design space, and also to predict the most effective locations for placement of damp-

ing.

Li first shows that when a design parameter of a system is modified, both the magnitude and frequency of the system's peak responses are changed. Slopes of the changes in these peaks yield information regarding the sensitivity of the system with respect to various design modifications. Using first- and second-order curve fits, he then predicts peak response for relatively large design changes. The sensitivities with respect to various design parameters, such as mass, stiffness, and damping values, are used to arrive at a steepest-descent direction of search, reducing the optimization problem to a sub-problem containing only one variable.

Secondly, Li extends the work of Mikaili [20] by placing Near Zero Section (NZS) dampers on a plate and measuring the sensitivity of the peak response with respect to each damper location. The locations possessing the largest sensitivity reduced peak response of the plate most effectively, as predicted. The work of this dissertation is similar to Li's use of NZS dampers in that small elements are used to predict damping effectiveness of larger elements. However, the work of this dissertation differs from Li's in that:

1. Instead of frequency response analysis of Li, real eigenvalue analysis is used, and modal damping is predicted instead of peak response.

2. Near zero section *springs* are used instead of NZS dampers to predict the effect of larger *dampers* on the system.

3. By placing one larger spring at a time in each prospective damper location and performing an additional real eigenvalue analysis for each added spring, one obtains enough information to quantitatively predict modal damping over

a large range of damping rate.

2.6.4 Optimization of Energy Dissipation Rate

Neubert [23] presents a method of evaluating locations for damping within a truss based on maximizing the rate of energy dissipation for a given set of initial conditions. He starts by premultiplying the matrix equations of motion for a system by a rate vector and isolating a Hamiltonian energy function which diminishes according to the rate of energy dissipation by damping within the structure.

Neubert identifies this rate as the Rayleigh energy dissipation function,

$$2D = \dot{u}^T C \dot{u} = \dot{q}^T \Phi^T C \Phi \dot{q} = q^T \Lambda \Phi^T C \Phi \Lambda q \quad . \tag{2.59}$$

He then solves the optimization problem of maximizing the Rayleigh energy dissipation function with respect to damping values c_i of each structural element. The derivatives of this function with respect to each of the damping values aid in this optimization and provide insight to the most valuable positions in which to place damping.

The optimization Neubert describes is dependent upon a certain set of initial conditions while that of this dissertation is directed toward achieving optimal modal damping ratios without regard to any given set of initial conditions.

2.6.5 Minimization of Error Integral

Gilheany [11] presents a method of determining optimal placement and magnitude of dampers in a lumped-parameter system. He uses as an optimization function the time integral of the weighted squares of the displacement and velocity errors resulting from the numerical integration of the matrix equations of motion. In this way,

a solution of the eigenvalue problem is completely avoided, but the optimum depends upon the initial conditions or forcing function chosen. For a given combination of damper locations, a Hooke-Jeeves pattern search is used to find the optimal damper magnitudes, and the best combination is found by an exhaustive search.

Some advantages of this procedure are that no eigenvalue solution is necessary and that nonlinearities and additional constraints are not a problem. The disadvantage is that the exhaustive search would become extremely time-consuming for a structure such as SPICE, which possesses many candidate damper locations.

2.6.6 Combined Control-Structure Optimal Design

Milman, Salama, Sheid, Bruno, and Gibson [21] use an approach which optimizes a combination of control and structural criteria for structural design. They show that such a combined optimization is never inferior to the conventional sequential design procedure, in which the structure is optimized before the control. The combined optimization is achieved by introducing a homotopy parameter which, as it varies between 0 and 1, effectively weights the contributions of control criteria and of structural criteria to the overall optimization criterion. In this way, a family of solutions which is useful in early design trade studies is produced.

The structural criterion used is minimization of mass while the control criterion is minimization of a linear quadratic Gaussian or regulator performance index. Local methods such as the Newton method can be used to expedite optimizations along the homotopy path. Examples show that the approach quantifies the trade-offs between control and structural factors in design. However, in examples, actuator placement is predetermined. The models of the present dissertation could be used to simplify

and speed a combined control-structure optimization, which would become quite computationally expensive for a complex structure.

CHAPTER 3. COMPLIANT MODEL OF A STRUCTURAL MODE

In this dissertation, a two-degree-of-freedom model will be assumed for each mode. This model results in simple expressions which accurately estimate the effects of added localized stiffness or damping on modal frequency or damping. The model will be referred to as the compliant model of a structural mode.

3.1 Development of the Compliant Model

3.1.1 Description of the Compliant Model

If one inserts a damper into an arbitrary location in a large structure (Figure 3.1) and looks at the resulting damping ratio as the damping c_i of the strut increases from zero to infinity, one might expect the damping ratio of a given mode to start at zero, increase to some point, and then "level off." One might also expect the damping ratio to decrease and again approach zero as the value of c_i approaches infinity. This provides the intuitive basis for modeling each mode of a structure as a compliant model.

The compliant model of a damped structural mode, similar to the generalized Maxwell model of a viscoelastic solid [2], is shown in Figure 3.2. In this figure, k_{Aj} represents the modal stiffness of undamped mode j; with a mass-normalized system, $m = 1$, and $k_{Aj} = {\omega_j}^2$. Any parallel stiffness k_p in the damper must be included in

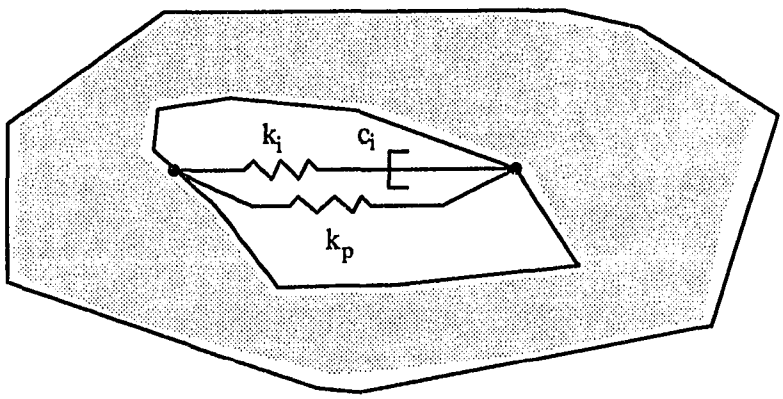

Figure 3.1: Model of a Damper Inserted into a Structure

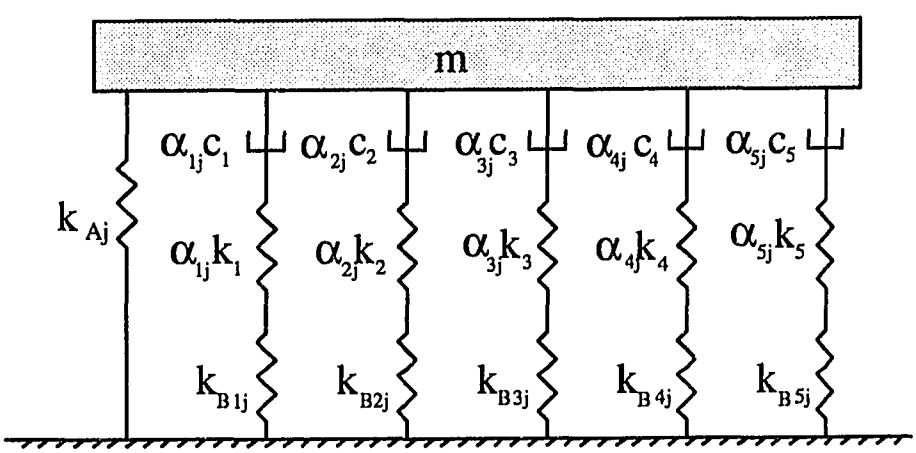

Figure 3.2: Compliant Model of a Structural Mode

k_{Aj}. The symbol k_{Bij} accounts for supposed compliances within the structure which influence the way the series components of the ith D-Strut, c_i and k_i, affect the modal damping ratio. The α_{ij} is a coefficient which "scales" the values of c_i and k_i for each mode j and each strut i.

In this model, k_{Aj}, k_{Bij}, and α_{ij} must be determined for each mode and strut location. Then k_i and c_i can be set to optimize a performance index using damping ratios estimated through the modal loss factor of the model.

3.1.2 Modal Loss Factor of the Compliant Model of a Structure

This section parallels Section 2.2.3, in which the loss factor of a damping strut was developed. In order to find a complex stiffness for the compliant model shown in Figure 3.2, a transfer function which describes the "vertical" force F_j on mass m_j necessary to achieve vertical displacement Q_j is written. Keeping only the stiffness and damping terms on the right,

$$\frac{F_j}{Q_j}(s) - m_j s^2 = k_{Aj} + \sum_{i=1}^{n} \frac{k_{sij}\alpha_{ij}c_i s}{k_{sij} + \alpha_{ij}c_i s} \tag{3.1}$$

where n is the number of struts. The substitutions

$$\kappa_{sij} = \frac{k_{sij}}{k_{A_j}} \tag{3.2}$$

$$\omega_{Bij} = \frac{k_{sij}}{\alpha_{ij}c_i} \tag{3.3}$$

$$r_{ij} = \frac{\omega}{\omega_{Bij}} \tag{3.4}$$

$$s = j\omega \tag{3.5}$$

result in

$$\frac{F_j}{Q_j}(\omega) + m_j\omega^2 = k_{Aj}\left(1 + \sum_{i=1}^{n} \kappa_{sij}\frac{j\omega}{\omega_{Bij} + j\omega}\right) \tag{3.6}$$

or

$$\frac{F_j}{Q_j}(r_{ij}) + m_j\omega^2 = k_{Aj}\left\{1 + \sum_{i=1}^{n}\left(\kappa_{sij}\frac{r_{ij}^2}{1 + {r_{ij}}^2} + \kappa_{sij}\frac{jr_{ij}}{1 + {r_{ij}}^2}\right)\right\} . \tag{3.7}$$

Taking modal loss factor to be the ratio of the real to imaginary parts of this complex stiffness,

$$\eta_j(r_{ij}) = \frac{\sum_{i=1}^{n} \kappa_{sij}\frac{r_{ij}}{1+{r_{ij}}^2}}{1 + \sum_{i=1}^{n} \kappa_{sij}\frac{{r_{ij}}^2}{1+{r_{ij}}^2}} . \tag{3.8}$$

3.2 Methods of Finding the Properties of the Compliant Structural Model

3.2.1 Method of Infinite Stiffness

In order to find the values of k_{Aj}, k_{Bij}, and α_{ij}, for a given mode j and strut location i, the following procedure can be used. First, a real eigenvalue analysis is performed on the structure alone without any added spring elements in parallel to damper locations included. Mass normalization is used, so that for mode j,

$$k_{Aj} = {\omega_{Aj}}^2 . \tag{3.9}$$

The value of k_{Bij} is found by placing a spring of infinite stiffness into the damper location, as Figure 3.3 shows. Assuming that the modal mass does not change, the value of k_{Bij} is determined by assuming that k_{Aj} and k_{Bij} act in parallel. With ${\omega_{Bij}}^2$ from the resulting real eigenvalue analysis,

$$k_{Bij} = {\omega_{Bij}}^2 - k_{Aj} . \tag{3.10}$$

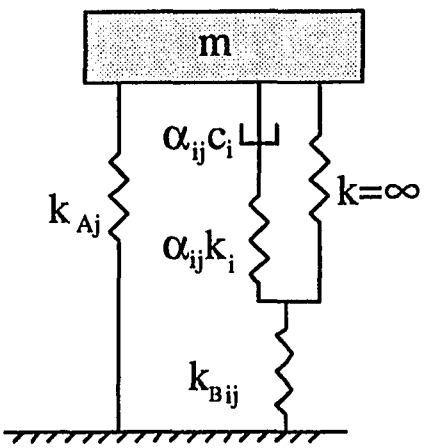

Figure 3.3: Model for Finding k_{Bij}

Next, the value of α_{ij} must be found. A spring of stiffness identical to that of k_i is placed across the damper location. This is equivalent to placing an infinitely-stiff spring in parallel with damper c_i. A third real eigenvalue analysis is done. Using the series-parallel properties of the springs in the model, a new spring equivalent to the combination of k_i and k_{Bij} is found.

$$\omega_{sij}{}^2 - k_{Aj} = k_{sij} = \frac{\alpha_{ij}k_i k_{Bij}}{\alpha_{ij}k_i + k_{Bij}} \tag{3.11}$$

With k_i having been selected and k_{Bij} and k_{sij} known,

$$\alpha_{ij} = \frac{1}{k_i}\frac{k_{sij}k_{Bij}}{(k_{Bij} - k_{sij})} \, . \tag{3.12}$$

Performing two more real eigenvalue analyses per strut and repeating the same computations completely determine the compliant model of the structure for any mode being analyzed. The values of k_{Aj}, k_{Bij}, and α_{ij} of the structure along with specified k_i provide enough information to compute κ_{sij}.

3.2.2 Method of Finite Stiffnesses

A second method of finding the values κ_{sij} and α_{ij} for the compliant structural model is to analyze the shifts in frequencies caused by individually inserting two massless springs of known stiffness.

The parallel stiffness of the mode, k_{Aj}, is given by Equation 3.9. Insertion of the value of the known strut stiffness k_i and computation of the frequencies determine k_{sij} (and so κ_{sij}) by Equation 3.11. The value of α_{ij} is found by repeating the frequency analysis with a massless spring of another known stiffness, $k_{di} = d \cdot k_i, d > 0$. If ${\omega_{sdi}}^2$ is the resulting eigenvalue, then

$$k_{sdi} = {\omega_{sdi}}^2 - k_{Aj} \ . \tag{3.13}$$

Then

$$\alpha_{ij} = \frac{d-1}{d \cdot k_i}\left(\frac{k_{dsi}k_{si}}{k_{si} - k_{dsi}}\right) \ . \tag{3.14}$$

To match the dynamic stiffness of the damper near optimal damping, d may be set to 1/2. Then

$$\alpha_{ij} = \frac{1}{k_i}\left(\frac{k_{dsi}k_{si}}{k_{si} - k_{dsi}}\right) \ . \tag{3.15}$$

The value k_{Bij} does not need to be found unless one desires to vary k_i in applying the model, in which case k_{Bij} is evaluated by rearranging Equation 3.11.

Like the method of infinite stiffnesses, this method requires $2n+1$ real eigenvalue analyses, where n is the number of strut locations being analyzed.

3.2.3 Method of Frequency Slopes

The preceding methods require two eigenvalue analyses for each damper location being considered. If more information can be obtained from the original eigenvalue

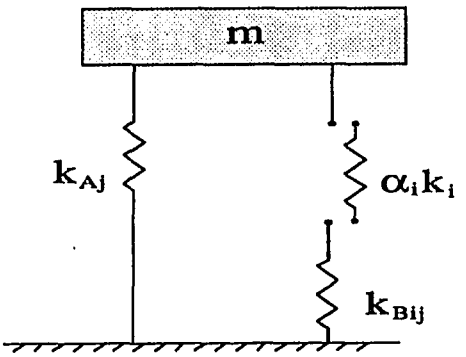

Figure 3.4: Compliant Model of Spring k_i Included in Structure

analysis, then fewer additional eigenvalue analyses are necessary. The method of frequency slopes provides from the original eigenvalue analysis information about the parameters of the compliant model by matching its derivative with respect to an added stiffness with that of the finite element model of the structure.

If a spring is placed in a position i of a structure, as pictured in Figure 3.4, the compliant model predicts the shifted frequency of the system will be

$$\omega_i{}^2 = k_{Aj} + \frac{\alpha_{ij} k_i k_{Bij}}{\alpha_{ij} k_i + k_{Bij}} \; . \tag{3.16}$$

The derivative of this frequency with respect to the added stiffness when $k_i = 0$ is

$$\left.\frac{\partial(\omega_i^2)}{\partial k_i}\right|_{k_i=0} = \left.\frac{\alpha_{ij} k_{Bij}(\alpha_{ij} k_i + k_{Bij}) - \alpha_{ij}(\alpha_{ij} k_i k_{Bij})}{(\alpha_{ij} k_i + k_{Bij})^2}\right|_{k_i=0} \tag{3.17}$$

$$= \left.\frac{\alpha_{ij} {k_{Bij}}^2}{(\alpha_{ij} k_i + k_{Bij})^2}\right|_{k_i=0} \tag{3.18}$$

$$= \alpha_{ij} \; . \tag{3.19}$$

This means that α_{ij} can be easily found if the eigenvalue derivative can be extracted from the modal analysis of the original structure.

The eigenvalue derivative is found as follows [9]. The eigenvalue problem for an undamped structure can be written as

$$[K - M\omega_j{}^2]\phi_j = \{0\} \ , \tag{3.20}$$

where $\omega_j{}^2$ and ϕ_j are the ith eigenvalue and eigenvector. Differentiating with respect to a structural change, such as an added strut, one obtains

$$[K' - M'\omega_j{}^2 - M\omega_j{}^2]\phi_j + [K - M\omega_j{}^2]\phi_j{}' = \{0\} \tag{3.21}$$

Premultiplying by $\phi_j{}^T$ and realizing $\phi_j{}^T[K - M\omega_j{}^2] = \{0\}^T$ due to symmetry,

$$\phi_j{}^T[K' - M'\omega_j{}^2 - M(\omega_j{}^2)']\phi_j = 0 \tag{3.22}$$

or

$$(\omega_j{}^2)' = \frac{\phi_j{}^T[K' - M'\omega_j{}^2]\phi_j}{\phi_j{}^T M\phi_j} \ . \tag{3.23}$$

With mass normalization, $\phi_j{}^T M\phi_j = 1$, so that by Equations 3.19 and 3.23

$$\alpha_{ij} = (\omega_j{}^2)' = \phi_j{}^T K'\phi_j \ . \tag{3.24}$$

Along with being the eigenvalue derivative, α_{ij} can also be interpreted as being the square of the axial displacement between the two nodes where the strut is to be added. For example if

$$K' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.25}$$

then

$$\alpha_{ij} = \begin{bmatrix} \phi_{1j} & \phi_{2j} & \phi_{3j} & \phi_{4j} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \\ \phi_{4j} \end{bmatrix} = (\phi_{2j} - \phi_{3j})^2 \,. \qquad (3.26)$$

This means that α_{ij} can be found from the modal displacements of the original structure.

Finite element packages such as MSC/NASTRAN output element strain energy values directly. This provides a convenient way to find α_{ij}. Since the strain energy of an axially-distorted element is $\frac{1}{2}kx^2$, where x is the axial displacement, its modal strain energy is

$$\text{S.E.}_{ij} = \frac{1}{2} k_{oi} {\phi_j}^T K' \phi_j = \frac{1}{2} k_{oi} \alpha_{ij} \qquad (3.27)$$

so that

$$\alpha_{ij} = \frac{2}{k_{oi}} \text{S.E.}_{ij} \,. \qquad (3.28)$$

For the original analysis, springs of near-zero stiffness (say $k_{oi} = 2$ units) placed in the structure will not appreciably affect its eigenvalues. Then, if k_{oi} is set to 2 units, α_{ij} for damper location i may be read directly from the strain energy output of a normal mode analysis.

Alternatively, element axial force information for each mode may be used to find α_{ij}. Since the force present in an axially-distorted element is $k\,x$, the axial displacement may be found by the expression

$$x = \frac{f_{ij}}{k_{oi}} \qquad (3.29)$$

where f_{ij} is the axial force present in a spring of near-zero stiffness k_{oi} in the jth mode of a mass-normalized modal analysis. If k_{oi} is set to one unit, then the modal displacement is read directly from the modal force output in the element k_{oi}. The value of α_{ij} is then

$$\alpha_{ij} = \left(\frac{f_{ij}}{k_{oi}}\right)^2 . \tag{3.30}$$

This method of finding α_{ij} has the advantage that the signs of the relative axial displacements are provided by the force output of a normal mode analysis. This becomes important when the compliant models of many modes are incorporated into a unified model in Section 3.3.

Still, k_{Bij} or equivalent information must be found. This is done by using Equation 3.11. If the stiffness k_i of the rod in series with each damper is known, then an individual real eigenvalue analysis is run for each k_i, so that the values of κ_{sij} may be found from Equations 3.2 and 3.11. For n possible strut locations, this method involves $n + 1$ real eigenvalue analyses.

3.2.4 Method of Slopes and Curvatures

To reduce the number of eigenvalue analyses further, one could use the second derivative of each eigenvalue $({\omega_j}^2)$ with respect to the added stiffness to find the value of k_{Bij}. This is the method of slopes and curvatures.

The second derivative of the compliant model with respect to a design change k_i is (see also Equation 3.17)

$$\left.\frac{\partial^2(\omega^2)}{\partial k_i^2}\right|_{k_i=0} = \left.\frac{-\alpha_{ij}\,{k_{Bij}}^2\,2\,(\alpha_{ij}k_i + k_{Bij})\,\alpha_{ij}}{(\alpha_{ij}k_i + k_{Bij})^4}\right|_{k_i=0} \tag{3.31}$$

$$= \frac{-2\,{\alpha_{ij}}^2\,{k_{Bij}}^3}{k_{Bij}} \tag{3.32}$$

so that

$$k_{Bij} = \frac{-2{\alpha_{ij}}^2}{({\omega_{ij}}^2)''|_{k_i=0}} . \tag{3.33}$$

The second eigenvalue derivative of the finite element model is found as follows [18]. First, differentiate Equation 3.23 with respect to a change e in the system.

$$\begin{aligned} ({\omega_j}^2)'' &= \frac{d}{de}\left\{{\phi_j}^T \left[K' - M'{\omega_j}^2\right] \phi_j\right\} && (3.34) \\ &= 2{\phi_j}^T \left[K' - M'{\omega_j}^2\right] {\phi_j}' + {\phi_j}^T \left[K'' - M''{\omega_j}^2 - M'({\omega_j}^2)'\right] \phi_j && (3.35) \end{aligned}$$

where ${\phi_j}'$ is the derivative of the ith eigenvector. When the change is only an added stiffness, $M' = 0$ and $K'' = 0$, so that

$$({\omega_j}^2)'' = 2{\phi_j}^T K' {\phi_j}'. \tag{3.36}$$

Evaluation of Equation 3.36 at $k_i = 0$ would provide the information needed to compute k_{Bj} using Equation 3.33.

Thus, the compliant model can be completely characterized by only one analysis of the normal modes, along with analysis of the first eigenvector derivatives and the first and second eigenvalue derivatives. Unfortunately, in order to compute the first eigenvector derivative, one needs to either [9]

1. invert an $n \times n$ matrix, where n is the index of freedom of the entire finite element model, or

2. use an expression which may become numerically sensitive if closely-spaced eigenvalues exist.

The matrix inversion is nearly as computationally costly to perform as an additional real eigenvalue analysis, and the computation of the second derivative appears more

difficult to implement. Therefore, the method of frequency slopes is used in this dissertation.

Other strategies, such as finite difference approximations, could be used to find α_{ij} and k_{Bij}, but will not be covered in this dissertation.

3.3 Spherical Model

3.3.1 Purpose

The compliant model of Section 3.1 deals with each mode separately; the assumption is that all modes behave independently of each other. However, in structures which possess closely-spaced eigenvalues, this is not the case. A unified compliant model to account for the effects of an added spring or damper on all modes is presented in this section.

The solution to the eigenvalue problem for a system of m modes may be visualized as an m-dimensional ellipsoid in which the length of each major axis is the reciprocal of an eigenvalue [19]. In the case of repeated roots, two or more of the eigenvectors are oriented arbitrarily within a hyper-planar cross-section of this ellipsoid orthogonal to each other and all other eigenvectors. When even a slight modification to the system is made, the orientation of these eigenvectors may change drastically. Therefore it is necessary to unify the compliant model so that all necessary modes are included.

3.3.2 Basis

In order to develop a model which is able to do this, the following observations from the compliant model are noted. The first is that one or several springs added

to a system will increase or leave unchanged any eigenvalue. This is related to the inclusion principle [19]. Second, frequency shifts for a set of modes due to a set of springs may be estimated by performing eigenvalue analyses with one spring inserted at a time and summing the results according to the compliant model. Even if the estimated shifts in frequency for each mode are incorrectly allocated amongst closely-spaced modes, the total of the shifts in frequency within the set of modes is approximately correct. This stems from the fact that the sum of the eigenvalues of a matrix is equal to its trace [19].

Third, if a single spring or dashpot is added to a system, only one of a set of identical eigenvalues will be shifted or damped. This is also related to the so-called inclusion principle [19].

Fourth, in a system for which damping is being estimated by a compliant model, all modes j which are highly-spaced have nearly equal ratios $\frac{k_{sij}}{\alpha_{ij}}$ for a given strut i. (These modes are generally accurately represented by the simple compliant model.) Then the values of α_{ij} found by the method of frequency slopes may be used to proportionately allocate the total frequency shift among all of the modes. The implication of the equal ratios $\frac{k_{sij}}{\alpha_{ij}}$ is that the same spring is affecting each of the separate compliant models of structural modes.

3.3.3 Description

All of these properties imply a *spherical* compliant model, described by an auxiliary eigenvalue problem which sorts out frequency (or complex eigenvalue) shifts induced in closely-spaced modes by a set of added springs (or dampers). The spherical compliant model is shown in Figure 3.5. The unit mass represents the modal

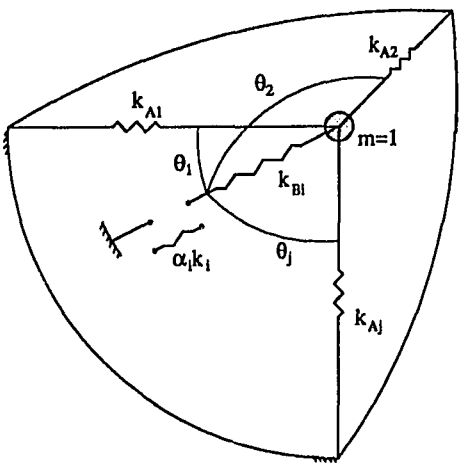

Figure 3.5: Spherical Compliant Model

mass of all of the modes. Each spring connects the modal mass to ground along one of the m orthogonal axes. The value of the spring on a given axis is equal to the corresponding modal stiffness, so that motion of the mass along the axis represents vibration of that mode. In this way, one mass and m springs are used to represent all normal modes.

When a spring k_i is added to the system, it modifies the frequencies according to α_{ij}'s and k_{sij}'s which are nearly proportionally distributed among the modes. This may be modeled by a single compliant series spring k_{Bi} oriented to achieve the observed frequency shift k_{sij} in each mode, as shown in Figure 3.5. This orientation is determined by letting the squares of its direction cosines be proportional to α_{ij} over all modes j for the ith strut. The value of k_{Bi} is set by observing all of the frequency shifts k_{sij} induced by inserting k_i into the structure so that

$$k_{Bi} = \sum_{j=1}^{m} k_{Bij} = \sum_{j=1}^{m} \frac{\alpha_{ij} k_i k_{sij}}{\alpha_{ij} k_i - k_{sij}} \tag{3.37}$$

and

$$\cos^2 \theta_{ij} = \frac{\alpha_{ij}}{\sum_{j=1}^{m} \alpha_{ij}} = \frac{\alpha_{ij}}{\alpha_i} \tag{3.38}$$

where

$$\alpha_i = \sum_{j=1}^{m} \alpha_{ij} \ . \tag{3.39}$$

3.3.4 Procedure

The procedure for defining the spherical model follows. First, the normal mode analysis of the original system provides the modal stiffness values k_{Aj} for each mode being studied along with the orientation magnitudes $|cos\theta_{ij}|$ from α_{ij} obtained by the method of frequency slopes (see Section 3.2). Next, n individual eigenvalue analyses with a known spring k_i placed in each of the n strut locations being studied, determine k_{si}. It is simply the summation of all of the frequency shifts for the set of consecutive frequencies included.

The values of α_{ij} and $\cos\theta_{ij}$ are obtained from the modal displacements as follows. Near-zero-stiffness elements are included in candidate locations of the original structure, as described in Section 3.2.3. The normal mode analysis on the original structure then yields an axial displacement x_{ij} between each candidate pair of nodes i for each mode j. This value may be positive or negative. As is stated in Section 3.2.3,

$$\alpha_{ij} = x_{ij}^2 \ . \tag{3.40}$$

The sign and magnitude of the direction cosines γ_{ij} are determined by

$$\gamma_{ij} \equiv \cos\theta_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}} = \frac{x_{ij}}{\sqrt{\alpha_i}} \ . \tag{3.41}$$

The direction cosines $\cos\theta_{ij}$ satisfy the properties that their squares are proportional to α_{ij} over all modes for a given strut. Also, the sum of the squares of $\cos\theta_{ij}$ for a given strut are equal to one. The γ_{ij} for a given strut i may be interpreted as the unit normalized displacements x_{ij}, while α_i is the sum of the squares of x_{ij}.

Equations 3.40 and 3.41 determine the values of α_{ij} and the orientations $\cos\theta_{ij}$ (complete with sign) of the locations of candidate struts in the spherical model.

3.3.5 Use in Evaluating Eigenvalue Shifts

Once the values of the direction cosine vector for each strut location have been determined, the spherical model may be used to predict frequency shift or loss factor achieved by a combination of added struts.

The mass matrix for the system of the spherical model is the m-dimensional identity matrix, where m is the number of modes being studied. The stiffness matrix of the unmodified spherical model of the structure then is the diagonal matrix containing the m eigenvalues or modal stiffnesses, that is

$$ {}_{o}K_{o} = \begin{bmatrix} k_{A1} & 0 & 0 & \cdots & 0 \\ 0 & k_{A2} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & k_{Am} \end{bmatrix} \tag{3.42} $$

3.3.5.1 Added Spring If the goal of the analysis is to estimate the effect of an added spring on the structure, a matrix corresponding to each additional element

k_i on the system is added to the stiffness matrix.

$$ {}_oK = {}_oK_o + \sum_{i=1}^{m} k_{si}\gamma_i{\gamma_i}^T \tag{3.43} $$

where

$$ k_{si} = \sum_{j=1}^{m} \frac{\alpha_{ij}k_i k_{Bij}}{\alpha_{ij}k_i + k_{Bij}} \tag{3.44} $$

$$ = \sum_{j=1}^{m} k_{sij} \tag{3.45} $$

$$ = \frac{\alpha_i k_i k_{Bi}}{\alpha_i k_i + k_{Bi}} \tag{3.46} $$

and

$$ {\gamma_i}^T = [\ \cos\theta_{i1} \quad \cos\theta_{i2} \quad \cdots \quad \cos\theta_{im}\ ]\ . \tag{3.47} $$

The $\gamma_i{\gamma_i}^T$ matrix is the result of an extension of Weaver's example for a two-degree system on p. 221 of [30].

The eigenvalues ${}_o{\omega_j}^2$ of the auxiliary, augmented stiffness matrix ${}_oK$ are then used as an estimate of the eigenvalues of the system. The eigenvectors ${}_o\phi_j$ of this matrix determine the new directions of oscillation of the spherical system when springs k_i are inserted. They are combined into an auxiliary modal matrix

$$ {}_o\Phi = [\ {}_o\phi_1 \quad {}_o\phi_2 \quad \cdots \quad {}_o\phi_m]\ . \tag{3.48} $$

Once the new frequencies and orientations (${}_o\Phi$) of the modes of the spherical model with added stiffnesses have been determined by an eigenvalue analysis of ${}_oK$, the contribution of each stiffness i to the modal stiffness of mode j may be found. The value of the frequency shift in a mode j attributable to each k_i within the augmented spherical system is equal to the value of k_{si} times the square of the cosine of the angle

between the new axis of the mode j and the axis of k_{si}. Both ${}_o\phi_j$ and γ_i are unit normalized, so by the Cauchy-Schwartz theorem the cosine of that angle is equal to their inner product.

$$ {}_ok_{sij} = (\gamma_i{}^T {}_o\phi_j)^2 k_{si} \tag{3.49} $$

In the new orientation of the modes in the spherical system, the values of ${}_ok_{sij}$ will be added to the summation of the contributions of the original k_{Aj} to the new modal frequencies. This summation is

$$ {}_ok_{Aj} = \sum_{k=1}^{m} \left({}_o\phi_j{}^T e_k \right)^2 k_{Ak} \ , \tag{3.50} $$

where e_k denotes the kth column of the $m \times m$ identity matrix. It should be true that the frequencies of the new modes are equal to the contributions of all of the springs in the model, that is

$$ {}_o\omega_j{}^2 = {}_ok_{Aj} + \sum_{i=1}^{n} {}_ok_{sij} \ . \tag{3.51} $$

This provides a check for computation of ${}_ok_{Aj}$ and ${}_ok_{sij}$.

In a similar way, an auxiliary complex eigenvalue problem could be solved for each combination of struts at each level of damping. This is not difficult since the number of modes of the auxiliary system is much smaller than the index of freedom of the finite element model. Such a solution is developed in Section 3.3.6. However, in order to speed computation of modal loss factor, the frequency shifts of the spherical model will be used to arrive at closed-form expressions for modal loss factor similar to those of the simple compliant model (Equation 3.8).

3.3.5.2 Added Spring-Damper Series If a set of series spring-damper pairs k_i—$c_i s$ are to be inserted into the existing structure, then a set of ${}_o\kappa_{sij}$ may be

obtained from the augmented model of the structure in which stiffnesses alone are inserted.

First, the re-orientation of the model due to added spring-damper elements must be estimated, then modified values of κ_{sij} for each spring-damper location may be estimated. The estimation of η_j is based on these new ${}_o\kappa_{sij}$.

To predict the frequency and orientation shifts of the spherical model near the peak damping influence of a spring-dashpot series k_i—c_i, a different stiffness is used in the second term of Equation 3.43. The peak damping occurs near

$$r_{ij} = \frac{\alpha_{ij}\, c_i\, \omega_j}{k_{sij}} = 1 \ . \tag{3.52}$$

The dynamic stiffness for such a series is

$$F_{ij} \approx \frac{i\,\alpha_{ij}\, c_i\, \omega_j\, k_{sij}}{i\,\alpha_{ij}\, c_i\, \omega_j + k_{sij}} \approx k_{sij} \frac{{r_{ij}}^2 + i\, r_{ij}}{{r_{ij}}^2 + 1} \tag{3.53}$$

so that the dynamic stiffness near the peak

$$F_{ij}\Big|_{r_{ij}=1} \approx \frac{k_{sij}}{2}(1+i) \ . \tag{3.54}$$

Since the real part of this dynamic stiffness is $\frac{k_{sij}}{2}$, the stiffness matrix for the spherical system becomes

$${}_oK = {}_oK_o + \sum_{i=1}^{m} \frac{k_{si}}{2} \gamma_i {\gamma_i}^T \ . \tag{3.55}$$

with k_{si}, α_i, and γ_i defined as in Equations 3.44 through 3.47.

Once the eigenvalue problem for the spherical system has been solved so that ${}_o\phi_j$ are known, Equations 3.49 and 3.50 may be used to find ${}_ok_{sij}$ and ${}_ok_{Aj}$ for the damped system. The check of Equation 3.51 becomes

$${}_o{\omega_j}^2 = {}_ok_{Aj} + \frac{1}{2}\sum_{i=1}^{n} {}_ok_{sij} \ . \tag{3.56}$$

Now, one can compute an approximate modal loss factor based upon the orientation of the spherical system with only stiffnesses present. Referring to Equations 3.49, 3.50, and 3.8, one can write this expression for the loss factor:

$$ {}_{o}\eta_j \approx \frac{\sum_{i=1}^{n} {}_{o}\kappa_{sij} \frac{r_{ij}}{1+r_{ij}^2}}{1 + \sum_{i=1}^{n} {}_{o}\kappa_{sij} \frac{r_{ij}^2}{1+r_{ij}^2}} \tag{3.57} $$

where

$$ {}_{o}\kappa_{sij} = \frac{{}_{o}k_{sij}}{{}_{o}k_{Aj}} \, . \tag{3.58} $$

Although estimates of η_j based upon

$$ {}_{o}r_{ij} = \frac{{}_{o}\alpha_{ij} c_i \sqrt{{}_{o}k_{Aj}}}{{}_{o}k_{sij}} \tag{3.59} $$

$$ {}_{o}\alpha_{ij} = \frac{{}_{o}k_{sij}}{k_{si}} \sum_{j=1}^{m} \alpha_{ij} \tag{3.60} $$

may be intuitively attractive, they have not proven more accurate.

The drawback of using this method instead of re-solving the complex eigenvalue problem for the auxiliary system is that with the spring-damper series combination, the modes of the spherical system will not orient to exactly the same position as they do when springs $\frac{k_{sij}}{2}$ alone are added to the system.

The advantage of using Equation 3.57 is that for each combination of damper locations, only one additional small, real eigenvalue analysis must be performed. The optimal damping for that combination may then be quickly found by optimizing ${}_{o}\eta_j$ using Equation 3.57.

3.3.6 State-Space Representation of Spherical Model

In order to develop a state-space representation of the spherical model of locally-modified structural modes, the model of Figure 3.5 is used. In it, the compliances

to added elements are connected to the modal mass at the angles θ_{ij}. A spring, damper, combination, or active control element may act upon the mass from this angle if placed in the corresponding position of the physical structure. In the following development, it is assumed that a series element k_i bears the force applied across the nodes of location i, so values of k_{si} may be found. The displacement of the control element in the realm of the spherical model is denoted z_i.

The equations of motion of the mass are now written. The force on the ith added element due to small displacements of the modal mass and z_i is

$$R_i = -k_{si}\left(\sum_{j=1}^{m} u_j \cos\theta_{ij} - z_i\right) , \tag{3.61}$$

and the component of this force in the jth direction is

$$R_{ij} = -k_{si}\left(\sum_{j=1}^{m} u_j \cos\theta_{ij} - z_i\right)\cos\theta_{ij} . \tag{3.62}$$

Then the equation of motion in the jth direction is

$$m\ddot{u}_j + k_{Aj} - \sum_{i=1}^{n} R_{ij} = 0. \tag{3.63}$$

With $m = 1$,

$$\ddot{u}_j + k_{Aj} + \sum_{i=1}^{n} \cos\theta_{ij}\left(\sum_{k=1}^{m} u_k \cos\theta_{ik}\right) - \sum_{i=1}^{n} k_{si} z_i \cos\theta_{ij} = 0 . \tag{3.64}$$

The following definitions are added to those of Section 3.3.5.

$$u_{(m\times 1)} = \left\{ u_1 \ \cdots \ u_j \ \cdots \ u_m \right\}^T \tag{3.65}$$

$$z_{(n\times 1)} = \left\{ z_1 \ \cdots \ z_i \ \cdots \ z_n \right\}^T \tag{3.66}$$

$$\Gamma_{(m\times n)} = [\ \gamma_1 \ \cdots \ \gamma_i \ \cdots \ \gamma_n\ ] \tag{3.67}$$

$$K_{A\ (m\times m)} = {}_oK_o \tag{3.68}$$

$$K_{s\ (n\times n)} = \begin{bmatrix} k_{s1} & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & k_{si} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & k_{sn} \end{bmatrix} \tag{3.69}$$

$$\alpha_{(n\times n)} = \begin{bmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \alpha_i & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \alpha_n \end{bmatrix} \tag{3.70}$$

(3.71)

Now Equations 3.64 may be written in compact matrix form

$$\ddot{u} + \left(K_A + \Gamma K_s \Gamma^T\right) u - \Gamma K_s z = \{0\} \ . \tag{3.72}$$

A force input f_i at z_i must be brought into the system. This is equal to the reaction force of Equation 3.61,

$$f_i = R_i = -k_{si} \left(\sum_{j=1}^{m} u_j \cos\theta_{ij} - z_i \right) \tag{3.73}$$

or in matrix notation

$$f = \left\{ \begin{array}{cccccc} f_1 & \cdots & f_i & \cdots & f_n \end{array} \right\}^T = -K_s \left(\Gamma^T u - z\right) \ . \tag{3.74}$$

This is restated as

$$z = \Gamma^T u + {K_s}^{-1} f \ . \tag{3.75}$$

The matrix inversion is simple since K_s is diagonal.

Now Equations 3.72 and 3.75 may be combined into a standard state space representation which includes $K_s{}^{-1}$ as a feedforward matrix,

$$\begin{Bmatrix} \ddot{u} \\ \dot{u} \end{Bmatrix} = \begin{bmatrix} 0_{(m\times m)} & -K_A \\ I_{(m\times m)} & 0_{(m\times m)} \end{bmatrix} \begin{Bmatrix} \dot{u} \\ u \end{Bmatrix} + \begin{Bmatrix} \Gamma \\ 0_{(m\times n)} \end{Bmatrix} f \tag{3.76}$$

$$z = \begin{bmatrix} 0_{(n\times m)} & \Gamma^T \end{bmatrix} \begin{Bmatrix} \dot{u} \\ u \end{Bmatrix} + K_s{}^{-1} f \ . \tag{3.77}$$

To get z in terms of a vector of physically measurable displacements y,

$$y = \alpha z \ . \tag{3.78}$$

An output feedback could be used for control purposes

$$f = -K_f z \tag{3.79}$$

where K_f is the $n \times n$ feedback matrix.

The controllability and observability of the system of equations 3.76 and 3.77 depend upon the rank of Γ. Although the state space representation is very simple, it retains all of the properties of the spherical compliant model of a structure.

Now in the case in which the control is due to viscous dashpots c_i, the force through k_{si} and c_i is equal to the control force

$$R_i = -k_{si} \left(\sum_{j=1}^{m} u_j \cos\theta_{ij} - z_i \right) = -c_i \alpha_i \dot{z}_i \tag{3.80}$$

that is,

$$C_d \alpha \dot{z} + K_s z - K_s \Gamma^T u = \{0\}_{(n\times 1)} \tag{3.81}$$

or

$$\dot{z} = -\left[C_d \alpha\right]^{-1} K_s z + \left[C_d \alpha\right]^{-1} K_s \Gamma^T u \tag{3.82}$$

where C_d is the diagonal matrix containing values c_i. Equation 3.82 may be combined with Equation 3.72 to form

$$\begin{Bmatrix} \ddot{u} \\ \dot{u} \\ \dot{z} \end{Bmatrix} = \begin{bmatrix} 0_{(m\times m)} & -K_A - \Gamma K_s \Gamma^T & \Gamma K_s \\ I_{(m\times m)} & 0_{(m\times m)} & 0_{(m\times n)} \\ 0_{(n\times m)} & [C_d\alpha]^{-1} K_s \Gamma^T & -[C_d\alpha]^{-1} K_s \end{bmatrix} \begin{Bmatrix} \dot{u} \\ u \\ z \end{Bmatrix}. \tag{3.83}$$

This matrix differential equation is suitable for solution by the *damp* function of Matlab. Examples in Sections 3.6.2 and 3.6.3 illustrate the use of this solution method.

3.4 Implications of Compliant Model on Damping Optimization

In order to find the optimal combination of damping strut locations and damping coefficients for a structure, damping must be evaluated repeatedly for each case in the search for an optimum. The use of complex eigenvalue analysis to compute modal damping ratios is very expensive, especially for structures whose finite element models are extremely large. Furthermore, derivative information is difficult to extract from the complex eigenvalue solution, making it difficult to discern the best direction of search with a design space.

Even though modal damping ratio is computationally expensive to compute through complex eigenvalue analysis, the shape of the modal damping plot versus strut damping coefficient is very predictable. The compliant model of a structural mode results in the closed-form expression of Equation 3.8 or 3.57 for modal damping ratio which very closely approximates this surface. The use of this expression can result in time savings in excess of two orders of magnitude compared to complex eigenvalue analysis. Gradient information is also easily obtainable through this ex-

pression. The derivative on η with respect to damping coefficient c_i in location i is found as follows.

$$\eta_j = \frac{\sum_{i=1}^{n} \kappa_{sij} \frac{r_{ij}}{1+{r_{ij}}^2}}{1 + \sum_{i=1}^{n} \kappa_{sij} \frac{{r_{ij}}^2}{1+{r_{ij}}^2}} = \frac{N_j}{D_j} \tag{3.84}$$

$$\frac{\partial \eta_j}{\partial r_{ij}} = \frac{(N_j)_r D_j - N_j (D_j)_r}{{D_j}^2} \tag{3.85}$$

$$= \kappa_{sij} \frac{(1-{r_{ij}}^2) D_j - 2 r_{ij} N_j}{(1+{r_{ij}}^2)^2 {D_j}^2} \tag{3.86}$$

where the r subscript denotes differentiation with respect to r_{ij}. Assuming that ω_j does not change much in the case of added damping,

$$r_{ij} = \frac{\alpha_{ij} c_i \omega_j}{k_{sij}} \tag{3.87}$$

$$= \frac{\alpha_{ij} c_i \omega_j}{k_{Aj} \kappa_{sij}} \tag{3.88}$$

$$= \frac{\alpha_{ij} c_i}{\omega_j \kappa_{sij}} \tag{3.89}$$

$$\frac{\partial r_{ij}}{\partial c_i} \approx \frac{\alpha_{ij}}{\omega_j \kappa_{sij}} \,. \tag{3.90}$$

Then

$$\frac{\partial \eta_j}{\partial c_i} = \sum_{i=1}^{n} \frac{\partial \eta_{ij}}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \tag{3.91}$$

$$\approx \sum_{i=1}^{n} \frac{\alpha_{ij}}{\omega_j} \frac{(1-{r_{ij}}^2) D_j - 2 r_{ij} N_j}{(1+{r_{ij}}^2)^2 {D_j}^2} \tag{3.92}$$

If complex eigenvalue analysis is used in conjunction with or as a check on the compliant model, the insight the compliant model supplies regarding the shape of the loss factor surface can be an aid in choosing a direction of search for a maximum.

