A Finite Element, Reduced Order, Frequency Dependent Model of Viscoelastic Damping

Jason Salmanoff

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Engineering Science and Mechanics

Daniel J. Inman, Chair
Romesh C. Batra
Michael W. Hyer

September 5, 1997
Blacksburg, VA

Keywords: Viscoelasticity, Finite Elements, Model Order Reduction, Vibrations

A Finite Element, Reduced Order, Frequency Dependent Model of Viscoelastic Damping

Jason Salmanoff

(ABSTRACT)

This thesis concerns itself with a finite element model of nonproportional viscoelastic damping and its subsequent reduction. The Golla-Hughes-McTavish viscoelastic finite element has been shown to be an effective tool in modeling viscoelastic damping. Unlike previous models, it incorporates physical data into the model in the form of a curve fit of the complex modulus. This curve fit is expressed by minioscillators. The frequency dependence of the complex modulus is accounted for by the addition of internal, or dissipation, coordinates. The dissipation coordinates make the viscoelastic model several times larger than the original. The trade off for more accurate modeling of viscoelasticity is increased model size.

Internally balanced model order reduction reduces the order of a state space model by considering the controllability/observability of each state. By definition, a model is internally balanced if its controllability and observability grammians are equal and diagonal. The grammians serve as a ranking of the controllability/observability of the states. The system can then be partitioned into most and least controllable/observable states; the latter can be statically reduced out of the system. The resulting model is smaller, but the transformed coordinates bear little resemblance to the original coordinates. A transformation matrix exists that transforms the reduced model back into original coordinates, and it is a subset of the transformation matrix leading to the balanced model. This whole procedure will be referred to as Yae's method within this thesis.

By combining GHM and Yae's method, a finite element code results that models nonproportional viscoelastic damping of a clamped-free, homogeneous, Euler-Bernoulli beam, and is of a size comparable to the original elastic finite element model. The modal data before reduction compares well with published GHM results, and the modal data from the reduced model compares well with both. The error between the impulse response before and after reduction is negligible. The limitation of the code is that it cannot model sandwich beam behavior because it is based on Euler-Bernoulli beam theory; it can, however, model a purely viscoelastic beam. The same method, though, can be applied to more sophisticated beam models. Inaccurate results occur when modes with frequencies beyond the range covered by the curve fit appear in the model, or when poor data are used. For good data, and within the range modeled by the curve fit, the code gives accurate modal data and good impulse response predictions.

Acknowledgments

The author would like to thank Dr. Inman of Virginia Tech for making this thesis possible, and for his guidance; Eric Austin, Greg Agnes, and Marca Lam for their patience in answering questions and for providing direction; visiting professor Dr. Friswell from Swansea, Wales, U. K., for reviewing results and providing additional insight; and the rest of MSL—Jens, Sergio, Debbie, and Clay—for friendship, and a new appreciation for the game of darts. Also, thanks to Drs. Radcliffe and Sommerton, Department of Mechanical Engineering, Michigan State University, for their guidance, and their belief that I had what it took to succeed at graduate school. Finally, to friends—thanks, with love.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The finite element method (FEM) is one of the most powerful numerical methods available to engineers today. It allows for an approximate solution of a large range of problems by breaking the geometric domain of the problem into several smaller elements and performing calculations at only a few discrete points within each element. Information from these calculations can be assembled into global matrices, thereby taking advantage of both matrix theory and modern high speed computing.

In the area of structural mechanics, one of the most challenging problems is the modeling of viscoelastically damped structures.   The most common, and easiest computationally, damping model is proportional damping. The main drawback of this technique is that damping factors must often be guessed at, and frequently are very different from actual characteristics of the damped system of interest. For instance, proportional damping produces real valued mode shapes, but measurements on structures with viscoelastic treatments yield complex mode behavior. By using a complex modulus in the finite element formulation, and incorporating a curve fit of actual viscoelastic data, the Golla-Hughes-McTavish (GHM) [1],[2] finite element provides a more accurate model of the response of damped structures. While it alleviates one problem, it creates another: the resulting finite element model is inflated by the addition of extra internal variables (called dissipation coordinates), which are necessary in obtaining damping information but have no physical meaning. In exchange for more accurate damping information, this model requires more computational time.

The problem of reducing a large model while retaining most of the important information contained therein is frequently encountered in control theory. Several methods have been developed to address this problem, most notably the internally balanced reduction method [10]. As before, something is lost for something gained: the retained states of the reduced model often bear almost no resemblance to the states of the original model.   By carrying out one more transformation, it can be shown that the retained states are linear combinations of the original states. Thus, results from the reduced model can be compared directly to the original model. This model order reduction technique will be referred to within this thesis as Yae's method [3], [4].

This thesis will show that by using the GHM viscoelastic finite element in conjunction with Yae's method to model the transverse vibrations of a beam, a finite element model is obtained that is the same size as an elastic finite element model, but computes damping factors and damped impulse response. Further, the natural frequencies and damping coefficients of the viscoelastic finite element model compare well with theory, and the impulse response obtained from the model agrees closely with an empirically determined impulse response.

The following is a brief outline of the rest of this thesis:

Chapter 2 will present the finite element theory and model order reduction theory. Beginning with the Rayleigh-Ritz method, an elastic finite beam element will be derived, followed by a viscoelastic GHM beam element. Yae's method will be explained, and shown how it applies to the GHM model.

In Chapter 3, verification of the viscoelastic finite element code will be shown by reproducing published results. Yae's method of model order reduction will then be applied to the model, and the impulse response of a beam before and after reduction will be compared. Two different sets of published GHM minioscillators will be used in the models.

Two other models that attempted to use data from the viscoelastic material ISD 112 will be discussed, and the reasons for their erroneous results explained, in Chapter 4. An interesting observation regarding the choice of states to be kept based on controllability/observability will also be inspected.

Finally, in Chapter 5, conclusions will be made based on the results, and possible directions for future work will be given.

Chapter 2: Background

The finite element method (FEM) was developed in the 1940's as a way to calculate the stresses in the aluminum skin of aircraft wings. Although derived independently, the FEM was later shown to be a Rayleigh-Ritz type method. With the advent of large memory and high speed digital computers, the FEM has become among the most popular, and powerful, computational methods used in structural dynamics. Rather than trying to solve an equation of motion over the entire physical domain of the problem for the displacement of every single point within the body, the FEM breaks the domain into several smaller subdomains, called elements. The displacement is calculated at discrete points within the element; these points are known as nodes. Using interpolation functions (also called shape functions), the displacement can be approximated across the entire element.

The finite element method will be used to find the mass and stiffness matrices from the differential boundary value problem. These matrices are used to form the equation of motion. The derivation will begin with the Euler-Bernoulli beam equation.

2.1 The Finite Element Method

The initial-boundary value problem for a homogeneous, isotropic, elastic, cantilever Euler-Bernoulli beam with a time varying load *F*(*t*) at the tip is:

$$EI\frac{\partial^4 y(x,t)}{\partial x^4} - m\frac{\partial^2 y(x,t)}{\partial t^2} = 0 \tag{2.1.1}$$

$$y(0,t) = 0, \frac{\partial y(0,t)}{\partial x} = 0, \frac{\partial^2 y(L,t)}{\partial x^2} = 0, \frac{\partial^3 y(L,t)}{\partial x^3} = F(t)/EI$$

$$y(x,0) = 0, \quad \dot{y}(x,0) = 0$$

where *E* is the Young's modulus of the beam, *I* is the moment of inertia, and *y*(*x*,*t*) is the lateral displacement of the neutral axis of the beam. Following the procedure of [5], the differential eigenvalue problem for a beam in free vibration is:

$$EIY''''(x) - \lambda mY(x) = 0 \tag{2.1.2}$$
$$Y(0) = Y'(0) = Y''(L) = Y'''(L) = 0$$

where $\lambda=\omega^2$. (Primes denote total derivatives with respect to x). In the above equation, the boundary conditions at x=0 are homogeneous and essential, while the ones at $x=L$ are natural.

The finite element formulation for the bending vibration of a beam will be developed from the eigenvalue problem of eq. (2.1.2). To begin, define a trial function $\Phi(x)$ that is a real valued function over the domain $0<x<L$. Let $\Phi(x)$ be a smooth function such that $\Phi(0)=\Phi'(0)=0$. Multiply eq. (2.1.2) by the trial function and integrate over the domain.

$$\int_0^L \Phi(EIY'''' - \lambda mY)\ dx = 0 \tag{2.1.3}$$

Integrate the first term of eq. (2.1.3) by parts, keeping in mind the boundary conditions, until Φ and Y are differentiated the same number of times. This yields:

$$\int_0^L \left(\Phi''EIY'' - \Phi\lambda mY\right)\ dx = 0 \tag{2.1.4}$$

Since eq. (2.1.4) contains second order derivatives of Φ and Y, both should have continuous first derivatives across the entire domain, and second derivatives that are discontinuous at a countable number of points in the domain (i.e. the nodes). Following standard finite element modeling (as outlined in [5]), $Y(x)$ is approximated by:

$$Y(x) \approx \phi^T \mathbf{q} \tag{2.1.5}$$

where $\mathbf{q}$ is a vector of generalized elemental nodal displacements such that $\mathbf{q}$=[$Y_1(x)$ $Y_1'(x)$ $Y_2(x)$ $Y_2'(x)$. . .]$^{\mathrm{T}}$; $Y_i(x)$ is the linear displacement at node i=1,2,. . . and Y_i'(x) is the angular displacement at node i. Similarly, the trial function $\Phi(x)$ can be approximated by:

$$\Phi(x) \approx \phi^T \mathbf{a} \tag{2.1.6}$$

where $\mathbf{a}$ is a vector of undetermined coefficients. Substituting eqs. (2.1.5) and (2.1.6) into eq. (2.1.4)

$$\mathbf{a}^{\mathbf{T}}\left[\int_0^L EI\phi''(x)\phi''^T(x)\ dx\right]\mathbf{q} - \mathbf{a}^{\mathbf{T}}\left[\int_0^L \lambda m\phi(x)\phi^T(x)\ dx\right]\mathbf{q} = 0 \tag{2.1.7}$$

However, eq. (2.1.7) should hold for any choice of constants **a**. Keeping this in mind, eq. (2.1.7) in matrix form becomes:

$$K\mathbf{q} = \lambda M\mathbf{q} \tag{2.1.8}$$

where

$$K = EI\int_0^L \phi''(x)\phi''^T(x)\ dx \quad \textit{(stiffness matrix)}$$

$$M = m\int_0^L \phi(x)\ \phi^T(x)\ dx \quad \textit{(mass matrix)}$$

$$\lambda = \text{ approximate eigenvalue}$$

By allowing the global displacement vector **q** to depend on time, and including the forcing function vector **Q**(t), the global finite element equation of motion can be written:

$$M\ddot{\mathbf{q}} + K\mathbf{q} = \mathbf{Q}(t) \tag{2.1.9}$$

A distinction between global and local (or elemental) functions needs to be made. When ϕ is defined over the global domain, it is referred to as a basis function. It is often more convenient to work in local coordinates attached to an element and then transform the results back to the global domain. When basis functions are restricted to an element, they are called shape functions. Note that it is not necessary that shape functions be defined in local coordinates; it merely makes computations easier. Figure 1 shows a typical two node uniform beam element. Nodes are located at *x*=(*j*-1)*h* and *x=jh*, where *j*=1,2,. . . *m* (*m* being the number of global nodes in the model) and *h* is the length of an element. Figure 2 shows the same element in local coordinates.

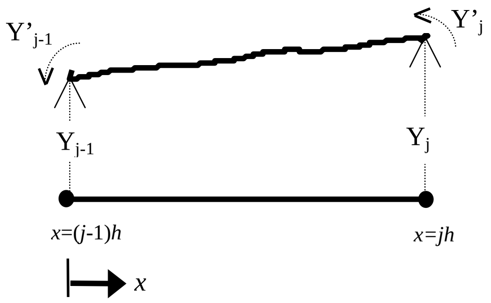


Figure 1. *Beam element in* x *coordinates*

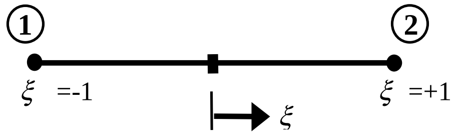


Figure 2. *Beam element in ζ coordinates*

The transformation between global and local coordinates x and ξ, respectively, is [5]

$$\xi = \frac{2x - h(2j-1)}{h}\ ,\ d\xi\ =\ \frac{2}{h}\ dx \qquad (2.1.10)$$

It should be noted that ξ is defined over [-1,1], in general, to make use of Gaussian integration as needed.

Having obtained a transformation between global and local coordinates, expressions for the elemental mass and stiffness matrices in local coordinates (denoted by superscript n) are:

$$K^{(n)} = EI\int_{(j-1)h}^{jh}\phi''(x)\phi''^{T}(x)\ dx = \frac{8EI}{h^3}\int_{-1}^{+1}\left(\frac{d^2\phi(\xi)}{d\xi^2}\frac{d^2\phi^T(\xi)}{d\xi^2}\right)d\xi \qquad (2.1.11)$$

$$M^{(n)} = m\int_{(j-1)h}^{jh}\phi(x)\phi^T(x)dx = \frac{mh}{2}\int_{-1}^{+1}\phi(\xi)\phi^T(\xi)dx \qquad (2.1.12)$$

The shape functions $\phi(\xi)$ are given by:

$$\phi(\xi)\ =\ \begin{Bmatrix}\phi_1(\xi)\\ \phi_2(\xi)\\ \phi_3(\xi)\\ \phi_4(\xi)\end{Bmatrix}\ =\ \begin{Bmatrix}\frac{1}{4}(2\ -\ 3\xi\ +\ \xi^3)\\ \frac{h}{8}(1\ -\ \xi\ -\ \xi^2\ +\ \xi^3)\\ \frac{1}{4}(2\ +\ 3\xi\ -\ \xi^3)\\ \frac{h}{8}(-1\ -\ \xi\ +\ \xi^2\ +\ \xi^3)\end{Bmatrix} \qquad (2.1.13)$$

The functions $\phi(\xi)$ are defined so that ϕ_1 and ϕ_3 (corresponding to nodal displacements Y_1 and Y_2) are equal to one at local nodes 1 and 2, respectively, and are zero at nodes 2 and 1. The functions ϕ_2 and ϕ_4 (corresponding to nodal rotations Y_1' and Y_2') have a slope of one at nodes 1 and 2, and have zero slope at nodes 2 and 1. Moreover, ϕ_1 and ϕ_3 have zero slope at both nodes, and ϕ_2 and ϕ_4 have zero displacement at both nodes. Applying eq. (2.1.13) to eqs. (2.1.11) and (2.1.12) yields the element stiffness and mass matrices for a beam:

$$K^{(n)}\ =\ \frac{EI}{h^3}\begin{bmatrix}12 & 6h & -12 & 6h\\ 6h & 4h^2 & -6h & 2h^2\\ -12 & -6h & 12 & -6h\\ 6h & 2h^2 & -6h & 4h^2\end{bmatrix},\ M^{(n)}\ =\ \frac{mh}{420}\begin{bmatrix}156 & 22h & 54 & -13h\\ 22h & 4h^2 & 13h & -3h^2\\ 54 & 13h & 156 & -22h\\ -13h & -3h^2 & -22h & 4h^2\end{bmatrix} \qquad (2.1.14)$$

Once the element mass and stiffness matrices are computed for each element, they can be assembled into global stiffness and mass matrices [6]. The vector of nodal displacements and rotations, and the natural frequencies of the system, can be determined by substituting eqs. (2.1.14) into eq. (2.1.8) and solving the eigenvalue problem.

2.2 The Golla-Hughes-McTavish (GHM) Viscoelastic Finite Element

The GHM element allows for the accurate modeling of viscoelastic damping within the framework of traditional elastic finite element analysis. The method owes its accuracy to the fact that empirical data for the complex modulus of a given viscoelastic material are curve fitted and incorporated into the finite element formulation. However, this inclusion generates spurious internal variables, or dissipation coordinates, that do not represent physical displacements. In the next section, eliminating the dissipation coordinates while retaining the natural frequency and damping information using Yae's model order reduction method will be discussed. A brief description of a curve fit routine is included in Appendix D.

Golla & Hughes [1], McTavish & Hughes [2], and Slater [7] begin their derivations of the viscoelastic finite element model with an elastic model. In order for Yae's model order reduction method to be successfully employed, though, a small amount of damping must be introduced into the system to assure asymptotic stability as required by the balancing procedure. Proportional damping of the form $D=cK$ will be assumed in the following derivation where c is a constant such that the value of the damping factor ζ shall be about 0.001 for the first mode. This accounts for air and strain rate damping. Equation (2.2.1) below represents the finite element equation of motion for a proportionally damped elastic beam; it was obtained by adding proportional damping to eq. (2.1.9).

$$M\ddot{\mathbf{q}}(t) + D\dot{\mathbf{q}}(t) + K\mathbf{q}(t) = \mathbf{Q}(t) \tag{2.2.1}$$

From eq. (2.1.14) it can be seen that the material modulus can be factored from the stiffness matrix K. Performing the factorization and transforming the resulting equation into the Laplace domain yields:

$$\left\{s^2 M + sD + E\overline{K}\right\}\mathbf{q}(s) = \mathbf{Q}(s) \tag{2.2.2}$$

where $K = E\overline{K}$ (E being Young's modulus from before) and s is the Laplace domain variable. To apply eq. (2.2.2) to a viscoelastic material, simply replace the elastic modulus E with a viscoelastic material modulus.

For a viscoelastic material, the stress-strain relationship includes not only the instantaneous strain, but the strain history as well. The relationship given in [15] is of the form:

$$\sigma(t) = E\varepsilon(t) + \int_0^t E(t - \varsigma)\,\frac{d\varepsilon(\varsigma)}{d\varsigma}\,d\varsigma \qquad (2.2.3)$$

The Laplace transform of eq. (2.2.3) yields:

$$\sigma(s) = \left[E + s\tilde{E}(s)\right]\varepsilon(s) = E[1 + h(s)]\varepsilon(s) \qquad (2.2.4)$$

By making the substitution s=jω (j=$\sqrt{-1}$) in eq. (2.2.4), the frequency response of σ(*s*) can be determined. The term complex modulus arises from this substitution, since the modulus can now be factored into real and imaginary parts, corresponding to the storage modulus and loss modulus of the viscoelastic material, respectively. (Storage and loss moduli will be discussed in Chapter 3). Fundamental to the GHM finite element model is the assumption that the frequency dependent part of the complex modulus, corresponding to the strain history term in the time domain, can be modeled by the function *h*(*s*), and that *h*(*s*) can be represented by the sum of rational polynomials consisting of minioscillator terms [1], [2]:

$$h(s) = \sum_k \frac{\alpha_k\,(s^2 + \beta_k s)}{s^2 + \beta_k s + \delta_k} \qquad (2.2.5)$$

where each term in the above summation is referred to as a minioscillator because it can be represented schematically by adding a fictitious spring-mass-damper system to the structure being modeled. For brevity, only one minioscillator term will be considered in this derivation.