3.5 Relationship to Other Methods of Damping Estimation

3.5.1 Compliant Model and Modal Strain Energy

The modal strain energy method gives similar results to the compliant model when the parameters of the compliant model are found by the method of frequency slopes. As α_{ij} is obtained by looking at the modal strain energy of elements in the positions of interest, so is $\frac{V_{ij}}{V_j}$ in the modal strain energy method. However, the modal strain energy method assumes that eigenvectors are essentially unchanged, while the compliant model uses modal compliance $\frac{1}{k_{Bij}}$ to attempt to account for the eigenvector changes. The modal strain energy method accurately predicts damping at low values of damping coefficient, but its accuracy decreases as c_i approaches its optimal value. In a case where the original structure contains a member in the position of interest so that the modal strain energy method may be used, the information used by the modal strain energy method is the same as that which would be used by the compliant model if k_{Bij} were assumed infinite.

It is possible, once the parameters for one of these methods are fully known for a given strut, to convert them to parameters in terms of the other method.

3.5.2 Modal Strain Energy to Compliant Model

One might convert a modal strain energy model into a compliant model in order to account for modes which are close in frequency by using the spherical model.

The parameters of a modal strain energy model for a given strut are

1. the ratio of the total modal strain energy for the strut position in question, $\frac{V_{ij}}{V_j}$, and

2. the strut loss factor for the position in question at the frequency of mode j, η_{ij}.

In order to convert these into parameters of the compliant model, it is assumed that a dashpot c_i is placed into the model. A correlation is made by equating the damping in mode j due to a single damping strut i as computed by the two models.

The values of $\frac{V_{ij}}{V_j}$, k_{ai} (static stiffness of damping strut i), k_{bi} (inner stiffness of damping strut i), and ω_j are known. The damping based on modal strain energy is

$$\eta_{ij} \approx \frac{V_{ij}}{V_j} \frac{\kappa_i r_{ij}}{1 + r_{ij}^2(1+\kappa_i)} \tag{3.93}$$

where

$$\kappa_i = \frac{k_{bi}}{k_{ai}} \tag{3.94}$$

and

$$r_{ij} = \frac{\omega_j c_i}{k_{bi}} \ . \tag{3.95}$$

Values $\tilde{k}_{sij}$ and $\tilde{\alpha}_{ij}$ of an equivalent compliant model are sought. With the definitions

$$\tilde{r}_{ij} = \frac{\alpha_{ij}\omega_j c_i}{\tilde{k}_{sij}} \tag{3.96}$$

and

$$\tilde{\kappa}_{sij} = \frac{\tilde{k}_{sij}}{k_{Aj}} = \frac{\tilde{k}_{sij}}{\omega_j^2} \ , \tag{3.97}$$

the damping in the equivalent compliant model is estimated by

$$\eta_{ij} \approx \tilde{\kappa}_{sij} \frac{\tilde{r}_{ij}}{1 + \tilde{r}_{ij}^2(1+\tilde{\kappa}_{sij})} \ . \tag{3.98}$$

The numerators and denominators respectively of Equations 3.93 and 3.98 are equated, resulting in

$$\tilde{\alpha}_{ij} = \frac{V_{ij}}{V_j} \frac{k_{Aj}}{k_{ai}} \tag{3.99}$$

and

$$\tilde{k}_{sij} = \omega_j^2 \frac{\kappa_i^2}{2(1+\kappa_i^2)} \left(\frac{V_{ij}}{V_j}\right)^2 \left(1 + \sqrt{1 + \frac{4(1+\kappa_i)}{\kappa_i^2}\left(\frac{V_j}{V_{ij}}\right)^2}\right) . \tag{3.100}$$

These equations result in a compliant model which gives damping values similar to those computed by the modal strain energy method for a single damping strut. With many dampers present, the following compliant model estimate may be used.

$$\eta_j \approx \frac{\sum_{i=1}^{n} \tilde{\kappa}_{sij} \frac{\tilde{r}_{ij}}{1+\tilde{r}_{ij}^2}}{1 + \sum_{i=1}^{n} \tilde{\kappa}_{sij} \frac{\tilde{r}_{ij}^2}{1+\tilde{r}_{ij}^2}} . \tag{3.101}$$

The parameters $\tilde{k}_{sij}$ and $\tilde{\alpha}_{ij}$ can also be used to build a spherical model including many modes. However, the signs of the direction cosines for each strut will be unknown.

3.5.3 Converting Compliant Model to Modal Strain Energy Model

In a similar way, one might desire to transform the parameters which have been obtained for the compliant model into a modal strain energy model. One might wish to do this in order to utilize the additional information inherent in the compliant model and yet maintain the simplicity of the expression for the modal strain energy method.

Again, correlations are drawn by equating the effect of a single damping strut in both models. The parameters k_{Aj}, k_{sij}, and α_{ij} of the compliant model are known. A value k_{bi} of the inner stiffness of the damping strut is also known.

The damping in mode j by the compliant model is

$$\eta_j \approx \frac{\kappa_{sij} r_{ij}}{1 + r_{ij}^2(1+\kappa_{sij})} . \tag{3.102}$$

where, as before,

$$\kappa_{sij} = \frac{k_{sij}}{k_{Aj}} \tag{3.103}$$

and

$$r_{ij} = \frac{\alpha_{ij} c_i \omega_j}{k_{sij}} \ . \tag{3.104}$$

Values of estimated modal strain ratio $\frac{\hat{V}_{ij}}{V_j}$ and loss factor $\hat{\eta}_{ij}$ of strut i for an equivalent modal strain energy model are sought. The damping due to damper i computed by this model is

$$\eta_{ij} \approx \frac{\hat{V}_{ij}}{V_j}\hat{\eta}_{ij} = \frac{\hat{V}_{ij}}{V_j}\frac{\hat{\kappa}_{ij}\hat{r}_{ij}}{1+\hat{r}_{ij}^2(1+\hat{\kappa}_{ij})} \tag{3.105}$$

where

$$\hat{\eta}_{ij} = \frac{\hat{\kappa}_{ij}\hat{r}_{ij}}{1+\hat{r}_{ij}^2(1+\hat{\kappa}_{ij})} \ , \tag{3.106}$$

$$\hat{\kappa}_{ij} = \frac{k_{bi}}{\hat{k}_{aij}} \ , \tag{3.107}$$

and

$$\hat{r}_{ij} = \frac{\omega_j c_i}{k_{bi}} \ . \tag{3.108}$$

The parameter $\hat{k}_{aij}$, an estimate of the static stiffness of the strut, is not needed in the development. With c_i set by the designer, $\hat{r}_{ij}$ are known.

The numerators and denominators respectively of Equations 3.102 and 3.105 are equated, resulting in

$$\frac{\hat{V}_{ij}}{V_j}\hat{\kappa}_{ij} = \alpha_{sij}\frac{k_{bi}}{k_{Aj}} \tag{3.109}$$

and

$$\hat{\kappa}_{ij} = \frac{\alpha_{ij}^2}{\kappa_{sij}}\left(\frac{k_{bi}}{k_{Aj}}\right)^2(1+\kappa_{sij}) - 1 \ . \tag{3.110}$$

Combining Equations 3.109 and 3.110 results in

$$\frac{\hat{V}_{ij}}{V_j} = \frac{k_{bi}}{k_{Aj}} \frac{\alpha_{ij}}{\alpha_{ij}^2 \left(\frac{k_{bi}}{k_{sij}}\right)^2 (1 + \kappa_{sij}) - 1} . \tag{3.111}$$

The loss factor of a system containing n struts is then estimated by

$$\eta_j \approx \sum_{i=1}^{n} \frac{\hat{V}_{ij}}{V_j} \hat{\eta}_{ij} . \tag{3.112}$$

3.6 Examples

3.6.1 Continuous System

3.6.1.1 Rod Acted Upon by a Spring In this section it is shown that the compliant model may be used to predict for a continuous system frequency shifts induced by discrete elements. The example system is a rod of uniform cross-section clamped on one end and acted upon by a grounded spring on the other, shown in Figure 3.6. In order to solve this system for the frequencies of axial vibration, a partial differential equation is written which applies over the length of the rod. The rod is fixed at the left boundary and acted upon by a spring force at the right.

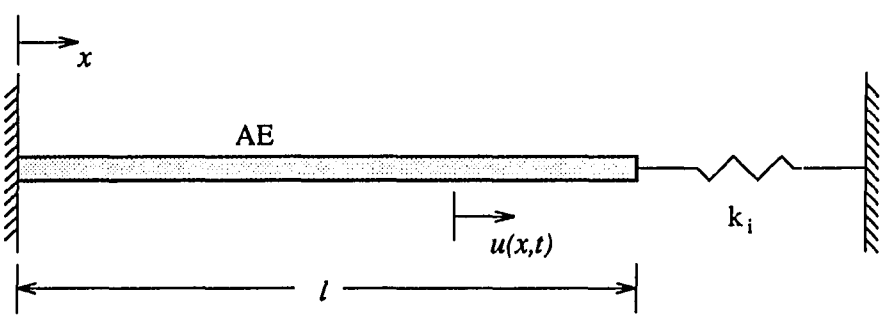

Figure 3.6: Continuous Rod with Spring

The solution to this differential equation is given by Weaver on p. 393 of [30]. The frequencies satisfy the transcendental equation

$$\xi_j \tan \xi_j = -\zeta_j \tag{3.113}$$

where

$$\zeta_j = \frac{m\ell\omega_j{}^2}{k_1} , \tag{3.114}$$

$$\xi_j = \frac{\omega_j \ell}{a} , \tag{3.115}$$

$m\ell$ is the mass of the rod, ℓ is its length, and a is the speed of wave propagation in the rod. Here ζ_j does not represent modal damping ratio. Equation 3.113 must be solved for ω_i at each value of k_i by iteration.

The continuous rod may be viewed as a compliant model, as in Figure 3.7. In this case the added spring corresponds to k_i in the compliant model. The value of k_{Aj} corresponds to the jth eigenvalue when $k_1 = 0$. The solution to

$$\tan \xi_j = \infty \tag{3.116}$$

is

$$\xi_j = \tfrac{\omega_j \ell}{a} = (j - \tfrac{1}{2})\pi \ , \quad j = 1, 2, 3, \ldots \infty \ . \tag{3.117}$$

For the first three modes,

$$k_{Aj} = \omega_j{}^2 = \left(\frac{\pi a}{2\ell}\right)^2 , \ \left(\frac{3\pi a}{2\ell}\right)^2 , \ \left(\frac{5\pi a}{2\ell}\right)^2 . \tag{3.118}$$

The value of k_{B1j} may be found by various methods, but here it is found by setting $k_1 = \infty$. The solution to the equation of motion of the bar with both ends

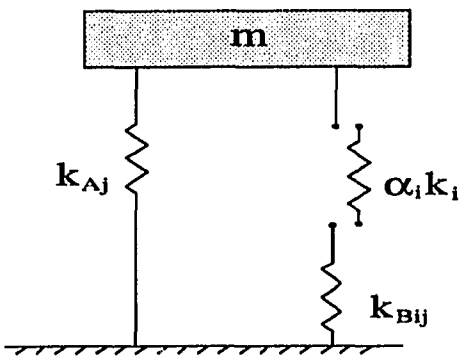

Figure 3.7: Compliant Model of a Structural Mode

fixed is

$$\xi_j \tan \xi_j = 0 \ . \tag{3.119}$$

then

$$\xi_j = \frac{\omega_j \ell}{a} = j\pi \ , \quad j = 1, 2, 3, \ldots \infty \ . \tag{3.120}$$

For the first three modes,

$$\omega_j = \frac{\pi a}{\ell} \ , \frac{2\pi a}{\ell} \ , \frac{2\pi a}{\ell} \tag{3.121}$$

so that

$$k_{B1j} = {\omega_j}^2 - k_{Aj} = \frac{3}{4}\left(\frac{\pi a}{\ell}\right)^2 \ , \frac{7}{4}\left(\frac{\pi a}{\ell}\right)^2 \ , \frac{11}{4}\left(\frac{\pi a}{\ell}\right)^2 \ . \tag{3.122}$$

Finally, the value of α_{1j} may be found by placing a known, finite value of k_1 in the system or by taking the derivative with respect to k_1 at $k_1 = 0$. The latter approach is used here. First, Equation 3.113 is rearranged and implicitly differentiated to arrive at the eigenvalue derivative.

$$f = \xi_j \tan \xi_j + \zeta_j = 0 \tag{3.123}$$

$$\frac{\partial \omega_j}{\partial k_1} = -\frac{\frac{\partial f}{\partial k_1}}{\frac{\partial f}{\partial \omega_j}} \tag{3.124}$$

$$= \frac{m\ell{\omega_j}^2}{{k_1}^2\left(\frac{\ell}{a}\tan\xi_j + \frac{\omega_j\ell^2}{a^2}\sec^2\xi_j + 2\frac{m\ell\omega_j}{k_1}\right)} \tag{3.125}$$

Rearranging Equation 3.113 results in

$$k_1 = -\frac{m\omega_j a}{\tan\xi_j} \tag{3.126}$$

which may be used to eliminate k_1 from Equation 3.125. After simplification,

$$\frac{\partial\omega_j}{\partial k_1} = \frac{\sin^2\xi_j}{m\omega_j\ell - am\sin\xi_j\cos\xi_j}\ . \tag{3.127}$$

At $k = 0$, $\xi_j = \frac{j\pi a}{2\ell}$, so that $\sin\xi_j = \pm 1$, $\cos\xi_j = 0$, and

$$\left.\frac{\partial\omega_j}{\partial k_1}\right|_{k_1=0} = \frac{1}{m\omega_j\ell}\ . \tag{3.128}$$

The derivative of ω_j^2 may be set equal to α_{1j}, as Equation 3.19 states. Then

$$\alpha_{1j} = \left.\frac{\partial({\omega_j}^2)}{\partial k_1}\right|_{k_1=0} \tag{3.129}$$

$$= 2\omega_j\left.\frac{\partial\omega_j}{\partial k_1}\right|_{k_1=0} \tag{3.130}$$

$$= 2\frac{\omega_j}{m\omega_j\ell} = \frac{2}{m\ell} \tag{3.131}$$

With the values of k_{Aj}, k_{B1j}, and α_{1j} derived above, the compliant model is used to estimate the first three frequencies of the spring-rod system for various values of k_1. These estimates are compared to the results of a Matlab program which finds the frequencies by iteratively solving Equation 3.113. Table 3.1 shows the values of the variables used for this example, and the Matlab program is listed in Appendix A. Figures 3.8 through 3.10 compare the results for modes 1 through 3, along with the percent error of the compliant model estimate compared to the "exact" solution. The compliant model gives a reasonably close estimate of frequency shift in all three cases. For higher-frequency modes, maximum error occurs at higher values of k_1.

Table 3.1: Parameters for Continuous Rod Example

Parameter	Value
m	3
ℓ	1
a	5

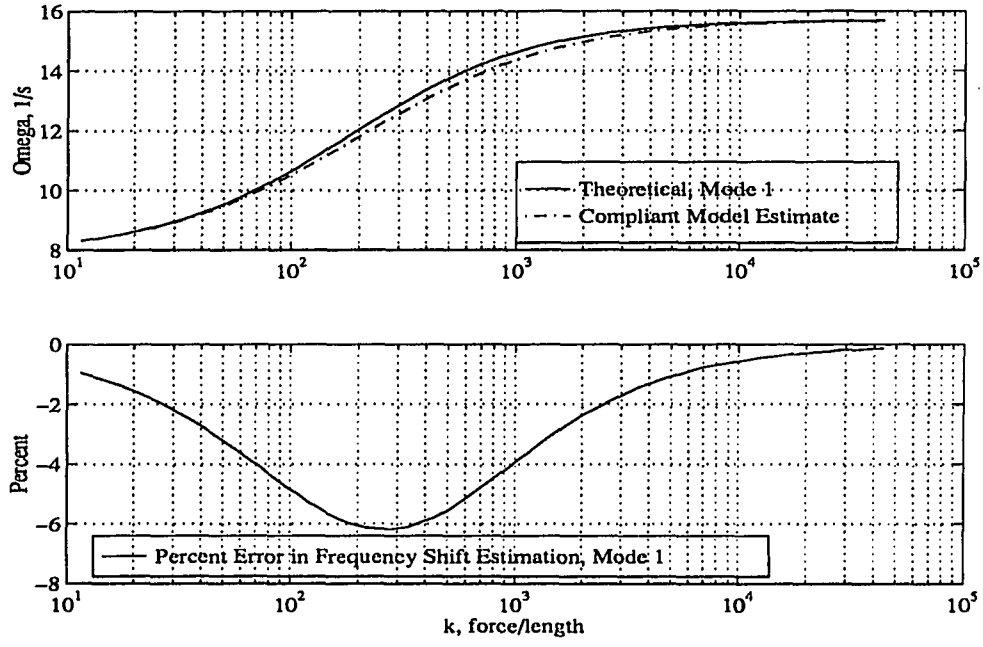

Figure 3.8: Comparison of Frequency Shift and Estimate for Mode 1

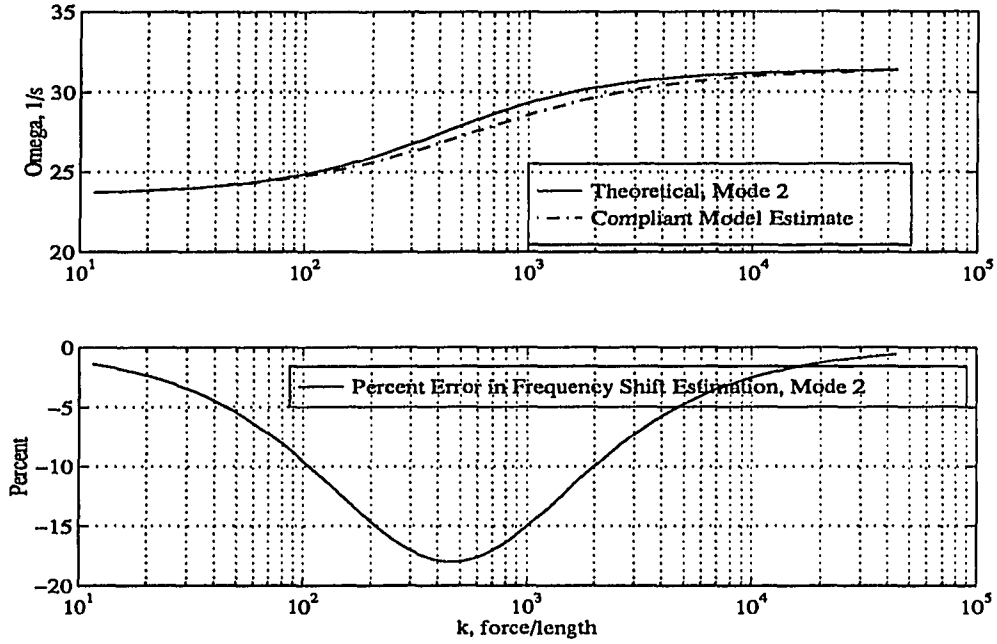

Figure 3.9: Comparison of Frequency Shift and Estimate for Mode 2

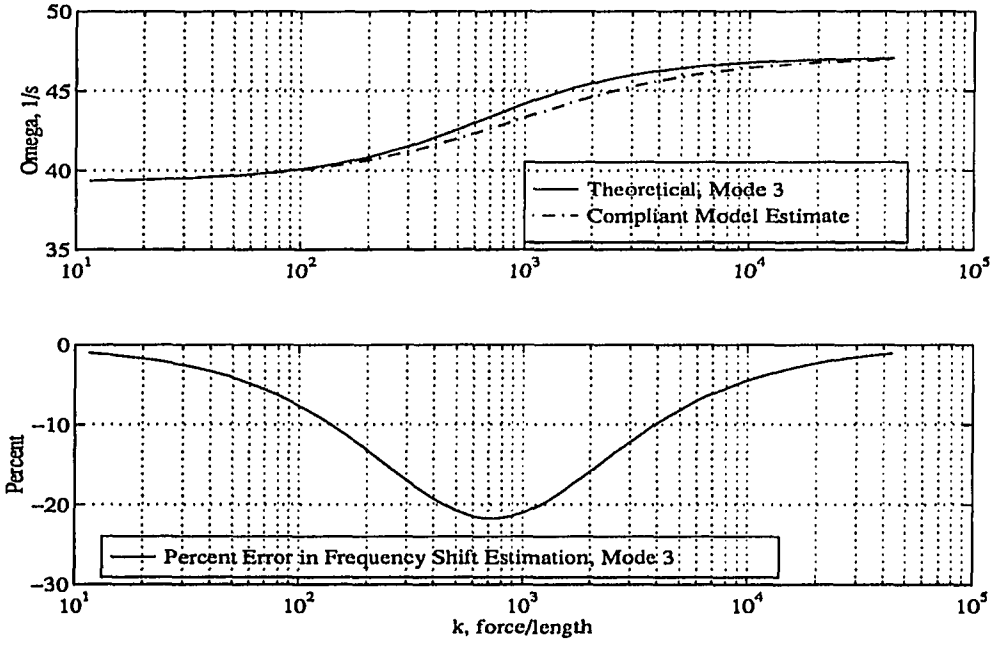

Figure 3.10: Comparison of Frequency Shift and Estimate for Mode 3

Next, the compliant model is used to predict changes in eigenvalues produced by inserting a damper.

3.6.1.2 Rod Acted upon by a Damper Next, a damper is placed on the end of the continuous rod, as shown in Figure 3.11. The differential equation of the rod remains the same as in the previous case, but the right-hand boundary is now acted upon by a velocity-dependent force. The solution of the equation under these conditions is

$$\tanh\left(\frac{\lambda\ell}{a}\right) = -\frac{ma}{c} \equiv -Q \tag{3.132}$$

where $\lambda \equiv \sigma + i\omega$ is the complex eigenvalue. Note this equation is very similar in form to Equation 3.113. Converting the hyperbolic function into exponential form results in

$$e^{\frac{2\ell\sigma}{a}} e^{\frac{2i\ell\omega}{a}} = \frac{Q-1}{Q+1} \,. \tag{3.133}$$

When $Q < 1$,

$$\frac{\ell\sigma}{a} = \ln\left(\sqrt{\frac{1-Q}{1+Q}}\right) \tag{3.134}$$

$$\frac{\ell\omega}{a} = \pi(j - \frac{1}{2}) \,. \tag{3.135}$$

When $Q > 1$,

$$\frac{\ell\sigma}{a} = \ln\left(\sqrt{\frac{Q-1}{Q+1}}\right) \tag{3.136}$$

$$\frac{\ell\omega}{a} = \pi j \,. \tag{3.137}$$

where j is the mode number.

Note that as c crosses the point where $Q = 1$, there is inevitably a value at which $\sigma \to -\infty$ and the imaginary part ω is undefined. The value of ζ is found at each

value of c by

$$\zeta = \frac{-\sigma}{\sqrt{\sigma^2 + \omega^2}} . \tag{3.138}$$

The plot for mode 1 is shown in Figure 3.12. Note that the peak at $c = ma$ is very sharp. The compliant model estimate computed by Equation 3.8 cannot match this shape. The root of the equation of the compliant model,

$$s^2 + k_A + \frac{\alpha_{ij} c s k_{sij}}{\alpha_{ij} c s + k_{sij}} = 0 \tag{3.139}$$

for the first mode j and the only damper i is also plotted in Figure 3.12. It is a closer estimate of the actual damping ratio because the compliant model estimate of Equation 3.8 is based on the assumption that $s \approx i\omega$. The Matlab program *cdamp.m* used to generate Figure 3.12 is listed in Appendix A.

3.6.1.3 Continuous Rod Acted upon by Compliant Elements However, any realistic structural element will act upon a structure in such a way that it can be modeled in series with a compliance. Therefore, the influence of a spring damper in series with a compliant spring acting upon an axially-vibrating rod is studied. Such a system is shown in Figure 3.13.

Again, the same differential equation governs the motion of the rod. However, this time the end condition is not as simple. The roots of the system can be expressed by

$$f \equiv \tanh\left(\frac{\lambda_j \ell}{a}\right) + \frac{am\lambda_j}{k} = 0 \tag{3.140}$$

where

$$k = \frac{k_i k_c}{k_i + k_c} \tag{3.141}$$

if the inserted element is a spring k_i, and

$$k = \frac{c_i \lambda_j k_c}{c_i \lambda_j + k_c} \tag{3.142}$$

if it is a dashpot c_i.

Since this equation is transcendental, solutions for various values of c_i may be found by a complex Newton method, where the complex derivative is

$$\frac{\partial f}{\partial \lambda_j} = \frac{\ell}{a} \frac{1}{\cosh^2\left(\frac{\lambda_j \ell}{a}\right)} - \frac{am}{{\lambda_j}^2 c_i} \tag{3.143}$$

in the case of an added damper c_i.

The Matlab program used to compute solutions in this problem is given in Appendix B. Results for this case are given in Figures 3.14 through 3.19. The parameters of the rod are the same as in the previous example; the stiffness of k_c is set to four times that of the rod.

The frequency shifts predicted by the compliant model much more closely approximate the theoretical solutions as Figures 3.14, 3.16, and 3.18 show. In the case of mode 1, Figure 3.15 reveals that the compliant model estimate of damping by Equation 3.8 is still inaccurate. This is due in part to the fact that this expression assumes that $s = i\omega$ to maintain simplicity. In such cases of high damping, this is not accurate. The actual root of the compliant model results in a much closer approximation to the actual damping ratio, as the same figure shows. Figures 3.17 and 3.19 show that the same pattern holds in modes 2 and 3. As peak damping ratio decreases, the simple expression of Equation 3.8 for compliant model loss factor becomes more and more realistic.

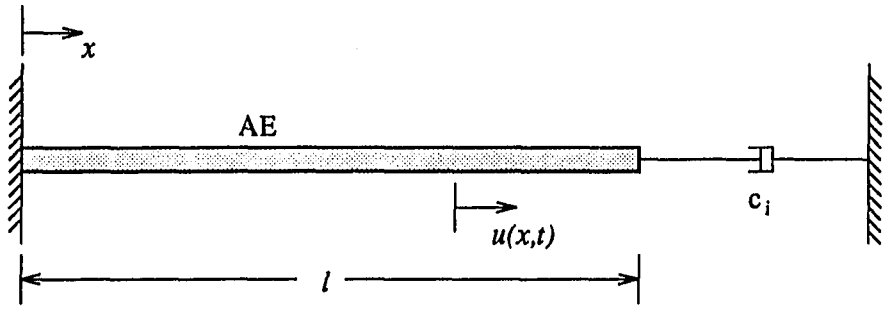

Figure 3.11: Continuous Rod Acted upon by a Viscous Damper

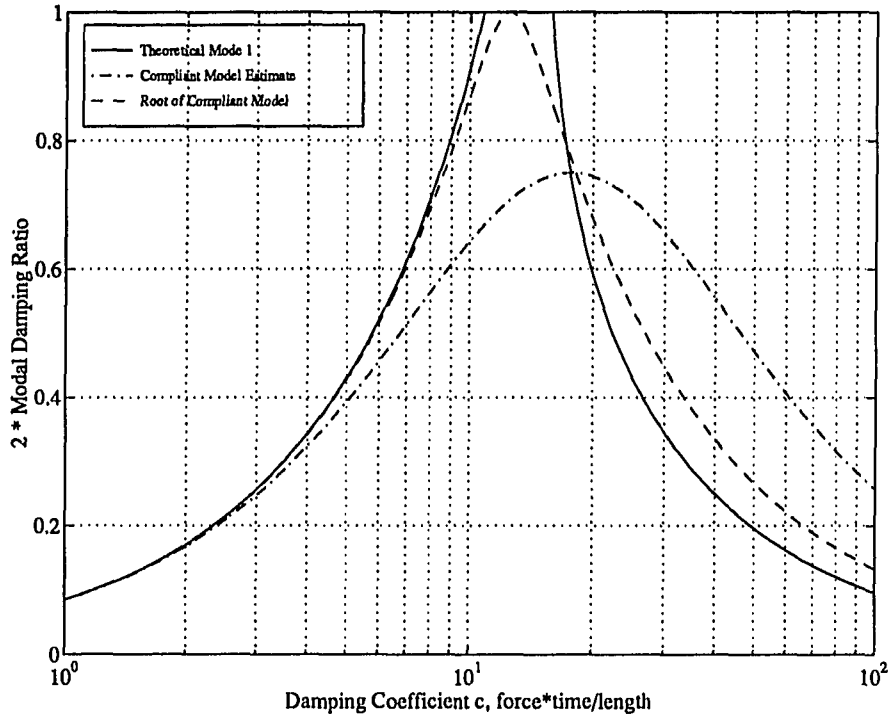

Figure 3.12: Theoretical and Compliant Damping Ratios for a Continuous Rod

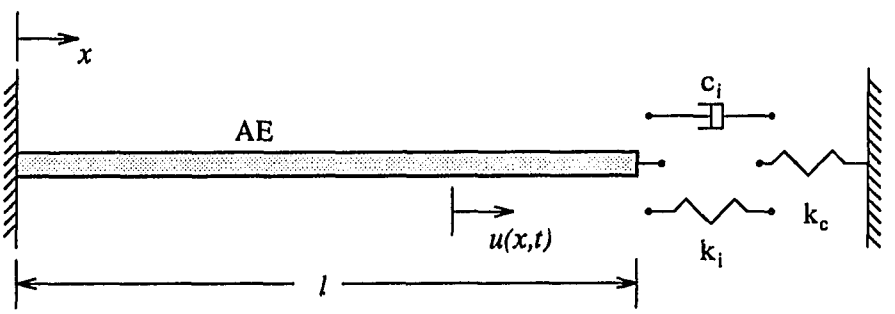

Figure 3.13: Continuous Rod Acted upon by an Element in Series with a Compliance

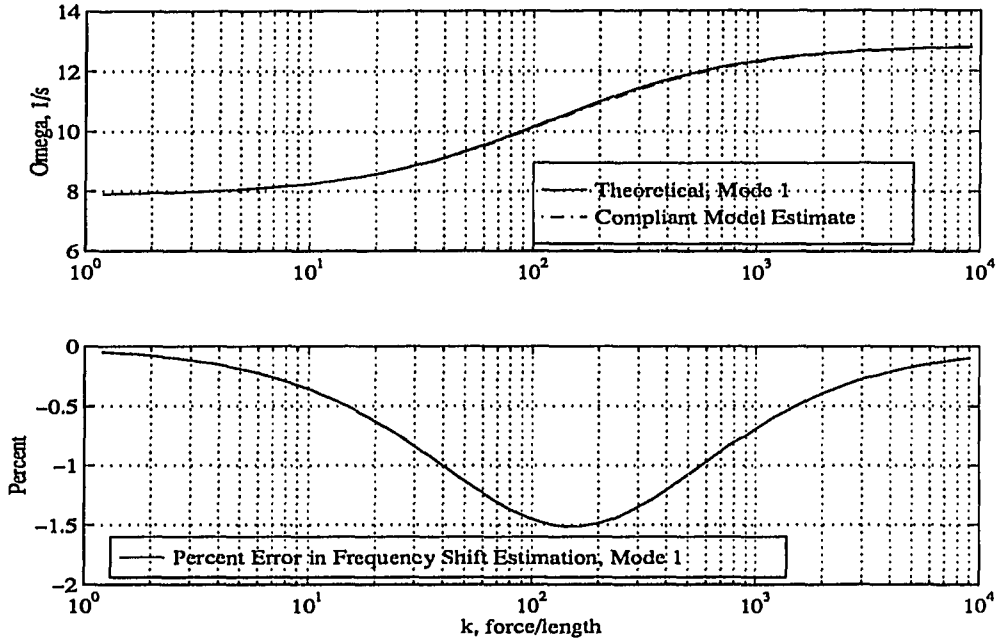

Figure 3.14: Frequency Shift of Mode 1 and Compliant Model Estimate for System of Figure 3.13 with Spring

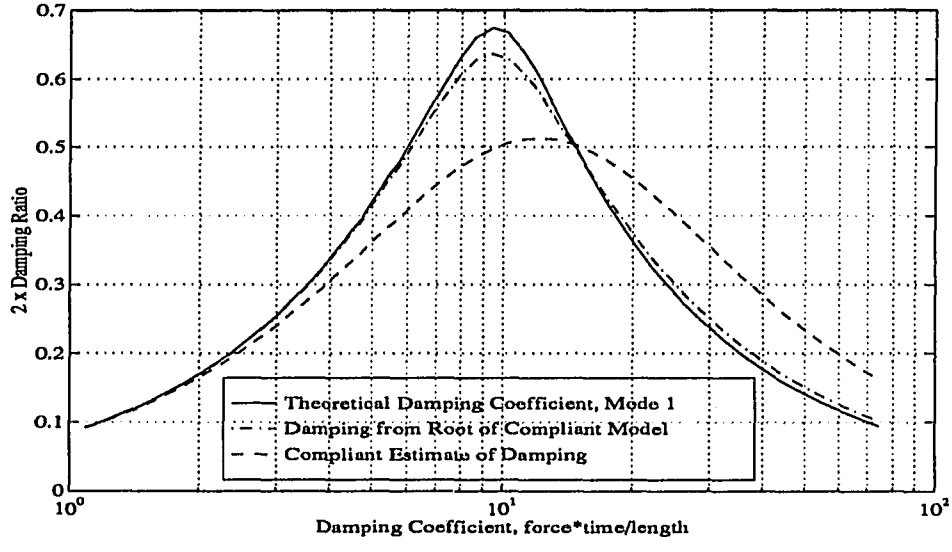

Figure 3.15: Damping Ratio of Mode 1 and Compliant Model Estimate for System of Figure 3.13 with Damper

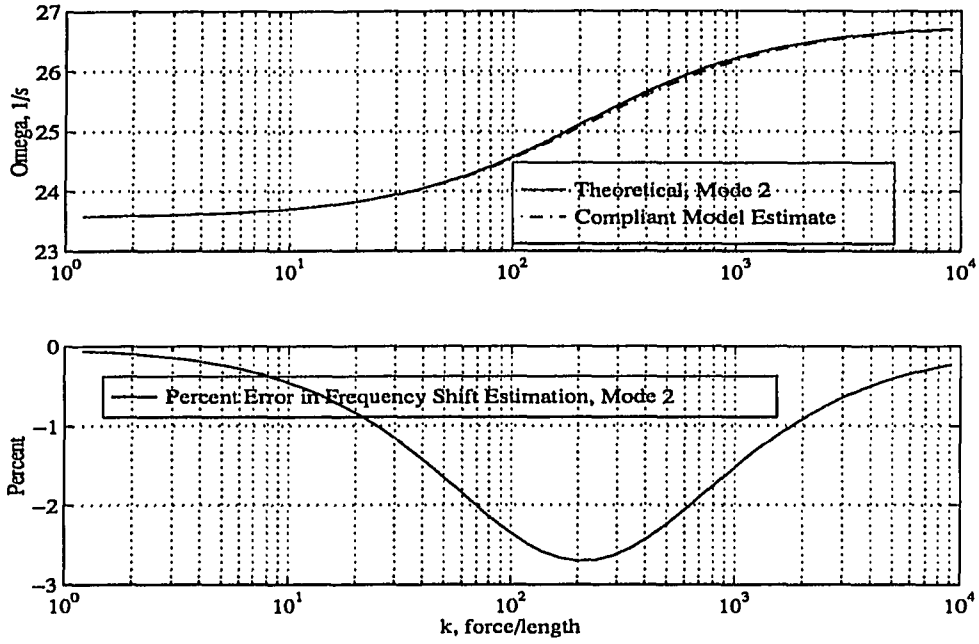

Figure 3.16: Frequency Shift of Mode 2 and Compliant Model Estimate for System of Figure 3.13 with Spring

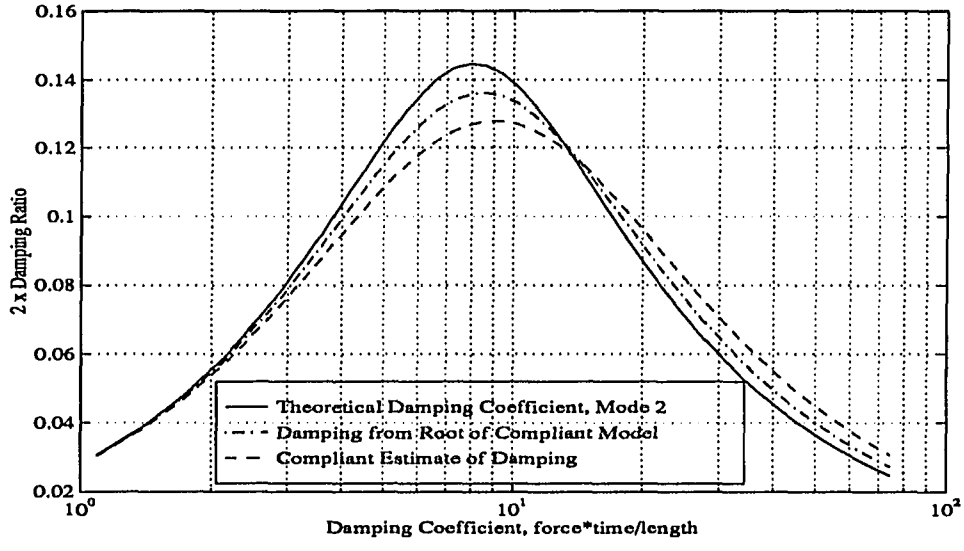

Figure 3.17: Damping Ratio of Mode 2 and Compliant Model Estimate for System of Figure 3.13 with Damper

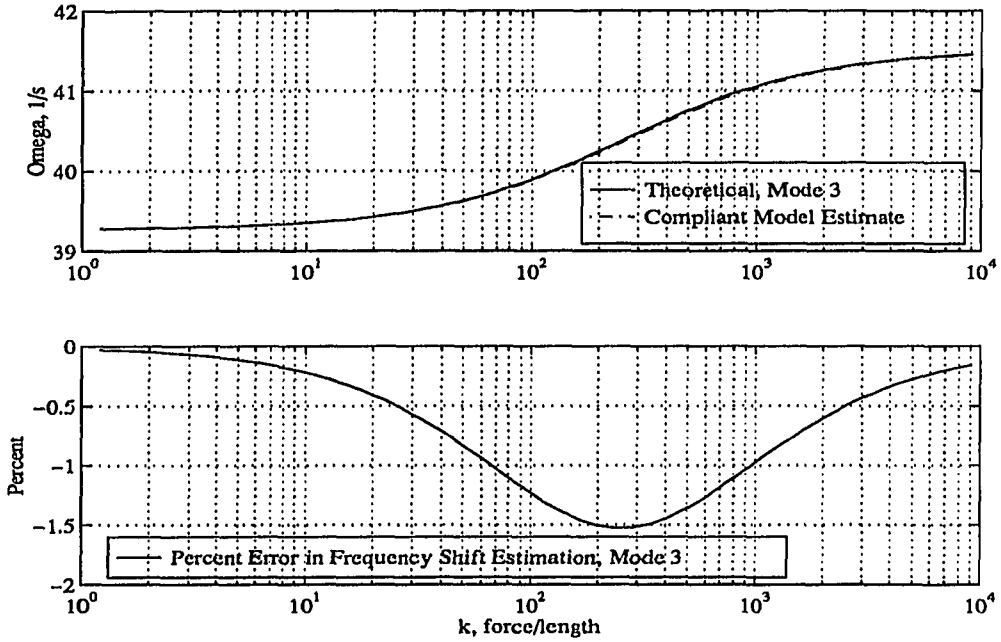

Figure 3.18: Frequency Shift of Mode 3 and Compliant Model Estimate for System of Figure 3.13 with Spring

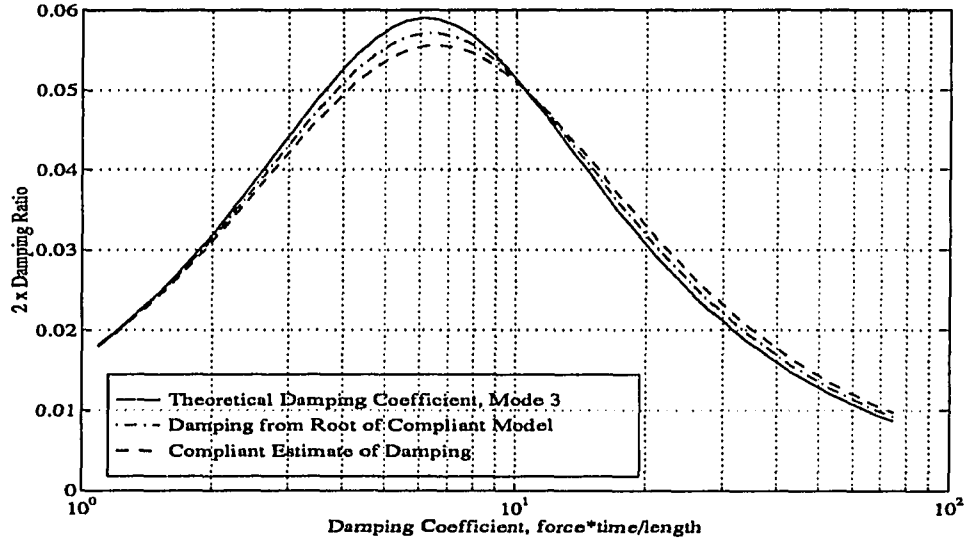

Figure 3.19: Damping Ratio of Mode 3 and Compliant Model Estimate for System of Figure 3.13 with Damper

3.6.2 One-Dimensional Lumped-Parameter System

In this section, the methods of estimating modal damping ratio will be applied to the one-dimensional lumped-parameter structure shown in Figure 3.20. The values of mass and stiffness for this structure are given in Table 3.2.

Table 3.2: Mass and Stiffness Values for the Lumped-Parameter System

Parameter	Value
k_{11}	0.5
k_{12}	0.5
k_{23}	1.0
k_{34}	1.0
k_{45}	0.5
k_{55}	0.5
k_{22}	1.0
k_{13}	0.7277
k_{24}	1.0
k_{35}	0.7277
k_{44}	1.0
k_{46}	0.6
k_{57}	0.6
m_1	2.0
m_2	2.0
m_3	2.0
m_4	2.0
m_5	2.0
m_6	1.0E-5
m_7	2.0E-5

In this example, damping struts are to be added between nodes 4 and 5 and between node 5 and the ground. Each damping strut consists of a spring in series with a viscous dashpot. Since there is a parallel stiffness present in the original model in both of these locations, the modal strain energy method may be used.