Replacing the elastic modulus *E* in eq. (2.2.2) with the expression for the complex modulus found in eq. (2.2.4), the finite element equation of motion for a viscoelastic beam is obtained:

$$\left\{s^2 M + sD + E[1 + h(s)]\overline{K}\right\}\mathbf{q}(s) = \mathbf{Q}(s) \qquad (2.2.6)$$

Introducing the following dissipation coordinates

$$\hat{\mathbf{z}}(s) = \frac{\delta}{s^2 + \beta\; s + \delta}\mathbf{q}(s) \qquad (2.2.7)$$

into eq. (2.2.6), and transforming the result back to the time domain yields the equation of motion, namely,

$$\left[\begin{array}{c:c} M & 0 \\ \hdashline 0 & \frac{\alpha}{\beta} K \end{array}\right]\left\{\begin{array}{c} \ddot{\mathbf{q}} \\ \hdashline \ddot{\hat{\mathbf{z}}} \end{array}\right\} + \left[\begin{array}{c:c} cK & 0 \\ \hdashline 0 & \frac{\alpha\beta}{\delta} K \end{array}\right]\left\{\begin{array}{c} \dot{\mathbf{q}} \\ \hdashline \dot{\hat{\mathbf{z}}} \end{array}\right\} + \left[\begin{array}{c:c} K(1+\alpha) & -\alpha K \\ \hdashline -\alpha K & \alpha K \end{array}\right]\left\{\begin{array}{c} \mathbf{q} \\ \hdashline \hat{\mathbf{z}} \end{array}\right\} = \left\{\begin{array}{c} \mathbf{Q}(t) \\ \hdashline \mathbf{0} \end{array}\right\} \qquad (2.2.8)$$

In general, the stiffness matrix is positive semidefinite. This implies that rigid body modes (i.e. modes with zero valued eigenvalues) could be admitted into the system. However, as seen in eq. (2.2.8), if K is positive semidefinite, the viscoelastic mass matrix will also be positive semidefinite. In finite element formulations, the mass matrix is positive definite. To ensure that the viscoelastic mass matrix will be positive definite, and in keeping with the assumptions of GHM [1], the zero valued eigenvalues must be removed from K. It is more convenient to factor the material modulus out of K and find the eigenvalues and eigenvectors of $\overline{K}$ where:

$$K = E\overline{K} = E\overline{R}\overline{\Lambda}\overline{R}^T \qquad (2.2.9)$$

In the above equation, $\overline{\Lambda}$ is a diagonal matrix of eigenvalues and $\overline{R}$ is a matrix whose columns are the eigenvectors of $\overline{K}$. Then remove the zero valued eigenvalues from $\overline{\Lambda}$ and the corresponding columns in $\overline{R}$. Factor E back into $\overline{\Lambda}$ to obtain the transformations of eq. (2.2.10).

$$\Lambda = E\overline{\Lambda}, \quad R = \overline{R}\Lambda, \quad \mathbf{z} = \overline{R}^T\hat{\mathbf{z}} \qquad (2.2.10)$$

Applying the transformations to eq. (2.2.8) and premultiplying the bottom row by $\overline{R}^T$ yields the final form of the viscoelastic equation of motion:

$$\left[\begin{array}{c:c} M & 0 \\ \hdashline 0 & \frac{\alpha}{\beta}\Lambda \end{array}\right]\left\{\begin{array}{c} \ddot{\mathbf{q}} \\ \hdashline \ddot{\mathbf{z}} \end{array}\right\} + \left[\begin{array}{c:c} cK & 0 \\ \hdashline 0 & \frac{\alpha\beta}{\delta}\Lambda \end{array}\right]\left\{\begin{array}{c} \dot{\mathbf{q}} \\ \hdashline \dot{\mathbf{z}} \end{array}\right\} + \left[\begin{array}{c:c} K(1+\alpha) & -\alpha\ R \\ \hdashline -\alpha\ R & \alpha\ \Lambda \end{array}\right]\left\{\begin{array}{c} \mathbf{q} \\ \hdashline \mathbf{z} \end{array}\right\} = \left\{\begin{array}{c} \mathbf{Q}(t) \\ \hdashline \mathbf{0} \end{array}\right\} \qquad (2.2.11a)$$

$$M_v\left\{\begin{array}{c} \ddot{\mathbf{q}} \\ \ddot{\mathbf{z}} \end{array}\right\} + D_v\left\{\begin{array}{c} \dot{\mathbf{q}} \\ \dot{\mathbf{z}} \end{array}\right\} + K_v\left\{\begin{array}{c} \mathbf{q} \\ \mathbf{z} \end{array}\right\} = \left\{\begin{array}{c} \mathbf{Q}(t) \\ \mathbf{0} \end{array}\right\} \qquad (2.2.11b)$$

If N minioscillator terms are used in the curve fit, the viscoelastic mass, stiffness, and damping matrices will have the following forms:

$$M_v = \begin{bmatrix} M & 0 & \cdots & 0 \\ 0 & \frac{\alpha_1}{\delta_1}\Lambda & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \frac{\alpha_N}{\delta_N}\Lambda \end{bmatrix},\ D_v = \begin{bmatrix} cK & 0 & \cdots & 0 \\ 0 & \frac{\alpha_1\beta_1}{\delta_1}\Lambda & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \frac{\alpha_N\beta_N}{\delta_N}\Lambda \end{bmatrix},$$

$$K_v = \begin{bmatrix} K(1+\sum_{k=1}^{N}\alpha_k) & -\alpha_1 R & \cdots & -\alpha_N R \\ -\alpha_1 R^T & \alpha_1\Lambda & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ -\alpha_N R^T & \cdots & 0 & \alpha_N\Lambda \end{bmatrix} \tag{2.2.12}$$

The matrices of eqs. (2.2.11) can be repartitioned to facilitate assembly of the global matrices [1]. During assembly, the spatial coordinates coincident to two neighboring elements are overlapped in the global matrix, but no such overlapping occurs for the dissipation coordinates; they remain internal to each element [8]. Note that the size of each element is increased by the size of **z**.

2.3 Yae’s Model Order Reduction Method

Yae’s model order reduction method is based on Moore’s internal balancing method [10]. Moore defines an internally balanced system as one in which the controllability and observability grammians are diagonal and equal. In Yae & Inman [3] and Yae [4], the model order reduction was taken one step further by applying an additional transformation that expresses the reduced model in terms of a subset of the original states.

Eq. (2.3.1) shows the state space representation of the GHM finite element model from the previous section.

$$\frac{d}{dt}\left\{\begin{array}{c}\mathbf{q}\\ \mathbf{z}\\ \hline \dot{\mathbf{q}}\\ \dot{\mathbf{z}}\end{array}\right\}=\left[\begin{array}{c|c}0 & I\\ \hline -M_v^{-1}K_v & -M_v^{-1}D_v\end{array}\right]\left\{\begin{array}{c}\mathbf{q}\\ \mathbf{z}\\ \hline \dot{\mathbf{q}}\\ \dot{\mathbf{z}}\end{array}\right\}+\left[\begin{array}{c}0\\ \hline M_v^{-1}\end{array}\right]\left\{\begin{array}{c}\mathbf{Q}(t)\\ \mathbf{0}\\ \hline \mathbf{0}\\ \mathbf{0}\end{array}\right\}$$

$$\dot{\mathbf{x}}=A\mathbf{x}+B\mathbf{u} \tag{2.3.1}$$

The matrix *A* is called the state matrix, and *B* the input matrix. In keeping with classical control theory, a third matrix *C*, called the output matrix, should be defined to indicate which states are to be measured as output.

$$\mathbf{y}(t)=C\mathbf{x}(t) \tag{2.3.2}$$

Eqs. (2.3.1) and (2.3.2) form the system (*A*, *B*, *C*, **x**).

To find the transformation matrix that will produce an internally balanced system, begin by finding the controllability and observability grammians W_c and W_o, respectively; they are defined as solutions to the Lyapunov stability equation.

$$\begin{aligned}AW_c+W_cA^T&=-BB^T\\ A^TW_o+W_oA&=-C^TC\end{aligned} \tag{2.3.3}$$

Next, take the Cholesky decomposition of W_c.

$$W_c=L_cL_c{}^T \tag{2.3.4}$$

Premultiply W_o by L_c , post multiply by $L_c{}^{\mathrm{T}}$, and solve the eigenvalue problem of eq. (2.3.5).

$$U^T(L_cW_oL_c{}^T)U=\Lambda^2 \tag{2.3.5}$$

The transformation matrix is given by

$$P = L_c U \Lambda^{-1/2} \tag{2.3.6}$$

The internally balanced model $(\hat{A}, \hat{B}, \hat{C}, \hat{\mathbf{x}})$ is given as

$$\begin{aligned} \dot{\hat{\mathbf{x}}} &= \hat{A}\hat{\mathbf{x}} + \hat{B}\mathbf{u} \\ \mathbf{y} &= \hat{C}\hat{\mathbf{x}} \end{aligned} \tag{2.3.7}$$

where

$$\hat{A} = P^{-1}AP,\ \ \hat{B} = P^{-1}B,\ \ \hat{C} = CP,\ \ \hat{\mathbf{x}} = P^{-1}\mathbf{x}$$

Let *W_c(P)*, *W_o(P)* denote the grammians defined in the balanced coordinate system. By the definition of an internally balanced model, $W_c(P)= W_o(P)=$diag $\{\sigma_{1,}\ \sigma_2, \ldots, \sigma_{2N}\}$. The quantities σ_i (i=1,2,. . .,2N; N being the number of degrees of freedom in the finite element model) are a measure of the controllability and observability of state *i*. It is helpful to reorder the states in descending order of controllability and observability, viz. $\sigma_1>\sigma_2>. . .> \sigma_{2N}$. The internally balanced model of eq. (2.3.7) can then be partitioned into retained states and reduced states (see eq. (2.3.8)), $\hat{\mathbf{x}}_r, \hat{\mathbf{x}}_d$ respectively, where $\hat{\mathbf{x}}_r$ denotes the column vector of those states with significant dynamics (large σ), and $\hat{\mathbf{x}}_d$ denotes the vector of those states with small σ. The reduced states will be of relatively low controllability and observability, and so contribute the least to the system dynamics.

$$\begin{Bmatrix} \dot{\hat{\mathbf{x}}}_r \\ \dot{\hat{\mathbf{x}}}_d \end{Bmatrix} = \begin{bmatrix} \hat{A}_{rr} & \hat{A}_{rd} \\ \hat{A}_{dr} & \hat{A}_{dd} \end{bmatrix} \begin{Bmatrix} \hat{\mathbf{x}}_r \\ \hat{\mathbf{x}}_d \end{Bmatrix} + \begin{bmatrix} \hat{B}_r \\ \hat{B}_d \end{bmatrix}\mathbf{u}\,, \quad \mathbf{y} = \begin{bmatrix} \hat{C}_r & \hat{C}_d \end{bmatrix} \begin{Bmatrix} \hat{\mathbf{x}}_r \\ \hat{\mathbf{x}}_d \end{Bmatrix} \tag{2.3.8}$$

The internally balanced reduced order model $(\hat{A}_r, \hat{B}_r, \hat{C}_r, \hat{\mathbf{x}}_r)$ is obtained by statically reducing eq. (2.3.8). Rather than just deleting $\hat{\mathbf{x}}_d$ and taking $\hat{A}_{rr}$ to represent the system, a static-like reduction formula is performed by setting the input **u** in eq. (2.3.8) equal to zero and considering the static equation $\dot{\hat{\mathbf{x}}} = \hat{A}\hat{\mathbf{x}}$. By setting $\dot{\hat{\mathbf{x}}}_d$ equal to zero, solving one of the static set of equations for $\hat{\mathbf{x}}_r$, and plugging back into the remaining equation, a state matrix $\hat{A}_r$ is found that is a function of the submatrices in eq. (2.3.8), namely $\hat{A}_r = \hat{A}_{rr} - \hat{A}_{rd}\hat{A}_{dd}^{-1}\hat{A}_{dr}$ [9]. Static-like reduction removes the undesirable states $\hat{\mathbf{x}}_d$ from the equation but retains their contribution to the dynamics of the system through the newly formed state matrix $\hat{A}_r$.

The main advantage of model reduction is that the reduced model is smaller than the original, which translates into less computational time and power required for the problem. What is not readily apparent is the relation between the coordinates of the reduced model and the original coordinates. The transformation matrix *P* expresses the coordinates of the reduced model as a linear combination of the original coordinates. In theory, the model reduction could result in the position states being eliminated, leaving only the velocity states. Again, referring to the GHM finite element model, it is desirable for the dissipation coordinates to be eliminated from the model, and the retained states to be in the original coordinates. To do this the remaining states must be related back to the original states in order to obtain physically meaningful information about the system's natural frequencies and damping factors.

If in the original model of eq. (2.3.1) there are 2*N* states, and *k* of those states are eliminated by model reduction, then the reduced model $(\hat{A}_r, \hat{B}_r, \hat{C}_r, \hat{\mathbf{x}}_r)$ consists of (2*N*-*k*) states. The *k* states removed can be considered as a set of constraints (i.e. $\hat{\mathbf{x}}_d$ = 0) applied to the original system (*A*, *B*, *C*, **x**). Thus, any (2*N*-*k*) states among the originals, denoted by $\{x_{j_1}, \ldots, x_{j_{(2N-k)}}\}$, can be chosen to represent (*A*, *B*, *C*, **x**). Applying the internally balanced model order reduction method to the (2*N*-*k*) original states will result in the retained states $\{\hat{x}_1, \ldots, \hat{x}_{(2N-k)}\}$. These states are linear combinations of the (2*N*-*k*) original states. Therefore, a matrix P_r that transforms the reduced model $(\hat{A}_r, \hat{B}_r, \hat{C}_r, \hat{\mathbf{x}}_r)$ into the original coordinates can be obtained simply by eliminating the rows and columns of *P* pertaining to the reduced states. The reduced model in original coordinates
$(A_r, B_r, C_\mathrm{r}, \mathbf{x})$ is

$$\dot{\mathbf{x}}_r = A_r \mathbf{x}_r + B_r \mathbf{u}, \ \ \mathbf{y} = C_r \mathbf{x}_r \tag{2.3.9}$$

where

$$A_r = P_r \hat{A}_r P_r^{-1}, \ \ B_r = P_r \hat{B}_r, \ \ C_r = \hat{C}_r P_r^{-1}, \ \ \hat{\mathbf{x}}_r = P_r^{-1} \mathbf{x}_r$$

The final transformation into original coordinates is Yae's method.

In terms of the GHM finite element model, eq. (2.3.9) will yield complex eigenvalues associated with the displacement and velocity states. Neither dissipation states nor their associated eigenvalues will appear in the model since those states will be of low controllability/observability and will be chosen for elimination. The desired complex eigenvalues will contain information on both damped natural frequencies and damping factors.

Chapter 3: Results

3.1 Introduction

By incorporating the Golla-Hughes-McTavish viscoelastic, nonproportional damping model and Yae's reduction method into a cantilever, isotropic, homogeneous, Euler-Bernoulli finite element beam model, a MATLAB code was developed that determines the response of the beam to an impulse forcing function at the tip before and after model reduction. Additionally, the difference between the two responses is also shown. The real and imaginary parts of the complex stiffness, reproduced using the GHM curve fit parameters α, β, and δ, are plotted as well. Finally, the controllability and observability grammians in the internally balanced state—$W_c(P)$ and $W_o(P)$, respectively—are plotted versus the transformed states. It will be shown in this chapter that the code is a satisfactory solution to the problem at hand, namely, the inclusion of viscoelastic effects in a dynamic FEM model using internal variables and model reduction for accurate time response prediction. The accuracy of the code will be shown by reproducing published results (a comparison of which can be seen in Appendix A), as well as a comparison of the predicted impulse response before and after reduction, their difference, and the Bode plots before and after reduction.

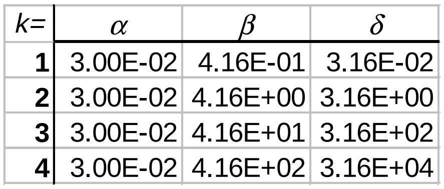
The results of this chapter are based on four cases using two different sets of curve fit data. Two cases used Golla & Hughes' [1] viscoelastic model using four minioscillators, and the other two cases used the three minioscillator model from McTavish, Hughes, Soucy, & Graham [8]. For each data set, two FEM meshes were used: one consisted of four elements, the other of eight elements. The impulse responses of these cases before and after model reduction compared very closely, as did their Bode plots, except for Case 2. Also, two other cases were tried, but did not give good results. These will be discussed in the next chapter. Beam dimensions, material properties, characteristic frequencies, and numerical modal data are listed in Appendix B.

3.2 Test cases

3.2.1 Case 1: 4 Elements, Data Set *ghmdat*, 4 Minioscillators

The first case run was a cantilever beam using the minioscillator terms from [1]; these parameters are listed in the table below.

Table 1: *Minioscillator Terms from* [1]



k=	α	β	δ
1	3.00E-02	4.16E-01	3.16E-02
2	3.00E-02	4.16E+00	3.16E+00
3	3.00E-02	4.16E+01	3.16E+02
4	3.00E-02	4.16E+02	3.16E+04

The objective was to reproduce the modal results in [1] of the four element, nondimensional, cantilever beam using GHM to model the viscoelastic behavior of the beam. A comparison of the numerical data can be seen in Appendix A. Additionally, Yae's method of model reduction was applied, and the response of the beam to an impulse excitation was determined. Comparison of the modal data, and of the impulse responses, before and after model reduction, and comparing both to [1], served as a benchmark to verify that the code gave reasonable results (see Appendix B for numerical results).

Figure 3 shows the real and imaginary parts of the complex modulus versus frequency for the minioscillator terms of Table 1. The real part of the complex modulus corresponds to the storage modulus, or the amount of strain energy stored within the body. The loss factor, η, is the ratio of the imaginary part of the complex modulus to the real part. In physical terms, it is the ratio between the energy dissipated per radian and the peak potential energy [11]. The loss factor for the complex modulus of figure 3 is about $2x10^{-2}$. The damping factor ζ is related to the loss factor η by ζ=η/2; this relation is only valid at resonance. The damping factor is about $1x10^{-2}$ for this complex modulus. The damping factor can also be calculated from the logarithmic decrement of the response [12]. Damping factors obtained from these two methods should compare well, thus providing a simple sanity check on the code.