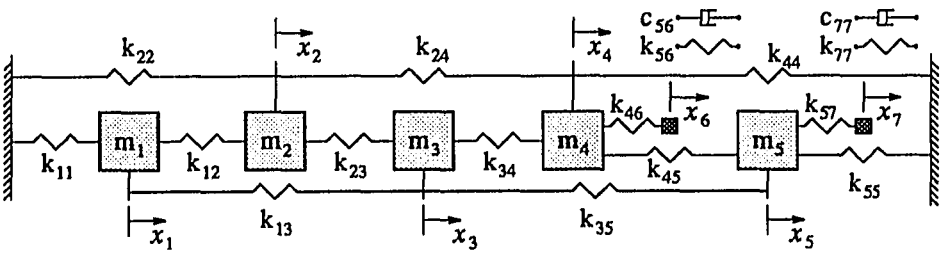

Figure 3.20: One-Dimensional Lumped-Parameter System

For the unmodified structure, the equations of motion are (see also Section 2.4.1)

$$[M\omega_j{}^2 - K]\phi_j = 0 \tag{3.144}$$

where ϕ_j represents the eigenvector,

$$M = \begin{bmatrix} m_1 & 0 & 0 & \dots & 0 \\ 0 & m_2 & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & m_7 \end{bmatrix} \tag{3.145}$$

and

$$K = \begin{bmatrix} k_{11} & & & & & & \\ +k_{12} & -k_{12} & -k_{13} & 0 & 0 & 0 & 0 \\ +k_{13} & & & & & & \\ & k_{12} & & & & & \\ & +k_{22} & & & & & \\ -k_{12} & & -k_{23} & -k_{24} & 0 & 0 & 0 \\ & +k_{23} & & & & & \\ & +k_{24} & & & & & \\ & & k_{13} & & & & \\ & & +k_{23} & & & & \\ -k_{13} & -k_{23} & & -k_{34} & -k_{35} & 0 & 0 \\ & & +k_{34} & & & & \\ & & +k_{35} & & & & \\ & & & k_{24} & & & \\ & & & +k_{34} & & & \\ 0 & -k_{24} & -k_{34} & +k_{44} & -k_{45} & -k_{46} & 0 \\ & & & +k_{45} & & & \\ & & & +k_{46} & & & \\ & & & & k_{35} & & \\ & & & & +k_{45} & & \\ 0 & 0 & -k_{35} & -k_{45} & & 0 & -k_{57} \\ & & & & +k_{55} & & \\ & & & & +k_{57} & & \\ 0 & 0 & 0 & -k_{46} & 0 & k_{46} & 0 \\ 0 & 0 & 0 & 0 & -k_{57} & 0 & k_{57} \end{bmatrix}. \quad (3.146)$$

Since this system has 7 nodes, there are 7 eigenvalues and eigenvectors. However,

since the masses at damper intermediate nodes 6 and 7 are set to very small values, only modes 1 through 5 emulate the low-frequency damped modes of a structure. These eigenvalues and eigenvectors from the normal mode analysis of the undamped system are presented in Table 3.3.

Table 3.3: Normal Modes of the Lumped-Parameter System

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
Eigenvalue, $\frac{1}{sec^2}$	0.2794	0.8201	1.2685	2.2937	2.2937
Eigenvector	0.3371	0.4925	0.3541	-0.0975	-0.0943
	0.2702	0.0861	-0.3333	0.4601	-0.3112
	0.3559	0.0000	-0.1647	0.0670	0.5845
	0.2702	-0.0861	-0.3333	-0.5186	-0.1990
	0.3371	-0.4925	0.3540	0.0736	-0.1140
	0.2702	-0.0861	-0.3333	-0.5186	-0.1990
	0.3371	-0.4925	0.3541	0.0736	-0.1140

3.6.2.1 Complex Eigenvalue Analysis of the Damped System In order to solve this system with added damping struts for modal damping by means of complex eigenvalue analysis, viscous dashpots are added to the model between nodes 6 and 5 and between node 7 and ground. Complex eigenvalues were computed as the identical damping coefficients of these two dashpots were varied between .01 and 10 units. The equations of motion become (see also Equation 2.26)

$$M\ddot{x} + C\dot{x} + Kx = [0] \,, \tag{3.147}$$

or

$$[\lambda_j{}^2\Phi^T M\Phi + \lambda_j\Phi^T C\Phi + \Phi^T K\Phi]q = [0] \tag{3.148}$$

in which the notation is the same as in Equation 2.16.

The square C matrix consists of all zeroes except

$$C(5,5) = C(6,6) = c_{56} \tag{3.149}$$

$$C(5,6) = C(6,5) = -c_{56} \tag{3.150}$$

$$C(7,7) = c_{77} \tag{3.151}$$

This problem may be solved by reducing it to a first-order system with matrix dimensions twice as large, as described in Section 2.4.2.2. The Matlab software used to solve this example has a function called *damp* which will find the modal damping in a system in the state variable form

$$\dot{z} = A_s z \tag{3.152}$$

where

$$z = \begin{Bmatrix} \dot{q} \\ q \end{Bmatrix} \tag{3.153}$$

and

$$A_s = \begin{bmatrix} -\Phi^T C \Phi & -\Lambda \\ I_{n\times n} & 0_{n\times n} \end{bmatrix}. \tag{3.154}$$

Here, n is the index of freedom of the system, and

$$\Lambda = \Phi^T K \Phi \tag{3.155}$$

is the diagonal matrix of eigenvalues of the undamped system.

3.6.2.2 Complex Eigenvalue Analysis Using Reduced Vector Space

For systems with many degrees of freedom, complex eigenvalue analysis becomes computationally expensive. To reduce this expense, the eigenvalues may be estimated

by solving the problem in a reduced vector space. Section 2.5.2 refers to this method [34]. In using this method, it is hoped that the complex eigenvectors of the damped structure may be expressed with sufficient accuracy by a linear combination of the members of a small subset of the eigenvectors of the undamped structure.

The displacements are represented by

$$x = \hat{\Phi}_{(m\times\ell)}\,\hat{q}_{(\ell\times 1)} \tag{3.156}$$

where $\hat{\Phi}$ is a subset of the vectors comprising Φ and $\ell < m$. Equation 3.148 is then replaced by

$$[\hat{\lambda}_j^2\hat{\Phi}^T M\hat{\Phi} + \hat{\lambda}_j\hat{\Phi}^T C\hat{\Phi} + \hat{\Phi}^T K\hat{\Phi}]_{(\ell\times\ell)}\,\hat{q}_{(\ell\times 1)} = [0] \tag{3.157}$$

or

$$[\hat{\lambda}_j^2 M_r + \hat{\lambda}_j C_r + K_r]\hat{q} = [0]\,. \tag{3.158}$$

In the above expression,

$$M_r = \hat{\Phi}^T M\hat{\Phi} = I_{(\ell\times\ell)}\,, \tag{3.159}$$

$$C_r = \hat{\Phi}^T C\hat{\Phi}\,, \tag{3.160}$$

and

$$K_r = \hat{\Phi}^T K\hat{\Phi} \tag{3.161}$$

is a diagonal matrix containing the eigenvalues corresponding to the vectors included in $\hat{\Phi}$.

As in Section 3.6.2.1, Equation 3.158 may be expressed in a form

$$\dot{\hat{z}} = A_r\hat{z} \tag{3.162}$$

where

$$\hat{z} = \begin{Bmatrix} \dot{\hat{q}} \\ \hat{q} \end{Bmatrix}, \tag{3.163}$$

and

$$A_r = \begin{bmatrix} -C_r & -K_r \\ M_r & 0_{\ell \times \ell} \end{bmatrix}_{(2\ell \times 2\ell)} . \tag{3.164}$$

A_r is the state coefficient matrix suitable for the Matlab *damp* function. In this example, $\hat{\Phi}$ includes eigenvectors corresponding to the five lowest frequencies.

3.6.2.3 Compliant Model Estimation To use the compliant model to estimate the modal damping in the lumped-parameter system, the method of frequency slopes of Section 3.2.3 is used. As has been shown before, modal stiffness is set equal to the modal frequency squared. Then α_{ij} are set equal to the derivatives of the eigenvalues with respect to a zero spring placed in each dashpot location i (that is, between nodes 4 and 6 and between nodes 5 and 7). As is shown in Section 3.2.3, α_{ij} for a system of axially-displaced elements is also equal to the square of the modal displacement across the location of the damper to be added.

The value of k_{si_j} is found by placing the stiffness which is to act in series with the dashpot across the damper location under study and observing the frequency shifts which occur in a new real eigenvalue analysis. In this case, the stiffness k_{46} is placed across nodes 4 and 5, and k_{57} is placed between node 7 and the ground. The results are given in Table 3.4.

3.6.2.4 Modal Strain Energy Method To apply the modal strain energy method to the example structure, the strain energy of each of the struts in the

Table 3.4: Parameters of the Compliant Model of a Lumped-Parameter System

		Strut 1		Strut 2	
Mode	k_{Aj}	α_{1j}	k_{s1j}	α_{2j}	k_{s2j}
1	0.2794	0.0045	0.0017	0.1137	0.0494
2	0.8201	0.1652	0.0532	0.2426	0.1276
3	1.2685	0.4724	0.2502	0.1253	0.1092
4	2.2937	0.3507	0.2890	0.0054	0.0041
5	2.2937	0.0072	0.0060	0.0130	0.0098

locations where damping is to be inserted is first computed. These struts, along with the added dashpot-spring combination, become the three-element damping strut like that described in Section 2.2. The loss factor of such a strut is evaluated by Equation 2.4. The overall modal loss factor is the summation of the loss factor of each such damping strut multiplied by its proportion of the total modal strain energy, as Equation 2.41 states.

The percent strain energy for a strut may be computed

$$\frac{V_{ij}}{V_j} = \frac{\phi_j{}^T K_i \phi_j}{\phi_j{}^T K \phi_j} \tag{3.165}$$

where K_i is the matrix containing only the contribution of k_i to the stiffness matrix. For the example being studied, these values are given in Table 3.5.

Table 3.5: Proportions of Modal Strain Energies for a Lumped-Parameter System

V_{ij}/V_j					
	Mode j				
Strut i	1	2	3	4	5
1	0.0080	0.1007	0.1862	0.0764	0.0016
2	0.2034	0.1479	0.0494	0.0012	0.0028

3.6.2.5 Compliant Model Based upon Perturbation The values of k_{Bij} or k_{sij} in the compliant model may be obtained by observing the frequency shift which occurs in an eigenvalue analysis of a modified structure. This frequency shift could be estimated by a first-order perturbation method in order to avoid the additional eigenvalue analysis. This method is also used to analyze the lumped-mass system.

Once the original normal mode analysis is performed, α_{ij} is obtained from the original eigenvectors (eigenvalue derivatives as in the method of frequency slopes (see Section 3.2.3). The values of k_{sij} for the compliant model were then found by the first-order perturbation of Equation 2.50 in Section 2.5.5. Since this is based on the original normal mode analysis, no additional eigenvalue analysis is required; however, no new information about the system is obtained.

The values of the compliant model parameters obtained in this manner for the lumped-parameter system are given in Table 3.6.

Table 3.6: Parameters of the Compliant Model Based on First-Order Perturbation

		Strut 1		Strut 2	
Mode	k_{Aj}	α_{1j}	k_{s1j}	α_{2j}	k_{s2j}
1	0.2794	0.0045	0.0027	0.1137	0.0682
2	0.8201	0.1652	0.0991	0.2426	0.1456
3	1.2685	0.4724	0.2835	0.1253	0.0752
4	2.2937	0.3507	0.2104	0.0054	0.0033
5	2.2937	0.0072	0.0043	0.0130	0.0078

3.6.2.6 Complex Perturbation Method As is developed in Section 2.5.5, each spring-dashpot combination added to a structure may be looked at as a complex element which perturbs the eigenvalues of the system in the complex plane. This approach is also used to estimate the damping in the lumped-parameter structure.

Since there is more than one damper being added in this case, the first-order perturbation results in

$$s_j = s_j(0) + \sum_{i=1}^{2} \epsilon_{ij} s_{ij}^{(1)} \tag{3.166}$$

where $s_j(0)$ represents the wholly imaginary root of the unperturbed system,

$$\epsilon_{ij} = \frac{s_j c_i k_i}{s_j c_i + k_i} , \tag{3.167}$$

and

$$s_{ij}^{(1)} = -\frac{\phi_j{}^T B_i \phi_j}{2 s_j \phi_j{}^T \phi_j} . \tag{3.168}$$

In the above equation, B_i is the matrix of changes for strut i only (see also Section 2.5.5).

The estimate for loss factor is

$$\eta_j \approx \frac{\sum_{i=1}^{2} \frac{\kappa_{pij}}{2} \frac{r_{ij}}{r_{ij}{}^2+1}}{1 + \sum_{i=1}^{2} \frac{\kappa_{pij}}{2} \frac{r_{ij}{}^2}{r_{ij}{}^2+1}} \tag{3.169}$$

where

$$\kappa_{pij} = k_i \frac{\phi_j{}^T B_i \phi_j}{\omega_j{}^2 \phi_j{}^T \phi_j} \tag{3.170}$$

and

$$r_{ij} = \frac{\omega_j c_i}{k_i} \tag{3.171}$$

and the rest of the notation is in accord with Section 2.5.5.

3.6.2.7 Spherical Model Damping Estimation The values of k_{sij} from the compliant model of the lumped-parameter system are reapportioned by means of the spherical model of Section 3.3.

Starting with the values of k_{Aj} for each mode and α_{ij} and k_{sij} for each strut, the locations of the dampers in the model are deduced, that is, the values of the cosine

vectors γ_i are found. To maintain simplicity, it is assumed that all of the cosines are positive in sign.

The k_{Aj} of each mode is assembled into the ${}_oK_o$ matrix. Then the matrix $\frac{1}{2}\sum_{i=1}^{2} k_{sij}\gamma_i\gamma_i{}^T$ is added to this matrix in order to emulate the reorientation of the modes within the spherical model at the values of c_i at which modal damping is maximum. The matrix ${}_o\Phi$ containing the orientation of the new modes is found. Then the values of k_{sij} in each of these directions are summed in order to arrive at ${}_ok_{sij}$. Loss factor η_j is estimated using Equation 3.57, with

$$ {}_o\kappa_{sij} = \frac{{}_ok_{sij}}{{}_ok_{Aj}} \tag{3.172} $$

and

$$ r_{ij} = \frac{\alpha_{ij}\, c_i\, \omega_j}{k_{sij}} \, . \tag{3.173} $$

The values of ${}_ok_{Aj}$ and ${}_ok_{sij}$ are given in Table 3.7 below, along with α_{ij}, which are the same as for the compliant model.

Table 3.7: Parameters of the Spherical Model of a Lumped-Parameter System

		Strut 1		Strut 2	
Mode	${}_ok_{Aj}$	α_{1j}	${}_ok_{s1j}$	α_{2j}	${}_ok_{s2j}$
1	0.2794	0.0045	0.0000	0.1137	0.0478
2	0.8201	0.1652	0.0265	0.2426	0.1013
3	1.2685	0.4724	0.2629	0.1253	0.1238
4	2.2937	0.3507	0.0002	0.0054	0.0069
5	2.2937	0.0072	0.3103	0.0130	0.0202

3.6.2.8 Damping from Eigenvalues of the Spherical Model The damped spherical model equation form of Section 3.3.6 is used for eigensolution and damping

output by the *damp* function of Matlab. For the lumped parameter example, all γ_{ij} are considered to be positive.

3.6.2.9 Results of Computations for the Lumped Example The Matlab program used to implement the methods discussed above for the lumped-parameter example system is listed in Appendix C. Figures 3.21 through 3.26 show twice the modal damping ratios computed by direct complex eigenvalue analysis, complex eigenvalue analysis using a reduced vector space, the compliant model, the modal strain energy method, the compliant model based upon perturbation, the complex perturbation method, and the spherical compliant model. Direct complex eigenvalue analysis provides the benchmark for comparison.

Complex eigenvalue analysis using a reduced vector space accurately predicts damping for low values of c. However, for higher values of c, damping is overestimated. This is due to the inability of the first five eigenvectors to model significant displacement in the series springs k_{56} and k_{77}. Since these springs appear overly stiff, predicted damping is too high. The augmented reduced space of Yiu and Weston [34] alleviates this problem, but because it appears somewhat more difficult to implement in a large system, it is not pursued further in this dissertation.

All other estimations for mode 1 (Figure 3.21) were good, except the compliant model based on perturbation. The compliant model estimate and the modal strain energy estimate are particularly accurate.

The damping in mode 2 (Figure 3.22) is difficult to estimate. The peak of each damping estimate occurs to the right of the true peak. All methods predict overly high damping ratios, except for the spherical model, which is too low.

The damping of mode 3 (Figure 3.23), which is the highest of the first five modes, is also difficult to estimate. All methods fall short of the actual damping peak, but the spherical model has the most accurate peak. The pattern of error for the spherical model and for the compliant models with and without perturbation is similar to that of the compliant model estimate of mode 1 in the continuous rod example (Figure 3.15).

Since modes 4 and 5 occur at the same frequency, they are plotted together. Figure 3.24 is scaled for the mode which is more highly damped. All modes fall short of the true peak damping, but the spherical model is strikingly accurate given the complicated shape of this mode. In this mode and mode 3 the modal strain energy and complex perturbation estimates fall far below the true damping values.

The remaining mode is lightly damped (Figure 3.25). The complex perturbation and compliant model with perturbation proved especially accurate estimates, but no method resulted in a large error in absolute terms.

The compliant model estimate of the summation of all damping ratios is the most accurate (Figure 3.26). However, the spherical model estimate is nearly as good. The compliant model estimate based on perturbation estimates the sum well, even though its estimate of most individual modes is far off. This may be related to the fact that the sum of the diagonal elements of the modified structure's system matrix must equal the sum of its eigenvalues. The similarity transformation performed on this matrix by the old normal modes does not change this sum.

From this example, it is apparent that the compliant and spherical model estimates are able to predict damping in multiple, closely spaced modes relatively well.

Figure 3.27 shows the damping found by solving the damped spherical system

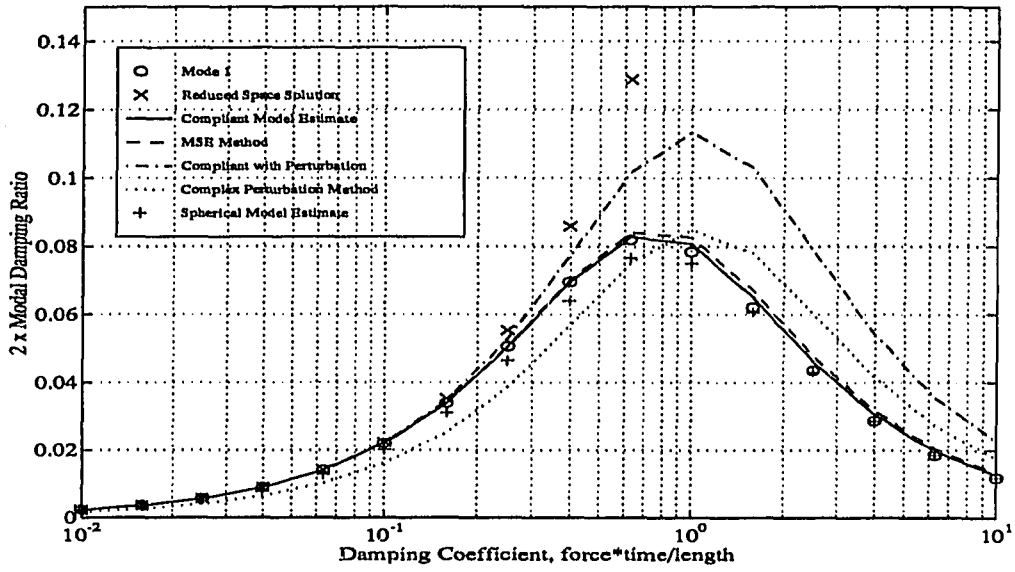

Figure 3.21: Damping and Estimations for Mode 1 of a Lumped-Parameter System

of Equation 3.83 directly using the *damp* command in Matlab. All five modes are estimated very closely by this method. In this case, the spherical model provides quite an accurate representation of the structure's response to localized damping.

3.6.3 SPICE Bulkhead

3.6.3.1 Bulkhead with One Damping Strut In order to check the accuracy of the compliant model in predicting modal damping, the effect of placing one damping strut into the NASTRAN model of the SPICE lab version of the bulkhead (translated from the Patran file blkhcorr.pat) is studied. This model is shown in Figure 2.2.

The damper is inserted between nodes 51 and 52 of the bulkhead. Since this position spans across two corners of an octahedral cell near the edge of the bulkhead,

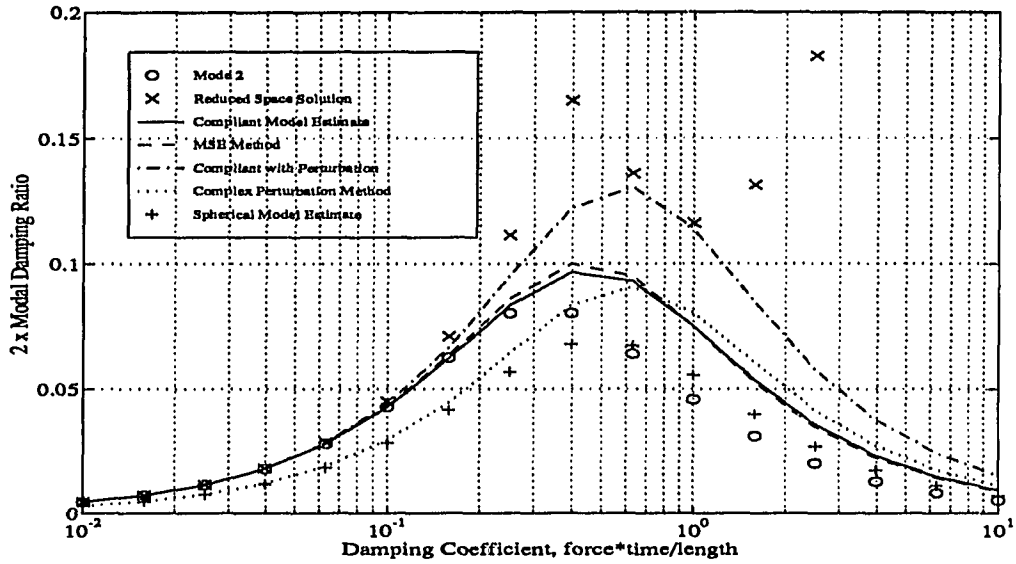

Figure 3.22: Damping and Estimations for Mode 2 of a Lumped-Parameter System

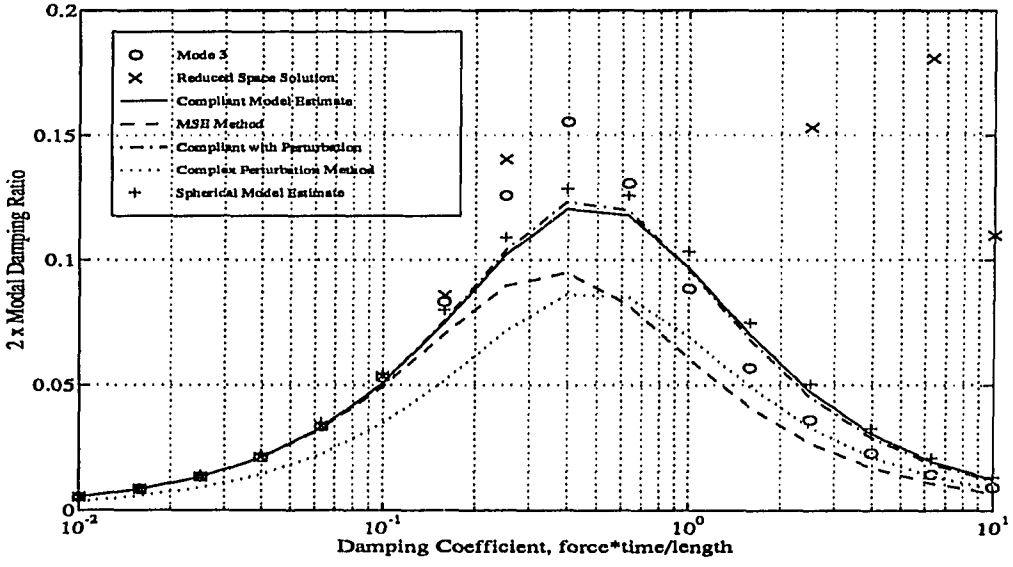

Figure 3.23: Damping and Estimations for Mode 3 of a Lumped-Parameter System

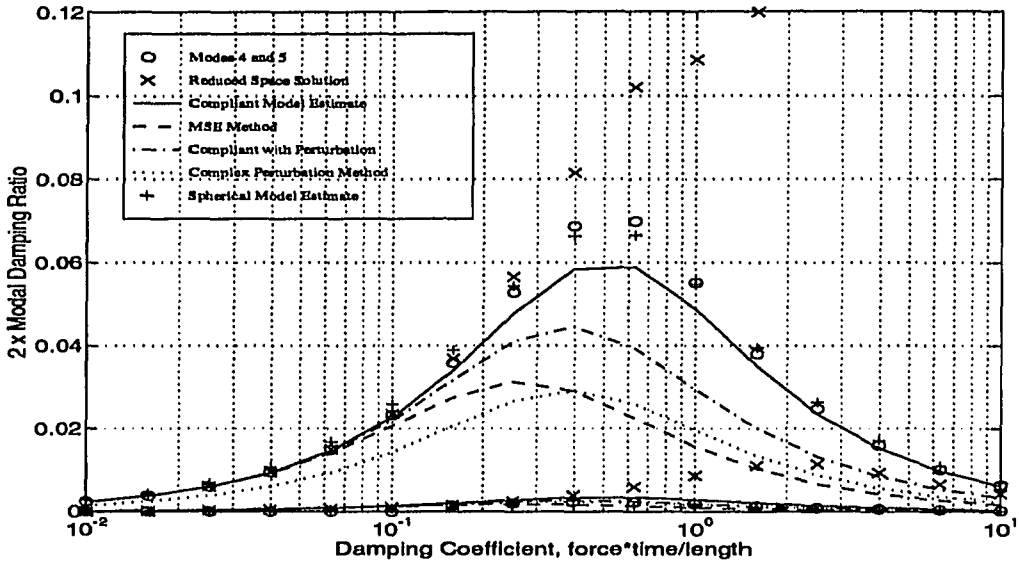

Figure 3.24: Damping and Estimations for Modes 4 and 5 of a Lumped-Parameter System

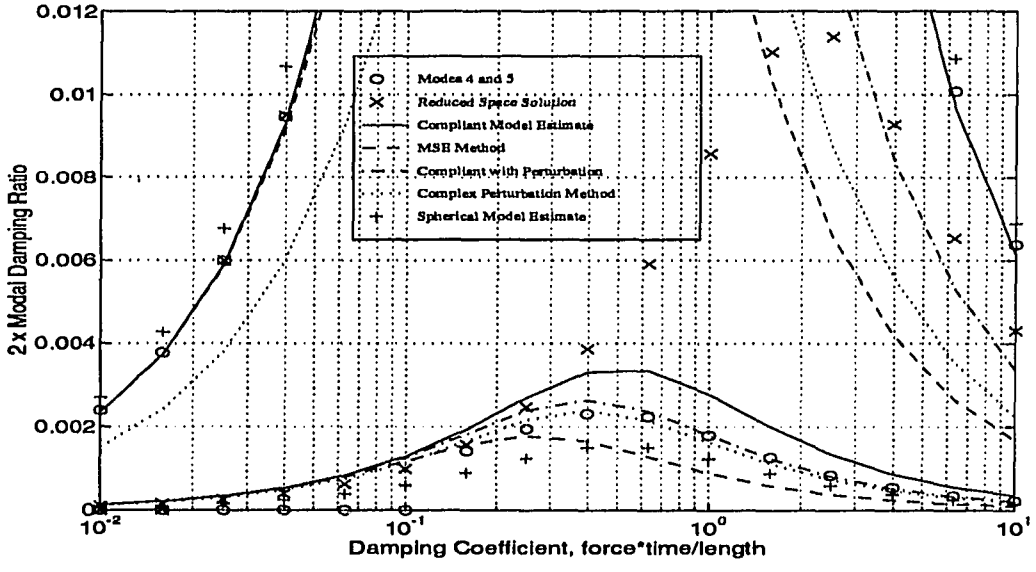

Figure 3.25: Damping and Estimations for More Lightly Damped of Modes 4 and 5

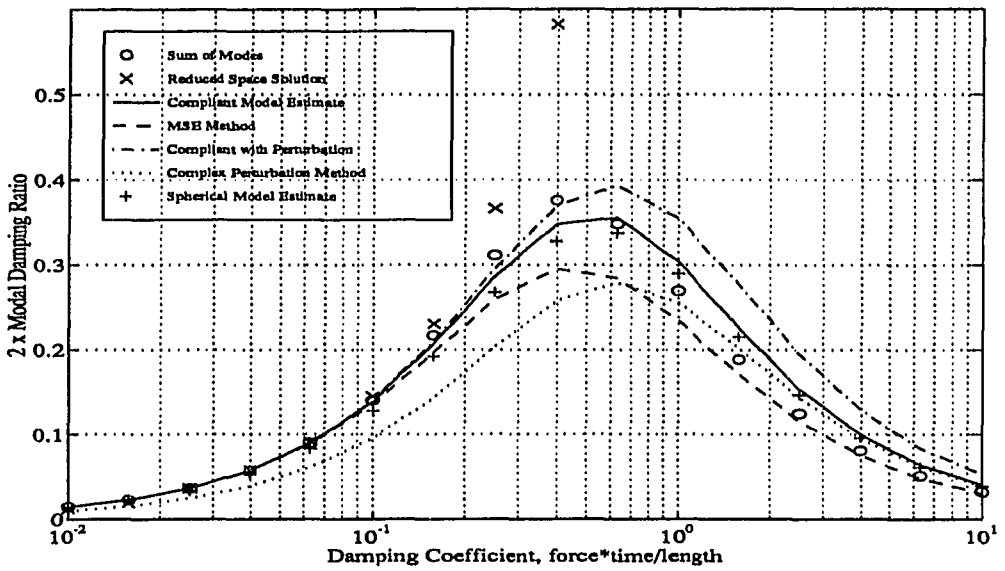

Figure 3.26: Sum of Damping and Estimations for Modes 1 through 5 of a Lumped-Parameter System

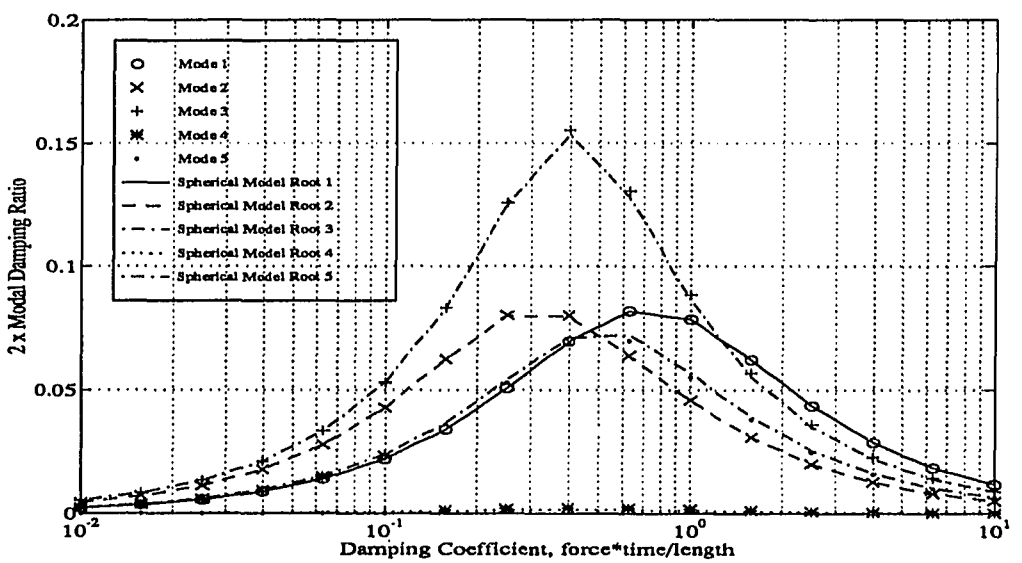

Figure 3.27: Damping Estimated by Finding the Eigenvalues of the Spherical Model of a Lumped-Parameter Structure

this is a diagonal damper.

The values of κ_{sij} and α_{ij} are obtained using the method of frequency slopes by considering the strain energy in a spring of stiffness $k_i = 2\ N/m$. For repeated roots, the sum of the values of α_{ij} from a given frequency is allocated to the mode with the highest value of κ_{sij}. Other α_{ij} at that frequency are set to zero. Using the compliant model, the estimated values of modal loss factor for 13 modes given in Table 3.8 are computed as a function of damping coefficient c of the strut and plotted in Figure 3.28.

Table 3.8: Undamped Modes of the SPICE Bulkhead

Mode	Frequency[a], Hz
1	54.36
2	54.36
3	82.38
4	87.83
5	106.52
6	119.39
7	119.40
8	132.71
9	132.72
10	144.69
11	144.69
12	148.09
13	148.09

[a]Digits beyond the decimal point are provided for the comparison of closely-spaced modes.

For comparison purposes, modal damping coefficients (2ζ) are also computed using NASTRAN. These results are plotted as symbols. The values computed by the compliant model correlate to the "correct" values computed by complex eigenvalue

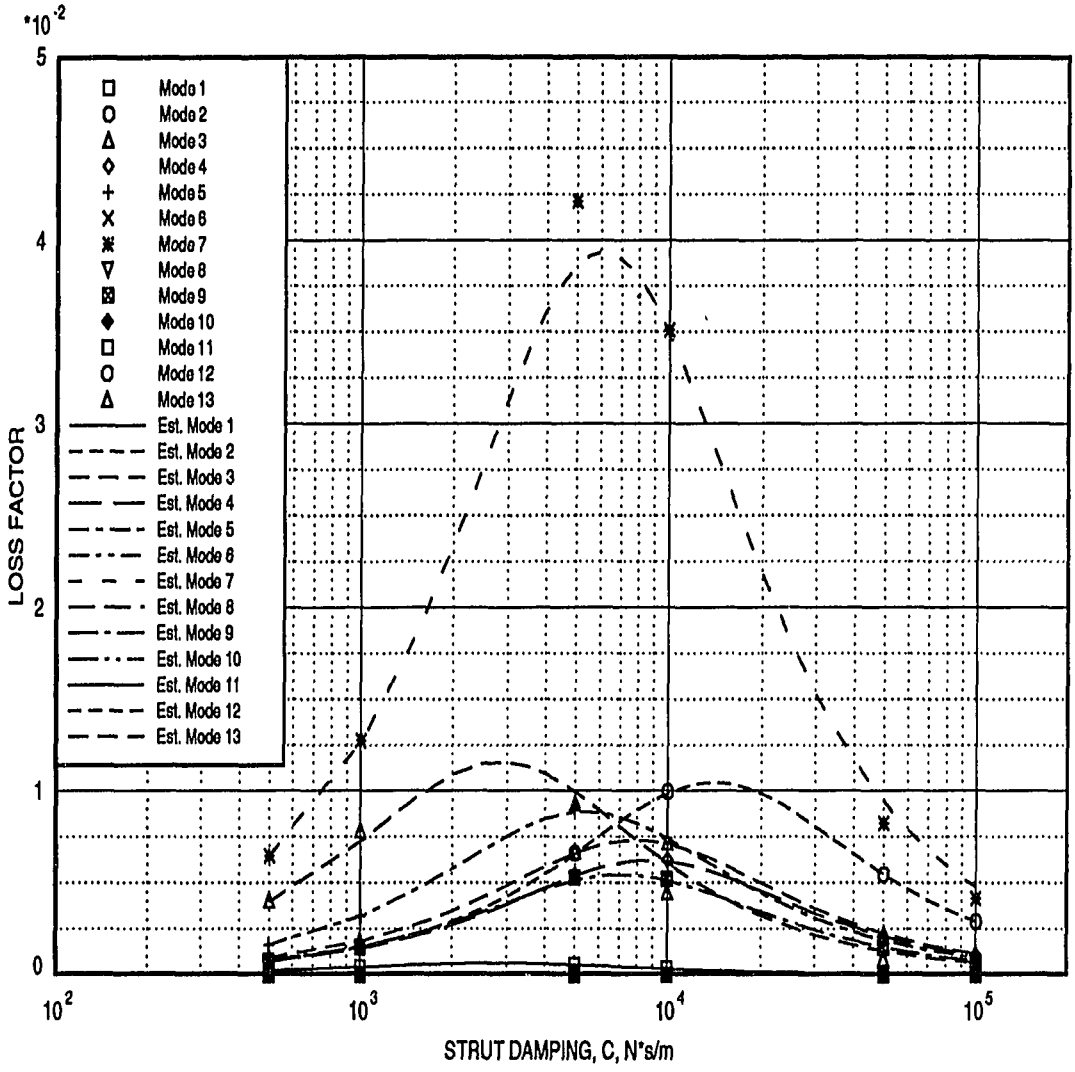

Figure 3.28: Modal Loss Factor Due to Damper in Location 5152

analysis with a high degree of accuracy.

3.6.3.2 Bulkhead with Five Damping Struts To see how well the compliant model predicts the modal damping achieved by multiple dampers, a similar test is run with five damping struts in the structure.

In-line damping struts are included between the node pairs 56-61, 6-18, 27-28, 28-34, and 29-30 of the SPICE lab finite element model of the bulkhead (see Figure 2.2).

Again, the values of κ_{si_j} and α_{ij} are found by the method of frequency slopes. This time, for modes of duplicated eigenvalues, the values of κ_{si_j} at a given frequency are re-allocated so that they are proportional to those of the corresponding α_{ij}. The compliant model is used to compute loss factor for each mode as a function of c, and the values are plotted in Figures 3.29 through 3.36.

The "correct" values of modal damping coefficient as computed by complex eigenvalue analysis are also shown. Modes 3, 4, and 5, which are not repeated eigenvalues, correlated well, but the modes at repeated eigenvalues are not predicted as accurately. Apparently, the damping is not distributed properly among each set of duplicated modes.

The spherical model also is used to solve the system based on the same values of k_{si} and α_{ij} obtained through the method of frequency slopes. In this case, the number of modes m is 13 and the number of struts n is 5. The matrix in Equation 3.83 used to find the eigenvalues of the damped spherical model is then 31×31. Although this is larger than in the lumped parameter example, it is still much smaller than the matrices associated with the finite element model. Since α_{ij} are obtained using