It should be mentioned that in [1], no reference is made to a real material, or any material at all, for that matter, to which the curve fit is made. As the example problems worked out in the paper are purely numerical, and no reference to either experiments or an actual material is made, the author is under the impression that while the minioscillator terms may be patterned after some unnamed material, the terms themselves are fictitious and were picked for ease of computation. Also, in [1], the GHM viscoelastic model used is purely nondimensional.

The time response of the beam model to an impulse excitation at the tip before and after reduction is shown in Figure 4. The top graph shows the response of the original model, where *yv* is the tip displacement of the original model; the middle graph shows the response of the reduced model—*yr* being the tip displacement of the reduced model; and the bottom graph shows the difference between the two responses. Any difference between *yv* and *yr* appears to be negligible. Also, both impulse responses appear as expected: the oscillations are large initially, but decrease exponentially. The oscillations appear to die out after 0.12 seconds.

The next figure (Figure 5) shows an overlay of the Bode plot of the original model and the reduced model. The two plots agree very well in both gain and phase up until about 10^5 rad/s. The gain plot contains information on both frequency and damping factor of a given mode. The natural frequencies are determined from the location of the peaks, or local maxima, on the *x*-axis. The damping factor is determined by the bandwidth of the peak at the half-power points [6], [12]. The natural frequencies and damping factors for the first seven modes before and after reduction are in agreement because the plots match up when overlaid. This conclusion is further supported by the fact that the phase plots of the original and reduced model line up closely as well. After about 10^5 rad/s, both the gain and phase plots after reduction diverge significantly from the plots of the original model. This is because the curve fit of *h(s)* to the complex modulus data was only performed over the frequency range up to 10^5 rad/s.

The main feature of the internally balanced reduction method [10], upon which Yae's method is based, is that by definition, the controllability and observability grammians in the balanced realization—$W_c(P)$ and $W_o(P)$ respectively—must be equal and diagonal. By sorting these matrices in descending order, a qualitative rank of the degree of controllability/observability is achieved. The least controllable/observable internally balanced states can then be statically reduced out of the model. In its application, it is desirable that the most controllable/observable states contain at least all of the physical states (linear displacement, rotational displacement, linear velocity, and rotational velocity of all the nodes). While the dissipation states may not have physical meaning, they do contribute to the dynamics of the system, as well as to the modal information, because of the coupling of physical displacements and dissipation 'displacements' in the stiffness matrix in eq. (2.2.11a). Static-like reduction removes the dissipation states from the model but still retains most of the information conveyed by those states in the form of the new $\hat{A}_r$ matrix. Figure 6 shows a bar graph of $W_c(P)$, $W_o(P)$ versus the transformed, or internally balanced, states. For a four element, four minioscillator finite element model the first sixteen states will correspond to physical states when they are transformed back into physical coordinates, while the remaining sixty-four states will correspond to dissipation states in the original coordinate system. The bar graph shows that while there are several jumps in controllability/observability, a significant one occurs between the sixteenth and seventeenth states. Since this also corresponds to the threshold between physical and dissipation states in the original coordinate system, it would seem logical—and justifiable by their rankings in $W_c(P)$ and $W_o(P)$—to assign the first sixteen transformed states to the states retained vector $\hat{\mathbf{x}}_r$, and assign the remaining states to the states reduced vector $\hat{\mathbf{x}}_d$.

The first case shows that GHM, coupled with Yae's model order reduction, is a viable method of modeling modal parameters of a viscoelastic material, as well as predicting the response.

3.2.2 Case 2: 8 Elements, Data Set *ghmdat*, 4 Minioscillators

The second case used the same GHM data and beam material properties as the first case, but the finite element model was comprised of eight elements instead of four. Figure 7 shows the impulse responses before and after reduction, and their difference, for the eight element model. As before, the response before and after reduction agrees well. The oscillations appear to die out around 0.12 seconds. The amplitude of oscillation is less in this case than it was for the four element model. The natural frequencies and damping factors before and after reduction agree with the dimensionalized theoretical data from [1]. However, looking at the Bode plot of figure 8 it is apparent that there are major flaws. Both the gain and phase plots line up closely up to the first mode at about $5x10^3$, but after that the plots diverge radically. In fact, while the gain plot before reduction has definite peaks and valleys (or poles and zeros), the gain plot after reduction has virtually no discernible valleys, and just barely peaks at about the same place as the prereduction plot. After the second drop-off in phase, the post reduction phase plot does not correspond to the phase changes in the prereduction plot. The behavior seen in the Bode plots could be attributed to the fact that the majority of the natural frequencies lie outside of the frequency range modeled by the complex modulus (10^1-10^4 rad/s). Further compounding the problem is the fact that the Golla-Hughes data was contrived to make numerical modeling easier, as opposed to being empirically determined. In both this case and the previous one, the nondimensional, nonphysical complex modulus data, which was intended only to calculate modal data and not responses, has been dimensionalized and imbued with physical meaning that it may or may not have. By doubling the mesh size, higher modes are incorporated into the model, and since most of the lower frequency modes are already outside of the range of the GHM model, the addition of more modes outside of this range could account for the fact that the four element model still gives good results, but the eight element model does not.

By doubling the number of elements, the number of physical states doubles as well from sixteen to thirty-two for the eight element model. In figure 9 it is seen that there is a jump in controllability/ observability after the thirty-second state, so this would make a good cut off point between states kept and states deleted. In this code, Yae's method is treated as a black box in that physical and dissipation states are given as input, and only physical states are obtained as output. The input and output states are in the same physical coordinate system. However, within the process, several transformations take place in which the 'physical' and 'dissipation' states become intermingled, only to be sorted out at the end. It is conceivable that the higher frequency modes and the dissipation states interact adversely to produce the results seen in figure 7. There is evidence in support of this hypothesis, but further work will have to be carried out to verify it; the dissipation states, although nonphysical, do affect the dynamics of the system through their coupling with the physical states in the GHM stiffness matrix.

3.2.3 Case 3: 4 Elements, Data Set *mhsgdat2*, 3 Minioscillators

Unlike the previous two cases, the next two cases use the complex modulus for Hysol epoxy TE 6175 resin with HD 3561 hardener [8]. It has a mass density of 1176 kg/m^3 and a zero frequency Young's modulus of 2.8779 GPa. The material is also lightly damped. The real and imaginary parts of its complex modulus versus frequency can be seen in figure 10. The data was empirically collected in the frequency range of 10^1 and 10^3 rad/s. Within this range, the real and imaginary parts are basically constant. The minioscillator terms for this data are listed in Table 3.2.2. The terms listed in [8] are in (α, $2\zeta\omega$, ω^2) parameters but were converted into (α, β, δ) parameters using the formulas listed in Appendix D.

Table 2: *Minioscillator Terms from* [8]

k=	α	β	δ
1	2.30E-02	1.04E+03	1.70E+04
2	8.11E-03	5.64E+03	4.96E+05
3	2.36E-02	3.84E+04	2.31E+07

In Case 3, four elements were used to model a beam composed purely of Hysol and its resin. The impulse responses are given in figure 11; the responses are almost identical. The Bode plots in figure 12 also compare very well up to about 5x10^3 rad/s, after which they diverge and do not appear to correspond to one another. In this case, the point of divergence is well outside of the frequency range modeled, so that could account for the divergence. The plot of the internally balanced controllability and observability grammians versus transformed states is the same for this case as the plot of figure 6. As such, sixteen states were kept in the model order reduction.

Case 4: 8 Elements, Data Set *mhsgdat2*, 3 Minioscillators

The impulse responses and Bode plots of Case 4 (figures 13 & 14) look remarkably similar to the ones of Case 3. The impulse responses of Case 4 appear to have the same amplitude as the responses of Case 3, and the Bode plots diverge at about the same frequency. The only discernible difference is that the Bode plots for Case 4 shows the effects of higher frequency modes which do not appear in Case 3, but this is to be expected since adding more elements incorporates higher frequency modes into the model. Doubling the number of elements does not change the impulse responses or the lower modes of the system. This supports the hypothesis that the problems that arose due to doubling the number of elements for the *ghmdat* data set are probably due to false physical interpretation of the numerical nondimensional data, but further tests need to be conducted to make a conclusive decision.

Thirty-two states were kept in the model reduction, corresponding to the thirty-two physical states in this model. See figure 9 for the controllability/observability grammian plot.

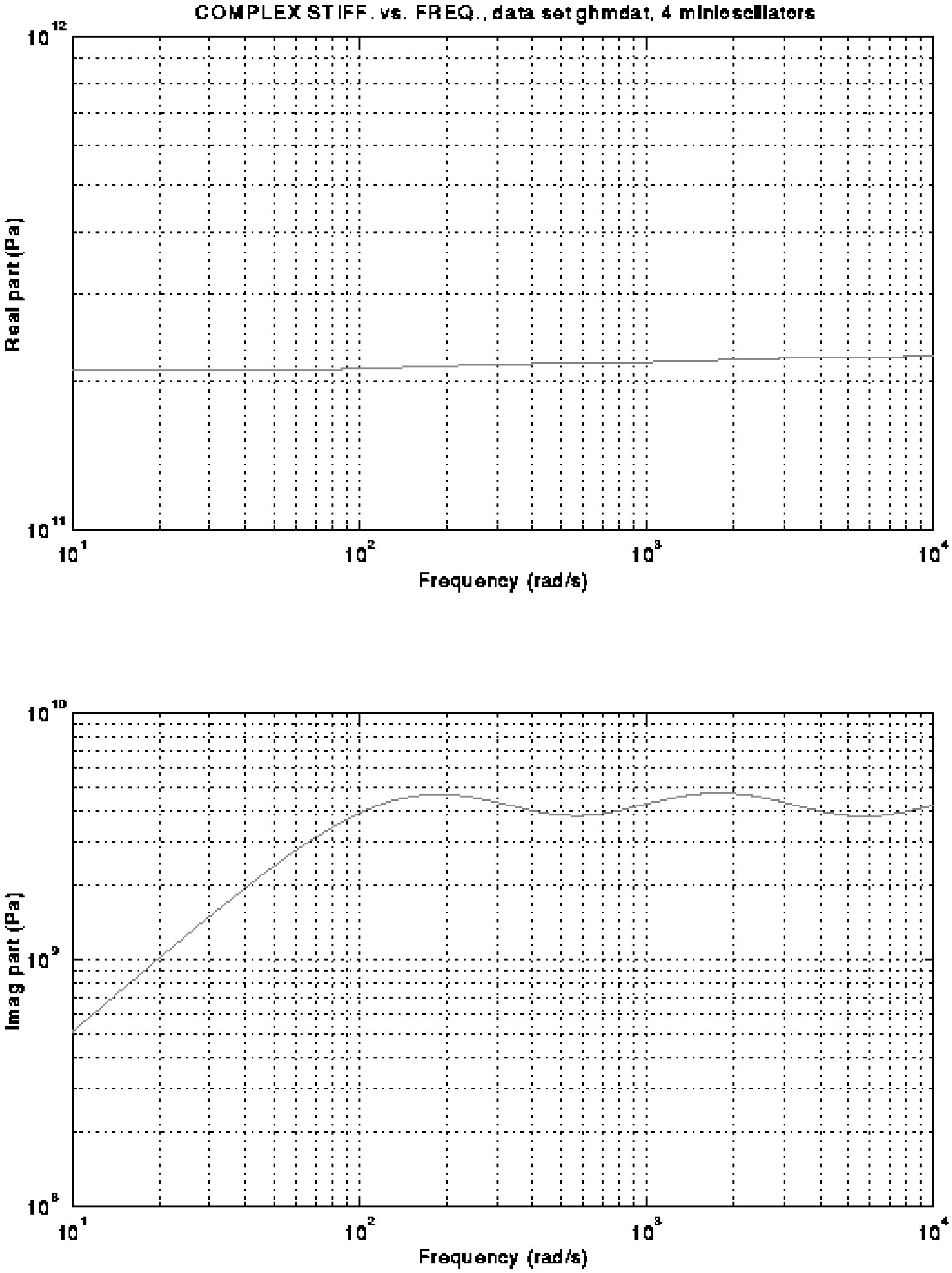


Figure 3. *Real and imaginary parts of complex stiffness vs. frequency, Case 1.*

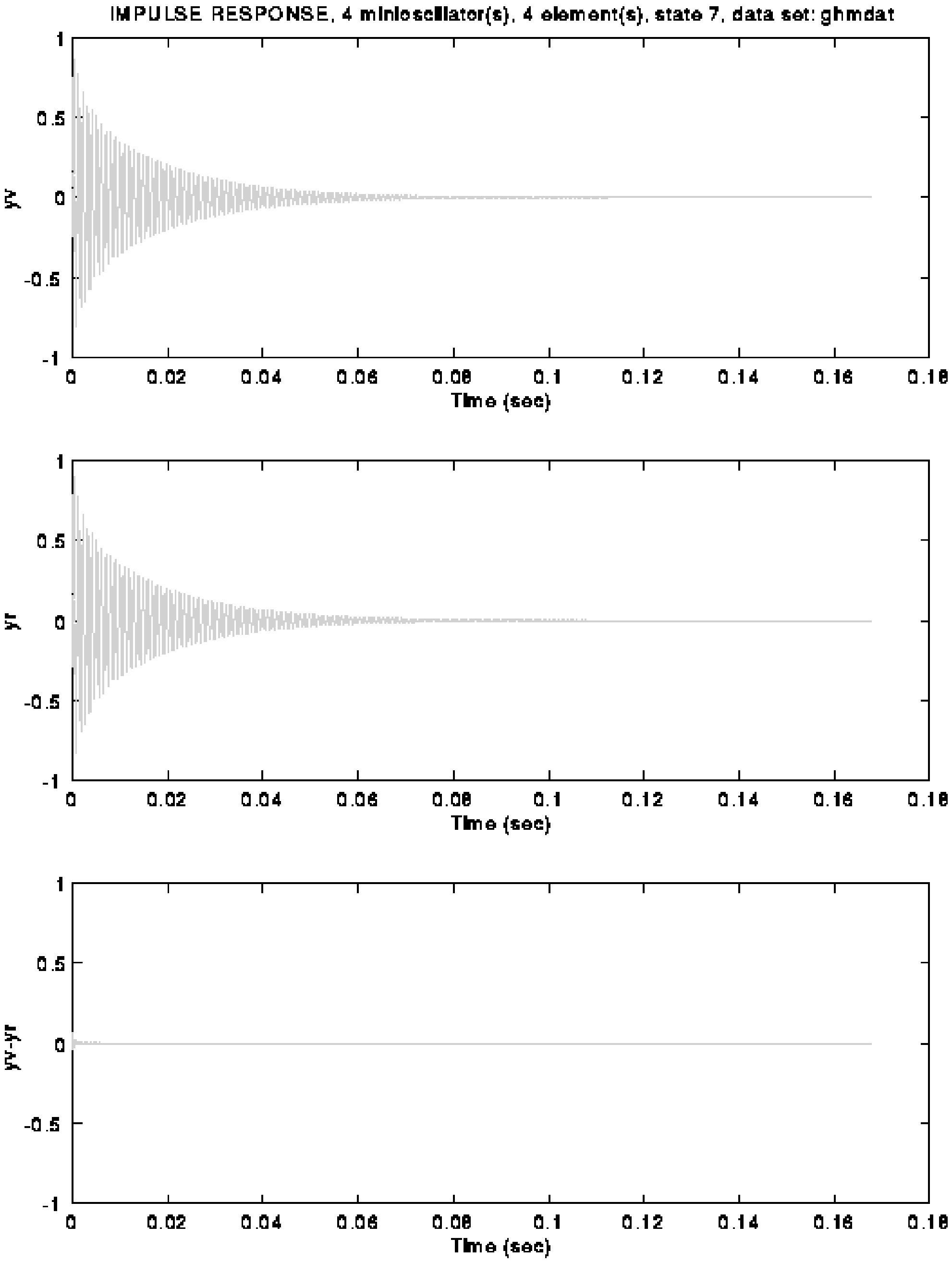


Figure 4. *Impulse responses before and after reduction, and their difference, Case 1.*