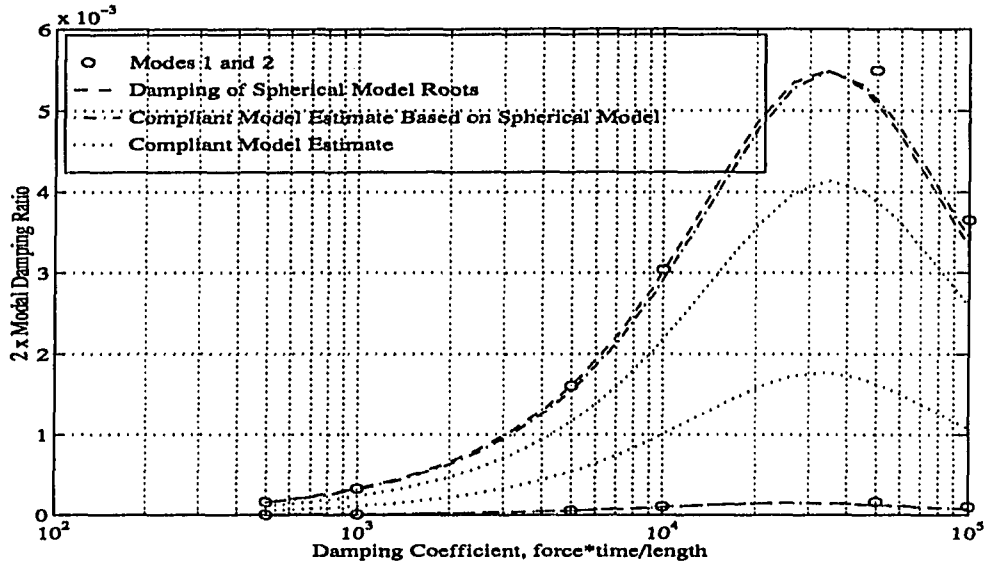

Figure 3.29: Damping and Estimations for Modes 1 and 2 of the Spice Bulkhead

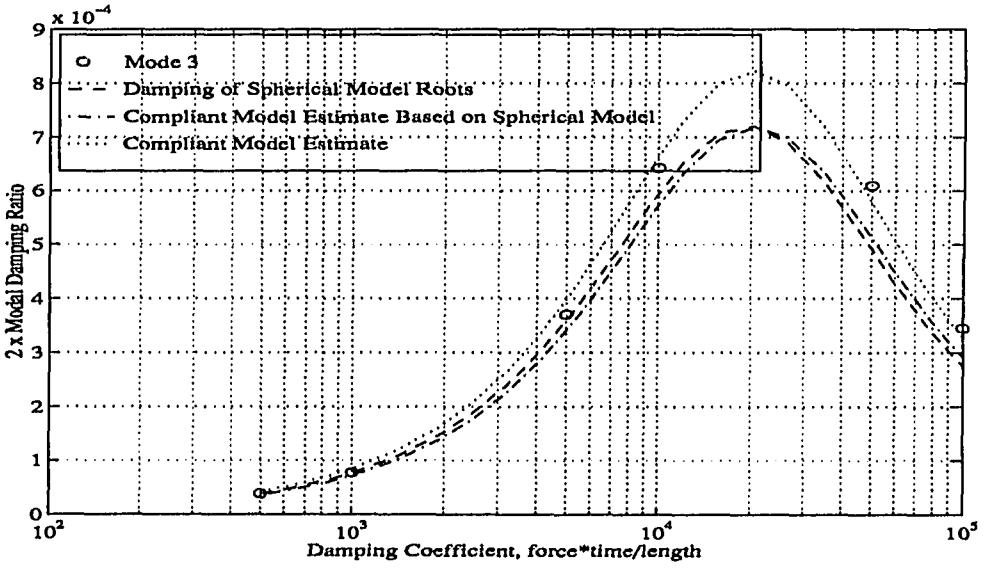

Figure 3.30: Damping and Estimations for Mode 3 of the Spice Bulkhead

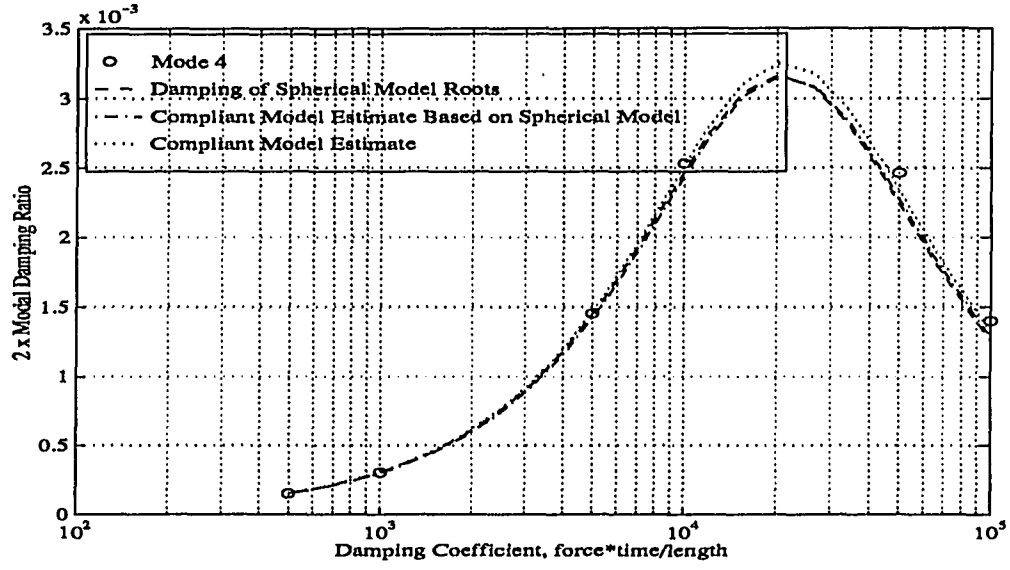

Figure 3.31: Damping and Estimations for Mode 4 of the Spice Bulkhead

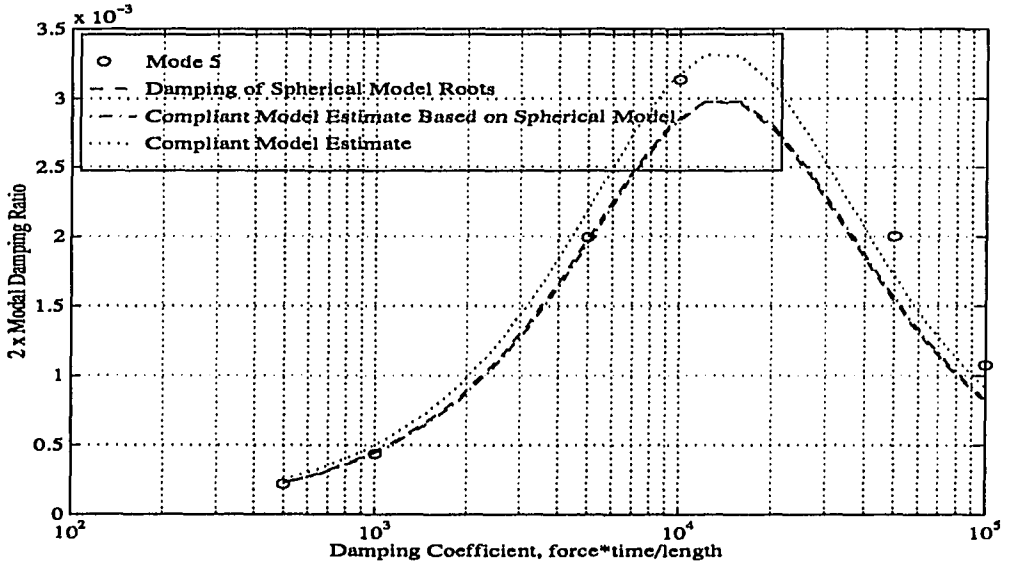

Figure 3.32: Damping and Estimations for Mode 5 of the Spice Bulkhead

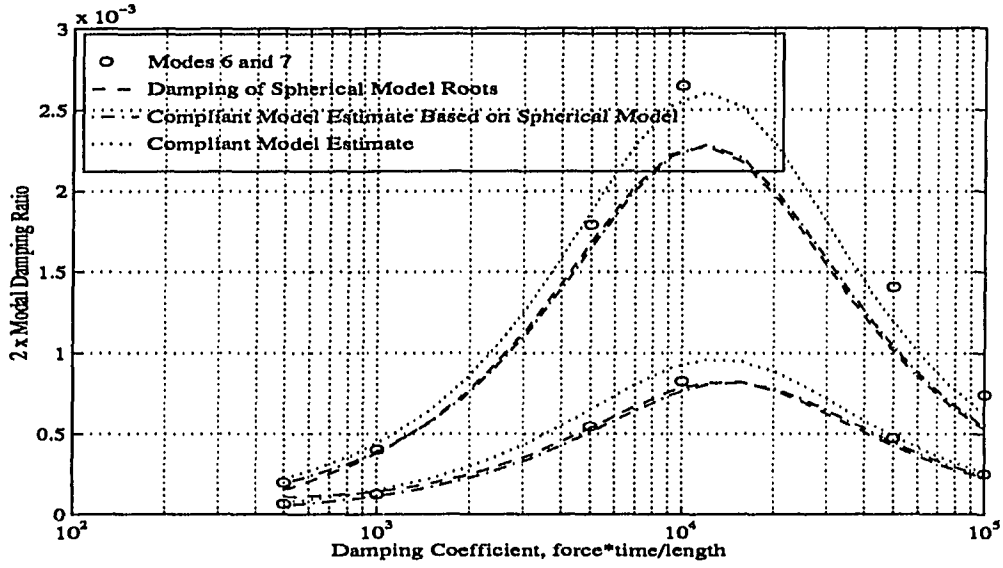

Figure 3.33: Damping and Estimations for Modes 6 and 7 of the Spice Bulkhead

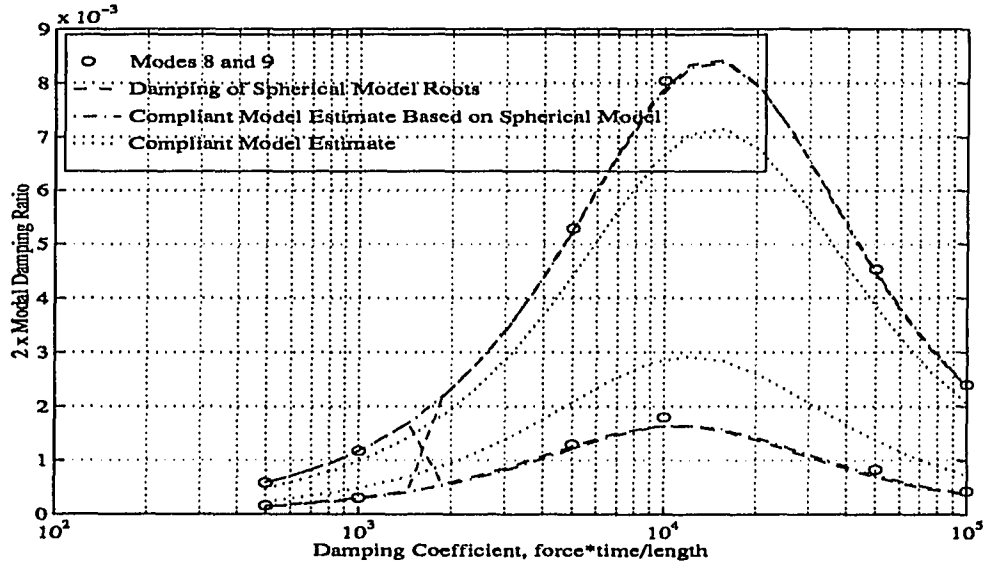

Figure 3.34: Damping and Estimations for Modes 8 and 9 of the Spice Bulkhead

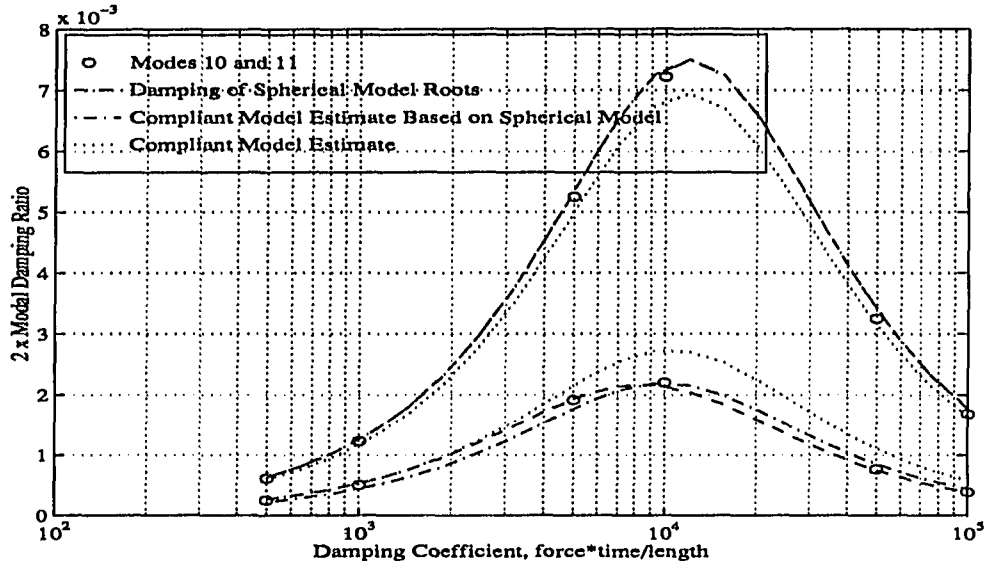

Figure 3.35: Damping and Estimations for Modes 10 and 11 of the Spice Bulkhead

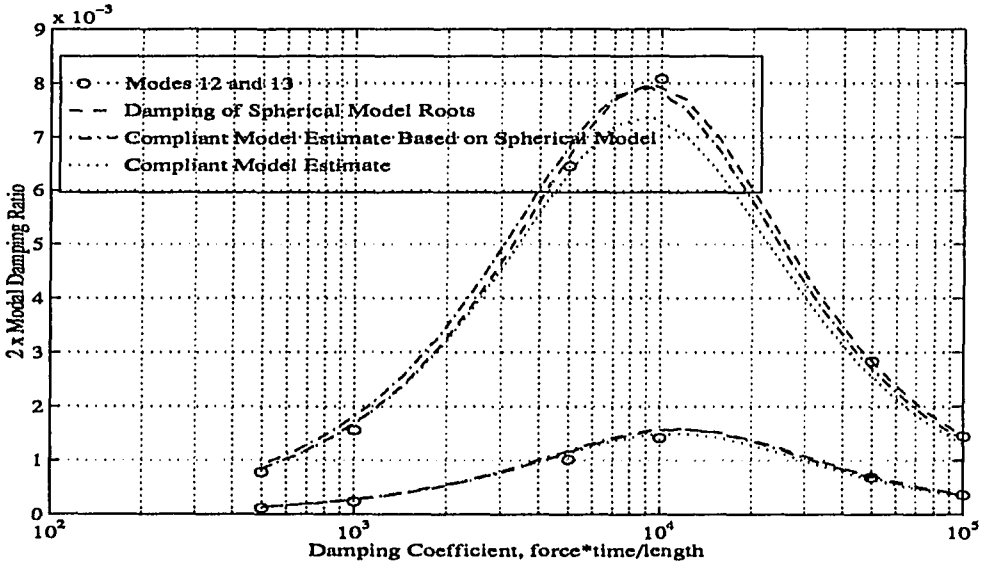

Figure 3.36: Damping and Estimations for Modes 12 and 13 of the Spice Bulkhead

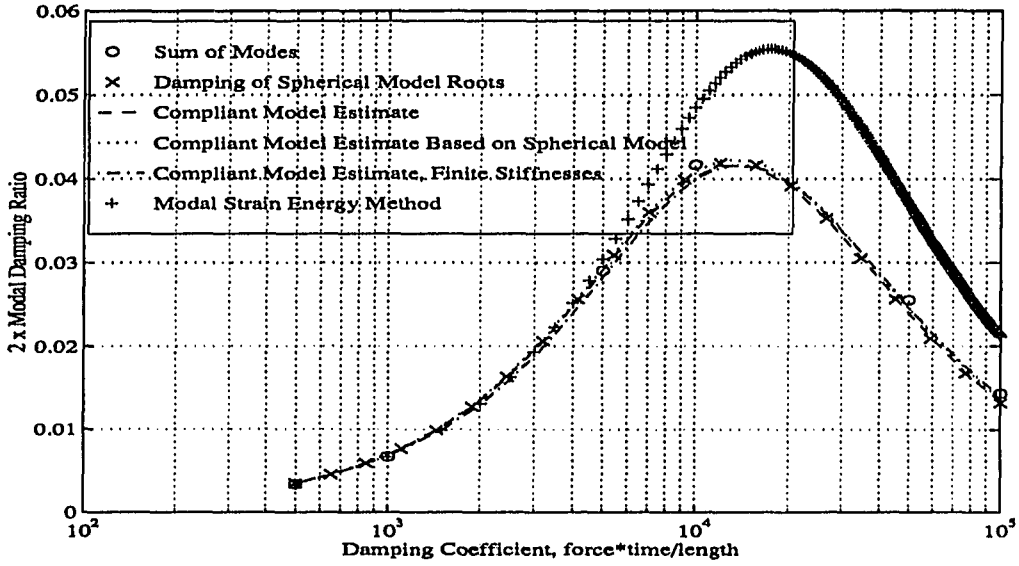

Figure 3.37: Sum of Damping and Estimations for Modes 1 through 13 of the Spice Bulkhead

element axial force, the signs of the Γ matrix are readily available.

The results of the computation are included in Figures 3.29 to 3.36 for comparison. The method of Section 3.3.5 was used to produce damping estimates of a revised compliant model based on a spherical model (also called the spherical model estimates). These estimates are very close to the values indicated by the spherical model roots (Equation 3.83). The spherical model distributes the damping estimate among modes more accurately than the simple compliant model.

Figure 3.37 shows the summation of the modal loss factors by various methods for all 13 modes evaluated. This summation could be used as a performance index. Although damping predicted by the compliant and spherical models is not allocated perfectly between modes of repeated eigenvalues, the summation correlates well with the "correct" summation obtained complex eigenvalue analysis.

The prediction due to the compliant model compares favorably to that of the modal strain energy method [15]. The modal strain energy method uses the modal strain energy of the element in the position under study to estimate the damping influence of a dashpot placed in that position. This strain energy is determined by modal displacements. Because added springs or dampers in these positions tend to stiffen the structure there, these modal displacements will not remain unchanged, and error in the damping estimation results. Reducing the stiffness of the original element to account for the stiffness of the added damper at frequencies near the target mode can mitigate these inaccuracies, but such a reduction has not been used in this example.

CHAPTER 4. OPTIMIZATION

4.1 Overview

In this chapter, passive damping optimization is performed on the bulkhead of the SPICE platform using both the spherical compliant model estimate and direct eignenvalue analysis. The procedure in both cases is to search for the optimal combination of strut locations by optimizing damping coefficient within each combination tested. The goal is not only to find the optimal passive damping in the structure, but also to compare the damping potential of diagonal strut locations to that of in-line locations. In this work, the focus will be on a system in which all struts possess identical damping coefficients.

4.2 Optimization of the Compliant Model

4.2.1 Advantages

In this section, the method for optimizing the damping of the compliant model is developed. Using the compliant model of a structure as the basis of optimization saves much computation time, since the expression for loss factor is simple to evaluate. It can provide derivative information to aid the search for the optimal damping coefficient. The model also suggests an index for ranking each strut according to its

predicted contribution to overall modal damping so that the combinatorial search may proceed efficiently.

4.2.2 Optimization within Each Strut Combination

4.2.2.1 Performance Indices In optimizing the compliant model of the SPICE bulkhead structure for damping, three separate performance indices are defined. The first is a weighted summation of modal damping loss factors for all modes of interest. It is simple to use and leads to a method of ranking damping locations according to estimated effectiveness.

The second, minimum specified modal damping ratio, is a summation of the portion of modal loss factor below the minimum prescribed value for each mode; it is a more useful objective for passive damping to be used in conjunction with active control, since it has been observed [4] that moderate passive damping levels in all modes is very effective in enhancing stability of the active control system. It is used in this dissertation for the passive damping optimization of the structure.

The third, the hyperbolic tangent index, is a smoothed approximation of the second index. It could be used for optimization by a Newton method since it possesses continuous second partial derivatives.

4.2.2.2 Weighted Summation Performance Index The damping performance index defined as the summation of all modal loss factors of interest is

$$\hat{J} \equiv \sum_{j=1}^{m} w_j \eta_j \tag{4.1}$$

where w_j is the weighting value for each mode, and m is the number of modes being studied. Combined with Equation 3.8, this becomes

$$\hat{J} = \sum_{j=1}^{m} w_j \frac{\sum_{i=1}^{n} \kappa_{sij} \frac{r_{ij}}{1+r_{ij}^2}}{1 + \sum_{i=1}^{n} \kappa_{sij} \frac{r_{ij}^2}{1+r_{ij}^2}} , \tag{4.2}$$

where κ_{sij} and r_{ij} may be derived from either the basic compliant models or the spherical model of the system.

The summation performance index is interesting because examples show that the compliant model alone gives accurate estimates of this summation even though estimated damping may not be distributed properly among the modes (see Section 3.6.3).

Although the summation performance index is not the criterion to be used in optimizing the bulkhead, it is useful in aiding that optimization. It provides a way to rank struts according to damping effectiveness, so that a combinatorial search may proceed more efficiently. It also provides a means of placing bounds on the number of struts needed to meet the requirements on the modal damping ratios. These uses are developed in Section 4.2.3.

4.2.2.2.1 Derivatives of the Weighed Summation Index The derivatives of the performance index with respect to strut damping coefficients c_i can be valuable in optimizing the damping for a given combination of damping strut locations. In Section 3.4, an expression for the derivatives of η_j with respect to c_i is derived (Equation 3.92).

The derivative of the summation index with respect to c_i is

$$\frac{\partial \hat{J}}{\partial c_i} = \sum_{j=1}^{m} \frac{\partial \hat{J}}{\partial \eta_j} \frac{\partial \eta_j}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \tag{4.3}$$

where

$$\frac{\partial \hat{J}}{\partial \eta_j} = w_j \tag{4.4}$$

$$\frac{\partial \eta_j}{\partial r_{ij}} = \kappa_{sij} \frac{(1 - {r_{ij}}^2) D_j - 2 r_{ij} N_j}{(1 + {r_{ij}}^2)^2 D_j^2} \tag{4.5}$$

$$\frac{\partial r_{ij}}{\partial c_i} = \frac{\alpha_{ij} \omega_j}{k_{sij}} \tag{4.6}$$

in which N_j and D_j are the numerator and denominator of Equation 3.84 for η_j.

The second derivatives may also be useful. They are

$$\frac{\partial^2 \hat{J}}{\partial c_k \partial c_i} = \frac{\partial}{\partial c_k} \sum_{j=1}^{m} \frac{\partial \tilde{J}}{\partial \eta_j} \frac{\partial \eta_j}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \tag{4.7}$$

$$= \sum_{j=1}^{m} \frac{\partial \hat{J}}{\partial \eta_j} \left(\frac{\partial}{\partial c_k} \frac{\partial \eta_j}{\partial r_{ij}} \right) \frac{\partial r_{ij}}{\partial c_i} \tag{4.8}$$

$$= \sum_{j=1}^{m} w_j \left(\frac{\partial}{\partial r_{kj}} \frac{\partial \eta_j}{\partial r_{ij}} \right) \frac{\partial r_{kj}}{\partial c_k} \frac{\partial r_{ij}}{\partial c_i} \tag{4.9}$$

$$= \sum_{j=1}^{m} w_j \frac{\partial^2 \eta_j}{\partial r_{kj} \partial r_{ij}} \frac{\partial r_{kj}}{\partial c_k} \frac{\partial r_{ij}}{\partial c_i} \tag{4.10}$$

where

$$i = 1,2,3,\ldots n \tag{4.11}$$

$$k = 1,2,3,\ldots n \tag{4.12}$$

and

$$\frac{\partial^2 \eta_j}{\partial r_{kj} \partial r_{ij}} = \frac{(-N_{r_{ij}} D_{r_{kj}} - N_{r_{kj}} D_{r_{ij}}) D_j + N_{r_{ij} r_{kj}} D_j^2 - N_j D_{r_{ij} r_{kj}} D_j + 2 N_j D_{r_{kj}} D_{r_{ij}}}{D_j^3} \tag{4.13}$$

in which

$$N_{r_{ij}} = \frac{\partial N_j}{\partial r_{ij}} = \kappa_{sij} \frac{1 - {r_{ij}}^2}{(1 + {r_{ij}}^2)^2} \tag{4.14}$$

$$D_{r_{ij}} = \frac{\partial D_j}{\partial r_{ij}} = \kappa_{sij} \frac{2r_{ij}}{(1 + {r_{ij}}^2)^2} \tag{4.15}$$

$$N_{r_{kj}r_{ij}} = \frac{\partial^2 N_j}{\partial r_{kj} \partial r_{ij}} \tag{4.16}$$

$$= \begin{cases} 0, & i \neq k \\ \kappa_{sij} \frac{2r_{ij}(r_{ij}^2 - 3)}{(1+r_{ij}^2)^3}, & i = k \end{cases} \tag{4.17}$$

$$D_{r_{kj}r_{ij}} = \frac{\partial^2 D_j}{\partial r_{kj} \partial r_{ij}} \tag{4.18}$$

$$= \begin{cases} 0, & i \neq k \\ \kappa_{sij} \frac{2(1-3r_{ij}^2)}{(1+r_{ij}^2)^3}, & i = k \, . \end{cases} \tag{4.19}$$

$$\tag{4.20}$$

In the case in which all n struts have identical damping coefficients,

$$c_i = c \tag{4.21}$$

$$d\hat{J} = \sum_{i=1}^{n} \frac{\partial \hat{J}}{\partial c_i} d\, c_i \tag{4.22}$$

$$\hat{J}_c \equiv \frac{\partial \hat{J}}{\partial c} = \sum_{i=1}^{n} \frac{\partial \hat{J}}{\partial c_i} \, . \tag{4.23}$$

The second derivative in this case is

$$d^2 \hat{J} = \sum_{k=1}^{n} \frac{\partial}{\partial c_k} \left(\sum_{i=1}^{n} \frac{\partial \hat{J}}{\partial c_i} d\, c_i \right) d\, c_k \tag{4.24}$$

$$\hat{J}_{cc} \equiv \frac{\partial^2 \hat{J}}{\partial c^2} = \sum_{k=1}^{n} \sum_{i=1}^{n} \frac{\partial^2 \hat{J}}{\partial c_k \partial c_i} \, . \tag{4.25}$$

In order to efficiently find the optimum value of c in the case of identical c_i, a Newton optimization procedure can be used. The shape of the summation curve is typically similar to that of a damping curve for one strut (see Figure 3.28). This means that the curvature (with respect to c) is negative in a region including $c = 0$ and the maximum. At a point somewhere beyond the maximum, the curvature

becomes positive. Within the region of negative curvature, the Newton method will converge to the maximum point. The procedure follows.

1. Make initial guess $c = c_o$.

2. Evaluate the $\hat{J}$, $\hat{J}_c$, and $\hat{J}_{cc}$ at $c = c_k$ (k being the index of iteration).

3. If $\hat{J}_{cc} \geq 0$, $c_{k+1} = c_k/2$ and return to step 2.

4. If $\hat{J}_{cc} < 0$,

$$c_{k+1} = c_k - \frac{\hat{J}_c}{\hat{J}_{cc}} . \tag{4.26}$$

5. If $\hat{J}_c < \varepsilon_1$ or $\hat{J}_{k+1} - \hat{J}_k < \varepsilon_2$, stop; otherwise return to step 2.

In the case where c_i are not necessarily identical, the partial derivatives of $\hat{J}$ with respect to c_i may be used to specify a direction of search for a new candidate design for the optimal values of c_i in n-dimensional space. Since the function is well-behaved and the first and second derivatives are known, an n-dimensional Newton optimization search could be used very efficiently.

4.2.2.2.2 Optimal Damping for a Single Mode It is interesting to study the optimal damping in a single mode in the case where damping coefficients c_i are not necessarily identical. With only one mode under consideration, the performance criteria is the modal loss factor for the mode, and the optimal damping occurs at the stationary point of modal loss factor with respect to each damping coefficient c_i (or r_{ij}).

The partial derivatives of the loss factor of mode j with respect to dimensionless

damping coefficients r_{ij} were stated in Equation 4.5 as

$$\frac{\partial \eta_j}{\partial r_{ij}} = \kappa_{sij} \frac{(1-{r_{ij}}^2)D_j - 2r_{ij}N_j}{(1+{r_{ij}}^2)^2 {D_j}^2}, \quad i = 1, 2, 3, \ldots n \tag{4.27}$$

where N_j and D_j, respectively, denote the numerator and denominator of the expression for η_j. At the stationary point, Equations 4.27 equal zeroes. Then

$$D_j \frac{1-{r_{ij}}^2}{(1+{r_{ij}}^2)^2} - N_j \frac{2r_{ij}}{(1+r_{ij}^2)^2} = 0, \quad i = 1, 2, 3, \ldots n \tag{4.28}$$

leads to

$${r_{ij}^*}^2 + 2\eta_j^* r_{ij}^* - 1 = 0, \quad i = 1, 2, 3, \ldots n\ . \tag{4.29}$$

where η_j^* is the optimal value of η_j at r_{ij}^*. Since Equation 4.29 applies for all struts i, all r_{ij}^* at the stationary point must be identical.

Now, assuming all r_{ij}^* to be identical, the values of ${r_{ij}^*}^2$ and η_j^* may be found in terms of κ_{sij}. Since at the optimum all values of r_{ij} are identical, the ratios containing it may be pulled out of the summations in Equation 3.8 resulting in

$$\eta_j^* = \frac{\frac{r_{ij}^*}{1+{r_{ij}^*}^2}\kappa_{sj}}{1 + \frac{{r_{ij}^*}^2}{1+{r_{ij}^*}^2}\kappa_{sj}} = \frac{\kappa_{sj} r_{ij}^*}{1 + {r_{ij}^*}^2(1+\kappa_{sj})} \tag{4.30}$$

where

$$\kappa_{sj} = \sum_{i=1}^{n} \kappa_{sij} \tag{4.31}$$

for mode j.

Now the stationary value of η_j is found by setting

$$\frac{\partial \eta_j}{\partial r_{ij}} = \frac{\kappa_{sij}[1 + {r_{ij}}^2(1+\kappa_{sij})] - \kappa_{sij} r_{ij}[2r_{ij}(1+\kappa_{sij})]}{[1 + {r_{ij}}^2(1+\kappa_{sij})]^2} = 0\ . \tag{4.32}$$

The solution for r_{ij}^* is

$$r_{ij}^* = \frac{1}{\sqrt{1+\kappa_{sj}}}\ . \tag{4.33}$$

Substituting this value of r_{ij}^* into Equation 4.30 results in

$$\eta_j^* = \frac{\kappa_{sj}}{2\sqrt{1+\kappa_{sj}}} . \tag{4.34}$$

The results above are similar to the case of a single strut, only the summation κ_{sj} for all of the struts replaces κ_s for the single strut in the equations for η^* and r^* (see Equations 2.5 and 2.6).

In order to achieve optimal damping in mode j, the values of c_i are set to

$$c_i = \frac{r_{ij}^* k_{sij}}{\alpha_{ij}\omega_j} . \tag{4.35}$$

If the performance index includes a summation of the loss factors of more than one mode, the optimization problem is not so simple.

4.2.2.3 Minimum Specified Modal Damping Ratio

4.2.2.3.1 Motivation Passive damping enhances stability and robustness of active structural vibration control systems [32]. It has been suggested [4] that moderate passive damping (say 2%) over a wide range of modes is very effective but that higher levels of passive damping provide little improvement. Therefore, the optimization criterion used in this work consists of the summation of the portion of modal damping ratio up to 2% in each mode of the frequency range being studied. Under this approach, individual modes could optionally be targeted for higher or lower threshold levels.

4.2.2.3.2 Statement of Criterion This optimization criterion may be stated as

$$J = \sum_{j=1}^{m} w_j \tau_j \text{sat}\left(\frac{\eta_j}{\tau_j}\right) \tag{4.36}$$

in which w_j is a set of weighting values (which may all be set to unity), τ_j is a set of threshold damping ratios (in this case, 2%) for each mode j, and the saturation function is defined by

$$\text{sat}(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \leq x \leq 1 \\ 1, & x > 1 \end{cases} \tag{4.37}$$

Of course, in the case of Equation 4.36, the argument of the saturation function is not negative.

The purpose of the saturation function in the above criteria is to reward modal damping only up to the threshold value τ_j in each mode. Beyond the threshold value, increased modal damping has no further effect on the performance index. Partial derivatives with respect to each of the design variables c_i may still be used to aid in the optimization. However, this performance criterion is not as smooth as the simple summation criterion.

The derivative of the saturation optimization criterion with respect to positive damping coefficient c_i is

$$\frac{\partial J}{\partial c_i} = \sum_{j=1}^{m} \frac{\partial J}{\partial \eta_j} \frac{\partial \eta_j}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \tag{4.38}$$

where $\frac{\partial \eta_j}{\partial r_{ij}}$ and $\frac{\partial r_{ij}}{\partial c_i}$ are given in Equations 4.5 and 4.6 respectively, and

$$\frac{\partial J}{\partial \eta_j} = w_j \left[u_s(\frac{\eta_j}{\tau_j} + 1) - u_s(\frac{\eta_j}{\tau_j} - 1) \right] \tag{4.39}$$

in which $u_s(x)$ denotes a unit step function

$$u_s(x) \equiv \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \, . \end{cases} \tag{4.40}$$

Since the second derivative of the saturation function is undefined when its argument is equal to unity, the second derivative of the saturation performance index is not defined when any value of $\eta_j = \tau_j$. Otherwise, the second partial derivatives are

$$\frac{\partial^2 J}{\partial c_k \partial c_i} = \frac{\partial}{\partial c_k} \sum_{j=1}^{m} \frac{\partial J}{\partial \eta_j} \frac{\partial \eta_j}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \tag{4.41}$$

$$= \sum_{j=1}^{m} \frac{\partial J}{\partial \eta_j} \frac{\partial^2 \eta_j}{\partial r_{kj} \partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \frac{\partial r_{kj}}{\partial c_k} \tag{4.42}$$

$$i = 1, 2, 3, \ldots n \tag{4.43}$$

$$k = 1, 2, 3, \ldots n \tag{4.44}$$

where $\frac{\partial^2 \eta_j}{\partial r_{kj} \partial r_{ij}}$ is given by Equation 4.13.

Since second derivative information is not complete, a Newton method would probably not be the best choice in an optimization of this criterion. In a problem possessing only one variable (identical values of c_i), a golden section search works well. In a multi-variable optimization, a conjugate gradient technique might be effective.

The next section describes a performance criterion which achieves nearly the same objective while maintaining differentiability.

4.2.2.4 Hyperbolic Tangent Index The hyperbolic tangent function is a smooth curve which approximates the saturation function. Because it is differentiable, using it instead of the pure saturation function in the performance criterion preserves the ability to use second derivative information for the compliant model.

The hyperbolic tangent performance criterion is

$$J_t = \sum_{j=1}^{m} w_j \tau_j \tanh\left(\frac{\eta_j}{\tau_j}\right) . \tag{4.45}$$

The first derivatives are

$$\frac{\partial J_t}{\partial c_i} = \sum_{j=1}^{m} \frac{\partial J_t}{\partial \eta_j} \frac{\partial \eta_j}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \tag{4.46}$$

$$i = 1, 2, 3, \ldots n \,, \tag{4.47}$$

where

$$\frac{\partial J_t}{\partial \eta_j} = w_j \mathrm{sech}^2 \left(\frac{\eta_j}{\tau_j} \right) \,, \tag{4.48}$$

and $\frac{\partial \eta_j}{\partial r_{ij}}$ and $\frac{\partial r_{ij}}{\partial c_i}$ are given by Equations 4.5 and 4.6.

The second derivatives are

$$\frac{\partial^2 J_t}{\partial c_k \partial c_i} = \frac{\partial}{\partial c_k} \sum_{j=1}^{m} \frac{\partial J_t}{\partial \eta_j} \frac{\partial \eta_j}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial c_i} \tag{4.49}$$

$$= \sum_{j=1}^{m} \left\{ \frac{\partial^2 J_t}{\partial \eta_j^2} \frac{\partial \eta_j}{\partial r_{kj}} \frac{\partial \eta_j}{\partial r_{ij}} + \frac{\partial J_t}{\partial \eta_j} \frac{\partial^2 \eta_j}{\partial r_{kj} \partial r_{ij}} \right\} \frac{\partial r_{kj}}{\partial c_k} \frac{\partial r_{ij}}{\partial c_i} \tag{4.50}$$

$$i = 1, 2, 3, \ldots n \tag{4.51}$$

$$k = 1, 2, 3, \ldots n \,. \tag{4.52}$$

where

$$\frac{\partial^2 J_t}{\partial \eta_j^2} = -2 \frac{w_j}{\tau_j} \tanh \left(\frac{\eta_j}{\tau_j} \right) \mathrm{sech}^2 \left(\frac{\eta_j}{\tau_j} \right) \,, \tag{4.53}$$

$\frac{\partial \eta_j}{\partial r_{ij}}$ is given by Equation 4.5, $\frac{\partial r_{ij}}{\partial c_i}$ is given by Equation 4.6, and $\frac{\partial^2 \eta_j}{\partial r_{kj} \partial r_{ij}}$ is given by Equation 4.13.

Since second derivatives of this performance index are fully known, a Newton method optimization could work well, whether or not damping coefficients are identical. If c_i are identical, then a Newton method procedure such as that given in Section 4.2.2.2 can be used, with $\hat{J}$ replaced by J_t, $\hat{J}_c$ replaced by

$$\frac{\partial J_t}{\partial c} = \sum_{i=1}^{n} \frac{\partial J_t}{\partial c_i} \,, \tag{4.54}$$

and $\hat{J}_{cc}$ replaced by

$$\frac{\partial^2 J_t}{\partial c^2} = \sum_{k=1}^{n} \sum_{i=1}^{n} \frac{\partial^2 J_t}{\partial c_k \partial c_i} \ . \tag{4.55}$$

Since this optimization index is slightly different than the saturation criterion, the saturation index might be re-evaluated at the optimum of the hyperbolic tangent index. This optimum might also be the starting point for a short search to locate the true optimum of the saturation damping performance index.

4.2.3 Optimization Among Combinations

4.2.3.1 Ranking Prospective Damping Locations The prospective damping locations are ranked according to each of three criteria so that the search for an optimal combination of damping locations may proceed efficiently.

4.2.3.1.1 Summation Ranking Index In order to derive a ranking index based on the summation performance index of Section 4.2.2.2, an estimate of relative contribution of each strut i to $\hat{J}$ in Equation 4.2 must be isolated. This is difficult to do because of the summation terms in the denominator. However, since $\hat{J}$ is relatively insensitive to the denominator term of a given strut, these terms may be neglected. Then the summations over i and j may be exchanged, so that a new estimate of $\hat{J}$ is

$$\tilde{J} = \sum_{i=1}^{n} \sum_{j=1}^{m} w_j \kappa_{sij} \frac{r_{ij}}{1 + {r_{ij}}^2} \ . \tag{4.56}$$

The contribution from strut i is

$$\tilde{J}_i = \sum_{j=1}^{m} w_j \kappa_{sij} \frac{r_{ij}}{1 + {r_{ij}}^2} \ . \tag{4.57}$$

Even though $\tilde{J}_i$ is not the exact contribution from strut i to the summation performance index, it reflects the relative importance of each strut, providing a means

of selecting struts in a combinatorial search. If a quick and sub-optimal design is acceptable, $\tilde{J}_i$ could be used as a criterion for selecting strut locations for passive damping, so that the combinatorial problem is avoided altogether.

The summation performance index has limitations. A given strut may rank highly because of very high damping in only two or three modes, but it may not be so valuable in achieving moderate modal damping in a large number of modes. Because of the simplified denominator term in Equation 4.2, the ranking index is an overly optimistic estimate of the damping contribution of a given strut location.

4.2.3.1.2 Distribution Ranking Index Struts that distribute modal damping evenly among the modes should be favored. This is reflected in the distribution strut ranking criterion given below:

$$J_{Hi} = \frac{\sum_{j=1}^{m} w_j}{\sum_{j=1}^{m} \frac{w_j}{1+\frac{\eta_{ij}}{r_j}}} \tag{4.58}$$

where

$$\eta_{ij} = \frac{\kappa_{sij} \frac{r_{ij}}{1+r_{ij}^2}}{1 + \kappa_{sij} \frac{r_{ij}^2}{1+r_{ij}^2}} . \tag{4.59}$$

The distribution strut ranking criterion is related to the harmonic mean.

4.2.3.1.3 Kappa Ranking Index Optimal damping of a lone strut for a given mode is

$$\eta_{ij}^* = \frac{\kappa_{sij}}{2\sqrt{1+\kappa_{sij}}} . \tag{4.60}$$

For small κ_{sij},

$$\eta_{ij}^* \approx \frac{\kappa_{sij}}{2} . \tag{4.61}$$

The kappa strut ranking criterion is a combination of the estimated η_{ij}^* given by

$$J_{\kappa i} = \sum_{j=1}^{m} \frac{\kappa_{sij} w_j}{2} \ . \tag{4.62}$$

Like the summation ranking index, this ranking criterion is not an underestimate and is usually an overestimate of a strut's contribution to damping, since all modal damping peaks do not usually occur at the same value of c_i. However, it is simpler to compute than the summation ranking index since no auxiliary optimization is required.

4.2.3.2 Bounding the Number of Struts Needed The desired performance criterion is to be achieved with the lowest possible number of damping struts. The first step in the combinatorial problem (that of finding the best combination of strut locations) is to find bounds on the number of struts required. The summation performance index may be used to set such bounds.

The lower bound on the number of struts required can be found by summing the strut ranking indices of the most highly-ranked damping locations until the summation exceeds the required saturation index for all modes. An upper bound may be found by optimizing the saturation index for the most highly-ranked struts. Struts are added until the summation exceeds the required performance index. These two procedures are detailed below.

4.2.3.2.1 Lower Bound on Number of Damping Struts To find a lower bound on the number of struts needed to achieve required damping, the summation and kappa ranking indices are used. Beginning with the summation ranking index, the strut ranking criteria $\tilde{J}_i$ of the most highly-ranked damping locations are

summed until the summation exceeds the overall saturation index J. This may be stated as follows.

Find n_ℓ such that if

$$J_{max} \equiv \sum_{j=1}^{m} w_j \tau_j \text{sat}\left(\frac{\eta_j}{\tau_j}\right)\Big|_{\eta_j \geq \tau_j} = \sum_{j=1}^{m} w_j \tag{4.63}$$

then

$$\sum_{i=1}^{n_\ell - 1} \tilde{J}_i < J_{max} \leq \sum_{i=1}^{n_\ell} \tilde{J}_i \tag{4.64}$$

To seek an improved lower bound, this process is then repeated using the kappa index.

Both the summation and kappa ranking indices generally provide an overestimate of the contribution of a strut to damping. Furthermore, these strut ranking indices contain no information concerning distribution of damping among the modes. Since the damping will almost inevitably not be distributed among the modes as the saturation or kappa index specifies, some damping in the n_ℓ damping struts will not be included in an optimization of the saturation index.

This means that the definition of n_ℓ stated above represents the very least number of damping struts necessary to achieve the optimal value of the saturation performance criterion J_{max}. Thus n_ℓ is a lower bound on the number of damping struts required.

4.2.3.2.2 Upper Limit on the Number of Damping Struts In order to find an upper bound n_u on the number of struts needed to optimize the saturation index, the following procedure may be used.

1. Start with the n_ℓ most highly-ranked struts according to the summation criterion.

2. Include the next most highly-ranked strut.

3. Optimize the combination according to the saturation index.

4. If $J = J_{max}$ stop. Then the number of struts is n_u.

5. Return to step 2.

The combination for which step 4 is satisfied is an acceptable design according to the saturation index; however, there may be other acceptable designs which include fewer damping struts. Therefore n_u is an upper bound on the number of struts required. The design is a point of comparison for future designs.

The above procedure is repeated using distribution and kappa ranking criteria in an attempt to reduce the upper bound and make the subsequent optimization more efficient.

4.2.3.3 Method of Combinatorial Search

4.2.3.3.1 Simulated Annealing The combinatorial problem is to find the most effective combination of n_{opt} damping strut locations. This is a difficult search problem since the contributions of each strut to overall modal damping is dependent upon which other struts are present. Furthermore, the many local maxima among the possible damping strut combinations obscure the global maximum.

The simulated annealing [16] optimization technique is well-suited for such a problem because randomness in its search and acceptance criteria helps prevent it from getting stuck in local maxima. Although it converges slowly, it is more likely to find the global maximum than deterministic search techniques.

4.2.3.3.2 The Search The simulated annealing technique is outlined in Section 2.6.1. Each iteration of the search involves the random selection of a new design and evaluation of the resulting performance criterion. All improving designs are accepted, and the probability of accepting a non-improving design is determined by the value of its performance criterion along with the present "temperature" of the system.

As the search proceeds, this temperature cools at a controlled rate, so that fewer and fewer non-improving designs are accepted, and the process ends with a local optimization about what is hoped to be the global maximum.

Since Chen, et al. [6], had success in using this method to determine damper and actuator placement in truss structures, this dissertation follows their procedure in general. However, some augmentations are made to take advantage of information included in the compliant model of the structural modes.

In the Chen paper, the combination of damping element locations is changed one at a time, at random. The new combination is accepted if it results in an improved function evaluation (in that case energy dissipation), or in a non-improving function evaluation which satisfies

$$P < e^{(E_{k+1}-E_k)/\theta} , \tag{4.65}$$

where P is a random number between zero and one, θ is the temperature, and E_k is the kth function evaluation. As the temperature cools, the probability of accepting a non-improving function evaluation decreases.

In the Chen paper, the temperature θ is adjusted each time the search reaches a thermal equilibrium, that is, until the same value of performance criterion is obtained a set number (say 10) times successfully. Then θ is multiplied by a factor of 0.8. At

this time the search returns to the previous design which exhibited the best function evaluation found so far. This is intended to speed convergence. Final convergence occurs when the temperature drops below a specified θ_{min} and the same function evaluation is obtained a set number of successive times (Chen suggested 30).

Figure 4.1 is a flowchart the algorithm *sachen.m* used in this dissertation. It follows the procedure of Chen except in the ways noted in the following paragraphs.

Chen's procedure begins with a random combination of locations, but since the compliant model results in a ranking of struts and upper bound on the number of struts needed, the search here is begun at the combination by which that upper bound has been determined. That is, the initial guess is a combination of the highest-ranking struts.

Chen's thermal equilibrium counter is reset each time a new design fit the acceptance criterion. To avoid very long searches, *sachen.m* only resets the counter when a new best function evaluation is found. The counter is required to reach 20 before thermal equilibrium occurs (Chen suggested 10). Furthermore, the criteria for final convergence is set to 50 successive function evaluations instead of 30.

The initial temperature is set so that a design displaying 99.5% of the initial function evaluation would have a 2% probability of being accepted.

The compliant model contains in κ_{sij}'s an approximation of the contribution of damping of each strut i to each mode j. In order to improve the probability that struts containing high damping potential in needy modes will be selected by the simulated annealing algorithm, an auxiliary weighting function is defined. Probabilities are then assigned to each strut according to a measure of their damping potential in each of the modes and how much additional damping in each mode is required to reach the

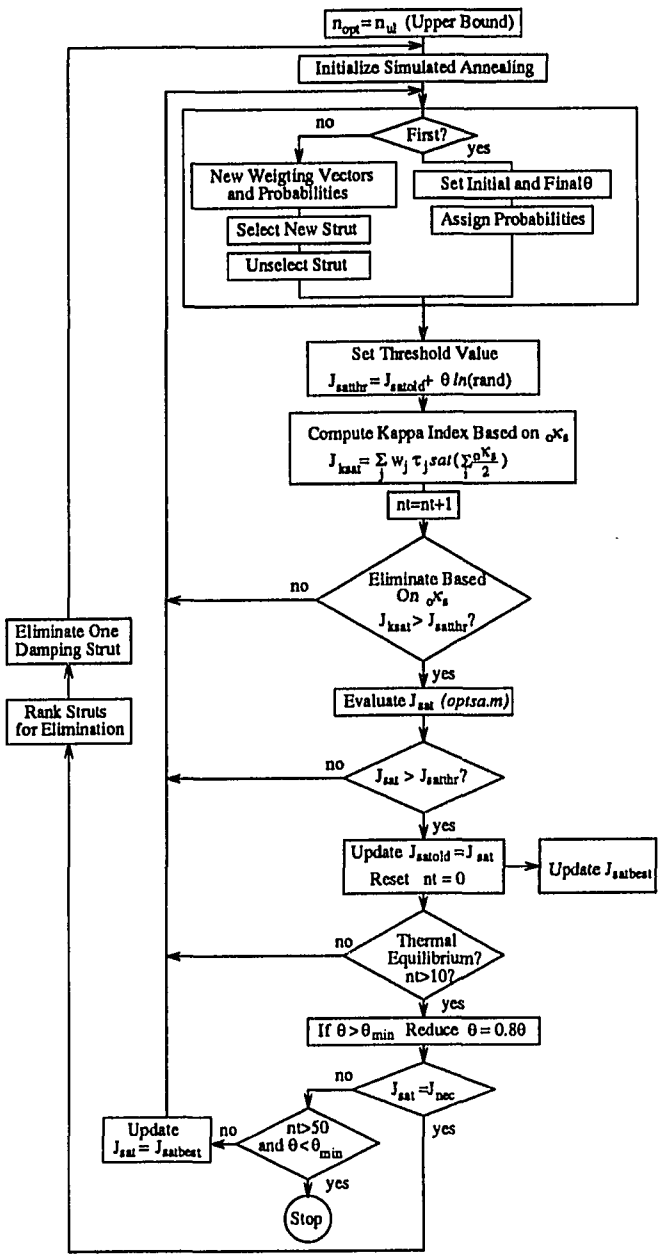

Figure 4.1: Flowchart of Simulated Annealing Algorithm

threshold value τ_j. These assignments are described below.

A given design in the search achieves damping values of η_j in each mode j, for which the threshold damping value is τ_j. Weighting functions w_κ are assigned according to

$$w_{\kappa j} \propto \max\left(\frac{\tau_j - \eta_j}{\tau_j}, 0.0001\right) \tag{4.66}$$

and

$$\sum_{j=1}^{m} w_{\kappa j} = 1 \ . \tag{4.67}$$

The result of Equation 4.66 is that modes possessing $\frac{\eta_j}{\tau_j}$ of 0.5, 1.0, and 2.0 respectively would have relative weights $w_{\kappa j}$ of 0.5, 0.0001, and 0.0001 before normalization.

Next, a new strut ranking index $J_{\kappa sel}$ for strut selection is computed. It is similar to the kappa strut ranking index, except that $w_{\kappa j}$ is used as the weighting function, so that struts with high values of κ_{sij} in modes needing more damping rank highly.

$$J_{\kappa sel\, i} = \sum_{j=1}^{m} \frac{w_{\kappa j} \kappa_{sij}}{2} \tag{4.68}$$

Next, a probability of selection for each strut not currently in the design set is computed as

$$p_{sel\, i} = \frac{J_{\kappa sel\, i}}{\sum_{k=1}^{n-n_{opt}} J_{\kappa sel\, k}} \quad i = 1, 2, \ldots, n - n_{opt} \tag{4.69}$$

over all locations i currently *excluded* from the design set. Here n_{opt} is the number of damping strut locations included in the present design set.

Similarly, a probability of replacement for each strut currently in the design set is computed as

$$p_{boot\, i} = \frac{\frac{1}{J_{\kappa sel\, i}}}{\sum_{k=1}^{n_{opt}} \frac{1}{J_{\kappa sel\, k}}} \quad i = 1, 2, \ldots n_{opt} \tag{4.70}$$

over all locations i currently *included* in the design set. After these probabilities are normalized, they are used to produce each new candidate design by randomly

replacing a strut in the design set by one currently outside the set. Struts exhibiting low values of $J_{\kappa sel}$ are seldom chosen.

Each design must satisfy the criterion of Equation 4.65 in order to be accepted. Since evaluation of J_{sat} for each combination of struts requires a golden section optimization for c, a quick evaluation technique is used to eliminate combinations which fall well short of this criterion. This technique involves a strut index based on ${}_o\kappa_{sij}$.

Since the kappa strut index contains an over-estimate of the contribution of each strut to each mode, a saturation index $J_{\kappa sat}$ based on ${}_o\kappa_s$'s will be larger than the actual value of J_{sat}. This means that if $J_{\kappa sat}$ for a given combination of struts will not meet the probabilistic acceptance criterion, then neither will J_{sat} itself. In that case, the optimization to compute J_{sat} may be bypassed. However, if $J_{\kappa sat}$ passes the criterion, then J_{sat} itself must be tested against the criterion to make a decision regarding acceptance of the new design.

$J_{\kappa sat}$ is given by the following equations. Since

$$\eta_{ij} \approx \frac{{}_o\kappa_{sij}}{2}, \tag{4.71}$$

$$J_{\kappa sat} = \sum_{j=1}^{m} w_j \tau_j \mathrm{sat}\left(\frac{1}{\tau_j}\sum_{i=1}^{n_{opt}} \frac{{}_o\kappa_{sij}}{2}\right) \tag{4.72}$$

where the summation on i includes only the struts included in the design under consideration.

The probabilistic acceptance criterion of Equation 4.65 involves computing a probability based upon the value of a non-increasing function evaluation. In order that the $J_{\kappa sat}$ and J_{sat} values may be evaluated using equivalent probabilities, a random number α_r between zero and one is chosen for each design and the criterion

is restated as

$$\text{Accept} \left\{ \begin{array}{c} J_{\kappa sat} \\ J_{sat} \end{array} \right\}_{k+1} \text{if} \left\{ \begin{array}{c} J_{\kappa sat} \\ J_{sat} \end{array} \right\}_{k+1} > J_{sat\,k} + \theta \ln(\alpha_{r\,k+1}) \; . \tag{4.73}$$

Here $J_{sat\,k}$ is the function evaluation for the last accepted design (step k). This statement of the criterion shows that a value of $J_{sat\,k+1}$ may be accepted even if it is somewhat lower than the previous value, $J_{sat\,k}$. The values of θ and α_r determine how much regression of J_{sat} may be accepted. This statement of the criterion enables both $J_{\kappa sat}$ and J_{sat} to be compared to a single acceptance threshold.

4.2.3.3.3 Reducing the Number of Struts The goal of the optimization of this dissertation is to determine the combination of damping strut locations which achieves the saturation index using the fewest number of damping struts. In order to do this, the simulated annealing algorithm must be executed a number of times, each with a set number of damping struts included in the design.

Section 4.2.3.2 outlined a method for determining upper and lower bounds on the number of damping struts needed to achieve the desired J_{sat} value. Assuming these bounds are known, a procedure for finding the minimum number of struts n^* must be developed.

Although simulated annealing algorithms are unlikely to get trapped in local optima, they have the disadvantage of converging rather slowly [16]. Therefore, multiple annealing runs which require extensive cooling should be avoided. Designs involving more than enough damping struts will likely reach the target J_{sat} before the simulated annealing algorithm is required to fully cool (there may be several combinations which will provide the requisite damping in each mode). Therefore, the number of

struts considered n_{opt} begins at the upper bound. The simulated annealing algorithm proceeds until the target J_{sat} is achieved. When this occurs, the algorithm eliminates the rightmost damping strut in the combination vector, so that the total number of damping struts is reduced by one. The simulated annealing algorithm begins again. This process continues until the simulated annealing algorithm converges (reaches a near-global minimum) before the target value of J_{sat} is achieved. At this point, there are not enough damping struts present to achieve target J_{sat}, and n^* is the number of struts plus one.

The last design which achieved the target J^*_{sat} using n^* is then an optimal design according to the saturation performance criterion. J^*_{sat} is the highest value which can be reached. It is possible that other combinations containing n^* damping struts also may reach J_{sat}, but not any higher value, since J^*_{sat} is saturated in all modes under consideration.

4.3 Computation

Using the spherical compliant model to evaluate damping, simulated annealing is used to seek the minimum number of struts necessary to achieve saturated damping in modes 1 through 16 of the SPICE bulkhead structure. The details of the procedure used are given in the following sections. Results of the computation are then presented in Section 4.3.4.

4.3.1 Obtaining Values of κ and α

In order to characterize the compliant model of the SPICE bulkhead structure, values of κ_{sij} and α_{ij} for each strut i and mode j must first be found. Values of κ_{sij}

are dependent upon the stiffness k_i of the series stiffness of the damper in a given candidate location.

The values of k_i are set to values similar to those attained in existing D-Strut designs. The D-Struts used in [33] had a ratio of effective inner to outer strut stiffness of about 1.2, and outer strut stiffness (k_A in Figure 2.4) of 81,000 lb/in. Then the equivalent inner (series) stiffness (including volumetric compliance of the fluid cavity) is about 98,000 lb/in ($17.2\times10^6 N/m$). Therefore, this value is used in this dissertation for the inner stiffness of damping struts collocated with existing struts. For these in-line locations, the existing strut stiffness is left in place if a damper is added, so that the existing strut along with the added damper and series stiffness comprise the replacement D-Strut. Although in references [32] and [33], outer strut stiffness was reduced about 50% from that of undamped struts in order to increase the ratio κ of the in-line strut, no such reduction is made in this dissertation.

In the case in which an added damper is to span diagonally across octahedral corner nodes, the physical damper must be longer by a factor of $\sqrt{2}$. Therefore, the stiffness of the inner member is reduced by a factor of $\sqrt{2}$ to 69,300 lb/in ($1.21 \times 10^6 N/m$). This reduction may be greater than necessary to account for the added length of the diagonal damping strut, since much of the equivalent series compliance of a D-Strut comes from volumetric compliance, which is independent of strut length. Although this series stiffness is lower than that of an in-line damper, the omission of parallel stiffness at this location in the structure keeps the values of κ_{sij} relatively high.

The method of frequency slopes then determines the values of κ_{sij} and α_{ij}. Unit stiffnesses are added to the original structure in each candidate location to obtain

mass-normalized axial node displacements in each normal mode, as described in Section 3.2.3. A shell script called *akk1sh* is used to call the NASTRAN finite element solution sequences to obtain the necessary frequency and displacement information. The shell script, along with the NASTRAN files are included in Appendix D, and a flowchart of this computation is shown in Figure 4.2.

Prior to running the shell script, separate scripts *make12k1sh* and *make_inputk1* use a list of candidate locations to write separate files containing bar (*cbar*) elements of stiffness k_i for each of the candidate damper locations. This script and the damper location list, *blistblk*, are also included in Appendix D.

The shell script first calls for a normal mode analysis of the original structure augmented by unit stiffnesses in candidate locations. These *conrods* are written into a NASTRAN input data file by *make12k1sh* and *make_inputk1*. Once the normal mode analysis is complete, frequencies are saved in one file, and displacements are saved in another file.

The *akk1sh* shell script then enters a loop in which it inserts one spring k_i at a time into the structure and performs a real eigenvalue analysis each time. The resulting frequencies, which provide enough information to easily determine κ_{sij}, are concatenated to the file containing modal frequencies.

Matlab files are used for the rest of the computation. The first one, called *bonk.m*, uses the compliant model parameters to determine lower and upper bounds on the number of struts needed to achieve damping saturation. The second, *sachen.m*, uses simulated annealing on the compliant model to optimize the saturation damping criteria for a decreasing number of struts until the minimum number of struts is determined. In both cases, the Matlab program reads frequencies and displacement

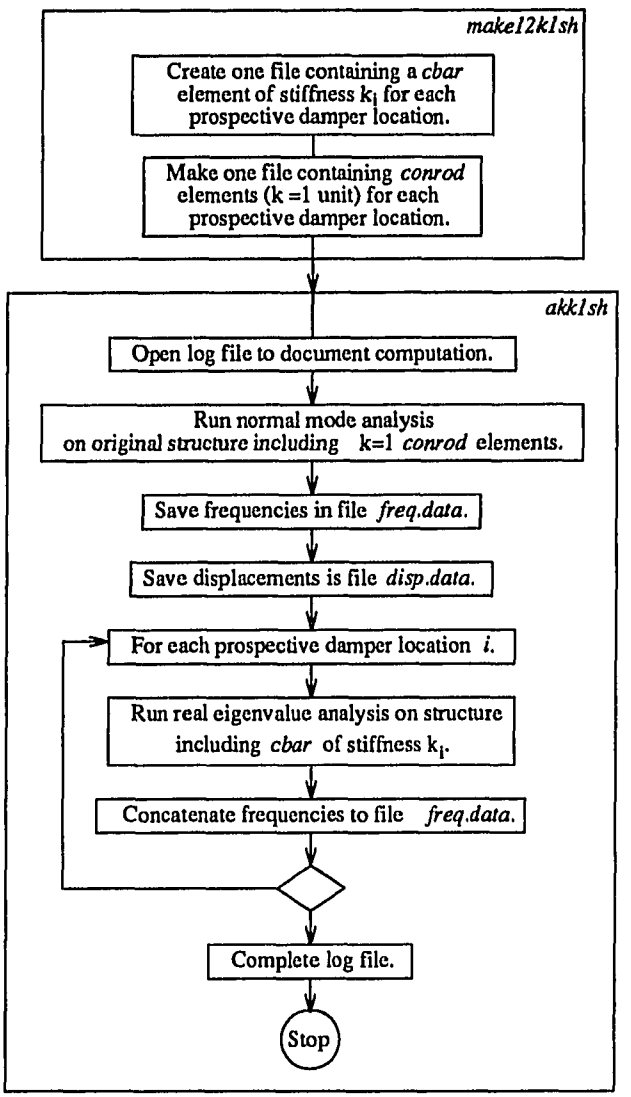

Figure 4.2: Flowchart of Computation to Obtain Frequency Shifts and Displacements for the Unmodified SPICE Bulkhead

files resulting from the NASTRAN runs called by the *akk1sh* shell script and computes the values of κ_{sij}, α_{ij}, and γ_{ij} for the bulkhead structure.

4.3.2 Program for Bounding the Number of Struts

The methods set forth in Section 4.2.3.2 are used by Matlab script file *bonk.m* to bound the number of damping struts needed to achieve saturated damping according to the compliant model. Figure 4.3 shows a flowchart of *bonk.m*, and Appendix D contains a program listing.

Program *bonk.m* first ranks the struts according to three criteria described in Section 4.2.3.2. They are the strut summation index, the distribution index, and the kappa index.

Then *bonk.m* finds the lower bound on the number of struts by adding up the strut summation indices of the kappa indices of successively larger numbers of the most highly ranked struts according to the respective indices.

Next, *bonk.m* seeks an upper bound for the number of struts by choosing the most highly ranked n_l struts and optimizing that combination as described below. A single strut is added each iteration and the combination is re-optimized each time until saturation damping is achieved. The number of struts included is then the upper bound. This is repeated for ranking according to all three strut ranking indices to achieve a reduced upper bound.

Each optimization among a strut combination is done by golden section. In order to do this, the approximate re-orientations within the spherical model for present combination are first computed by inserting springs of $\frac{k_{si}}{2}$ in each damper location. This results in a set of ${}_o\kappa_{sij}$ for the present combination.

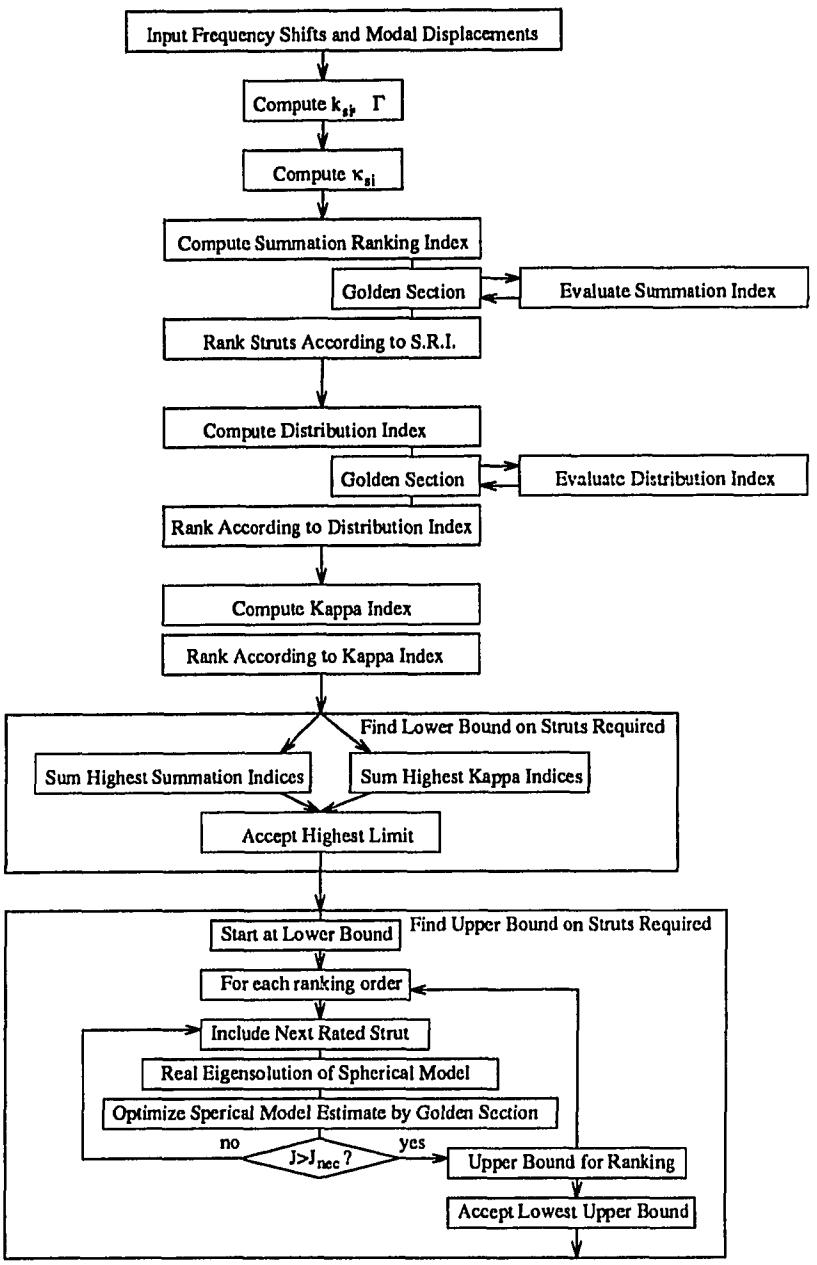

Figure 4.3: Flowchart of *bonk.m* for Bounding the Number of Struts Required

The first and second derivatives of Section 4.2.2 have been used to perform a Newton optimization of the tangent index of the structure in an attempt to close in on the optimal point of the saturation index in a reduced time. Although the number of function evaluations is reduced, the operations involved in evaluating the necessary first and second derivatives increase computation time, so the Newton optimization is bypassed. The golden section searches require an average of 0.68 CPU seconds each on a DEC 3000-300L workstation.

The above golden section procedure is used to optimize damping within a combination by *sachen.m* in solving the combinatorial optimization problem.

4.3.3 Simulated Annealing Program

The Matlab script *sachen.m* is used to find the number of struts needed to achieve saturated damping in each of the target modes. This program, outlined by the flowchart of Figure 4.1, is listed in Appendix D.

The algorithm used to "anneal" the system for a given number of struts is close to that of Chen [6]. See Sections 2.6.1 and 4.2.3.3. However, each time a new strut is accepted, the kappa criterion is used to reassign the probabilities that a given strut will be selected (or de-selected). Raising these probabilities to a set power (*propowsel* or *propowboot*) prior to normalization determines the degree to which "better" struts are more likely to be chosen. Program *sachen.m* also includes a way to use ${}_o\kappa_{sij}$ information to bypass the need for a golden section optimization if damping for a combination falls far short of a threshold value. Setting *propowsel* and *propowboot* to zero causes all struts to be given an equal probability, as was done by Chen. This setting also deactivates the re-ordering of struts based on the kappa strut ranking

criterion prior to strut elimination. However, the initial design for *sachen.m* uses the most highly ranked struts regardless of *propowsel* and *propowboot.*

The number of struts included in the simulated annealing operation begins at the upper bound, with the most highly rated struts. The algorithm searches until it finds a combination which achieves saturated damping in each target mode. Once such a combination is found, the rightmost strut in the combination vector is eliminated, and the simulated annealing routine begins on a design possessing one less damping strut location. It is assumed that if a combination able to achieve saturation exists, the algorithm will find it. Therefore, when the simulated annealing algorithm converges using n_{opt} struts without achieving saturation damping, it is concluded that $n_{opt}+1$ damping struts are necessary, and the final combination comprising $n_{opt}+1$ struts is printed.

Symmetry is not used in the solution, since a non-symmetric combination of struts could well be the most effective.

4.3.4 Results of Simulated Annealing Using the Compliant Model

The results of the computations described in the preceding section are presented in Tables 4.1 through 4.5 and Figures 4.4 through 4.6. In the following discussion, the computation time is described according to the following definitions. User time is the time spent in the system, system time is the time spent in execution of the command on the diagnostic output system, and real time is the elapsed time during the execution of a command; CPU time is the sum of user and system time.

Table 4.1 presents the results of the *bonk.m* program for bounding the necessary number of struts, and Table 4.2 ranks the top 72 strut locations according to the strut

Table 4.1: Results of the *bonk.m* Program

Criterion	Lower Bound, n_l	Upper Bound, n_u
Summation	4	46
Distribution		48
Kappa	4	53
Used	4	46

summation ranking, along with their rank according to the distribution and kappa criteria. It also lists the end nodes of each location and whether it is a diagonal or in-line strut. The most highly ranked struts are diagonal.

Figure 4.4 shows the function evaluations, temperature schedule, and number of struts over a simulated annealing run of *sachen.m*. For this run, the probability exponent *propowsel* is set to 1, while *propowboot* is left at 0. The simulated annealing action can be seen to repeat each time the damping design is reduced by one strut. The simulated annealing algorithm finally converges below the saturation damping limit when $n_{opt} = 16$. This means that $n^* = 17$ struts are needed to achieve required damping; this is the solution to the problem posed according to the compliant model.

The total CPU time consumed on a DEC 3000-300L OSF Alpha workstation by the *sachen.m* program with *propowsel* = 1 is 949 seconds. The total real time of 1032 seconds is quite short. Over the whole range of n_{opt}, 1389 function evaluations are performed by *sachen.m*. Because of the feature which bypasses golden section optimizations for certain combinations based on ${}_o\kappa_{sij}$, only 690 golden section optimizations are necessary; that is, the golden section search is bypassed 50.3% of the time. The function evaluations are quick. Using the spherical compliant estimate, each golden section search requires an average of only 0.68 CPU seconds to perform.

The combination which converged at $n^* = 17$ is presented in Table 4.3. Many

Table 4.2: Ranking of Struts Resulting from *bonk.m*

Ranking			Strut			Type
Summation	Distribution	Kappa	Number	Nodes	J	(D or I)[a]
1	10	2	277	30-43	1.98E-01	D
2	9	1	74	29-44	1.98E-01	D
3	2	3	155	58-59	1.98E-01	D
4	1	4	175	06-07	1.98E-01	D
5	3	5	54	52-63	1.98E-01	D
6	4	6	257	02-09	1.98E-01	D
7	8	7	154	63-64	1.50E-01	D
8	7	8	177	02-04	1.50E-01	D
9	16	10	279	44-56	1.50E-01	D
10	15	9	75	16-30	1.50E-01	D
11	6	11	52	45-58	1.50E-01	D
12	5	12	256	06-23	1.50E-01	D
13	11	13	57	59-67	1.39E-01	D
14	12	14	259	03-07	1.39E-01	D
15	13	15	157	51-52	1.39E-01	D
16	14	16	172	15-09	1.39E-01	D
17	18	17	72	43-57	1.39E-01	D
18	17	18	274	17-29	1.39E-01	D
19	35	19	163	40-41	3.41E-02	D
20	36	20	168	26-27	3.41E-02	D
21	19	21	61	33-48	3.41E-02	D
22	20	22	265	20-33	3.41E-02	D
23	23	23	66	27-42	3.40E-02	D
24	24	24	270	28-41	3.40E-02	D
25	29	27	181	52-53	2.81E-02	I
26	30	28	199	09-12	2.81E-02	I
27	31	29	81	53-59	2.81E-02	I
28	32	30	283	07-12	2.81E-02	I
29	33	25	99	36-43	2.81E-02	I
30	34	26	301	29-36	2.81E-02	I
31	27	31	111	53-60	2.79E-02	I
32	28	32	212	08-12	2.79E-02	I
33	21	33	110	46-53	2.79E-02	I
34	22	34	213	12-18	2.79E-02	I
35	25	35	44	36-50	2.78E-02	I
36	26	36	43	22-36	2.78E-02	I
37	37	37	158	53-54	2.53E-02	D
38	38	38	173	12-13	2.53E-02	D
39	41	39	71	21-36	2.53E-02	D
40	42	40	275	36-49	2.53E-02	D
41	39	41	56	39-53	2.53E-02	D
42	40	42	260	12-25	2.53E-02	D
43	47	49	84	60-64	2.41E-02	I
44	48	50	286	04-08	2.41E-02	I
45	43	51	184	45-46	2.41E-02	I
46	44	52	196	23-18	2.41E-02	I
47	53	53	298	16-22	2.41E-02	I
48	54	54	96	50-56	2.41E-02	I
49	51	43	101	22-29	2.41E-02	I
50	52	44	304	43-50	2.41E-02	I
51	49	45	78	46-52	2.40E-02	I
52	50	46	281	09-18	2.40E-02	I
53	45	47	202	07-08	2.40E-02	I
54	46	48	179	59-60	2.40E-02	I
55	74	55	63	19-34	2.26E-02	D
56	73	56	268	34-47	2.26E-02	D
57	106	57	166	34-35	2.25E-02	D
58	57	58	246	54-59	2.12E-02	I
59	58	59	142	07-13	2.12E-02	I
60	55	60	10	09-25	2.12E-02	I
61	56	61	12	39-52	2.12E-02	I
62	59	62	144	21-29	2.12E-02	I
63	60	63	244	43-49	2.12E-02	I
64	69	70	126	49-56	2.07E-02	I
65	70	71	224	16-21	2.07E-02	I
66	65	72	28	04-13	2.07E-02	I
67	66	73	32	54-64	2.07E-02	I
68	71	74	122	23-25	2.07E-02	I
69	72	75	228	39-45	2.07E-02	I
70	61	64	115	47-54	2.05E-02	I
71	62	65	216	13-19	2.05E-02	I
72	63	66	114	39-47	2.05E-02	I

[a] D denotes diagonal, I denotes inline

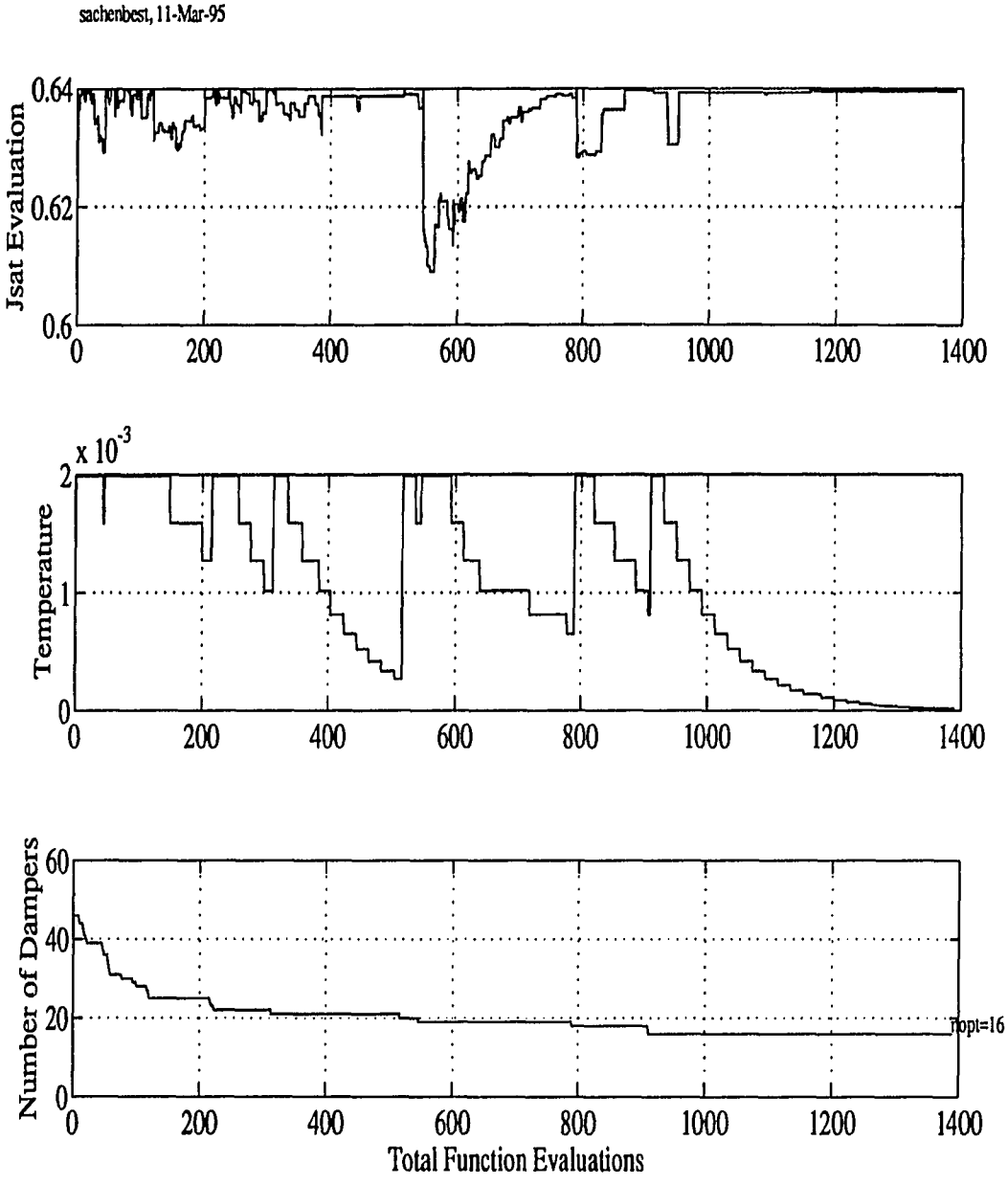

Figure 4.4: Results of Simulated Annealing with *propowsel* = 1

Table 4.3: Optimal Design According to *sachen.m* with *propowsel* = 1

Strut Number	Node Pair	Type (D=diagonal)	Strut Number	Node Pair	Type (D=diagonal)
155	58–59	D	157	51–52	D
257	02–09	D	89	34–41	I
54	52–63	D	291	52–58	I
261	32–45	D	279	44–56	D
163	40–41	D	57	59–67	D
277	30–43	D	259	03–07	D
265	20–33	D	70	04–14	D
61	33–48	D	270	28–41	D
274	17–29	D	0	–	

of these struts are of the diagonal type. This combination is not unique.

Figure 4.5 shows a bar graph comparing the actual damping estimations (2ζ) for each of the modes of this design computed by the spherical model compared to that computed by direct complex eigenvalue analysis. Even with so many struts present, the spherical model loss factor estimate provides a good approximation of the actual modal damping values in several modes. The estimate is less accurate in modes 3, 4, 12, 14, 15, and 16. Each mode except 4 and 12 did have at least 2% damping, but some modes had much more than needed.

Damping estimates from the roots of the spherical model provide more accurate results. They could provide a better basis for optimization.

To reveal whether the adjusted probabilities improved convergence of the simulated annealing algorithm, the program is run again, this time with *propowsel* = *propowboot* = 0, so that all locations have an equal probability for being chosen to be replaced into the damping design set. The results are given in Figure 4.6.

With *propowsel* = 0, the total CPU time consumed is 1087 seconds, somewhat

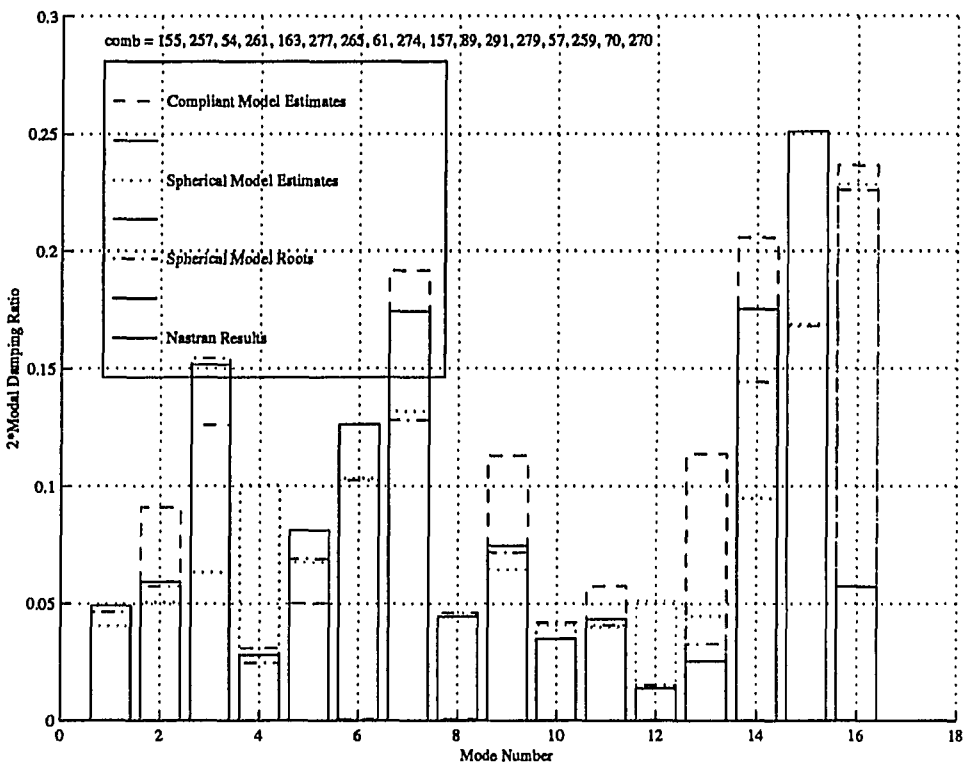

Figure 4.5: Correlation of Damping Estimation of Optimal Design by Compliant Model

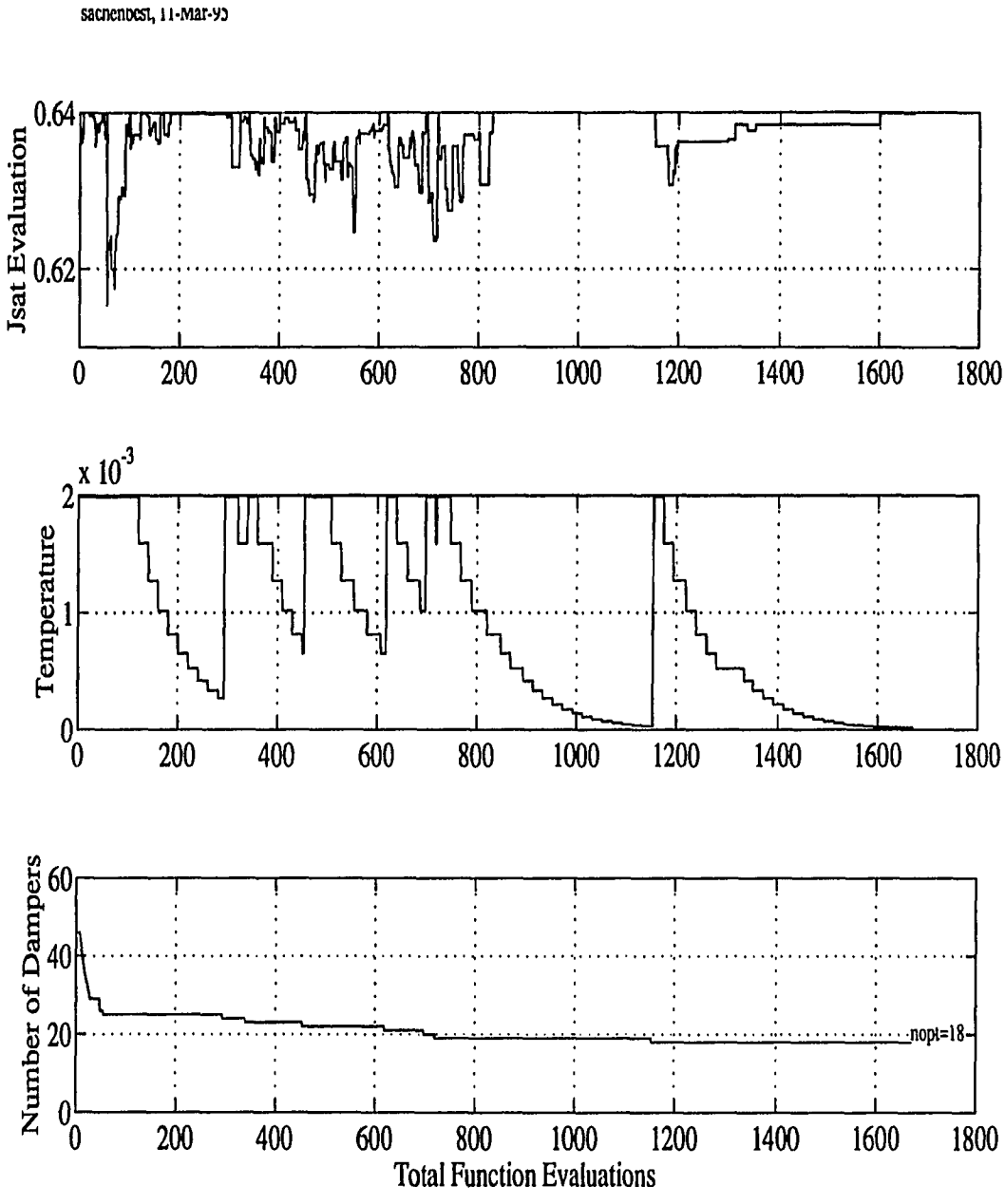

Figure 4.6: Results of Simulated Annealing with *propowsel* = 0

longer than with *propowsel* = 1, but still reasonable. A total of 1671 function evaluations are performed.

Since simulated annealing is probabilistic in nature, results of *sachen.m* varied. The runs shown in this dissertation are fairly typical. Using *propowsel* = 0, five runs averaged 1478 function evaluations resulting in a mean of 19 damping struts in the solution set. With *propowsel* = 1, six runs averaged 1235 function evaluations and 17.8 dampers. Apparently, the strut selection probability weighting scheme enacted by setting *propowsel* = 1 is successful in reducing computation time as well as the number of function evaluations needed.

Various other non-zero values of *propowsel* have been tried, but the values presented here seem to give the best results. Non-zero *propowboot* seem to reduce the randomness of the algorithm and degrade performance. Both the reordering of struts prior to elimination and the strut selection probabilities specified by the non-zero value of *propowsel* appear to contribute to the improved performance of the algorithm.

The strut numbers of the last (smallest) set to obtain saturated damping is given in Table 4.4. Several of the struts included are the same as in the solution which used *propowsel* = 1.

Each run involved is set up to log how much time it took to run. These times are recorded in Table 4.5 (the times for *sachen.m* correspond to *propowsel* = 1). The specifications of the computers which performed the calculations are given in Appendix E. The Iris machine performed all NASTRAN computations presented in this dissertation.

Table 4.4: Optimal Design According to *sachen* with *propowsel* = 0

Strut Number	Node Pair	Type (D=diagonal)	Strut Number	Node Pair	Type (D=diagonal)
293	27–34	I	16	40–53	I
277	30–43	D	256	06–23	D
164	42–43	D	287	19–26	I
55	23–32	D	57	59–67	D
274	17–29	D	157	51–52	D
65	07–20	D	23	08–20	I
261	32–45	D	75	16–30	D
177	02–04	D	29	13–27	I
54	52–63	D	60	09–26	D
191	33–34	I	0	–	

Table 4.5: Run Time for Each Program Involved in Optimization Based on the Compliant Model

Process	Type	Machine	Speed MHz	Time, sec User	System	Real
akk1sh	NASTRAN	SGI Indigo/R4000	100	4,277	788	11,006
bonk.m	matlab	DEC 3000-300L	100	172	25	215
sachen.m	matlab	DEC 3000-300L	100	834	115	1032

4.3.5 Simulated Annealing Search Using NASTRAN Complex Eigenvalue Analysis

4.3.5.1 Description of Search In the preceding sections, the compliant model has been used to evaluate damping in order to speed the simulated annealing search for the optimal damping design. In this section, simulated annealing is done by using NASTRAN complex eigenvalue analysis to evaluate modal damping for function evaluations. This is a "truth model" against which the results of the simulated annealing of the compliant model may be compared for speed and accuracy.

The simulated annealing using complex eigenvalue analysis is to use nearly the same procedure as Chen used [6]. In this case, struts are selected and de-selected using uniform probabilities; that is, all struts outside of the current design have the same probability of being chosen for inclusion. When the algorithm reaches target damping in all modes so that the number of struts is to be reduced, the rightmost strut listed in the current combination is eliminated. The optimization within a given combination of struts is done by golden section. In order to keep the algorithm as similar as possible to Chen's, the thermal equilibrium counter is reset whenever the function evaluation improves over the previously accepted one (but not whenever a new design is accepted). However, thermal equilibrium occurs when the counter reaches 20 (Chen suggested 10).

Because of the time involved in performing the computations, n_{opt} begins at 20 instead of n_{ul} =46, with the struts of the initial design randomly selected. The beginning and final temperature are set to the same values as in the optimization (*sachen.m*) of the compliant model.

The solution is implemented using a Matlab file called *sanas.m*, which carries

out the simulated annealing procedure, selecting struts and testing whether or not new designs are to be selected. This program is listed in Appendix F. Each time the Matlab file needs a function evaluation, it "shells out" and calls on the NASTRAN program to determine modal damping.

4.3.5.2 Results of Simulated Annealing Search Using NASTRAN

Figure 4.7 shows three plots describing function evaluations, temperature, and number of struts as the algorithm progresses.

The results of the computation are divided into four segments. Segment A occurs between 1 and 248 function evaluations. During this segment, the J_{sat} evaluation sees relatively fast improvement. However, this portion of the computation was apparently stopped because of a file system problem.

Segment B occurs between function evaluations 249 and 714. In this run, the program *sanas.m* reaches its final convergence criteria without finding a combination which provided saturated damping in each mode. However, this condition is approached, with J_{sat} ending up at 0.6328 (saturated damping occurs at 0.6400). Only modes 4,10, and 13 are slightly short of reaching target damping.

The simulated annealing algorithm was restarted from the conclusion of segment B to see if further searching would yield a combination with saturated damping. Segments C and D, 715 to 1113 and 1114 to about 1300 function evaluations, respectively, are the results of these restarts. For some reason, a design giving saturated damping is not found. Possibly the algorithm is stuck in a local maximum.

The rest of the discussion applies to the results of segments A and B of the computation. The annealing pattern and temperature schedule appear similar to those of the simulated annealing optimization of the spherical compliant model estimate

performed by *sachen.m* with *propowsel* = 1.

The final strut combination is also similar to that obtained by *sachen.m*. This can be seen by comparing Table 4.6, which shows the struts included in the resulting design, to Table 4.3. Both designs include damping strut locations 163, 257, and 277. Furthermore, both designs contain several other diagonal locations which are ranked among the top 24 according to Table 4.2. Half of the struts in the final design of *sanas.m* are diagonal. Since only 72 of the 306 possible strut locations are diagonal, there is a clear preference toward diagonal struts by *sanas.m*.

However, the time involved in carrying out this optimization is much larger than that consumed with the compliant model, as Table 4.7 shows. A total of 714 function evaluations (searches by golden section) are done by calling NASTRAN, and each typically requires about 10 to 14 NASTRAN complex eigenvalue runs. Each of these runs typically requires 1:04.77 real minutes, 44.61 user seconds, and 7.95 system seconds. By comparison, 1389 function evaluations (690 of them golden section searches) are performed in optimizing the compliant model, considering all simulated annealing runs over the whole range of n_{opt}. The program using complex eigenvalue analysis consumes approximately 630 CPU seconds per golden section search, while the program using the compliant model takes only 0.68 CPU seconds per golden section search. Allowing for possible differences in computing speeds of the comparable DEC and SGI machines, the compliant model reduces golden section computation CPU time by over two orders of magnitude.

Because *sanas.m* starts with a random set of damping locations, about 180 function evaluations are consumed before J_{sat} even reaches 0.60. The *sachen.m* algorithm avoids this effort by starting with a combination of highly-ranked damping struts.

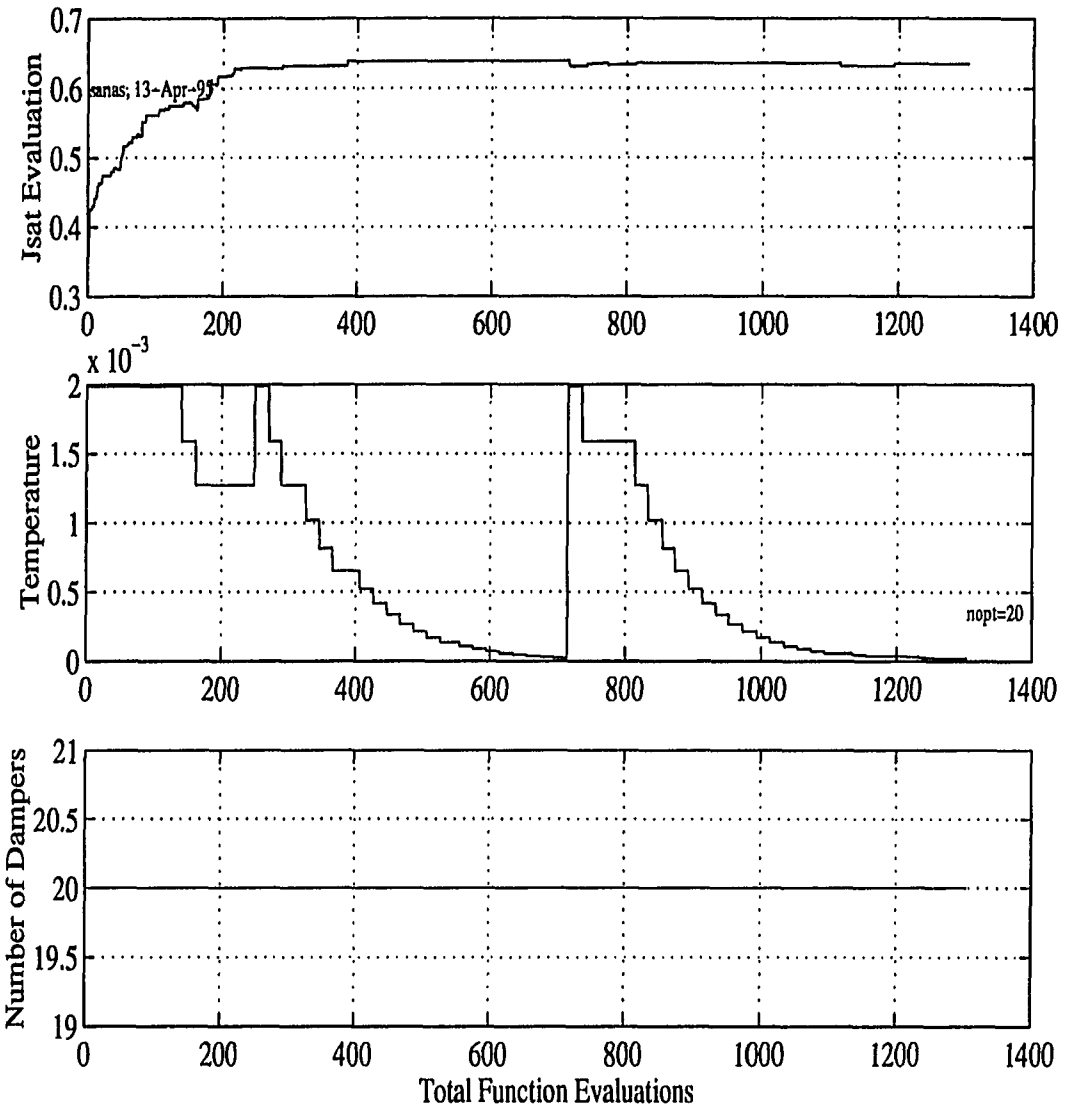

Figure 4.7: Results of Simulated Annealing Computation Using Complex Eigenvalue Analysis

Table 4.6: Optimal Design According to *sanas.m*

Strut Number	Node Pair	Type (D=diagonal)	Strut Number	Node Pair	Type (D=diagonal)
8	32–46	I	30	27–41	I
146	08–14	I	154	63–64	D
47	29–43	I	39	21–35	I
303	64–67	I	157	51–52	D
277	30–43	D	7	18–32	I
66	27–42	D	164	42–43	D
72	43–57	D	238	47–52	I
162	38–39	D	257	02–09	D
293	27–34	I	256	06–23	D
163	40–41	D	29	13–27	I

Table 4.7: Approximate Run Time for Optimization Based on Complex Eigevalue Analysis

Process	Type	Machine	Speed MHz	Time, sec User	System	Real
sanas.m	Matlab/NASTRAN	SGI Indigo/R4000	100	377K	68K	784K

CHAPTER 5. CONCLUSIONS

5.1 Summary of Goals

The purpose of this dissertation is to evaluate the effectiveness of viscous damping placed in diagonal truss locations in achieving optimal passive damping in several target modes. This is done by developing and implementing a model and procedure to estimate the damping coefficient produced by combinations of struts placed in various locations. This compliant model, which offers a fresh approach to structural modeling, greatly speeds function evaluations for the optimization search.