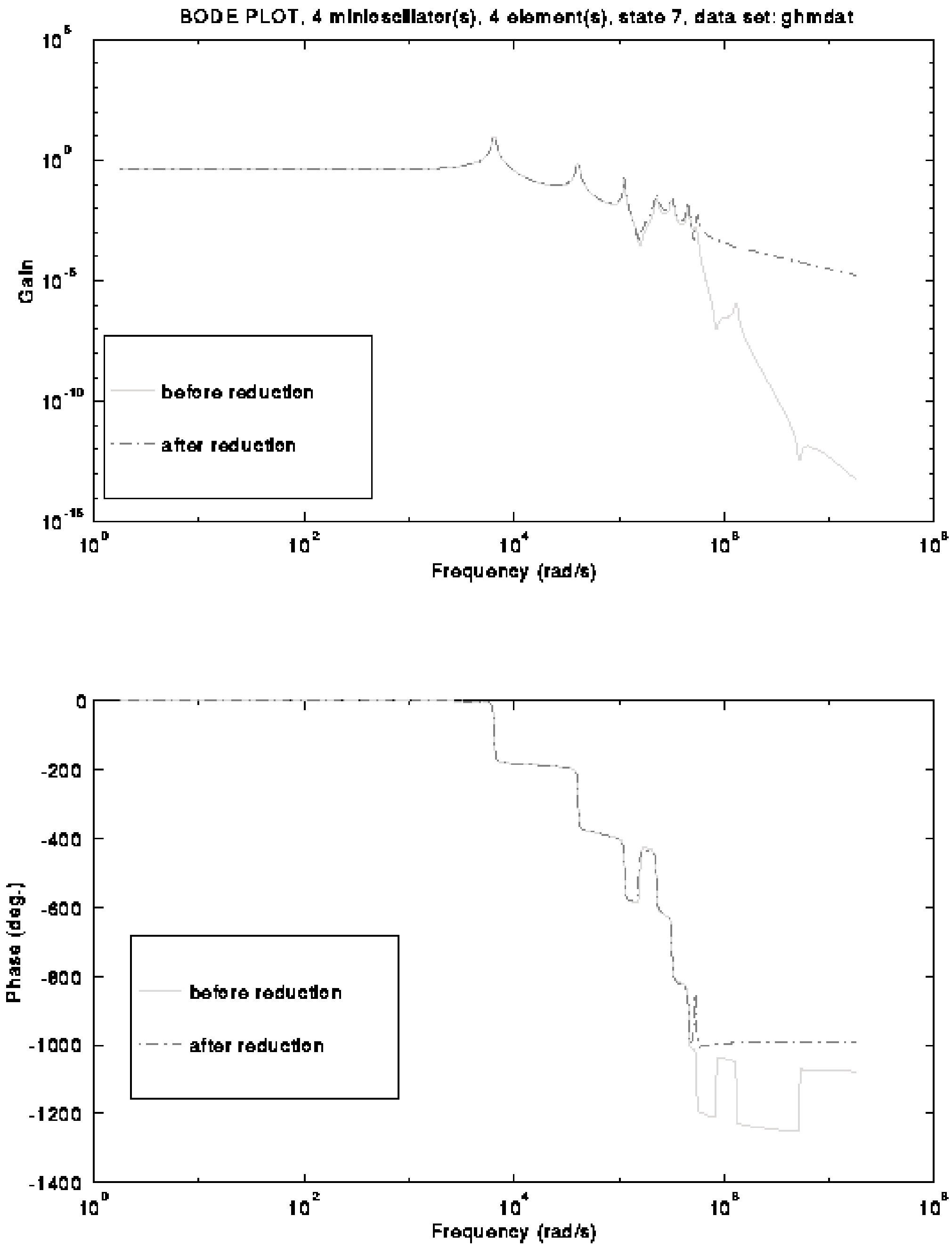


Figure 5. *Overlay of Bode plots before and after reduction, Case 1.*

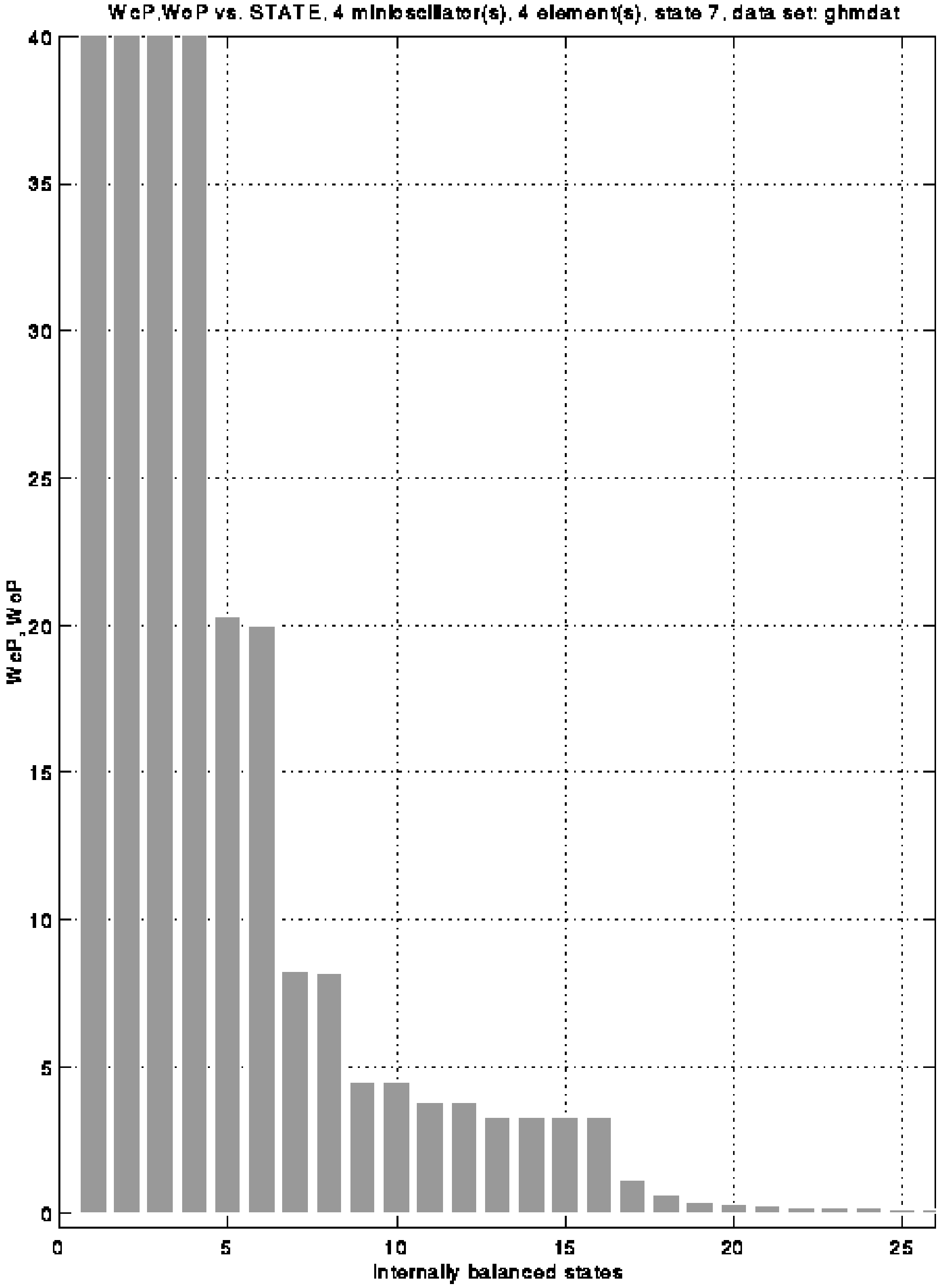


Figure 6. *WcP, WoP vs. Internally balanced states, Case 1.*

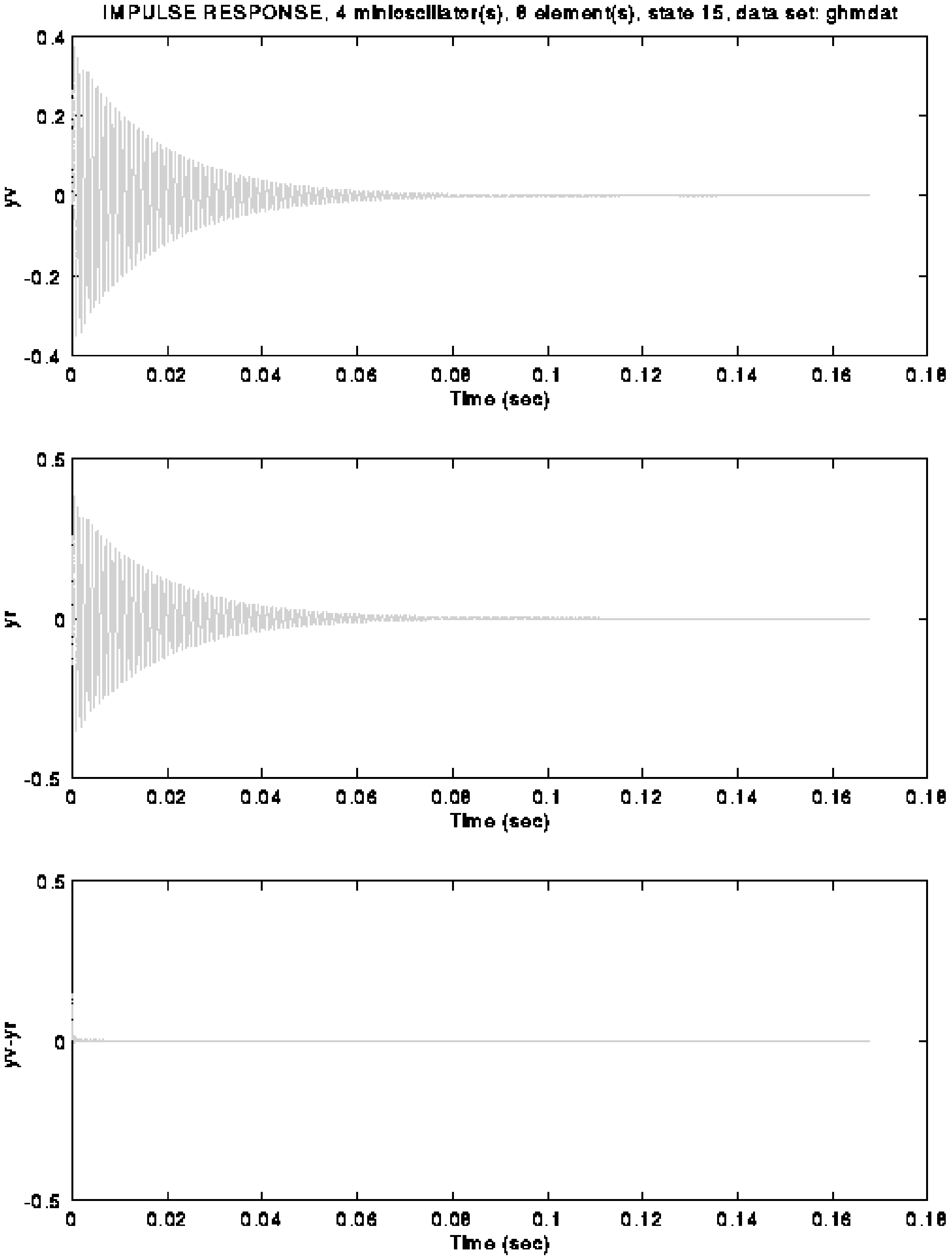


Figure 7. *Impulse response before and after reduction, and their difference, Case 2.*

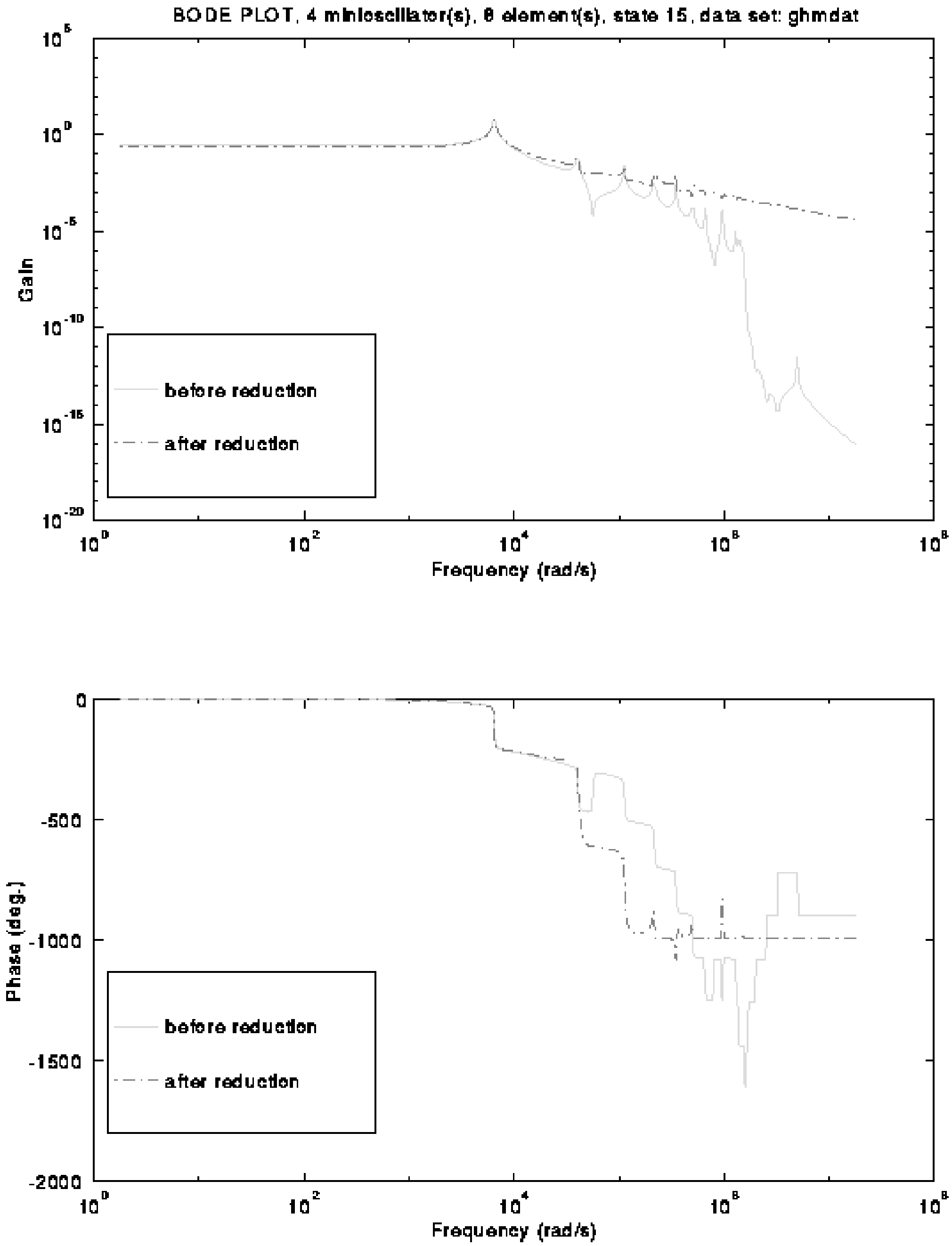


Figure 8. *Overlay of Bode plots before and after reduction, Case 2.*

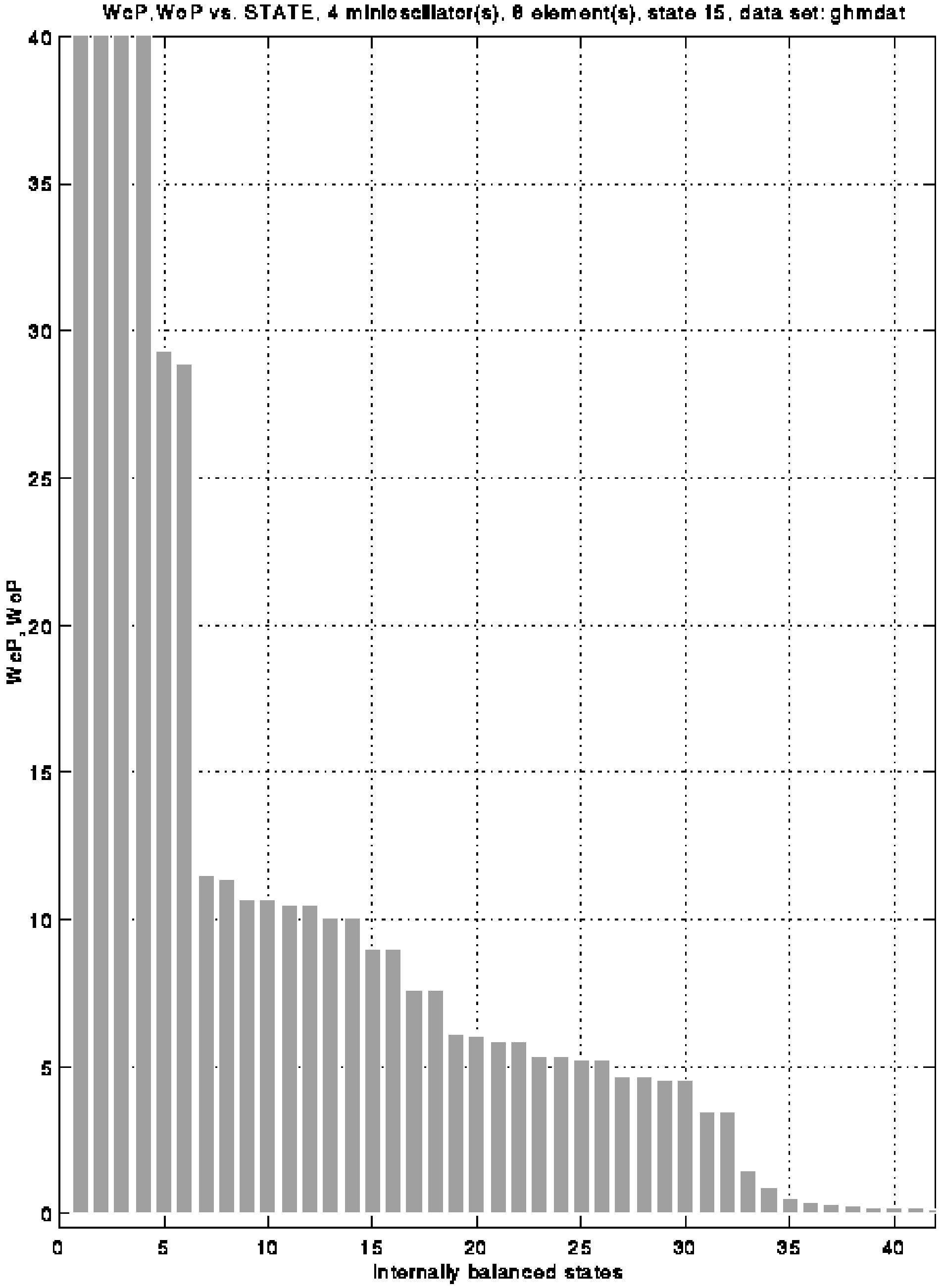


Figure 9. *WcP, WoP vs. Internally balanced states, Case 2.*

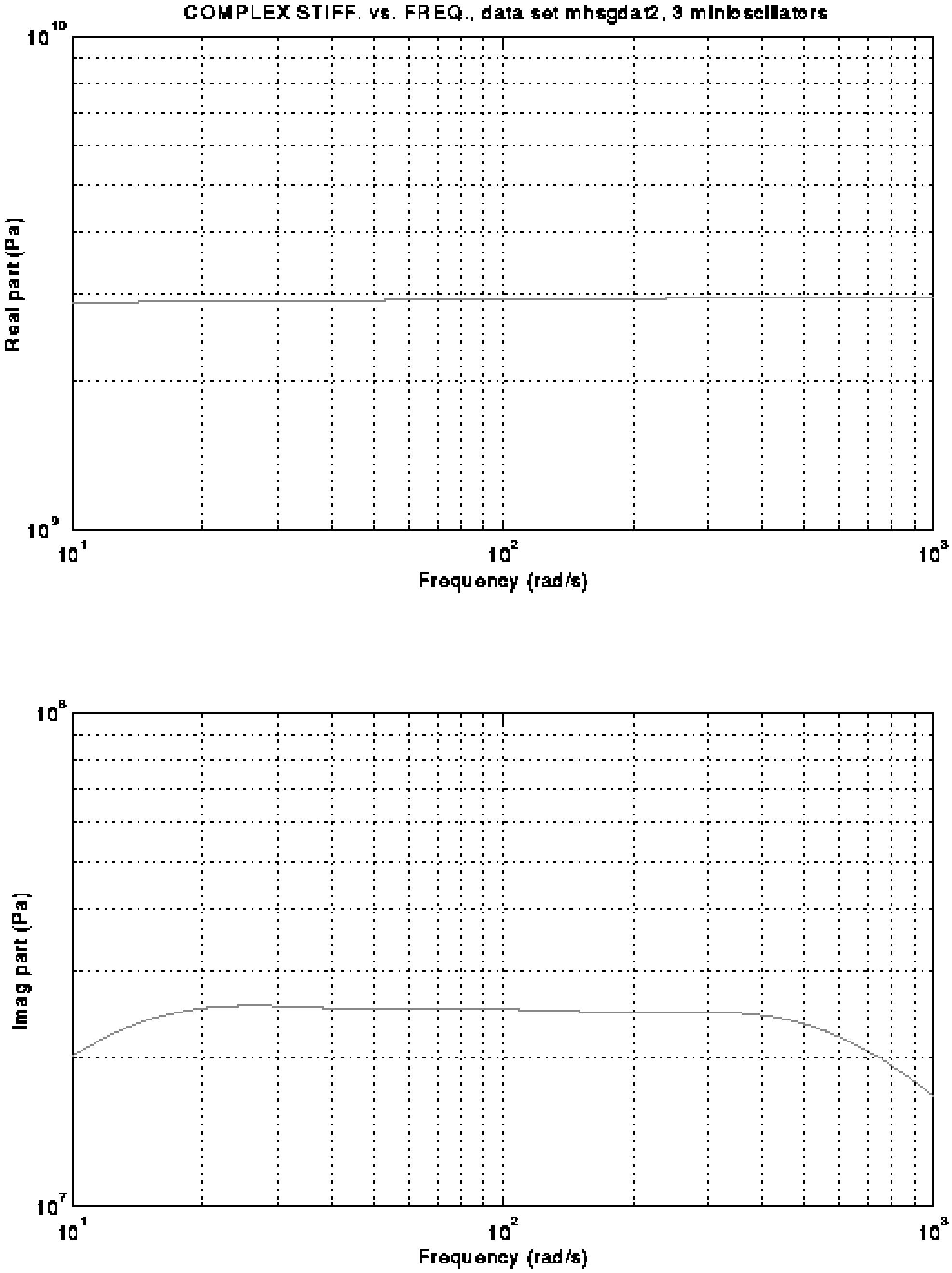


Figure 10. *Real and imaginary parts of complex stiffness vs. frequency, Case 3.*

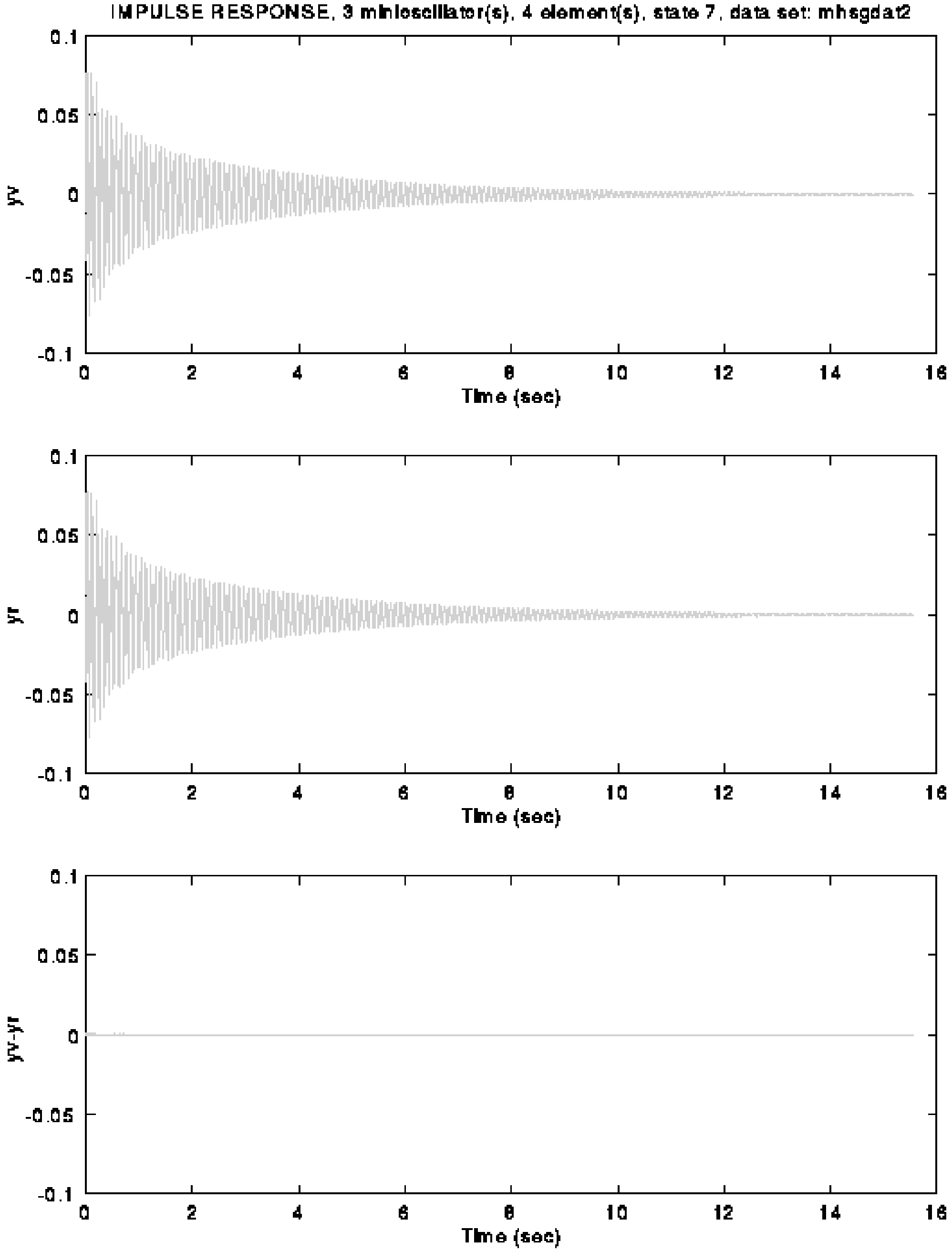


Figure 11. *Impulse response before and after reduction, and their difference, Case 3.*

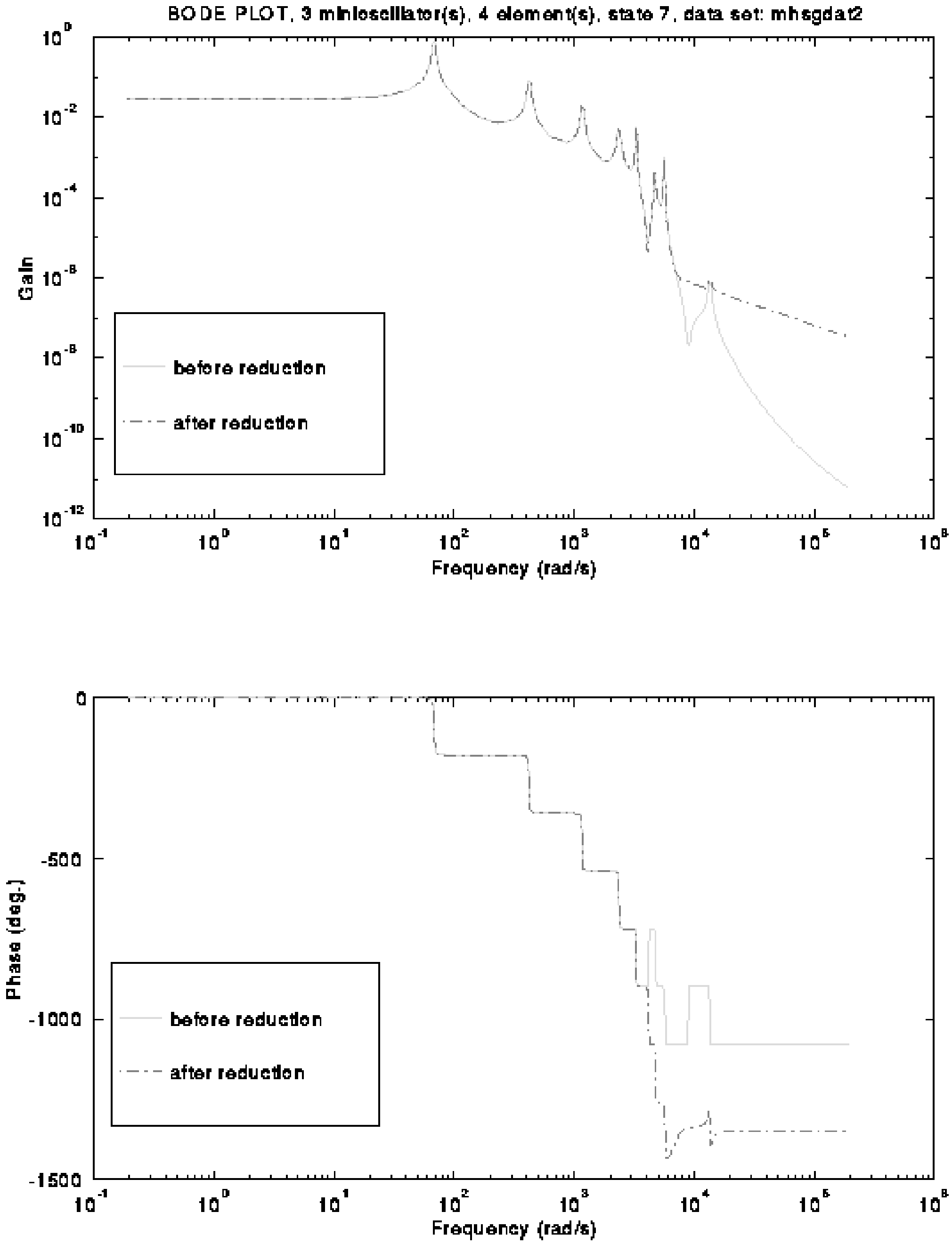


Figure 12. *Overlay of Bode plots before and after reduction, Case 3.*

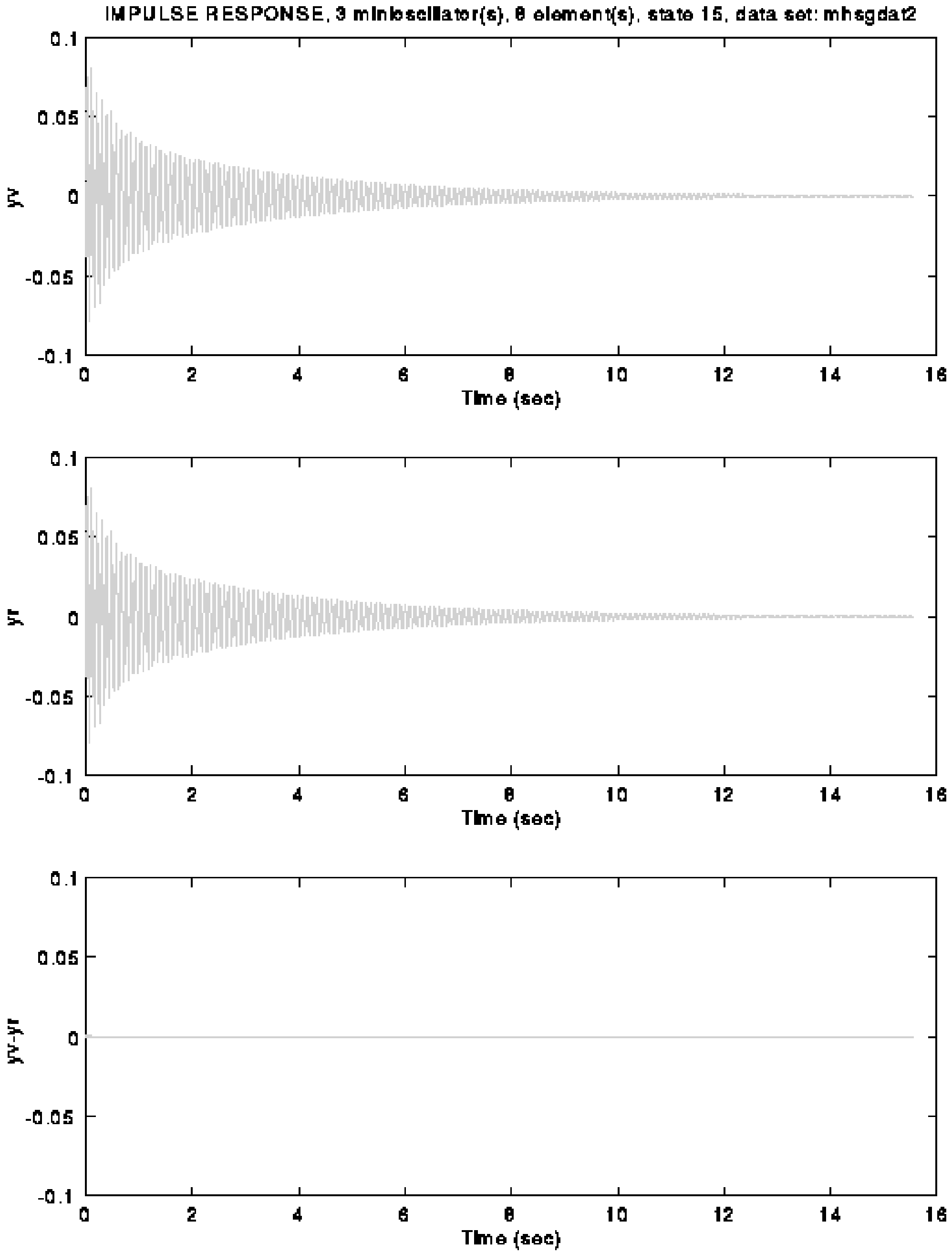


Figure 13. *Impulse responses before and after reduction, and their difference, Case 4.*

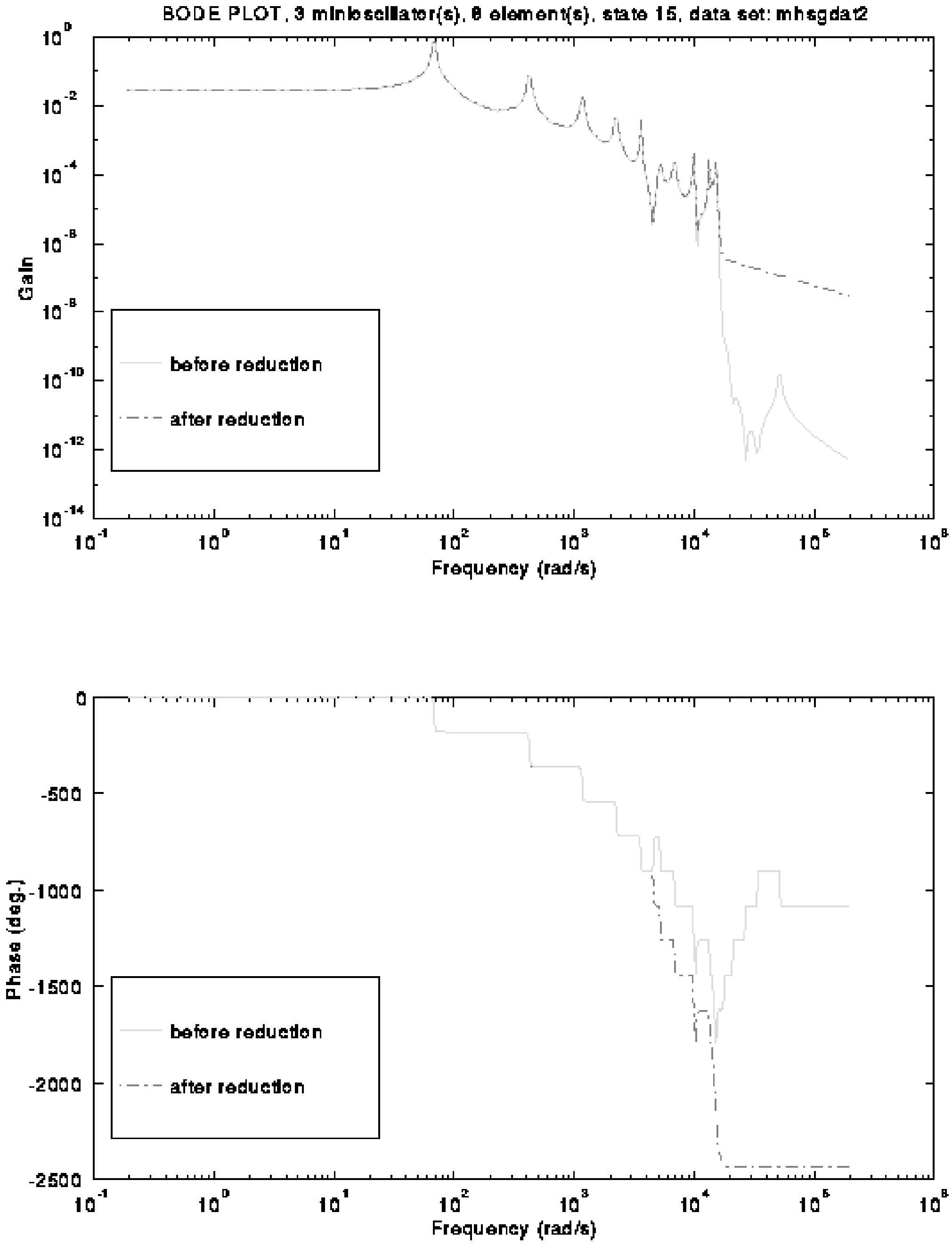


Figure 14. *Overlay of Bode plots before and after reduction, Case 4.*

Chapter 4: Other Cases

The previous chapter listed results for two different data sets: *ghmdat*, taken from [1]; and *mhsgdat2*, taken from [8]. Two other data sets were tried in addition, but did not give satisfactory results. The main cause of trouble was trying to incorporate data meant for a more sophisticated model into the homogeneous Euler-Bernoulli beam model.

GHM begins its derivation with an elastic finite element model, and generalizes to a viscoelastic one. It was thought that the GHM finite element model using Euler-Bernoulli beam theory presented in this thesis could be used as a low order approximation of a sandwich beam consisting of a base of aluminum, 3M's ISD 112 damping material [13] and then another layer of aluminum. To accurately model this beam, shear stresses between the layers need to be taken into account. Shear deformation is the primary source of damping for this beam; as such, the storage modulus for ISD 112 is listed in terms of shear modulus. The model used in this thesis does not take into account the fact that more than one material is present, nor does it account for shear stress within the beam. The finite element model started with an elastic model of an aluminum beam, and then used ISD 112 properties in the GHM formulation. Physically, this model would correspond to a material with the stiffness of aluminum and the damping properties of ISD 112. The problem is that there is a mismatch between the material properties of aluminum and those of ISD 112. The fundamental elastic model of the aluminum beam uses Young's modulus of aluminum, which has an order of magnitude of 10^{11}. The order of magnitude of the zero-frequency (or elastic) shear modulus of ISD 112 is 10^{6}. For a single finite element, the parts derived from the elastic model are roughly five orders of magnitude larger than those derived from GHM. When the elemental mass matrices are assembled into a global mass matrix and then inverted to form the state matrix as required by Yae's method, the matrix becomes ill conditioned and produces erroneous results. The physical representation of this is that two separate, distinct materials are being used in an attempt to model a single material that supposedly has the properties of both, but they are distributed evenly throughout the body rather than located in discrete portions of the body as is the case in a sandwich beam.