5.2 Summary of Methods Developed

Various analysis methods are developed in implementing the compliant model in order to search for optimal placement of passive damping in the example SPICE bulkhead structure. These methods are outlined below.

In the simple compliant model, the influence of stiffnesses and viscous dampers added to an existing structure are modeled by placing the added element in series with a stiffness in a spring-mass model of the structure. The spherical model combines the masses of compliant models for various modes into a single mass which may vibrate in multiple directions. The relative influence of an added strut on each mode determines the orientation of that strut within the spherical model. The spherical model can

then be expressed in compact state-space form. Methods of analyzing a structure to determine the parameters of this model by use of real eigenvalue analysis are also developed.

The compliant model leads to various means of ranking prospective damping locations according to predicted effectiveness in a passive damping design. These rankings are used to augment probabilities of selection of damping strut positions in order to increase the rate of convergence in a combinatorial search based on simulated annealing.

5.3 Summary of Important Results

5.3.1 Uniform Rod

As an example, the compliant model is used to predict eigenvalue shift in an axially vibrating uniform rod under the influence of an ideal spring or damper acting on its free end. In the case of a spring, the compliant model predicts frequency shift with moderate accuracy over the complete range of stiffness. In the case of a damper, the root of the compliant model predicts the modal damping ratio with moderate accuracy over most of the range of damping from zero to infinity. However, there is one value of damping coefficient at which the damping ratio peaks sharply to a value of 1.0. The model does not accurately predict damping near that value.

The example rod displays large eigenvalue shifts and and a theoretically infinite number of degrees of freedom. Even so, the compliant model permits fairly accurate predictions of eigenvalue shifts using information from only one mode at a time. Therefore the compliant model may be expected to predict frequency shifts even more accurately for large actual structures with lower potential for damping.

5.3.2 Lumped Paramter System

The compliant and spherical models are then used to predict damping in a system containing five lumped masses plus two intermediate nodes. The resulting estimations are compared to the predictions based upon several other methods including the modal strain energy method and complex eigenvalue analysis in a reduced vector space. The spherical model estimate supplied the most accurate predictions overall even though no complex eigenvalue analysis is used to obtain them. The roots obtained by complex eigenvalue analysis of the spherical model predict modal damping nearly perfectly in all five of the low-frequency modes even though only information corresponding to those five modes is used. When the same five modes are used as a reduced vector space to approximate damping, estimates of damping are inaccurate in the third through fifth mode. These results illustrate the potential of the spherical compliant model for modeling structural vibration and predicting damping in large structures accurately while including fewer modes in the analysis.

5.3.3 SPICE Bulkhead

An example in which viscous dampers were added to a finite element model of the SPICE bulkhead shows that the compliant model provides fairly accurate damping predictions for such a structure which contains multiple modes. The roots of the spherical model predicts damping even more accurately.

5.3.4 Optimization

The spherical compliant model greatly speeds the search for an optimal damping combination for the SPICE bulkhead structure. First, it allows function (damping)

evaluations without computationally expensive complex eigenvalue analysis. For a given combination of damping struts, only one real eigenvalue analysis on the target modes is necessary. The resulting compliant model parameters are then used to estimate damping repeatedly using a closed-form expression. As a result, the CPU time required for a optimization by golden section for a given combination is reduced by over two orders of magnitude.

The compliant model is also used to rank the struts for damping potential in the target modes. Then highly ranked struts are assigned higher probabilities of selection within the simulated annealing search algorithm. This results in a reduction in number of function evaluations required to complete the search. Furthermore, values of κ_{ij} supplied by the spherical model provide a means of quickly computing an upper bound on the damping index and rejecting some strut combinations which would certainly not be accepted by the algorithm.

Finally, examination of the smallest combination for which the simulated annealing algorithm using the spherical model estimate converged reveals that 15 of the 17 damper locations selected are diagonal; that is, they connect nodes not spanned by struts in the original structure. This implies that the original hypothesis asserting that these diagonal positions provide relatively high damping potential is true.

5.4 Future Research

The results presented in this dissertation provide several opportunities for future research.

Since diagonal damping locations have been shown to provide high damping, these locations should be exploited in future designs of passively damped tetrahedron

trusses.

The spherical compliant model greatly reduces the time needed to optimize a structure for passive damping. The procedure followed in this dissertation could be integrated in to a software package which would optimize local passive viscous damping for any structure. So far only springs and viscous dampers are considered in the compliant model. It should be augmented to include added mass as well.

The increased speed and decreased accuracy of the optimization based on the loss factor of the spherical model suggest a hybrid optimization approach. Such an algorithm would seek a near-optimal design using the loss factor or roots of the spherical model, and further optimize the design based on more accurate complex eigenvalue analysis.

Since the spherical compliant model may be expressed in compact, standard state space form, it is suitable to be analyzed by highly developed modern control techniques. This model could be used to actively control an actual structure embedded with actuators. The compliance included in each mode improves accuracy when a limited number of modes is included in the model, and may reduce problems with control spillover [14] which typically may occur due to modal truncation. Additionally, the few parameters involved in characterizing the model would facilitate system identification or even adaptive control.

BIBLIOGRAPHY

[1] Aubrun, J.N., 1980, "Theory of Control of Structures by Low-Authority Controllers," *Journal of Guidance, Control, and Dynamics*, Vol. 3, No. 5, pp. 444-51.

[2] Bland, D.R., 1960, *The Theory of Linear Viscoelasticity*, Pergamon Press, New York, p. 10.

[3] Blankenship, R.M., Breakwell, J.A., Dettmer, J.W., Hamilton, B.D., Richardson, J.M., 1992, "The SPICE Program," *Proceedings, Rocky Mountain Guidance and Control Conference,* Keystone, Colorado, February.

[4] Blankenship, R.M. , 1993, Conversation with Ross Blankenship, Lockheed Missiles and Space Company, December 1993.

[5] Caughey, T.K., and O'Kelly, M.E.J., 1965, "Classical Normal Modes in Damped Linear Dynamic Systems," *ASME Journal of Applied Mechanics*, September, pp. 583-8.

[6] Chen, G.S., Bruno, R.J., Salama, M., 1991, "Optimal Placement of Active/Passive Members in Truss Structures Using Simulated Annealing," *AIAA Journal*, Vol. 29, No. 8, August, pp. 1327-34.

[7] Davis, L.P. and Ginter, S.D., 1991, "An Advanced D-Strut," *Proceedings, Damping '91,* USAF Wright Lab., Flight Dynamics Dir., WL-TR-91-3078, San Diego, California, Feb. (reprint courtesy Honeywell).

[8] Foss, K.A., 1958, "Co-Ordinates which Uncouple the Equations of Motion of Damped Linear Dynamic Systems," *Journal of Applied Mechanics, ASME*, 25, pp. 361-4.

[9] Fox, R.L., Kapoor, M.P., 1968, "Rates of Change of Eigenvalues and Eigenvectors," *AIAA Journal*, Vol. 6, No. 12, December, pp. 2426-9.

[10] Fung, Y.C., 1965, *Foundations of Solid Mechanics*, Prentice-Hall, Englewood Cliffs, New Jersey, pp. 25-28.

[11] Gilheany, J.J., 1989, "Optimum Selection of Dampers for Freely Vibrating Multidegree of Freedom Systems," *Proceedings, Damping '89*, West Palm Beach, Florida, Flight Dynamics Laboratory of the Air Force, 8-10 February, pp. FCC-1 to FCC-18.

[12] MacNeal-Schwendler Corporation, 1983, *MSC/NASTRAN Handbook for Dynamic Analysis*, MSC/NASTRAN Version 63, M.A. Gockel, editor, June, p. 8.4-6.

[13] Hurty, W.C. and Rubenstein, M.F., 1964, *Dynamics of Structures*, Prentice-Hall, Englewood Cliffs, New Jersey, p. 340.

[14] Inman, D.J., 1989, *Vibration with Control, Measurement, and Stability*, Prentice Hall, Englewood Cliffs, New Jersey, pp. 9, 60-1, 70-1, 308-9.

[15] Johnson, C.D. and Kienholz, D.A., 1982, "Finite Element Prediction of Damping in Structures with Constrained Viscoelastic Layers," *AIAA Journal*, Vol. 20, No. 9, Sept., pp. 1284-1290.

[16] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P., 1983, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, 13 May, pp. 671-80.

[17] Lancaster, P., Tismenetsky, M., 1985, *The Theory of Matrices, Second Edition, With Applications*, Academic Press, Orlando, Florida, pp. 383-405.

[18] Li, T.H., 1993, *Using sensitivity derivatives of resonant response to guide structural redesign*, Ph.D. Dissertation, Iowa State University, Ames, Iowa.

[19] Meirovitch, L., 1980, *Computational methods in structural dynamics*, Sijthoff & Noordhoff, Rockville, Maryland, pp. 56, 61-5, 90-4, 102-9, 138-157.

[20] Mikaili, Afshin, 1992, *Using size sensitivities to guide structural shape optimization*, Ph.D. Dissertation, Iowa State University, Ames, Iowa.

[21] Milman, M., Salama, M., Scheid, R.E., Bruno, R., Gibson, J.S., 1991, "Combined control-structure optimization," *Computational Mechanics* (1991) 8, pp. 1-18.

[22] Mimovich, M.E., 1992, *Correlation of the SPICE Beam Expander Structural Model with Component and System Level Modal Test Results*, Report for Contract #F29601-91-C-0025, Phillips Laboratory Structures and Controls Division (PL/VTSA), Kirtland AFB, New Mexico, June.

[23] Neubert, V.H., 1989, "Optimization of Energy Dissipation Rate in Structures," *Proceedings, Damping '89*, West Palm Beach, Florida, Flight Dynamics Laboratory of the Air Force, 8-10 February, pp. FCD-1 to FCD-24.

[24] Rayleigh, J.W.S. (Baron), 1894, *The Theory of Sound*, 2nd ed., Volume I, Macmillan, London (Reprinted by Dover Publications, New York, 1945), p. 133.

[25] SPace Integrated Controls Experiment (SPICE), 1992, Technical Interchange Meeting, Task Area 2, Albuquerque, New Mexico, June 24.

[26] Thompson, T.J. and Baumgarten, J.R., 1993, "A Brief Study of Passive Viscous Damping for the SPICE Bulkhead Structure," *Proceedings, The 14th Biennial Conference on Mechanical Vibration and Noise* (ASME), Albuquerque, New Mexico, September.

[27] Thompson, T.J., Baumgarten, J.R., Flugrad, D.R., 1994, "Passive Damping Prediction and Optimization for the SPICE Structure Using a Maxwell Structural Model," *Final Report for Summer Research Extension Program* at Phillips Laboratory for the Air Force Office of Scientific Research, Bolling Air Force Base, Washington, D.C., March.

[28] Thompson, T.J. and Baumgarten, J.R., 1994, "Passive viscous damping of a complex strut-built structure," *Proceedings, 19th International Seminar on Modal Analysis and Structural Dynamics*, Leuven, Belgium, September.

[29] W.T. Thomson, 1981, *Theory of Vibration with Applications* 2nd Ed., Prentice Hall, Englewood Cliffs, New Jersey, pp. 27, 70, 75, 132-9, 197.

[30] Weaver, W. Jr., Timoshenko, S.P., Young, D.H., 1990, *Vibrational Problems in Engineering*, 5th ed., John Wiley & Sons, New York, pp. 72, 221, 275, 393.

[31] Wilde, D.J., 1964, *Optimum Seeking Methods*, Prentice-Hall, Englewood Cliffs, New Jersey, pp. 32-35.

[32] Yiu, Y.C., Davis, L.P., Napolitano, K., and Ninneman, R., 1992a,"Design, Analysis, and Testing of High Frequency Passively Damped Struts," *Fifth NASA-DoD Controls-Structures-Interaction Technology Conference,* Lake Tahoe, Nevada, March, pp. 239-249.

[33] Yiu, Y.C., Davis, L.P., and Kienholz, D.A., 1992b, "Development of High Frequency Passively Damped Struts," Final Report for Subtask 02-08, Task 4, Space Integrated Controls Experiment Program, Contract No: F29601-89-C0015, April, pp. 23-26.

[34] Yiu, Y.C., Weston, E.L., "Computation of Complex Modes in Reduced Vector Subspaces," 1992c, *Proceedings, 33rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Paper 92-2515, Dallas, April 13-15.

ACKNOWLEDGMENTS

The help of the following parties is gratefully acknowledged.

Mr. Rory Ninneman of Phillips Lab, Kirtland AFB, New Mexico, helped conceptualize the problem statement and provided technical and computational support.

The Air Force Office of Scientific Research provided funding for much of this work through the Summer Research Program (1992) and the Summer Research Extension Program (proposal 93-157).

The Carver Lab at Iowa State University supplied computer time for the NASTRAN computations.

APPENDIX A. MATLAB PROGRAMS FOR ESTIMATING EIGENVALUE SHIFTS IN A CONTINUOUS ROD

The following program is used to compute the frequency shift in a continuous rod acted upon by a spring at the free end. This is referred to in Section 3.6.1. In this and other appendices, files with *.m* extensions are Matlab script or function files.

cont.m:

```
% cont.m
% Matlab script to solve and plot the
% transcendental equation for axial vibrations
% in a continuous beam with a spring
% on the end.
% 6-30-94
clear;
a=5;
ell=1;
m=3;
del=100;
low=10;
delk=1.15;
klim=60;
 set(gcf,'DefaultTextFontName','Times');
 set(gcf,'DefaultTextFontSize',8);
 set(gcf,'DefaultAxesFontName','Times');
for jj=1:3
 omo=(jj-.5)*pi*a/ell;
 k=low;
 for kk=1:klim;
  %k=(kk-.99)*del;
  k=k*delk;
  om1=omo+0.00001;
  xi=om1*ell/a;
  f1=xi*tan(xi)+m*ell*(om1^2)/k;
  om2=(jj*pi*a/ell);
  xi=om2*ell/a;
  f2=xi*tan(xi)+m*ell*(om2^2)/k;
  f=1;
  while abs(f)>.001
   omg=(om1+om2)/2;
   xi=omg*ell/a;
   f=xi*tan(xi)+m*ell*(omg^2)/k;
   if f*f1>0
    f1=f;
    om1=omg;
    else
    f2=f;
    om2=omg;
```

```
    end % if
   end % while
   kkk(kk)=k;
   omans(kk)=omg;
   if kk~=1
    d1(kk)=(omans(kk)-omans(kk-1))/del;
    if kk~=2
      d2(kk)=(d1(kk)-d1(kk-1))/del;
     else
      d2(kk)=0;
    end
    else
     d1(kk)=0;
   end
  end % for
  % Estimate omega's by slope at zero and frequency at infinity.
  domega2=2/m/ell;
  omega2inf=(jj*pi*a/ell)^2;
  kb=omega2inf-omo^2;
  kkeg=low;
  for k=1:klim
   %kke(k)=(k-.99)*1*del;
   kkeg=kkeg*delk;%
   kke(k)=kkeg;%
   ome(k)=sqrt(omo^2 + domega2*kke(k)*kb/(domega2*kke(k)+kb));
   erp(k)=(ome(k)-omans(k))*100/(ome(k)-omo);
  end % for
  subplot(2,1,1),semilogx(kkk,omans,'y-',kke,ome,'m-.');
  grid on;
  mode=num2str(jj);
  legend(['Theoretical, Mode ' mode],'Compliant Model Estimate');
  ylabel('Omega, 1/s');
  subplot(2,1,2),semilogx(kkk,erp,'r-');
  grid on;
  legend(['Percent Error in Frequency Shift Estimation, Mode ' mode]);
  ylabel('Percent');
  xlabel('k, force/length');
  yes=0;
  disp('Enter 1 to save to a postscript file.');
  yes=input('');
  if yes==1
   eval(['print mode',num2str(jj)]);
  end
 end % for jj
  set(gcf,'DefaultTextFontSize',12);
  set(gcf,'DefaultTextFontName','Helvetica');
  set(gcf,'DefaultAxesFontName','Helvetica');
```

The following program is used to compute the damping in a continuous rod acted upon by a damper at the free end. This is referred to in Section 3.6.1.

cdamp.m:

```
% Matlab file to compute theoretical damping on an
% ideally-damped uniform rod vibrating axially.
% Viscous dashpot applies force at the free end.
%
% Thomas J. Thompson
%
% 7-8-94
%
clear;
a=5;
m=3;
ell=1;
```

```
delc=.05;
cmin=1;
cmax=100;
kkmax=500;
base=(cmax/cmin)^(1/(kkmax-1));
mm=cmin/base;
alpha=2/m/ell;
 set(gcf,'DefaultTextFontName','Times');
 set(gcf,'DefaultTextFontSize',8);
 set(gcf,'DefaultAxesFontName','Times');
for jj=1:5 % Modes 1 through 3
 xi(jj)=(jj-.5)*pi;
 omo(jj)=a*xi(jj)/ell;
 ka=omo(jj)^2;
 xii(jj)=jj*pi;
 omi(jj)=a*xii(jj)/ell;
 ks(jj)=omi(jj)^2-ka;
 kappa(jj)=ks(jj)/ka;
 for kk=1:kkmax  % For each value of c.
  c(kk)=mm*base^kk;
  [jj kk]
  c(jj)
  % Get the value of theoretical damping coefficient.
  qbig=c(kk)/a/m;
  if qbig<1.
   vbig=sqrt((1-qbig)/(1+qbig));
   zetax2(kk,jj)=-2*log(vbig)/sqrt(xi(jj)^2 + (log(vbig))^2);
  end
  if qbig==1.00
   zetax2(kk,jj)=2.;
  end
  if qbig>1.00
   vbig=sqrt((qbig-1)/(1+qbig));
   zetax2(kk,jj)=-2*log(vbig)/sqrt(xii(jj)^2 + (log(vbig))^2);
  end
 %  Do estimates of damping coefficient.
  r=alpha*c(kk)*omo(jj)/ks(jj);
  eta(kk,jj)=kappa(jj)*r/(1+(r^2)*(1+kappa(jj)));
% Get roots of the compliant model.
  f=1;
  if kk==1
    lambdac=i*(omo(jj)+.00001);
  end % if
  while abs(f)>.0001
   f=lambdac^3*alpha*c(kk)+lambdac^2*ks(jj)+alpha*c(kk)*lambdac*(ka+ks(jj))+ka*ks(jj);
   fpr=3*alpha*c(kk)*lambdac^2 + 2*ks(jj)*lambdac + alpha*c(kk)*(ka+ks(jj));
   lambdac=lambdac-.95*f/fpr;
  end % while
  lamcvec(kk)=lambdac;
  zetacx2(kk,jj)=2*(-real(lambdac))/sqrt(imag(lambdac)^2+real(lambdac)^2);
 end % kk
clf;
semilogx(c,zetax2(:,jj),'y-',c,eta(:,jj),'m-.',c,zetacx2(:,jj).','r--');
axis([1 100 0 1]);
legend(['Theoretical Mode ' num2str(jj)],'Compliant Model Estimate',...
'Root of Compliant Model');
grid on;
xlabel('Damping Coefficient c, force*time/length');
ylabel('2 * Modal Damping Ratio');
yes=0;
disp('Enter 1 to save to a postscript file.');
yes=input('');
if yes==1
 eval(['print dmodo',num2str(jj)]);
end
end  % jj
 set(gcf,'DefaultTextFontSize',12);
 set(gcf,'DefaultTextFontName','Helvetica');
 set(gcf,'DefaultAxesFontName','Helvetica');
```

APPENDIX B. MATLAB PROGRAM FOR ESTIMATING EIGENVALUE SHIFTS FOR A CONTINUOUS ROD INFLUENCED BY A SERIES COMBINATION

```
% concom.m
% This is a Matlab program to find the frequency
% and damping in a continuous beam acted upon
% by a spring-(spring or damper) in series.
% It also compares the results with predictions
% of the compliant model.
%
% 7-12-94
%
clear;
a=5;
ell=1;
m=3;
mult=4;
delk=1.2;
delc=.09;
low=1;
 set(gcf,'DefaultTextFontName','Times');
 set(gcf,'DefaultTextFontSize',8);
 set(gcf,'DefaultAxesFontName','Times');
kc=mult*m*a*a/ell;
alpha=2/m/ell;
for jj=1:3   % Each mode jj
 omo(jj)=(jj-.5)*pi*a/ell;   % Frequencies at ki or ci = 0
 ominf(jj)=jj*pi*a/ell;
 % Find the frequency at kplain=kc
 f=1;
 omg=omo(jj)+.00001;
 while abs(f)>.0001
  f=tan(omg*ell/a)+m*a*omg/kc;
  fpr=ell/a/(cos(omg*ell/a))^2 + m*a/kc;
  omg=omg-f/fpr;
 end
 omkc(jj)=omg;
 % Information for the compliant model
 ka=omo(jj)^2;
 kb=omkc(jj)^2-ka;
 kappa=kb/ka;
 %
 klim=50;
 k=low;
 for kk=1:klim
  kk
  %ki(kk)=delk*(kk-.99);
  k=k*delk;%
  ki(kk)=k;%
  % Iterate to get omega for each ki value.
  if kk==1
    omg=omo(jj)+.00001;
   else
```

```
  omg=om(kk-1);
 end % if
 kplain=kc*ki(kk)/(kc+ki(kk));
 f=1;
 while abs(f)>.0001
  f=tan(omg*ell/a)+m*a*omg/kplain;
  fpr=ell/a/(cos(omg*ell/a))^2 + m*a/kplain;
  omg=omg-f/fpr;
 end % while
 om(kk)=omg;
 % Get estimate for frequency from compliant model
 ome(kk)=sqrt(ka+  (kb*alpha*ki(kk))/(kb + alpha*ki(kk)) );
 erp(kk)=(ome(kk)-om(kk))*100/(ome(kk)-omo(jj));
 % Get roots by complex Newton method
 ci(kk)=(1+delc)^kk;
 f=1;
 if kk==1
   lambdag=i*(omo(jj)+.00001);
 end % if
 while abs(f)>.0001
  kplain=ci(kk)*lambdag*kc/(ci(kk)*lambdag+kc);
  f=tanh(lambdag*ell/a)+a*m*lambdag/kplain;
  fpr=ell/a/(cosh(lambdag*ell/a))^2 - a*m/lambdag^2/ci(kk);
  lambdag=lambdag-.95*f/fpr;
 end % while
 lamvec(kk)=lambdag;
 zetax2(kk)=2*(-real(lambdag))/sqrt(imag(lambdag)^2+real(lambdag)^2);
 r=ci(kk)*alpha*omo(jj)/kb;
 eta(kk)=r*kappa/(1+(r^2)*(1+kappa));
 % Get actual roots of compliant model
 f=1;
 if kk==1
  lambdac=i*omo(jj)-.1;
 end % if
 while abs(f)>.001
  f=lambdac^3*alpha*ci(kk)+lambdac^2*kb+alpha*ci(kk)*lambdac*(ka+kb)+ka*kb;
  fpr=3*alpha*ci(kk)*lambdac^2 + 2*kb*lambdac + alpha*ci(kk)*(ka+kb);
  lambdac=lambdac-.95*f/fpr;
 end % while
 lamcvec(kk)=lambdac;
 zetacx2(kk)=2*(-real(lambdac))/sqrt(imag(lambdac)^2+real(lambdac)^2);
end % kk
%set(gcf,'PaperPosition',[0 0 8.5 11])
subplot(2,1,1),semilogx(ki,om,'y-',ki,ome,'m-.');
grid on;
mode=num2str(jj);
legend(['Theoretical, Mode ' mode],'Compliant Model Estimate');
ylabel('Omega, 1/s');
subplot(2,1,2),semilogx(ki,erp,'r-');
grid on;
legend(['Percent Error in Frequency Shift Estimation, Mode ' mode]);
ylabel('Percent');
xlabel('k, force/length');
yes=0;
disp('Enter 1 to save to a postscript file.');
yes=input('');
if yes==1
 eval(['print fr',num2str(jj),'mult',num2str(mult)]);
end
clf;
semilogx(ci,zetax2,'y-',ci,zetacx2,'m-.',ci,eta,'c--');
legend(['Theoretical Damping Coefficient, Mode ' mode], ...
 'Damping from Root of Compliant Model',...
 'Compliant Estimate of Damping');
ylabel('2 x Damping Ratio');
xlabel('Damping Coefficient, force*time/length');
grid on;
yes=0;
disp('Enter 1 to save to a postscript file.')
yes=input('');
if yes==1
```

```
 eval(['print damp',num2str(jj),'mult',num2str(mult)]);
end
% Plot Root locus of continuous and compliant models
plot(real(lamvec),imag(lamvec),'y-',real(lamcvec),imag(lamcvec),'m-.')
legend('Continuous Model','Compliant Model');
xlabel('Real, 1/sec');
ylabel('Imaginary, i/sec');
grid on;
axis('equal');
yes=0;
disp('Enter 1 to save to a postscript file.')
yes=input('');
if yes==1
 eval(['print rloc',num2str(jj),'mult',num2str(mult)]);
end
end % jj
set(gcf,'DefaultTextFontSize',12);
set(gcf,'DefaultTextFontName','Helvetica');
set(gcf,'DefaultAxesFontName','Helvetica');
```

APPENDIX C. MATLAB PROGRAM FOR COMPUTING DAMPING IN A LUMPED-PARAMETER STRUCTURE

lump.m

```
% This is a Matlab program to solve a
% lumped-mass damping program using various methods.
%
% Thomas J. Thompson
%
% 7-21-94
%
% Define system

clear;
m=[2 2 2 2 2 .00001 .00002];
M=diag(m);

% Spring values. Indices are node numbers; identical
% indices denote grounded springs.

k(1,1)=.5;
k(1,2)=.5;
k(2,2)=1;
k(1,3)=.7277;            %1.0743;
k(2,3)=1;
k(2,4)=1;
k(3,4)=1;
k(3,5)=.7277;             %1.0743;
k(4,5)=.5;
k(4,4)=1;
k(4,6)=.6  ; % Connected to intermediate node.
k(5,5)=.5;
k(5,7)=.6  ; % Connected to intermediate node.

K=zeros(7,7);
for ii=1:5
 if k(ii,ii)~=0
  K(ii,ii)=K(ii,ii)+k(ii,ii);
 end
 for jj=(ii+1):(ii+2)
  if k(ii,jj)~=0
   K(jj,ii)=K(jj,ii)-k(ii,jj);
   K(ii,jj)=K(ii,jj)-k(ii,jj);
   K(ii,ii)=K(ii,ii)+k(ii,jj);
   K(jj,jj)=K(jj,jj)+k(ii,jj);
  end % if
 end % for jj
end % for ii

% Perform normal mode analysis.

[V, E] = eig(K,M);
[e, jvec] = sort(diag(E));
```

```
Lambda=diag(e);
V=V(:, jvec);
for jj=1:7
 Phi(:,jj)=V(:,jj)/(sqrt(V(:,jj).'*M*V(:,jj)));
end
lambda5=diag(Lambda(1:5,1:5));
lambda5rt=sqrt(lambda5);

eigchk=norm(K*Phi-M*Phi*Lambda);
kchk=norm(Phi.'*K*Phi-Lambda);
mchk=norm(Phi.'*M*Phi-eye(7));

% Set values of damping coefficients to be evaluated.

cmin=.01;
cmax=10;
nsteps=15;
base=(cmax/cmin)^(1/nsteps);

% Do complex analysis

ceig

% Do complex analysis estimate

%cest

% Do modal strain energy method

mse

% Do compliant model estimation

comp

% Do compliant model estimation based
% partially upon perturbation estimation
% of frequency shifts.

pcomp

% Do estimate based on complex perturbation
% method.

pert

% Do estimate based on reduced subspace

reds

% Plot results

s1='o';
s2='x';
s3='+';
s4='*';
s5='.';
p1='-';
p2='--';
p3='-.';
p4=':';
p5='-.';
c1='y';
c2='m';
c3='c';
c4='r';
c5='g';
c6='b';
c7='w';
res1='etams';
res2='etaco';
```

```
res3='etapc';
res4='etapt';
meth1='Reduced Space Solution';
meth2='Compliant Model Estimate';
meth3='MSE Method';
meth4='Compliant with Perturbation';
meth5='Complex Perturbation Method';
meth6='Spherical Model Estimate';
  set(gcf,'DefaultTextFontName','Times');
  set(gcf,'DefaultTextFontSize',8);
  set(gcf,'DefaultAxesFontName','Times');
for lll=1:4
 yes=0;
 method=['meth' num2str(lll)];
 result=['res' num2str(lll)];
 disp(['To plot results from ' eval(method) ', enter 1.']);
 yes=input('');
 if yes==1
  clf;

semilogx(c,2*zeta(1,:),[c1 s1], ...
  c,2*zeta(2,:),[c2 s2], ...
c,2*zeta(3,:),[c3 s3], ...
c,2*zeta(4,:),[c4 s4], ...
c,2*zeta(5,:),[c5 s5], ...
c, eval([eval(result) '(1,:)']) ,[c1 p1], ...
c, eval([eval(result) '(2,:)']) ,[c2 p2], ...
c, eval([eval(result) '(3,:)']) ,[c3 p3], ...
c, eval([eval(result) '(4,:)']) ,[c4 p4], ...
c, eval([eval(result) '(5,:)']) ,[c5 p5] )
   legend(['Mode ' num2str(1)],...
          ['Mode ' num2str(2)],...
          ['Mode ' num2str(3)],...
          ['Mode ' num2str(4)],...
          ['Mode ' num2str(5)],...
  ['Mode ' num2str(1) ' ' eval(method)],...
  ['Mode ' num2str(2) ' ' eval(method)],...
  ['Mode ' num2str(3) ' ' eval(method)],...
  ['Mode ' num2str(4) ' ' eval(method)],...
  ['Mode ' num2str(5) ' ' eval(method)])
   pause;
   xlabel('Damping Coefficient, force*time/length')
   ylabel('2 x Modal Damping Ratio')
   grid on;
 %set(gca,'FontSize',12)
 end % if
end % for lll
for jj = 1:5
 yes=0;
 disp(['Enter 1 to see results for mode ' num2str(jj) ' plotted.'])
 yes=input('');
 if yes==1
  semilogx(c,2*zeta(jj,:),[c1 s1], ...
           c,2*zeta1(reet(jj),:),[c2 s2],...
           c,etaco(jj,:) ,[c3 p1], ...
           c,etams(jj,:) ,[c4 p2], ...
           c,etapc(jj,:) ,[c5 p3], ...
           c,etapt(jj,:) ,[c6 p4], ...
           c,neta(jj,:),[c7 s3]);

  rmax=max...
  ([2*zeta(jj,:) etams(jj,:) etaco(jj,:) etapt(jj,:) etapc(jj,:) neta(jj,:)]);
  legend(['Mode ' num2str(jj)], ...
        [meth1],[meth2],[meth3],[meth4],[meth5],[meth6]);
   xlabel('Damping Coefficient, force*time/length');
   ylabel('2 x Modal Damping Ratio');
 grid on;
  disp('Enter the vertical axis upper range for the present plot.')
 range=input('range=');
 axis([cmin cmax 0 range]);
  yes=0;
```

```
   disp('Enter 1 to save to a postscript file.');
   yes=input('');
   if yes==1
    eval(['print lmode' num2str(jj) 'k' num2str(k(4,6))]);
   end % if
  end % if
end % for
disp(['Enter 1 to see results for sum of modes plotted.'])
yes=input('');
if yes==1
  semilogx(c,sum(2*zeta),[c1 s1], ...
           c,sum(2*zeta1),[c2 s2],...
           c,sum(etaco) ,[c3 p1], ...
           c,sum(etams) ,[c4 p2], ...
           c,sum(etapc) ,[c5 p3], ...
           c,sum(etapt) ,[c6 p4], ...
           c,sum(neta),[c7 s3]);

 legend(['Sum of Modes'], ...
        [meth1],[meth2],[meth3],[meth4],[meth5],[meth6]);
  xlabel('Damping Coefficient, force*time/length');
  ylabel('2 x Modal Damping Ratio');
 grid on;
  disp('Enter the vertical axis upper range for the present plot.')
 range=input('range=');
 axis([cmin cmax 0 range]);
 yes=0;
 disp('Enter 1 to save to a postscript file.');
 yes=input('');
 if yes==1
  eval(['print lsumk' num2str(k(4,6))]);
 end % if
end % if
  set(gcf,'DefaultTextFontSize',12);
  set(gcf,'DefaultTextFontName','Helvetica');
  set(gcf,'DefaultAxesFontName','Helvetica');
```

ceig.m

```
% ceig.m
% Matlab program to compute damping in the system
% through complex eigenvalue analysis.
%
% Thomas J. Thompson
%
% 7-21-94
%

C=zeros(7,7);

for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end

% Define C matrix.
for ii=[5]
 C(ii,ii)=c(kkk);
 C(ii,ii+1)=-c(kkk);
 C(ii+1,ii)=-c(kkk);
 C(ii+1,ii+1)=c(kkk);
end
C(7,7)=c(kkk);

Astate=[-Phi.'*C*Phi -Lambda; eye(7) zeros(7,7)];
```

```
[omnc,zetac]=damp(Astate);
[omnc,jsortc]=sort(omnc);
zetac=zetac(jsortc);
i5=1;
 for ii=1:14
  if omnc(ii) < min(lambda5rt)
    omnc(ii)=0;
  elseif omnc(ii) > 1.2*max(lambda5rt)
    omnc(ii)=0;
  end
  if ii~=14
   if abs(omnc(ii)-omnc(ii+1))<.01*min(lambda5rt)
    omnc(ii)=0;
   end % if
  end % if
  if abs(zetac(ii)-1.0)<.001
   omnc(ii)=0;
  end % if
  if omnc(ii)~=0
   omn(i5,kkk)=omnc(ii);
   zeta(i5,kkk)=zetac(ii);
   i5=i5+1;
   if i5>6
    disp('i5 problem: pause')
    pause
   end % if
  end % if
 end % for ii
end
```

mse.m

```
% Matlab file to estimate the damping of
% a lumped-parameter structure by the modal
% strain energy method.
%
%  Thomas J. Thompson
%
% 7-21-94
%
% Get modal strain energy ratio for mode jj, strut ii;
% strut 1 is between nodes 4 and 5, and strut 2 is between
% node 5 and the ground.

K1=zeros(7,7);
K2=zeros(7,7);
K1(4,4)=k(4,5);
K1(4,5)=-k(4,5);
K1(5,4)=-k(4,5);
K1(5,5)=k(4,5);
K2(5,5)=k(5,5);
Phit=Phi.';

% Modal strain energy of each strut ii

for ii=1:2
 for jj=1:5
  eval(['se(ii,jj)=Phit(jj,:)*K' num2str(ii) '*Phi(:,jj)/2;']);
 end % jj
end % ii

% Modal strain energy of mode jj

for jj=1:5
 mset(jj)=Phit(jj,:)*K*Phi(:,jj)/2;
end % jj
```

```
% Ratios of modal strain energy

for jj=1:5
 VV(:,jj)=se(:,jj)/mset(jj);
end

ki(1)=k(4,6);
ki(2)=k(5,7);
kappa(1)=ki(1)/k(4,5);
kappa(2)=ki(2)/k(5,5);

% Step through values of c

for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end

 % Get dimensionless r(ii,jj)

 for ii=1:2
  for jj=1:5
   r(ii,jj)=lambda5rt(jj)*c(kkk)/ki(ii);
  end % jj
 end % ii

 % Get loss factor for each strut

 for jj=1:5
  for ii=1:2
   num= kappa(ii)*(r(ii,jj)/(1+r(ii,jj)^2));
   den= 1 + kappa(ii)*((r(ii,jj)^2)/(1+r(ii,jj)^2));
   etast(ii,jj)=num/den;
  end % ii
 end % jj

 % Get modal loss factor estimate

 for jj=1:5
  etams(jj,kkk)=sum(etast(:,jj).*VV(:,jj));
 end % jj

end
```

comp.m

```
% comp.m
% Matlab program to use compliant model to
% estimate modal damping in a lumped-parameter
% model.
%
%  Thomas J. Thompson
%
% 7-22-94
%

% Get kA values

kA=lambda5;

% Get alpha(ii,jj) values for each mode jj and strut ii

for jj=1:5
 alpha(1,jj)=(Phi(5,jj)-Phi(4,jj))^2;
 alpha(2,jj)=Phi(5,jj)^2;
```

```
end % jj

% Get ks(ii,jj) by auxiliary real eigenvalue analysis

Kc1=zeros(7,7);
Kc2=zeros(7,7);
Kc1(4,4)=k(4,6);
Kc1(4,5)=-k(4,6);
Kc1(5,4)=-k(4,6);
Kc1(5,5)=k(4,6);
Kc2(5,5)=k(5,7);

for ii=1:2
 eval(['Kaux=K+Kc' num2str(ii) ';']);
 [V, E] = eig(Kaux,M);
 [e, jvec] = sort(diag(E));
 Lambdaaux=diag(e);
 V=V(:, jvec);
 for jj=1:7
  Phiaux(:,jj)=V(:,jj)/(sqrt(V(:,jj).'*M*V(:,jj)));
 end
 lambda5aux=diag(Lambdaaux(1:5,1:5));
 eigchkaux=norm(Kaux*Phiaux-M*Phiaux*Lambdaaux);
 kchkaux=norm(Phiaux.'*Kaux*Phiaux-Lambdaaux);
 mchkaux=norm(Phiaux.'*M*Phiaux-eye(7));
 normaux=norm([eigchkaux kchkaux mchkaux]);
 if normaux > .0001
  disp(['Norm of aux normal mode analysis is ' num2str(normaux)]);
  pause;
 end % if
 for jj=1:5
  ks(ii,jj)=lambda5aux(jj)-kA(jj);
 end % jj
end % ii

% Redistribute ks according to alpha

for ii=1:2
 kssum(ii)=sum(ks(ii,4:5));
 alphasum(ii)=sum(alpha(ii,4:5));
end % ii
for ii=1:2
 ks(ii,4:5)=kssum(ii)*alpha(ii,4:5)/alphasum(ii);
end % ii

% Redistribute ks according to spherical model
% and compute damping in a separate script program.

compsph

% Compute values of kappas(ii,jj)

for jj=1:5
 for ii=1:2
  kappas(ii,jj)=ks(ii,jj)/kA(jj);
 end % ii
end % jj

% Set values of c

for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end

 % Get values of r(ii,jj)

 for jj=1:5
  for ii=1:2
```

```
   r(ii,jj)=alpha(ii,jj)*c(kkk)*lambda5rt(jj)/ks(ii,jj);
  end % ii
 end % jj

 %Get values of modal loss factor etaco(jj)

 for jj=1:5
 num=0;
 den=1;
  for ii=1:2
   r2=r(ii,jj)^2;
   num=num+kappas(ii,jj)*(r(ii,jj)/(1+r2));
   den=den+kappas(ii,jj)*r2/(1+r2);
  end % ii
  etaco(jj,kkk)=num/den;
 end % jj

 % Get actual roots of compliant model

 for jj=1:5
  f(jj)=1;
  if kkk==1
   sc(jj)=i*lambda5rt(jj)-.1;
  end % if
  while abs(f(jj))>.0001
   f(jj)=sc(jj)^2+kA(jj);
   fpr(jj)=2*sc(jj);
   for ii=1:2
    f(jj)=f(jj)+(ks(ii,jj)*alpha(ii,jj)*c(kkk)*sc(jj))/...
              (ks(ii,jj)+alpha(ii,jj)*c(kkk)*sc(jj));
    fpr(jj)=fpr(jj)+(alpha(ii,jj)*c(kkk)*ks(ii,jj)^2)/...
((alpha(ii,jj)*c(kkk)*sc(jj)+ks(ii,jj))^2);
   end % for ii
   sc(jj)=sc(jj)-.95*f(jj)/fpr(jj);
  end % while
  rsc=real(sc(jj));
  isc=imag(sc(jj));
  zetax2co(jj,kkk)=-2*rsc/sqrt(rsc^2 + isc^2);
 end % jj
end % kkk
```

pcomp.m

```
% pcomp.m
% Matlab program to use compliant model based
% partially upon perturbation method to
% estimate modal damping in a lumped-parameter
% model.
%
%  Thomas J. Thompson
%
% 7-22-94
%

% Get kA values

kA=lambda5;

% Get alpha(ii,jj) values for each mode jj and strut ii

for jj=1:5
 alpha(1,jj)=(Phi(5,jj)-Phi(4,jj))^2;
 alpha(2,jj)=Phi(5,jj)^2;
end % jj

% Get ks(ii,jj) estimate by perturbation estimate

Kc1=zeros(7,7);
```

```
Kc2=zeros(7,7);
Kc1(4,4)=k(4,6);
Kc1(4,5)=-k(4,6);
Kc1(5,4)=-k(4,6);
Kc1(5,5)=k(4,6);
Kc2(5,5)=k(5,7);

for ii=1:2
 eval(['Kaux=K+Kc' num2str(ii) ';']);
 Lambdapc=diag(Phi.'*Kaux*Phi);
 for jj=1:5
  kspc(ii,jj)=Lambdapc(jj)-kA(jj);
 end % jj
end % ii

% Compute values of kappas(ii,jj)

for jj=1:5
 for ii=1:2
  kappaspc(ii,jj)=kspc(ii,jj)/kA(jj);
 end % ii
end % jj

% Set values of c

for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end

 % Get values of rpc(ii,jj)

 for jj=1:5
  for ii=1:2
   rpc(ii,jj)=alpha(ii,jj)*c(kkk)*lambda5rt(jj)/kspc(ii,jj);
  end % ii
 end % jj

 %Get values of modal loss factor etapc(jj)

 for jj=1:5
 num=0;
 den=1;
  for ii=1:2
   r2=rpc(ii,jj)^2;
   num=num+kappaspc(ii,jj)*(rpc(ii,jj)/(1+r2));
   den=den+kappaspc(ii,jj)*r2/(1+r2);
  end % ii
  etapc(jj,kkk)=num/den;
 end % jj
end % kkk
```

pert.m

```
% pert.m
% Matlab program to use complex perturbation
% method to estimate modal damping in a
% lumped-parameter model.
%
%  Thomas J. Thompson
%
% 7-22-94
%

% Get B matrices
```

```
B1=zeros(7,7);
B2=zeros(7,7);
B1(4,4)=1;
B1(4,5)=-1;
B1(5,4)=-1;
B1(5,5)=1;
B2(5,5)=1;

Phit=Phi.';

% Get values for kp and kappap(ii,jj)

kp(1)=k(4,6);
kp(2)=k(5,7);

for jj=1:5
 for ii=1:2
 kbot=Phit(jj,:)*Phi(:,jj);
  eval(['kappap(ii,jj)=kp(ii)*Phit(jj,:)*B' num2str(ii) ...
      '*Phi(:,jj)/lambda5(jj)/kbot;']);
 end % ii
end % jj

% Set values of c

for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end

% Get values for rp(ii,jj)

 for jj=1:5
  for ii=1:2
   rp(ii,jj)=lambda5rt(jj)*c(kkk)/kp(ii);
  end % ii
 end % jj

 %Get values of modal loss factor etapt(jj)

 for jj=1:5
 num=0;
 den=1;
  for ii=1:2
   r2=rp(ii,jj)^2;
   num=num+kappap(ii,jj)*(rp(ii,jj)/(1+r2))/2;
   den=den+kappap(ii,jj)*r2/(1+r2)/2;
   etapt(jj,kkk)=num/den;
  end % ii
  etapt(jj,kkk)=num/den;
 end % jj
end % kkk
```

reds.m

```
% reds.m
% Matlab program to get the complex eigenvalues
% of the lumped mass example using a reduced
% subspace of the normal modes.
%
%  Thomas J. Thompson
%
%  10-1-94
%
C=zeros(7,7);
rset=[1:5];
```

```
for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end
% Define C matrix.
for ii=[5]
 C(ii,ii)=c(kkk);
 C(ii,ii+1)=-c(kkk);
 C(ii+1,ii)=-c(kkk);
 C(ii+1,ii+1)=c(kkk);
end
C(7,7)=c(kkk);

Mr=Phi(:,rset).'*M*Phi(:,rset);
Kr=Phi(:,rset).'*K*Phi(:,rset);
Cr=Phi(:,rset).'*C*Phi(:,rset);

Ar=[-Cr -Kr;eye(5) zeros(5,5)];
[omnr,zetar]=damp(Ar);
[omnr,jsortr]=sort(omnr);
zetar=zetar(jsortr);
zeta2(:,kkk)=zetar;
omnr2(:,kkk)=omnr;
i5=1;
 for ii=1:10
  if omnr(ii) < min(lambda5rt)
    omnr(ii)=0;
  end
  if ii~=10
   if abs(omnr(ii)-omnr(ii+1))<.0001*min(lambda5rt)
    omnr(ii)=0;
   end % if
  end % if
  if abs(zetar(ii)-1.0)<.001
   %omnr(ii)=0;
  end % if
  if omnr(ii)~=0
   omn1(i5,kkk)=omnr(ii);
   zeta1(i5,kkk)=zetar(ii);
   i5=i5+1;
   if i5>6
    disp('i5 problem: pause')
    pause
   end % if
  end % if
 end % for ii
end % kkk
```

compsph.m

```
% compsph.m
% Matlab program to take compliant model parameters
% and use the spherical model to redistribute
% damping among the modes.
%
%  Thomas J. Thompson
%
% 7-22-94
%

% Get ksi(ii), summation of ks(ii,jj)

for ii=1:2
 ksi(ii)=sum(ks(ii,1:5));
end % for ii
```

```
% Get alphai(ii), summation of alpha(ii,jj)

for ii=1:2
 alphai(ii)=sum(alpha(ii,1:5));
end % for ii

for ii=1:2
 kbi(ii)=alphai(ii)*ki(ii)*ksi(ii)/(alphai(ii)*ki(ii)-ksi(ii));
 %hksi(ii)=alphai(ii)*ki(ii)*kbi(ii)/(alphai(ii)*ki(ii)/2+kbi(ii))/2;
 hksi(ii)=ksi(ii)/2;
end % for ii

% Get squares of direction cosines cos2th(ii,jj)

for ii=1:2
 cos2th(ii,:)=alpha(ii,1:5)/alphai(ii);
end % for ii

% Get gamma(ii,jj)

for ii=1:2
 gamma(ii,:)=sqrt(cos2th(ii,:));
end % for ii

% Get the naught stiffness matrix

nKn=diag(kA);
nK=nKn;
for ii=1:2
 nK=nK+hksi(ii)*gamma(ii,:).'*gamma(ii,:);
end % for ii

% Get roots of spherical model

compsphr

% Perform normal mode analysis on the spherical
% system.

[V, E] = eig(nK);
[e, jvec] = sort(diag(E));
nLambda=diag(e);
nPhi=V(:, jvec);

sphchk=norm(nPhi.'*nPhi - eye(5));
spheigchk=norm(nPhi.'*nK*nPhi-nLambda);
normsph=norm([sphchk spheigchk]);
if normsph > .0001
  disp(['Norm of spherical normal mode analysis is ' num2str(normsph)]);
  pause;
end % if

% Get nks(ii,jj)

for jj=1:5
 for ii=1:2
  nks(ii,jj)=((gamma(ii,:)*nPhi(:,jj))^2)*ksi(ii);
  hnks(ii,jj)=((gamma(ii,:)*nPhi(:,jj))^2)*hksi(ii);
 end % for ii
end % for jj

% Get nkA(jj)

for jj=1:5
 nkA(jj)=0;
 for kk=1:5
  nkA(jj)=nkA(jj)+(nPhi(kk,jj)^2)*kA(kk);
 end % for kk
end % for jj
```

```
% Do check on nkA and nks

nkchk=norm(nkA+sum(hnks)-diag(nLambda).');
if nkchk > .0001
 disp('Naught matrices fail check on frequency shift.');
 pause;
end % if

nomega=nkA+sum(nks);

% Get nkappas(ii,jj)

for jj=1:5
 nkappas(:,jj)=nks(:,jj)/nkA(jj);
end % for jj

% Get nalpha(ii,jj)

for ii=1:2
 nalpha(ii,:)=alphai(ii)*nks(ii,:)/ksi(ii);
end % for ii

% Set values of c

for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end

% Get values of nr(ii,jj)

 a=1.01; % Target mixing ratio (not used for dissertation).
 b=a-1;

 for jj=1:5
  for ii=1:2
   nr(ii,jj)=alpha(ii,jj)*c(kkk)*lambda5rt(jj)/ks(ii,jj);
   nnr(ii,jj)=alpha(ii,jj)*c(kkk)*sqrt(nkA(jj))/ks(ii,jj);
   r1=((alpha(ii,jj)*c(kkk)*lambda5rt(jj)/ks(ii,jj)));
   mix1=(b+(1-a)*r1)/(b+r1);
   mix2=a*r1/(b+r1);
   kso(ii,jj)=mix1*ks(ii,jj)+mix2*nks(ii,jj);
   alphao(ii,jj)=mix1*alpha(ii,jj)+mix2*nalpha(ii,jj);
   Lambdao(jj)=mix1*lambda5(jj)+mix2*nLambda(jj,jj);
   %ro(ii,jj)=alphao(ii,jj)*c(kkk)*sqrt(Lambdao(jj))/kso(ii,jj);
   %ro(ii,jj)=mix1*nr(ii,jj)+mix2*nnr(ii,jj);
   %ro(ii,jj)=(nr(ii,jj)^2+nnr(ii,jj))/(nr(ii,jj)+1);
  end % for ii
  kAo(jj)=mix1*kA(jj)+mix2*nkA(jj);
 end % for jj

% Get values of kappao(ii,jj)

 for jj=1:5
  kappaso(:,jj)=kso(:,jj)/kAo(jj);
 end % for jj

% Get values of neta(jj,kkk)

 for jj=1:5
 num=0;
 den=1;
  for ii=1:2
   r2=nr(ii,jj)^2;
   num=num+nkappas(ii,jj)*(nr(ii,jj)/(1+r2));
   den=den+nkappas(ii,jj)*r2/(1+r2);
  end % ii
  neta(jj,kkk)=num/den;
 end % jj
```

```
end % kkk
```

compsphr.m

```
% compsphr.m
% Matlab program to take the spherical model parameters and
% get the modal damping coefficients.
%
%   Thomas J. Thompson
%
% 8-1-94

Ks=diag(ksi);

Asph=[zeros(5,5) -nKn-gamma.'*Ks*gamma gamma.'*Ks;
      eye(5) zeros(5,5) zeros(5,2);
      zeros(2,5) zeros(2,5) zeros(2,2)];

alphainm=inv(diag(alphai));

% Set values of c(kkk)

for kkk=1:nsteps+1
 if kkk==1
   c(kkk)=cmin;
  else
   c(kkk)=c(kkk-1)*base;
 end

 Cdks=(alphainm/c(kkk))*Ks;

 Asph(11:12,6:10)=Cdks*gamma;
 Asph(11:12,11:12)=-Cdks;

 [omsph,zetasph]=damp(Asph);
 [omsph,jsortc]=sort(omsph);
 zetasph=zetasph(jsortc);
 zetas(:,kkk)=zeros(5,1);
 i5s=1;
 for ii=1:11
  if abs(zetasph(ii)-1.)>.001
   if zetasph(ii)==zetasph(ii+1)
    zetas(i5s,kkk)=zetasph(ii);
    i5s=i5s+1;
   end
  end
 end
end
yes=0;
disp('Enter 1 to show damping for the spherical model roots.');
 yes=input('');
if yes==1
 s1='o';
 s2='x';
 s3='+';
 s4='*';
 s5='.';
 p1='-';
 p2='--';
 p3='-.';
 p4=':';
 p5='-.';
 c1='y';
 c2='m';
 c3='c';
 c4='r';
 c5='g';
 c6='b';
```

```
 set(gcf,'DefaultTextFontName','Times');
 set(gcf,'DefaultTextFontSize',8);
 set(gcf,'DefaultAxesFontName','Times');
 semilogx(c,2*zeta(1,:),[c1 s1], ...
  c,2*zeta(2,:),[c2 s2], ...
c,2*zeta(3,:),[c3 s3], ...
c,2*zeta(4,:),[c4 s4], ...
c,2*zeta(5,:),[c5 s5], ...
c,2*zetas(1,:),[c1 p1], ...
c,2*zetas(2,:),[c2 p2], ...
c,2*zetas(3,:),[c3 p3], ...
c,2*zetas(4,:),[c4 p4], ...
c,2*zetas(5,:),[c5 p5] )
 legend('Mode 1','Mode 2','Mode 3','Mode 4','Mode 5',...
      'Spherical Model Root 1',...
      'Spherical Model Root 2',...
      'Spherical Model Root 3',...
      'Spherical Model Root 4',...
      'Spherical Model Root 5')
 axis([.01 10 0 .2])
 xlabel('Damping Coefficient, force*time/length')
 ylabel('2 x Modal Damping Ratio')
 grid on;
 yes=0;
 disp('Enter 1 to save to a postscript file');
  yes=input('');
 if yes==1
  eval(['print sphrk' num2str(k(4,6))]);
 end % if
 set(gcf,'DefaultTextFontSize',12);
 set(gcf,'DefaultTextFontName','Helvetica');
 set(gcf,'DefaultAxesFontName','Helvetica');
end
```

APPENDIX D. COMPUTER PROGRAMS USED IN OPTIMIZING STRUCTURAL DAMPING USING THE COMPLIANT MODEL

The shell script which calls NASTRAN to characterize the compliant model of the SPICE bulkhead is given below.

akk1sh:

```
#! /bin/tcsh
#  This is akk1sh.
#  It is patterned roughly after akk2sh.
#  It does a nastran normal modes analysis on the structure
#  with k=1 springs in the prospective damper locations.
#  It then does an separate eigenvalue analysis with a
#  finite stiffness in each of the locations.
#  Puts values of original frequencies and shifted frequencies
#  in a file called /akk1/freq.data.  The values of modal
#  forces (displacements) are saved to a file called
#  /akk1/disp.data.
#
set datevar=`date`
set mach=`hostname`
set day=$datevar[2]$datevar[3]
set homedir=`pwd`
set logfile=$homedir/akk1/akk1$day.log
set scrdir='/usr/indy2/tmp'
set outdir="$homedir"/narun/ak
set barlistfile=`cat bar/barlistfile`
rm -f $logfile
echo "Log file for $homedir/akk1sh run \
on $mach at $datevar. \
List of damper locations from bar/$barlistfile. ">$logfile
echo 'Make sure you have run bar/make12k1sh to make cbar files.'
#
cp /dev/null narun/cba.dat
cp /dev/null narun/ese.dat
cp /dev/null akk1/freq.data
cp /dev/null akk1/disp.data
echo "Frequencies gotten from $homedir/akk1sh \
at $datevar. \
List of damper locations from bar/$barlistfile." >> akk1/freq.data
echo "Displacements gotten from $homedir/akk1sh \
at $datevar. \
List of damper locations from bar/$barlistfile." >> akk1/disp.data

echo 'set 4 = 30000 thru 40000' >> narun/ese.dat
echo 'force(punch) = 4' >> narun/ese.dat

time nastran narun/ak.dat bat="no" scr=yes sdi=$scrdir out=$outdir old=no \
notify=no >> $logfile
```

```
set tmp=$scrdir/tom
rm -f $tmp
grep '1.000000E+00' "$outdir".f06 > $tmp
grep -v 'EL' $tmp >> $homedir/akk1/freq.data
egrep '(real)|(user)|(sys)' $outdir.log >> $logfile
rm -f $tmp
rm -f $homedir/akk1/dispall.data
cp /dev/null $homedir/akk1/conrods.data
echo "List of conrods made from akk1sh on $day. \
Should be based on bar list file $barlistfile." >> $homedir/akk1/conrods.data
grep '^     3' narun/ak.pch > $homedir/akk1/dispall.data
set conrods=`awk -F, '{print $2}' $homedir/narun/cbk1.dat`
foreach conrod ( $conrods )
  echo $conrod >>$homedir/akk1/conrods.data
  grep " $conrod " $homedir/akk1/dispall.data >>$homedir/akk1/disp.data
end

set realmin=0
set usermin=0
set sysmin=0
set realsec=0
set usersec=0
set syssec=0

foreach file (bar/cbar[0-9]*)
 cp x2{$file}x2 narun/cba.dat

 time nastran narun/ak.dat bat="no" scr=yes sdi=$scrdir out=$outdir old=no \
 notify=no >> $logfile

 rm -f $tmp "$tmp"2
 more narun/cba.dat >$tmp
 grep '^cb' $tmp > "$tmp"2
 set n1=`awk -F, '{print $4}' "$tmp"2`
 switch ("$n1")
      case ?
              set n1=0$n1
      breaksw
 endsw
 set n2=`awk -F, '{print $5}' "$tmp"2`
 switch ("$n2")
      case ?
              set n2=0$n2
      breaksw
 endsw
 echo "3$n1$n2">> $homedir/akk1/freq.data
 echo "3$n1$n2">> $logfile
 rm -f $tmp
 grep '1.000000E+00' narun/ak.f06 > $tmp
 grep -v 'EL' $tmp  >> $homedir/akk1/freq.data
 egrep '(real)|(user)|(sys)' $outdir.log >> $logfile
 set real=`egrep 'real' $outdir.log | awk '{print $2}'`

 set user=`egrep 'user' $outdir.log | awk '{print $2}'`

 set sys=`egrep 'sys' $outdir.log | awk '{print $2}'`

end
echo 'Done looping in akk1sh.' >>$logfile
rm -f $tmp
```

The NASTRAN file which ran the necessary real eigenvalue analyses is given below, followed by some of the NASTRAN data files which were "included." The file *model.dat* containing the SPICE bulkhead model is not listed.

ak.dat:

```
$id  ak
$ Thomas Thompson   10-21-93
sol 103  $ real
time 720
diag 8
cend
$
title = Alpha and Kappa Computation
echo=none
method=1  $ real
set 3 = 3,26,28,48,51,57
include 'narun/ese.dat'
disp(punch) = 3
SPC = 1
$
begin bulk
$param,tiny,1.e-20
param,tiny,0.0
include 'narun/eig.dat'
include 'narun/model.dat'
include 'narun/cba.dat'
include 'narun/cbk2.dat'
mat1,20,1.1e+11,2.65e+9,,0.  $out with model5a model
pbar,11,20,8.045e-5,5.00e-8,5.0e-8,1.e-7  $need to ch for modelfull
pbar,211,20,1.609e-4,5.00e-8,5.0e-8,1.e-7
$pbar,10002,20,1.875-11  $ for k=2, l=1.03
$conrod,32834,28,34,20,1.875-11
$celas2,32834,2.0,28,6,34,6
enddata
```

eig.dat:

```
$ Copied from /local/usr2/thomps/iris/narun on 4-29-95.  Some comments deleted.
eigrl,1, 10.,170.
```

Typical *cba.dat*:

```
$  cbar 306
$  Stiffness 2
$  Cell  0
cbar,20306,211, 56, 62,99
,56,56
```

cbk2.dat:

```
conrod,31524,15,24,20,9.375-12
conrod,32438,24,38,20,9.375-12
conrod,33851,38,51,20,9.375-12
conrod,32331,23,31,20,9.375-12
conrod,33145,31,45,20,9.375-12
conrod,30618, 6,18,20,9.375-12
conrod,31832,18,32,20,9.375-12
conrod,33246,32,46,20,9.375-12
```

```
conrod,34658,46,58,20,9.375-12
conrod,30925, 9,25,20,9.375-12
conrod,32539,25,39,20,9.375-12
conrod,33952,39,52,20,9.375-12
conrod,30212, 2,12,20,9.375-12
conrod,31226,12,26,20,9.375-12
conrod,32640,26,40,20,9.375-12
conrod,34053,40,53,20,9.375-12
conrod,35363,53,63,20,9.375-12
conrod,30719, 7,19,20,9.375-12
conrod,31933,19,33,20,9.375-12
conrod,33347,33,47,20,9.375-12
conrod,34759,47,59,20,9.375-12
conrod,30308, 3, 8,20,9.375-12
conrod,30820, 8,20,20,9.375-12
conrod,32034,20,34,20,9.375-12
conrod,33448,34,48,20,9.375-12
conrod,34860,48,60,20,9.375-12
conrod,36067,60,67,20,9.375-12
conrod,30413, 4,13,20,9.375-12
conrod,31327,13,27,20,9.375-12
conrod,32741,27,41,20,9.375-12
conrod,34154,41,54,20,9.375-12
conrod,35464,54,64,20,9.375-12
conrod,30514, 5,14,20,9.375-12
conrod,31428,14,28,20,9.375-12
conrod,32842,28,42,20,9.375-12
conrod,34255,42,55,20,9.375-12
conrod,35565,55,65,20,9.375-12
conrod,31021,10,21,20,9.375-12
conrod,32135,21,35,20,9.375-12
conrod,33549,35,49,20,9.375-12
conrod,34961,49,61,20,9.375-12
conrod,31122,11,22,20,9.375-12
conrod,32236,22,36,20,9.375-12
conrod,33650,36,50,20,9.375-12
conrod,35062,50,62,20,9.375-12
conrod,31629,16,29,20,9.375-12
conrod,32943,29,43,20,9.375-12
conrod,34356,43,56,20,9.375-12
conrod,31730,17,30,20,9.375-12
conrod,33044,30,44,20,9.375-12
conrod,34457,44,57,20,9.375-12
conrod,34558,45,58,20,1.326-11
conrod,33146,31,46,20,1.326-11
conrod,35263,52,63,20,1.326-11
conrod,32332,23,32,20,1.326-11
conrod,33953,39,53,20,1.326-11
conrod,35967,59,67,20,1.326-11
conrod,32540,25,40,20,1.326-11
conrod,34760,47,60,20,1.326-11
conrod,30926, 9,26,20,1.326-11
conrod,33348,33,48,20,1.326-11
conrod,35465,54,65,20,1.326-11
conrod,31934,19,34,20,1.326-11
conrod,34155,41,55,20,1.326-11
conrod,30720, 7,20,20,1.326-11
conrod,32742,27,42,20,1.326-11
conrod,34962,49,62,20,1.326-11
conrod,31328,13,28,20,1.326-11
conrod,33550,35,50,20,1.326-11
conrod,30414, 4,14,20,1.326-11
conrod,32136,21,36,20,1.326-11
conrod,34357,43,57,20,1.326-11
conrod,31022,10,22,20,1.326-11
conrod,32944,29,44,20,1.326-11
conrod,31630,16,30,20,1.326-11
conrod,33845,38,45,20,9.375-12
conrod,32431,24,31,20,9.375-12
conrod,34652,46,52,20,9.375-12
conrod,31523,15,23,20,9.375-12
```

```
conrod,33239,32,39,20,9.375-12
conrod,35359,53,59,20,9.375-12
conrod,31825,18,25,20,9.375-12
conrod,34047,40,47,20,9.375-12
conrod,36064,60,64,20,9.375-12
conrod,30609, 6, 9,20,9.375-12
conrod,32633,26,33,20,9.375-12
conrod,34854,48,54,20,9.375-12
conrod,31219,12,19,20,9.375-12
conrod,33441,34,41,20,9.375-12
conrod,35561,55,61,20,9.375-12
conrod,30207, 2, 7,20,9.375-12
conrod,32027,20,27,20,9.375-12
conrod,34249,42,49,20,9.375-12
conrod,30813, 8,13,20,9.375-12
conrod,32835,28,35,20,9.375-12
conrod,35056,50,56,20,9.375-12
conrod,30304, 3, 4,20,9.375-12
conrod,31421,14,21,20,9.375-12
conrod,33643,36,43,20,9.375-12
conrod,30510, 5,10,20,9.375-12
conrod,32229,22,29,20,9.375-12
conrod,31116,11,16,20,9.375-12
conrod,35158,51,58,20,9.375-12
conrod,35863,58,63,20,9.375-12
conrod,36367,63,67,20,9.375-12
conrod,34552,45,52,20,9.375-12
conrod,35259,52,59,20,9.375-12
conrod,35964,59,64,20,9.375-12
conrod,33846,38,46,20,9.375-12
conrod,34653,46,53,20,9.375-12
conrod,35360,53,60,20,9.375-12
conrod,36065,60,65,20,9.375-12
conrod,33139,31,39,20,9.375-12
conrod,33947,39,47,20,9.375-12
conrod,34754,47,54,20,9.375-12
conrod,35461,54,61,20,9.375-12
conrod,32432,24,32,20,9.375-12
conrod,33240,32,40,20,9.375-12
conrod,34048,40,48,20,9.375-12
conrod,34855,48,55,20,9.375-12
conrod,35562,55,62,20,9.375-12
conrod,32325,23,25,20,9.375-12
conrod,32533,25,33,20,9.375-12
conrod,33341,33,41,20,9.375-12
conrod,34149,41,49,20,9.375-12
conrod,34956,49,56,20,9.375-12
conrod,31518,15,18,20,9.375-12
conrod,31826,18,26,20,9.375-12
conrod,32634,26,34,20,9.375-12
conrod,33442,34,42,20,9.375-12
conrod,34250,42,50,20,9.375-12
conrod,35057,50,57,20,9.375-12
conrod,30919, 9,19,20,9.375-12
conrod,31927,19,27,20,9.375-12
conrod,32735,27,35,20,9.375-12
conrod,33543,35,43,20,9.375-12
conrod,30612, 6,12,20,9.375-12
conrod,31220,12,20,20,9.375-12
conrod,32028,20,28,20,9.375-12
conrod,32836,28,36,20,9.375-12
conrod,33644,36,44,20,9.375-12
conrod,30713, 7,13,20,9.375-12
conrod,31321,13,21,20,9.375-12
conrod,32129,21,29,20,9.375-12
conrod,30208, 2, 8,20,9.375-12
conrod,30814, 8,14,20,9.375-12
conrod,31422,14,22,20,9.375-12
conrod,32230,22,30,20,9.375-12
conrod,30410, 4,10,20,9.375-12
conrod,31016,10,16,20,9.375-12
```

```
conrod,30305, 3, 5,20,9.375-12
conrod,30511, 5,11,20,9.375-12
conrod,31117,11,17,20,9.375-12
conrod,36364,63,64,20,1.326-11
conrod,35859,58,59,20,1.326-11
conrod,36061,60,61,20,1.326-11
conrod,35152,51,52,20,1.326-11
conrod,35354,53,54,20,1.326-11
conrod,35556,55,56,20,1.326-11
conrod,34647,46,47,20,1.326-11
conrod,34849,48,49,20,1.326-11
conrod,33839,38,39,20,1.326-11
conrod,34041,40,41,20,1.326-11
conrod,34243,42,43,20,1.326-11
conrod,33233,32,33,20,1.326-11
conrod,33435,34,35,20,1.326-11
conrod,32425,24,25,20,1.326-11
conrod,32627,26,27,20,1.326-11
conrod,32829,28,29,20,1.326-11
conrod,31819,18,19,20,1.326-11
conrod,32021,20,21,20,1.326-11
conrod,31509,15, 9,20,1.326-11
conrod,31213,12,13,20,1.326-11
conrod,31416,14,16,20,1.326-11
conrod,30607, 6, 7,20,1.326-11
conrod,30810, 8,10,20,1.326-11
conrod,30204, 2, 4,20,1.326-11
conrod,36465,64,65,20,9.375-12
conrod,35960,59,60,20,9.375-12
conrod,36162,61,62,20,9.375-12
conrod,35253,52,53,20,9.375-12
conrod,35455,54,55,20,9.375-12
conrod,35657,56,57,20,9.375-12
conrod,34546,45,46,20,9.375-12
conrod,34748,47,48,20,9.375-12
conrod,34950,49,50,20,9.375-12
conrod,33940,39,40,20,9.375-12
conrod,34142,41,42,20,9.375-12
conrod,34344,43,44,20,9.375-12
conrod,33132,31,32,20,9.375-12
conrod,33334,33,34,20,9.375-12
conrod,33536,35,36,20,9.375-12
conrod,32526,25,26,20,9.375-12
conrod,32728,27,28,20,9.375-12
conrod,32930,29,30,20,9.375-12
conrod,32318,23,18,20,9.375-12
conrod,31920,19,20,20,9.375-12
conrod,32122,21,22,20,9.375-12
conrod,30912, 9,12,20,9.375-12
conrod,31314,13,14,20,9.375-12
conrod,31617,16,17,20,9.375-12
conrod,30708, 7, 8,20,9.375-12
conrod,31011,10,11,20,9.375-12
conrod,30405, 4, 5,20,9.375-12
conrod,30302, 3, 2,20,9.375-12
conrod,30206, 2, 6,20,9.375-12
conrod,30615, 6,15,20,9.375-12
conrod,30407, 4, 7,20,9.375-12
conrod,30709, 7, 9,20,9.375-12
conrod,30923, 9,23,20,9.375-12
conrod,30508, 5, 8,20,9.375-12
conrod,30812, 8,12,20,9.375-12
conrod,31218,12,18,20,9.375-12
conrod,31824,18,24,20,9.375-12
conrod,31013,10,13,20,9.375-12
conrod,31319,13,19,20,9.375-12
conrod,31925,19,25,20,9.375-12
conrod,32531,25,31,20,9.375-12
conrod,31114,11,14,20,9.375-12
conrod,31420,14,20,20,9.375-12
conrod,32026,20,26,20,9.375-12
```

```
conrod,32632,26,32,20,9.375-12
conrod,33238,32,38,20,9.375-12
conrod,31621,16,21,20,9.375-12
conrod,32127,21,27,20,9.375-12
conrod,32733,27,33,20,9.375-12
conrod,33339,33,39,20,9.375-12
conrod,33945,39,45,20,9.375-12
conrod,31722,17,22,20,9.375-12
conrod,32228,22,28,20,9.375-12
conrod,32834,28,34,20,9.375-12
conrod,33440,34,40,20,9.375-12
conrod,34046,40,46,20,9.375-12
conrod,34651,46,51,20,9.375-12
conrod,32935,29,35,20,9.375-12
conrod,33541,35,41,20,9.375-12
conrod,34147,41,47,20,9.375-12
conrod,34752,47,52,20,9.375-12
conrod,33036,30,36,20,9.375-12
conrod,33642,36,42,20,9.375-12
conrod,34248,42,48,20,9.375-12
conrod,34853,48,53,20,9.375-12
conrod,35358,53,58,20,9.375-12
conrod,34349,43,49,20,9.375-12
conrod,34954,49,54,20,9.375-12
conrod,35459,54,59,20,9.375-12
conrod,34450,44,50,20,9.375-12
conrod,35055,50,55,20,9.375-12
conrod,35560,55,60,20,9.375-12
conrod,36063,60,63,20,9.375-12
conrod,35661,56,61,20,9.375-12
conrod,36164,61,64,20,9.375-12
conrod,35762,57,62,20,9.375-12
conrod,36265,62,65,20,9.375-12
conrod,36567,65,67,20,9.375-12
conrod,30623, 6,23,20,1.326-11
conrod,30209, 2, 9,20,1.326-11
conrod,31831,18,31,20,1.326-11
conrod,30307, 3, 7,20,1.326-11
conrod,31225,12,25,20,1.326-11
conrod,33245,32,45,20,1.326-11
conrod,30819, 8,19,20,1.326-11
conrod,32639,26,39,20,1.326-11
conrod,30513, 5,13,20,1.326-11
conrod,32033,20,33,20,1.326-11
conrod,34052,40,52,20,1.326-11
conrod,31427,14,27,20,1.326-11
conrod,33447,34,47,20,1.326-11
conrod,31121,11,21,20,1.326-11
conrod,32841,28,41,20,1.326-11
conrod,34859,48,59,20,1.326-11
conrod,32235,22,35,20,1.326-11
conrod,34254,42,54,20,1.326-11
conrod,31729,17,29,20,1.326-11
conrod,33649,36,49,20,1.326-11
conrod,35564,55,64,20,1.326-11
conrod,33043,30,43,20,1.326-11
conrod,35061,50,61,20,1.326-11
conrod,34456,44,56,20,1.326-11
conrod,32324,23,24,20,9.375-12
conrod,30918, 9,18,20,9.375-12
conrod,33138,31,38,20,9.375-12
conrod,30712, 7,12,20,9.375-12
conrod,32532,25,32,20,9.375-12
conrod,34551,45,51,20,9.375-12
conrod,30408, 4, 8,20,9.375-12
conrod,31926,19,26,20,9.375-12
conrod,33946,39,46,20,9.375-12
conrod,31320,13,20,20,9.375-12
conrod,33340,33,40,20,9.375-12
conrod,35258,52,58,20,9.375-12
conrod,31014,10,14,20,9.375-12
```

```
conrod,32734,27,34,20,9.375-12
conrod,34753,47,53,20,9.375-12
conrod,32128,21,28,20,9.375-12
conrod,34148,41,48,20,9.375-12
conrod,35963,59,63,20,9.375-12
conrod,31622,16,22,20,9.375-12
conrod,33542,35,42,20,9.375-12
conrod,35460,54,60,20,9.375-12
conrod,32936,29,36,20,9.375-12
conrod,34955,49,55,20,9.375-12
conrod,36467,64,67,20,9.375-12
conrod,34350,43,50,20,9.375-12
conrod,36165,61,65,20,9.375-12
conrod,35662,56,62,20,9.375-12
```

The script file used to prepare *cba.dat* and *cbk2.dat* files are given below.

make12k1sh:

```
#/! /bin/tcsh
#  Run from vislab.me.iastate.edu (Vincent station) 8-24-94.
rm -f barlistfile
echo $1 > barlistfile
rm -f cbar[0-9]*
make_inputk1 $1
cp ../narun/cbk1.dat ../narun/cbk2.dat
cd /local/usr2/thomps/iris/x2bar/
rm -f cbar[0-9]*x2
make_inputx2 /local/usr2/thomps/iris/bar/$1
cd /local/usr2/thomps/iris/bar
#
```

make_inputk1:

```
#!/usr/local/bin/perl
# chmod u+rx make_input
# issue the command 'man perl' to learn more about this language
# Taken from a file written by Charles Randall.

$infile = $ARGV[0];
if ( $infile eq "" ) {
  die "$0: you must specify an input file";
}
open (OUTPUT2,'>../narun/cbk1.dat') || die "$0: could not open output file ../narun/cbk1.dat";
   print "Working on file ../narun/cbk1.dat\n";

open (INPUT, "<$infile") || die "$0: could not open input file $infile";

while ( $line=<INPUT> ) {
  chop $line;
  @dummy = split(' ',$line);
  if ( $#dummy != 3 ) {
    die "$0: error, line $. of file $infile. Line does not have 4 numbers.";
  }
  ($n,$x,$y,$z)=@dummy;
# open new output file
  $outfile = 'cbar'.$n;
  if ( $n < 100 ) {
     $outfile = 'cbar0'.$n;
  }{}
  if ( $n < 10 ) {
     $outfile = 'cbar00'.$n;
  }{}
  {if ($x < 10 ) {
$xx="0$x" }
     else { $xx = "$x" }}
```

```
  {if ($y < 10 ) {
$yy="0$y" }
     else { $yy="$y" }}

  open (OUTPUT,">$outfile") || die "$0: could not open output file $outfile";
  print "Working on file $outfile\n";
  printf (OUTPUT "%1s  %s %3d\n","\$",'cbar',$n);
  printf (OUTPUT "%1s %s %1d\n","\$",' Stiffness',1);
  printf (OUTPUT "%1s %s %2d\n","\$",' Cell',$z);
  printf (OUTPUT "%3s,%4d,%3d,%3d,%3d,%2d\n",'cbar',10000+$n,11,$x,$y,99);
  print OUTPUT ',56,56' . "\n";
  close (OUTPUT);
  if ( $z < 1 ) {
   printf (OUTPUT2 "%s,%2d,%2d,%s,%s\n",'conrod,3'.$xx.$yy,$x,$y,20,'9.375-12');}
    else {
   printf (OUTPUT2 "%s,%2d,%2d,%s,%s\n",'conrod,3'.$xx.$yy,$x,$y,20,'1.326-11');}

}

close (INPUT);
close (OUTPUT2);
```

File containing list of damper locations, node pairs, and types of locations (zero values in the fourth column denote in-line; non-zero values denote diagonal).

blistblk

```
1 15 24 0
2 24 38 0
3 38 51 0
4 23 31 0
5 31 45 0
6 6 18 0
7 18 32 0
8 32 46 0
9 46 58 0
10 9 25 0
11 25 39 0
12 39 52 0
13 2 12 0
14 12 26 0
15 26 40 0
16 40 53 0
17 53 63 0
18 7 19 0
19 19 33 0
20 33 47 0
21 47 59 0
22 3 8 0
23 8 20 0
24 20 34 0
25 34 48 0
26 48 60 0
27 60 67 0
28 4 13 0
29 13 27 0
30 27 41 0
31 41 54 0
32 54 64 0
33 5 14 0
34 14 28 0
35 28 42 0
36 42 55 0
37 55 65 0
```

38 10 21 0
39 21 35 0
40 35 49 0
41 49 61 0
42 11 22 0
43 22 36 0
44 36 50 0
45 50 62 0
46 16 29 0
47 29 43 0
48 43 56 0
49 17 30 0
50 30 44 0
51 44 57 0
52 45 58 22
53 31 46 16
54 52 63 25
55 23 32 10
56 39 53 19
57 59 67 27
58 25 40 13
59 47 60 23
60 9 26 7
61 33 48 17
62 54 65 26
63 19 34 11
64 41 55 20
65 7 20 5
66 27 42 14
67 49 62 24
68 13 28 8
69 35 50 18
70 4 14 3
71 21 36 12
72 43 57 21
73 10 22 6
74 29 44 15
75 16 30 9
76 38 45 0
77 24 31 0
78 46 52 0
79 15 23 0
80 32 39 0
81 53 59 0
82 18 25 0
83 40 47 0
84 60 64 0
85 6 9 0
86 26 33 0
87 48 54 0
88 12 19 0
89 34 41 0
90 55 61 0
91 2 7 0
92 20 27 0
93 42 49 0
94 8 13 0
95 28 35 0
96 50 56 0
97 3 4 0
98 14 21 0
99 36 43 0
100 5 10 0
101 22 29 0
102 11 16 0
103 51 58 0
104 58 63 0
105 63 67 0
106 45 52 0
107 52 59 0
108 59 64 0

```
109 38 46 0
110 46 53 0
111 53 60 0
112 60 65 0
113 31 39 0
114 39 47 0
115 47 54 0
116 54 61 0
117 24 32 0
118 32 40 0
119 40 48 0
120 48 55 0
121 55 62 0
122 23 25 0
123 25 33 0
124 33 41 0
125 41 49 0
126 49 56 0
127 15 18 0
128 18 26 0
129 26 34 0
130 34 42 0
131 42 50 0
132 50 57 0
133 9 19 0
134 19 27 0
135 27 35 0
136 35 43 0
137 6 12 0
138 12 20 0
139 20 28 0
140 28 36 0
141 36 44 0
142 7 13 0
143 13 21 0
144 21 29 0
145 2 8 0
146 8 14 0
147 14 22 0
148 22 30 0
149 4 10 0
150 10 16 0
151 3 5 0
152 5 11 0
153 11 17 0
154 63 64 27
155 58 59 25
156 60 61 26
157 51 52 22
158 53 54 23
159 55 56 24
160 46 47 19
161 48 49 20
162 38 39 16
163 40 41 17
164 42 43 18
165 32 33 13
166 34 35 14
167 24 25 10
168 26 27 11
169 28 29 12
170 18 19 7
171 20 21 8
172 15 9 4
173 12 13 5
174 14 16 6
175 6 7 2
176 8 10 3
177 2 4 1
178 64 65 0
179 59 60 0
```

180 61 62 0
181 52 53 0
182 54 55 0
183 56 57 0
184 45 46 0
185 47 48 0
186 49 50 0
187 39 40 0
188 41 42 0
189 43 44 0
190 31 32 0
191 33 34 0
192 35 36 0
193 25 26 0
194 27 28 0
195 29 30 0
196 23 18 0
197 19 20 0
198 21 22 0
199 9 12 0
200 13 14 0
201 16 17 0
202 7 8 0
203 10 11 0
204 4 5 0
205 3 2 0
206 2 6 0
207 6 15 0
208 4 7 0
209 7 9 0
210 9 23 0
211 5 8 0
212 8 12 0
213 12 18 0
214 18 24 0
215 10 13 0
216 13 19 0
217 19 25 0
218 25 31 0
219 11 14 0
220 14 20 0
221 20 26 0
222 26 32 0
223 32 38 0
224 16 21 0
225 21 27 0
226 27 33 0
227 33 39 0
228 39 45 0
229 17 22 0
230 22 28 0
231 28 34 0
232 34 40 0
233 40 46 0
234 46 51 0
235 29 35 0
236 35 41 0
237 41 47 0
238 47 52 0
239 30 36 0
240 36 42 0
241 42 48 0
242 48 53 0
243 53 58 0
244 43 49 0
245 49 54 0
246 54 59 0
247 44 50 0
248 50 55 0
249 55 60 0
250 60 63 0

```
251 56 61 0
252 61 64 0
253 57 62 0
254 62 65 0
255 65 67 0
256 6 23 4
257 2 9 2
258 18 31 10
259 3 7 1
260 12 25 7
261 32 45 16
262 8 19 5
263 26 39 13
264 5 13 3
265 20 33 11
266 40 52 19
267 14 27 8
268 34 47 17
269 11 21 6
270 28 41 14
271 48 59 23
272 22 35 12
273 42 54 20
274 17 29 9
275 36 49 18
276 55 64 26
277 30 43 15
278 50 61 24
279 44 56 21
280 23 24 0
281 9 18 0
282 31 38 0
283 7 12 0
284 25 32 0
285 45 51 0
286 4 8 0
287 19 26 0
288 39 46 0
289 13 20 0
290 33 40 0
291 52 58 0
292 10 14 0
293 27 34 0
294 47 53 0
295 21 28 0
296 41 48 0
297 59 63 0
298 16 22 0
299 35 42 0
300 54 60 0
301 29 36 0
302 49 55 0
303 64 67 0
304 43 50 0
305 61 65 0
306 56 62 0
```

Matlab files used to perform the simulated annealing procedure.