The next case was an attempt to reconcile the mismatch between the two materials. The storage modulus of ISD 112 was artificially raised to be on the same order of magnitude as that of aluminum; the real and imaginary parts of the complex modulus were adjusted so that the loss factor would remain the same. This would correspond to a beam made out of a viscoelastic material having about the same stiffness as aluminum. The loss factor of ISD 112 is around 1.0, so this beam would be overdamped. If the loss factor were lowered a couple of orders of magnitude, it would be expected that the beam would be underdamped and exhibit oscillatory behavior due to an impulse excitation, thus serving as a verification of the model. The attempt produced poor results. Although the material properties of ISD 112 were made comparable to those of aluminum, the act of artificially altering the material properties of ISD 112 could have extended the data beyond its range of validity. Also, there was still the problem of trying to couple two properties without taking into consideration the interactions between them through shear deformations.

An interesting problem arose regarding the choice of states to keep. Figure 15 shows the controllability/observability of a four element model. The first sixteen states should be the physical ones, and from what was seen in the cases of the previous chapter, it is expected that there would be a significant jump in controllability/observability between the last state to be kept and the first to be reduced. Upon inspection, though, the least controllable/observable physical state is at least as controllable/observable as the first few most controllable/observable dissipation states. It didn’t matter whether the first several dissipation states were kept or not for the four element model, but when the mesh size was increased to eight elements, the code could not produce a meaningful, albeit erroneous, impulse response if the first several dissipation states were not included in the reduced model. This result suggests that the choice of a cut-off point for the reduced model should reflect the controllability/observability of the states, and that states with controllabilities/observabilities above this cut-off must be kept, even if those states belong to the class of states one desires to eliminate (i.e., dissipation states).

Also of interest is the fact that this case necessitated the static-like reduction of the $(\hat{A}, \hat{B}, \hat{C}, \hat{\mathbf{x}})$ system. When eight elements were used to model the system, and before static-like reduction was applied, the reduced model predicted a constant impulse response of the order of 10^{100}—a number clearly beyond the tolerance of the computer. However, after the static-like reduction was applied, the response at least resembled a typical impulse response, although it was different from the prereduction response. As has been mentioned earlier in this thesis, the static-like reduction retains the dynamic characteristics of the eliminated states. Clearly, in this case, simply eliminating the dynamic contributions of the dissipation states throws away significant dynamic information such that a reasonable (although not necessarily accurate) impulse response is not possible. By performing the static like reduction, as well as keeping those dissipation states that have the same controllability/observability as the physical states with the smallest values of σ_i, a believable impulse response is obtained.

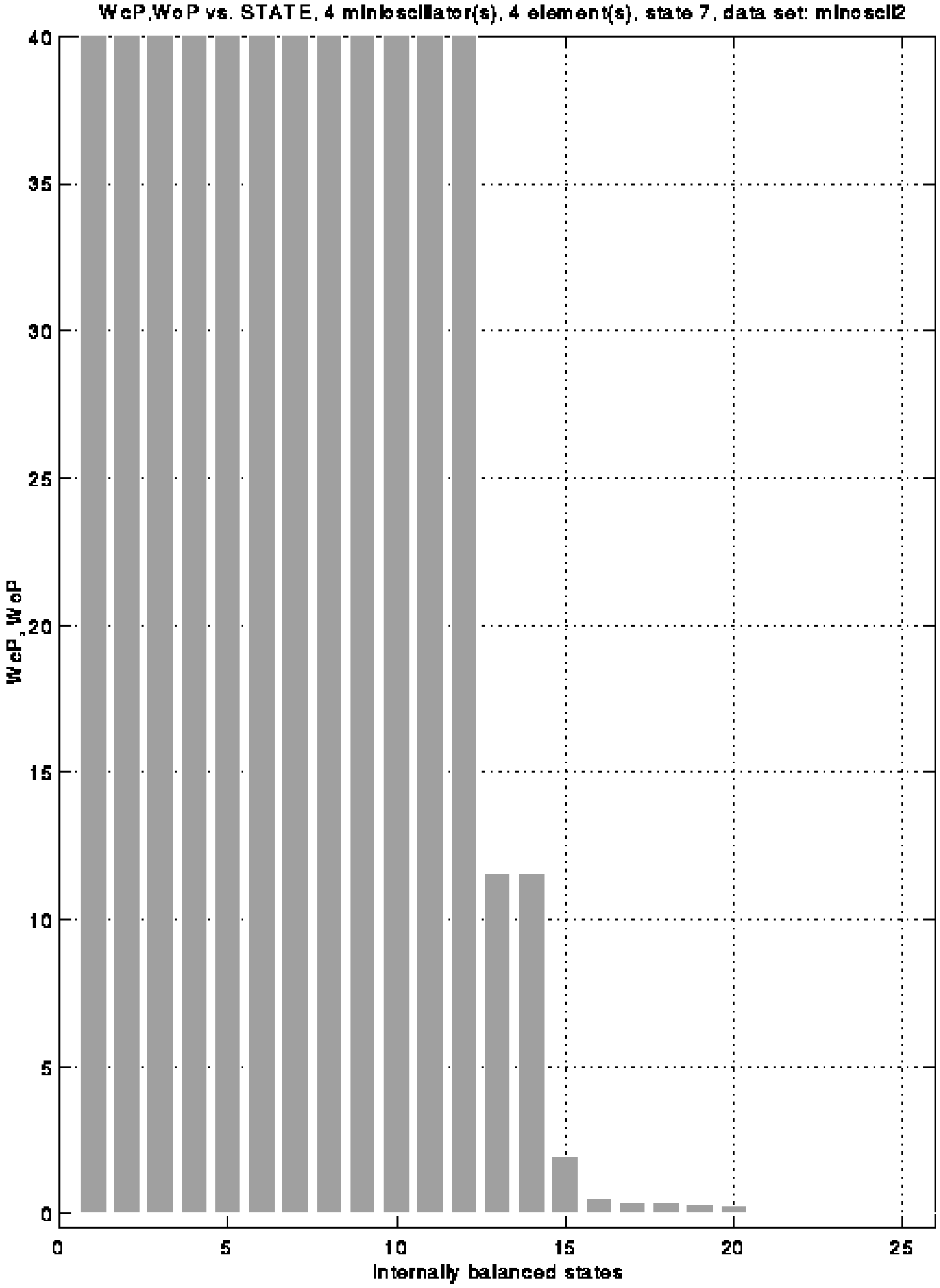


Figure 15. *WcP, WoP vs. internally balanced states for data set*

Conclusions

The following conclusions can be made based on the results:

1. The code incorporating GHM viscoelastic modeling and Yae’s model order reduction method is valid. The code is able to reproduce published results, and the modal data and impulse response before and after model order reduction compare closely. The resulting reduced model accurately accounts for nonproportional damping effects and is of a size similar to the original elastic model.

2. The code is limited in what it is capable of accurately modeling. The code developed here can model a purely viscoelastic beam, but was not able to model sandwich beam behavior. Since GHM and Yae’s method incorporates global mass, stiffness, and damping matrices, it can be extended to a sandwich beam finite element model.

3. The quality of the results is affected by the curve fit of the complex modulus. Natural frequencies outside of the range over which the data is fit produce erroneous results. The modulus of the original elastic model and that associated with the viscoelastic properties evenly distributed throughout the beam must be of similar order of magnitude.

4. Static-like reduction of the dissipation states is preferable to merely deleting them because static reduction retains some of the dynamic characteristics of those states whereas deletion does not.

5. The choice of states to keep must be made based on the controllability/observability of all of the states. Dissipation states having controllability/observability comparable to the least controllable/observable physical states need to be kept in the reduced model because they contribute significantly to the dynamics of the system.

References

1. Golla, D.F., Hughes, P.C. "Dynamics of Viscoelastic Structures—A Time-Domain, Finite Element Formulation." *ASME Journal of Applied Mechanics*, Vol. 52, pp. 897-906, December, 1985.

2. McTavish, D.J., and Hughes, P.C. "Finite Element Modeling of Linear Viscoelastic Structures: The GHM Method." *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 33rd*, Vol. 4, pp. 1753-1763, Dallas, TX, April 13-15, 1992.

3. Yae, K.H., and Inman, D.J. "Control Oriented Reduction of Finite Element Method." *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 115, No. 4, pp. 708-711, December 1993.

4. Yae, K.H. "Reduced Order Modeling and Analytical Model Modification for Structural Dynamics and Control." Ph.D. dissertation, State University of New York at Buffalo, 1987.

5. Batra, R.C. *ESM 5734: An Introduction to the Finite Element Method, Class Notes.* Department of Engineering Science and Mechanics, Virginia Tech, Blacksburg, VA, 1995.

6. Meirovitch, L. *Principles and Techniques of Vibrations*, Prentice Hall, Inc., New Jersey, 1997.

7. Slater, J.C. "Modeling of Non-proportional Frequency Dependent Viscoelastic Damping in Space Structures." Master's thesis, State University of New York at Buffalo, October 1991.

8. McTavish, D.J., Hughes, P.C., Soucy, Y. and Graham, W.B. "Prediction and Measurement of Modal Damping Factors for Viscoelastic Space Structures." *AIAA Journal*, Vol. 30, No. 5, pp. 1392-1399, May 1992.

9. Guyan, R.J. "Reduction of Stiffness and Mass Matrices," *AIAA Journal*, Vol. 3, No. 2, p. 380, 1965.

10. Moore, B.C. "Principal Component Analysis in linear Systems: Controllability, Observability, and Model Reduction." *IEEE Transactions on Automatic Control*, Vol. AC-26, No. 1, pp. 17-32, February, 1981.

11. Sun, C.T. and Lu, Y.P. *Vibration Damping of Structural Elements*, Prentice Hall, Inc., New Jersey, 1995.

12. Meirovitch, L. *Elements of Vibration Analysis*, 2nd ed., McGraw-Hill, Inc., New York, 1986.

13. *3M Product Information and Performance Data: Scotchdamp Vibration Control Systems*, 3M Industrial Tape and Specialties Division, 1993.

14. Lam, M. *Ph.D. dissertation*, Virginia Polytechnic Institute and State University, to be published in 1997.

15. Inman, D.J., "Vibration analysis of Viscoelastic Beams by Separation of Variables and Modal Analysis." *Mechanics Research Communications*, Vol. 16, no. 4, pp. 213-218, March 1989.

Appendix A: Comparison of Viscoelastic FEM Code to Published Results

Below are tables comparing the results of the viscoelastic code, before model reduction, to results published by Golla & Hughes [1]. The model is a clamped-free beam of length 0.3808 m, width 0.0316 m, height 0.0062 m, Young's modulus 2.10x10^{11}, and density 10 kg/m^3. The Golla & Hughes results were originally given in nondimensional form, but were dimensionalized using the beam's characteristic frequency *c*; for this model, *c*=1788.6 rad/s.

Table 3: *Comparison of Natural frequencies (rad/s)*

mode #	Golla-Hughes	FEM	% error
1	6.4927E+03	6.4782E+03	2.2333E-01
2	4.1317E+04	4.0534E+04	1.8951E+00
3	1.1680E+05	1.1248E+05	3.6986E+00
4	2.3252E+05	2.2736E+05	2.2192E+00

Table 4: *Comparison of Damping Factors*

mode #	Golla-Hughes	FEM	% error
1	8.6000E-03	8.5960E-03	4.6512E-02
2	8.2700E-03	8.3205E-03	-6.1064E-01
3	8.4200E-03	8.3702E-03	5.9145E-01
4	7.4600E-03	7.5379E-03	-1.0442E+00

Appendix B: Numerical Modal Data for Cases 1-4

Case 1

4 elements, data set *ghmdat*

Beam dimensions:
L=0.3808 m, W=0.0316, H= 0.0062

Material properties:
E_o=2.1000E+11 Pa, ρ=10 kg/m^3

Characteristic frequency:
c=1788.6 rad/s

Table 5: *Natural frequencies (rad/s), Case 1*

original model	*reduced model*
6.8199E+01	6.8199E+01
4.2379E+02	4.2379E+02
1.1743E+03	1.1743E+03
2.3620E+03	2.3620E+03

Table 6: *Damping factors, Case 1*

original model	*reduced model*
4.3209E-03	4.3372E-03
4.1195E-03	4.1844E-03
2.5038E-03	2.5907E-03
1.3400E-03	1.3937E-03

Case 2

8 elements, data set *ghmdat*

Beam dimensions:
L=0.3808 m, W=0.0316, H= 0.0062

Material properties:
E_o=2.1000E+11 Pa, ρ=10 kg/m^3

Characteristic frequency:
c=1788.6 rad/s

Table 7: *Natural frequencies (rad/s), Case 2*

original model	*reduced model*
6.4853E+03	6.4854E+03
4.0673E+04	4.0673E+04
1.1244E+05	1.1244E+05
2.1635E+05	2.1633E+05
3.4828E+05	3.4821E+05
5.0138E+05	5.0131E+05
6.6238E+05	6.6234E+05
9.4901E+05	9.4900E+05

Table 8: *Damping factors, Case 2*

original model	*reduced model*
8.5969E-03	8.6305E-03
8.3114E-03	8.4872E-03
8.3696E-03	8.6860E-03
7.7350E-03	8.6056E-03
5.2302E-03	5.9706E-03
3.1244E-03	3.4846E-03
1.8670E-03	1.9784E-03
8.3880E-04	8.8364E-04

Case 3

4 elements, data set *mhsgdat*

Beam dimensions:
L=0.3808 m, W=0.0316, H= 0.0062

Material properties:
E_o=2.87789E+09 Pa, ρ=1176 kg/m^3

Characteristic frequency:
c=19.308 rad/s

Table 9: *Natural frequencies (rad/s)*

original model	*reduced model*
6.4782E+03	6.4783E+03
4.0534E+04	4.0533E+04
1.1248E+05	1.1248E+05
2.2736E+05	2.2734E+05

Table 10: *Damping factors*

original model	*reduced model*
8.5960E-03	8.6409E-03
8.3205E-03	8.5357E-03
8.3702E-03	8.7782E-03
7.5379E-03	9.3687E-03

Case 4

8 elements, data set *mhsgdat*

Beam dimensions:
L=0.3808 m, W=0.0316, H= 0.0062

Material properties:
E_o=2.87789E+09 Pa, ρ=1176 kg/m^3

Characteristic frequency:
c=19.308 rad/s

Table 11: *Natural frequencies (rad/s)*

original model	*reduced model*
6.4853E+03	6.4854E+03
4.0673E+04	4.0673E+04
1.1244E+05	1.1244E+05
2.1635E+05	2.1633E+05
3.4828E+05	3.4821E+05
5.0138E+05	5.0131E+05
6.6238E+05	6.6234E+05
9.4901E+05	9.4900E+05

Table 12: *Damping factors*

original model	*reduced model*
8.5969E-03	8.6305E-03
8.3114E-03	8.4872E-03
8.3696E-03	8.6860E-03
7.7350E-03	8.6056E-03
5.2302E-03	5.9706E-03
3.1244E-03	3.4846E-03
1.8670E-03	1.9784E-03
8.3880E-04	8.8364E-04

Appendix C: Finite Element Code

This appendix contains the finite element code *eigtest.m* and its associated functions. Also contained is the mesh generating script file *meshgen.m*. The finite element code applies the procedures outlined in Chapter 2. In brief, it begins with the proportionally damped elastic model of a clamped-free beam, uses them to form the GHM mass, stiffness, and damping matrices and then casts them in state space form, internally balances the state matrix, statically reduces out the dissipation states, employs Yae's method to transform the resulting model back to physical coordinates, and then subjects the model to an impulse excitation.

Files:

eigtest.m Main program

meshgen.m Script file that generates 1-D beam meshes

assmblmat.m Function that assembles the elastic element matrices into a global matrix

vassmblmat2.m Function that assembles the GHM element matrices into a global matrix

loss.m Given the GHM minioscillator terms, this function recreates the associated complex modulus and plots its real and imaginary parts vs. frequency

logdec.m This function calculates the logarithmic decrement of the impulse response

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                   %
%  eigtest.m                                        %
%                                                   %
%  (c)  Jason Salmanoff  4/97                       %
%                                                   %
%                                                   %
%  This program employs the finite element method   %
%  using Hermitian shape functions and the Golla-   %
%  Hughes-McTavish (GHM) element damping model      %
%  to model the damped response of a clamped-free   %
%  viscoelastic, homogeneous, Euler-Benoulli beam.  %
%                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%  Material properties

E0=210e9;  % Pa      (Young’s modulus)
rho=10;    % kg/m^3  (density)
%E0=2.8779e9;
%rho=1176;
%---------------------------------------------------------------

%  Load mesh information

if (1==1)
       input_file=’mesh’
else
       input_file = input(’Input file name? (no extension!!) ’,’s’);
end

eval([’load ’ input_file])


%  Create vector of X coordinates of all the nodes.

X=nloc(:,2);

%  Determine the number of elements and nodes in the mesh

numel=length(con(:,1));
numnode=length(nloc(:,1));

mssg=[num2str(numel),’ element(s) in model’];
disp(’’);
disp(’’);
disp(’’);
disp(mssg);
disp(’’);
disp(’’);
%---------------------------------------------------------------

% Beam geometry
L=(nloc(numnode,2)-nloc(1,2));  % m
W=.0316;  %  m
H=.0062;  %  m

I0=W*H^3/12;     % m^5

% Cross sectional area of beam
Ac=W*H;  % m^2

% Mass of beam
m=rho*Ac*L;
```

```
% Define characteristic frequency
c=(1/L^2)*sqrt(E0*I0/(rho*Ac));

%===============================================================
%===============================================================

%  Predefine the global matrices and load vector
%
%  NOTE:  ’2*numnode’ is used to determine the size of the
%         global matrices because there are 2 degrees of
%         freedom per node.

M=zeros(2*numnode);
K=zeros(2*numnode);

%  Create VISCOELASTIC element matrices using GHM method

%  Select data file for GHM curve fit data
%  (’avosc’=number of available oscillators (i.e. max) )
choice=3;
if(choice==1)
  filename=’minosciii2’;
  avosc=4;
elseif(choice==2)
  filename=’minoscii2’;
  avosc=4;
elseif(choice==3)
  filename=’ghmdat’;
  avosc=4;
elseif(choice==4)
  filename=’origdatii2’;
  avosc=4;
elseif(choice==5)
  filename=’mhsgdat_my_fit_ii2’;
  avosc=3;
elseif(choice==6)
  filename=’mhsgdat_my_fit2’;
  avosc=3;
elseif(choice==7)
  filename=’mhsgdat2’;
  avosc=3;
end

disp(’ ’)
disp(’ ’)
disp(’ ’)
disp([’GHM curve fit data taken from file ’,filename])
disp(’ ’)
disp(’ ’)
disp(’ ’)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                         %
%  curve fit data files:                                  %
%  --------------------                                   %
%                                                         %
%  minosciii2     nondimensionalized data for 3M ISD 112  %
%                 damping material, storage modulus       %
%                 scaled to be of similar magnitude to    %
%                 aluminum, loss factor scaled so that    %
%                 oscillations are possible               %
%                                                         %
%  minoscii2      nondimensionalized data for 3M ISD 112  %
%                 damping material, storage modulus       %
%                 scaled to be of similar magnitude to    %
%                 aluminum; system should not oscillate   %
%                 (loss factor is 10 times less than      %
%                 above)                                    %
%                                                           %
%  ghmdat         curve fit data from Golla-Hughes paper  %
%                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%  The size of the global VISCOELASTIC matrices is determined
%  determined as follows:
%
%  The element matrices are of size 12X12.  For m=number of
%  elements, the size of the VISCOELASTIC global matrix is given
%  by the relation
%
%                       nv=(2*nmosc+4)*m-2*(m-1)
%
%  In this case, m=numel, nmosc=number of minioscillator terms in
%  the GHM curve fit, ’4’ refers to the size of the originial
%  elastic element matrices (4x4).

q=numel;
numosc=input([’How many minioscillator terms? (’,num2str(avosc),’ max.) >> ’]);
nv=(2*numosc+4)*q-2*(q-1);
ne=4*q-2*(q-1);

[Ksw,w,eta_c]=loss(filename,numosc,c,E0);

real_t0=clock;
comp_t0=cputime;

Mv=zeros(nv);
Kv=zeros(nv);
Dv=zeros(nv);

for NL=1:numel    %  loop over number of elements

  %  Redefine X vector for an element
  %  using the ’con’ matrix.

  Xl=X([con(NL,2) con(NL,3)])’;

  %  Calculate element length and Jacobian
  he=Xl(2)-Xl(1);

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  %  5/11/97
  %
  %  NOTE:  To reduce the amount of error in the calculations,
  %  and to make life easier in general, exact expressions
  %  for Mel and Kel for Hermitian cubic beam elements
  %  were hard coded, rather than using Gaussian integration
  %  to derive the elements within the program.  This is just
  %  -ified by the fact that 1D beam elements are pretty easy
  %  to derive by exact integration.
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


  %  Hard code nondimensional beam element from Golla-Hughes

  chi=sqrt(numel);

  Kel=[ 12*chi^6  6*chi^4 -12*chi^6  6*chi^4
         6*chi^4  4*chi^2  -6*chi^4  2*chi^2
       -12*chi^6 -6*chi^4  12*chi^6 -6*chi^4
         6*chi^4  2*chi^2  -6*chi^4  4*chi^2  ];
```

```
   mu=1/chi;

   Mel=(mu^2/420)*[   156    22*mu^2    54    -13*mu^2
                     22*mu^2  4*mu^2  13*mu^2  -3*mu^4
                       54    13*mu^2   156    -22*mu^2
                    -13*mu^2 -3*mu^4 -22*mu^2   4*mu^4 ];

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %
   %  5/20/97
   %
   %  NOTE:  To compare results of this code with the results
   %  in the Golla-Hughes paper "Dynamics of Viscoelastic
   %  Structures--A Time-Domain, Finite Element Formulation,"
   %  compute the eigenvalue problem using the
   %  CONSISTENT MASS MATRIX!!!!!!!!!!!!!!!!!!
   %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


   %  Set value for damping coefficient
   c_d=1.5*.0007;  %  NOTE:  this value will give zeta_elastic=.001


   %  Use Golla-Hughes curve fit data
   [Kvel,Mvel,Dvel]=eigtestel(Kel,Mel,numel,0,numosc,filename);

   %..................................................................


   %  Assemble element ELASTIC matrices into global matrices

   M=assmblmat(M,Mel,con(NL,:),'matrix');
   K=assmblmat(K,Kel,con(NL,:),'matrix');
   C=c_d*K;


   %  Assemble element VISCOELASTIC matrices into global matrices
   Mv=vassmblmat2(Mv,Mvel,NL,numosc);
   Kv=vassmblmat2(Kv,Kvel,NL,numosc);
   Dv=vassmblmat2(Dv,Dvel,NL,numosc);

end         %  end loop over elements


%---------------------------------------------------------------

%  Apply homogeneous essential b.c. for clamped-free beam
K(1:2,:)=[];
K(:,1:2)=[];

M(1:2,:)=[];
M(:,1:2)=[];

C(1:2,:)=[];
C(:,1:2)=[];


Kv(1:2,:)=[];
Kv(:,1:2)=[];

Mv(1:2,:)=[];
Mv(:,1:2)=[];

Dv(1:2,:)=[];
Dv(:,1:2)=[];
```

```
%---------------------------------------------------------------

%  Find global ELASTIC eigenpairs

Ae=inv(M)*K;
[ve,lamE]=eig(Ae);

%  proportional damping matrix

%  Arrange eigenvalues in ascending order, and adjust
%  ve accordingly

[temp,index]=sort(diag(lamE));
lamE=diag(temp);
ve=ve(:,index);

%  Compute natural frequencies: w=sqrt(lambda)
we=sqrt(diag(lamE));

%  Ignore first two eigenvalues
%wea=we(3:ne);
wea=we;

%  Exact values of natural frequencies
fil=sqrt(E0*I0/(m*L^4));
w1=(1.875^2)*fil;
w2=(4.694^2)*fil;
w3=(7.855^2)*fil;
w4=(10.996^2)*fil;
w5=(14.137^2)*fil;

for mm=6:20
  tt(mm-5)=((2*mm-1)*pi/2)^2*fil;
end

wx=[w1 w2 w3 w4 w5 tt]’;

check=[wea(1:numel) wx(1:numel)];

%  Normalize the eigenvectors w.r.t. the mass matrix
for zz=1:length(lamE)
  cr=ve(:,zz)’*M*ve(:,zz);
  ve(:,zz)=ve(:,zz)/sqrt(cr);
end

%  Check to see if damping factor for ELASTIC model with
%  PROPORTIONAL damping for first mode is about .001
idk=diag( (c_d*ve’*K*ve)/(2*wea(1)) );

%  Since if u is an eigenvector, a*u (a=const) is also
%  an eigenvector, the eigenvectors, can be scaled
%  without affected the solution.
%ve=ve/1000;

%---------------------------------------------------------------
%  Construct state matrix from Mv, Kv, and Dv

Z=zeros(nv-2);
I=eye(nv-2);

Av=[    Z       I
     -Mv\Kv  -Mv\Dv ];

Bv=[    Z
     inv(Mv) ];
```

```
%  Reorder Av so that q, q_dot are grouped
%  together and z, z_dot are also grouped
%  together

p=2*numosc;

q_s=zeros(1,2*2*numel);
z_s=zeros(1,p*2*numel);

cca=1:(4+2*numosc-2):2*(nv-1);
ccb=cca-1;


count=1;
for ww=1:2*numel
  f=2*ww-1;
  s=2*ww;
  q_s(f:s)=[(p+1)+ccb(count) (p+2)+ccb(count)];
  z_s(p*ww-(p-1):p*ww)=[cca(count):cca(count)+(p-1)];
  count=count+1;
end

rp=[q_s z_s];

%  State matrix
Av=Av(rp,rp);


%  Input matrix
T=eye(size(Av));
Bv=T(rp,:)*Bv;

%  Output matrix
lala=length(q_s)/2;
Cv=zeros(lala,2*(nv-2));
Cv(1:lala,1:lala)=eye(lala);

%  Find the displacement state at the tip
%  by using length(q_s)/2-1 and store in
%  a vector consisting of all zeros, except
%  one ’1’ at the displacement state of
%  the tip

istate=zeros((nv-2),1);
istate(lala-1)=1;

%  The column of the input matrix Bv
%  corresponding to the tip is given
%  by Bv*istate, and the row
%  of the ouput matrix Cv also
%  corresponding to the tip
%  is given by istate’*Cv

%  Compute VISCOELASTIC eigenvalues
[vv,lamV]=eig(Av);

%  Compute damping ratio and natural frequencies
lv=sort(diag(lamV));

%  Sort VISCOELASTIC eigenvalues
lv=sort(lv);

%  The eigenvalues pertaining to modes of
%  vibration of the beam occur in complex
%  conjugate pairs.  So, sort out the
%  complex eigenvalues and get rid of
%  half of them.
```

```
qq=find(abs(imag(lv))>1.0000);
lva=lv(qq);
pos=[1:2:length(qq)];  % every position eigenvalue
lva=lva(pos);

%  Compute natural frequencies and damping ratios
wv=abs(lva);
zeta=-real(lva)./abs(lva);

%------------------------------------------------
% Yae’s Method
%------------------------------------------------
%  Solve Lyapunov equations for Wo, Wc:
%
%         Av*Wc+Wc*Av’=-Bv*Bv’
%         Av’*Wo+Wo*Av=-Cv’*Cv
%
Wo=lyap(Av’,Cv’*Cv);

Wc=lyap(Av,Bv*Bv’);

%-------------------------------------------

%  Laub’s method
%%%%%%%%%%%%%%%%

%  Find transformation P for balanced state
Lct=chol(Wc);
Lc=Lct’;  % (Lc must be lower triangular)

Lc=fix(Lc*1e4)/1e4;

H2=Lc’*Wo*Lc;

[U,G]=eig(H2);
Gamma=sqrtm(real(G));

P=Lc*U*Gamma^(-.5);

%-------------------------------------------

%  Form state matrices in balanced state
aii=inv(P)*Av*P;
bii=inv(P)*Bv;
cii=Cv*P;


%  Transform Wc, Wo, into balanced state
%  NOTE:  By definition, Wc, Wo must be
%  diagonal and equal in the balanced state
WcP=inv(P)*Wc*(inv(P))’;
WoP=P’*Wo*P;

%  The imaginary parts of WcP, WoP are of order 1e-22
%  (eps=1e-16).  Imaginary parts are negligible, so
%  ignore them.
WcP=real(WcP);
WoP=real(WoP);

%  Check that WcP, WoP are diagonal and equal
test_c=sort([eig(WcP) diag(WcP)]);
test_o=sort([eig(WoP) diag(WoP)]);

test_c=[test_c(:,1) test_c(:,2) test_c(:,1)-test_c(:,2)];
test_o=[test_o(:,1) test_o(:,2) test_o(:,1)-test_o(:,2)];
```

```
WcP=diag(real(WcP));
WoP=diag(real(WoP));

%  Number of most controllable/observable z states to keep
%z_keep=numel+10;
z_keep=1;

%  Perform static reduction on z states
row=size(cii,1);
col=size(bii,2);
elim=length(q_s)+[z_keep:length(z_s)];

[ai,bi,ci,di]=modred(aii,bii,cii,zeros(row,col),elim);

%  Get rid of GHM terms
xt=[1:length(q_s)+z_keep-1];

%  Form transformation matrix Pr from subset of P
Pr=P(xt,xt);

%  Transform balanced state matrices back
%  into physical coordinates
Ar=Pr*ai(xt,xt)*inv(Pr);
Br=Pr*bi(xt,:);
Cr=ci(:,xt)*inv(Pr);

%  Solve eigenvalue problem on reduced state
[vvr,lvrb]=eig(Ar);
lvr=sort(diag(lvrb));

%  Find eigenvalues pertaining to physical data
rr=find(abs(imag(lvr))>1);
lvra=lvr(rr);

%  Since eigenvalues of damped problem occur
%  in complex conjugate pairs, consider only
%  every other eigenvalue
pos=[1:2:length(lvra)];
lvrb=lvra(pos);
wvr=abs(lvrb);
zetar=-real(lvrb)./abs(lvrb);


%-----------------------------------------------------

%  Output

Arr=real(Ar);
Brr=real(Br);

%  Impulse response before reduction
t=[0:.01:300];
[yv,xv,tv]=impulse(Av,Bv*istate,Cv(lala-1,:),0,1,t);

%  Impulse response after reduction, using time vector ’tv’
%  from before to produce response on the same time scale
[yr,xr,tr]=impulse(Arr,Brr*istate,Cr(lala-1,:),0,1,tv);

%  Dimmensional plots
figure(2)
orient tall

subplot(3,1,1),
   plot(tv/c,yv*L)
   xlabel(’Time (sec)’)
   ylabel(’yv’)
   h=axis;
```

```
subplot(3,1,2),
   plot(tr/c,yr*L)
   xlabel('Time (sec)')
   ylabel('yr')
   axis(h);

subplot(3,1,3),
   plot(tv/c,(yv-yr)*L)
   xlabel('Time (sec)')
   ylabel('yv-yr')
   axis([h(1) h(2) -h(4) h(4)]);

txt=[num2str(numosc),' minioscillator(s), ',num2str(numel),' element(s),'...
     ' state ',num2str(find(istate==1)),', data set: ',filename ];

subplot(3,1,1),title(['IMPULSE RESPONSE, ',txt])


%  Bode plot, before reduction
%  NOTE:  [mag,phase,w]=bode(A,B,C,D) returns
%         'w' in Hz, 'phase' in degrees, and
%         'mag' in something other than dB
%
%  Conversions:  w[rad/s]=w[Hz]*2*pi
%                mag[dB]=20*log10(mag)

%  Nondimmensional magnitudes, phases, and frequencies
[mag,phase,w]=bode(Av,Bv*istate,Cv(lala-1,:),0);
[magr,phaser,wr]=bode(Arr,Brr*istate,Cr(lala-1,:),0,1,w);

figure(3)
orient tall
subplot(2,1,1),
   loglog(w*c,mag,'-',wr*c,magr,'-.')
   xlabel('Frequency (rad/s)')
   ylabel('Gain')
   legend('before reduction','after reduction')
   title(['BODE PLOT, ',txt])

subplot(2,1,2)
   semilogx(w*c,phase,'-',wr*c,phaser,'-.')
   xlabel('Frequency (rad/s)')
   ylabel('Phase (deg.)')
   legend('before reduction','after reduction')


figure(4)
orient tall
xstate=1:length(WcP);
[xx,yy]=bar(WcP);
fill(xx,yy,'c')
hold
[xx2,yy2]=bar(WoP);
fill(xx2,yy2,'b')
xlabel('Internally balanced states')
ylabel('WcP, WoP')
title(['WcP,WoP vs. STATE, ',txt])
h=axis;
axis([0 length(q_s)+10 -.5 40])
grid

real_t=etime(clock,real_t0);
comp_t=cputime-comp_t0;

%  Logarithmic decrement of impulse response 'yv'
x1=max(yv);
```

```
nn=find(yv==min(yv));
mm=find(yv(nn:length(yv))==max(yv(nn:length(yv))));
x2=yv(nn+mm-1);
log_dec=log(x1/x2);
```

```
%   meshgen.m
%
%  Generate con, nloc matrices for mesh of n beam
%  elements.

L=.3808;     % global length of beam  x:[0,L]
%L=1;
n=input(’How many elements? >>’);% number of elements

xcoord=0;

nloc=zeros(n+1,2);
con=zeros(n,3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Define ’nloc’ and ’con’ matrices as follows:
%
%      node #   X-coord
%     [                ]
%nloc=[                    ]
%     [                ]
%
%
%     elem. #   n1      n2
%    [                     ]
%con=[                 ]
%    [                     ]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for nnd=1:(n+1)
  nloc(nnd,:)=[nnd xcoord];
  xcoord=xcoord+L/n;
end

for nel=1:n
  con(nel,:)=[nel nel nel+1];
end

%%%%%%%% MESH NAME!!!!
NAME=’mesh’;
eval([’save ’ NAME ’ nloc’ ’ con’])
```

```
function[Kvel,Mvel,Dvel]=eigtestel(Kel,Mel,numel,c,numosc,input_file)

%  This function creates the Golla-Hughes-Mctavish
%  viscoelastic element matrices following the
%  procedures of McTavish-Hughes and Golla-Hughes

%input_file='ghmdat';
eval(['load ',input_file]);

alpha=vemp(1:numosc,1);
beta=vemp(1:numosc,2);
delta=vemp(1:numosc,3);

%  Define chi, mu
chi=sqrt(numel);
mu=1/chi;

%  Define transformation matrix R
a=2*sqrt(3)*chi^3;
b=sqrt(3)*chi;

RT=[ 0  chi  0  -chi
     a   b  -a    b  ];

R=RT';
R1=R(1:2,:);
R2=R(3:4,:);


%  Define nondimensional Kvel, Dvel, Mvel as
%  in Golla-Hughes

K11=Kel(1:2,1:2);
K12=Kel(1:2,3:4);
K22=Kel(3:4,3:4);

M11=Mel(1:2,1:2);
M12=Mel(1:2,3:4);
M22=Mel(3:4,3:4);

phi=1+sum(alpha);

I=eye(2);
Z=zeros(2);

Kvel=zeros(4+2*numosc);
Mvel=zeros(4+2*numosc);
Dvel=zeros(4+2*numosc);

penult=2*numosc+3;  %  penultimate position in row or col
ult=2*numosc+4;     %  ultimate position in row or col

%  Define corners of Kvel, Mvel
Kvel(1:2,1:2)=phi*K11;                %  top left
Kvel(1:2,penult:ult)=phi*K12;         %  top right
Kvel(penult:ult,1:2)=phi*K12';        %  bottom left
Kvel(penult:ult,penult:ult)=phi*K22;  %  bottom right

Mvel(1:2,1:2)=M11;                %  top left
Mvel(1:2,penult:ult)=M12;         %  top right
Mvel(penult:ult,1:2)=M12';        %  bottom left
Mvel(penult:ult,penult:ult)=M22;  %  bottom right

%  5/21/97
%
```

```
%  Add proportional damping from elastic model
%  to GHM model.
%  NOTE:  Proportional damping required for
%  stability in Yae’s model order reduction
%  method.