bonk.m:

```
% bonk.m
% Matlab scipt to input information on modal
% frequencies and displacements found by
% nastran in shell akk1sh and use the compliant
% model to rank the struts and bound the number
% of struts needed to obtain the minimum damping
% requirements.
```

```
%
%  Thomas J. Thompson
%
% 8-26-94
%
clear
time=clock;
hr=num2str(time(4));
mne=num2str(time(5));
tic;
cpu_time=cputime;
!rm -f /tmp/freqf.data /tmp/dispf.data /tmp/conf.data /tmp/temp
!grep '  [0-9]' freq.data > /tmp/freqf.data
load /tmp/freqf.data
!rm -f /tmp/conrods.data
!grep '^3' conrods.data >/tmp/conrods.data
load /tmp/conrods.data
!grep ' 3' disp.data > /tmp/dispf.data
load /tmp/dispf.data
!grep '^3' freq.data >/tmp/conf.data
load /tmp/conf.data
% Check order of prospective damping locations.
if norm(conf-conrods)>.01
  disp('Problem with order of damping locations!');
end

% Output results to a log file bonk?.log.

logfile=['bonk' date '.log'];
eval(['!rm -f ' logfile])
eval(['!echo ''This is the log file created by bonk.m (matlab script)'' > ' logfile]);
eval(['!echo ''on ' date ' at ' hr ':' mne ' (24hr).'' >> ' logfile])
message='The file beginning with the following header was used as input.'
eval(['!echo ' message '>> ' logfile]);
eval(['!echo '' '' >> ' logfile])
!rm -f /tmp/temp
!grep '^[a-zA-Z]' freq.data>/tmp/temp
eval(['!cat /tmp/temp >> ' logfile])
eval(['!echo '' '' >> ' logfile])

flops(0);
[n,dum]=size(conrods);
[mn,dum]=size(dispf);
m=mn/n;
for jj=1:m
 w(jj)=1;
 tau(jj)=.04;
end

omo2=freqf(1:m,3);
omo=sqrt(omo2);
for ii=1:n
 row=m*(ii-1);
 beta(ii,1:m)=dispf(row+1:row+m,2).';
 om2(ii,1:m)=freqf(row+m+1:row+2*m,3).';
end

for ii=1:n
 alphai(ii)=sum(beta(ii,1:m).^2);
 gamma(ii,1:m)=beta(ii,1:m)/sqrt(alphai(ii));
 alpha(ii,:)=beta(ii,1:m).^2;
end

kAj=omo2;
for jj=1:m
 ks(:,jj)=om2(:,jj)-kAj(jj);
end
ksi=sum(ks.');
for ii=1:n
 ks(ii,:)=gamma(ii,1:m).^2*ksi(ii);
end
```

```
for jj=1:m
 kappas(:,jj)=ks(:,jj)/kAj(jj);
end

for ii=1:n
 for jj=1:m
  relse(ii,jj)=omo(jj)*alpha(ii,jj)/(ks(ii,jj)+1.e-15);
 end
end

% Get summation strut ranking index Jtil

for ii=1:n
 c=25000;
 eps=1;
 it=0;
 while abs(eps) > 1e-12
  it=it+1;
  Jtil(ii)=0;
  Jtilc=0;
  Jtilcc=0;
  for jj=1:m
   r=c*relse(ii,jj);
   r2=r^2;
   Jtil(ii)=Jtil(ii)+w(jj)*kappas(ii,jj)*r/(1+r2);
   Jtilc=Jtilc+w(jj)*(alpha(ii,jj)/sqrt(omo2(jj)))*(1-r2)/((1+r2)^2);
   Jtilcc=Jtilcc+(w(jj)*(alpha(ii,jj)^2)*r*2/ks(ii,jj))*(r2-3)/((1+r2)^3);
  end % for jj
  next=c-Jtilc/Jtilcc;
  if next<c/4
    c=c/4;
   elseif Jtilcc>0
    c=rand*c;
   else
    c=next;
  end % if
 eps=Jtilc;
 end % while
 cstr(ii)=c;
 Jtilc;
end % for ii

% Rank struts according to the strut ranking index.

[sri,sriorder]=sort(1./Jtil);
sri=1./sri;

% Distribution ranking index

for ii=1:n
 a=0;
 b=100000;
 eta1=1;
 while eta1>.1
  c=a+0.39*(b-a);
  d=a+0.61*(b-a);
  rc=c*relse(ii,:);
  rd=d*relse(ii,:);
  etac=kappas(ii,:).*rc./(1+rc.*rc.*(1+kappas(ii,:)));
  etad=kappas(ii,:).*rd./(1+rd.*rd.*(1+kappas(ii,:)));
  denc=0;
  dend=0;
  for jj=1:m
   denc=denc+w(jj)/(1+etac(jj)/tau(jj));
   dend=dend+w(jj)/(1+etad(jj)/tau(jj));
  end
  JHic=sum(w)/denc - 1;
  JHid=sum(w)/dend - 1;
  eta1=abs(d-c);
  if JHic>JHid
    b=d;
```

```
   else
    a=c;
  end % if
 end % while
 JHi(ii)=JHic;
 cstrd(ii)=c;
end % for ii

% Rank struts according to the distribution index.

[dis,disorder]=sort(1./JHi);
dis=1./dis;

% Kappa ranking

for ii=1:n
 Jk(ii)=sum(kappas(ii,:).*w/2);
end

% Rank struts according to the kappa index.

[kap,kaporder]=sort(1./Jk);
kap=1./kap;

disp('Strut ranking complete.');

% Get lower bound on the number of struts needed.

%   Use summation index

Jnec=sum(w.*tau);
lbs=0;
J=0;
while J<Jnec&lbs<n
lbs=lbs+1;
 J=sum(sri(1:lbs));
end % while
sum(sri(1:lbs))

% Use kappa index

lbk=0;
J=0;
while J<Jnec&lbk<n
 lbk=lbk+1;
 J=sum(kap(1:lbk));
end % while
sum(kap(1:lbk))

% Lower bound is the higher of lbk and lbs.

[lb,lbmax]=max([lbk lbs]);
maxind=['kappa    '; 'summation'];

% Call ul.m to find the upper limit on the number of struts.

ul

eval(['diary ' logfile]);
tau
lbk
lbs
disp('The lower bound is');
disp(lb)
nuk
disp(['The upper bound of ' num2str(nu) ...
' was found using the ' rofmin(kmin,:) ' ranking index.']);
operations=flops
toc
cpu_time=cputime-cpu_time
diary off
```

```
% Write outputs to results file for dissertation.

resfil=fopen('bonkres.tex','w');
fprintf(resfil,'Summation & %3i & %3i\\\\ \n', lbk(1),nuk(1));
fprintf(resfil,'Distribution & & %3i\\\\ \n', nuk(2));
fprintf(resfil,'Kappa & %3i & %3i\\\\ \n', lbs,nuk(3));
fprintf(resfil,'\\hline \n Used & %3i & %3i\\\\ \n', lb,nu);

fclose(resfil);

load ../sa/blistblk;
[dum,disoo]=sort(disorder);
disrking=disoo(sriorder);
[dum,kapoo]=sort(kaporder);
kaprking=kapoo(sriorder);
brfil=fopen('bonkrank.tex','w');
for ii=1:72
 snum=sriorder(ii);
 n1=['00' num2str(blistblk(snum,2))];
 n2=['00' num2str(blistblk(snum,3))];
 n1=n1((size(n1,2)-1):size(n1,2)) ;
 n2=n2((size(n2,2)-1):size(n2,2));
  if blistblk(snum,4)>0
  ty='D'; else ty='I'; end

 fprintf(brfil,'%3i & %3i & %3i & %3i & %s--%s & %4.2E & %s\\\\ \n',...
  ii,disrking(ii),kaprking(ii),snum,n1,n2,Jtil(snum),ty);
end % for ii
fclose(brfil);
```

ul.m (called by *bonk.m*):

```
% Matlab file to find the upper limit on the number of
% damping struts needed to achieve damping specified by
% the saturation index.  Called by  bonk.m.
%
% Thomas J. Thompson
%
% 9-5-94
%
clear nuk rank ranks comb
% Rankings according to the summation ranking, distribution, and
% kappa indexes.

ranks=[sriorder.' disorder.' kaporder.']
indices=[Jtil(sriorder).' JHi(disorder).' Jk(kaporder).']
save ranks.data ranks -ascii
save indices.data indices -ascii
eval(['!cat ranks.data >> ' logfile ';']);
eval(['!cat indices.data >> ' logfile ';']);

if lb==n
 disp('Problem:  lower bound equals total number of struts.');
 pause
end
flopsold=flops;
Jsat=0;
% For each type of ranking.
for k=1:3
 rank=ranks(:,k);
 nuk(k)=lb;
 found=1;
 while found==1
  clc;
  [Jsat Jnec]
  nuk(k)=nuk(k)+1
  k
```

```
   comb=ranks(1:nuk(k),k).'
   %indices(1:nuk(k),k).'
   %pause
   opt;
   if nuk(k)==n
     found=0
     disp('All struts used and saturation index not reached!');
     pause;
    else
     found=Jsat<Jnec;
   end
  end % while
   plot(Jsatv);
   grid on;
   pause(1);
end
[nu,kmin]=min(nuk);
rofmin=['summation   ';'distribution';'kappa       '];
disp(['The upper limit was found using the ' rofmin(kmin,:) ' ranking index.']);
```

opt.m (called by *ul.m*):

```
% Opt.m
% Matlab script file to optimize one combination of struts
% based on the spherical compliant model estimate of
% damping ratio.  Called by bonk.m.
%
% Thomas J. Thompson
%
% 8-31-94
%
%

[dum,nopt]=size(comb);

clear r rcsat rdast;
clear p2Jtpck p2drki p2etaprki p2nrki pdpr   petapr;
clear petaprchk petaprchk   pnpr  Jtc Jtcc Jt Jtci ;

% Solve real eigenvalue problem to get naught compliant estimate.
naught;

flopsst=flops;

% Begin the optimization.

if 1<0
% Use the tanh index first.

clear cvec;
clear Jt;
clear Jtold;
c=1000;
Jtc=0; %1.;
try=0;
perc=0; %1;
delc=100;
while abs(Jtc)>1e-8 | abs(perc)>.01
%for c=1000:delc:5000
try=try+1;
cvec(try)=c;
% Get Jt and its derivatives.
r=c*relse(comb,:);
 for jj=1:m
  bign(jj)=0;
  bigd(jj)=1;
  for ii=1:nopt
   r2=r(ii,jj)^2;
```

```
    bign(jj)=bign(jj)+nkappas(ii,jj)*(r(ii,jj)/(1+r2));
    bigd(jj)=bigd(jj)+nkappas(ii,jj)*r2/(1+r2);
   end % for ii
   eta(jj)=bign(jj)/bigd(jj);
   for ii=1:nopt
    r2=r(ii,jj)^2;
    petapr(ii,jj)=nkappas(ii,jj)*((1-r2)*bigd(jj)-2*r(ii,jj)*bign(jj))/...
((1+r2)^2*bigd(jj)^2);
    pnpr(ii,jj)=nkappas(ii,jj)*(1-r2)/(1+r2)^2;
    pdpr(ii,jj)=nkappas(ii,jj)*2*r(ii,jj)/(1+r2)^2;
    petaprchk(ii,jj)=(pnpr(ii,jj)*bigd(jj)-bign(jj)*pdpr(ii,jj))/bigd(jj)^2;
   end % ii
  end % jj
  etau=eta./tau;
  if exist('Jt')==1
   Jtold=Jt;
  end % if
  Jt=sum(w.*tau.*tanh(etau));
  pJtpeta=w.*(sech(etau).^2);
  for ii=1:nopt
   Jtci(ii)=0;
    for jj=1:m
    Jtci(ii)=Jtci(ii)+pJtpeta(jj)*relse(comb(ii),jj)*petapr(ii,jj);
   end % for jj
  end % for ii
  if exist('Jtc')==1
   Jtcold=Jtc;
  end
  Jtc=sum(Jtci.');
  p2Jtpeta2=-2*(w./tau).*tanh(etau).*(sech(etau).^2);
  p2Jtpcki=zeros(nopt,nopt);
  p2nrki=zeros(nopt,nopt);
  for jj=1:m
   for ii=1:nopt
    rterm=1+r(ii,jj)^2;
    p2nrki(ii,ii)=nkappas(ii,jj)*2*r(ii,jj)*(r(ii,jj)^2-3)/rterm^3;
    p2drki(ii,ii)=nkappas(ii,jj)*2*(1-3*r(ii,jj)^2)/rterm^3;
   end % for ii
   for ii=1:nopt
    for kk=1:ii
     p2etaprki(kk,ii)=((-pnpr(ii,jj)*pdpr(kk,jj)-pnpr(kk,jj)*pdpr(ii,jj))*bigd(jj)+...
2*pdpr(ii,jj)*pdpr(kk,jj)*bign(jj)+p2nrki(kk,ii)*bigd(jj)^2-...
bign(jj)*p2drki(kk,ii)*bigd(jj))/(bigd(jj)^3);
     p2Jtpcki(kk,ii)=p2Jtpcki(kk,ii)+...
(p2Jtpeta2(jj)*petapr(kk,jj)*petapr(ii,jj)+pJtpeta(jj)*p2etaprki(kk,ii))...
  *relse(comb(kk),jj)*relse(comb(ii),jj);
     p2Jtpcki(ii,kk)=p2Jtpcki(kk,ii);
    end % for kk
   end % for ii
  end % for jj
  Jtcc=sum(sum(p2Jtpcki).');
  %try;
  %clc;
  c;
  Jt;
  if exist('Jtold')==1
   perc=(Jt-Jtold)*100/Jtold;
  end
  Jtc;
  Jtcc;
  (Jtc-Jtcold)/delc;
  Jtcc/ans;
  %pause;
  petapr;
  petaprchk;
  %if 1<0  %
  next=c-Jtc/Jtcc;
  if next<c/4
     c=c/4;
   elseif next>10*c
     c=10*c;
```

```
  else
    c=next;
 end
 %end %
end
cvec(try+1)=c;
end % if 1<0
flopstan=flops-flopsat;

% Next, re-optimize for the saturation index using golden section.

a=3000;
b=60000;
c=a+.38*(b-a);
d=a+.62*(b-a);
cd=c;
cd=d;

eps1=10;
eps2=10;
trygs=0;
Jcsat=0;
Jdsat=0;
while eps1>1 & eps2>.00001& Jcsat<Jnec & Jdsat<Jnec
 trygs=trygs+1;
 Jcold=Jcsat;
 Jdold=Jdsat;
 if Jcold<Jdold

   a=c;
   c=d;
   d=a+0.62*(b-a);
   Jcsat=Jdsat;
   cd=d;

 else

   b=d;
   d=c;
   c=a+0.38*(b-a);
   Jdsat=Jcsat;
   cd=c;

 end % if

  rsat=cd*relse(comb,:);
  for jj=1:m
   r2=rsat(:,jj).^2;
   numsat(jj)=sum(nkappas(:,jj).*rsat(:,jj)./(1+r2));
   densat(jj)=1+sum(nkappas(:,jj).*r2./(1+r2));
  end % jj
  etasat=numsat./densat;
  for jj=1:m
   cdsat(jj)=min([etasat(jj)/tau(jj) 1]);
  end % jj
  Jc=sum(w.*etasat);
  Jcdsat=sum(w.*tau.*cdsat);
 if Jcold<Jdold
   Jdsat=Jcdsat;
  else
   Jcsat=Jcdsat;
 end % if

 %clc
 %[a c d b]
 %[Jcsat Jdsat]
 %[Jc Jcsat Jdsat Jd]
 %pause
 eps1=abs(d-c);
 eps2=abs(Jdsat-Jcsat);
end % while
```

```
Jsat=Jcsat;
Jsatv(nopt)=Jsat;
cstrsat=c;
%Jt;
cvec;
flopsgs=flops-flopsst-flopstan
```

naught.m (called by *opt.m*):

```
clear nkappas ;
clear nks;

% Solve real eigenvalue problem to get naught compliant estimate.
%flops(0);
KA=diag(kAj);
Ks=diag(ksi(comb));
nK=KA+gamma(comb,:).'*(Ks/2)*gamma(comb,:);
[V,E]=eig(nK);
[e, jvec] = sort(diag(E));
nom2=diag(e);
nPhi=V(:, jvec);
%flops
nKA=diag(diag(nPhi.'*KA*nPhi));

for jj=1:m
 for ii=1:nopt
  nks(ii,jj)=((gamma(comb(ii),:)*nPhi(:,jj))^2)*ksi(comb(ii));
  nkappas(ii,jj)=nks(ii,jj)/nKA(jj,jj);
 end % for ii
end % for jj
%sum(nkappas)/2
%pause
```

sachen.m (*optsa.m* is very similar to *opt.m*):

```
% Matlab file sachenbest.m to search for the best combination
% of struts for damping a structure.
%
%  Thomas J. Thompson
%
% 1-20-94
%
clear
path(path,'/home/thomps/res/spice/matlab/sa');
time=clock;
hr=num2str(time(4));
mne=num2str(time(5));
stdate=date;
tic;
cpu_time=cputime;

!rm -f /tmp/freqf.data /tmp/dispf.data /tmp/conf.data /tmp/temp /tmp/conrods.data
!grep '  [0-9]' ../akk1/freq.data > /tmp/freqf.data
load /tmp/freqf.data
!grep '^3' ../akk1/conrods.data >/tmp/conrods.data
load /tmp/conrods.data
!grep ' 3' ../akk1/disp.data > /tmp/dispf.data
load /tmp/dispf.data
!grep '^3' ../akk1/freq.data > /tmp/conf.data
load /tmp/conf.data
% Check order of prospective damping locations.
if norm(conf-conrods)>.01
  disp('Problem with order of damping locations!');
end
load blistblk
```

```
type=blistblk(:,4);

flops(0);
[n,dum]=size(conrods);
[mn,dum]=size(dispf);
m=mn/n;

load ../akk1/ranks.data;

nul=46;
lb=4;

logfile=['sachen' stdate '.log'];
eval(['!rm -f ' logfile])
eval(['!echo ''This is the log file created by sachen.m (matlab script)'' > ' logfile]);
eval(['!echo ''on ' stdate ' at ' hr ':' mno ' (24hr).'' >> ' logfile])
message='The file beginning with the following header was used as input.'
eval(['!echo ' message '>> ' logfile]);
eval(['!echo '' '' >> ' logfile])
!rm -f /tmp/temp
!grep '^[a-zA-Z]' ../akk1/freq.data>/tmp/temp
eval(['!cat /tmp/temp >> ' logfile])
eval(['!echo '' '' >> ' logfile])

kaporder=ranks(:,3);
comb=kaporder(1:nul).';
combnot=kaporder(nul+1:n).';
%comb=1:nul;      %  Experiment
%combnot=nul+1:n; %
combold=comb;
combnotold=combnot;
combfirst=comb;

%rand('seed',0);
rand('seed',sum(100*clock));
prob1=0.2;
thr1=.995;
propw=0;
propwsel=1.0;
propwboot=0.0;
nopt=nul;
nt=0;
ct=0;
Jsat=0;
Jsatold=0;
gscount=0;
gstime=0;

w=ones(1,m);
tau=0.04*ones(1,m);
Jnec=sum(w.*tau);

% Get kappas, relse, etc.

omo2=freqf(1:m,3);
omo=sqrt(omo2);
for ii=1:n
 row=m*(ii-1);
 beta(ii,1:m)=dispf(row+1:row+m,2).';
 om2(ii,1:m)=freqf(row+m+1:row+2*m,3).';
end

for ii=1:n
 alphai(ii)=sum(beta(ii,1:m).^2);
 gamma(ii,1:m)=beta(ii,1:m)/sqrt(alphai(ii));
 alpha(ii,:)=beta(ii,1:m).^2;
end

kAj=omo2;
for jj=1:m
 ks(:,jj)=om2(:,jj)-kAj(jj);
```

```
end
ksi=sum(ks.');
for ii=1:n
 ks(ii,:)=gamma(ii,1:m).^2*ksi(ii);
end
for jj=1:m
 kappas(:,jj)=ks(:,jj)/kAj(jj);
end

for ii=1:n
 for jj=1:m
  relse(ii,jj)=omo(jj)*alpha(ii,jj)/(ks(ii,jj)+1.e-15);
 end
end
 set(gcf,'DefaultTextFontName','Times');
 set(gcf,'DefaultTextFontSize',8);
 set(gcf,'DefaultAxesFontName','Times');
!rm -f /tmp/freqf.data /tmp/dispf.data /tmp/conf.data /tmp/temp /tmp/conrods.data

% Outer loop to set the number of dampers.

while nt<50 & nopt>lb-1
 first=1;
 Jsat=0;
 Jsatbest=0;
 Jsatold=0;
 combbest=comb;
 combnotbest=combnot;
 comblast=comb;
 etasat=zeros(1,nopt);
 theta=1;
 thetamin=0;

 disp(['nopt= ' num2str(nopt)]);

 % Loop to iterate on combinations.

 while (nt<50 | theta>thetamin) & Jsat<Jnec
 f1=0;
 while (nt<20 & Jsat<Jnec) | f1==0
 f1=1; f2=0; Jksat=0; Jsatthr=0;
 while (nt<20 & Jsat<Jsatthr) | f2==0
 f2=1; f3=0;
 while (nt<20 & Jksat<Jsatthr) | f3==0
 f3=1;

  clf;
  % Loop to set initial temperature or to reselect a new position.

  if first==1
    first=0;
    nt=0;
    etaksat=min(tau,sum(kappas(comb,:)/2));
    Jksat=sum(w.*etaksat);
    theta1=(thr1-1)*Jnec/log(prob1);
    theta=theta1;
    thetamin=0.01*theta;
    % Assign new weighting matrices.
    for jj=1:m
     %wk(jj)=2/(1+(etaksat(jj)/tau(jj))^2);
      wk(jj)=max((tau(jj)-etaksat(jj))/tau(jj),.0001);
     % wk(jj)=max((tau(jj)-etaksat(jj))/tau(jj),-.1);
    end
    wk=wk/sum(wk);
    % Assign new probabilities.
    pro;
    combpro=comb;
   else
    comb=combold;
    combnot=combnotold;
    etasat=etasatold;
```

```
    if norm(combpro-comb)>0
     % Assign new weighting matrices.
     for jj=1:m
      wk(jj)=max((tau(jj)-etasat(jj))/tau(jj),.0001);
      %wk(jj)=max((tau(jj)-etasat(jj))/tau(jj),-.1);
     end
     wk=wk/sum(wk);
     % Assign new probabilities.
     pro;
     combpro=comb;
    end % if norm(combpro-comb)>0
    % Select new position.
    ii=1;
    select=1;
    rnum=rand;
    while pselcum(ii)<rnum
     ii=ii+1;
     select=ii;
    end

  %if size(pselcum,2)>select& select>1 pselcum(select-1:select+1)
  %end
  %rnum
  %psel(select)
  %if select>1 pselcum(select)-pselcum(select-1)
  %end
  %pause

    % Unselect one of the present strut locations.
    ii=1;
    unselect=1;
    rnum=rand;
    while punselcum(ii)<rnum
     ii=ii+1;
     unselect=ii;
    end


   if type(unselect)==0
    uslstr='co';
    else
    uslstr='cx';
   end
   if type(select)==0
    slstr='co';
    else
    slstr='cx';
   end
if ct/10-round(ct/10)==0 & 1<0
%bar(Jksel(combnotold));
  subplot(3,1,1), bar(Jksel(combold));
hold on;
subplot(3,1,1), bar(pboot,'m-.');
hold on;
subplot(3,1,1),bar(unselect,pboot(unselect),uslstr)
  subplot(3,1,2), bar(Jksel(combnotold));
hold on;
subplot(3,1,2), bar(psel,'m-.');
hold on;
subplot(3,1,2),bar(select,psel(select),slstr)
  subplot(3,1,3), bar(wk);
hold on;
subplot(3,1,3), bar(max((tau-etasat)./tau,.0001),'rx');
pause(1);
    end % if


    temp=combold(unselect);
    comb(unselect)=combnotold(select);
    combnot(select)=temp;
  selpr=psel(select)*(n-nopt);
```

```
  bootpr=pboot(unselect)*nopt;
  disp(['selpr=' num2str(selpr)]);
  disp(['bootpr=' num2str(bootpr)]);
  if sum(Jksel)~=0
   sr=Jksel(select)/mean(Jksel(combnot));
   br=Jksel(unselect)/mean(Jksel(comb));
   else
   sr=0;
   br=0;
  end
  disp(['sel ratio= ' num2str(sr)])
  disp(['boot ratio= ' num2str(br)])

%pause
  end % if first==1

  %disp(comb);

  % Set threshold.
  Jsatthr=Jsatold + theta*log(rand);

  % Get nkappas for the present combination.

  naught;

  % Get Jksat based on nkappas.
  for jj=1:m
   etaksat(jj)=min(tau(jj),sum(nkappas(:,jj)/2));
  end % for jj
  Jksat=sum(w.*etaksat);
  %disp(['[Jksat Jsatthr] = ' num2str(Jksat) ' ' num2str(Jsatthr)])

   % ct is the number of function evaluations
    ct=ct+1;
    Jvec(ct)=Jsatold;
    tvec(ct)=theta;
    noptvec(ct)=nopt;

  nt=nt+1

  if ct>1
   srvec(ct)=sr;
   brvec(ct)=br;
   disp(['Mean sel ratio= ' num2str(mean(srvec))]);
   disp(['Mean boot ratio= ' num2str(mean(brvec))]);
  end % if ct>1

  % Compare to threshold value based on Jksat.
 end %  while (nt<20 & Jksat<Jsatthr) | f3==0

   disp('Jksat>=Jsatthr')

   % Compare based on optimization of the present combination.
    gscp=cputime;
   optsa  % call subroutine
   gstime=gstime+cputime-gscp;
   gscount=gscount+1;
   disp(['[Jsat Jsatthr] = ' num2str(Jsat) ' ' num2str(Jsatthr)])

% Check to make sure that Jksat>Jsat
  Jrat=Jksat/Jsat; disp(['Jrat=' num2str(Jrat)])
  if Jrat<1
disp('Trouble: Jksat<Jsat')
  pause
  end

 end % while (nt<20 & Jsat<Jsatthr) | f2==0

   if Jsat>=Jsatthr    %2-9

    disp('Jsat>=Jsatthr')    %2-9
```

```
%    % Reset nt counter
   if Jsat>Jsatbest
%   if Jsat>Jsatold
%   if Jsat>Jsatthr
     nt=0;
    end % if Jsat>Jsatthr or
% if Jsat>Jsatold  or
        % if Jsat>Jsatbest

    Jsatold=Jsat;
    combold=comb;
    combnotold=combnot;
    etasatold=etasat;
    Jvec(ct)=Jsatold;

    % Update Jsatbest.
    if Jsatold>Jsatbest
      disp('Jsatbest reassigned!');
      Jsatbest=Jsatold;
      combbest=combold;
      combnotbest=combnotold;
      etasatbest=etasatold;
    end

    % Assign new weighting matrices.

%    for jj=1:m
%     %wk(jj)=2/(1+(etasat(jj)/tau(jj))^2);
%      wk(jj)=max((tau(jj)-etasat(jj))/tau(jj),.0001);

%    end
%    wk=wk/sum(wk);

%    % Assign new probabilities.

%    pro;
%     combpro=comb;

  end %  if Jsat>=Jsatthr  %2-9


  %plot(pboot);
  %bar([Jksel(comb) 0 Jksel(combnot)]);
  %bar(Jksel(combnot));
   %bar(Jksel(comb));
  %comblast=comb;
  %pause(1);


    % Check for thermal equilibrium.

 end % while (nt<20 & Jsat<Jnec) | f1=0

     if theta>thetamin
       disp('Thermal Equilibrium!')
       theta=0.8*theta;
       nt=0;
       disp(['theta = ' num2str(theta)]);
      end

    disp(['[Jsat Jsatold Jsatbest nt] = ' num2str(Jsat) ' ' num2str(Jsatold) ...
' ' num2str(Jsatbest) ' ' num2str(nt)]);

%  Reset solution to Jsatbest.
   if Jsat<Jsatbest & Jsat<Jnec &(nt<50|theta>thetamin)
    Jsatold=Jsatbest;
    combold=combbest;
    combnotold=combnotbest;
    etasatold=etasatbest;
   end % if Jsat>Jsatbest & Jsat<Jnec &(nt<50|theta>thetamin)
```

```
 end % while (nt<50|theta>thetamin) & Jsat < Jnec

 if nt<50
  % Deselect a strut.
  combnp1=comb;
  etanp1=etasat;
  cstrnp1=c;


%bar(sort(Jksel(combold)));
%pause;
  if propow>0 | propowsel>0 | propowboot>0
   [dum,indJk]=sort(1./Jksel(combold));
   combold=combold(indJk);
  end % if propow>0 | propowsel>0 | propowboot>0
  deselpec=Jksel(combold(nopt))*nopt/sum(Jksel);

  bubblestrut=combold(nopt);
  nopt = nopt-1;

  combold=combold(1:nopt);
  combnotold(n-nopt)=bubblestrut;
  comb=combold;
  combnot=combnotold;
%bar(sort(Jksel(combold)));
%pause;
 end % if nt<50

 % Plot Jvec and tvec

 clf;
 subplot(3,1,1), plot(Jvec)
 ylabel('Jsat Evaluation')
 text(1,Jnec*1.02,['sachenbest, ' stdate]);
 grid on;
 subplot(3,1,2), plot(tvec)
 ylabel('Temperature');
 grid on;
 subplot(3,1,3), plot(noptvec);
 text(ct-1,nopt+2.5,['nopt=' num2str(nopt)]);
 xlabel('Total Function Evaluations');
 ylabel('Number of Dampers');
 grid on;
 pause(1);
 eval(['print sachen' stdate ';']);
 eval(['!zwrite thomps -m "New plot from sachenbest. nopt = ' num2str(nopt) ...
 ', ct = ' num2str(ct) '." &']);

end % while nt<50 & nopt > lb-1

eval(['diary ' logfile]);
etasat
disp(['propowsel = ' num2str(propowsel)]);
disp(['propowboot = ' num2str(propowboot)]);
disp(['prob1 = ' num2str(prob1)]);
disp(['thr1 = ' num2str(thr1)]);

disp('combnp1 = ');
disp(combnp1);
if propow~=0 | propowsel~=0 | propowboot~=0
 save combnp1.data combnp1 -ascii;
end % if
disp(['cstrnp1 = ' num2str(cstrnp1)]);
disp('The values of eta at the last acceptable design (combnp1).');
etanp1
total_cpu_time=cputime-cpu_time
total_elapsed_time=toc
disp(['The total number of function evaluations is: ' num2str(ct)])
disp(['The total number of golden sections is: ' num2str(gscount)])
disp(['Cpu time per golden section search is: '...
 num2str(gstime/gscount)]);
```

```
diary off;

% Put results in a table file for the latex document.

etastar=size(combnp1,2);
rows=ceil(etastar/2);
eval(['tabfil=fopen([''comboptpp'' num2str(propowsel) ''.tex''],''w'');']);
for ii=1:rows
snum=combnp1(ii);
 n1=['00' num2str(blistblk(snum,2))];
 n2=['00' num2str(blistblk(snum,3))];
 n1=n1((size(n1,2)-1):size(n1,2)) ;
 n2=n2((size(n2,2)-1):size(n2,2));
 if blistblk(snum,4)>0
  ty='D'; else ty='I'; end
 if rows+ii<=etastar
  snumr=combnp1(rows+ii);
  n1r=['00' num2str(blistblk(snumr,2))];
  n2r=['00' num2str(blistblk(snumr,3))];
  n1r=n1r((size(n1r,2)-1):size(n1r,2)) ;
  n2r=n2r((size(n2r,2)-1):size(n2r,2));
  if blistblk(snumr,4)>0
  tyr='D'; else tyr='I'; end
  else
   n1r=' ';
   n2r=' ';
   snumr=0;
   tyr=' ';
 end % if
 fprintf(tabfil,'%3i &%s--%s & %s &%3i &%s--%s & %s \\\\ \n',...
snum,n1,n2,ty,snumr,n1r,n2r,tyr);
end % for ii
fclose(tabfil);

set(gcf,'DefaultTextFontSize',12);
set(gcf,'DefaultTextFontName','Helvetica');
set(gcf,'DefaultAxesFontName','Helvetica');

% Save plot to a file for dissertation.
 eval(['print pp' num2str(propowsel) 'res;']);
```

pro.m (called by sachen.m):

```
% Assign new probabilities.

clear pboot psel;
for ii=1:n
 Jksel(ii)=sum(wk.*kappas(ii,:))/2;
end % for ii

for ii=1:n-nopt
 psel(ii)=(Jksel(combnot(ii))/sum(Jksel(combnot)))^propowsel; %^propow;
end % for ii

for ii=1:nopt
 pboot(ii)=(1/(Jksel(comb(ii))*(sum(1./Jksel(comb)))))^propowboot; %^propow;
end % ii

psel=psel/sum(psel);
pboot=pboot/sum(pboot);

pselcum=cumsum(psel);
punselcum=cumsum(pboot);
```

APPENDIX E. CHARACTERISTICS OF COMPUTERS

NASTRAN computations in this dissertation were performed on the Silicon Graphics Indigo workstation described by the following file.

```
1 100 MHZ IP20 Processor
FPU: MIPS R4010 Floating Point Chip Revision: 0.0
CPU: MIPS R4000 Processor Chip Revision: 2.2
On-board serial ports: 2
On-board bi-directional parallel port
Data cache size: 8 Kbytes
Instruction cache size: 8 Kbytes
Secondary unified instruction/data cache size: 1 Mbyte
Main memory size: 32 Mbytes
Integral Ethernet: ec0, version 1
Disk drive: unit 4 on SCSI controller 0
Tape drive: unit 2 on SCSI controller 0: DAT
Disk drive: unit 1 on SCSI controller 0
Integral SCSI controller 0: Version WD33C93B, revision C
Iris Audio Processor: revision 10
Graphics board: LG1
```

Most of the Matlab computations in this dissertation were performed on the DEC Alpha workstation described by the following files.

```
From DEC factsheet:

DEC 3000 Model 300L AXP Workstation
CPU DECchip 21064 RISC Microprocessor
Clock Speed 100 MHz
On-chip Cache 8 Kbytes of instruction/8 Kbytes of data
On-board Cache 256 Kbytes, write back


  uerf version 4.2-011 (119)

********************************** ENTRY     1. **********************************

----- EVENT INFORMATION -----

EVENT CLASS                              OPERATIONAL EVENT
OS EVENT TYPE                  300.      SYSTEM STARTUP
SEQUENCE NUMBER                  0.
OPERATING SYSTEM                         DEC OSF/1
OCCURRED/LOGGED ON                       Tue Dec  6 09:22:17 1994
OCCURRED ON SYSTEM                       pvO2a8
SYSTEM ID                x0007000F       CPU TYPE: DEC
                                         CPU SUBTYPE: KN16AA
MESSAGE                                  secondary cache was auto sized to 32
                                          _pages
                                         Alpha boot: available memory from
                                          _0x6ae000 to 0x4000000
                                         DEC OSF/1 V1.3 (Rev. 111); Mon Jul 25
                                          _17:01:55 CDT 1994
                                         physical memory = 62.00 megabytes.
                                         available memory = 53.76 megabytes.
                                         using 238 buffers containing 1.85
                                          _megabytes of memory
                                         tcO at nexus
                                         tcdsO at tcO slot 4
                                         ascO at tcdsO slot O
                                         rz3 at ascO bus O target 3 lun O (DEC
                                          _   RZ26     (C) DEC 392A)
                                         lnO: DEC LANCE Module Name: PMAD-BA
                                         lnO at tcO slot 5
                                         lnO: DEC LANCE Ethernet Interface,
                                          _hardware address: 08:00:2b:39:79:b2
                                         sccO at tcO slot 5
                                         bbaO at tcO slot 5
                                         fbO at tcO slot 6
                                          1024X768
                                         DEC3000 - M300 system
                                         Firmware revision: 3.3
                                         PALcode: OSF version 1.35
                                         lvmO: configured.
                                         lvm1: configured.
                                         lvm2: configured.
```

APPENDIX F. COMPUTER PROGRAMS USED IN OPTIMIZING STRUCTURAL DAMPING USING COMPLEX EIGENVALUE ANALYSIS

Matlab program to perform simulated annealing by calling on NASTRAN to perform complex eigenvalue analysis. (The Matlab script *optsanas.m* is very similar to opt.m.)

sanas.m:

```
% Matlab file sanas.m to search for the best combination
% of struts for damping a structure.
% Run on indy using sbat , i.e. matlab < sanas.m > sanasout.log
% Plot results using sanplot.m .
%
%  Thomas J. Thompson
%
% 9-10-94
%
clear
time=clock;
hr=num2str(time(4));
mne=num2str(time(5));
stdate=date;
tic;
cpu_time=cputime;
nnc=0;
tnc=0;
iris='/tmp_mnt/home/usr2/thomps/iris';
%rand('seed',sum(100*clock))

m=16;
n=306;
eval(['load ' iris '/bar/blistblk']);
for ii=1:n
 n1=['00' num2str(blistblk(ii,2))];
 n2=['00' num2str(blistblk(ii,3))];
 n1=n1((size(n1,2)-1):size(n1,2)) ;
 n2=n2((size(n2,2)-1):size(n2,2));
nodes(ii,:)=[n1 n2];
end
blistblk(:,2:3);
nodes;

flops(0);
m=16;
n=306;
```

```
%nul=45;
nul=20;
lb=4;

logfile=['sanas' stdate '.log'];
eval(['!rm -f ' logfile])
eval(['!echo ''This is the log file created by sanas.m (matlab script)'' > ' logfile]);
eval(['!echo ''on ' stdate ' at ' hr ':' mne ' (24hr).'' >> ' logfile])
eval(['!echo '' '' >> ' logfile])

% Scramble struts and select beginning combination.

scrand=rand(n,1);
[scrsort,scramble]=sort(scrand);
comb=scramble(1:nul).';
combnp1=comb;
combnot=scramble(nul+1:n).';
combold=comb;
combnotold=combnot;

prob1=0.2
thr1=.995
nopt=nul;
nt=0;
ct=0;
Jsat=0;
Jsatold=0;
cstrsat=0;
cstrnp1=0;

w=ones(1,m);
tau=0.04*ones(1,m);
Jnec=sum(w.*tau);

% Do a restart if a date is given in restart.date .

[dum,restart]=unix('cat restart.date');
[dee,ressiz]=size(restart);
restart=restart(1:ressiz-1);
if dum==0
 if exist('restart')==1
  disp(['This is a restart using the reference date:']);
  disp(restart);
  disp(' ');
  [dum2,tom]=unix(['cp Jvec' restart '.data Jvec.data']);
  if dum2==0
   disp('Restart appears successful!');
   else
   disp('Restart unsuccessful!');
   quit
  end % if
  eval(['!cp tvec' restart '.data tvec.data']);
  eval(['!cp noptvec' restart '.data noptvec.data']);
  eval(['!cp comb' restart '.data comb1.data']);
  eval(['!cp combnot' restart '.data combnot.data']);
  load Jvec.data;
  load tvec.data;
  load noptvec.data;
  load comb1.data;
  load combnot.data;
  ct=size(Jvec,2);
  comb=comb1;
  nopt=size(comb,2);
 end % if
end % if

 set(gcf,'DefaultTextFontName','Times');
 set(gcf,'DefaultTextFontSize',8);
 set(gcf,'DefaultAxesFontName','Times');
```

```
% Outer loop to set the number of dampers.

while nt<50 & nopt>lb-1
 first=1;
 Jsat=0;
 Jsatbest=0;
 Jsatold=0;
 combbest=comb;
 combnotbest=combnot;
 comblast=comb;
 etasat=zeros(1,nopt);
 theta=1;
 thetamin=0;

 disp(['nopt= ' num2str(nopt)]);

 % Loop to iterate on combinations.

 while (nt<50 | theta>thetamin) & Jsat<Jnec
 f1=0;
 while (nt<20 & Jsat<Jnec) | f1==0
 f1=1; f2=0; Jksat=0; Jsatthr=0;
 while (nt<20 & Jsat<Jsatthr) | f2==0
 f2=1;

  clf;
  % Loop to set initial temperature or to reselect a new position.

  if first==1
    first=0;
    nt=0;
    if exist('restart')==1
     theta1=tvec(ct);
     else
     theta1=(thr1-1)*Jnec/log(prob1);
    end
    theta=theta1;
    thetamin=.01*(thr1-1)*Jnec/log(prob1);
   else
    % Select new position.
    select=ceil(rand*(n-nopt));

    % Unselect one of the present strut locations.
    unselect=ceil(rand*nopt);
    comb=combold;
    combnot=combnotold;
    unselect;
    size(combold)
    temp=combold(unselect)
    comb(unselect)=combnotold(select)
    combnot(select)=temp;

  end % if first==1

  %disp(comb);

  % Set threshold.
  Jsatthr=Jsatold + theta*log(rand);

  % Evaluate maximum for this combination

   optsanas  % call subroutine
%Jsat=.63
  %cstrsat=1800
   cstrsat

   % Place values in vectors for output.
   % ct is the number of function evaluations

    ct=ct+1;
    Jvec(ct)=Jsatold;
```

```
    tvec(ct)=theta;
    noptvec(ct)=nopt;
    combnp1=comb;
    save Jvec.data Jvec -ascii
    save tvec.data tvec -ascii
    save noptvec.data noptvec -ascii

  nt=nt+1

   % Compare based on optimization of the present combination.

 end % while (nt<20 & Jsat<Jsatthr) | f2==0

   if Jsat >= Jsatthr

    disp('Jsat>=Jsatthr')

    % Reset nt counter
    if Jsat>Jsatold
     nt=0;
    end % if Jsat>Jsatold

    Jsatold=Jsat;
    combold=comb;
    combnotold=combnot;
    Jvec(ct)=Jsatold;
    combnp1=comb;

    % Update Jsatbest.
    if Jsatold>Jsatbest
      disp('Jsatbest reassigned!');
      Jsatbest=Jsatold;
      combbest=combold;
      combnotbest=combnotold;
    end

  end %  if Jsat>=Jsatthr

% if 1<0    % Don't do this--problems plotting in background.
%-----------------------------------------------
 % Plot Jvec and tvec

 subplot(3,1,1), plot(Jvec)
 ylabel('Jsat Evaluation')
 text(1,.6,['sanas, ' stdate]);
 grid on;
 subplot(3,1,2), plot(tvec)
 ylabel('Temperature');
 grid on;
 subplot(3,1,3), plot(noptvec);
 xlabel('Total Function Evaluations');
 ylabel('Number of Dampers');
 text(ct,nopt+2,['nopt=' num2str(nopt)]);
 grid on;
 pause(1);
 eval(['print sanas' stdate ';']);
%-----------------------------------------------
% end % if 1<0

    % Check for thermal equilibrium.

 end % while (nt<20 & Jsat<Jnec) | f1=0

     if theta>thetamin
       disp('Thermal Equilibrium!')
       theta=0.8*theta;
       nt=0;
       disp(['theta = ' num2str(theta)]);
      % Save stuff for a possible restart.
       eval(['!cp Jvec.data Jvec' stdate '.data']);
       eval(['!cp tvec.data tvec' stdate '.data']);
```

```
        eval(['!cp noptvec.data noptvec' stdate '.data']);
        eval(['save comb' stdate '.data comb -ascii;']);
        eval(['save combnot' stdate '.data combnot -ascii;']);
        eval(['!echo ' stdate ' > restart.date;']);

% Put results in a table file for the latex document.

        etastar=size(combnp1,2);
        rows=ceil(etastar/2);
        tabfil=fopen('combopt.tex','w');
        for ii=1:rows
        snum=combnp1(ii);
         n1=['00' num2str(blistblk(snum,2))];
         n2=['00' num2str(blistblk(snum,3))];
         n1=n1((size(n1,2)-1):size(n1,2)) ;
         n2=n2((size(n2,2)-1):size(n2,2));
         if blistblk(snum,4)>0
          ty='D'; else ty='I'; end
         if rows+ii<=etastar
          snumr=combnp1(rows+ii);
          n1r=['00' num2str(blistblk(snumr,2))];
          n2r=['00' num2str(blistblk(snumr,3))];
          n1r=n1r((size(n1r,2)-1):size(n1r,2)) ;
          n2r=n2r((size(n2r,2)-1):size(n2r,2));
          if blistblk(snumr,4)>0
          tyr='D'; else tyr='I'; end
          else
           n1r=' ';
           n2r=' ';
           snumr=0;
           tyr=' ';
         end % if
         fprintf(tabfil,'%3i &%s--%s & %s &%3i &%s--%s & %s \\\\ \n',...
        snum,n1,n2,ty,snumr,n1r,n2r,tyr);
        end % for ii
        fclose(tabfil);

       end

    disp(['[Jsat Jsatold Jsatbest nt] = ' num2str(Jsat) ' ' num2str(Jsatold) ...
' ' num2str(Jsatbest) ' ' num2str(nt)]);

%  Reset solution to Jsatbest.
   if Jsat<Jsatbest & Jsat<Jnec &(nt<50|theta>thetamin)
    Jsatold=Jsatbest;
    combold=combbest;
    combnotold=combnotbest;
   end % if Jsat>Jsatbest & Jsat<Jnec &(nt<50|theta>thetamin)

 end % while (nt<50|theta>thetamin) & Jsat < Jnec

 if nt<50
  % Deselect last strut.
  combnp1=comb;
  etanp1=etasat;
  cstrnp1=cstrsat;
  bubblestrut=combold(nopt);

  nopt = nopt-1;

  combold=combold(1:nopt);
  combnotold(n-nopt)=bubblestrut;
  comb=combold;
  combnot=combnotold;
      % Save stuff for a possible restart.
       eval(['!cp Jvec.data Jvec' stdate '.data']);
       eval(['!cp tvec.data tvec' stdate '.data']);
       eval(['!cp noptvec.data noptvec' stdate '.data']);
       eval(['save comb' stdate '.data comb -ascii;']);
       eval(['save combnot' stdate '.data combnot -ascii;']);
       eval(['!echo ' stdate ' > restart.date;']);
```

```
 end % if nt<50

 % Plot Jvec and tvec

 % (Put here if only to print each new nopt.)
 %eval(['!cp sanas' stdate '.ps sanas.ps']);

end % while nt<50 & nopt > lb-1
eval(['diary ' logfile]);
disp(['etasat = ' num2str(etasat)]);
combnp1;
disp('cstrnp1 = ');
disp(num2str(cstrnp1));
disp('Total number of nastran calls:')
nnc
diary off;
set(gcf,'DefaultTextFontSize',12);
set(gcf,'DefaultTextFontName','Helvetica');
set(gcf,'DefaultAxesFontName','Helvetica');
```

nasdamp.m:

```
% nasdamp.m
% Matlab file to call nastran to perform complex
% eigenvalue analysis.
%
% 10-5-94
%
function [damping]=nasdamp(c,m,iris);

% Make something like   pvisc,3,500.,0.
pvfile=fopen([iris '/ncrun/pv.dat'],'w');
fprintf(pvfile,'pvisc,3,%6.1f,0.\n',c);
fclose(pvfile);
nnasps=3;
while nnasps>2
 % Check for other nastran processes.
 [dum,nasps]=unix('ps -edalf | grep mscV68 | wc | cut -c9-10');
 nnasps=eval(nasps);
 if nnasps>2
  disp('Other nastran process running; wait for 1 minute.');
  !date;
  pause(60);
  else
  !who
  eval(['!nastran ' iris '/ncrun/nc.dat bat="no" scr=yes out=' iris ...
  '/ncrun/nc old=no notify=no']);
  disp('Pause only 5 seconds.');  % Change made 11-7-94
  pause(5);
 end % if
end % while
!rm -f ncres.data
[dum,dee] = unix(['grep ''E+'' ' iris '/ncrun/nc.f06>ncres.data']);
if dum==0
 load ncres.data;
 else
 ncres=[0 0 0 0 0 0]
end % if
damping=ncres(:,6).';
mres=size(ncres,1);
if mres<m
 damping(mres+1:m)=zeros(1,m-mres);
 disp('Less than m modes found.');
end % if
if mres>m
 damping=damping(1:m);
 disp('More than m modes found.');
end % if
```

NASTRAN files used to evaluate modal damping. *nc.dat*:

```
$ Copied from /local/usr2/thomps/iris/ncrun/ on 4-29-95.  Some comment lines deleted.
$id  ak
$ Thomas Thompson   10-21-93
sol 107 $ complex solution with provision for restart
time 1440
diag 8
cend
$
title = Complex eigenvalue analysis of structure
$
echo=none
cmethod = 3  $ complex
set 3 = 3,26,28,48,51,57
disp=none
$set 2 = 2 thru 100
$disp(punch) = 3
$ese(punch) = 2
SPC = 1
$
begin bulk
param,tiny,1.e-20
include '/tmp_mnt/home/usr2/thomps/iris/ncrun/eig.dat'
include '/tmp_mnt/home/usr2/thomps/iris/narun/model.dat'
include '/tmp_mnt/home/usr2/thomps/iris/ncrun/vstr.dat'
include '/tmp_mnt/home/usr2/thomps/iris/ncrun/pv.dat'
mat1,20,1.1e+11,2.65e+9,,0.  $out with model5a model
pbar,11,20,8.045e-5,5.00e-8,5.0e-8,1.e-7  $need to ch for modelfull
pbar,211,20,1.609e-4,5.00e-8,5.0e-8,1.e-7  $"
enddata
```

ncrun/eig.dat:

```
eigc,3,clan,,,,1.e-5
,-1.,345.,,,,,2
,-40.,950.,,,,,14
```

Portion of *vstr.dat*:

```
$ Copied from /local/usr2/thomps/iris/ncrun 4-29-95.
cordlr,3246,32,46,99
grid,43246,3246,0.,0.,.01,3246
cbar,5323246,211,32,43246,99
,56,1
cbar,5324646,211,43246,46,99
,,56
cvisc,6323246,3,32,43246
```

pv.dat:

```
pvisc,3,11628.4,0.
```