%if (1==1),
%  c=.0001;  % proportional damping factor

  Dvel(1:2,1:2)=c*K11;                %  top left
  Dvel(1:2,penult:ult)=c*K12;         %  top right
  Dvel(penult:ult,1:2)=c*K12’;        %  bottom left
  Dvel(penult:ult,penult:ult)=c*K22;  %  bottom right
end

I=eye(2);
Z=zeros(2);

count=1;

for pos=3:2:2*numosc+1

   %  Bordering rows and col
   Kvel(1:2,pos:pos+1)=alpha(count)*R1;
   Kvel(pos:pos+1,penult:ult)=alpha(count)*R2’;
   Kvel(penult:ult,pos:pos+1)=alpha(count)*R2;
   Kvel(pos:pos+1,1:2)=alpha(count)*R1’;

   %  Block diagonal
   Kvel(pos:pos+1,pos:pos+1)=alpha(count)*I;


   %  Block diagonal
   Mvel(pos:pos+1,pos:pos+1)=alpha(count)/delta(count)*I;


   %  Block diagonal
   Dvel(pos:pos+1,pos:pos+1)=alpha(count)*beta(count)/delta(count)*I;

   count=count+1;

end
```

```
function [A]=assmblmat(A,B,con,type)

%  This function assembles the global stiffness matrix
%  and the global mass matrix.
%
%  For each node there are 2 degrees of freedom.

%  NOTE: In this subroutine, ’con’ is a vector of nodes belonging to a
%  given element; this vector is taken from one of the rows of the
%  connectivity matrix.

%  First, for an element, find all the 1 dof for each node

a=2*con([2:3])-1;

%  Next, find all the 2 dof for each node

b=2*con([2:3]);

%  Create a position vector such that the entries are arranged
%  as follows:  node1, 1dof; node1, 2dof; node2, 1dof; etc.

pos=zeros(1,4);

pos([1 3])=a;
pos([2 4])=b;

%  Finally, add the B matrix to the appropriate positions
%  in the A matrix

if (type==’matrix’),
       A(pos,pos)=A(pos,pos)+B;
elseif (type==’vector’),
       A(pos,1)=A(pos,1)+B;
end
```

```
function [A]=vassmblmat(A,B,m,nmosc)

%  This function assembles the global VISCOELASTIC
%  stiffness, mass, and damping matrices.
%
%  The size of the element matrices is (2*nmosc+4)X(2*nmosc+4).
%  The size of the global matrices is given
%  by the relation
%
%                  n=(2*nmosc+4)*m-2*(m-1)
%
%  where m=the number of elements.

n=2*nmosc+4;

%  Let ’pos’ be the positions in the global matrix
%  corresponding to the rows and columns of the first
%  element matrix.

pos=[1:n];

%  During the assembly procedure, the last entries in the last
%  two rows and columns of an element matrix and the first two
%  rows and columns of the next element matrix will be added
%  in the global matrix.

vpos=(n-2)*(m-1)+pos;

A(vpos,vpos)=A(vpos,vpos)+B;
```

```
function[Ksw,w,eta_c]=loss(filename,numosc,c,G0)

%  This function plots the loss factor and
%  storage modulus vs. frequency given
%  the (nondimensional) GHM curve fit terms
%  and the characteristic frequency
%  (to redimensionalize the curve fit terms)

eval([’load ’,filename])

a=vemp(1:numosc,1);
B=vemp(1:numosc,2)*c;
d=vemp(1:numosc,3)*c^2;

num=0;
den=0;

d1=1;

upper=ceil(log10(c));
if(upper>3),
  d2=upper;
elseif(upper<3)
  d2=3;
end

w=logspace(d1,d2,200);
w=w;

term=zeros(length(w),4);

for j=1:nmosc
   for k=1:length(w)

%   num=num+...
%   a(j)*((B(j)^2*w(k)^2+w(k)^4-d(j)*w(k)^2)+i*a(j)*B(j)*d(j)*w(k));

%   den=den+...
%   w(k)^4+B(j)^2*w(k)^2+d(j)^2-2*w(k)^2*d(j);

   term(k,j)=a(j)*(-w(k)^2+B(j)*w(k)*i)/(-w(k)^2+B(j)*w(k)*i+d(j));
   end


end

Ksw=G0*(1+term(:,1)+term(:,2)+term(:,3)+term(:,4));

stor=real(Ksw);
%loss=imag(Ksw)./real(Ksw);
loss=imag(Ksw);

figure(1)
orient tall
txt=[’COMPLEX STIFF. vs. FREQ., ’,’data set ’,filename,’, ’,num2str(numosc),’
minioscillators’];

subplot(2,1,1)
   loglog(w,stor)
   grid
   xlabel(’Frequency (rad/s)’)
%   ylabel(’Storage modulus (Pa)’)
   ylabel(’Real part (Pa)’)
    title(txt)
   h=axis;
   axis([10^d1 10^d2 h(3) h(4)])
   title(txt)
```

```
subplot(2,1,2)
   loglog(w,loss)
   grid
   xlabel('Frequency (rad/s)')
%   ylabel('Loss factor')
   ylabel('Imag part (Pa)')
   h2=axis;
   axis([10^d1 10^d2 h2(3) h2(4)])

%  Determine loss factor at characteristic frequency 'c'
realK=interp1(w,real(Ksw),c);
imagK=interp1(w,imag(Ksw),c);

eta_c=imagK/realK;
```

```
function [delta]=logdec(y)

%  This function computes the logarithmic decrement
%  of a response between the first two crests

%  Find the first local maximum
x1=max(y);

%  Find the first local minimum
nn=find(y==min(y));

%  The second local maximum should occur
%  after the first local minimum, so
%  ignoring all data up to the first
%  local min., the next maximum should
%  be the second one
mm=find(y(nn:length(y))==max(y(nn:length(y))));

x2=y(nn+mm-1);

%  Knowing x1, x2, compute the logarithmic decrement
delta=log(x1/x2);
```

Appendix D: Curve Fit Program

This appendix includes the GHM curve fit program *ghmcon.m* and all of its function files. Given an initial guess vector **s0** that contains the zero frequency modulus (here denoted by G_0) and the curve fit parameters α_k, ζ_k, and ω_k (k=1, 2, 3, . . .), this script file finds a constrained minimum of the vector **s0** using the MATLAB function *constr.m*. The finite element code *eigtest.m* expects the curve fit parameters α_k, β_k, and δ_k, where β_k=2 ζ_k ω_k and δ_k= ω_k^2, in nondimensional form. Nondimensionalization of the parameters α_k, β_k, and δ_k is carried out using the characteristic frequency c=($G_0 I/(\rho \ AL))^{0.5}$ (I=moment of inertia, ρ =density, A=cross sectional area, and L=length of beam) as follows:

$$\hat{\alpha}_k \ = \ \alpha_k , \ \ \hat{\beta}_k \ = \ \beta_k / c \ , \ \ \hat{\delta}_k \ = \ \delta_k / c^2$$

where the hatted variables denote nondimensional variables.

All files were written by M. Lam [14].

Files:

ghmcon.m Main program

fit.m Function that applies constraints from *constrai.m* to curve fit parameters and calculates the error *e* between the actual complex modulus *Ks* and the curve fit approximate complex modulus *Kns*

store.m Calculates the complex stiffness using the curve fit parameters

constrai.m Applies constraints on individual curve fit parameters for the function MATLAB function *constr.m*

data1.m Contains data, in vector form, of values of storage modulus and loss factor at discrete frequencies to be curve fit

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%  ghmcon.m  (c) Marca Lam                                                    %
%                                                                             %
%  This program curve fits the storage and loss moduli for 3M Scotchdamp      %
%  SJ 2015 ISD 112 viscoelastic material.                                     %
%                                                                             %
%  subprograms:                                                               %
%                                                                             %
%  constr.m   * MATLAB file from older version                             %
%  constrai.m * function file with constraints for optimization            %
%  data1.m    * data file with GHM data from 3M experimental data           %
%  fit.m             * function file which fits the curve
%
%  store.m    * function file which calculates the new oscillator equations  %
%  lastry.mat        * mat file with last try values
%
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  NOTE:  vemp=[alpha beta delta] in dimensional form.
%  to nondimensionalize using the characteristic frequency c (from eigvalnd.m),
%  alpha=alpha (its already nondimensional!), beta=beta/c, and delta=delta/c^2
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  This program will try to curve fit what and zhat for mini-oscillator.
%  Ahat will be assumed to be less than 10.

clear all;
i      = sqrt(-1);

% input the data points to be curve fit

data1      %  material data
w = ww;    %  vector of frequencies from ’data1.m’
Ks = Ksw;  %  complex modulus from ’data1.m’

%[Ksw,w,E0]=mhsgdat_gen;
%Ks=Ksw’;  % complex modulus from ’mhsgdat_gen’

%  Ask if want new values or from previous run

a = input(’ Do you want to load the values from the previous run? (y/n) ’,’s’);
%a=’n’;

if strcmp(a,’y’),
  load lastry
else
%  Size of s0 controls how many minioscillator terms are used.

real_t0=clock;

%       G0   a1 z1   w1  a2 z2  w2  a3 z3 w3  a4 z4   w4
%  s0 = [E0   .1 10 10000 .1 1 20000 .1 1 5000 .1 .7 7500]; % .1 1 1000 .1 1
1000];

s0=[E0 .013005 130.37 4 .008110 704.40 4 .012552 3805.86 4];


%s0=[E0 .009 150 4 .008 900 4 .009 5000 4 .009 25000 11];
%s0=[E0*1e-2 .0018 500 100e-3 .18 14000  900e-1 .00018 800000 200e-4 .018 40000
20e2]*1e2;
%s0=[E0 1 90000 50 1.3 140000 50 18 80000 5];
%s0=[E0 20 550000 50];

if(1==0)
```

```
%s0=[E0 .013005 130.37 4 .008110 704.40 4 .012552 3805.86 4];
[e,g,Kns]=fit(s0,w,1,Ks);
subplot(2,1,1)
  loglog(w,real(Ks),'b',w,real(Kns),'--')
  legend('real(Ks)','real(Kns)')
subplot(2,1,2)
  loglog(w,imag(Ks),'b',w,imag(Kns),'--')
  legend('imag(Ks)','imag(Kns)')
stop
end


%  To use more minioscillator terms, add sets of three numbers to
%  s0 above to account for ak,zk,wk k=1,2,. . .,n
%  (ak,zk,wk)=(.1,1,1000) is a good guess
%.1 .5 500 .1 3 300 .1 1 100 .1 3 300 .1 3 300 .1 3 300];

% .1 .5 500];% .1 3 300];% .1 1 100];

  %  Make sure that s0(1) is a good guess.  It should be a little under the
  %  first value.  The program has much more difficulty converging if it is
  %  overestimated.  The other numbers are arbitrary.  s0(1) is the guess, G*,
  %  for the shear modulus G(s) = G*(1+...).
end

for j = 1:length(w)
etact(j) = imag(Ks(j))/real(Ks(j));     %  Compute the actual eta
end

n = (length(s0)-1)/3;                   %  Find out how many oscillators incl.

vlb = zeros(1,length(s0));       %  Values for upper and lower bound
                                        %    when using constr optimization
%vlb([2 5 8]); % 11]);% 14 17])=1e-3*[1 1 1 1];% 1 1];
vlb([2 5 8])=1e-3*[5 5 5];
vlb([4 7 10])=1e-1*[4 4 4];


for j = 1:n
vlb(3+3*(j-1))=1;
end

vub = [];                        %  Values for the upper bound
%vub=1e10*ones(1,length(s0));
%vub([4 7 10 13])=[4 4 4 4];


options = [0;1e-10;1e-10];       %  Set options for optimization routine
options(7) = 0;
options(14) = 100;  % Total number of iterations taken in each optimization
                    % routine.  It's better to run more loops on k than
                    % to crank up this number.  If the program keeps
                    % returning close to singular message, then make this
                    % loop even smaller to reset the optimization more
                    % frequently.  This often solves the problem.

for k = 1:100      %  Ask if the graph is good enough.  Can rerun
k                %  optimization as many times as necessary times
[e,g,Kns]=fit(s0,w,n,Ks);
%[e,Kns]=fit(s0,w,n,Ks);

  [s,opts]=constr('fit',s0,options,vlb,vub,[],w,n,Ks);
  s0 = s;

  Knew = store(s0,n,w);                 %  Calculate new stiffness matrix

  err = (Ks-Knew)./Ks;                  %  Error function
  e = err*err';

  if e<8e-5;                    %  Condition to meet in optimization
%  if e<.01;
   break
  end

end


for j = 1:length(w)             %  Calculate new eta
  eta(j) = imag(Knew(j))/real(Knew(j));
end


figure(3)                       %  Plot some figures of data and GHM
subplot(211)                    %  curve fit
  plot(w,real(Ks),'*',w,real(Knew),'g')
  xlabel('Frequency (rad/s)')
%  ylabel('Storage Modulus (Pa)')
  ylabel('Real part (Pa)')
%  title(['Data file ',datafile])
subplot(212)
%  plot(w,imag(Ks)./real(Ks),'*',w,imag(Knew)./real(Knew),'g')
  plot(w,imag(Ks),'*',w,imag(Knew),'g')
  xlabel('Frequency (rad/s)')
  %ylabel('Loss Factor')
  ylabel('Imag part (Pa)')

figure(4)
subplot(211)
  loglog(w,real(Ks),'*',w,real(Knew),'g')
  xlabel('Frequency (rad/s)')
%  ylabel('Storage Modulus (Pa)')
  ylabel('Real part (Pa)')
subplot(212)
%  loglog(w,imag(Ks)./real(Ks),'*',w,imag(Knew)./real(Ks),'g')
  loglog(w,imag(Ks),'*',w,imag(Knew),'g')
  xlabel('Frequency (rad/s)')
%  ylabel('Loss Factor')
  ylabel('Imag part (Pa)')

%  Make sure that met conditions, otherwise must rerun program.

e
b='';
b=input('Is the curve fit good enough? (y/n) ', 's')
%b='y';
if (b == 'y')
  save GHMconst w s0

  %  Partition s0 into [G0 a1 z1 w1 a2 z2 w2 . . .]
  ll=length(s0);

  G0=s0(1);

  ak=s0([2:3:ll])';
  zk=s0([3:3:ll])';
  wk=s0([4:3:ll])';

  nmosc=length(ak);

  vemp=[ak 2*zk.*wk wk.^2];

  save  mhsgdat_my_fit_iv G0 vemp nmosc
else
  disp(' The program must be rerun in order to optimize more.')
  save lastry s0
end


real_t=etime(clock,real_t0);
```

```
function [e,g,Kns]=fit(s0,w,n,Ks)

%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        %
%  fit.m  (c) Marca Lam  %
%                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  This function is used in the constrained optimization function
%  ’constr.m’

i = sqrt(-1);

Kns = store(s0,n,w);             %  Calculate GHM stiffness for each
                           %     optimization cycle
err = (Ks-Kns)./Ks;        %  Error function vector

e = err*err’;                   %  Scalar of error

g = constrai(s0,n);        %  Constraints
```

```
function K = store(s0,n,w)

%  This function gives the K for a given s0

%Kns = ones(1,length(w));
term=zeros(length(w),n);

K=0;
%Kns=zeros(1,length(w));
Kns=0;
for j = 1:n
  for k = 1:length(w)
%    Kns(j) = Kns(j) + ...
%           s0(2+3*(k-1))*(-w(j)^2+2*s0(3+3*(k-1))*s0(4+3*(k-1))*w(j)*i)/...
%           (-w(j)^2+2*s0(3+3*(k-1))*s0(4+3*(k-1))*w(j)*i+s0(4+3*(k-1))^2);
  a=s0(2+3*(j-1));
  W=s0(3+3*(j-1));
  z=s0(4+3*(j-1));

%  term(k,j)=a*(-w(k)^2+2*z*W*w(k)*i)/(-w(k)^2+2*z*W*w(k)*i+W^2);
  num=a*(-w(k)^2+2*z*W*w(k)*i);
  den=(-w(k)^2+2*z*W*w(k)*i+W^2);
  term(k,j)=num/den;
  end
end

for kk=1:n
  Kns=Kns+term(:,kk);
end

[row,col]=size(Kns);
Kns=reshape(Kns,col,row);
K = s0(1)*(1+Kns);
```

```
function [g] = constrai(s0,n)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                             %
%  constrai.m  (c) Marca Lam  %
%                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Function to find the constraints on ahat’s or what’s

%  I have constrained the ahat’s here to be less that 500.  This is
%  totally arbitrary, but will give more accuracy if allowed to go where
%  it wants to.

for j=1:n
%  g(j) = s0(2+3*(j-1)) - 500;
%  g(j) = 100000 - s0(4+3*(j-1));
   g(j)=s0(4+3*(j-1))-10;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          %
%  data1.m  (c) Marca Lam  %
%                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Input file for VEM Scotchdamp SJ 2015 ISD 112

%  ww=vector of frequencies in Hertz
%  G= vector of storage modulus (Young’s modulus)
%  et=vector of loss factors (eta)

E0=210e9;
G0=5e4;
fudge=E0/G0;

if(1==0)
ww  = 2*pi*[10 20 30 40 50 60 70 80 90 100 200 300 415 500 600 700 800 900 1000
2000 3000 4150];
G  = [3.3 4.7 5.7 6.4 7.1 8 9 9.1 10 11.2 13.7 16.1 22 22.5 24.0 26.7 31.4 32.0
32.8 45.2 51.8 62]*1e5;
et = [.97 1 1.01 1.03 1.06 1.06 1.06 1.06 1.06 1.05 1 .96 .93 .924 .918 .91 .9
.88 .83 .73 .7 .62];

else
ww  = [1 2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80 90 100 200 300 415 500 600 700
800 900 1000 2000 3000 4150];
G  = [1.32 1.71 1.82 2.2 2.34 2.5 2.6 3 3.16 3.3 4.7 5.7 6.4 7.1 8 9 9.1 10 11.2
13.7 16.1 22 22.5 24.0 26.7 31.4 32.0 32.8 45.2 51.8 62]*1e5;
et = [.7 .8 .83 .89 .9 .92 .93 .94 .96 .97 1 1.01 1.03 1.06 1.06 1.06 1.06 1.06
1.05 1 .96 .93 .924 .918 .91 .9 .88 .83 .73 .7 .62];
end

G=G*fudge;

vemse=1e-10;
et=et*vemse;

%  Calculation of complex modulus Ksw
for j = 1:length(ww)
%   et(j) = et(j) + 0.1;
   Ksw(j) = G(j).*(1+et(j)*i);
end
```

Vita

Jason Salmanoff

Jason was born in Skokie, Illinios. He completed his undergraduate studies in Mechanical Engineering at Michigan State University, E. Lansing, MI, in May 1995, including a semester (Spring 1994) studying at the Rheinisch Westphaelische Technische Hochsule in Aachen, Germany. He was appointed as a summer researcher to Argonne National Laboratory in Argonne, IL, for Summer 1995. After that he began his graduate studies at Virginia Polytechnic Institute and State University in Blacksburg, Virginia and earned his Master of Science in Engineering Science and Mechanics in September, 1997. Afterwards, Jason began working at CSA Engineering, Inc., in Palo Alto, CA.